UNIVERSITÄT SIEGEN

Reliable, Energy-Efficient and Temporally Predictable Time-Triggered Hybrid TSN Systems Combing Wireless and Wirebound Networks

DISSERTATION

zur Erlangung des Grades eines Doktors

der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Haytham Baniabdelghany

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät

der Universität Siegen

Siegen 2023

Betreuer und erster Gutachter

Prof. Dr.-Ing. Roman Obermaisser

Universität Siegen

Zweiter Gutachter

Prof. Dr. Wilfried Elmenreich

Alpen-Adria-Universität

Klagenfurt

Tag der mündlichen Prüfung
31. Januar 2023

Printed and published with the support of the German Academic Exchange Service

Acknowledgements

I would like to express my special appreciation and thanks to my advisor Professor Dr.Roman Obermaisser, who has been a tremendous mentor for me. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. His guidance on both research as well as on my career has been priceless. I would like also to thank Dr.Ala Khalifeh for his support during my research period.

Special thanks to my parents and my family for their endless love and support throughout my life. To all my friends and colleagues thank you for your motivation and encouragement.

Zusammenfassung

Hybride Netze kombinieren kabelgebundene und drahtlose Technologien und werden gegenwärtig in großem Umfang in Automatisierungssystemen, intelligenten Städten und Home-Area-Networks eingesetzt. Einerseits bieten kabelgebundene Netze aufgrund der Stabilität der Topologie und einer geringeren Wahrscheinlichkeit von Paketverlusten eine höhere Zuverlässigkeit und Leistung. Andererseits bieten drahtlose Netze im Vergleich zu kabelgebundenen Infrastrukturen eine größere Flexibilität und sind weniger kostspielig. In jüngster Zeit haben viele Systeme hybride Netze für Anwendungen mit Echtzeit- und Sicherheitsanforderungen vorgeschlagen. Die Arbeitsgruppe für TSN (Time-Sensitive Networking) hat eine Reihe von IEEE 802.1-Unterprotokollen für drahtgebundene Ethernet-Netze eingeführt um eine fehlertolerante und deterministische Kommunikationsinfrastruktur anzubieten. Die Abbildung der Konzepte von TSN auf drahtlose Umgebungen erfordert jedoch die Berücksichtigung der dynamischen Topologie, Signalstörungen, Zuverlässigkeit und des Datenübertragungsverhaltens. Deshalb erweiterte diese Arbeit die TSN-Fähigkeiten auf hybride (drahtlose und drahtgebundene) Systeme. Das hybride TSN-fähige Netz kann auf verschiedene Architekturen angewendet und modifiziert werden, um weitere von der TSN-Arbeitsgruppe eingeführte Unterstandards zu integrieren. Dadurch können verschiedene TSN-Merkmale wie Taktsynchronisierung, zeitabhängige Formgebung und fehlertolerante Nachrichtenübermittlung über Knoten in einem hybriden System implementiert werden.

Diese Arbeit befasst sich mit Echtzeitsystemen, bei denen verteilte Rechen- und Kommunikationsaktivitäten über eine globale Zeitbasis koordiniert werden. Daher wird in dieser Arbeit eine Erweiterung des Standard-Zeitprotokolls (d.h. IEEE 802.1AS) für hybride TSN-Systeme vorgeschlagen um die Güte der Uhrensynchronisation zu verbessern. Das erweiterte Protokoll berücksichtigt die deterministischen Verzögerungen und Taktdrift, so dass die Genauigkeit der Taktsynchronisation des Standards 802.1AS erheblich verbessert wird. Außerdem wird das Problem der asymmetrischen Verzögerungen bei der Übertragung der Zeitpakete berücksichtigt, um dynamischen Kommunikationseinstellungen mit mobilen Knoten Rechnung zu tragen. Zur Überwachung des Verkehrsverhaltens von Zeitpaketen und der Ausreißerwerte, die aufgrund dynamischer und asymmetrischer Ereignisse auftreten können, wird ein Path Deviation Delay (PDD)-Filter bereitgestellt. Die Simulationsergebnisse zeigen, dass das erweiterte Protokoll die Synchronisationsgenauigkeit im Vergleich zum 802.1AS-Standard auf unter eine Mikrosekunde verbessert. Im Gegensatz dazu steigt der Synchronisationsfehler mit dem Standardprotokoll stark an, wenn asymmetrische Verzögerungsverhältnisse und Szenarien mit mobilen Knoten vorhanden sind.

Wenn alle Knoten in einem hybriden TSN-Netz mit einer globalen Zeitbasis synchronisiert sind, müssen die Aufgaben einer Echtzeitanwendung an drahtlose Hosts geplant und ihre übertragenen Nachrichten so geplant werden, dass alle Echtzeitanforderungen erfüllt werden. Das Problem der Task- und Nachrichtenplanung ist besonders in drahtlosen Systemen eine Herausforderung, da die räumliche und zeitliche Verteilung der abhängigen Tasks an die drahtlosen Hosts die Nebenbedingungen erfüllen und den Energieverbrauch minimieren muss. Die gegenseitige Störung zwischen den Signalen muss abhängig von den Nachrichtenübertragungszeiten und der räumlichen Nähe ebenfalls berücksichtigt werden. Interferenzen können den Empfang verhindern, Signale verfälschen und die Signalqualität beeinträchtigen. Daher konzentriert sich diese Arbeit auf die Planung von Tasks und Nachrichten in drahtlosen Echtzeitsystemen.

Die meisten der bisherigen Planungsalgorithmen für drahtlose Netze berücksichtigen entweder die Perspektive der Einsparung von Netzressourcen, Einschränkungen bei der Weiterleitung von Nachrichten oder die Auswirkungen von Störungen. Um neben der

Taskplanung auch die Interferenzen und Routing-Beschränkungen für die Nachrichtenplanung zu berücksichtigen, wurde ein neuer Algorithmus zur Task- und Nachrichtenplanung namens Task and Message Scheduling for Wireless Networks (TMS) vorgeschlagen. In diesem Algorithmus werden für die zeitgesteuerten Nachrichten der Berechnungsaufgaben die optimalen Routen mit minimaler Interferenz verwendet. Darüber hinaus wird jede Nachricht fragmentiert und mehreren Zeitschlitzen zugeordnet. In jedem Zeitschlitz können Nachrichten gleichzeitig übertragen werden, indem ein physikalisches Interferenzmodell verwendet wird. TMS unterstützt die Multi-Cast-Kommunikation unter Berücksichtigung der Präzedenzbedingungen zwischen Berechnungsaufgaben mit Zeitbeschränkungen. Zur Bewertung des vorgeschlagenen Algorithmus werden mehrere Algorithmen mit unterschiedlichen Routing-Strategien implementiert. Die experimentellen Ergebnisse zeigen, dass TMS in den Simulationstests besser abschneidet als Lösungen aus verwandten Arbeiten. Dies betrifft die Echtzeitfähigkeit, die verbrauchte Energie, Fehlerraten und Ausfälle. Außerdem wird in den Experimenten festgestellt, dass die Änderung der Parameter des verwendeten Interferenzmodells die Leistung der verglichenen Algorithmen beeinflusst.

Die nahtlose Wiederherstellung von korrektem Verhalten bei Fehlern ist ein weiterer kritischer Punkt für Echtzeit- und zeitempfindliche Systeme. Daher werden in dieser Arbeit Algorithmen auf der Grundlage des TMS vorgestellt, welche eine Nachrichtenreplikation über redundante Routen unterstützen. Es werden zwei zuverlässige Algorithmen vorgestellt, welche als Reliable Task and Message Scheduling for Wireless Networks (R-TMS) und Optimized Reliable Task and Message Scheduling for wireless networks (OR-TMS) bezeichnet werden. OR-TMS verwendet einen bioinspirierten Optimierungsalgorithmus um bessere Lösungen zu finden. Die vorgeschlagenen zuverlässigen Algorithmen verwenden einen Ansatz namens Frame Replication and Elimination for Reliability (FRER) um die Kommunikationsnachrichten über redundante disjunkte Routen zu replizieren.

Bei R-TMS wird die Systemzuverlässigkeit durch die Planung von Tasks in drahtlosen Hosts verbessert. Dabei werden die Ankunftszeit der Nachrichten, der Energieverbrauch und die Ausfallraten optimiert. Es wird auch ein Zuverlässigkeitsmodell eingeführt, um die Zuverlässigkeit des Systems zu bestimmen. Das Zuverlässigkeitsmodell berechnet die Zuverlässigkeit jedes Tasks in Abhängigkeit von der Zuverlässigkeit aller eingehenden Nachrichten. Die Zuverlässigkeit eines Tasks, der keine Nachrichten weiterleitet, stellt die globale Zuverlässigkeit des Gesamtsystems dar. R-TMS wird mit Algorithmen aus verwandten Arbeiten verglichen, welche die kürzesten oder lastabhängigen Routen zum Senden von Nachrichten verwenden oder die Aufgaben auf Hosts verteilen, welche die geringste Bereitschaftszeit haben. Die experimentellen Ergebnisse zeigen, dass die Zuverlässigkeit des von R-TMS berechneten Systems im Vergleich zu anderen Algorithmen verbessert wird und gleichzeitig die Skalierbarkeit des Netzentwurfs und die Aktualität gewährleistet ist. Die Auswirkungen von Verbindungsfehlern auf die Nachrichtenübermittlungsrate werden ebenfalls untersucht.

Der OR-TMS-Algorithmus basiert auf der diskreten Partikelschwarmoptimierung (DPSO), wobei eine Nutzenfunktion zur Optimierung des Task- und Nachrichtenplanungsproblems aufgestellt wird. Es wird iterativ versucht eine bessere Lösung zu finden. Die Nutzenfunktion unterstützt die Energieeinsparung, Verringerung der Aufgabenerledigungszeit und die Senkung der Fehlerquote. Ein Lastenausgleichsmechanismus wird angewendet, um die Aufteilung der Tasks auf mehrere Hosts zu verbessern. Analyse und Simulationsergebnisse zeigen, dass OR-TMS die Echtzeitfähigkeit, die Zuverlässigkeit und die Energieeffizienz optimiert.

Abstract

At the present time, there is widespread deployment of hybrid networks including wired (e.g., Ethernet networks) and wireless technologies, especially in automation systems, smart cities and home area networks. On the one hand, due to topology stability and a lower chance of packet loss, wired networks provide improved reliability and performance. On the other hand, compared to wired infrastructure, wireless networks offer greater flexibility and are less expensive. Recently, many systems have introduced hybrid networks in applications with real-time and safety requirements. The Time-Sensitive Networking (TSN) task group introduced for wirebound Ethernet networks a series of IEEE 802.1 sub-protocols to offer a fault-tolerant and deterministic communication infrastructure. Mapping the concepts of TSN to wireless environments requires taking into account the dynamic topology, signal interference, reliability and data transmission behaviour. Therefore, this thesis extends the TSN capabilities over hybrid (wireless and wirebound) systems. The hybrid TSN-enabled network can be applied to different architectural designs and it can be modified to incorporate further sub-standards introduced by the TSN task group. Thereby, different TSN features including clock synchronization, time-aware shaping, and fault-tolerant message delivery can be implemented over nodes in a hybrid system.

This thesis focuses on real-time systems, where distributed computational and communication activities are coordinated using a global time base. Therefore, this work firstly proposes an extension of the standard time protocol (i.e. IEEE 802.1AS) for hybrid TSN systems to improve the clock synchronization. The extended protocol considers the deterministic delays and the clock drift, and the precision of the clock synchronization of the standard 802.1AS scheme is significantly improved. Furthermore, the issue of asymmetrical delays in the transmission of the timing packets is also taken into consideration in order to accommodate dynamic communication settings with mobile nodes. In order to monitor the traffic behavior of timing packets and to reject outlier values that may appear as a result of dynamic and asymmetric events, a Path Deviation Delay (PDD) filter is provided. The simulation results demonstrate that, in comparison to the standard 802.1AS protocol, the extended protocol increases the synchronization precision. The outcomes further demonstrate that it improves synchronization precision to under 1 microsecond. In contrast, when asymmetric delay ratios and mobile node scenarios are present, the synchronization error with the standard protocol greatly rises.

When all nodes in a hybrid TSN network are precisely synchronized with a global timebase, the tasks of a real-time application have to be scheduled to wireless hosts and their transmitted messages have to be scheduled in a manner that all real-time constraints are fulfilled. The task and message scheduling problem is challenging particularly in wireless systems because the spatial and temporal distribution of dependent tasks to wireless hosts must satisfy the precedence constraints and minimize energy consumption. The mutual interference between signals must be also considered depending on the message transmission intervals and the spatial proximity. Interference may prevent reception, cause the corruption of signals or affect the signal quality. Therefore, this thesis focuses on task scheduling and message scheduling in real-time wireless systems.

Most of prior scheduling algorithms for wireless networks consider either the perspective of saving-network resources, routing message restrictions, or the impact of interference. Therefore, to address the interference and routing restrictions for the message scheduling besides the task scheduling, a novel task and message scheduling algorithm named Task and Message Scheduling for wireless networks (TMS) is proposed. In this algorithm, Time-Triggered (TT) messages of the computational tasks use the optimal routes with minimum latency from all available routes. In addition, each message is fragmented and assigned to several time-slots. In each time-slot, messages can be transmitted simultaneously by using a physical interference model. TMS supports multi-cast communication while respecting the precedence constraints between computation tasks and period constraints. To evaluate the proposed algorithm, several algorithms that use different routing strategies are implemented. The experimental results show that TMS outperforms solutions from the related work in the simulation tests, where makespan, consumed energy, failure rate and deadline missing cases are used as metrics. Moreover, in experiments, it is observed that changing the parameters of the used interference model affects the performance of the compared algorithms.

The seamless recovery from faulty behaviours is a critical issue for real-time and time-sensitive systems. Therefore, this thesis presents algorithms on top of the TMS to incorporate message replication over selected redundant routes to avoid any potential link or node failure. Two reliable algorithms are presented, the first one named Reliable Task and Message Scheduling for wireless networks algorithm (R-TMS), the second one uses a bio-inspired optimization algorithm to find better solutions, which is named Optimized Reliable Task and Message Scheduling for wireless networks algorithm (OR-TMS). The proposed reliable algorithms use an approach called Frame Replication and Elimination for Reliability (FRER) to replicate the communication messages through redundant disjoint routes.

In R-TMS, the system reliability is improved through scheduling tasks to wireless hosts with high performance in terms of message arrival time, energy consumption and failure rate. A reliability model is also introduced to determine the reliability of the system. The reliability model computes the reliability of each task depending on the reliability of all its incoming messages. The reliability of the leaf task, which has no forwarding messages, presents the global reliability of the overall system. R-TMS is compared with state-of-the-art TT algorithms that use the shortest or load-aware routes to send messages or that schedule the tasks to hosts that have the minimum ready-time. The experimental results show that the reliability of the system computed by R-TMS is improved compared to the other algorithms while also ensuring scalability in the network design and timeliness. The impact of the injected link failures on the message delivery ratio is also studied.

The Discrete Particle Swarm Optimization (DPSO) algorithm, on which OR-TMS is based, establishes a utility function for optimizing the task and message scheduling problem by trying to find a better solution iteratively. The defined function is created to achieve objectives like energy conservation, reducing task completion time, and lowering failure rates. A load balance mechanism is applied to improve balancing of the tasks on several hosts. Analysis and simulation results show the optimization of OR-TMS timeliness, deadline miss ratio, failure rates and energy efficiency compared it to prior algorithms that schedule tasks to hosts that produce the minimum completion time.

Contents

List of Figures

List of Tables

List of Abbreviations

ACL	**A**ccess **C**ontrol **L**ist
ALU	**A**rithmetic **L**ogic **U**nit
AVB	**A**udio **V**ideo **B**ridging
AV	**A**udio **V**ideo traffic
BER	**B**ackward **E**rror **R**ecovery
BEATA	**B**alanced **E**nergy **A**ware **T**ask **A**llocation algorithm
BAG	**B**andwidth **A**llcation **G**ap
BE	**B**est **E**ffort traffic
BMCA	**B**est **M**aster **C**lock **A**lgorithm
BPS	**B**idirectional **P**ipelining **S**chedule
BCs	**B**oundary **C**locks
BPDU	**B**ridge **P**rotocol **D**ata **U**nit
CSMA/CA	**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
CCU	**C**entral **C**omputing **U**nit
CR-SLF	**C**hannel **R**euse-based **S**mallest **L**atest-start-time **F**irst
CoS	**C**lass of **S**ervice
CCC	**C**lear **C**hannel **C**ounter
CCF	**C**lear **C**hannel **F**lag
CTS	**C**lear **T**o **S**end
CMTS	**C**luster-based **M**aximum consensus **T**ime **S**ynchronization
CCTS	**C**lustered **C**onsensus **T**ime **S**ynchronization
CATP	**C**ompatibility **A**ware **T**ask **P**artition
CAP	**C**ontention **A**ccess **P**eriod
CFP	**C**ontention **F**ree **P**eriod
CW	**C**ontention **W**indow
CAN	**C**ontroller **A**rea **N**etwork
CBQ	**C**redit **B**ased **Q**ueuing
CBS	**C**redit **B**ased **S**haping
CRC	**C**yclic **R**edundancy **C**heck
CSRO	**C**umulative **S**caled **R**ate **O**ffset
DoS	**Denial** of **S**ervice
DPSO	**D**iscrete **P**article **S**warm **O**ptimization algorithm
DCTS	**D**istributed **C**onsensus **T**ime **S**ynchronization algorithm
DCF	**D**istributed **C**oordination **F**unction
DDPaS	**D**istributed **D**ynamic **P**acket **S**cheduling
DIFS	**D**istributed **I**nter-Frame **S**pace
DOOTA	**D**istributed **O**ptimal **O**nline **T**ask **A**llocation algorithm
DRTS	**D**istributed **R**eal-Time **S**ystem
DSSS	**D**irect **S**equence **S**pread **S**pectrum
DTAW	**D**ivisible **T**ask scheduling **A**lgorithm for **W**ireless sensor networks
DHRP	**D**ynamic **H**yper **R**ound **P**olicy
DMS	**D**ynamic **M**odulation **S**caling

DVS	**D**ynamic **V**oltage **S**caling
EDF	**E**arliest **D**eadline **F**irst
EDL	**E**arliest **D**eadline **L**ate
ET	**E**event **T**riggered
FT	**F**ault **T**olerance
FCR	**F**ault **C**ontainment **R**egion
FTAOA	**F**ault-tolerant **T**ask **A**llocation **A**lgorithm
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FIFO	**F**irst **I**n **F**irst **O**ut
FER	**F**orward **E**rror **R**ecovery
FQETS	**F**orwarding and **Q**ueuing **E**nhancements for **T**ime **S**ensitive streams
FRER	**F**rame **R**eplication and **E**limination for **R**eliability approach
FDMA	**F**requency **D**ivision **M**ultiple **A**ccess
FD-PaS	**F**ully **D**istributed **P**acket **S**cheduling
GCL	**G**ate **C**ontrol **L**ist
GA	**G**enetic **A**lgorithm
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
GMC	**G**rand **M**aster **C**lock
GUI	**G**raphical **U**ser **I**nterface
GTS	**G**uaranteed **T**ime **S**lots
HMI	**H**uman **M**achine **I**nterface
I-EDF	**I**mplicit **E**arliest **D**eadline **F**irst protocol
ITAS	**I**ntelligent **T**ask **A**llocation **S**cheme
JSON	**J**ava **S**cript **O**bject **N**otation
LST	**L**atest transmission **S**tart **T**ime
LCM	**L**east **C**ommon **M**ultiple
LACP	**L**ink **A**ggregation **C**ontrol **P**rotocol
LLC	**L**ogical **L**ink **C**ontrol
LPRT	**L**ow **P**ower **R**eal **T**ime protocol
MTU	**M**aximum **T**ransmission **U**nit
MAC	**M**edium **A**ccess **C**ontrol
MSG	**M**ulti **S**chedule **G**raph
MLTS	**M**inimum traffic **L**oad **T**ask **S**cheduling algorithm
MMTS	**M**in-**M**in **T**ask **S**cheduling algorithm
mRFTAS	modified **R**eal-time **F**ault-tolerant **T**ask **A**llocation **S**cheme
MOP	**M**ulti-objective **O**ptimization **P**roblem
MIMO	**M**ultiple **I**nput **M**ultiple **O**utpot
MMRP	**M**ultiple **M**AC **R**egistration **P**rotocol
MSTP	**M**ultiple **S**panning **T**ree **P**rotocol
MSRP	**M**ultiple **S**tream **R**eservation **P**rotocol
NRR	**N**eighbour **R**ate **R**atio
NIC	**N**etwork **I**nterface **C**ard
NTP	**N**etwork **T**ime **P**rotocol
NMR	**N**-**M**odular **R**edundancy technique
NC	**N**ormal **C**lock
NoTP	**N**ormal **T**ransmission **P**eriod
NVP	**N-Version P**rogramming
OTSA	**O**ptimal **T**ask **S**cheduling **A**lgorithm
OR-TMS	**O**ptimized **R**eliable **T**ask and **M**essage **S**cheduling algorithm for wireless networks
OFDM	**O**rthogonal **F**requency **D**ivision **M**ultiplexing modulation technique

PTN	**P**acket **T**ransport **N**etwork
PSO	**P**article **S**warm **O**ptimization algorithm
PDT	**P**ath **D**elay **T**able
PDD	**P**ath **D**eviation **D**elay
PR-MAC	**P**ath oriented **R**eal-time **M A C**
TLV	Path trace **T**ype **L**ength **V**alue
PIFS	**P**CF **I**nter-**F**rame **S**pace
PVST	**P**er **V**LAN **S**panning **T**ree
PLL	**P**hase **L**ocked **L**oop
PCF	**P**oint **C**oordination **F**unction
PTP	**P**recision **T**ime **P**rotocol
PCP	**P**riority **C**ode **P**oint
PQ	**P**riority **Q**ueuing
PrCF	**Pr**otocol **C**ontrol **F**rame
RC	**R**ate **C**onstrained traffic
RMS	**R**ate **M**onotonic **S**cheduling
RRMAC	**R**eal-time and **R**eliable **M A C**
RFS	**R**eal-time **F**low **S**cheduling
RTWN	**R**eal-**T**ime **W**ireless **N**etwork
RD-PaS	**R**eliable **D**ynamic **Pa**cket **S**cheduling
R-MMTS	**R**eliable **M**in-**M**in **T**ask **S**cheduling algorithm
R-TMS	**R**eliable **T**ask and **M**essage **S**cheduling algorithm for wireless networks
RBD	**R**eliability **B**lock **D**iagram
RMEC	**R**eliability **M**aximization with **E**nergy **C**onstraint algorithm
RTS	**R**equest **T**o **S**end
RP	**R**etransmission **P**eriod
STS	**S**hortest **T**ask **S**cheduling algorithm
SIFS	**S**hort **I**nter-**F**rame **S**pace
SINR	**S**ignal to **I**nterference and **N**oise **R**atio
SNR	**S**ignal to **N**oise **R**atio
STP	**S**panning **T**ree **P**rotocol
STD	**S**tate **T**ransition **D**iagram
SRP	**S**tream **R**eservation **P**rotocol
SDR	**S**ymmetric **D**egree **R**atio
TCVCXO	**T**emperature-**C**ompensated **V**oltage-**C**ontrolled **C**rystal **O**scillator
TAS	**T**ime-**A**ware **S**haper
TDMA	**T**ime-**D**ivision **M**ultiple **A**ccess
TSN	**T**ime-**S**ensitive **N**etworking
TSCH	**T**ime-**S**lotted **C**hannel **H**opping
TT	**T**ime-**T**riggered traffic
TC	**T**ransparent **C**lock
TMR	**T**riple **M**odularity **R**edundancy technique
UTC	**U**niversal **C**oordinated **T**ime
UAV	**U**nmanned **A**erial **V**ehicle
VTS	**V**irtual **T**DMA for **S**ensors
WBAN	**W**ireless **B**ody **A**rea **N**etwork
TMS	**T**ask and **M**essage **S**cheduling algorithm for wireless networks
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WNCS	**W**ireless **N**etworked **C**ontrol **S**ystem
WSN	**W**ireless **S**ensor **N**etwork

Chapter 1

Introduction

1.1 Context and Motivation

Real-time hybrid communication systems combine wireless and wirebound networks have been gaining increasing popularity in recent years thanks to the ease of deployment, enabling flexibility, stability and accessibility. Therefore, they have been engaged in a variety of applications such as border monitoring, environmental tracking, home automation and smart architecture [1]–[3]. Whenever a hybrid system interacts with a real environment through its nodes, it normally decomposes an application into several tasks that are scheduled on several wireless hosts (e.g., sensors and actuators) so that messages are used to pass content between the interconnected tasks.

To ensure that all tasks are done in the correct order and before deadlines, firstly, it is very helpful that hardware clocks are synchronized accurately so that all local clocks share the same global (i.e. reference) time. Secondly, it is necessary to optimize the scheduling of tasks over the potential wireless hosts, taking into account the scheduling of the connecting messages. Due to the stochastic nature of wireless communication, the task and message scheduling problem is considered a significant challenge in the face of bandwidth and energy limitations. Bandwidth and energy are consumed due to the effects of media access control and contention resolving, signal interference and channel fading. Therefore, the task and message scheduling solution needs to preserve the network resources by balancing the traffic load over several nodes and reducing the signal interference that causes message retransmissions.

Many real-time hybrid networks not only rely on guaranteed timely delivery of data, but its nodes and links can be subject to failures. In order to prevent a failure from degrading the network performance, fault tolerance becomes a necessity for the continuity and functionality of hybrid networks. Fault tolerance becomes even more important when adopting wireless technologies in hard real-time systems (e.g., medical applications, battlefield surveillance) where the transmitted data is critical and it should arrive with stringent deadlines [4]. These fault-tolerance requirements add further difficulties to the already demanding problem of scheduling tasks among wireless hosts of a hybrid system.

The current trend is towards time-triggered wireless approaches due to the increasing demand for reliable and deterministic wireless infrastructures in real-time and safety-critical systems [5]. The use of the time-triggered technique is advantageous from the perspective of (i) building a network with predictable real-time response times, (ii) establishing an architecture with a global idea of time and clearly defined interfaces reduces system complexity and thus makes it easier to create real-time applications, (iii) distributing an application's tasks across several hosts to reduce maintenance costs, (iv) messages transmitted with less jitter and latency. The time-triggered wired approach has already been successfully applied to safety-critical systems in transportation applications [6]–[8] (e.g. vehicle applications, railway control systems and flight-critical functions in aircraft).

Time-Sensitive Networking (TSN) [9] and TTEthernet [10] are examples of time-triggered infrastructures that satisfy mixed-critically requirements, while at the same time satisfying real-time constraints. However, it is challenging to offer in a wireless environment TSN capabilities, which are interoperable and comparable with existing wired TSN standards. Recent advances in 5G and IEEE 802.11 wireless connectivity technologies with low latency and high reliability have generated significant interest towards extending TSN capabilities to wireless systems. In this context, scheduling tasks to hosts interconnected by a wireless network with TSN capabilities is an open research challenge. In the following, the terminology *'WirelessTSN'* is used to refer to a wireless network that extends IEEE 802.1 TSN capabilities to wireless media.

In *WirelessTSN*, time synchronization is critical because message injection times and task dispatching involve a time-triggered communication schedule that is defined with respect to a global time base. Time also provides the frame of reference between all nodes on the network. Without sharing a global time base, the temporal coordination of activities is difficult. Therefore, the concept of a precise global time base has to be considered in order to achieve a correct and meaningful behaviour. IEEE 802.1AS [11] is an extension of the Precision Time Protocol (PTP) [12] used for precise timing in wired TSN standards and Audio Video Bridging (AVB) [13] with switched Ethernet networks. To meet the challenges of *WirelessTSN* in a hybrid TSN network, IEEE 802.1AS has to be improved and extended.

1.2 Thesis Objectives and Contributions

Highly reliable, scalable and deployable networks satisfying strict temporal constraints are inevitable for future cyber-physical systems. Due to the widespread usage and success of wireless technologies and the capabilities of wired TSN standards that provide real-time capabilities and performance improvements, this work aims to extend TSN services from wired to wireless domains and operation models. Therefore, this thesis presents a *WirelessTSN* model, which is developed as an extension to an Ethernet-based model in the Riverbed simulator [14]. The hybrid model provides models of generic TSN-enabled nodes that are used to share services between wireless and wirebound TSN segments. The hybrid model also allows verifying the integrated services of TSN with respect to time-aware shaping and timeliness. In other words, the node models are defined to support IEEE 802.1AS, IEEE 802.1Qbv [15] and IEEE 802.1CB [16] features. To be more specific, IEEE 802.1Qbv introduces a novel scheduling mechanism called Time-Aware Shaper (TAS). TAS determines which frame can be transmitted at each instant of time based on the Gate Control List (GCL) idea. On the other hand, Frame Replication and Elimination for Reliability (FRER) is a revolutionary fault-tolerance method defined by IEEE 802.1CB [16]. FRER offers highly reliable communication for TT traffic by forwarding messages via redundant routes.

To achieve the real-time capabilities especially in hard real-time systems which have strict temporal requirements and based on existing wireless protocols (e.g., IEEE 802.11), a robust clock synchronization mechanism becomes a necessity. The importance of having precise synchronization is due to the fact that TSN time-based properties (e.g. IEEE 802.1Qbv and IEEE 802.1QCB standards) are based on the precision of the global time. Thereby, this thesis presents an extension for IEEE802.1AS standard synchronization protocol that is, in turn, based on IEEE 1588 V2 [17]–[19]. The extended protocol enables the time distribution between wired and wireless TSN domains over IEEE 802.11. A Best Master Clock Algorithm (BMCA) algorithm is implemented to help choose which clock to use as the grandmaster of timing on the TSN network. In addition, an asymmetric peer delay mechanism and a Path Deviation Delay (PDD) filter are introduced to compute the induced deterministic delays, clock drift and asymmetric delays in *WirelessTSN*. The simulation framework of time-aware systems shows significant results in terms of reducing the synchronization error between the grandmaster clock and its slave clocks with variable asymmetry ratios and variable frequency offsets. Moreover, the behavior of the extended TSN synchronization mechanism is evaluated in the presence of different mobile speeds and different numbers of the participating wireless TSN-enabled nodes.

The thesis also provides solutions to challenges faced by wireless time-triggered communication in terms of signal interference, channel fading and limited resources. Most of the state-of-the-art solutions adopt strategies of scheduling tasks, either to consider the system's completion time and energy consumption, or to consider the workload balancing, or even to support a reliable task scheduling. Hence, the previous approaches limit themselves to finding a suitable task scheduling solution that fulfills the requirements of real-time systems. This thesis provides contributions beyond the state-of-the-art by presenting heuristic task and message scheduling strategies that consider the following features: FRER is used to send multiple copies of messages through redundant routes, which limits the effect of link and node faults and makes the system more reliable. A physical interference model is applied to mitigate the negative impact of signal interference during the message transmissions. At the physical model, the message transmission is successfully performed between sender a and receiver b if the Signal to Interference and Noise Ratio (SINR) at b is above a certain threshold (i.e. β), which depends on the channel specifications and the standard of *WirelessTSN*. Furthermore, we consider the precedence constraints, period and deadline of the tasks. Finally, the thesis considers the resource utilization (i.e. energy consumption) and the workload balancing during the task and message scheduling process.

The presented heuristic algorithms lead to significant enhancements of the reliability, timeliness, efficiency and resource utilization compared to the state-of-art algorithms. To be more specific, our Task and Message Scheduling (TMS) algorithm for wireless networks is compared with other algorithms like Min-Min Task Scheduling (MMTS) [20] and Minimum traffic Load Task Scheduling (MLTS) [21], [22] algorithms. MMTS is based on scheduling a task to a wireless host that has the minimum ready time (i.e. the host that has the minimum start time to execute the task). MMTS is a traditional offline algorithm, which does not consider the effect of the message scheduling on the transmission paths during the task scheduling process. In MLTS, a task is scheduled to a host depending on routes that can be used to send messages from the parent tasks and have the least traffic load. MLTS uses the computed traffic load value to recognize when a route is becoming congested and then it chooses an alternate route. Routes with lower value and less congestion are preferred. While TMS firstly sorts all tasks according to their priorities, scheduling a sorted task to an available host takes into account the message scheduling over fixed duration time slots. The message scheduling considers, in turn, the period and the precedence of the cooperating tasks. In the event that all required messages have arrived, the task is scheduled to the host that receives the messages as fast as possible provided that it meets the task deadline condition. The task scheduling process continues until all sorted tasks are completed. The simulation tests show the improvement of timeliness, saving network resources, and the efficiency of the proposed task and message scheduling by TMS.

Reliable Task and Message Scheduling (R-TMS) and Optimized Reliable Task and Message Scheduling (OR-TMS) algorithms are proposed to find reliable task and message scheduling solutions using task priorities and Discrete Particle Swarm Optimization (DPSO). DPSO [23] is defined as a problem optimization tool, which is used in OR-TMS to solve the task and message scheduling problem by iteratively trying to improve a possible solution with regard to a cost function value. The cost function value considers multiple objectives including the system's completion time, total energy consumption and total failure rates. Thereby, the computed cost function value determines the cost of a task scheduling instance. R-TMS uses the same cost function value but in a different manner, where the function value is computed to select a suitable wireless host for every task gradually until finishing all tasks. A novel reliability model is introduced in R-TMS, which assesses the reliability of a system using interactions between interconnected tasks in the form of time-triggered messages. The reliability of message transmission is computed as a function of the reliability of the network nodes that engage in the message delivery. Consequently, it can be determined which nodes are more critical in the overall reliability of the system.

Extensive simulation tests are implemented to show the significance of the results of our fault-tolerant task and message scheduling algorithms in terms of reliability, timeliness, deadline miss ratio, failure rate, network lifetime and energy conservation. R-TMS outperforms MMTS, MLTS and the Shortest Task Scheduling (STS) [24], [25], which depends basically on selecting the shortest routes to send messages. To show the improvement in performance and reliability using OR-TMS, it is compared with R-TMS, Reliable Min-Min Task Scheduling (R-MMTS), and Reliable Shortest Task First Scheduling (R-STFS) [26] algorithms that have been developed to send multiple messages over disjoint redundant routes.

The main contributions of this thesis can be summarized as follows:

- A simulation framework is implemented supporting TSN capabilities (i.e. IEEE 802.1Qbv and IEEE 802.1QCB standards) over real-time wireless networks. The developed framework also simulates redundant transmission and scheduling mechanisms of real-time systems [27].
- The simulation models use an extended synchronization protocol based on IEEE802.1AS [27] for improved precision in asymmetric *WirelessTSN*. The proposed protocol includes different procedures such as selecting the best grandmaster clock, asymmetric peer delay measurement and path deviation delay filtering.
- TMS strategy based on a list task scheduling approach for time-triggered communication [28]. This strategy finds feasible task and message scheduling solutions by employing the following techniques:
  - Prioritizing each task according to the top-level values and then arranging tasks in ascending order.
  - Message scheduling over fixed duration time slots considering the interference that is induced by sending multiple messages at the same time. The message scheduling takes also into account the period and the precedence of the tasks.
  - Each task is scheduled to a wireless host that has the minimum start time provided that the task deadline is met.
- R-TMS strategy [29] is built on top of TMS and aims to improve the system reliability using the following techniques:
  - FRER features to support fault-tolerant message delivery.
  - A novel reliability model is used to keep the *WirelessTSN* system working properly even upon task failures.
  - A cost function comprised of factors including the task start time, task's consumed energy and task failure rate. The least cost function value is used to select the most suitable host for that task and the task and message scheduling process continues until scheduling all sorted tasks.

- OR-TMS strategy [30], which aims to improve the system reliability addressed by R-TMS by using further assumptions:
  - Improve the reliability by using the DPSO algorithm. The cost function in OR-TMS refers to the total energy saving, total completion time and total failure rates of a task scheduling instance. To the end, OR-TMS tries iteratively to improve the task scheduling instance with regard to the minimum cost function value.
  - A workload mechanism is implemented to distribute the execution of the tasks along with several available hosts.
  - A metascheduler based on OR-TMS is generated to provide a transition to a verified schedule at run-time. Therefore, the proposed metascheduler pre-computes schedules at design time.

1.3 Thesis Structure

After the introduction presented in this chapter, the remainder of the thesis is structured as follows:

- **Chapter 2**: Through an in-depth analysis, this chapter points out the challenging requirements of hybrid (i.e. wired/wireless) networks in real-time applications, then provides an overview of related work in the context of the precision time synchronization and TT task scheduling in different wired/wireless environments and finally provides an analysis of the research gaps.
- **Chapter 3**: Outlines the background theory to provide an understanding of the fundamental concepts used throughout this work. It starts with the concepts of real-time operating systems and distributed real-time systems including time-triggered and event-triggered communication. It then explores different aspects such as dependability, fault-tolerance techniques and timing protocols. This chapter provides also a detailed introduction to the IEEE 802.1AS standard, link scheduling in wireless networks and IEEE 802.11 DCF/PCF modes with their limitations. Finally, this chapter shows real-time communication protocols in wired and wireless systems.
- **Chapter 4**: introduces the system model that is used throughout the thesis. It contains architecture and application graphs that serve as inputs to simulate the proposed algorithms and compare them with related ones.
- **Chapter 5**: extends the TSN capabilities of the Ethernet-based network over the wireless network. The hybrid network is based on the system model from chapter 4, which covers hybrid real-time systems including wireless and wirebound networks. Wireless bridges and host models with support for TSN features are presented.
- **Chapter 6**: explains in detail the extended synchronization protocol based on IEEE802.1AS for improved the precision time synchronization in a hybrid TSN network.
- **Chapter 7**: optimizes the task and message scheduling solution, particularly in $WirelessTSN$. This chapter provides a detailed description of our TMS algorithm. After that, simulation results are presented to compare the proposed algorithm with the state-of-the-art algorithms such as MLTS and MMTS algorithms.
- **Chapter 8**: describes how the task and message scheduling strategies which are introduced in the TMS algorithm can be extended to support reliability requirements of

WirelessTSN. Therefore, this chapter provides a detailed description of our reliable R-TMS algorithm. The simulation results show the improvement in reliability and performance compared to previous work. After that, it provides a detailed description of our optimized OR-TMS using the DPSO algorithm. OR-TMS explains how to map a task scheduling instance into DPSO. Experimental results show a comparison of R-TMS and OR-TMS with prior algorithms such as R-STFS and R-MMTS algorithms. Finally, a metascheduler based on OR-TMS is implemented and evaluated using several experiments.

- **Chapter 9**: presents the conclusion of the thesis.

Chapter 2

Related Work and Research Gap

2.1 Requirements for Hybrid TSN Networks

Wireless networks have been adopted in several industrial systems thanks to their flexibility and reducing setup costs. Wireless networks often represent an effective alternative solution to avoid cabling that turns out to be expensive and cumbersome in many cases. Conversely, wireless networks are well known for problems like shadowing, interference and multipath propagation, which have a negative impact on the performance of the industrial real-time systems that are often known by their requirements in terms of reliability and timeliness. The features of wired and wireless networks can be combined by extending the existing wired networks with wireless segments. The resulting integrated configuration is known as a hybrid network. In this kind of network, controllers, servers, and human machine devices are located often in the wired segment because of stability and reliability, thus, all nodes in the wireless segment (i.e. sensors, access points, wireless routers) should interoperate seamlessly with the wired communication system. Consequently, the combination of a wired and wireless system is a critical issue, which has to satisfy the requirements of timeliness and reliability in real-time systems.

TSN standards, which are developed by the Time-Sensitive Networking task group, provide guaranteed delivery and minimized jitter for critical data that can be transmitted with non-critical data over the converged and synchronized standard Ethernet-based network. Moreover, TSN standards provide seamless redundancy for data that cannot tolerate message losses. The gained properties are relevant for hybrid systems since the tested and proven TSN features of the Ethernet-based segment can be extended to the TSN wireless medium. To achieve that, the following requirements appear for hybrid TSN networks:

- **Bounded latency**: the hybrid TSN network requires guaranteed end-to-end latency (i.e. predictable and bounded) for the delivery of messages. To do that, the available data rate and net throughput in the wireless medium have to be considered during the message transmissions. Even operating at the maximum data rate, the net throughput may be lower compared with the TSN wired network, because of the shared medium among all wireless nodes. Moreover, the random access techniques (e.g., Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA)) are mostly used by the wireless MAC protocols (e.g., IEEE 802.11). This means that unpredictable access to the medium and network congestion can lead to unbounded delays and nonnegligible jitter. Hence, this kind of hybrid networks would not be acceptable for real-time applications.

- **Interference avoidance**: besides the random access, wireless channels are also prone to signal interference that may increase transmission errors and compromise network reliability. Hence, scheduling access to the wireless medium using time-division multiple access is considered as an efficient solution to avoid the impact of signal interference and simultaneous message retransmissions.

- **Energy saving**: a serious drawback of a hybrid TSN network is the imposed energy overhead caused by the operations in every node of the wireless network segment.
- **Precise global time service**: the establishment of an accurate global notion of time is an important service to successfully coordinate and organize the tasks of a real-time application that is distributed throughout hosts in a hybrid TSN system. Therefore, it is essential to first take into account the limitations of the wireless medium as well as factors that reduce the precision of synchronization, such as aging hardware components and inaccurate time-stamping.
- **Optimized task and message scheduling solution**: challenges of wireless communication like signal fading (i.e. signal interference), bounded latency and limited energy make scheduling of tasks in a hybrid system a critical issue. In other words, the constraints of the real-time application must be fulfilled during the scheduling of the tasks to the distributed hosts. In particular, these constraints demand that all tasks finish before their deadlines, any task waits for all incoming messages before starting its execution, the transmitted messages have to be scheduled in specific times to consider the period of the sending tasks and the negative impacts of signal interference, limited energy, and routing loops have to be avoided.
- **Reliable task and message scheduling**: If a system needs high reliability in the event of link or node failures, the distribution of dependent tasks among multiple hosts becomes more complicated. Therefore, fault tolerance techniques have to be imposed to continue operating the tasks without interruption in the presence of faults.
- **Data traffic regulation**: wired and wireless mediums are connected through specific nodes called bridges (i.e. relay nodes). These nodes have different or dissimilar structures, complexity, and functionality from those of wired and wireless nodes. Therefore, they have to operate efficiently on reforming, buffering, re-timing, and regulating the access to the shared medium for the passing messages.

Based on the aforementioned challenges, we conclude that the interoperability between wireless IEEE 802.11-based extensions to TSN Ethernet-based networks is not as straightforward an interconnection. Moreover, unpredictable behaviours caused by interference, random access procedures, and transmission error, may significantly impair the performance and reliability of hybrid TSN networks. Therefore, this thesis aims to build a hybrid TSN system model that satisfies the mentioned requirements.

2.2 Related Work

In this section, we review related work in the areas of bounded latency and limited energy, interference-awareness, time synchronization, task scheduling algorithms, and fault-tolerant task scheduling techniques. These areas are discussed, because they correspond to the research requirements identified in the previous section and addressed in the thesis.

2.2.1 Related Work on Guarantees for End-to-End Delay and Energy-saving

Authors introduced several algorithms to bound the message communication latency with the purpose of minimizing the end-to-end delay. Most researchers adopted Time-Division

Multiple Access (TDMA) to ensure deterministic real-time data transmission. Centralized and static schemes for schedulability of real-time messages [31]–[33] fit well for small-scale networks but their efficiency degrades on finding feasible solutions in large networks or where disturbances exist. Another thread of research advances the static schemes by providing dynamic schemes for the message scheduling. Among these schemes, the authors in [34] handled disturbances (e.g. link qualities and traffic demands) in Real-Time Wireless Networks (RTWNs). Dynamic and deterministic schemes have been proposed by supporting admission control to add and remove streams. Although the proposed schemes, which are based on the earliest-deadline-first policy, meet all deadlines in hard real-time multi-hop wireless systems, they do not consider how to deal with precedence constraints between message senders. The work in [35] presented emergency communication protocols that reserved time slots to satisfy the deadline constraints in the case of emergencies. Hence, regular operations may not be finished before their deadlines. According to the suggestion in [36], each node should create its own transmission schedule using local data. Hard real-time requirements cannot be met by the chosen schedule because nodes that only use local information and have an impact on their surrounding neighbors cannot ensure end-to-end data delivery.

To ensure bounded latency and reliability bounds, the authors in [37] proposed a generic heuristic scheduler called SchedEx algorithm, which guarantees the reliability of the end-to-end transmission. The work in [38] addressed strict reliability and energy limits by using channel hopping and medium access control techniques. The authors in [39] addressed a message-centric policy and dynamically reallocate the multi-hop message transmissions among links depending on their qualities.

Several techniques have been proposed that focus on the communication cost to conserve the depleted energy in wireless systems. For instance, task clustering techniques [40], [41] and routing approaches [42], [43] have been introduced to balance transmission load to dedicated clusters or reduce the message transmission distance, respectively. In addition, some researchers introduced compression techniques to decrease the size of the transmitted messages [44], [45]. The work in [46], [47] studied sleep/wakeup schemes to save the energy spent by idle components.

2.2.2 Related Work on Interference-Aware Message Scheduling

A set of methods were proposed to avoid the induced interference due to transmitting different messages from different senders simultaneously. For instance, Chipara, Octav, et al. [48] presented a conflict-free message transmission scheme named Real-time Flow Scheduling (RFS). It supports spatial reuse, reliability by incorporating message retransmissions during the link scheduling process and scalability by dividing the network components into neighbourhoods. Jamthe, Anagha, et al. [49] scheduled multiple transmissions from different health care sensors in Wireless Body Area Networks (WBANs). The proposed approach uses IEEE 802.15.6 (TG6) as a standard protocol to avoid inter and intra-WBAN interferences. The coordinators in WBANs exchange information between them before sending the sensors' messages and they divide the channel into super-frames. The coordinators then assign time-slots, so that a sensor classified as the highest priority sends its data in the first available slot. However, this scheme does not consider the changes in the network topology and in the surrounding environments. In addition, the transmission time is high in the waiting mode and it is low for high-priority data. A heuristic static off-line algorithm is presented in [50] to schedule the data traffic and predict the radio interference according to the analysis of the global network state. Besides, a routing update scheme is applied to distribute the interfering streams provided that deadline constraints are met. This algorithm can produce highly optimized solutions in small-scale and static networks, in contrast to highly dynamic networks. Fateh, Benazir, and Manimaran Govindarasu [51] formulated the scheduling of tasks and messages as a joint problem. The authors proposed a three-phase heuristic to perform the task scheduling by taking into account deadline, interference and precedence constraints. The objective of this work is to improve the efficiency of energy-saving by using Dynamic Modulation Scaling (DMS) [52] for messages and Dynamic Voltage Scaling (DVS) [53] for tasks. This work was also improved to perform dynamic task scheduling.

2.2.3 Related Work on the Time Synchronization of Distributed Clocks

Due to the distributed nature of wired/wireless nodes, precise synchronization plays an important role in coordinating tasks especially those facing real-time constraints. In the case of asynchronous clocks, messages may arrive in an undesirable order, then whole system operations may not produce correct results, especially synchronous tasks with precedence constraints that make analysis and innovation of experimental task scheduling algorithms more complex. Moreover, the synchronization error causes inconsistencies in the times when the hosts are ready to perform the tasks. Thus, tasks cannot be executed at specific times. To meet the challenges of precise time synchronization, several excellent types of research have been proposed.

In the last years, several work studied intensively different time synchronization protocols using various simulation frameworks, especially for wireless technologies. Some research addresses asymmetric delays to mitigate the temporal unpredictability of the network. For instance, the authors of [54] applied the Precision Time Protocol (PTP) to IEEE 802.11 Wireless Local Area Network (WLAN) to solve the problem of asymmetric links. A delay filtering algorithm based on Kalman filters was designed as a pure software-based system and applied to coordinate each local clock with its grandmaster clock. The algorithm dealt with asymmetric delays in a single-hop WLAN environment by considering deterministic and random delays. Software time-stamping in the application layer has a negative impact since there will be an extra overhead passing through layers. It can also take longer, because of more CPU usage and more complexity. In contrast, hardware time stamping makes the time synchronization more precise. The authors in [19] proposed an enhanced IEEE 1588 time synchronization algorithm. Procedures called block burst transmission and offset correction are introduced to calculate the asymmetry ratio of the communication link and to calculate the offset values, respectively. The proposed algorithm requires additional message exchanges for femtocell network environments and Packet Transport Networks (PTNs).

Some work assumes that the hardware clock of the network nodes is accurate in frequency [55] or even the effect of the clock drift is neglected [56]. In fact, the change of the frequency leads to a remarkable drifting of the clocks in the same network. Therefore, some research addresses the clock drift compensation to establish accurate synchronization protocols. For instance, the authors in [57] proposed an optimized version of PTP for simple wireless networks. By utilizing the delayed transmission feature in the DW1000 transceiver, the optimized protocol inserted the exact sending time-stamp into the synchronization message to reduce the timing packet overhead. The goal of [58] was computing clock drift more precisely by applying adaptive time compensation. The implementation used standard Time-Slotted Channel Hopping (TSCH) control messages [59] in multi-hop networks. The accuracy of the clock offset computation for a standard PTP scheme was greatly increased by the authors in [60] by using a clock drift factor. The symbol timing synchronization was also studied, which enabled robust time-stamp message decoding during the clock synchronization period. The authors in [61] used the CC2420 transceiver which is compliant with the physical layer of IEEE 802.15.4/ZigBee. The authors maintain a clock offset of 200 nanoseconds between the local clocks and their reference clock. To eliminate the difference of the frequency skew between two clocks, a local clock computes both the drift rates of the reference clock and its clock within the sending synchronization interval and in the arrival synchronization interval, respectively.

Most time synchronization schemes are restricted to applying software time-stamping due to the lack of hardware time-stamping capability [62]–[64]. One of the software time-stamping disadvantages is that it generates non-deterministic delays. Therefore, some research is addressed to improve the time synchronization by moving the time-stamping to lower layers. As part of improving the performance of hardware-based time-stamping solutions, the authors of [55] presented a physical layer time-stamping approach for IEEE 802.11b protocol in a Field-Programmable Gate Array (FPGA) platform. The overall clock synchronization accuracy was shown to be sub-100 picoseconds and the standard deviation to be around 500 picoseconds. The proposed time-stamping approach is only applied by a stable oscillator with a fixed frequency. Moreover, multipath propagation is considered as a limitation factor. The work in [65] presented an expansion time synchronization module and a circuit that is used for clock-rate adjustment and as a global clock source in a FireFly3 wireless sensor platform. The clock-rate adjustment system has the advantage for adjusting the clock rate for FireFly3's processor while it is in sleep mode. This leads to precise instantaneous time-stamping, reducing the clock drift to 0.01 ppm and saving 60% of the consumed energy compared to the original design. The disadvantage of this approach is that extra hardware requires more cost and energy consumption. In addition, the system is only internally synchronized. Another approach that is proposed to use hardware time-stamping is explained in [66], where time synchronization timers are used for hardware time-stamping so that both offset and rate correction can be performed at the node.

The authors of [67] proposed an algorithm called Clustered Consensus Time Synchronization (CCTS) for Wireless Sensor Networks (WSNs). Their algorithm is based on the Distributed Consensus Time Synchronization (DCTS) algorithm with a clustering technique. CCTS improves the convergence rate due to the incorporation of the Cluster Head (CH) which is responsible for updating skew compensation and offset compensation parameters for intra-cluster time synchronization. After that, the CH exchanges messages via gateway nodes in order to support inter-cluster time synchronization. CCTS has some drawbacks. For instance, the static synchronization period is a limitation of this algorithm and the requirement of initial consensus does not make it an ideal solution for unpredictable and dynamic topologies that require continuous adaptation. Moreover, packet losses, asymmetric links, node failures and cluster membership changes will degrade the performance of CCTS. Another method that incorporates the clustering technique is proposed in [68]. The authors of [68] presented a Cluster-based Maximum consensus Time Synchronization (CMTS) method which includes intra-cluster and inter-cluster time synchronization parts. The impact of the communication delays is minimized by the Revised-CMTS method. This work shows the effect of using the clustering technique on the convergence time and overhead. On the other side, the energy reduction over time and the mobility of nodes cause variation in the transmission ranges, variation in the cluster membership and dynamic communications delays. Therefore, the CMTS method needs to be improved to deal with dynamic network topologies.

Some state-of-the-art dealt with both asymmetric delays and clock drifts. For instance, the study by Lam et al. [69] provided two approaches. In the first approach, the standard PTP algorithm is improved to deal with clock skews in an industrial WLAN. The improved protocol adds two additional timestamps, transmission and reception timestamps of the follow-up synchronization message, without changing the number of messages sent between two clocks. In the second approach, a method is derived to deal with asymmetric hardware processing latencies to reduce the error caused by asymmetric message delay on download and upload links.

For hybrid networks, the authors in [70] have optimized an IEEE 1588v2 protocol to compensate non-deterministic delays for synchronization messages over a wireless segment: The reference clock adjusts the sending rate of the synchronization messages according to the induced forwarding jitter. Hence, this work leads to variable and unpredictable message overhead as well as synchronization error in the range of microseconds. Similarly, the accuracy obtained using [71] is in the millisecond range. The authors in [72] have presented IEEE 802.1AS based solutions over audio-video bridging networks. Therefore, it is not optimized to work with cyclic control loops and time-triggered data transmissions. The work in [73] is presented to enhance the net throughput rather than reducing delays.

Clock synchronization based on Ethernet has been researched in many papers. The main difference between wired and wireless frameworks is that the message transmission delay and the time of medium access are already known in the wired medium, therefore, the clock synchronization in fixed topology wired systems is easier. For instance, the authors in [74], [75] presented time synchronization protocols, which are based on industrial Ethernet, to address the clock drift and precise timestamping, respectively.

In contrast to the related work, our work extends IEEE 802.1AS to improve the time synchronization for scalable hybrid TSN networks by considering the aforementioned factors (i.e. asymmetric delays, clock drift and precise time-stamping) that affect the time synchronization accuracy. The timestamping in the extended protocol is applied to the MAC layer instead of higher layers to avoid traffic overload, to be more accurate and ensure a faster response to time synchronization messages. Moreover, the extended protocol focuses on accurately measuring clock drift and path delays. Although the path delay is a sensitive value, deterministic delays are considered to reduce the asymmetric delays brought on by unpredictably changing environmental conditions (e.g., mobility [76]). As an additional step, a PDD filter is applied to exclude outlier path delay values resulting from traffic congestion or lossy links in a dynamic network. To the best of our knowledge, this work is the only one that addresses a global and unified clock for a hybrid network with TSN-based wireless and wirebound systems. While there is a trend to apply hybrid TSN networks in various industrial fields [77], the time synchronization in hybrid systems is an open research question.

2.2.4 Related Work on the Task Scheduling Schemes

Based on precise time synchronization for all nodes of hybrid systems, a necessary step comes to provide an feasible solution to solve the task scheduling problem of temporally and spatially mapping tasks to nodes. This problem has been intensively studied because it is considered a big challenge particularly for wireless technologies. As a result, several work has lately been developed with the aim of shortening task completion times. Another thread of work in the literature advances the state of the art by providing a variety of energy-aware task scheduling algorithms. Therefore, in this subsection, we review the most significant results in the field of task scheduling algorithms for RTWNs.

Numerous task scheduling schemes have been proposed for wired networks (e.g., [78]–[80]). These schemes do not satisfy the desired efficiency in wireless networks. Therefore, the task scheduling scheme for wireless industrial environments [81] is ongoing research, which is considered to be a sensitive issue and a challenging requirement for our thesis.

Prior work considered unpredicted internal and external disturbances when guaranteeing Quality of Services (QoS). For instance, an on-line framework is used in [82] to adjust the static schemes to respond to external and sporadic disturbances. A Wireless Networked Control System (WNCS) reconfigures at a bounded time after system changes to reduce the impact of system dynamics on current flows. Zhang, Tianyu, et al. [83] introduced a Distributed Dynamic Packet Scheduling (DD-PaS) framework. DD-PaS locally constructs task scheduling schemes at individual components, which in turn minimizes the disseminated information. DD-PaS also applies a packet dropping algorithm at a centralized gateway to determine on-line which packets can be eliminated as a result of disturbances. Zhang, Tianyu, et al. [84] later introduced another framework named Fully Distributed Packet Scheduling (FD-PaS) framework for handling disturbances faster and locally without centralized control. However, the mentioned work assumes that the wireless links are perfect without noisy or harmful factors that lead to packet loss.

On the other side, a set of research has addressed message scheduling to improve the reliability of communication over lossy links during the task scheduling process. For instance, the solution in [85] is based on Medium Access Control (MAC) [86] and TSCH at industrial environments. Chen, Yu, et al. [87] proposed a framework to translate the communication requirement and the link reliability into a fixed number of transmission opportunities reserved for each packet to achieve the required delivery packet ratio in a wireless sensing and control system. The lack of flexibility to deal with frequent changes in link quality and the irregularity of failure occurrence is one of the drawbacks of this work. Therefore, a message-centric policy is proposed in [39] to reallocate message retransmissions among links more flexibly. The aforementioned work handled message scheduling in static wireless environments, while the work in [88] is proposed to handle the packet scheduling over lossy links in RTWNs. However, the retransmissions policy in [87], [88] leads to consuming significant network bandwidth.

The task scheduling problem has been solved using some bio-inspired optimization methods, such as Genetic Algorithms (GA) [89], [90] and Particle Swarm Optimization (PSO) [91], [92]. For instance, Jin, Yichao, et al. [89] proposed an adaptive Intelligent Task Allocation Scheme (ITAS) based on a genetic algorithm. A hybrid fitness function is employed to increase the lifetime of multi-hop wireless networks through balancing the workload (i.e. independent tasks) among several nodes. The proposed scheme takes care of the application deadlines but the genetic algorithm may get stuck in a local optimum. Yang, Jun, et al. [91] proposed a Modified version of the Binary Particle Swarm Optimization approach (MBPSO-based) to be used in the task scheduling problem. The particles of MBPSO are identified to represent potential task scheduling instances. Makespan, energy consumption and workload balancing are considered as factors to design a hybrid fitness function that is used to make a trade-off among these factors and reach the optimized task scheduling solution. The precedence and the connectivity between nodes are taken as constraints. Although the feasibility and the significant simulation results for the proposed approach, deadlines have not been ensured which is a critical aspect for real-time applications.

Some researchers focus on the network lifetime through energy-aware task scheduling solutions. For instance, the authors in [93] proposed an algorithm named Distributed Optimal Online Task Allocation (DOOTA), which is applied for cluster-based WSNs to consider the consumed energy during processing, sensing, connecting and sleeping times. The DOOTA algorithm aims to save the energy of the collaborative nodes and the experimental results show that the proposed algorithm increases the network lifetime and decreases the algorithm run time compared with other off-line task scheduling methods. It works well for complex applications because it does not require prior information for all network parameters, but it is unsuitable for real-time applications because it ignores task deadlines. Neamatollahi, Peyman, et al. [94] applied a clustering protocol, which is based on a dynamic task scheduling process to minimize the clustering overhead and extend the network lifetime in WSNs. Therefore, an energy-saving policy is presented in the Dynamic Hyper Round Policy (DHRP) algorithm. In a similar way to DOOTA, DHRP significantly extends the network lifetime, but it cannot achieve deadline constraints due to its distributed nature.

Dai, Liang, et al. [95] aimed to reduce the task completion time and fully utilize the network resources. Therefore, a Divisible Task scheduling Algorithm for Wireless sensor networks (DTAW) and a model to find the optimized task distribution among actors are presented in clustered WSNs. The authors in [95] used a strategy to split the original load into a number of chunks and distribute these chunks to clusters in a specific order. Similarly, an Optimal Task Scheduling Algorithm (OTSA-WSN) in a clustered WSN is presented in [96]. OTSA-WSN is proposed for clustered networks and based on divisible load theory. The proposed algorithm consists of two phases: intra and inter-cluster task scheduling. OTSA-WSN removes the idle gaps and the induced collisions as a result of two data transmissions. Thus, avoiding the effect of idle and signal interference leads to minimized task completion time and improved network resource utilization. All mentioned work considers either the completion time and/or the consumed energy to schedule independent tasks.

Xie, Tao, and Xiao Qin. [97] proposed a scheme called Balanced Energy-Aware Task Allocation (BEATA) for heterogeneous embedded systems. BEATA makes a trade-off between the schedule length and the energy depletion by an energy-delay tunable task scheduling strategy. The work in [97] does not consider workload balancing. Thus, some nodes that are intensively used are depleted early. In contrast, work in [89], [93]–[96] can well balance the task scheduling on collaborative hosts. In addition, the authors in [97] do not consider the task execution deadlines, therefore it is not appropriate for real-time systems.

For faulty real-world RTWNs, Gong, Tao, et al. [98] suggested a reliable task scheduling framework to manage the extra data traffic brought on by disturbances. The introduced Reliable Dynamic Packet Scheduling (RD-PaS) improves the QoS (i.e. the reliability of the data delivery) in terms of task scheduling and message traffic scheduling models. RD-PaS guarantees timely and reliable delivery for critical messages while minimizes the reliability for less important messages in the presence of disturbances. However, the RD-PaS framework does not provide multipath routing techniques. A Reliability Maximization with Energy Constraint (RMEC) algorithm is developed by the authors in [99]. RMEC is applied on dependent tasks by using a blend of three phases: frequency selection, processor assignment and task priority establishment while adopting the DVS technique. RMEC makes a balance between the reliability and the energy consumption in a heterogeneous computing platform of a cluster. However, the aforementioned algorithms consider fault-free wireless systems, while in practice wireless technology encounters different types of failures. Hence, fault tolerance and reliability of RTWNs are critical issues and require a corresponding task scheduling process. Therefore, several work was proposed to investigate different aspects of the fault-tolerant task scheduling problem as shown in the next subsection.

2.2.5 Related Work on the Fault-tolerant Task Scheduling Schemes

Another thread of work in the literature advances the state of the art by providing a variety of fault-tolerant task scheduling algorithms. Fault tolerance aims at improving the reliability, safety and availability of a system.

Many task scheduling algorithms were proposed to support fault tolerance in RTWNs. The authors in [92] proposed a real-time Fault-tolerant Task Allocation Algorithm (FTAOA) to find a solution in the presence of node failures. FTAOA adopts a primary/backup technique to tolerate failed nodes by applying the Discrete Particle Swarm Optimization (DPSO) algorithm. The DPSO-based algorithm considers the consumed energy and guarantees that each task is executed before its deadline. However, this work does not consider the precedence restrictions and the communication cost is also neglected. Marshall, Francis Franklin, et al. [100] suggested a modified Real-time Fault-tolerant Task Allocation Scheme (mRF-TAS) for WSNs, which uses active replication to address the fault tolerance and the task processing time. The proposed scheme needs to be experimentally evaluated and compared to other related algorithms. Moreover, it needs to be extended to minimize the energy consumption in WSNs. These studies are concerned with node failures during the fault-tolerant task scheduling process, whereas this thesis focuses also on fault tolerance in case of a failure in the communication system, which is not covered in the mentioned work.

The work in [101] studies the capability of tolerating transient faults by partitioning periodic tasks on homogeneous multi-core platforms. The task quantization exploits the relations and recovery costs under Rate Monotonic Scheduling (RMS) policy [102]. A metric named “compatibility index” is used to measure how much a group of tasks is compatible when they are scheduled on the same core. Based on this metric, two partitioning schemes are proposed: Compatibility Aware Task Partition (CATP) and Group-wise Compatibility Aware Task Partition (G-CATP). CATP schedules just one task at a time to the most compatible core, while G-CATP looks for the most compatible set of tasks, then it assigns them to a specific core. This work needs more investigation in the partitioning decisions. For instance, the deadline parameter can be added to the “compatibility index” metric. In addition, there should be a trade-off between the execution speed and task compatibility. For example, a task is executed faster in a core but it is more compatible with a task group in another core. The precedence constraint is also not considered yet. Some search focuses mainly on high energy efficiency besides tolerating transient faults. For instance, the authors of the work in [103] aim to minimize the consumed energy by improving task scheduling algorithms. The improved algorithms show significant results in energy savings and tolerate multiple transient faults under the preemptive Earliest Deadline First (EDF) policy. Similarly, Wei, Tongquan, et al. [104] presented quasi-static task scheduling algorithms, which are used to investigate the fault tolerance and DVS technique. The proposed algorithms contain off-line and on-line components, where the off-line components are used to enable the on-line ones to save energy by using dynamic slack. Han, Qiushi, et al. [105] introduced an algorithm that ensures fault tolerance for hard real-time systems. The proposed work determines checkpoints that ensure a successful task scheduling process under the worst-case scenarios (i.e. multi faults on a single core). The task scheduling, speed assignment and checkpoint configuration play crucial factors to design a system with high efficiency in saving energy and fault tolerance. Guo, Yifeng, et al. [106] defined a generalized standby-sparing technique that can tolerate permanent and transient faults. It partitions cores into primary and secondary groups, where the primary group contains the main tasks and the secondary group contains the backup tasks. All tasks are executed under EDF and Earliest-Deadline-Late (EDL) [107] policies. Task partitioning into two core groups aims to improve energy savings and minimize overlapping executions.

2.3 Research Gap

The requirements of hybrid systems as identified in Section 2.1 are only partially satisfied in prior work.

Precise time synchronization in a hybrid network must take into account all factors that cause synchronization errors. For instance, research addressed problems like clock drift, asymmetric delays and precise timestamping. Our thesis provides a contribution beyond the state of the art by improving the IEEE 802.1AS protocol to deal with all these problems. Moreover, the algorithm proposed in this thesis skips the outlier path delays as a result of variable surrounding conditions (e.g. traffic congestion and link loss in dynamic topologies) during the synchronization process in a hybrid TSN system.

TMS, R-TMS and OR-TMS are presented in this thesis to deal with the challenges related to the deadlines (i.e. bounded latencies), message scheduling and constraints of real-time applications during the task and message scheduling process. Each of these three algorithms shares the following phases: 1) Building a topological order for the cooperating tasks depending on specific criteria to establish the task priorities. 2) Avoiding the negative impact of signal interference, which causes increased packet losses and retransmissions, by applying a model named the physical interference model. This model is used to schedule TT messages in predefined time slots. 3) Considering deadlines, precedence and period constraints, the transmission times of messages are planned and the tasks are scheduled to the distributed wireless hosts.

In TMS, the task and message is based on a heuristic approach using list scheduling. The main purpose of TMS is to schedule each task to the most suitable host. The host selection is performed according to the minimum start time. The start time of a host mainly depends on the time of arrival of all incoming messages when scheduling them in their time slots. The message scheduling process considers the essential constraints for TT real-time applications including the period and the precedence constraints. In the end, the solution obtained by TMS should satisfy all task deadlines.

R-TMS and OR-TMS algorithms are built on top of TMS with all constraints inherited. Moreover, limited energy and fault tolerance requirements are taken into account. The proposed fault-tolerant algorithms maximize the reliability by sending multiple copies of messages through redundant routes. Our proposed algorithms consider three non-trivial subproblems, namely, energy-saving, minimizing task completion time and failure rate in heterogeneous wireless nodes as a combinational multi-objective optimization problem. OR-TMS is based on the bio-inspired PSO algorithm that finds a solution with high efficiency and less computational complexity compared to our heuristic methods (i.e. R-TMS and TMS).

2.3.1 Comparison of the Extended IEEE 802.1AS that Addressed the Time Synchronization with Prior Work

Table 2.1 shows a comparison of the extended 802.1AS and prior work that aimed at improving the time synchronization. The related work aims to improve the time synchronization by dealing with the asymmetric delays, clock drift and time-stamping in real-time wireless networks. The extended protocol addresses all the mentioned aspects which in turn reduce the synchronization error between the grandmaster and slave clocks. This is important to avoid task scheduling errors for time-critical applications. In addition, the table shows that the extended protocol allows to provide the features of TSN protocols, which have proven effective in Ethernet networks.

TABLE 2.1: Comparison of the extended IEEE 802.1AS with prior work that addressed the time synchronization process.

Algorithm	**Asymmetric delays mitigation**	**Clock drift handling**	**Precise time-stamping**	**Applied in TSN based systems**
Reinhard Exel [55]			X	
Buevich, Maxim, et al. [65]		X	X	
Chen, Jian, et al. [66]			X	
Chen, Wu, et al. [54]	X		X	
Sungwon Lee [19]	X			
Lv, Shuai, et al. [227]	X			
von Zengen, Georg, et al. [57]		X	X	
Elsts, Atis, et al. [58]		X	X	
Shrestha, Deep, et al. [60]		X	X	
Cho, Hyuntae, et al. [61]		X		
Wu, Jie, et al. [67]		X		
Wang, Zhaowei, et al. [68]		X		
Lam, Duc Khai, et al. [69]	X	X	X	
H.Baniabdelghany, et al. [27]	X	X	X	X

2.3.2 Comparison of TMS, R-TMS and OR-TMS that Address the Task and Message Scheduling with Prior Work

This section outlines the major challenges and requirements considered for task and message scheduling in hybrid real-time systems and compares the proposed algorithms (TMS, R-TMS, and OR-TMS) with prior work.

Table 2.2 shows the contributions of the proposed algorithms to meet the requirements and challenges. We note that some research is limited to meeting the requirements of an accurate timing system, including those that aim to schedule tasks regardless of whether the system is exposed to a potential failures. Other work is limited to finding solutions to conserve the consumed energy, while there are authors who study only how to schedule messages and avoid interference in signals using a specific timing for each sender.

To be more specific, Table 2.3 compares the proposed algorithms separately with extra prior works and more challenges. For instance, researchers in [93]–[96] scheduled independent tasks while taking into account the saving energy and workload balancing aspects to meet the task deadlines. Regardless of how much energy was consumed, the authors in [100], [101] minimized the total failure rates. The works in [83], [84], [98], [228] aimed to fulfil the task deadlines by using routing approaches to conserve communication costs. The authors in [92] neglected the communication costs because of no message transmission between the independent tasks. Energy-saving and minimizing of the makespan were considered in [89], [91], [97], [99] without regard to message scheduling or supporting fault-tolerant message delivery. The authors in [228] considered the task deadlines, precedence and period constraints to found task scheduling solutions in TSN-enabled Ethernet networks. In contrast, the TMS [28]algorithm includes many aspects that are mentioned including the deadline, precedence and period constraints. In addition, it uses routing approaches and scheduling the messages into fixed time slots to conserve the communication cost in fault-free Wireless TSN networks. R-TMS [29] is built on top of TMS to support fault-tolerant message delivery and consider the energy-saving and failure rate during the task and message scheduling process. Due to using the DPSO algorithm, OR-TMS [29] produces better solutions than R-TMS. Moreover, OR-TMS has more advantages because of the aspect of workload balancing.

TABLE 2.2: Comparison of the thesis algorithms with prior work.

Algorithm	Bounded latency	Energy saving	Message scheduling	Task scheduling	Fault-tolerant task scheduling	Accurate global notion of time
Zimmerling, Marco, et al. [34]	X	X	X			
Brummet, Ryan, et al. [39]	X		X			
Zhao, Miao, et al. [40]	X	X		X		
Yu, Wanli, et al. [44]		X	X			
Chipara, Octav, et al. [48]	X		X			
Jamthe, Anagha, et al. [49]		X	X			
Fateh, B. and Govindarasu, M. [51]	X	X	X	X		
Chen, Wu, et al. [54]	X	X				X
Lee, Sungwon [19]	X					X
von Zengen, Georg, et al. [57]						X
Buevich, Maxim, et al. [65]		X				X
Yang, Jun, et al. [91]	X	X		X		
Yu, Wanli, et al. [93]		X		X		
Dai, Liang, et al. [95]	X	X		X		
Marshall, Francis Franklin, et al. [100]	X			X	X	
Guo, Wenzhong, et al. [92]	X	X		X	X	
Han, Qiushi, et al. [101]	X			X	X	
The thesis algorithms	X	X	X	X	X	X

TABLE 2.3: Comparison of TMS, R-TMS and OR-TMS algorithms that address the task and message scheduling with extra prior work.

Algorithm	**Precedence and period constraints**	**Routing approaches to conserve comm-costs**	**Fulfilling task deadlines**	**Energy saving**	**Minimizing of failure rate**	**Minimizing of makespan**	**Time-slotted message scheduling**	**Fault tolerant message delivery**	**Workload balancing**
Jin, Yichao, et al. [89]		X	X	X		X			X
Yu, Wanli, et al. [93]				X					X
Neamatollahi, Peyman, et al. [94]		X		X			X		X
Dai, Liang, et al. [95]				X		X			X
Dai, Liang, et al. [96]				X		X			X
Xie, Tao, and Xiao Qin. [97]	X			X		X			
Gong, Tao, et al. [98]		X	X			X	X		
Zhang, Tianyu, et al. [84]		X	X			X	X		
Zhang, Tianyu, et al. [83]		X	X			X	X		
P. Maryam, and R. Obermaisser [228]	X	X	X			X			
Zhang, Longxin, et al. [99]	X			X	X	X			
Yang, Jun, et al. [91]	X			O[a]		O[a]			O[a]
Guo, Wenzhong, et al. [92]			X	O[a]	O[a]	O[a]			O[a]
Marshall, Francis Franklin, et al. [100]			X		X	X			
Han, Qiushi, et al. [101]			X		X	X			
H.Baniabdelghany, et al. [28]	X	X	X			X	X		
H.Baniabdelghany, et al. [29]	X	X	X	C[b]	C[b]	C[b]	X	X	
H.Baniabdelghany, et al. [30]	X	X	X	O[a]	O[a]	O[a]	X	X	X

[a] An object in a multi-objective cost function, which is used to find a global task and message scheduling solution.
[b] The algorithm contributes during the scheduling of every task gradually on an available host until all tasks in a sorted list are completed.

Chapter 3

Background Theory

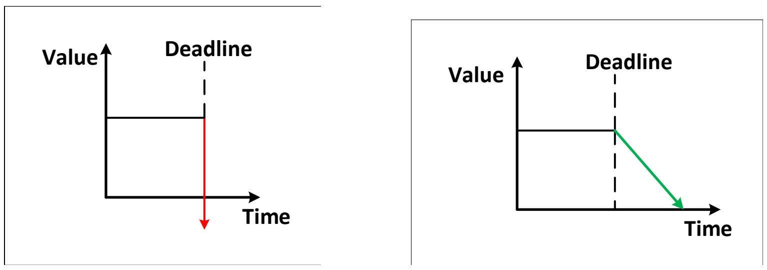
This chapter describes background theories that are relevant to understand the rest of our thesis. Section 3.1 gives a brief overview of real-time systems that are widely deployed in different domains to provide a wide range of services. Section 3.2 explains the concept of the distributed real-time systems. Time-triggered and event-triggered systems are introduced in Section 3.3. Section 3.4 discusses different aspects of dependability such as dependability threats and the means to obtain dependability. Fault-tolerance techniques are discussed in Section 3.5 to preserve the correctness of the system in the event of failures. Section 3.6 compares timing protocols (i.e. Precision Time Protocol (PTP) and Network Time Protocol (NTP)) and it demonstrates the limitations of the NTP protocol. The standard 802.1AS for clock synchronization is described in Section 3.7. Section 3.8 discusses the link scheduling modes to avoid interference due to collisions and to minimize the end-to-end delay. Section 3.9 gives an overview of the 802.11 protocol and its limitations in real-time systems. Section 3.10 analyzes real-time communication protocols for wireless systems, while well-known real-time communication protocols in wired systems are discussed in Section 3.11.

3.1 Real-time Systems

A real-time system is defined as a time-bounded system intended to be used in applications where the correctness of the outputs depends mainly on the logical order of computations and the timing at which the computations are implemented [108]. In other words, the correctness of the results in real-time systems depends basically on the predictability of the task execution time and the message communication time. Each task and each message has to finish before its deadline. This timeliness condition also demands that the real-time systems should not be degraded in case of peak-load traffic as a result of overlapping zones of two senders. Moreover, to ensure reliability, these systems should be designed to be fault-tolerant. For example, the air traffic control real-time system, where the timeliness is critical to coordinate aircraft during landing and take-off. The control process becomes more difficult when wireless networks are involved, where timeliness and reliability strongly depend on the surrounding factors.

In general, real-time systems can be classified into two groups: soft and hard real-time systems. For the purpose of preventing dangers to the user or the environment, a hard real-time system [109] is any software or hardware that must function within strict, predetermined deadlines, as shown in Figure 3.1a. The application may cause disasters and massive system failure if it does not complete its tasks within a specified time. Many examples of hard real-time systems are present in our daily life such as robots, automotive and railway systems, nuclear power plants, etc. Consequently, a timing failure can lead to intolerable costs in terms of human lives or major economic losses.

A soft real-time system [110] tolerates variations in delivery time and deadline violations as shown in Figure 3.1b. Such a system degrades the application performance but it does not endanger the user or the environment. For example, multimedia applications are considered as soft real-time systems because deadline violations degrade the quality but do result in safety risks. Banking applications, reservation systems, and entertainment systems can also work properly under variable delays. In general, all components involved in a real-time system must support the real-time requirements including for example the operating systems such as VxWorks [111] and MaRTE OS [112] which have the capability to guarantee timing requirements in task scheduling.



(A) Hard deadline (B) Soft deadline

FIGURE 3.1: Timeliness for hard and soft real-time systems [113].

3.2 Distributed Real-Time Systems

A Distributed Real-Time System (DRTS), as shown in Figure 3.2, contains multiple computing nodes (e.g. sensors) interconnected by a real-time communication network. The nodes are physically distributed along the real-time system rather than communicating within a single processing unit. This is due to a number of reasons, first, the spatial expansion of the application may demand a distributed system. In other words, the computation of a real-time system is based on gathering data from spatially separated and heterogeneous nodes, the results of the processing are sent to central processing and monitoring nodes that are located at different places. Second, distributed applications may require special hardware specifications. Therefore, it would be difficult to connect them to the same unit. Third, fault tolerance and workload balancing must be structured in such a way that faults can be contained, detected, and masked. This guarantees that even in the event of failures, the system will continue to function properly.

In distributed systems, insufficient precision or unexpected communication delays lead to some nodes execute tasks at different sequences. Thus, some nodes may have an incorrect and inconsistent view of the situation [114]. Therefore, the typical distributed real-time system not only preserves the task casual order but also ensures that the tasks are executed before their deadlines [4]. This means that message communication delays have to be accurately predicted. Based on that, real-time protocols that offer delay guarantees and deterministic access to the medium are required.

Most protocols are designed to distribute tasks to different nodes either to maximize data rate or to conserve network resources without service connectivity after the distribution has been established. For this reason, there are some dedicated communication protocols that are implemented using specialized communication hardware.

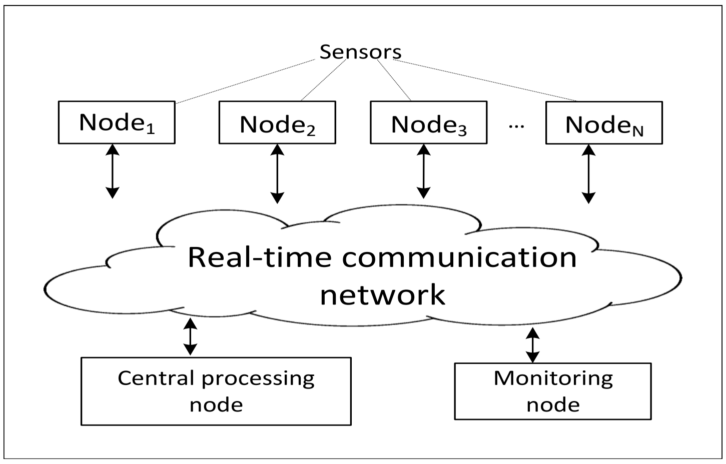


FIGURE 3.2: An example of distributed real-time system.

3.3 Time-Triggered and Event-Triggered Embedded Systems

The processing and communication activity in real-time embedded systems can be classified into two categories depending on when the tasks and messages are triggered [115]–[117]: In Time-Triggered (TT) systems, the tasks and messages are initiated periodically at predefined point of times. In Event-Triggered (ET) systems, the tasks and messages are initiated in response to significant sporadic events. On the one hand, the task scheduling in most ET systems is associated with task pre-emption, meaning that a high priority task can interrupt the execution of a lower priority task, with the intention of resuming its execution at a later time. On the other hand, task scheduling in most TT systems is associated with task co-operation, meaning that the current task continues executing, then it relinquishes the control to another task co-operatively. Therefore, ET systems excel in flexibility, whereas TT systems excel in temporal predictability.

In TT systems, each task and each message is triggered when the global time reaches a specific point. The periodic clock signal is widely used as a trigger to perform TT tasks in communication and control systems. Therefore, the precise synchronization for all nodes that perform TT periodic tasks plays a critical role in TT embedded systems. In ET systems, tasks and messages are triggered when a significant event occurs such as a progress event (e.g. termination of a preceding task) or an external signal that comes from the surrounding environment (e.g. an interrupt). There are two types of events: predictable events that occur at certain times determined by pre-defined laws and chance events that cannot be predicted deterministically. The latter events are unplanned and cannot be described by specific laws in contrast to the predictable ones. Compared to the ET design, the TT design is simpler, having less overhead, resource sharing is more straightforward, and testing is simplified.

For the integration of TT and ET communication systems, many contention-based schemes have been proposed. For example, Figure 3.3 illustrates three different contention-based schemes: 1) In the non-preemptive scheme, if a TT periodic message overlaps with an ET message, the ET is not pre-empted by the TT message and the transmission of the ET message continues until completion. Thereby, the TT message transmission may be delayed at worst by one ET message transmission. 2) The preemptive scheme allows interfering TT message to pre-empt the currently transmitted ET message. Afterwards, the ET message is resumed to be transmitted at the time of the TT message completion. 3) The time-reserved scheme reserves an interval of time before each TT message to avoid the potential overlapping with expected ET messages. However, this scheme is not suitable for systems without prior knowledge about ET applications due to depending on the maximum size of an ET message. Therefore, sending large size ET messages may severely affect the messages latencies and the utilization of the network.

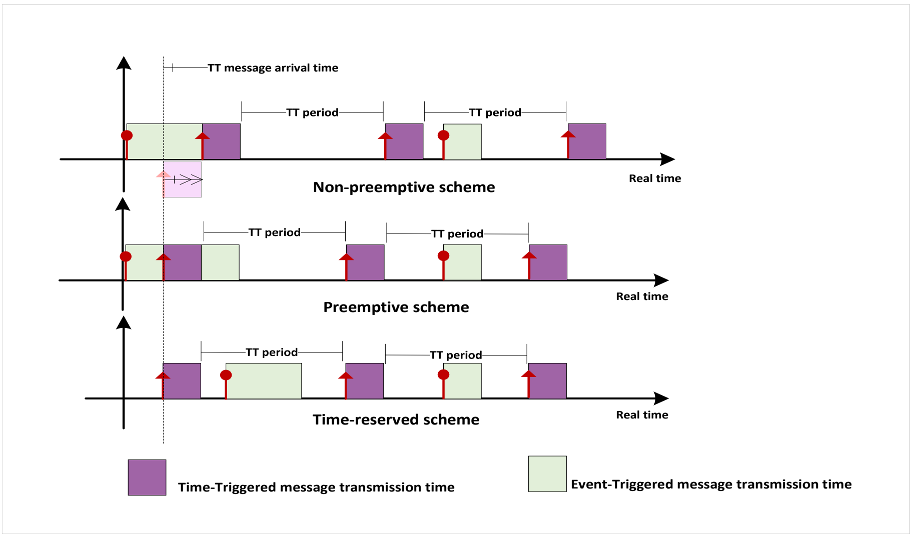


FIGURE 3.3: Contention-based schemes for TT and ET message transmissions.

3.4 Dependability

Dependability of a real-time system is defined as the ability to deliver a trusted service within a period of time [118]. It usually consists of a set of attributes including **availability**, **reliability**, **security**, **safety**, and **maintainability**. **Availability** is also known as the uptime and it defines whether a correct service can be used or accessed within a specific period of time. Distributed real-time systems typically have a big advantage compared with non-distributed systems in terms of service availability. **Reliability** is the likelihood that a service will carry out its intended purpose successfully within a specific time frame or continue to function without failure in a specific environment. For this reason, real-time systems develop services as a series of fault-tolerant functions that operate correctly regardless of faults [119]. **Security** is the ability of the system to protect its data. There are two types of data: public data can be used by any user and private data that should only be accessed by authorized users with defined privileges. **Safety** means that the computing system is free from harms or dangers to human life and property. **Maintainability** measures the ability of the computing system to be repaired or modified to specified conditions.

The concepts of dependability also distinguish different types of threats and the means by which dependability is attained.

3.4.1 Threats: Faults, Errors and Failures

A **fault** is described as a physical flaw or defect in the system's hardware or a bug in the system's software. On one hand, the fault is active when it causes an error, otherwise, it is dormant. On the other hand, it is not necessary that the presence of a fault leads to a failure. For instance, a system may contain a fault but the conditions may never activate this fault and lead to a failure. The faults are classified into three categories: *transient*, *intermittent*, and *permanent*.

- A *transient* fault is defined as a fault that temporarily affects network connectivity or involves the loss of computational components and services for a short period. Normally, transient faults disappear without a repair action.
- An *intermittent* fault is a malfunction of a system or software that happens at irregular intervals. The system or the software is working normally outside these intervals. Moreover, it masquerades as a transient fault but it returns unpredictably. Therefore, this type of fault is difficult to trace and deal with.
- A *permanent* fault is a malfunction that does not go away when certain conditions exist and needs a repair action to be removed, whereas intermittent faults are permanent faults that sporadically lead to failures (e.g. loose contact).

An **error** is a deviation between the expected state of a computing system and its actual state. It occurs when a part of the system enters into an unpredicted state due to activation of a fault. If the error reaches a service of the system and changes its behaviour, the error causes a subsequent failure. Special observation mechanisms such as log files or error messages are used to indicate the presence of the errors. Undetected errors are called latent errors.

A **failure** occurs when the system provides a behaviour that deviates from the specification. Therefore, fault tolerance techniques are used to avoid the effects of failures and to preserve the overall operation of the system. A fault-tolerant real-time system normally includes subsystems called Fault Containment Region (FCR) [120]. The fault must be discovered within the FCR to mitigate its effects on the services of other regions. To achieve this, error containment coverage is provided within each FCR. In a distributed system, nodes and communication links are usually considered FCRs if they are completely independent while designing the system.

In general, the way that the computing system and communication can fail is called the failure mode. The following failure modes are distinguished in practice:

- *Omission failures*: the nodes or participants keep working, but the connection between at least two of them is lost or at least one of the messages is dropped or delivered with unexpected delay (i.e. channel failure). The omission failure can also occur when a process in the system is crashed (i.e. process failure).
- *Crash failures*: a node shuts down or halts unpredictably. The reason relates to errors in the environment, the output result of an application, or simply a loss of power.
- *Simultaneous or repeated failures*: these are correlated failures, in which several nodes go down simultaneously or one node keeps failing.
- *Fail-stop failures*: the computing system with the fail-stop failure stops working, then the system does not return any result or the result shows that the system is failed. Depending on the time model, we can determine the occurrence of this type of failure. For example, it can be detected in time-triggered systems, while it is difficult to be detected in event-triggered systems.

Figure 3.4 illustrates the chain of threats to dependability that shows a causal relationship between faults, errors, and failures. The arrows in the chain start from the activation of a fault, followed by the propagation of the produced error until the induced failure.

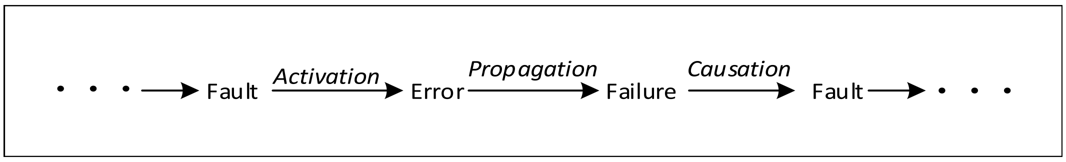


FIGURE 3.4: Fault-Error-Failure chain of threats to dependability [118].

3.4.2 The Means to Obtain Dependability

To increase the dependability of a system and to break the chains that threaten dependability, a set of four techniques are combined [118] including **fault prevention** to prevent fault occurrence or the introduction of faults, **fault tolerance** to provide correct service in the presence of faults, **fault removal** to reduce the number of existing faults, and **fault forecasting** to estimate the current number of faults, future incidence, and the possible consequences of faults.

Fault prevention is a proactive technique used to identify the areas where the faults could occur and to close the gaps. Fault prevention is accomplished during the manufacturing phase (i.e. the design process) of software applications and hardware components. To be achieved, several development methodologies and techniques are used.

Fault tolerance aims to keep the system correctly working in the event of failures, although it may degrade the performance of the system. For example, two engines of an airplane represent a fault-tolerant system. If one of them fails, the other remains operational to let the airplane continue to fly. In addition, fault-tolerant mechanisms need to be evaluated with respect to the dependability attributes (e.g, reliability and availability, ease of design and maintenance, ease to understand, data security).

Fault tolerance aims at recovering the system from a state that includes active errors to a state without errors. Implementation of error handling and fault management is necessary to complete the recovery process. Error management either does a roll-forward or puts the system back in a safe condition. Fault handling aims to prevent the current fault to be activated again. To achieve that, firstly it identifies and detects the factors causing errors, secondly, the faulty components are isolated and prevented to participate in the service delivery, thirdly switching into non failed components, and finally, the system will be re-initialized.

Fault removal is divided into two phases to remove the faults before putting the system into production. The first phase is fault removal during the development phase, and the second phase is fault removal during the operating phase. Fault removal during the development phase checks if the system adheres to specific verification conditions such as static verification conditions (e.g., data flow analysis) and dynamic verification conditions (e.g., verification testing). If not, the system analyses the faults that prevent satisfying these conditions, and then the required corrections are performed. Fault removal during the operating phase is either corrective maintenance or preventive maintenance. Corrective maintenance is used to remove reported faults that cause errors, while preventive maintenance is used to remove faults before they may cause errors during the live operation.

Fault forecasting is an assessment of the behaviour of the system concerning the occurrence of the fault. The assessment is fulfilled in terms of qualitative and quantitative aspects that are based on classifying failure modes and probabilities of satisfying the attributes of dependability.

3.5 Techniques for Fault Tolerance in Critical Systems

3.5.1 Fault Hypothesis

A system's ability to continue operating in the event that some of its components fail is known as the Fault Tolerance (FT) of the system. It becomes a prerequisite, particularly for life-critical and high-availability systems [121]. Fault-tolerant system are confronted with two types of faults: covered and uncovered faults. The covered faults are addressed during the development phase of the system. In the event of a covered fault occurrence, the fault is guaranteed to be tolerated to prevent any effect on the system functionality. Uncovered faults are those faults that are not contained in the fault hypothesis and there is no provided strategy against them. Therefore. It is necessary to prove that the uncovered fault occurs rarely to avoid a failure which in turn, leads to a failure of the entire system [122].

Research in fault-tolerant techniques includes interdisciplinary work. All possible faults have to be carefully considered especially when the system is being complicated, the fault tolerance is a necessity for high-value systems such as the military, transport systems, and public facilities. The field of research topics in the area of fault tolerance is wide such as software models, hardware reliability, and parallel processing [123]. Therefore, in this section, we will introduce briefly some fault-tolerance techniques.

3.5.2 Error Detection and Correction Codes

During data transmission from a transmitter to a receiver, data packets can suffer from noise that may cause errors in the sequence of the binary bits. Therefore, redundant codes can be implemented in both the transmitter and receiver to ensure the integrity of the transmitted data.

Error detection codes: they append some added bits to the original transmitted packet to detect the errors at the receiver. For example, 1) *Simple parity code* [124]: There are two types of simple parity codes, namely even and odd parity codes. A parity bit is added to the original binary code to make the total number of $1's$ odd or even depending on which parity code is implemented. Thereby, the parity code is useful for detecting a single bit flips in the received packet. 2) *Two-dimensional Parity check* [124]: This method implements the simple parity code for each row of the original data, similarly, the same parity code is implemented for each column. Then all calculated parity bits are appended with the transmitted original packet. 3) *Hamming code* [125]: Up to two erroneous bits can be detected using this method, to do that, multiple parity bits are generated and added at specific positions in the original transmitted packet. At the receiver, a recalculation process is performed to detect the errors and find out the position of the bit errors. 4) *Checksum* [126]: In this method, small segments are derived from the original message and all segments are added to each other using $1's$ complement arithmetic to compute the sum, which is sent to the receiver. Similarly, the receiver adds the received segments to compute its sum, and if the complement of the receiver's sum is identical with the transmitted one, then the data is accepted, otherwise, an error is detected. 5) *Cyclic Redundancy Check* (*CRC*) [127]: It is based on binary division by using a generator polynomial. Firstly, the transmitted packet is divided by a generator polynomial, and then the remainder is added to the original packet. At the receiver, the received packet is also divided with the same generator polynomial. If the remainder of the division is zero then the data is correct, otherwise, an error is detected. An example for a generator polynomial is $x^3 + 1$, which can be represented as 1001. Another example is $x^2 + x + 1$ that represents 0111.

Error correction codes: In addition to error detection, error correction codes are used to correct the detected errors in the received packet at the receiver. For example, *Hamming* *code* has the capability to correct a single bit error in the received packet [125] and *CRC* is an effective correction technique to correct the errors [128] so that the original can be restored.

3.5.3 Damage Confinement

Damage confinement [129] is based on structuring the system to prevent or stop the spreading of the damage caused by failures of some components during the repairing process. In other words, the damage confinement works as a firewall against failures that can affect adversely the efficiency of the system. The modular decomposition provides a static or dynamic damage confinement. In the static confinement, data traffic is imposed to be forwarded through well-defined paths. In the dynamic confinement, atomic actions are used to keep the system in a consistent state.

3.5.4 Error Recovery

There are two approaches to restore (i.e. recover) the system from an erroneous state to an error-free state [130]:

- Forward Error Recovery (FER): This approach is usually used in non-stop applications to remove and correct the current state and then allow the system to move forward. This is possible when the errors and the damage caused by a fault can be accurately predicted and assessed (i.e. damage assessment).
- Backward Error Recovery (BER): On the contrary, when the errors and the damage caused by a fault cannot be accurately predicted and assessed, then it is not possible to remove the errors from the current state. Therefore to avoid the errors, the current state is reverted to a previous stable and error-free state (i.e. checkpoint). BER is frequently used due to its simplicity and it is independent of unanticipated faults. But on the other side, it requires more overhead for the recovery process and there is no guarantee that all components are recoverable or that the errors will not occur again.

3.5.5 Fault Treatment

The fault treatment [131] prevents the error from reoccurring. For example, restarting the system then moves the system to an error-free state in case of a transient fault. Further examples of fault treatment are patching the system with a new error-free version, online-intervention while the system is executing or correcting the human behaviour so that the person will not cause the same error again.

3.5.6 Redundancy of Computing Systems

Redundancy enables a system to be protected against damage by extra nodes added to the system. There are different ways to enable the system to continue operating properly and make it more resistant to faults. However, adding extra nodes inevitably increases the complexity of the system. Therefore, it is necessary to avoid redundant nodes that are not required in the system. This section discusses briefly the forms of redundancy.

Hardware Redundancy

In this type of redundancy, two or more hardware nodes are independent and completely equipped with processors, memories, and peripheral nodes [132]. The redundant nodes can cooperate in three ways: 1) Passive technique (i.e. fault-masking): In this simple technique, all hardware nodes work independently, and the output results are compared. If there is a different result in a node, it means a fault occurred in that outlier node. Thereby, no explicit action or fault detection is required in this type of redundancy. 2) Active technique: this technique requires a sequence of operational steps including detection, localization, containment, and recovery to isolate the faulty hardware node. This technique does not prevent faults from producing errors in the system. Compared to the passive technique, the active technique is not expensive because only one hardware node is in the operating mode and all other standby nodes are available if the current operating node fails. In addition, it recognizes a fault and takes an action after producing an error. Hence it is useful in systems that tolerate temporal errors while they are running. 3) Hybrid approach: it is used in fault-masking to prevent spreading the errors and it also detects faults and reconfigures all independent nodes to isolate the faulty one. In the following, we demonstrate some examples for each hardware redundancy technique.

Passive hardware redundancy

Triple Modularity Redundancy (TMR) [133]: It is considered one of the most known passive fault-tolerance techniques. Figure 3.5 illustrates the structure of TMR, where a hardware node (N) is replicated into three nodes that operate simultaneously in parallel. The results of the three nodes are inserted into a voter (V). The voter then presents the common result as an output value.

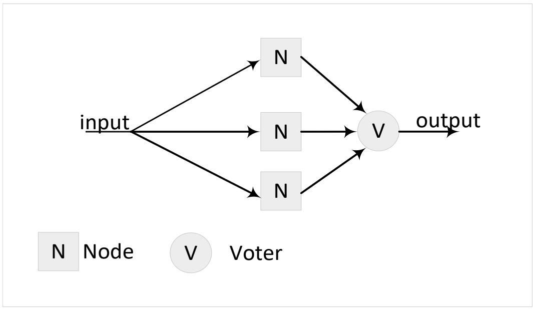


FIGURE 3.5: Structure of TMR FT technique [133].

The advantage of TMR is the ability of masking a single failure in one of the three operational nodes. As mentioned, TMR is a passive technique that does not require special actions such as error recovery. TMR is more suitable for transient faults since it does not have the ability to remove or isolate the failed node. In addition, it does not handle two failed nodes. Therefore, if one node fails, it is necessary to be sure that the other two remaining nodes work properly.

N-Modular Redundancy (NMR) [134]: It is a generic FT technique preferred to be used in highly critical systems. The concept of NMR is based on the assumption that at least $\frac{N+1}{2}$ nodes are working correctly at any time, where N is an odd number denoting the total number of replicated nodes. Thus, NMR will tolerate the failure of up to $\frac{N-1}{2}$ nodes. For example, if NMR replicates 5 nodes that are operating at the same time, then at least 3 nodes should work properly to mask at most 2 failed nodes. In contrast to TMR, NMR enhances reliability and guarantees continuity for a system that contains more than one failed node.

Active hardware redundancy

As mentioned the passive (i.e. static) redundancy techniques are more expensive, where, three replicated nodes mask one failed node, five replicated nodes mask two failed nodes, and so on. In contrast to passive redundancy, in active (i.e. dynamic) FT techniques, only two replicated nodes are required to mask one failed node and three nodes to mask two failed nodes. To illustrate how active techniques work, some active techniques are addressed in this section.

Standby sparing [135]: To demonstrate the concept of standby sparing, Figure 3.6 shows a switch that is controlled by a fault detection scheme to compare the outputs of each node with an error report that is issued from its corresponding error detection circuitry. This comparison makes the switch aware if a node is working properly or it is a faulty node. In case of a faulty node, the output is taken from another spare (i.e. standby) node. Standby sparing is divided into two types, hot and cold standby sparing. In hot standby sparing, the spare node is operating in synchrony with its online node and is prepared to take over at any time. This type is fast and does not need power-up time, in contrast, it affects adversely the power consumption. In cold standby sparing, the spare is powered-up only when it is required to work instead of the faulty node. It requires more time than the hot sparing to bring the spare into the operational state with less power consumption.

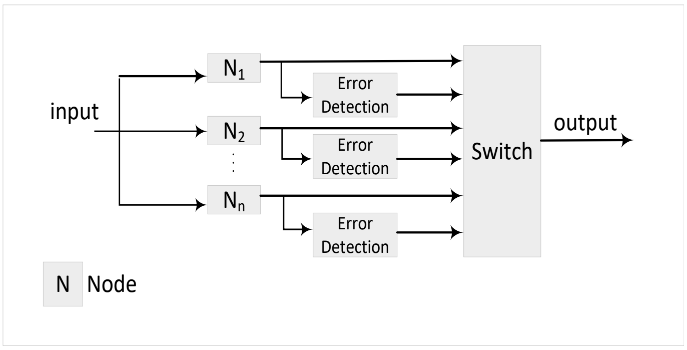


FIGURE 3.6: Structure of standby sparing FT technique [136].

Duplication with comparison [137]: A comparison of the results is made between two identical nodes which perform the same computation. The duplication concept can detect faults but it cannot determine which node is faulty. This technique causes problems. For instance, the input data may be faulty, which means that the same error is inserted into the duplicated nodes and then they produce the same erroneous result. The comparator (C) may not perform the exact comparison due to problems in matching or synchronization. In addition, a failed comparator causes the entire technique to stop operating.

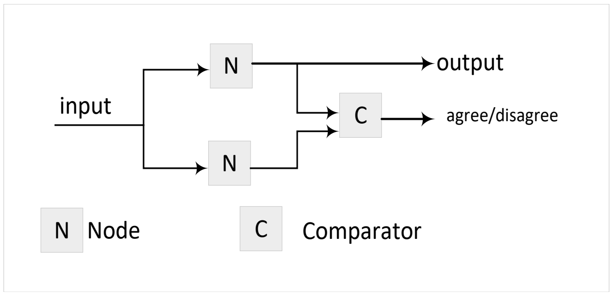


FIGURE 3.7: Structure of duplication with comparison FT technique [137].

Pair and spare [138]: It combines standby sparing and duplication with comparison. The concept uses N replicated nodes to N-to-2 switch rather than N-to-1 switch. The reason is that the comparator receives two results to verify if they are the same (agree) or not (disagree). As long as the output results match, the spares are not required. If the output results disagree, the error reports will be used to locate the faulty node and then the result will be taken from another spare. In this technique, the error detection is done at two levels, the first level is the standby (i.e. at the node level) and the second level is done by the comparator at the output level. Pair and spare works correctly if the faults are independent. Correlation between faults could cause a failure of the entire technique.

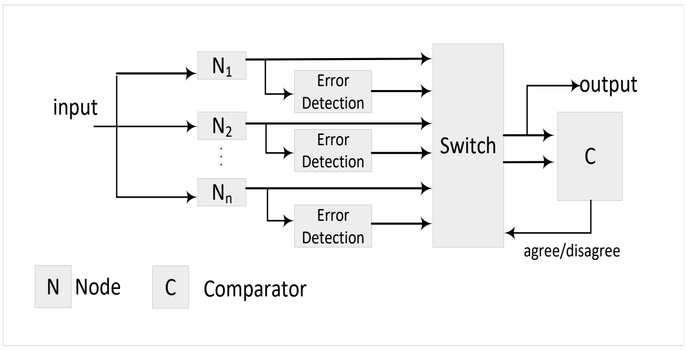


FIGURE 3.8: Structure of pair and spare FT technique [138].

Hybrid hardware redundancy

This type of redundancy technique combines passive and active redundancy. It prevents spreading of errors in a system by using fault-masking and it also detects the faults and reconfigures the system to isolate and remove the faulty nodes. Therefore, the hybrid approaches are expensive, but with more fault-tolerance capability. Some types of hybrid redundancy are shown below:

Self-purging redundancy [139]: In this technique, the failed nodes are purged by using a voter threshold. The most important feature of this technique is that every node has the ability to remove itself from the active nodes if it discovers itself as a faulty node. Thus, it is preferred for time-critical systems because maintenance personnel can isolate the faulty modules and replace them without causing interruptions during the operational time.

NMR with spares [140]: Figure 3.9 shows the hybrid approach of combining NMR with the standby sparing technique. The technique works as a passive NMR technique until the disagreement detector determines a fault through a voter that compares the individual results from the nodes with the voter's output. The node that disagrees is labelled as a faulty node, removed from the system and replaced by a spare (S) node. NMR requires five nodes to mask two faults, whereas this technique requires only three nodes and one spare to mask the same number of faults.

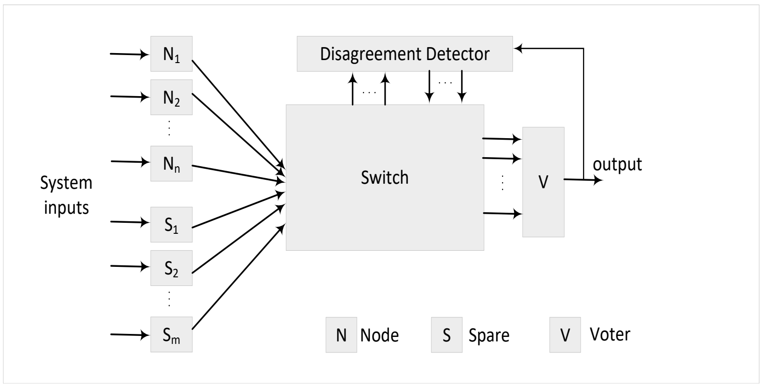


FIGURE 3.9: Structure of NMR with spares FT technique [132].

Triple-duplex [141]: Figure 3.10 shows the hybrid approach of combining TMR with the duplication and comparison technique. Each node in the triple-duplex setup is represented as two duplicates with a comparator (i.e. duplication with comparison). The represented nodes are then inserted into TMR. Thereby, this hybrid approach allows TMR to mask the faults and the duplication with comparison technique serves for the detection of the faults and removes the faulty nodes from the system. The triple-duplex setup can mask two faults.

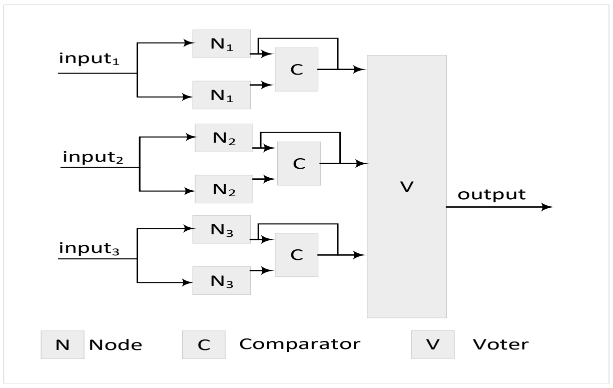


FIGURE 3.10: Structure of triple-duplex FT technique [132].

Software Redundancy

To find and hide software and hardware faults, software redundancy can be used. Hardware faults can be dealt with either by testing the consistency, for instance, executing some instructions in an Arithmetic-Logic Unit (ALU) to compare the output results with precomputed ones that exist in Read-only memory (ROM), or testing the capability, for instance, using a memory test. Software redundancy can also be used to detect software faults. For example, N-Version Programming (NVP) [142] implements the same functionality using different software versions (e.g. different development teams, different languages, different operating systems), and then it compares the results. However, the ambiguous specifications of the versions and correlated faults could make NVP fail.

Time Redundancy

In time redundancy, no extra hardware is required, the same task is executed several times, and then the output results are compared to confirm the integrity of the computations or detecting transient faults.

Information Redundancy

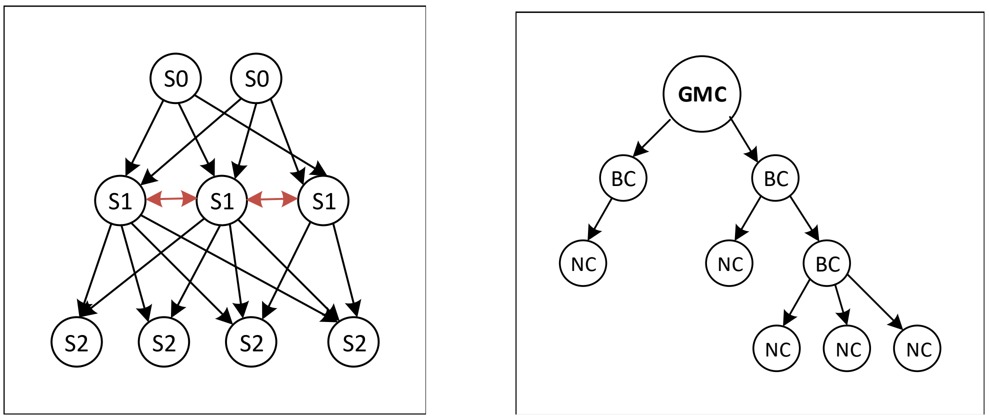
In information redundancy, extra information is required to detect faults. For example, storing the same information in different places (i.e. data backup). Another example are extra codes that are used to detect or correct errors in the transmitted data.

3.6 NTP vs PTP Timing Protocols

Clock synchronization is one of the most crucial services for coordinating distributed access to shared resources, comparing timestamps set up at various locations, and measuring the performance of distributed real-time systems [143]. Due to the effects of variable latencies which exist in any network, the establishment of a global notion of time with a predefined precision by ensuring bounded maximum offsets becomes a challenge. Two prevalent protocols are designed to reduce the impact of time synchronization errors. The first protocol is NTP [12], [144], [145] which is defined by IETF in RFC 5905. It is widely adopted as a networking protocol to synchronize packet-switched networks. The second protocol is PTP [12], [145], [146] which is developed on top of the IEEE 1588-2008 standard [17]–[19]. PTP, in ideal conditions, can synchronize the network nodes with a reference global time with less than one-microsecond synchronization error. It is normally used in local area networks that depend on broadcasting for data transmission. NTP, in particular, is applied in the internet which uses the unicast principle for data transmission with millisecond range synchronization error. For non real-time applications, NTP is typically sufficient. In the case of time-sensitive applications, the PTP protocol becomes a suitable solution.

In principle, NTP and PTP are based on the same procedure. The nodes are arranged in a tree topology, with the reference source clocks at its top. Normally, the source clock is synchronized with an external high-precision global clock (e.g., Global Positioning System (GPS) clock [147]). The nodes below periodically exchange timing packets with their source clocks to adjust their clocks by minimizing the induced offset due to effects like unpredictable delays and changes in temperature.

In NTP, the tree structure consists of layers (i.e. stratum levels), *Stratum 0* (i.e S0) at the top layer contains atomic clocks (e.g. Global Navigation Satellite System (GNSS) [147]). Every main time server in *Stratum 1* (i.e S1) is connected with *Stratum 0* clocks to achieve microsecond-level synchronization, it is also connected with other *Stratum 1* clocks to make fast tests and backups to each other. As shown in Figure 3.11a, *Stratum 1* clocks are clients to *Stratum 0* clocks, *Stratum 2* (i.e S2) clocks are clients to *Stratum 1* clocks. NTP supports up to 15 stratum levels, thus, increasing the number of stratums leads to a higher synchronization error from *Stratum 0* clocks. In PTP, the tree structure contains slaves that are synchronized to their masters. Each slave in the tree may be synchronized by different masters using different paths. The clock at the top is called the Grand-Master Clock (GMC). The clocks that can be slaves to the upper clocks and masters for other salve clocks at the same time are called Boundary Clocks (BCs). The clocks (i.e. leaf clocks) that have one port are called Normal Clocks (NCs). The set of clocks that is synchronized with each other is called "PTP domain" as illustrated in Figure 3.11b.



(A) NTP structure. (B) PTP domain.

FIGURE 3.11: Tree structure of NTP and PTP [145].

3.6.1 Source Selection

The source selection in NTP is different than in PTP [145]. In NTP, the client selects the best sources out of several available sources to synchronize to. The selection is based on sources that have the best characteristics and the shortest paths to the primary servers in *Stratum 0*. The selected sources can be combined to minimize the accumulated error and to give the client a more accurate time. Moreover, NTP rejects the falseticker NTP source that has been identified by an NTP client as providing inaccurate time due to time software failure or configuration mistake. In the presence of three source clocks, NTP can detect one falseticker. With five sources it can detect two falsetickers, and so on. Thereby, the NTP source selection procedure results in a fault-tolerant protocol.

In PTP, nodes identify themselves as slave, passive, or grandmaster clocks by listening to *Announce* messages that are broadcast from all normal clocks in the network. The *Announce* message contains the properties of the clock that sent it. If the normal clock receives an *Announce* message from a better clock, the status of that clock changes to be a slave or passive clock. If a normal clock does not receive *Announce* messages with better characteristics during a period called *Announce-time-outInterval*, the status of the normal clock goes to be a grandmaster clock. The selection process may take several iterations until reaching a stable system, where ultimately there is only one grandmaster clock that periodically sends synchronization messages ($SYNC$) to all slaves to adjust the time of their clocks.

3.6.2 Synchronization Process in NTP

NTP has three modes to measure time offsets [145]. The first mode is the client-server mode which is the most common mode. The client sends its server a client mode message, the server responds to the client with a server mode message. The client gets then four timestamps as a result of exchanging the mode messages: the client mode message records two timestamps t_1 and t_2 of the time of sending it from the client and receiving it at the server, respectively. Similarly, the server mode message records two timestamps t_3 and t_4 of the time of sending it from the server and receiving it at the client, respectively, as seen in Figure 3.12. The offset of the client's clock is the mean value of the delays $(t_2 - t_1)$ and $(t_3 - t_4)$. The assumption is that the request and response delays are symmetric (which they are usually not), thus, the measured offset has an error and leads, in turn, to a synchronization error with the server.

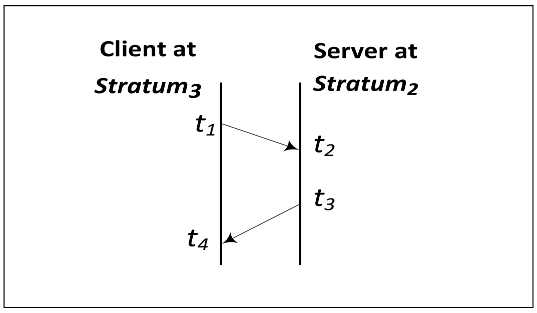


FIGURE 3.12: Client/server mode in NTP [145].

In the second mode which is the symmetric mode, the synchronization is typically used between NTP peers (i.e., servers) to change the server in case one of the NTP peers losses the connection with the upper servers. The main difference compared to the client-server mode is that the synchronization is done in both directions. For example, when $P1$ peer in Figure 3.13 sends a request message to another peer (i.e. $P2$) at the same stratum, $P2$'s response message is also considered as a request message toward $P1$. At the time $P1$ receives the incoming message, it gets all four timestamps required to measure its offset with $P2$. After that, $P1$ responds to the incoming request message with a response message. $P2$, in turn, gets all required timestamps to measure its offset with $P1$.

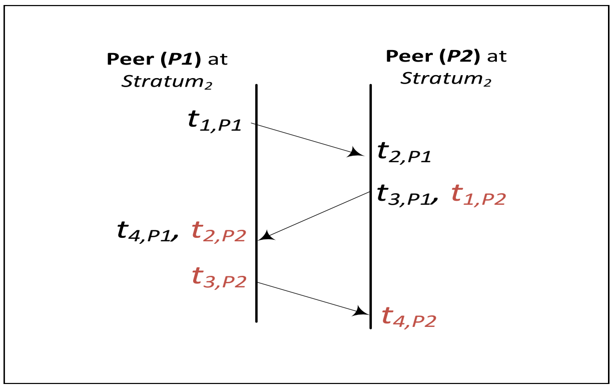


FIGURE 3.13: Symmetric mode in NTP [145].

In the third mode which is the broadcast mode, there is no message exchange between two nodes. The purpose of the broadcast message is to send the configuration setup from a server to all nodes in a large size network. For example, the broadcast mode can be used to distribute the list of NTP servers as a broadcast message to all clients instead of configuring each client manually or using a distributing protocol like Dynamic Host Configuration Protocol (DHCP).

3.6.3 Synchronization Process in PTP

In PTP, periodic $SYNC$ messages are sent from the grandmaster clock to all slave nodes that record two timestamps. The first time-stamp represents the sending time and the second time-stamp represents the time when the slave node receives the $SYNC$ message. However, two timestamps are not enough to measure the $SYNC$ message transmission delay that is used to compute the offset of the slave clock to the grandmaster.

Therefore, two mechanisms are used to measure the $SYNC$ message transmission delay [145]: End-to-End (E2E) communication uses a request/response message along the path from the slave to the grandmaster clock. The Peer-to-Peer (P2P) mechanism uses a request/response message between two directly connected peers. E2E is the same as the client-server mode in NTP protocol. E2E measures the *SYNC* message transmission delay along the path that contains switches and routers between the requesting slave node and its grandmaster clock. In the P2P mechanism, the delay is measured between two adjacent peers that are a part of the path between the slave node and its grandmaster clock. The measured delay is aggregated and loaded into a specific field named *correction-field* inside the *SYNC* message. At the instant the *SYNC* message is received at the slave node, the total delay for the whole path is accumulated and used to compute the slave node offset to the grandmaster clock. It is worth noting that the two mentioned mechanisms consider the residence time of the transmitted messages in each transparent clock (router, switch or hub) in order to measure that delay.

3.6.4 Asymmetry in the Measured Delay

One of the main sources that negatively affects the accuracy of the measured offset are asymmetric delays [145]. The standard NTP and PTP protocols assume that the message transmission delay (i.e. request and response delays) is identical in both directions. In real life especially in wireless environments the symmetry is a rare case. Figure 3.14 illustrates an aspect of the asymmetric delay. For example, if a message needs 50 microseconds to be transmitted from a client to a server and it needs 100 microseconds to be sent back, the measured offset will be decreased with an error of 25 microseconds. Thereby, the client will actually be running 25 microseconds behind its server.

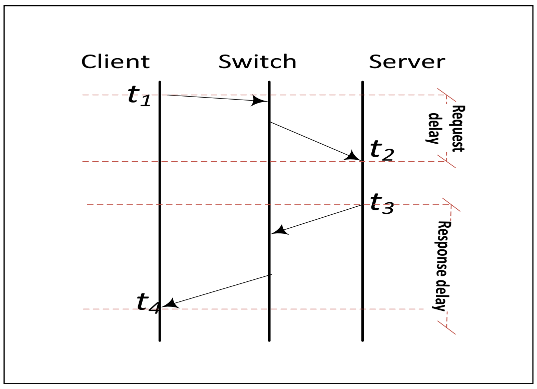


FIGURE 3.14: Asymmetry in the measured delay [145].

Sources of asymmetry are difficult to be detected or avoided. They may be at different layers such as the link layer. For example, the message may go over different routes in the bidirectional transmission. Furthermore, the delays (i.e. propagation and transmission delays) in the wireless systems are changeable as a result of using different wireless broadband communication standards (e.g. WiMAX, Wi-Fi, LTE) or there are variable queueing delays in the routers and switches. Moreover, the processing delay is not identical for all network components. The slave and the client cannot independently measure the error due to the asymmetry, even if the reference time source is directly connected with the slave and client.

The processing and the queueing delays can be corrected in the PTP protocol by using the *Correction-field* inside the *SYNC* message as mention in the P2P mechanism. All PTP-aware routers and switches that support that field are called Transparent Clocks (TCs). When grandmaster clock sends periodic *SYNC* messages, the difference between the reception and transmission times of the *SYNC* message (i.e. the residence time) over each TC is aggregated besides the peer-to-peer delays to the value of the *Correction-field*. At the slave node, the accumulated delay value is used to compute the slave's offset. But on the other hand, the standard PTP protocol does not address other sources of asymmetry like asymmetric routing and changing delays, and it does not carry out methods to solve the asymmetry. The best solution would be that NTP servers are directly connected with their clients.

3.6.5 Clock Drift

Another critical issue that should be considered in NTP and PTP is the node's clock drift, because it causes divergence during the synchronization process. Every node is susceptible to drift, which varies from node to node depending on the quality of the clock (for example, a clock oscillator (XO) or a Phase Locked Loop (PLL) synthesizer) and the surrounding environment (e.g., temperature, power-level stability). Because of this, the slave/client clock's rate of operation differs somewhat from that of its master/server clock. After some time any local clock will gradually drift apart and exhibit an offset from its reference clock. Therefore, another field has to be integrated into the $SYNC$ message to adjust the slave/client clocks to the same frequency as the master/server clock.

3.6.6 Time-stamping Errors

Inaccuracy of the time-stamping means that the time-stamp does not correspond to the exact time at the instant of sending or receiving a message [145]. It is also a source of the synchronization error. The time-stamping can occur at three places: in the user space, in the kernel (i.e. software time-stamping), or in the Network Interface Card (NIC) (i.e. hardware time-stamping). Figure 3.15 shows the inaccuracy of the time-stamping in the NTP protocol. The client at $Stratum3$ misses the exact time (indicated in a black circle) at the times of sending the client mode message and receiving the server mode message. Similarly, the inaccuracy is shown in the server at $Stratum2$.

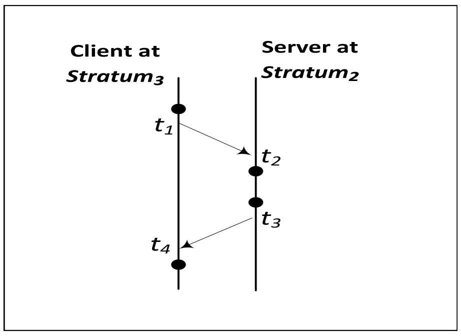


FIGURE 3.15: A simple example shows the inaccuracy of the time-stamping in the NTP protocol [145].

Since context switching, processing time, and network stack waiting time are not necessary, software time-stamping in the kernel is more precise than in user space. Similarly, the hardware time-stamping in the NIC is more accurate than in the kernel because the timestamp is generated using the hardware clock of the NIC. However, there are several challenges with hardware and software time-stamping that do not exist in the user space time-stamping. For example, hardware time-stamping is not supported by every network card which also has its own clock. Therefore, an accurate synchronization between the NIC clock and the system clock is required. The kernel in the software time-stamping does not know where to include the transmit time-stamp, unlike the user space application and the hardware time-stamping where the server/master and the NIC take care of including the transmit time-stamp in the messages, respectively.

In addition, the user space and software time-stamping are less accurate because of interrupt coalescing, which is implemented to avoid flooding the node with a lot of interrupts by buffering the received messages at the NIC and processing them in one single interrupt. This causes an increase in the time-stamping error in the user space and software time-stamping as a result of the buffering time.

If the total time-stamping error induced in the transmit and receive timestamps for a message sent from node $N1$ to node $N2$ is the same total time-stamping error as for a message sent from $N2$ to $N1$, then the errors will cancel each other. If there is asymmetry in transmit and receive timestamps of the two directions, this error is not considered in the time synchronization process. Therefore, it is recommended to use a ratio formula to determine the relationship between the two asymmetry error values, and to use the same time-stamping method and the same NIC model on both sides.

The advantages of NTP is using multiple sources, the client can select the best sources for synchronization, whereas PTP needs to allow multiple grandmasters to provide the slaves with more information for selecting the best grandmaster. The advantage of PTP is better time-stamp correction and addressing the delay correction but it needs substantial solutions to address the asymmetric delays. On the contrary, NTP requires adopting the concept of delay correction.

3.7 IEEE 802.1AS Standard Protocol

As mentioned in the previous section, PTP is better suited for time-sensitive and control applications because of its synchronization accuracy as well as the transmit time-stamp and delay corrections. On the contrary, NTP is not suitable for many industrial time-sensitive applications due to its synchronization error with a margin of milliseconds. Therefore, this section discusses the IEEE 802.1AS protocol that is based on the PTP protocol, generalized Precision Time Protocol (gPTP) is another name for it. Later, it will be the basis to synchronize clocks in the thesis. The IEEE 802.1AS protocol is built on top of the IEEE 1588 standard with more enhanced features in order to satisfy the requirements of real-time applications.

3.7.1 IEEE 802.1AS Node Types

IEEE 802.1AS is capable to implement its features provided that all attached time-aware nodes support gPTP services, thus, a gPTP domain is established. To do that, each time-aware node examines if its adjacent node is time-aware by using the P2P request/response mechanism. For this reason, a time-aware node sends a request message ($Delay\_Req$) to its adjacent node and then waits for its response. The time-aware node identifies its neighbour as a standard node (i.e. it does not support gPTP services) if there is no response, or if there are multiple response messages ($Delay\_Res's$) or the received timestamps are illogical and unpredictable. IEEE 802.1AS categorizes the time-aware nodes into two types:

Time-aware end node: it is a node with one port, and it acts as a slave or grandmaster clock. In other words, if the port is identified as a slave state, the clock of the node is synchronized with its grandmaster at the time of receiving $SYNC$ messages. In contrast, if the port is in the master state, the node plays the role of the grandmaster. Its clock is synchronized with an accurate external reference time (e.g. GPS) and sends $SYNC$ messages periodically to its slave nodes in the gPTP domain.

Time-aware intermediate node: it is a multi-port relay node (i.e. bridge, router or switch). The state of each port is either in slave or master state depending on the port's state of the connected node. Thus, the time-aware intermediate node may play the role of slave and master simultaneously. Namely, a port of a time-aware intermediate node is in the slave state if it connected with a node through its master port. The port is in the master state if it is connected with a port in the slave state. It is worth noting that a port in a master state means that it forwards the grandmaster reference time to the next slave port that synchronizes the slave node with the received reference time. When the time-aware intermediate node receives the *SYNC* message of the grandmaster clock at the ingress port. It modifies the received timing information by adding the residence time and the P2P path delay which is measured by exchanging a series of request/response messages between two neighbouring time-aware nodes. Eventually, it forwards the modified *SYNC* message to the egress port toward the next time-aware node. At the time-aware end node, the received timing information is used to adjust the local clock with the grandmaster clock. However, the time-aware intermediate node forwards non-gPTP and management messages in a similar way to the standard node.

3.7.2 Synchronization in the 802.1AS Domain

In the gPTP (802.1AS) domain, firstly, the best source clock should be selected as the grandmaster clock by using an algorithm named the Best Master Clock Algorithm (BMCA). Then all states for all time-aware nodes are identified to build the slave-master hierarchy. The selected grandmaster sends periodically *SYNC* messages which include its current clock time to all slave clocks in the domain. During the *SYNC* message transmission, the time-aware intermediate nodes modify the timing information in the *SYNC* message and then forward it until arriving at the time-aware end node.

3.7.3 Best Master Clock Algorithm

Each clock has its own set of attributes including a user-configurable priority, clock class, clock accuracy, and clock variance. To select the best available source clock as a grandmaster, time-aware nodes send their set of attributes to all nodes in the gPTP domain through *Announce* messages. The time between these messages is normally configured to be a couple of seconds. BMCA algorithm is executed in each port of a clock to compare the local attributes of the clock with other sets of attributes for time-aware clocks once receiving their *Announce* messages. The priority attribute of a clock is configured manually. A clock is preferred over another for specific reasons like the position where a centralized position gives a clock higher priority. If two clocks have the same priority, they will be compared by the clock class, which defines the role of the clocks and whether they are used as a grandmaster clocks. If two clocks have the same class, the BMCA will prefer the clock that has better accuracy. If the accuracy is the same, then the clock variance is used. In the case of the same variance, BMCA determines which node acts as an intermediate or as an end node. If none of the mentioned attributes breaks the tie to determine the best grandmaster, then the clock that has a higher MAC address will be selected as a grandmaster clock. The data set comparison of each clock determines which clock is the best to maintain the timing network. Besides, the output of the comparison is also used to update the state of each port that receives the *Announce* messages from all nodes in the gPTP domain [11].

Figure 3.16 illustrates an example of a master-slave hierarchy in the gPTP domain. It is assumed that *time-aware-end-node*$_1$ has an external reference clock (e.g. GPS) and possesses the best data set among all clocks. Thereby, it is selected as the grandmaster clock. *time-aware-end-node*$_1$ receives timing information from GPS and sends this information to other nodes via its master port, if the selected grandmaster clock does not have an external timing source, it will then send its local time and encapsulates it in the outgoing *SYNC* message. *time-aware-intermediate*$_1$, *time-aware-intermediate*$_2$, and *time-aware-intermediate*$_3$

receive the *SYNC* message provided by *time-aware-end-node*$_1$ at their slave ports, adjust their local clocks. Afterwards they forward the *SYNC* message through their master ports to peer-to-peer connected slave ports in the next neighbour clock along the route toward the time-aware end nodes.

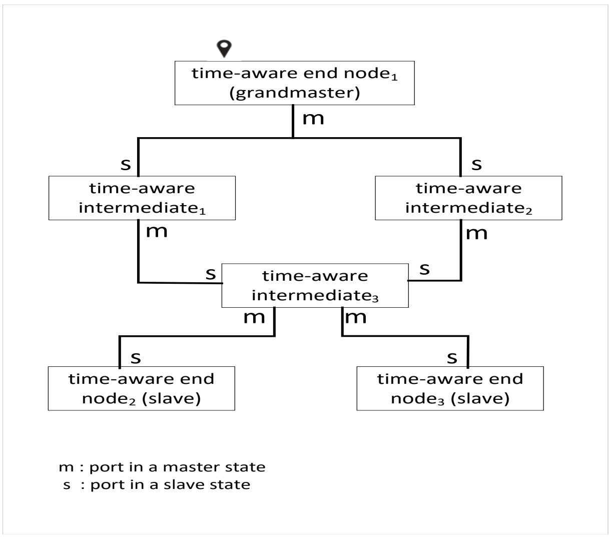

time-aware end node₁ (grandmaster)
m
s
s
time-aware intermediate₁
time-aware intermediate₂
m
m
s
time-aware intermediate₃
s
m
m
s
s
time-aware end node₂ (slave)
time-aware end node₃ (slave)
m : port in a master state
s : port in a slave state

FIGURE 3.16: An example of master-slave hierarchy in the gPTP domain.

3.7.4 SYNC Message Delay Measurement

In the gPTP domain, time-aware clocks are interconnected through separate communication links. Therefore, when the slave end node receives the *SYNC* messages from the grandmaster, it is required to measure the *SYNC* message delay along the forwarding paths to calculate the accurate clock offset concerning the grandmaster clock. To do that, the P2P mechanism is implemented to measure the delay between two peers of time-aware nodes. Therefore, every two ports of the time-aware nodes calculate the delay of a path which is a part of the forwarding paths of the *SYNC* message. The path delay is added to the *Correction-field* that is later used for the slave clock correction.

P2P mechanism is implemented through exchanging *Delay_Req*/ *Delay_Res* timing messages and possibly *Follow-Up-Delay-Res* messages which are used to send the origin timestamps to avoid potential time-stamp errors. As denoted in Figure 3.17, the pattern of the P2P mechanism is:

- *Node*$_1$ *Clock* dispatches a *Delay_Req* message towards *Node*$_2$ *Clock* and records the transmission time-stamp t_1.
- *Node*$_2$ *Clock* generates the time-stamp t_2 upon arrival of the *Delay_Req* message.
- *Node*$_2$ *Clock* sends back a *Delay_Res* message and generates the transmission time-stamp t_3. Dispatching of the *Delay_Res* message is done immediately after receiving the *Delay_Req* message to minimize the impact of the clock offset between *Node*$_1$ and *Node*$_2$ *Clocks*. *Node*$_2$ sends the timestamps t_2 and t_3 to *Node*$_1$ in one of the following ways:
  - The *Delay_Res* message carries both timestamps t_3 and t_2.
  - When a two-step clock is used, the *Delay_Res* message and the *Follow-Up-Delay-Res* message carry the time-stamp t_2 and the time-stamp t_3, respectively.

- $Node_1$ generates the time-stamp t_4 upon arrival of the *Delay_Res* message or the *Follow-Up-Delay-Res* message when a two-step clock is used.

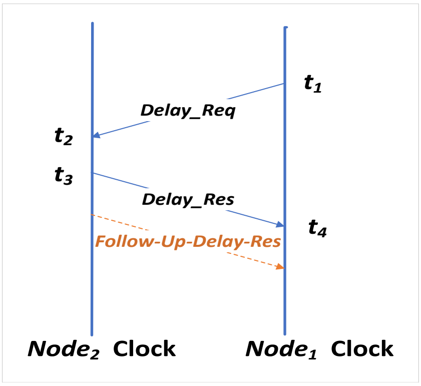


FIGURE 3.17: P2P mechanism between adjacent time-aware node clocks.

After exchanging these P2P messages, $Node_1$ gets all required timing information to measure the *MeanPathDelay* as denoted in the following equation:

$$MeanPathDelay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{3.1}$$

The assumption in Eq. 3.1 is that the communication delay between $Node_1$ and $Node_2$ in both directions is symmetrical. In fact, asymmetric sources (e.g., asymmetric routing and changing delays) which exist in wireless systems, result in time-stamping errors and make the measured path delay an inaccurate measurement. Therefore, more improvements in the standard IEEE 802.1AS are needed to take all asymmetric sources into account to achieve a high precision of synchronization.

3.8 Link Scheduling in Wireless Networks

In a wireless network, a message is usually transmitted from a source node to a sink node through intermediate relays (i.e. switches and routers). To do that, several sequences of P2P connections are required, thus a wireless network is multi-hop in nature. In addition, multiple P2P connections corresponding to different source-sink pairs are used to exploit the channel capacity. Therefore, the design of wireless networks faces many challenges particularly in the design of the physical and MAC layers to achieve high throughput and improve the data rate. At the physical layer, several methods are used to increase the capacity of the channel. For instance, Multiple Input Multiple Output (MIMO) [148] multiplies the capacity of the channel using multiple transmitting and receiving propagation paths, and Orthogonal Frequency Division Multiplexing modulation technique (OFDM) [149] which carries data in parallel. At the MAC layer, the authors in [150] take advantage of multiple channels by switching channels dynamically. The survey in [151] addresses several methods that are proposed at the MAC layer to enhance the utilization of the channel by improving the spatial reuse.

Achieving high network throughput plays a critical role in real-time applications, because data traffic has to be delivered to the intended receivers with predefined deadlines. Besides the channel utilization, the message transmission should also consider assessing the level of the interference and avoid the retransmission commonly used in packet-switched computer networks. To eliminate the induced interference due to possible collisions and to limit the end-to-end delay, the literature addressed the problem by routing and scheduling the message access in the wireless medium [152].

Several medium access mechanisms can be implemented such as TDMA and Frequency Division Multiple Access (FDMA). In real-time wireless networks, TDMA is more feasible and preferred to be used over FDMA, because TDMA is more efficient in using the spectrum, more users can use the same spectrum and different types of data traffic (e.g., data, audio and video streams) that require different data rates can be transmitted using the same medium using multiple time slots. Moreover, TDMA-based protocols are used to achieve deterministic communication properties. TDMA slots are assigned for each user (i.e. transmitter) periodically, thus, the receiver receives the transmitted data with fixed duration TDMA time slots. Therefore, the associated actions can be executed at the right time and the absence of data transfer can be easily detected by the system.

The link scheduling using TDMA should be well designed so that all packets transmitted through the wireless medium are received at the receiver successfully within limited time. In the literature, physical and protocol interference models are proposed for specifying the methodology for successful data reception. The following two subsections demonstrate the concept of each model.

3.8.1 Link Scheduling Based on the Protocol Interference Model

In the protocol interference model [153], [154], the message is successfully received at the intended receiver only if the transmitter is within the communication range of the receiver and other interfering transmitters have to be outside the interference range of the receiver. In other words, the protocol model imposes a silence zone around a receiver during the time of the data reception.

Assume $re_{i,j}$ is a receiver that receives messages using a connection path j at time slot i. The received power at $re_{i,j}$ from its transmitter $tr_{i,j}$ is $\frac{P_{tr}}{N_0 D^\alpha\left(tr_{i,j}, re_{i,j}\right)}$, where P_{tr} is the transmitted power, N_0 denotes the power of noise in the bandwidth, D is the physical distance between $tr_{i,j}$ and $re_{i,j}$ and α denotes the path loss to the distance between the transmitter and the receiver. The received power at $re_{i,j}$ from other interfering transmitters is $\sum_{\substack{k=1\\k\neq j}}^{M_i} \frac{P_{tr}}{D^\alpha\left(tr_{i,k}, re_{i,j}\right)}$, where M_i is the total number of unintended transmitters, k is a connection path that is used by an unintended transmitter $tr_{i,k}$ that interferes the received signal to $re_{i,j}$ at time slot i. Thus, the Signal to Noise Ratio (SNR) at receiver $re_{i,j}$ is measured by the following equation [155]:

$$\mathrm{SNR}_{re_{i,j}} = \frac{P_{tr}}{N_0 D^\alpha\left(tr_{i,j}, re_{i,j}\right)} \tag{3.2}$$

Depending on the definition of the protocol model, the data from $tr_{i,j}$ to $re_{i,j}$ is successfully received if:

- The SNR value from $tr_{i,j}$ at $re_{i,j}$ is no less than the communication threshold (i.e. β), from Eq. 3.2 the communication range (R_{comm}) is defined as [155]:

$$D\left(tr_{i,j}, re_{i,j}\right) < \left(\frac{P_{tr}}{N_0\beta}\right)^{\frac{1}{\alpha}} =: R_{comm} \tag{3.3}$$

- The SNR value from any unintended transmitter $tr_{i,k}$ at $re_{i,j}$ is less than the interference threshold (i.e ψ), from Eq. 3.2 the interference range (R_{int}) is defined as [155]:

$$D\left(tr_{i,k}, re_{i,j}\right) > \left(\frac{P_{tr}}{N_0\psi}\right)^{\frac{1}{\alpha}} =: R_{int} \qquad \forall k = 1, \ldots, M_i, k \neq j \tag{3.4}$$

According to Equations. 3.3 and 3.4, the message transmission is successful if the distance between the transmitter-receiver pair is less than the communication range of the receiver and no other transmitters locate in a distance less than the interference range of the receiver.

For example, Figure 3.18 shows a communicating transmitter-receiver pair A→ B, where A denotes the node which transmits a packet and B denotes the node which receives that packet. The transmission is successful because node B (i.e. the receiver) locates inside the communication range of node A. Moreover, the communication pair C→ D can communicate concurrently in the same time slot with A→ B because node C locates outside the interference range of node B during its communication with node D.

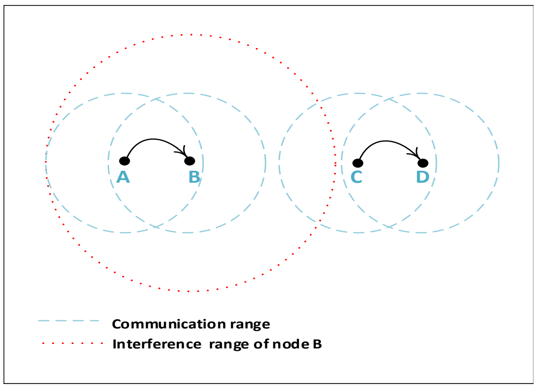


FIGURE 3.18: A simple example shows the communication and the interference ranges of the nodes.

3.8.2 Link scheduling Based on the Physical Interference Model

In the physical interference model [153], [156], A message is only successfully received at a receiver if the Signal to Interference and Noise Ratio (SINR) there is equal to or greater than the communication threshold (i.e. β). In other words, the transmission at the transmitter-receiver pair $t_{i,j} \rightarrow r_{i,j}$ is successful if:

$$\frac{\frac{P_{tr}}{D^{\alpha}\left(tr_{i,j}, re_{i,j}\right)}}{N_0 + \sum_{\substack{k=1 \\ k\neq j}}^{M_i} \frac{P_{tr}}{D^{\alpha}\left(tr_{i,k}, re_{i,j}\right)}} \geq \beta \tag{3.5}$$

Where the value on the left side of the inequality denotes the SINR value at the receiver $r_{i,j}$, see [157].

3.8.3 Limitation of the Protocol Interference Model

A wide variety of wireless networks employ the protocol model due to its simplicity. Hence, we notice that the protocol interference model is less complex than the physical interference model but it is more restrictive. The protocol model divides the TDMA wireless networks into communication and interference zones. Thus the link scheduling problem is converted to be a graph edge-colouring problem [158] so that the researchers seek to reduce the number of colors (i.e. time slot numbers) regardless of the network design. Despite its simplicity, there are also doubts about its validity, which means the protocol model does not guarantee conflict-free transmissions. The conflict-freedom is satisfied if the SINR at every receiver is always more than a specified communication threshold (i.e. β). However, the algorithms that are based on the protocol model do not necessarily maximize the throughput of a TDMA wireless network as a result of the following reasons:

- Accumulative receiver interference caused by scheduling multiple transmitter and receiver pairs in the same time slot depends on strict communication and interference thresholds [159]. This, in turn, leads to a decrease in the SINR values at the receivers when increasing number of concurrent transmissions. Thus, some transmissions are unsuccessful according to the condition in Eq. 3.5.

  For example, assume the TDMA wireless network in Figure 3.19 consisting of six nodes whose labels and coordinates are 1: $(-360, 0)$, 2: $(-450, 0)$, 3: (90, 0), 4: (0, 0), 5: (360, 0) and 6: (450, 0), and three transmitter-receiver pairs $1 \to 2$, $3 \to 4$, and $5 \to 6$. Table 3.1 shows the system parameters that are implemented on the tested network. According to Equations 3.3 and 3.4, the computed R_{comm} and R_{int} for all nodes are 100m and 177.8m, respectively.

TABLE 3.1: System parameters for a TDMA network [155].

Parameter	Value
Transmission power (P_{tr})	10 mW
Communication threshold (β)	20 dB
Interference threshold (ψ)	10 dB
Path loss (α)	4
Noise power (N_0)	-90 dBm

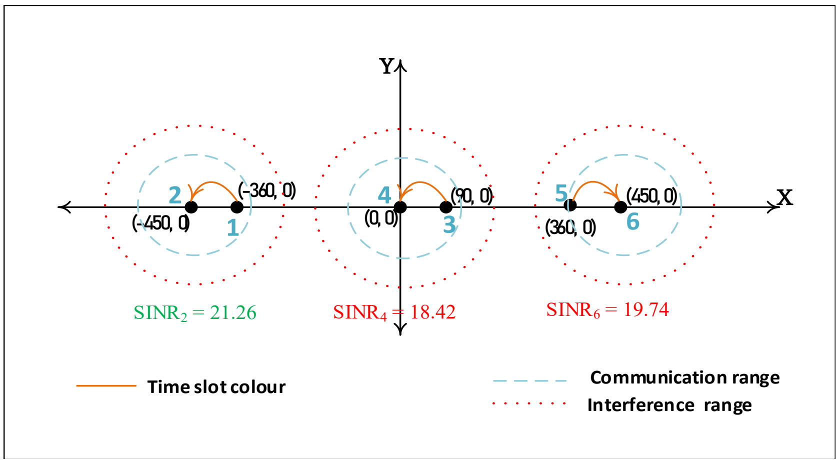


FIGURE 3.19: Six nodes are deployed on a TDMA network and the protocol model produces one color time slot [155].

According to the protocol model conditions, we see that all transmissions can be received and scheduled in the same color (i.e. time slot). To make it easy to understand, all transmissions are symbolled with orange-coloured arrows as shown in Figure 3.19. However, the computed SINR values at nodes 2, 4, and 6 are 21.26 dB, 18.42 dB, and 19.74 dB respectively. According to the condition in Eq. 3.5, the transmission $1 \to 2$ is successful because the SINR value at node 2 (i.e. 21.26 dB) is more than β (i.e. 20 dB), while $3 \rightarrow 4$ and $5 \rightarrow 6$ transmissions are unsuccessful because SINR values at nodes 4 and 6 (i.e. 18.42 dB and 19.74 dB) are less than β, respectively. Thereby, this example shows that scheduling all transmission at the same time slot leads to low network throughput due to unsuccessful transmissions.

- On the contrary, the protocol model can result in more colors than required. For example, Figure 3.20 shows a TDMA wireless network with four nodes whose labels and coordinates are 1: (0, 0), 2: (50, 0), 3: (220, 0), and 4: (170, 0), and two transmitter-receiver pairs $1 \rightarrow 2$ and $3 \rightarrow 4$. Because the transmission pairs are located in the interference range of each other, the protocol model will typically color the transmissions with different colors, thus, the transmission pairs will be scheduled in different time slots. The computed SINR value at receiver 2 and receiver 4 are both 32.04 dB.

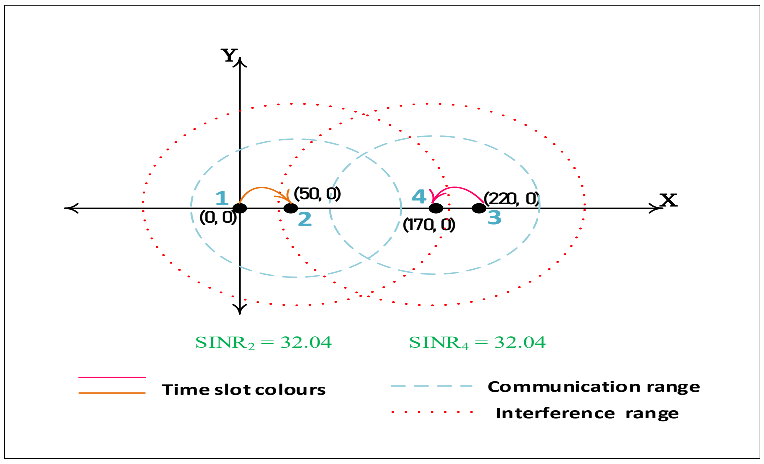


FIGURE 3.20: Four nodes are deployed on a TDMA network and the protocol model produces two color time slots [155].

We notice that the computed SINR in the protocol model at both receivers is well above the value of β. In contrast, the physical model schedules the two transmitter-receiver pairs in the same time slot according to the condition in Eq. 3.5. The graph edge-colouring in Figure 3.21 shows that the new computed SINR at receivers 2 and 4 are both 20.91 dB, which are still above the communication threshold of 20 dB. Thus, the physical model schedules the two transmissions successfully because the signal power at the receivers is high enough to tolerate the induced interference. In summary, the mentioned example demonstrates that the protocol model will schedule several transmissions in different time slots and it, in turn, leads to a decrease in the network throughput compared to the scheduling provided by the physical mode. In essence, though SINR compliance is satisfied in the protocol model the network throughput can be further enhanced without affecting the success of the transmissions.

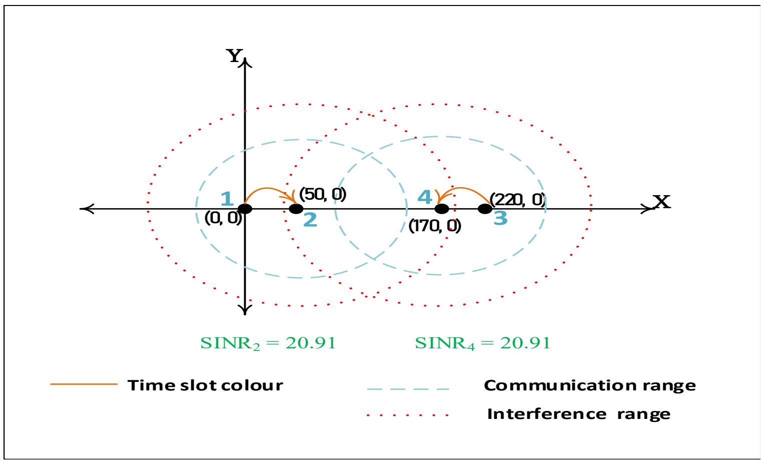


FIGURE 3.21: Four nodes are deployed on a TDMA network, the physical model produces one color time slot [155].

- The protocol model cannot recognize the topology of the network, thus, it builds its link scheduling without interesting in the positions of the transmitters and receivers.

Due to low throughput and more time slots than required as a result of the link scheduling in the protocol model. Alternatively, the literature addressed algorithms based on the physical model to maximize the network throughput and minimize the necessary time slots to get a shorter schedule which is an essential issue in real-time systems.

3.9 IEEE 802.11 Standard Protocol

Wireless nodes are equipped with communication interfaces to communicate with other nodes in the network. In general, these communication interfaces are based on IEEE 802.11 [160] that defines a set of protocols (i.e. standards) for communications at both 2.4 GHz and 5 GHz frequency bands. Several protocols have emerged from the IEEE 802.11 original standard such as 802.11b, 802.11g, and 802.11n protocols.

For the physical layer, IEEE 802.11 defines two forms of spread spectrum modulation: Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) [161] that are operating at 2.4GHz frequency and 1-2 Mbps data rate. In the IEEE 802.11 MAC architecture, two medium access modes are defined: Distributed Coordination Function (DCF) and Point Coordination Function (PCF) [162]. The default DCF mode is implemented in both infrastructure networks (based on nodes connected with a base station) and ad hoc networks. In order to access the medium, DCF employs CSMA/CA [163] with an exponential backoff algorithm. The nodes compete to get access and they implement inter-frame spacing before starting the transmission, whereas the medium access in PCF is centralized by a coordinator that controls which node can transmit.

The advantages of the IEEE 802.11 protocol are the low cost and relatively high-frequency range. On the contrary, this protocol is sensitive to traffic disruptions and depends generally on the backoff algorithm to randomly access the medium to avoid the collision. The non-deterministic medium access control makes this type of protocol unsuitable for time-sensitive applications.

3.9.1 IEEE 802.11 DCF (Contention-based) Mode

As mentioned, DCF mode employs CSMA/CA rather than a technique to detect collisions (e.g., CSMA/CD) in the wireless system. The reason is that the half-duplex mode is used for data transmission. In other words, wireless transceivers cannot transmit and receive on the same channel at the same time. Therefore, each unicast data transmission has to be acknowledged.

In DCF, any node that wants to send a packet on the medium has to wait for a fixed time interval called Distributed Inter-Frame Space (DIFS). At the instant the DIFS interval ends, the node senses the medium and if the medium is free, the node immediately sends its packet. If instead, the medium is busy (i.e. another node is currently sending its packet), the node selects randomly a backoff counter (i.e. number of waiting time slots) from a set called Contention Window (CW). Typically, the size of CW is between 0 and 15 and it is directly proportional to the size of the network. This technique is used to avoid collisions with another node that wants to send packets at the same time. However, after the backoff counter is selected, the node continues listening to the medium. When the medium becomes free again, the node firstly waits for the DIFS interval before starting to decrease its backoff counter. During the decreasing process, if the medium becomes busy again, the node stops the decreasing process and it is resumed the process later with the remaining counter when the medium returns to be free. At the moment the backoff counter of the node reaches zero and the medium is still free, the node has to wait for another DIFS interval time before starting the packet transmission. The sink node must wait for the Short Inter-Frame Space (SIFS) period after it receives the packet, which gives the acknowledgement (ACK) priority over the data. When SIFS ends, the sink node starts sending back the ACK packet to the transmitting node.

Figure 3.22 shows a simple example of the DCF operation. Node $N1$ has sent packet $P0$ and it is ready to send the second packet $P1$. As discussed, node $N1$ waits for the DIFS interval time and senses the medium. As shown in the figure, the medium is free, hence, node $N1$ starts sending packet $P1$. During the transmission of packet $P1$, nodes $N2$ and $N3$ are ready to transmit their packets $P2$ and $P3$, respectively. $N2$ and $N3$ sense the medium and they find out that the medium is busy. Therefore, they generate backoff counters independently. $N2$ generates 4 waiting time slots while $N3$ generates 7 waiting time slots. When $N1$ finishes the transmission of packet $P1$, both $N2$ and $N3$ wait for DIFS and decrease their backoff counters. $N2$ finishes its countdown before $N3$ (i.e. its backoff counter reaches zero) and starts sending its packet $P2$. $N3$ senses the medium again and it finds out that it is busy, so $N3$ stops decreasing its backoff counter and it resumes decreasing the remaining backoff counter (3 waiting time slots) after the DIFS interval time has elapsed. At the time the medium is free and the backoff counter of $N3$ is zero, $N3$ starts to send its packet $P3$.

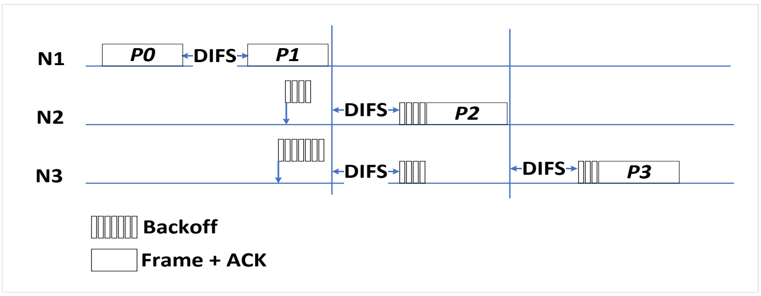


FIGURE 3.22: IEEE 802.11 DCF operation in 802.11 protocol [164].

In wireless networks, there is a problem called the hidden node problem [165]. It occurs when a node is visible to a receiver and not visible to another node that is communicating with the same receiver. Therefore, IEEE 802.11 protocol uses a mechanism called Request To Send (RTS)/Clear To Send (CTS) [166] which uses RTS and CTS packets to check the medium and avoid a potential collision. When a node wants to send data, firstly it sends the RTS packet to the receiver to determine if the receiver is ready to receive the data. The receiver will send back the CTS packet, and the data transmission begins. Thus, all neighbour nodes overhear the exchanged RTS/CTS packets and they are prevented from sending their data until the data transmission is finished and another RTS packet is sent from another node.

3.9.2 IEEE 802.11 PCF (Contention-free) Mode

PCF is an optional mode, which means that it is not required to be implemented in IEEE 802.11 nodes. PCF mode is intended for the transmission of time sensitive information. The base station (i.e. access point) works as a coordinator and controls all connected nodes. It enables the PCF mode and it polls the node that has the right to access the medium. The coordinator has higher priority than other nodes to access the medium. Therefore, it waits for an interval of time called PCF Interframe Space (PIFS) which is less than DIFS. During a contention-free period, the coordinator steps through all nodes and polls one node at a time. The polled node that receives a contention-free poll frame starts transmitting its traffic and all other nodes need to wait. In case the node has no packets in its queue, it has to transmit a null frame. After that, the coordinator polls the next node until all nodes had the chance to transmit. Since the priority of PCF is higher than DCF, the nodes that support only DCF mode may not get permission to access the medium. Therefore, a repetition interval has been implemented to consider both DCF and PCF traffic.

PCF ensures regular and deterministic delays for PCF-enabled nodes. Thus, PCF mode is suitable to transmit messages such as video and control traffic that require robust timing guarantees.

3.9.3 Limitations of 802.11 for Wireless Systems

In general, the IEEE 802.11 protocol exhibits several limitations for transmitting time-sensitive data due to its inability to support multi-hop data transmission and the random backoff strategy.

Random Backoff Problem

Due to the randomness in selecting the backoff counter and the probability that several nodes may select the same backoff counter, the RTS/CTS and backoff techniques are considered as indeterministic solutions to solve the collision problem. The absence of determinism in data transmission may also lead to the false blocking problem [167] that occurs in case the nodes are unable to reply to the received RTS packet when they are in a deferring state. This, in turn, motivates researchers to find deterministic alternatives, especially for real-time applications.

Multi-hop Transmission

The 802.11 protocol is mainly implemented to transmit data in one-hop bidirectional communication from nodes connected to access points that are presented as bridges to wired networks. However, the nodes that use this protocol are not supported with routing capabilities to propagate data among multi-hop nodes [168]. Despite the low cost and relatively high communication range, the lack of multi-hop capability and non-deterministic medium access demonstrate the need to find solutions that address real-time constraints. Therefore, the next two sections demonstrate briefly the recent research in real-time communication protocols for wireless and wired systems.

3.10 Real-time Communication Protocols in Wireless Systems

In general, wireless networks are less reliable than wired ones because the probability of error occurrence is higher. The challenges include the possibility of interference due to the node mobility, high power lines, electrical components, long-distance P2P communication. Collision detection and correction methods become more difficult in wireless networks due to the fact that the nodes do not listen to the shared channel while data transmitting. As discussed in the previous section, the majority of methods to solve the collision problem relies on random backoff techniques which produces solutions that depend mainly on non-deterministic waiting times in the MAC layer [169]. This could result in waiting times and undesirable margins of end-to-end delays in real-time applications. Therefore this section describes Real-Time (RT) MAC protocols that are applied for wireless technologies. The literature classifies RT MAC protocols into hard and soft protocols.

3.10.1 Hard Real-Time MAC Protocols

Real-time Protocols Based on Time-Division Multiple Access

As mentioned in Section 3.8, TDMA is used to access a channel in a shared medium network by dividing time into different time slots. This method has many advantages such as determinism, different types of traffic can be sent by allocating them to different time slots, guarantee of non-interference as a result of simultaneous transmission and sharing among multiple users the same carrier frequency. Therefore, several researchers suggested TDMA to be adopted in their solutions for real-time requirements [170]–[172].

Real-time and Reliable MAC (RRMAC) [173] is a TDMA based protocol that depends on a tree design to forward the messages in one super-frame from the bottom-level nodes to the nodes at the top-level. Figure 3.23 shows the structure of the super-frame which is divided into a beacon period, active period (i.e. Contention Access Period (CAP), Contention Free Period (CFP)), and inactive period. These periods are used to synchronize the message transmission. Using CSMA/CA, the nodes in CAP compete for access to the channel. A portion of CAP remains for contention access by new nodes joining the network. In CFP, Guaranteed Time Slots (GTS) are assigned to messages that require low latency without any contention during transmission. The nodes are controlled by coordinators that coordinate the GTS requests sent from connected nodes with time-sensitive data. The nodes go into a low-power mode when they are not in use to conserve energy and prolong the battery life. The beacon period is used to forward the beacon through multiple-hops.

The structure of the super-frame is flexible [174]. If the nodes do not have real-time data, CFP can be eliminated from the super-frame structure. The nodes in RRMAC are located in clusters [174], [175], where the cluster head (i.e. coordinator) is that the top level and collects the data from bottom-level nodes in its cluster. Radio Frequency (RF) power for top-level nodes is higher than for lower-level nodes in order to increase the transmission range. However, the disadvantage of RRMAC is the difficulty of the global time synchronization in large wireless networks.

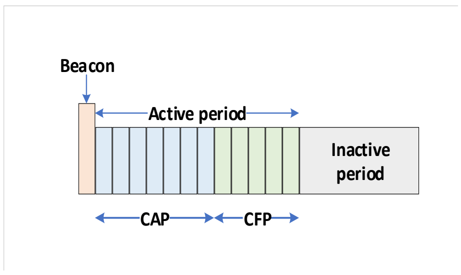


FIGURE 3.23: RRMAC superframe structure [174].

Dual Mode Real-time Protocols

Dual-mode RT protocol [176], [177] is a hard real-time protocol, which is subdivided into protected and unprotected modes. In the unprotected mode, the transmission of the messages is unreliable but at full speed. On the contrary, the messages in the protected mode incur a higher delay but have an improved reliability. The unprotected mode is used when collisions are not likely. When a node wants to send data, it initializes the backoff-time at the instant of receiving a signalling message from a sender. This time is inversely proportional to the distance from the sender, which means that the farthest node sends its messages first and all other nodes reset their backoff-times and are used only for forwarding the message.

In the protected mode, nodes are clustered into groups so that each node of a group can connect with another node in a neighbouring group. The protected mode provides a guarantee for reliable collision-free communication during run-time. This mode uses signaling messages to reserve a group between the source and sink nodes. The reserved group cannot generate messages until the transmission is completed. Initially, the unprotected mode begins but in case a node detects a collision, it immediately sends an alarm message to inform the neighbour nodes and to switch their modes to the protected mode. However, this protocol requires that all nodes are well-synchronized and accurately know their positions. Hence, it is hard to be implemented in randomly deployed wireless systems. Moreover, energy consumption is not considered in this type of protocols.

Real-time Protocols Based on Frequency Division Multiple Access

Implicit Earliest Deadline First (I-EDF) [178], [179] is another hard RT MAC protocol that is based on a cellular structure for periodic message transmissions in WSNs. On one hand, collisions are avoided by using FDMA among neighbouring cells so that inter-cell messages are reliably exchanged through capable routers. On the other hand, multi-cast intra-cell messages are exchanged inside fully-connected cells using EDF with implicit contention. However, this protocol is not suitable for event-triggered messages and dynamic topology networks because it is based on the periodic nature of the messages and fixed-size cells [175]. In addition, it needs specialized multi-channel radio sensor hardware and precise time synchronization.

Real-Time Protocols Based on Message Ordering

The authors in [180] presented a hard RT MAC protocol that gives priority to the messages in direct proportion to their waiting time caused by sleeping time or busy channels. Since the messages are arranged chronologically according to transmit timings, the older messages have a higher priority and can access the channel first if it is idle. However, this protocol is intended for one-hop networks rather than multi-hop ones.

3.10.2 Soft Real-Time MAC Protocols

Real-time Protocols Based on Medium Access Scheme

Virtual TDMA for Sensors (VTS) protocol [181], [182] is designed to be a soft real-time MAC protocol in WSNs. The message scheduling in VTS is a contention-free protocol for the participating nodes. The super-frame of VTS is shown in Figure 3.24 which is a cyclic frame with Nc cycles, where Nc represents the number of the participating nodes. The length of the VTS's super-frame is dynamically adapted with joining or leaving nodes. Each node is only allowed to send its data in its dedicated cycle (i.e. TDMA slot). The TDMA slot is divided into two periods: *Listen period* (i.e. $DATA$ and synchronization period) and *Inactive period*. The ratio of the listen period to the inactive period is called the duty cycle. At the beginning of each frame, the owner of the frame sends its control packet ($SYNC$ packet), which is used to coordinate the duty cycle, provide keep-alive indication, reserve the channel and discover new nodes. RTS/CTS [166] is used to avoid collisions when sending $DATA$ and $SYNC$ packets with fixed backoff contention windows.

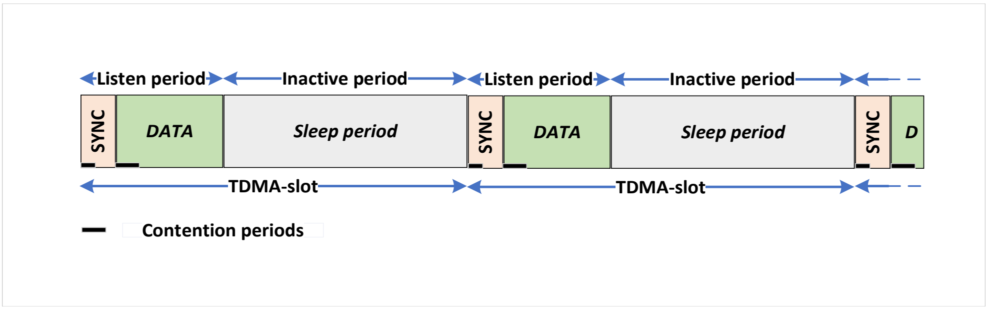


FIGURE 3.24: VTS superframe structure [181]

The super-frame in VTS is virtual, which means a node does not know the arrangement of its TDMA slot and the number of the available cycles in the super-frame. If a node leaves the network, the super-frame length is reduced automatically. A short contention period is used before the owner of the current TDMA-slot to let new nodes contend with the owner for access to the medium. Finally, to keep the deadlines, each node adjusts its duty cycle depending on its real-time application requirements.

VTS decreases the energy consumption and latency of the packet transmission when there are only few nodes. However, in multi-hop WSNs, VTS cannot properly work because each node generates a high packet rate for a given TDMA-slot and this leads to an increase in the energy consumption and super-frame length. Therefore, this protocol guarantees timeliness but it gets slower as the network grows. Moreover, it is not considered optimal in terms of the spatial channel reuse.

Another soft RT MAC protocol in [175] is designed for randomly distributed single-stream wireless applications, it depends on a feedback scheme to access the medium rather than contention schemes. To illustrate the problem of the contention scheme, the source node (N_0) in Figure 3.25 wants to send four packets ($P0$, $P1$, $P2$, $P3$) to a sink node (N_{10}). Firstly, N_0 competes to access the medium and wins at time t_1. Thus, N_0 starts sending packet $P0$ to node N_1. At the instant when transmitting packet $P0$ is completed, N_0 and N_1 compete to access the medium in order to send their packets. If N_0 again wins the competition at time t_2, then it starts sending packet $P1$ to N_1. In the scenario of real-time transmission, the packets should be transmitted in a defined order. Therefore, N_1 should access the medium in order to send packet $P0$ to node N_2 rather than N_0. Along the times from time t_3 until time t_7, we notice that the packet transmission sequence is unexpected and random.

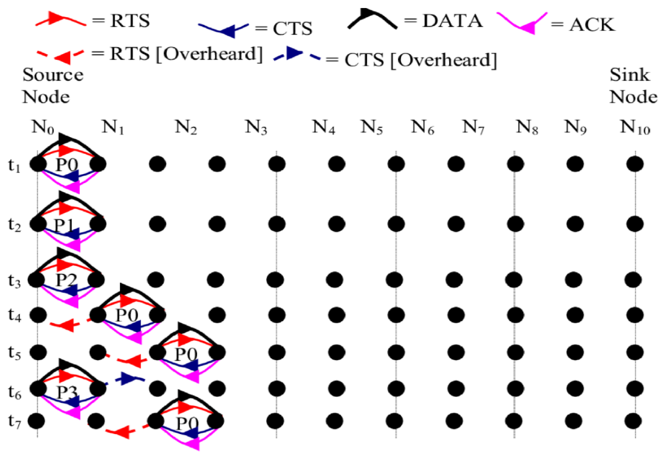


FIGURE 3.25: A sequence of the packet transmissions in a protocol, which is based on the contention scheme [175].

Therefore, a control packet named *'CC'* is used to change the boolean Clear Channel Flag (CCF) for each node that receives that *CC* packet. At the initializing phase, the CCF value for all nodes equals 1, meaning that all nodes can receive and send packets. On the contrary, the CCF of a node equals 0 when the node can only receive packets. The source node sends a *CC* packet to its neighbours, and *CC* contains an integer value called Clear Channel Counter (CCC). CCC at the source node is initialized to be 3 and decreases by one with a one-hop transmission. For example, if a node receives the *CC* packet with CCC equal 3, then CCC is decreased to be 2 and to node changes CCF in the receiver to be 0. As long as the CCC value is more than 1, the receiver node still forwards data without initializing it. In case the CCC value equals 0 or 1, the CCF value in the receiver node is changed to be 1 and the node can initialize data for transmission.

To illustrate the concept of the protocol, Figure 3.26 shows that source node N_0 sends packet $P0$ to node N_1 by using the RTS/CTS/data/ACK sequence. After getting ACK from N_1, N_0 sets its CCF to 0, meaning that it cannot transmit its data until switching its CCF to 1 again. As long as any intermediate node receives packet $P0$, it will set its CCF value to 0. This protocol is designed to send a *CC* control packet to the previous node, which is four hops away after the data packet arrives. *CC* is then periodically sent to the previous node, which is two hops away after the data packet arrives. Figure 3.26 shows that when node N_3 receives ACK from the next node N_4, then N_4 waits a designated time prior to forwarding packet $P0$ to node N_5. In the meantime, N_3 sends a *CC* packet to its previous node (i.e. node N_2) and sets its CCF to 0. N_2 then sends the *CC* packet to N_1 and sets its CCF to 0. N_1 then sends the *CC* packet to N_0 and sets its CCF to 1. At the instant that N_0 receives the *CC* packet from N_1, N_0 can send a new data packet (i.e. packet $P1$) to N_1, after that N_1 sends $P1$ to N_2 and waits there for the next *CC* control packet.

The disadvantages of the protocol are supporting only a single-stream and consuming high network resources because of periodic sending control packets.

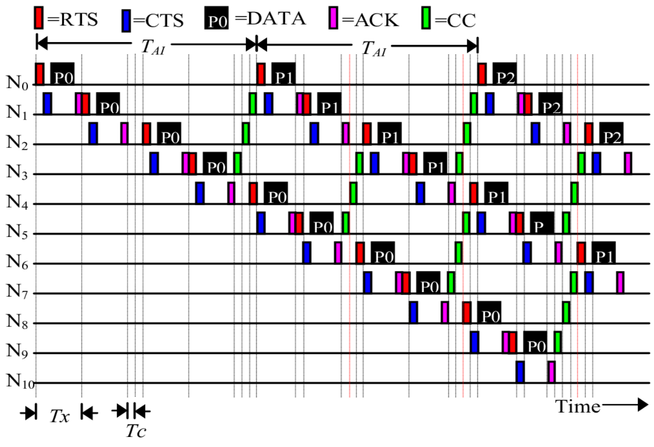


FIGURE 3.26: Timing diagram for the packet transmission in the protocol, which is based on the feedback scheme [175].

Real-time Protocols Based on Channel Reuse

The authors in [183] proposed the Channel Reuse-based Smallest Latest-start-time First (CR-SLF) protocol to increase the channel reuse in soft real-time multi-hop sensor networks. This protocol is based on partitioning the message in mobile WSNs into disassembled sets where each set is transmitted through the medium in parallel while avoiding collisions and interference. The message transmission in each set depends on the minimum Latest transmission Start Time (LST). CR-SLF considers the collision and deadline constraints, but the efficiency of message scheduling is degraded in large networks. In addition, the energy consumption is not considered and the proposed protocol requires precise position information for message scheduling.

Real-time Protocols Based on Direct-MAC

The Path oriented Real-time MAC (PR-MAC) [184] is a soft real-time protocol based on Direct-MAC (D-MAC) [185], which is proposed to guarantee the end-to-end delay by using the Bidirectional Pipelining Schedule (BPS) algorithm to eliminate the sleep delay in both directions of message transmission. In addition, the proposed protocol reduces the communication delay and prevents interference by using the multi-channel technique for message communication. PR-MAC requires time synchronization with a reference clock, where the performance can be improved by spatial channel reuse.

3.10.3 Hybrid Real-time Protocols

A hybrid scheduling protocol named Low-Power Real-Time (LPRT) is proposed in [186], which is based on dynamic CSMA/CA to transmit Best Effort (BE) traffic and TDMA for real-time traffic. The star topology is applied, where the base station directly connects and controls the traffic with its end nodes. Figure 3.27 shows the structure of the LPRT superframe that includes the Beacon (B) frame, CAP and CFP. CFP, in turn, is divided into a Retransmission Period (RP) and a Normal Transmission Period (NoTP). The beacon frame is used to send Resource Grant (RG) information from the base station to its nodes to determine which node is selected to access the medium. During the CAP period, BE traffic is transmitted using CSMA/CA while the CFP period is used to transmit real-time traffic by a node determined by its base station. RP and NoTP periods in CFP are optional periods used to retransmit the traffic due to a potential collision. LPRT decreases the energy consumption and the required overhead due to the applied star topology. Nevertheless, it is not considered as a suitable solution in large-scale wireless networks.

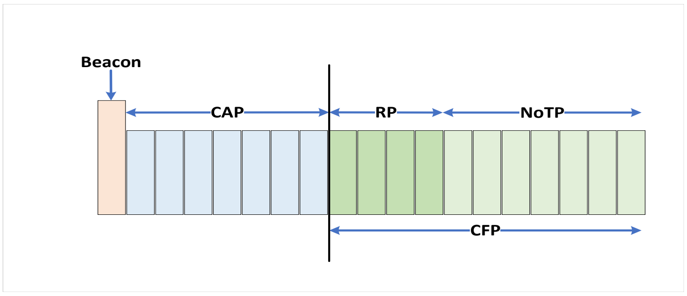


FIGURE 3.27: LPRT superframe structure [186].

3.11 Real-time Communication Protocols in Wired Systems

Many real-time protocols have been proposed for wired communication. For example, Controller Area Network (CAN) is a vehicle bus standard, which allows nodes to communicate with each other without a central master. Process Field Bus (PROFIBUS) [187] and Process Field Net (PROFINET) [188] are other widely used standards designed for data communication in automation technology. However, these types of protocols are implemented using dedicated hardware components, in addition, some disadvantages such as network size limitations and potential for undesirable connection.

Today Ethernet-based communication network become widely used because of the advantages that Ethernet technology offers like low latency, high throughput, low cost and reliability. In recent years, one of the most actively researched areas of industrial communication has been Ethernet-based communication systems. Nevertheless, the features of standard Ethernet technology are not sufficient to meet stringent real-time requirements. Therefore, extensions have been developed to solve this limitation. For example, a real-time extension for CSMA protocols is virtual time CSMA [163]. It grants access to the shared channel depending on variable-rate clocks to avoid collisions and offer better throughput and less latency. The authors in [189] rely on the sensing and collision detection features of the native CSMA protocol to provide priorities for real-time traffic by applying round-robin service among all cooperating hosts. In [190], a token is passed among all nodes to control access to the shared communication medium. Transmitted messages are classified as synchronous or asynchronous, where the synchronous messages are used for real-time applications. Another solution is using a master/slave architecture, where the master polls slaves to access the communication medium at any given time [191]. Hanssen, Ferdy, and Pierre G. Jansen [192] show a comprehensive overview of real-time communication protocols for wired communication.

3.11.1 TTEthernet

As mentioned above, standard Ethernet does not consider real-time requirements for deterministic, time-critical, and safety-relevant systems [193]. Therefore, Time-Triggered Ethernet (TTEthernet) [10], [194] has been developed to address these requirements by extending standard Ethernet fro safety-critical applications such as industrial control and automotive systems. TTEthernet synchronizes the nodes and schedules the packet switching. In addition, it partitions the bandwidth, provides fault-tolerance techniques to isolate and eliminate error expansion and ensures low latency jitter for messages. If no bounded latency traffic is sent, it operates as a full-duplex Ethernet to be compatible with IEEE802.3 and IEEE802.1 standards without any modification in the software or hardware designs. The fault-tolerance, communication and synchronization services have been certified for avionics. TTEthernet unifies critical and uncritical traffic into a single coherent communication architecture without interference. Thereby, TTEthernet supports different types of applications including soft real-time systems (e.g., multimedia) and safety-critical real-time systems. TTEthernet provides three different types of messages (i.e. classes) that are defined as follows:

- Time-Triggered (TT): the time-triggered messages are sent periodically and dispatched at pre-defined time-slots where the message scheduling limits the potential interference and collisions in the access to network resources. To guarantee the bounded latency and low jitter in TT communication, all participating network components have to be synchronized accurately with a global time. The message scheduling depends on the network architecture (i.e. topology), transmission speed, computing model and the specifications of the TT messages. Therefore, any modification in a network requires recalculating the scheduling tables.

- Rate-Constrained (RC): RC messages are transmitted sporadically according to the rate-constrained paradigm. Unlike TT messages, RC traffic does not depend on a global base time or Protocol Control Frames (PrCF) to maintain the synchronization. Instead, each transmitter uses a traffic shaping function for RC traffic to ensure that there is a minimum gap between any two successive messages. Each intermediate switch checks the sequence of the sent messages. Hence, the switch drops messages if they are sent too early (i.e. after less time than the minimum inter-message gap). The transmission latency in RC traffic can be bounded according to the network specifications and the allocated bandwidth.

- Best-Effort (BE): these messages are sent via First In First Out (FIFO) queues at output ports and there are no restrictions concerning the transmission rates. Thereby, there is no bounded latency or guarantee whether the messages can be sent. There is no knowledge of the delays, message delivery and whether the delivery satisfies any quality of service. BE messages have the lowest priority and are served by the remaining bandwidth of the network after sending TT and RC traffic [195].

3.11.2 Audio Video Bridging Protocol

Audio Video Bridging (AVB) [13] is a set of IEEE standards introduced by the AVB task group as a series of extensions to the standard IEEE 802.1 protocol. AVB addresses deterministic playback, low latency and low jitter of audio and video traffic using Credit Based Queuing (CBQ) and Priority Queuing (PQ) algorithms [196]. It is suited for infotainment and audio systems such as audio conferencing and in-vehicle entertainment. AVB defines the following standards to achieve the Audio Video (AV) traffic requirements.

- IEEE P802.1AS [11]: it is built on top of IEEE 1588 protocol to synchronize the clocks of all components with a reference global time base.

- IEEE P802.1Qat [197]: it is also known as Stream Reservation Protocol (SRP). This protocol is used to reserve the necessary network resources for AV traffic and maintain the reserved resources until AV traffic is completed. Two protocols are defined in SRP: The first protocol is called Multiple MAC Registration Protocol (MMRP), which is a registration protocol used to register a group of MAC addresses (i.e. listeners and talkers) on a set of switches. MMRP aims to confine the multicast AV traffic in specific areas of bridged LANs. The second protocol is the Multiple Stream Reservation Protocol (MSRP) which maintains the required QoS by providing the network components with the technique to conserve network resources and guarantee that AV traffic is successfully transmitted along the network. MSRP permits the talkers to advertise AV traffic across AVB switches and the listeners can register for the traffic. Thereby, MSRP controls the establishments and termination of the traffic. Moreover, MSRP supplies modules such as QoS to setup hardware resources that guarantee the requested bandwidth. gPTP domains are used to keep up with the latest information about the latency of the streams.

- IEEE P802.1Qav [196]: it is also known as Forwarding and Queuing Enhancements for Time-Sensitive streams (FQETS). FQETS ensures the bounded latency and low jitter during the transmission of the AV traffic by specifying shaping and queueing mechanisms. AV traffic is classified in AVB switches into three types: 1) SR class A: this type of traffic has the highest priority and it is sent over seven AVB hops in less than two milliseconds. 2) SR class B: it is the second-highest priority and AV traffic of class B is sent over seven hops in less than 50 milliseconds. 3) BE traffic: it is the lowest priority traffic. Any traffic that does not belong to either class A or class B, is classified as BE traffic.

As shown in Figure 3.28, the AVB switch contains several ingress and egress ports. The AVB switch presents an egress port model to perform and schedule BE and AV traffic. PQ and CBQ transmission algorithms are used to customize the traffic depending on the application requirements. The default PQ algorithm is used to insert BE traffic into different egress queues according to their priorities that are embedded in their Class-of-Service (CoS) field. The high priority BE traffic is sent through the best-effort egress port before lower priority traffic. Besides PQ, the CBQ algorithm is a credit-based shaping algorithm used in conjunction with the SRP protocol to guarantee low latency for AV traffic and regulate the data transmission for AV and BE traffic. Hence it performs the transmission through the AVB egress port. In the beginning, a new incoming frame is queued and its credit is adjusted to be zero. If the AVB egress port is not busy and no higher priority frames are waiting for their turn in transmission, then the queued frame is selected for transmitting. During the transmission time, the frame credit decreases at a rate of *SendSlope* until the transmission is complete. Otherwise, the frame's credit increases at a rate of *IdleSlope* while another queued frame is being sent. In the event that SR class A does not have frames for transmission, the lower-priority class (i.e. SR class B) that contains a queued frame has the ability to transmit. Hence, the CBQ algorithm exploits the unused bandwidth. This means that traffic, which uses the PQ algorithm (i.e. BE traffic), is able to use the unused bandwidth.

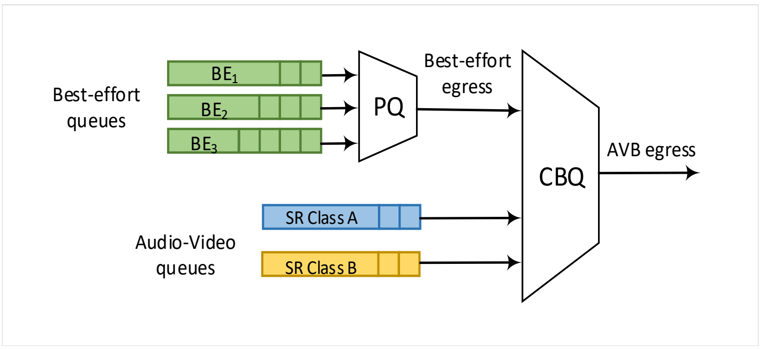


FIGURE 3.28: Egress port model of AVB switch [196].

3.11.3 Time Sensitive Networking

TSN [9] is a set of standard protocols produced by the TSN task group as an extension to the IEEE 802.1 standard. The TSN task group was established in 2012 based on the AVB task group to define techniques for real-time requirements of time-sensitive data. The TSN standards converge the real-time AV traffic and real-time control traffic of high availability and very low transmission latency in a TSN-enabled node. Despite spreading AVB in many fields like music venues and entertainment systems, AVB does not fulfil the requirements of critical safety applications [13]. AVB switches only guarantee low transmission latency and low jitter for AV traffic by providing different levels of priority and bandwidth reservation. In addition, the traffic transmission over AVB switches is non-preemptive, meaning that time-sensitive traffic cannot interrupt lower priority traffic and it has to stay in the waiting state until the ongoing traffic is completely transmitted. Hence, the credit-based shaping used by AVB switches does not satisfy the bounded transmission latency of the time-sensitive traffic. Therefore, the TSN standard protocols emerged.

The TSN protocols are developed to be compatible with Ethernet standards while being suitable for different safety-critical systems (e.g. automotive and industrial systems). First and foremost, IEEE 802.1AS [11] standard protocol is implemented and used to synchronize TSN-enabled clocks with a precise reference global clock. IEEE 802.1Qbv [15], [198], [199] adds TT traffic to the time-aware shaper, it permits temporal isolation and compositional TSN structures when TT traffic is scheduled along the network from the sender to the receiver. IEEE 802.1CB [16] introduces replicas of each frame and sends them over disjoint routes to provide seamless redundancy and prevent traffic loss in the event of failure. IEEE 802.1Qci [200] applies policy mechanisms by filtering incoming traffic in order to protect the TSN-enabled switches and TSN-enabled end nodes from a wide range of attacks such as Denial of Service (DoS) and man-in-the-middle attacks. The traffic filter in the 802.1Qci standard uses a time-aware Access Control List (ACL) as a matching rule to pass or drop traffic with specified IDs. Moreover, ACL identifies the size of the Maximum Transmission Unit (MTU) and the target egress port for each incoming message. 802.1Qcc [201] manages and fully distributes the configuration of the SRP protocol to replace the centralized configuration management with a decentralized one. It also enhances the stream reservation protocol and improves performance. The aforementioned properties of TSN, motivate us to extend the mechanisms and protocols to wireless systems as will be explained in details in Chapter 5.

Chapter 4

System Model

This chapter introduces in Section 4.1 the system model in the form of graphs that are used as inputs to the proposed algorithms mentioned in Chapter 2, which perform the task and message scheduling in a hybrid TSN. Section 4.2 displays the task and message scheduling model. Section 4.3 describes our hybrid modelling approach that we use in our exploration environment to test and analyse the proposed algorithms.

4.1 System Model

The system model used in the thesis is built from two TSN-based models (i.e., application and architecture models). The application model is described with specific task vertices and message edges, while the architecture model is a structured design of nodes and links. The application models provided are mapped to the architecture, the goal of mapping is to schedule the arrival of messages to the medium and to find optimized task schedules so that the task and message scheduling model meets the overall requirements mentioned in Chapter 2.

4.1.1 Formal Models Using Graphs

The architecture model is represented as a unidirectional graph $G_{Arc}(N, L)$ using the tuple $< N, L >$. Each vertex $n \in N$ represents a node, which is either a wireless TSN-enabled host ($h \in H$), a wirebound TSN-enabled station ($st \in St$), or a TSN-enabled router ($r \in R$), i.e., $N = H \cup St \cup R$. Each router r is defined either as an access point (ap), an Ethernet router (er), a wireless router (wr), or a bridge (b). The access point works as a wireless router besides acting as a gateway for wireless hosts. We assume that all wireless nodes are spread randomly in a Euclidean area and one available channel is shared for them.

Each edge ($l \in L$) is defined as a bidirectional physical link between nodes. It means that a second TT message can be sent over the same link while the first is being transmitted in the opposite direction. A wireless link is defined as a method of sending messages from one wireless node to another using a radio frequency over a limited space (communication range), while wired links generally use cables with fixed speed and known characteristics. All TSN nodes in the architecture graph are synchronized to the global time. Each node ($n \in N$) is presented by the tuple $< id, type, f\_rate, pos, mob, drift, max\_energy >$, where each attribute in the tuple is defined as follows:

- id: it denotes the identification number of the node.
- $type$: it denotes the type of the node (e.g. host, wired station, access point, bridge,…).
- F: it is the node failure rate, which is the inverse of the Mean Time To Failure (MTTF). MTTF is often calculated by dividing the total runtime of the node by the total of failures encountered.

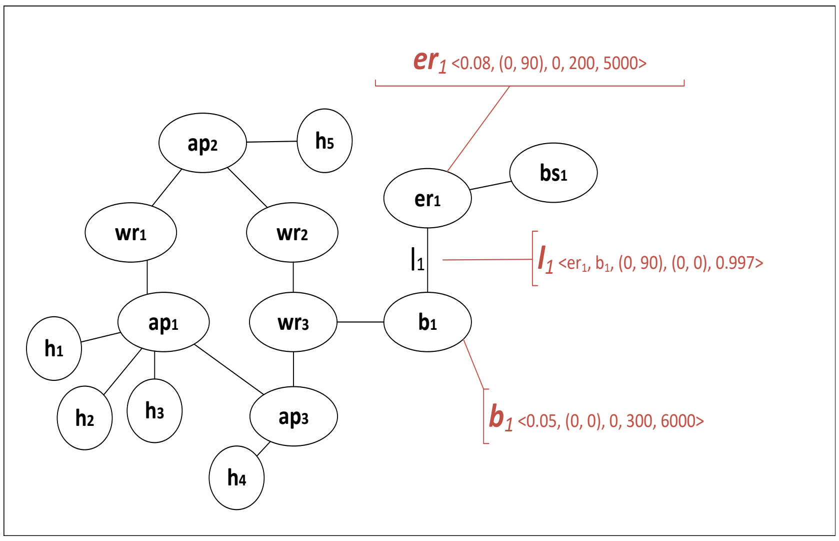
- *pos*: it indicates the position of the node and it is defined as x, y coordinates. The nodes are placed in a rectangular region, where the center of the region is the center of the coordinate system. If the node is in motion, the coordinates are used to find out the new location of the node and the distance travelled.
- *mob*: it is a boolean attribute that denotes whether the node is stationary or in motion.
- $drift$: it refers to the clock drift of the node's clock and it is defined in terms parts per million (e.g. 200 ppm).
- *max_energy*: it refers to the full energy of a wireless node.

Each link ($l \in L$) is presented by the tuple $< id, start, end, pos\text{-}start, pos\text{-}end, Rl_t, distance\_link >$, each attribute in the tuple is defined as in the following:

- *id*: it denotes the identification number of the link.
- *start* and *end*: they are defined as the identification numbers of the start and end nodes of the link, respectively.
- *pos-start* and *pos-end*: they are the coordinate positions of the start and end nodes of the link, respectively.
- Rl_t: it is the reliability of the link and it increases as the MTTF value of the link increases.

Figure 4.1 illustrates a simple architecture graph, which contains several nodes presented as vertices consisting of five wireless hosts in the range of $[h_1, h_5]$, three access points in the range of $[ap_1, ap_3]$, three wireless routers in the range of $[wr_1, wr_3]$, one bridge (b_1), one Ethernet router (er_1) and one base-station (bs_1). For instance, the failure rate of the bridge b_1 is 0.05. It is located at the center of the region, it is defined as a stationary node, its clock drift is 300 parts per million and the full energy of the bridge is 6000 joule.

Bidirectional wired and wireless links are used to connect different nodes. For instance, link (l_1) in Figure 4.1 starts from the Ethernet router (er_1) and ends at the bridge (b_1). The coordinates (0, 90), (0, 0) represent the positions of the start and end nodes, respectively. The reliability of the link (0.997) is assigned off-line and normally it is less for the wireless links than for the wired ones.



er_1 <0.08, (0, 90), 0, 200, 5000>
ap2
h5
bs1
er1
wr1
wr2
l1
l_1 <er1, b1, (0, 90), (0, 0), 0.997>
ap1
wr3
b1
h1
h2
h3
ap3
b_1 <0.05, (0, 0), 0, 300, 6000>
h4

FIGURE 4.1: An example for an architecture graph.

The application model is represented as a directional graph $G_{App}(T, M)$ using the tuple $< T, M >$. Each vertex $t \in T$ is defined as a real-time periodic task. Each edge $m \in M$ represents a message transmitted between the interconnected tasks. For instance, message m_{ab} represents the relationship between task $t_a \in T$ and task $t_b \in T$ such that t_a is the parent task to t_b (i.e. child task). These tasks and TT messages are represented as vertices and edges respectively. Each parent task can have multiple child tasks. Therefore, the proposed model supports multi-cast TT messages.

Each task ($t \in T$) is presented by the tuple $< id, et[H], en[H], P, tl, dl, can\_run\_on[] >$, where each attribute in the tuple is defined as follows:

- *id*: it denotes the identification number of the task.
- $et[H]$: it is a list that specifies the Worst-Case Execution Time (WCET) for the task on each host $h \in H$. The WCET depends on the processing capability of the host at which the task is executed.
- $en[H]$: it is a list that includes the consumed worst-case energy for executing the task on each host $h \in H$, where the consumed energy depends on the energy efficiency of the host at which the task is executed.
- *P*: it presents the period of the task. The periods of the sending and recipient tasks affect the period and the scheduling of the transmitted messages. More details about the number of times a sending task should be repeated before starting to send its message are given in Chapter 7.
- *tl*: it defines the top-level value of the task and denotes the cost of the longest path from the root task (i.e. a task with no incoming messages) to that task. The value of *tl* is the sum of the costs of vertices and edges of the longest path. Chapter 7 illustrates how to compute the top-level value for each task. A task with lower top-level value is considered to have higher priority for scheduling.
- *dl*: it defines the deadline of the task execution. Because the task deadline is fixed, a host should be carefully selected for every task to avoid deadline missed.
- $can\_run\_on[]$: it denotes a list of the available hosts on which the task can run.

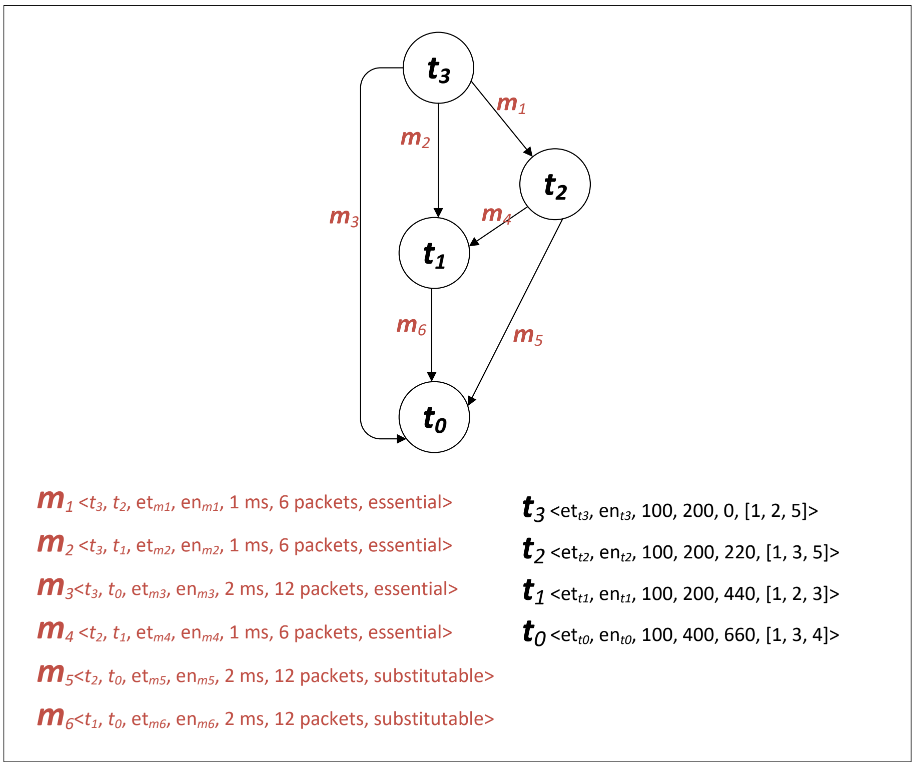
Each edge $m \in M$ is presented by a tuple $< id, se, re, et[R], en[R], sz, comm, c >$ and defined as follows:

- *id*: it denotes the identification number of the message.
- *se* and *re*: they are defined as identification numbers of the sender and receiver tasks of the communication message, respectively.
- $et[R]$: it is a list that includes the routing time of the message on each router $r \in R$. The routing time depends on the queuing and the routing capability of the router through which it passes.
- $en[R]$: it is a list that includes the consumed energy for routing the message on each router $r \in R$. The consumed energy depends on the queuing and the routing capability of the router through which it passes.
- *sz* and *comm*: they represent the amount of transmitted data (i.e. the message size) and the required time to transmit message, respectively. The required time depends mainly on the data rate (i.e. bandwidth).

- *c*: it presents conditional precedence restrictions and indicates whether the message from the sender task (i.e. *se*) is essential. In case the message is essential, the receiver task (i.e. *re*) can only start its execution as soon as it receives that message from its sender. If the message is substitutable, this means that the receiver task needs to receive at least one message from its sender tasks with the substitutable condition to start its execution. Based on this concept, the receiver task still executes correctly even if one of the sender tasks fails.

Figure 4.2 illustrates a simple application graph, which consists of four computational tasks that are numbered in the range of $[t_0, t_3]$ and presented as vertices. For instance, task t_3 has a WCET (et_{t3}) and consumed energy (en_{t3}) that depend on which host it runs. Its deadline is 100ms and its period is 200ms. The top-level of the task t_3 (i.e. the longest path) is 0, and task t_3 can be run on either host 1, host 2, or host 5.

Six periodic TT messages that are represented as arrows between different tasks $[m_1, m_6]$. For instance, message m_1 is sent by the task t_3 and received by the task t_2. Message m_1 has available routing time (et_{m1}) and consumed energy (en_{m1}) that depend on the router it passes through. The message size is 6 packets and the message needs 1 ms to transmit. The conditional feature (i.e. essential or substitutable) is assigned to each message off-line. For instance, task t_0 starts its execution provided that message m_3 from the sender t_3 is received (i.e. essential), besides receiving either m_5 or m_6 (i.e. substitutable) from tasks t_2 or t_1, respectively. It is worth noting that all tasks have precedence constraints, which means the execution of a task comes after the execution of all its parent tasks. The complexity of the application graphs increases with the number of computational tasks and the increase in messages exchanged between them.



m_1 <t_3, t_2, et$_{m1}$, en$_{m1}$, 1 ms, 6 packets, essential>

m_2 <t_3, t_1, et$_{m2}$, en$_{m2}$, 1 ms, 6 packets, essential>

m_3<t_3, t_0, et$_{m3}$, en$_{m3}$, 2 ms, 12 packets, essential>

m_4 <t_2, t_1, et$_{m4}$, en$_{m4}$, 1 ms, 6 packets, essential>

m_5<t_2, t_0, et$_{m5}$, en$_{m5}$, 2 ms, 12 packets, substitutable>

m_6<t_1, t_0, et$_{m6}$, en$_{m6}$, 2 ms, 12 packets, substitutable>

t_3 <et$_{t3}$, en$_{t3}$, 100, 200, 0, [1, 2, 5]>

t_2 <et$_{t2}$, en$_{t2}$, 100, 200, 220, [1, 3, 5]>

t_1 <et$_{t1}$, en$_{t1}$, 100, 200, 440, [1, 2, 3]>

t_0 <et$_{t0}$, en$_{t0}$, 100, 400, 660, [1, 3, 4]>

FIGURE 4.2: An example for an application graph.

4.2 Task and Message Scheduling Model

The scheduling algorithms in this thesis are executed off-line to optimize the task and message scheduling by mapping different application models to an existing architecture model. The message scheduling determines when a transmitted message is injected into the medium and how to allow several transmitted messages to access the channel without interference by allocating unique time-slots for each message. The task scheduling process aims to find the best available host at which a task can run. Therefore, during the task and message scheduling process several variables are continuously updated as illustrated in the following:

- When choosing a route (i.e. r) to send a message m, the energy consumption (i.e. en_n) and the execution time (i.e. et_n) in every node $n \in r$ are continuously updated as a result of scheduling the message on that route. Also, The injection time of the message m (i.e. m_{IT}) into the medium, the message end to end delay (i.e. m_{e2eD}) and the arrival time of the message at the receiver (i.e. m_{arrival}) are determined at the end of the scheduling.
- The energy consumption of scheduling the message (i.e. en_m) is the sum of en_n in every node $n \in r$. Likewise, the failure rate of scheduling that message (i.e. F_m) is the sum of the failure rate (F_n) of every node $n \in r$. The reliability of the message m (i.e. Rl_m) is computed based on the reliability of the nodes (i.e. Rl_n) and reliability of the links (i.e. Rl_l) that the message passes through.
- The consumed energy (i.e. en_t), and the failure rate (i.e. F_t) determined by the task and message scheduling are constantly updated until scheduling the required messages and the task schedule is obtained.
- The reliability of the task t (i.e. Rl_t) depends mainly on the reliability of the incoming messages and the reliability of its parent tasks. t_{rt} is the ready time for executing the task t and it is updated until arriving all required incoming messages.
- At the end of the task and message scheduling process, a host is selected from an available host set (i.e. $can\_run\_on[]$), and it is inserted into the $runs\_on$ attribute to execute the task.

Figure 4.3 represents a possible solution for the task and message scheduling problem by assigning the tasks from Figure 4.2 onto a set of hosts from Figure 4.1. Figure 4.3 shows the mapping of the application graph onto the architecture. The transmission time is neglected between two tasks scheduled to the same host. The size of the transmitted message and the network capacity are taken into account when calculating the transmission time between two tasks assigned to different hosts. Since the tasks do not specify which host to use, the applied algorithm to perform task and message scheduling must be optimized in order to obtain solutions that consider the constraints of the hybrid TSN systems.

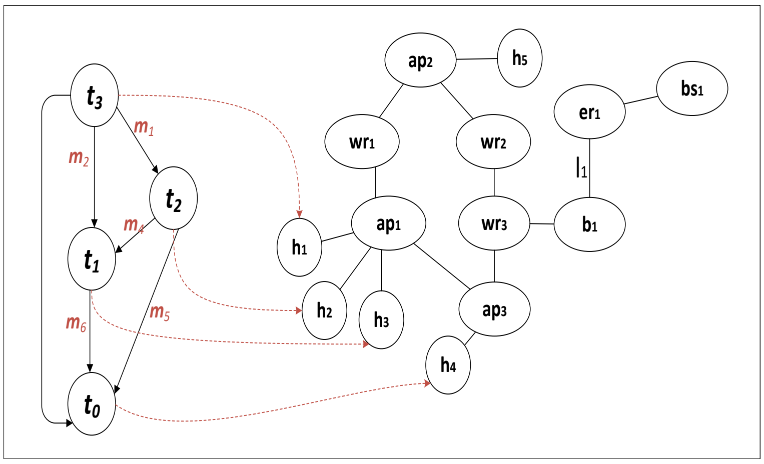


FIGURE 4.3: Example of mapping an application graph onto an architecture graph.

4.3 Description for Our Hybrid Modelling Approach

Figures 4.4 and 4.5 show two different architecture graphs (i.e. hybrid networks) that are later used as inputs to evaluate our proposed algorithms. Every graph represents two zones. The first zone is a wired TSN domain that contains wirebound station (*st*) used for controlling and monitoring operations, time-aware Human-Machine Interface (HMI) and Central Computing Unit (CCU), they are connected through five fully connected TSN-enabled Ethernet routers (*er*). The wired TSN domain is a static and stable network. The second zone is a wireless TSN domain that contains a group of three TSN-enabled wireless routers (*wr*), two TSN bridges (*b*) and five TSN access points (*ap*). In the architecture graph, the TSN bridges are used to combine the wired and wireless zones and reformulate the timing and control messages while transmitting between zones. Each TSN access point is connected with four mobile wireless TSN hosts (*h*) (i.e. laptops, iPods and mobile phones) or four fixed wireless TSN smart hosts (i.e. wireless stations). In our thesis, the wired clocks in the TSN wired domain have higher priority than in the TSN wireless domain. Higher priority gives preference to a wirebound station to be the grandmaster clock because wired clocks are more stable than wireless ones in terms of energy stability and not being exposed to surrounding influences that, in turn, negatively affect the TSN network performance.

4.3.1 Attributes of the Building Blocks

The computational resources in the described hybrid network are described by a set of general attributes as shown in the following.

Wireless nodes include mobile, fixed wireless hosts, and wireless routers sharing the following attributes: 802.11g is the Wi-Fi version that supports the communications of wireless local networks. The data rate is set to be 1-54 Mbps, channels are auto-assigned, transmit power is 0.1 watt, buffer size is 32 Kbytes, the beacon broadcast interval is 0.02 seconds and the maximum receive time is 0.5 seconds.

The Ethernet part of the bridges is built on 802.1w spanning tree protocol for loop-free logical topology Ethernet networks. The Bridge Protocol Data Unit (BPDU) and packet service rates are assigned to be 500 kilo packets per second. The addresses are auto-assigned and the operational mode is full-duplex.

Wired nodes include base-stations and Ethernet routers sharing the following attributes: the spanning tree protocol is 802.1w. BPDU and packet service rates are assigned to be 100 and 500 Kilo packets per second, respectively. The link model is 100BaseT, whereas, the link speed, link type, and duplex mode are auto-designed.

4.3.2 Services

The services that are supposed to be provided by the hybrid network are as follows:

- Clock synchronization:

  The clock synchronization establishes a global time in a hybrid TSN domain and forms the basis of the successful implementation of an application. Because of the deadlines of critical tasks in a real-time application, it is necessary to make sure that the task deadlines are not missed, therefore, the clocks of the hybrid system have to be accurate.

- Time-triggered communication:

  The periodic tasks generate time-triggered messages at regular intervals. Therefore, the proposed algorithms select the egress ports of the cooperating nodes to determine the instant of time for transmitting TT messages.

- Message scheduling:

  The egress ports must be selected so that all messages required for a task arrive before its deadline, especially for a wireless environment where many messages are sent simultaneously, causing potential interference. Therefore, each message must meet the deadline and interference constraints.

- Fault tolerant message scheduling and routing:

  In case of link or node failures, a fault-tolerance technique is needed for sending copies of the same message through several routes, which reduces significantly the impact of miss task deadlines and message retransmissions. The routing of the replicated messages plays also a critical role in the reliability of the system. Selecting the shortest routes with the least joint routers increases the capability of establishing a reliable and fault-tolerant system.

- Task scheduling process:

  Besides the message scheduling, selecting the best available host, on which the task can run, contributes significantly to implementing a real-time application in a hybrid system successfully. Therefore, this thesis targets to find an optimal mapping of the vertices of the task graph to available hosts so that the overall challenging requirements mentioned in Chapter 2 are met.

In this chapter, we support two topologies in the models, Figure 4.4 shows the wireless TSN domain in a ring topology, whereas Figure 4.5 shows it in a grid topology. All traffic travelling in both topologies is bidirectional which means any topology provides two pathways between any two nodes.

In the ring network, each wireless TSN relay (such as a TSN bridge, TSN router, or TSN access point) is connected to exactly two other nodes, creating a single continuous path for signals to travel through each node.

We summarize the advantages of a ring topology as follows:

- It does not require a central node to manage the connectivity between the nodes.

- It is quite easy to install and reconfigure since adding or removing a node requires moving just two connections.
- Point-to-point line configuration makes it easy to identify and isolate faults.
- Ring protection reconfiguration for line faults of bidirectional rings can be very fast, as switching happens at a high level, and thus the traffic does not require individual re-routing.

The disadvantages of a ring topology are summarized in the following points:

- One malfunctioning node can create problems for the entire network.
- Network performance may be impacted by node changes, additions, and moves.
- Communication delay is directly proportional to the number of nodes in the network.
- On each link connecting the nodes, bandwidth is shared.
- Limited routes between a sender and its receiver.

In the grid topology, each node is connected to two or more other nodes with a point-to-point bidirectional link. As a result, it is possible to utilize a portion of the physical complete connectivity and redundancy of a mesh topology without having to incur the cost and complexity of connecting each node individually.

The grid networks have advantages over the ring ones, as in the following:

- Easy scalability, this means you can quickly and easily change the size of the network.
- Resistant to problems, this means this type of topology helps keep the network running even if a failure occurs.
- Sender has many alternative routes to send its traffic to its receiver.

In contrast, a network designer wants to be aware before committing to this type of network, as in the following:

- Increasing the required network resources.
- Initial network setup can be complicated.

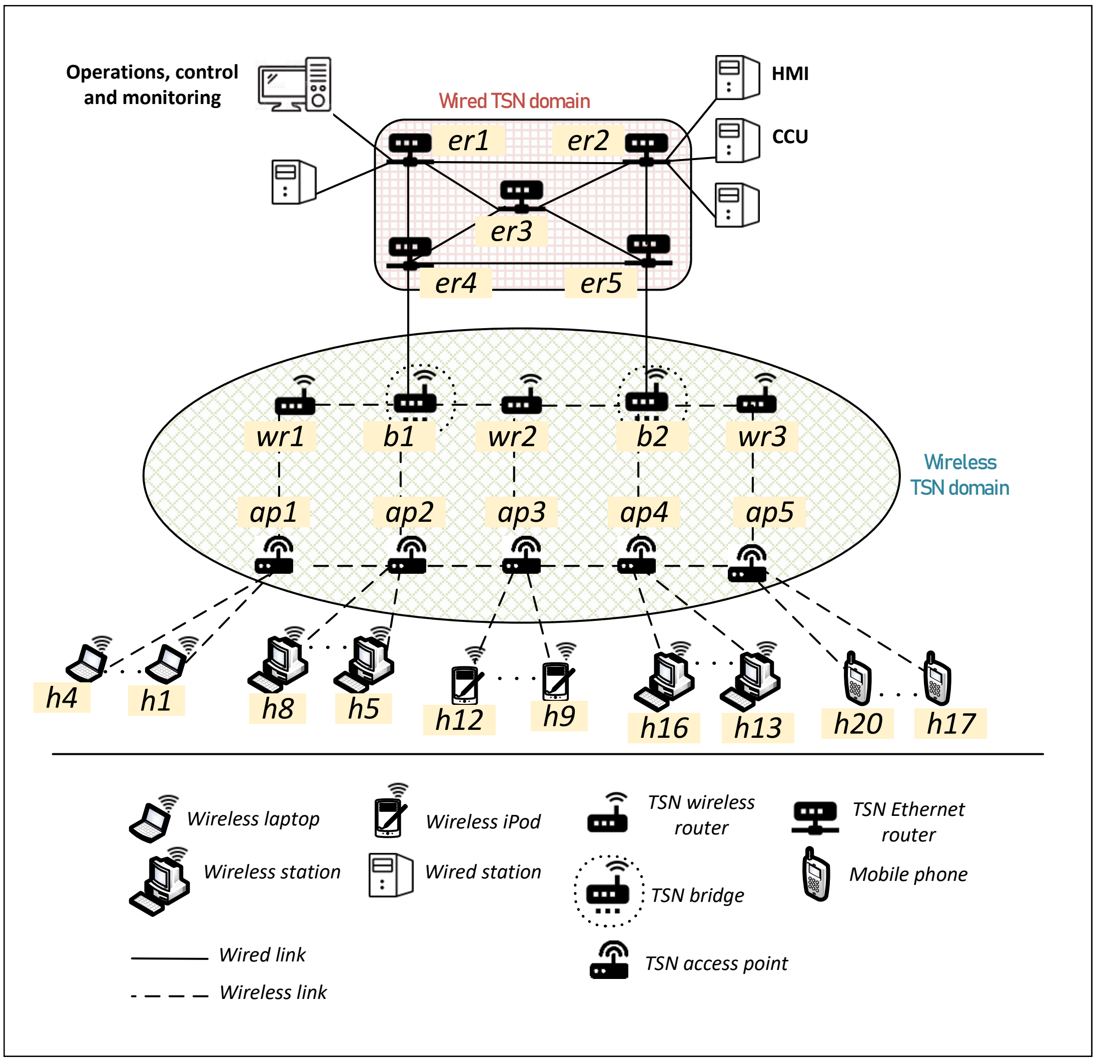


FIGURE 4.4: The grid topology used in our system model.

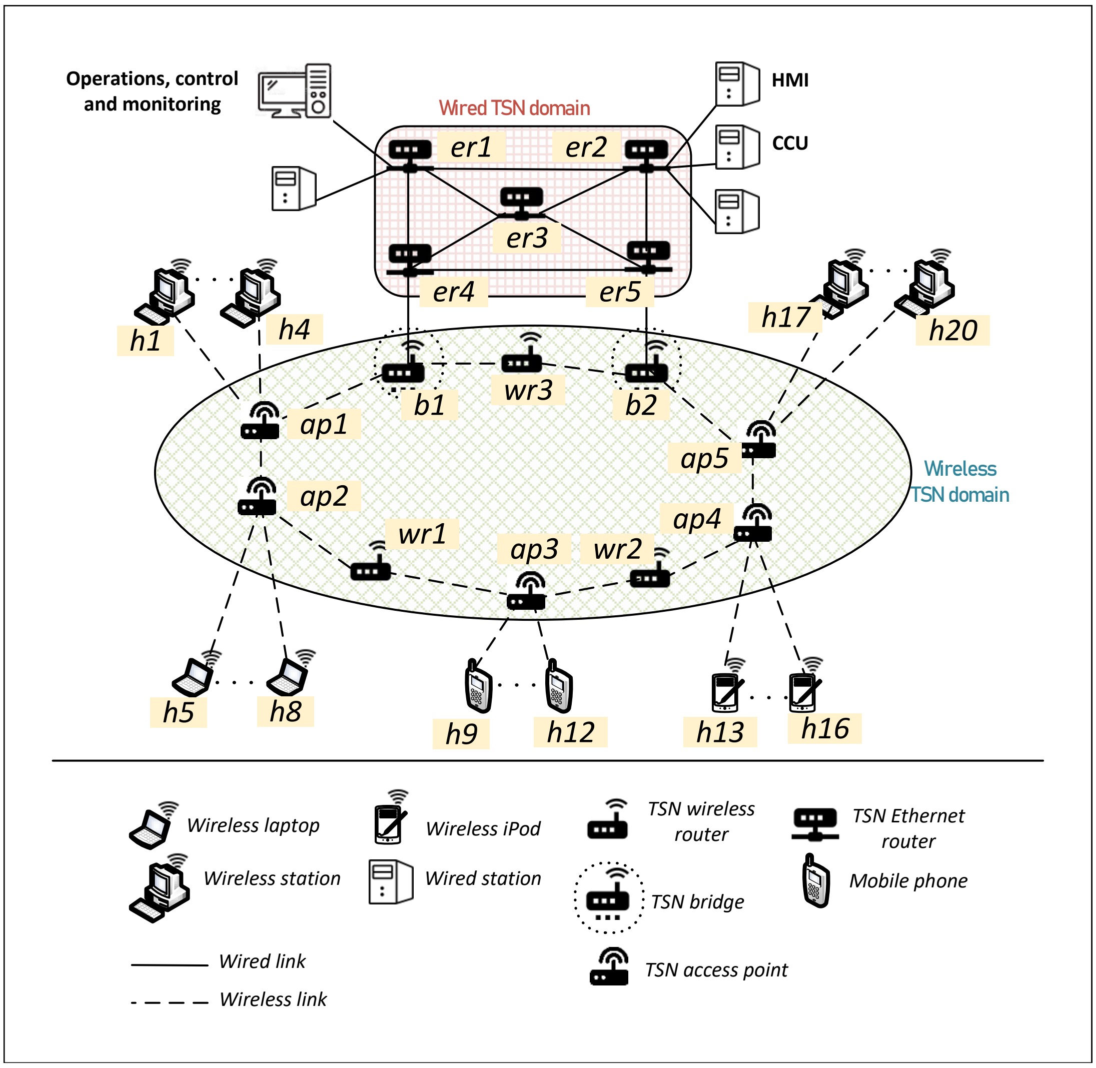


FIGURE 4.5: The ring topology used in our system model.

Chapter 5

Extending TSN Capabilities over Hybrid Systems

5.1 Introduction

Wireless and wirebound networks are widespread in dependable and scalable real-time applications. More and more applications combine wireless and wirebound nodes in a single system. In such systems, the real-time MAC protocols addressed in Chapter 3 may face several challenges such as 1) precision of the clock synchronization. 2) support for dynamic topologies and event-triggered applications. 3) channel assignment problem in wireless networks. 4) management of multi traffic types. 5) energy efficiency despite control packets.

TSN as explained in Subsection 3.11.3, is designed to fulfil the real-time requirements in wired (i.e. Ethernet-based) environments and offers services that are not covered in AVB systems including deterministic latency and limited jitter for time-critical applications. For time-sensitive streams, the IEEE 802.1 TSN task group creates standards to achieve correct time distribution and timeliness with high reliability.

The standards of TSN make it an ideal solution to address the challenges and problems that are faced by real-time MAC protocols in wireless environments. Thereby, this motivates us to implement and extend TSN wire-bound systems for hybrid environments in order to include the capabilities of TSN features. Due to the nature of the medium access, channel competition, asymmetry communication delay and packet error rate, the TSN standards have to be improved for mapping TSN capabilities to wireless systems. In addition, not all wireless technologies are configured to support the TSN features, but 5G and IEEE 802.11/Wi-Fi standards are suitable base technologies for TSN. To achieve that, this chapter discusses how to extend TSN capabilities for wireless systems.

The rest of the chapter is structured as follows: Section 5.2 discusses IEEE 802.11/Wi-Fi and TSN support. This section details time synchronization, traffic scheduling, traffic shaping and fault tolerance based on 802.11. Section 5.3 presents modelling of the hybrid TSN with a simulation framework using the OPNET network simulator.

5.2 IEEE 802.11/Wi-Fi and TSN Support

According to the limitations of 802.11 such as random backoff and multi-hop problems mentioned in Subsection 3.9.3, extensions become necessary to fulfil the requirements of real-time applications over wireless technologies. To achieve that, the standards of TSN require the support of 802.11 MAC/PHY to work properly as demonstrated in this section.

5.2.1 Time Synchronization (802.1AS) over 802.11

Section 3.7 gives an overview of 802.1AS and the selection of the grandmaster as a source reference clock. Although 802.1AS has already been developed for LANs [11], our thesis proposes an extended protocol [27] to deal with the challenges of defining a profile of 802.1AS for 802.11 networks. Figure 5.1 illustrates an example for the time synchronization referring to a source reference time (i.e. grandmaster clock) between wired (Ethernet) and wireless TSN domains by enabling 802.1AS over Ethernet and 802.11. To achieve that, 802.1AS timing packets (i.e. *SYNC*, *Announce* and P2P request/response messages) have the priority when accessing the channel to perform the TT data traffic. Moreover, the timing packets that are transmitted through the wireless medium have to take into account the asymmetric delays due to variable data rates, deterministic and random delays and outlier path delays due to unpredictable data collisions. To select the best grandmaster clock, stationary nodes with an Ethernet connection get higher priority than the fixed or mobile wireless nodes. The reason is that the network topology and the packet transmission are more stable compared to the nodes located on the wireless side. More details about extending 802.1AS for wired and wireless networks will be shown in Chapter 6.

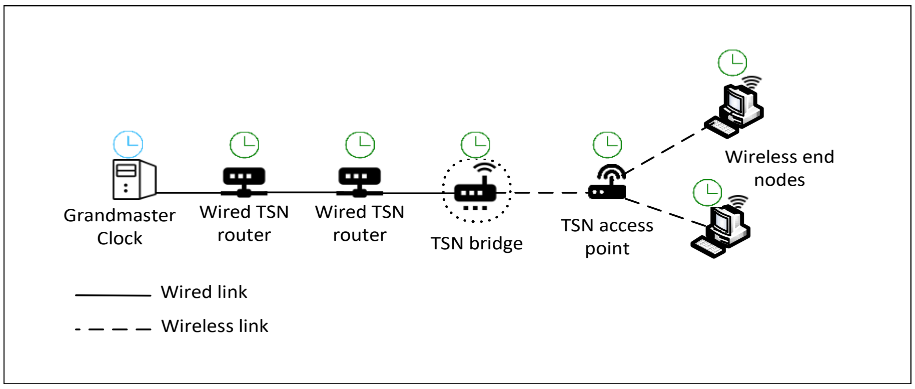


FIGURE 5.1: Time synchronization between wired and wireless TSN domains by enabling 802.1AS over Ethernet and 802.11 [202].

5.2.2 Traffic Scheduling and Shaping (802.1Qbv) over 802.11

One of the limitations of 802.11 is the random backoff mechanism that results in random latency in CSMA/CA. To eliminate the random latency and ensure that time-sensitive TT traffic can be delivered with low latency and high reliability, a deterministic mechanism that can classify and schedule the traffic according to deadline constraints becomes necessary.

The IEEE 802.1Qbv [15], [199] standard is an off-line scheduling mechanism that classifies and shapes all types of traffic and schedules them according to the characteristics of the traffic and the distribution of the TSN-enabled nodes. It is basically used to classify the time-sensitive traffic in TSN-enabled nodes and give them higher priority than other types of traffic (i.e. BE and AV traffic) during the transmission through the wireless access medium. IEEE 802.1Qbv supports preemptive scheduling which means that the low priority traffic is interrupted in case of scheduling higher priority traffic. Thereby, high-priority traffic is delivered before deadlines.

The egress port of a *WirelessTSN* intermediate node (i.e. a TSN-enabled bridge or a wireless router) has its own queues, and each queue has a priority value used to classify the incoming messages. Figure 5.2 illustrates eight ports sorted from the lowest priority at the top of the figure (i.e. BE_0) to the highest priority at the bottom of the figure (i.e. TT_7). TT traffic is enqueued in the higher priority queues. The following queues are used to prioritize different classes of AV messages. BE messages are enqueued in the remaining queues. The priority for messages of the same type depends on the value of a tag called Priority Code Point (PCP) which is contained in each message and used to forward it to the corresponding queue. The transmission procedure starts at the highest priority queue that has messages and the gate of the queue is opened. We assume that the gates have mutually exclusive patterns meaning that no two gates can be opened at the same time. When a message is preempted by a higher priority message, the transmitting message is interrupted before initiating transmission of the higher priority message (e.g. TT traffic), such that the higher priority message can immediately initiate its transmission. After finishing the transmission, the preempted message resumes its transmission.

The gates of the queues are controlled to be opened and closed by a Time Aware Shaper (TAS), which applies a Gate Control List (GCL) to dictate the state of each queue's gate at a defined time. It is worth noting that the GCL is configured separately for each egress port. For instance, the GCL in Figure 5.2 shows a set of queue masks that identifies the status of each queue's gate at an output egress port. The GCL of the 802.1Qbv-enabled node starts with t0: 10000000, which means that 't_0' is the start time of the GCL. Thereby, the enqueued TT traffic in Queue # 7 is allowed to be transmitted from its queue so that the first index of the GCL is opened '1' and the other gates of the queues are closed due to their corresponding indexes in the GCL being closed '0'. At the next start time (i.e. t_1), the AV traffic in Queue # 6 is allowed to be transmitted because the second index of the GCL is opened and the other indexes are closed. The opening mechanism of the gates of all queues continues until the end of a predefined period.

The configured period of time in the structure of a port-specific GCL depends on the periods of all traffic that is designated to that 802.1Qbv-enabled node. To be more specific, the period of a port-specific GCL is computed according to the Least Common Multiple (LCM) of the periods of the traffic passing through that node. To avoid interference and to make the scheduling more deterministic, the GCL is computed offline for TAS in each TSN-enabled node. This means that every node has prior knowledge and it is aware of the network topology, the transmitted streams and their attributes (e.g. transmitter, receiver, period and deadline). Moreover, TAS has to know the structure of the output queues to facilitate the classification and shaping process. This guarantees that an active TT queue is the only available queue with its gate opened and the gates of all other queues have been closed. Thus, each TT message reserves time slots, similar to the principle of TDMA, to eliminate the latency caused by nodes competing for access to the wireless medium. Thereby, scheduled operation are performed with a minimum bounded latency.

An enqueued AV message is transmitted if the following conditions are satisfied 1) the gate of the AVB queue is opened, 2) the Credit-Based Shaping (CBS) of the AV message permits it to be transmitted and 3) no TT traffic is enqueued. Thus, the combination with IEEE 802.1Qat stream reservation protocol, which reserves the resources for specific traffic, allows the transmission of the AV messages whenever the credit value is zero or positive. The CBS concept enables the transmission of all enqueued AV messages and prevents low priority messages from getting stuck and waiting for a long time. Therefore, CBS performs and shapes the AV messages. At the initialization, the credits of all AV messages are set to zero. The credit of a message is decreasing with a $SendSlope$ during the message transmission. The credit is frozen if the message is interrupted by a higher priority message. The credit increases with an $IdleSlope$ if the message is in the waiting state. The message transmission is only initiated if the credit is zero or positive and the gate is opened. If no messages are in the queue, the credit is reset to zero.

An example of how CBS works is illustrated in Figure 5.3 where a wireless bridge has a TT message, two AVB queues, as well as a BE queue. Figure 5.3 shows a time-line for the transmission at the bottom, where a rectangle is a part of a message (fragment), with the width representing the transmission time. The values of the credit over time for AVB streams are presented above the time-line. Let us explain the transmission of all streams using the events e_0 to e_{10}.

At event e_0, message m_1 which arrived at AVB1-Queue starts transmission. Its credit (i.e. Credit AVB1) is initialized to zero and decreases according to the $SendSlope$. At event e_1, the credit of AVB1-Queue is negative while the credit of AVB2-Queue is positive because it waits for the completion of the transmission of m_1 ($IdleSlope$). Therefore, at e_1, AVB2-Queue opens the transmission for m_3. Message m_2 arrives and is enqueued in AVB1-Queue. During the period from e_2 to e_3 no transmission occurs for m_2 because the gate of AVB1-Queue is closed. AVB1-Queue opens the gate to transmit m_2 at e_3. During the transmission time, messages m_4 and m_5 arrive and are enqueued in AVB2-Queue and BE-Queue, respectively. At event e_4, message m_6 arrives at TT-Queue and immediately the gated is opened to transmit m_6. The credits of AVB-Queues are frozen during this period. At event e_5, transmission of m_6 finishes, the gate of AVB1-Queue opens and the transmission of m_2 is resumed. At event e_6, the enqueued m_5 is transmitted and the credits of AVB-Queues are increasing. At event e_7, the transmission of m_5 finishes. During the period from e_7 to e_8 no transmissions for messages m_4 and m_7 takes place because the gates of AVB-Queues are closed. During this time, the credits of the AVB-Queues keep accumulating during until event e_8. At event e_8, AVB2-Queue transmits m_4 and its credit decreases again, while the credit of AVB1-Queue keeps increasing. At event e_9, AVB1-Queue resumes decreasing its credit because of sending m_7. AVB2-Queue returns to accumulate its credit according to the $IdleSlope$ until event e_{10}.

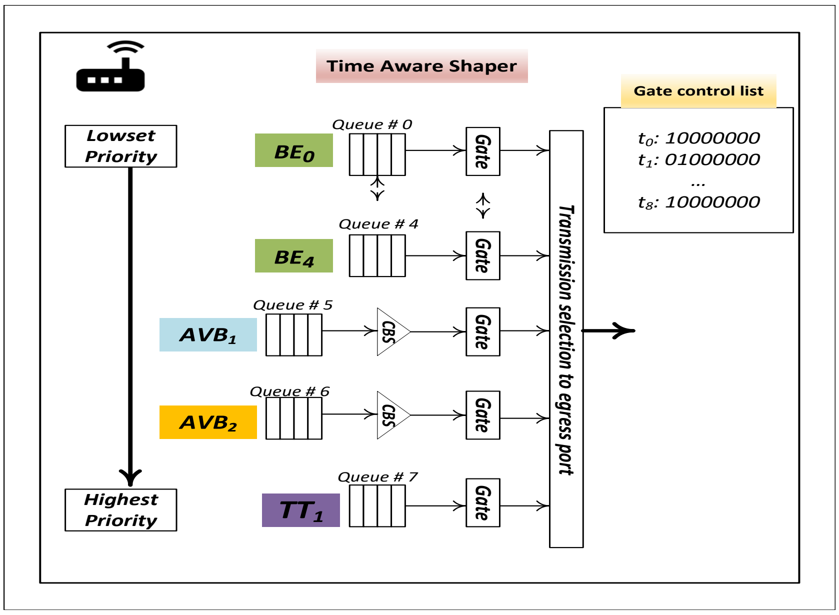


FIGURE 5.2: A simple view of an 802.1Qbv-enabled node [198], which transmits different types of traffic at an output egress port.

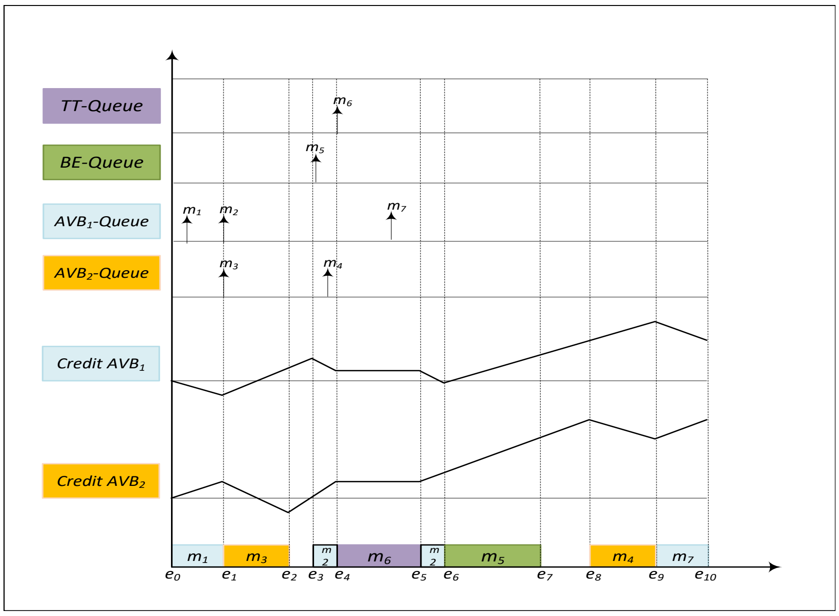


FIGURE 5.3: Example for traffic shaping and scheduling in an 802.1Qbv-enabled node.

5.2.3 Fault Tolerance Management (802.1CB) over 802.11

The IEEE 802.1CB [16] standard – known as Frame Replication and Elimination for Reliability (FRER) - foresees sending duplicated copies of a message over disjoint redundant routes. The proactive seamless redundancy improves the reliability of the message transmissions, especially for sensitive applications that do not tolerate message losses. Each TSN-enabled wireless node that has the message replication capability generates sequence numbers for each outgoing message. Each copy is then forwarded through a different route, and in case of a route failure, the message reaches the destination through another redundant route. The message replication can use the traffic type and the route information to minimize network congestion. The redundant routes have to contain disjoint intermediate nodes because a failure of a joint node would lead to the failure of the routes that contain that node. To avoid network overloading at an intermediate node, all following messages are eliminated and the first received message is forwarded through other redundant routes toward the next node. At the destination node when a message is received through a route, all following messages are eliminated and the first received message is processed.

FRER consists of different functions, which are described in the following.

- Sequencing function: at the sender, this function generates a unique identification number (i.e. sequence number) for each transmitted message.
- Sequence recovery function: this function is used at the egress port to examine the messages that come from different ingress ports and eliminate the duplicated messages. Moreover, the egress port reorders the messages before dispatching them into the medium. To achieve that, each FRER-enabled node has its own history to save the sequence numbers of the received messages. This is useful to discard message that are out of the expected sequence range.
- Individual recovery function: every message passed to an ingress port is checked and eliminated if it is a duplicate message with the same sequence number of a message that has been received previously.

- Message splitting function: this function creates one, two, or more copies of a message according to the number of the extracted disjoint redundant routes. Thereby, these copies are transmitted to the destination via separate routes.
- Encoding and decoding sequencing: this function is used to encode the generated sequence numbers by the sequencing function. At the receiver, a decoding process is implemented to retrieve the original sequence number provided that the encoding process is known to all wireless intermediate and end nodes. Otherwise, a problem can occur during the sequence number retrieving mechanism.
- Message identification function: at the FRER-enabled bridge node, any message is identified by a message identifier to determine to which stream the received message belongs (i.e. TT traffic, AV or BE stream). To do that, the identifier examines the source MAC address and Virtual LAN_id fields of the received message. Thus, the received message is classified and shaped at the egress port by implementing TAS.

The design of the FRER-enabled node is flexible. To be more specific, FRER and non-FRER nodes can coexist in the same network provided that the non-FRER nodes do not affect the TSN constraints.

Figure 5.4 depicts how FRER operates. The sender (i.e. the leftmost box) sends its message (e.g. TT message) to its access point, the access point generates and encodes a sequence number and embeds it for each outgoing copy of the message. At the next wireless routers where two copies of the message are received, each ingress port checks the sequence numbers and eliminates the duplicates. At the egress port, the copies of the same message are eliminated if they come from different ingress ports while the first copy of the messages is forwarded through the redundant routes toward the next wireless router. The same process of elimination and replication of the message is repeated until arriving at the access point that is connected with the receiver. The receiver's access point (i.e. the rightmost box) passes the first copy of the message to the receiver host. In this setup, the transmitted message is delivered and multiple failures can be overcome. To further explain, Figure 5.5 shows two path failures on the paths between the sender and the receiver of the message. Nevertheless, the message is successfully received due to the adoption of the principle of replicating the message and use of available paths that in turn lead to overcoming the failures in the network. In the network without FRER in Figure 5.6 two failures lead to preventing the message delivery.

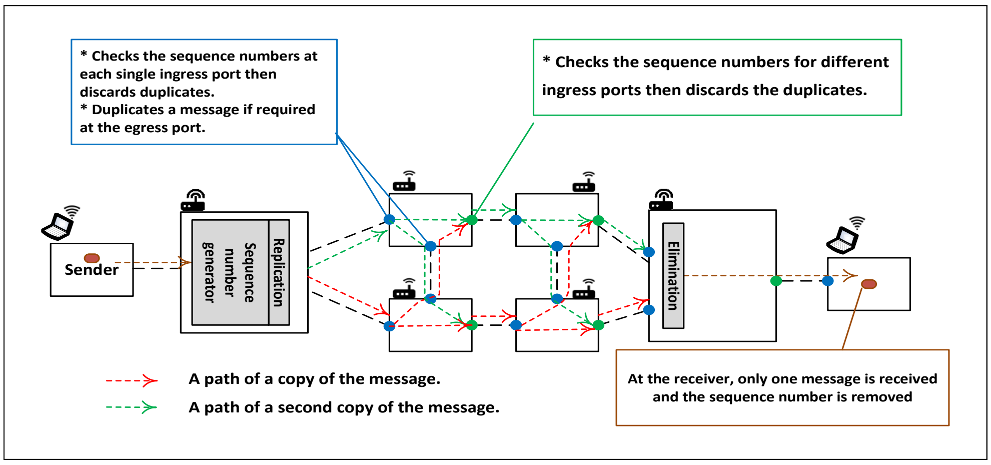

* Checks the sequence numbers at each single ingress port then discards duplicates.
* Duplicates a message if required at the egress port.
* Checks the sequence numbers for different ingress ports then discards the duplicates.
Sender
Replication
Sequence number generator
Elimination
A path of a copy of the message.
A path of a second copy of the message.
At the receiver, only one message is received and the sequence number is removed

FIGURE 5.4: Operation of message replication and elimination at FRER-enabled nodes.

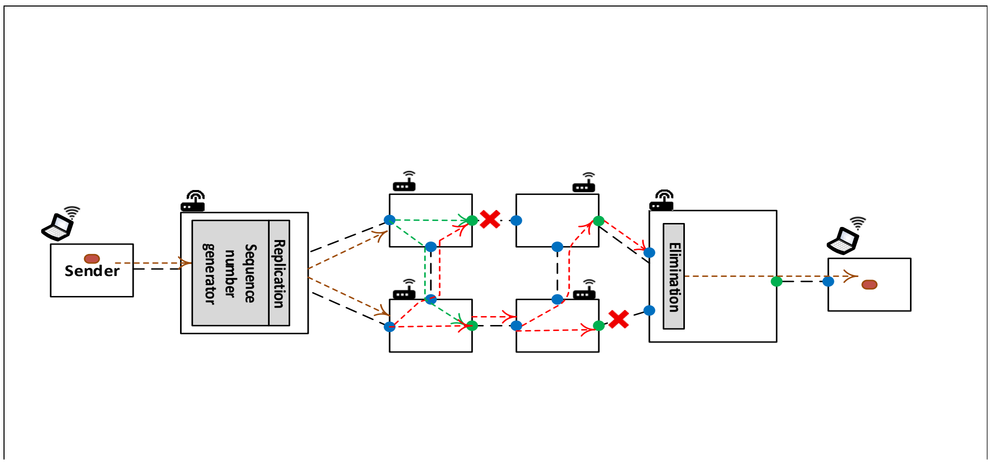


FIGURE 5.5: Simple example showing the fault tolerance and guarantee of message delivery by using FRER.

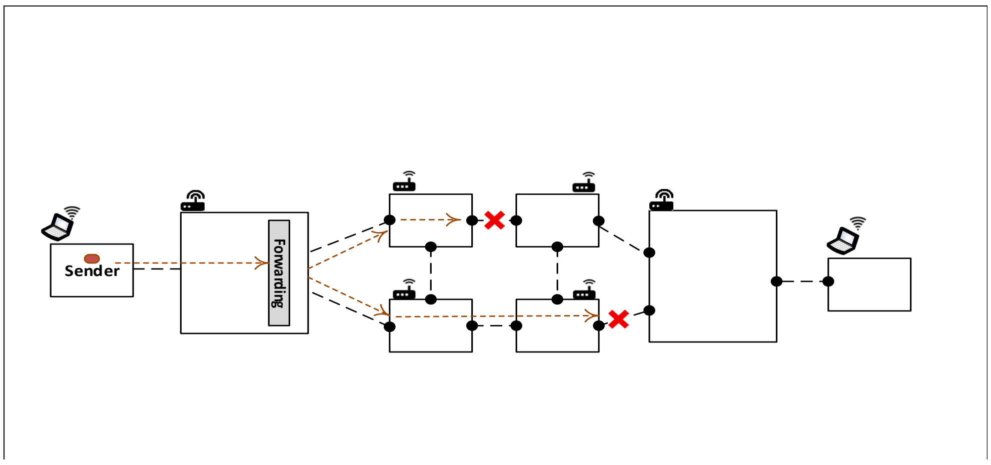


FIGURE 5.6: Simple example with a message failing arrive in a network that does not support FRER.

5.3 Modelling of Hybrid TSN Networking in a Simulation Framework

The implementation of TSN functions in hardware is a time-consuming and costly process. Therefore, modelling and simulation frameworks can be used to create an environment for emulating and imitating the configurations of the real hardware nodes. This section presents a TSN simulation framework for a hybrid environment that includes wired and wireless components. The described implementation is modular, which was integrated and improved as an extension of the TSN Ethernet-based network in [203]. The time-aware shaping (i.e. 802.1Qbv) and fault tolerance (i.e. 802.1CB) capabilities of TSN are extended into the existing models used by the standard Ethernet and wireless models with adding necessary functionalities to fulfil the requirements of the time-sensitive applications.

To provide the functionalities of TSN, the simulation model of a TSN-enabled node implements the TSN clock synchronization, which also provides an opportunity to evaluate the synchronization accuracy of real-time TSN solutions. Therefore, each time-aware node requires a local clock synchronized accurately with a source reference clock as also required for the deterministic delivery of time-triggered traffic.

Evaluation of TSN functions using simulation tools before starting the manufacturing process of TSN-compliant components is useful due to the high cost and manufacturing time, which are required to reach the final approved components. Therefore, network simulators are considered as time and cost-efficient options for analysing and debugging as part of the verification and validation of the TSN functions. The simulation tools have the features to evaluate the tested network through different parameters like end-to-end-delay and throughput. Moreover, the networking scenarios can be applied by using various topologies and considering different faults.

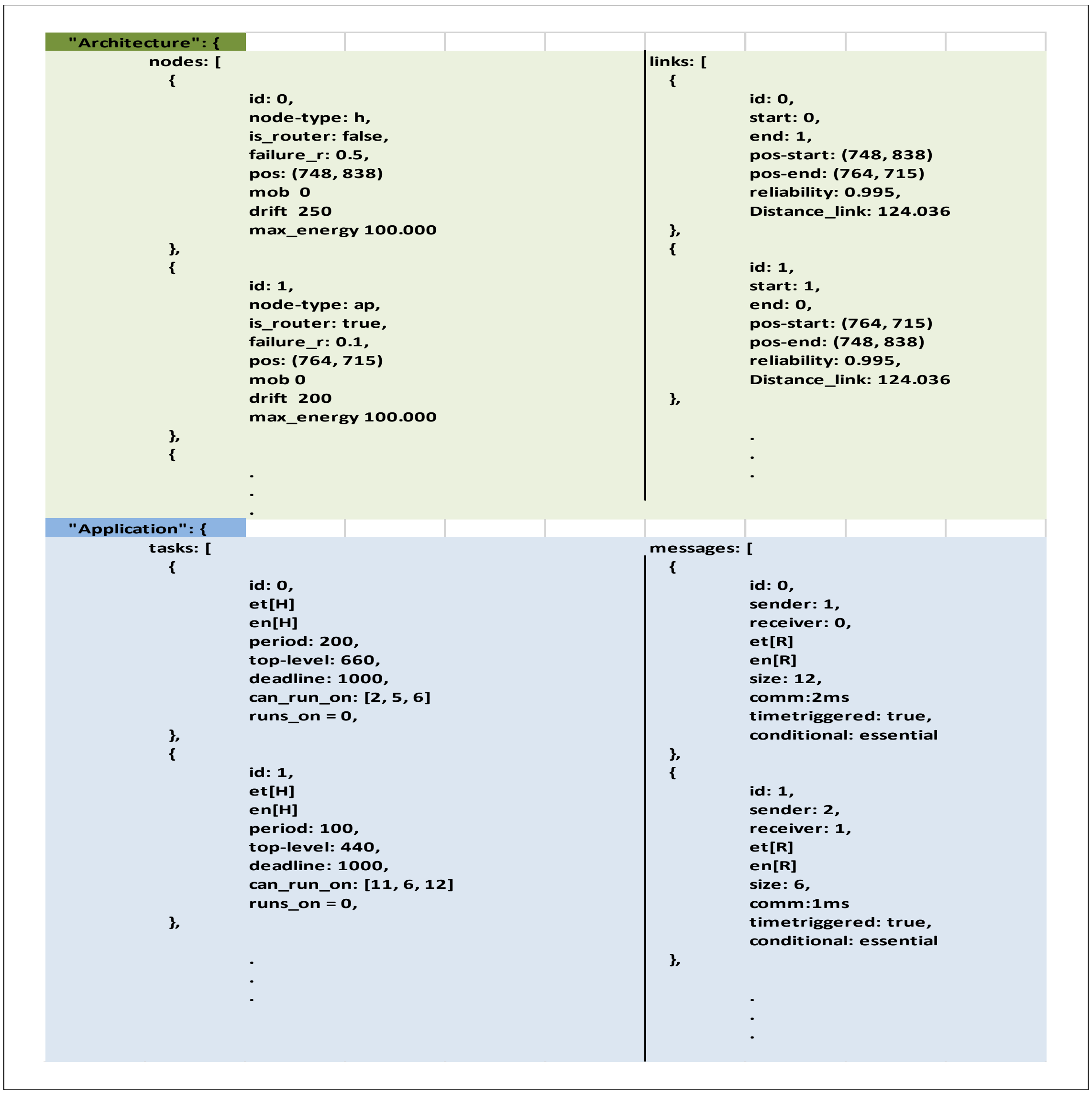
There are several simulation tools designed for research, industry and teaching purposes. The Network Simulator-2 (NS-2) [204], NS-3 [205], Qualnet [206], OMNeT++ [207] and OPNET (Riverbed) [14] simulators are the most common network software simulation tools. NS-2 and OMNeT++ are both free and non-commercial open-source primarily built for network frameworks. Qualnet is a commercial software used especially for scientists and network planners to mimic the behaviour of scalable network topologies, but it is not free and source codes for some files cannot be modified [205]. On the other hand, OPNET is a commercial tool with discount licenses for educational usage. It is a discrete-event simulation tool and it is widely used in several academic and scientific institutes because of continuously updated models for many protocols. OPNET provides to its users various scenarios and predefined node models which can be easily imported from the OPNET library to enable the required functions. Besides that, Graphical User Interfaces (GUIs) let the users interact with the network models and easily access the message structures and source code at different OSI layers. OPNET also allows its users to modify or build process modules that are used to implement user-defined network models.

The simulations in OPNET occur at hierarchy levels and the user can investigate specifically what is going on at that level. The modelling development tool in OPNET consists of three editors: network, node and process editors. The **network editor** is used to build the network model by defining the nodes and connecting links. The **node editor** is used to edit the node model, which may include processors, sinks, queues, and external modules among other types of modules. Both static links and packet streams can be used to connect the modules. The **process editor** is used to define and control the processes inside the node's modules, which are designed as State Transition Diagrams (STDs). The code in the states is written in Proto-C language and each state is coloured either in green if it is a forced state or red if it is an unforced state. The change between the related states is denoted by an arrow. To analyse the behaviour and performance of the system, simulation and execution tools are implemented to execute the simulation sequence and change the simulation behaviour. A probe editor is used to collect static and animated outputs in different formats. Finally, result analysis tools use filters to collect and display the results in graphs after completing the simulation execution.

Based on the above, the OPNET simulator is used in our thesis for modelling the network infrastructure of the hybrid TSN system because of the support of OPNET for debugging and performance analysis for a wide range of wired and wireless systems. To be more specific, the OPNET simulator is considered a suitable and flexible user-defined environment to build or modify models for several time-triggered protocols, which can be improved to satisfy the timing and reliability requirements of the hybrid networks. Chapter 4 presents a network model that contains associated node models of switches, bridges, routers and end nodes in wired and wireless TSN domains and Figures 4.4 and 4.5 illustrate the architecture graphs of our system model. TSN domains support the time-based TSN properties (i.e. IEEE 802.1Qbv) and fault-tolerance properties (i.e. IEEE 802.1CB) with an improved and precise IEEE 802.1AS synchronization protocol.

5.3.1 System Model Setup Definition

OPNET is use for the system model that contains the description of network components and physical links of an architecture model. The description of the tasks and messages that form an application model occurs in JavaScript Object Notation (JSON) formatted files. OPNET then converts JSON files into XML files. The reason for using XML files is that different hybrid network topologies are easily generated by importing XML files of the desired topologies. As shown in Figure 5.7, the JSON files contain JavaScript objects with attributes, where objects define the nodes and links of the architecture model, tasks and messages of the application model in a high-level definition, where the attributes are explained in Section 4.1. In other words, JSON files save time and avoid common pitfalls in the configuration settings by assigning several attributes using default values. Many developers prefer to use JSON because of its simplicity however it limits the flexibility and further details for the setup of the system configuration.



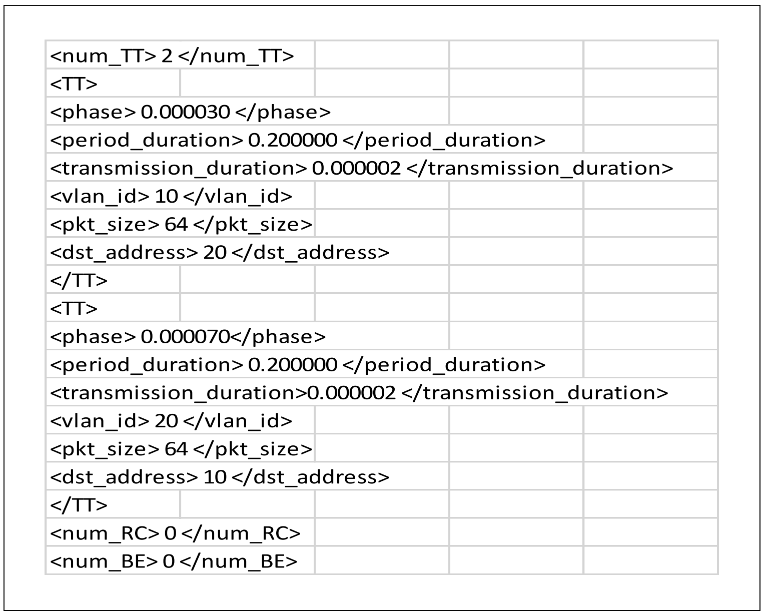
```
"Architecture": {
        nodes: [                                   links: [
          {                                          {
                  id: 0,                                     id: 0,
                  node-type: h,                              start: 0,
                  is_router: false,                          end: 1,
                  failure_r: 0.5,                            pos-start: (748, 838)
                  pos: (748, 838)                            pos-end: (764, 715)
                  mob  0                                     reliability: 0.995,
                  drift  250                                 Distance_link: 124.036
                  max_energy 100.000                 },
          },                                         {
          {                                                  id: 1,
                  id: 1,                                     start: 1,
                  node-type: ap,                             end: 0,
                  is_router: true,                           pos-start: (764, 715)
                  failure_r: 0.1,                            pos-end: (748, 838)
                  pos: (764, 715)                            reliability: 0.995,
                  mob 0                                      Distance_link: 124.036
                  drift  200                         },
                  max_energy 100.000
          },                                                 .
          {                                                  .
                  .                                          .
                  .
                  .
"Application": {
        tasks: [                                   messages: [
          {                                          {
                  id: 0,                                     id: 0,
                  et[H]                                      sender: 1,
                  en[H]                                      receiver: 0,
                  period: 200,                               et[R]
                  top-level: 660,                            en[R]
                  deadline: 1000,                            size: 12,
                  can_run_on: [2, 5, 6]                      comm:2ms
                  runs_on = 0,                               timetriggered: true,
          },                                                 conditional: essential
          {                                          },
                  id: 1,                             {
                  et[H]                                      id: 1,
                  en[H]                                      sender: 2,
                  period: 100,                               receiver: 1,
                  top-level: 440,                            et[R]
                  deadline: 1000,                            en[R]
                  can_run_on: [11, 6, 12]                    size: 6,
                  runs_on = 0,                               comm:1ms
          },                                                 timetriggered: true,
                                                             conditional: essential
                  .                                  },
                  .
                  .                                          .
                                                             .
                                                             .
```

FIGURE 5.7: Forming the system model as JSON files.

5.3.2 Configuring of TSN-enabled nodes

In the TSN hybrid framework, TSN traffic generating nodes (e.g. wireless hosts) are configured to send and receive their traffic depending on their traffic generation profiles. The TSN-enabled intermediate nodes (e.g. routers and bridges) use their queuing system to shape and schedule the ongoing traffic. Therefore, to establish the TSN properties of fault tolerance and traffic scheduling, every node obtains prior knowledge of the configuration including the network components, assigned tasks and their associated traffic profiles, besides, the message scheduling to schedule the traffic in temporal and spatial coordination. Ingress configuration files are used to identify the incoming traffic by its traffic identifier and define the egress port in the TSN-enabled intermediate nodes. If any message does not match TT or AV traffic, it is classified as BE traffic that does not have time constraints.

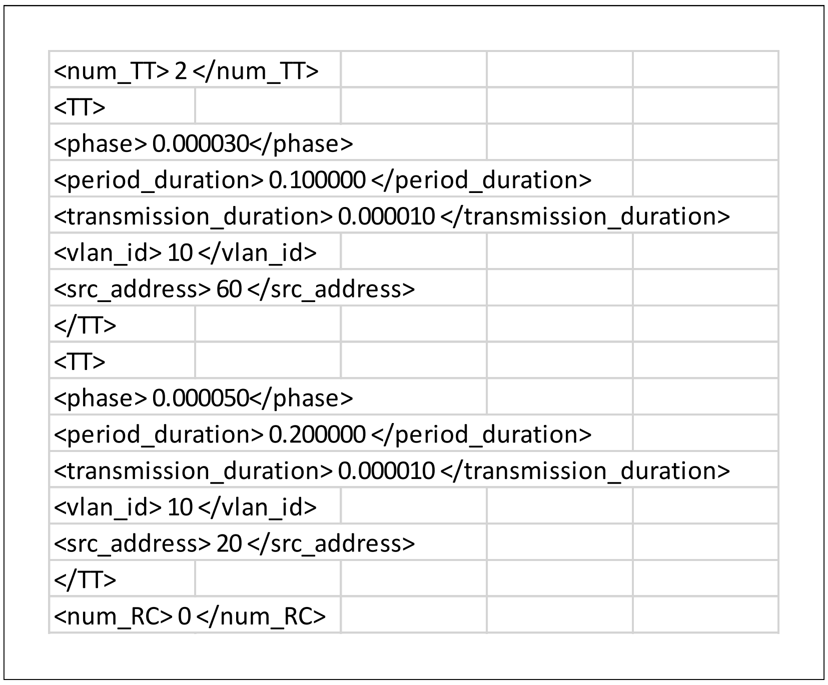
Figure 5.8 shows an example of a TT traffic generation profile at a sending node. Two TT messages are generated and the attributes of each message are defined. For example, the phase (e.g. 30 μs) defines the expected time at which a node expects to receive the message. The period (e.g. 200 ms) defines the recurrence time of the message. The transmission time (e.g. 2 μs) specifies how long the messages occupies the channel. VLAN_id (e.g. 10) is the VLAN identifier for the message, destination address (e.g. 20) is the final address for the generated message.



```
<num_TT> 2 </num_TT>
<TT>
<phase> 0.000030 </phase>
<period_duration> 0.200000 </period_duration>
<transmission_duration> 0.000002 </transmission_duration>
<vlan_id> 10 </vlan_id>
<pkt_size> 64 </pkt_size>
<dst_address> 20 </dst_address>
</TT>
<TT>
<phase> 0.000070</phase>
<period_duration> 0.200000 </period_duration>
<transmission_duration>0.000002 </transmission_duration>
<vlan_id> 20 </vlan_id>
<pkt_size> 64 </pkt_size>
<dst_address> 10 </dst_address>
</TT>
<num_RC> 0 </num_RC>
<num_BE> 0 </num_BE>
```

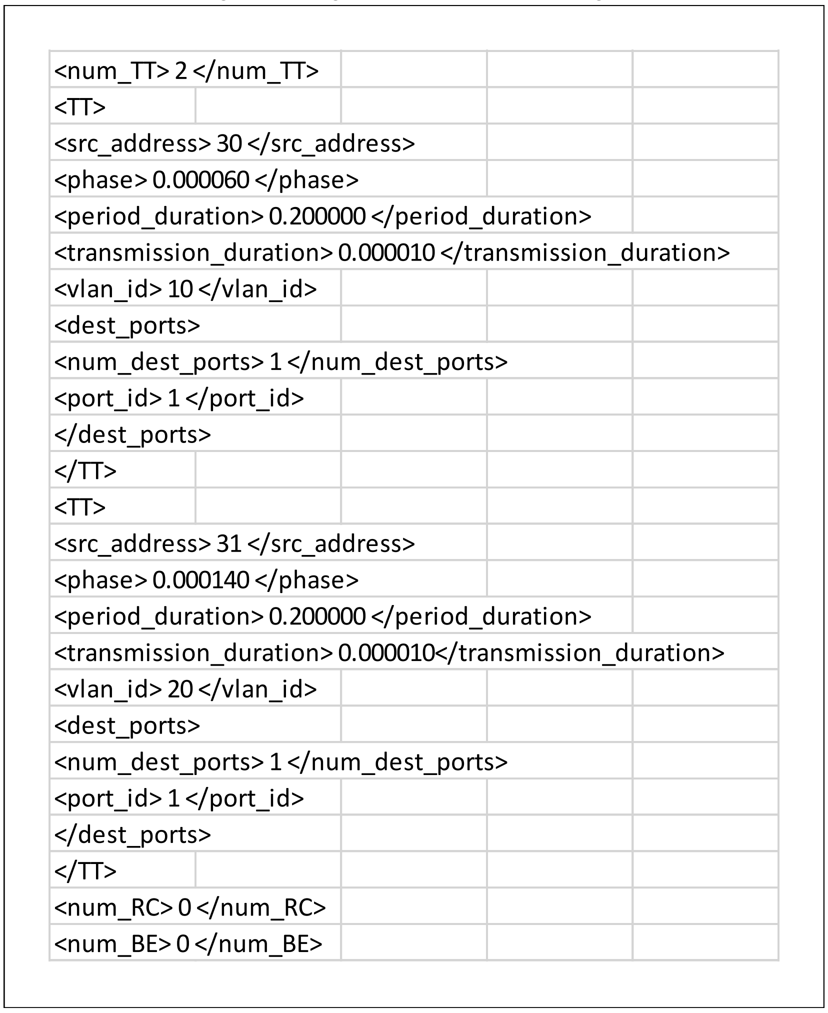
FIGURE 5.8: TT traffic generation profile at a sending node.

Figures 5.9a and 5.9b show the attributes of the ingress configuration files at receiving and intermediate nodes, respectively. The primary distinction between them is that the receiving end node specifies the prior node's address (src_address), whereas the ingress file of the intermediate node (such as a bridge) specifies additionally the egress port (dest_port) where the incoming message should be transmitted. The route of each copy of a TT message is specified off-line by using FRER.



```
<num_TT> 2 </num_TT>
<TT>
<phase> 0.000030</phase>
<period_duration> 0.100000 </period_duration>
<transmission_duration> 0.000010 </transmission_duration>
<vlan_id> 10 </vlan_id>
<src_address> 60 </src_address>
</TT>
<TT>
<phase> 0.000050</phase>
<period_duration> 0.200000 </period_duration>
<transmission_duration> 0.000010 </transmission_duration>
<vlan_id> 10 </vlan_id>
<src_address> 20 </src_address>
</TT>
<num_RC> 0 </num_RC>
```

(A) An ingress configuration file at a receiving end node.

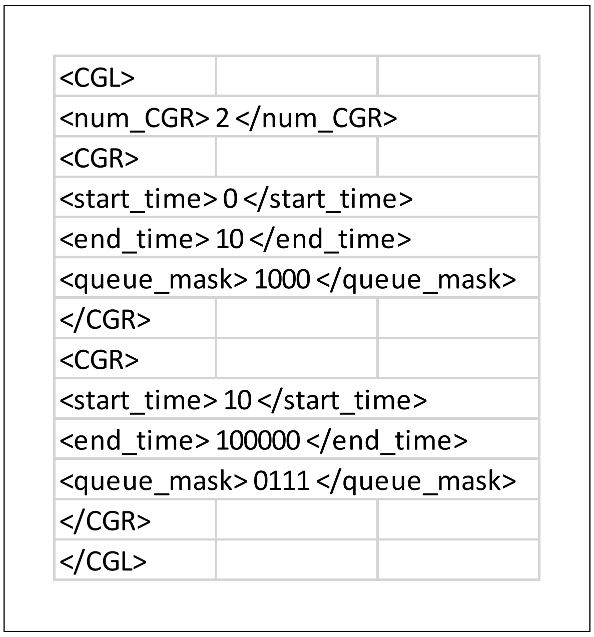


```
<num_TT> 2 </num_TT>
<TT>
<src_address> 30 </src_address>
<phase> 0.000060 </phase>
<period_duration> 0.200000 </period_duration>
<transmission_duration> 0.000010 </transmission_duration>
<vlan_id> 10 </vlan_id>
<dest_ports>
<num_dest_ports> 1 </num_dest_ports>
<port_id> 1 </port_id>
</dest_ports>
</TT>
<TT>
<src_address> 31 </src_address>
<phase> 0.000140 </phase>
<period_duration> 0.200000 </period_duration>
<transmission_duration> 0.000010</transmission_duration>
<vlan_id> 20 </vlan_id>
<dest_ports>
<num_dest_ports> 1 </num_dest_ports>
<port_id> 1 </port_id>
</dest_ports>
</TT>
<num_RC> 0 </num_RC>
<num_BE> 0 </num_BE>
```

(B) An ingress configuration file at an intermediate node.

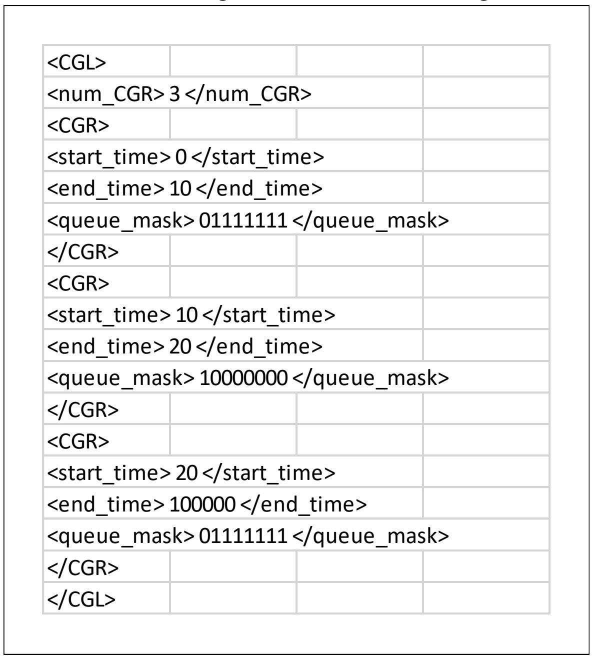
FIGURE 5.9: Ingress configuration files at receiving and intermediate nodes.

As mention in Subsection 5.2.2, the queue's gates of a port-specific GCL are mutually exclusive patterns and are continuously opened within a certain period of time which is assigned according to the LCM of periods of all periodic messages designated to that port. Figures 5.10a and 5.10b demonstrate the attributes of the GCL configuration files for sending and intermediate nodes, respectively. As an illustration, Figure 5.10a uses 1000 as the queue mask and 0 and 10 microseconds, respectively, for the start and end times. The gate of queue number 4 would be open and able to send traffic at any moment between 0 and 10 microseconds if these parameters were met. In the time frame specified, all other queues are closed. To offer a deterministic and synchronized transmission schedule, all TSN nodes run their GCLs simultaneously.



```
<CGL>
<num_CGR> 2 </num_CGR>
<CGR>
<start_time> 0 </start_time>
<end_time> 10 </end_time>
<queue_mask> 1000 </queue_mask>
</CGR>
<CGR>
<start_time> 10 </start_time>
<end_time> 100000 </end_time>
<queue_mask> 0111 </queue_mask>
</CGR>
</CGL>
```

(A) A GCL configuration file for a sending node.



```
<CGL>
<num_CGR> 3 </num_CGR>
<CGR>
<start_time> 0 </start_time>
<end_time> 10 </end_time>
<queue_mask> 01111111 </queue_mask>
</CGR>
<CGR>
<start_time> 10 </start_time>
<end_time> 20 </end_time>
<queue_mask> 10000000 </queue_mask>
</CGR>
<CGR>
<start_time> 20 </start_time>
<end_time> 100000 </end_time>
<queue_mask> 01111111 </queue_mask>
</CGR>
</CGL>
```

(B) A GCL configuration file for an intermediate node.

FIGURE 5.10: GCL configuration files for sending and intermediate nodes.

The transmission schedule at each output egress port (i.e. GCL) is generated using one of the task and message scheduling algorithms that are discussed in Chapters (7, 8). The main purpose of the algorithms is scheduling time-triggered tasks to the most suitable hosts in order to fulfil the real-time requirements and to consider the constraints of the wireless environment. Therefore, each algorithm determines the port-specific GCLs by taking the structure of the hybrid network and the specifications of the TT tasks and their associated periodic messages as inputs. In the output, the task and message scheduling solution specifies on which host a task can be run and which route an associated message can use to traverse. Thereby, the solution reflects the injection time and arrival time for each transmitted message routed via TSN-enabled nodes. Namely, every message is sent at a particular time slot and it is repeated with the period of that message.

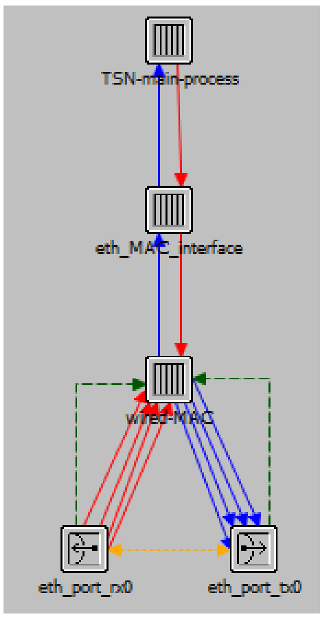
5.3.3 Modelling of a TSN-enabled Wirebound/Wireless End node

The models of standard end nodes that are situated in the wired and wireless TSN domains should incorporate additional modules in order to simulate a TSN hybrid environment. Thus, the modified modules can accommodate the TSN properties including accurate clock synchronization, traffic shaping and FRER. Figures 5.11a and 5.11b represent the node model of wired and wireless end nodes in OPNET, respectively.

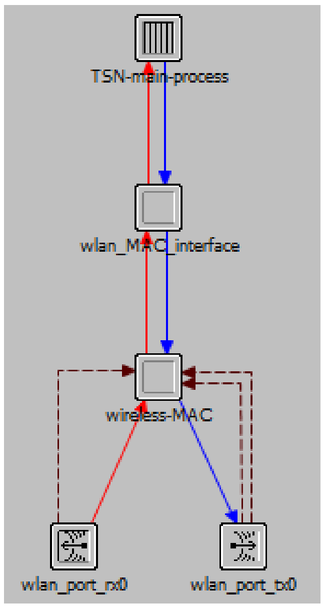
The end node model consists basically of three modules:

- *TSN-main-process* module.
- *MAC-interface* module.
- *MAC* module

In the TSN-enabled end node model, the *TSN-main-process* has the responsibility to generate messages. It is worth noting that our work focuses on generating only TT messages without other types of messages like AV or BE messages. The reason is that our thesis targets scheduling and transmission of periodic time-sensitive data used in real-time applications. Therefore, the rest of this thesis addresses scheduling TT traffic at a TSN hybrid environment. The generated TT messages are sent to the lower module (i.e. *MAC-interface*). The *MAC* module sends the received TT messages form the *TSN-main-process* to the directly connected nodes through the physical layer with the help of peer-to-peer transmitter and receiver units (i.e. *port-tx* and *port-rx*). At the instant of transmitting a message, the *MAC* module adds the local time in the message and encapsulates it at the *port-tx* unit. At the peer receiver node, the *port-rx* unit receives the message. Then the *MAC* module decapsulates the received TT message and forwards it through the *MAC-interface* to the upper layer to execute tasks that depend on the received TT messages.



(A) model of a wired TSN-enabled end node.



(B) Model of a wireless TSN-enabled end node.

FIGURE 5.11: models of TSN-enabled end nodes.

TSN-main-process

The *TSN-main-process* is the main module that has the responsibility to offer TSN services including TAS, FRER and time synchronization. The traffic flow in Figure 5.12 depicts how a TT message (coloured in purple) is generated, enqueued and dequeued into an egress port in a TSN-enabled end node. TAS performs time-based events that depend on a deterministic synchronization process. Figure 5.12 shows that four queues are dedicated to different traffic types. A TT message is enqueued in the highest priority queue. If non-TT messages (i.e. AV and BE messages) are generated, the lower priority queue is used to enqueue an AV message (coloured in cyan). The lowest two queues are used for different types of BE traffic (coloured in green). However, the services of the *TSN-main-process* are split into three child process modules as follows:

- *TSN-traffic-generator* module: it is responsible for the TSN FRER service.
- *TSN-enqueue-dequeue* module: it is responsible for the TSN time-aware shaping service.
- *TSN-sync-time* module: it is responsible for TSN timing services including the BMCA and synchronization.

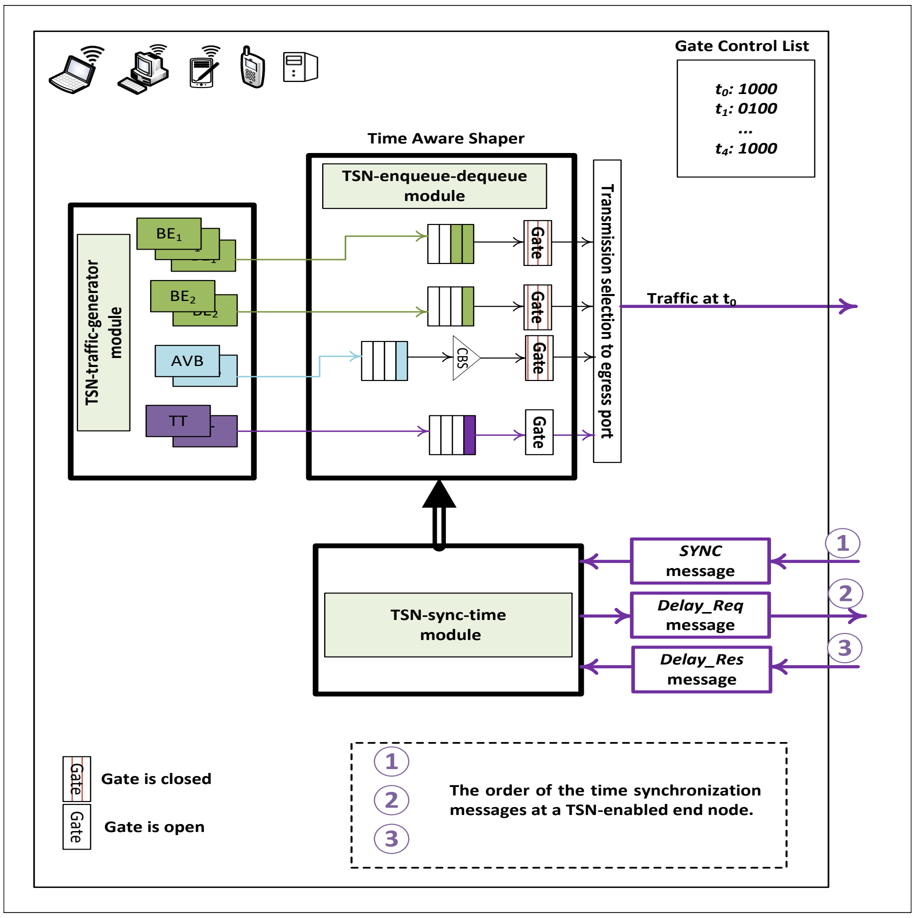


FIGURE 5.12: Diagram showing the services that are offered by the *TSN-main-process* module at a TSN-enabled end node.

Figure 5.13 shows the process model of the *TSN-main-process* that contains three forced and unforced states: *Init*, *Spawn* and *Idle* states. In the forced *Init* state, the main process allocates memory for flags used to indicate if the invoked child process module is in free or in busy status. The unforced *Spawn* state is used to initiate the child modules so that every child module is ready to operate its services. The unforced *Idle* state invokes a particular child module upon receiving a message from the lower layer (i.e. *MAC* layer). Thus, the *Idle* state coordinates and controls the operate of the child modules depending on the type of the incoming message. For instance, the *TSN-sync-time* module is invoked if the received message is an *Announce*, *Delay_Res* or *SYNC* message.

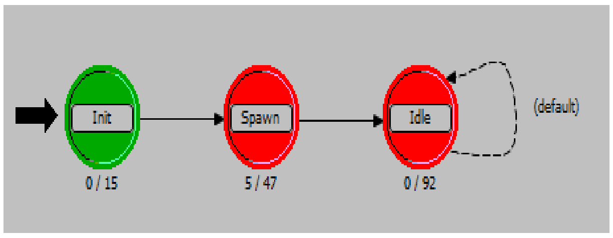


FIGURE 5.13: Process model of the *TSN-main-process* module at a TSN-enabled end node.

The *TSN-traffic-generator* module is triggered to generate copies of TT messages and send them through the disjoint redundant routes according to the TT traffic generation profile. Figure 5.8 illustrates the attributes of a TT generation traffic profile.

The *TSN-traffic-generator* module can also be used to generate AV and BE messages. The difference between TT and AV messages is that an AV message does not require a global clock to determine when an AV message should be sent. Instead, the sender determines the minimum time between successive AV messages belonging to the same AV stream. The determined time is called Bandwidth Allocation Gap (BAG) and depends on the network specifications. Therefore, two AV messages cannot be sent at a time less than the BAG of an AV stream. For generating BE messages, FRER can be skipped because the loss of a BE message cannot cause critical failures in the application. A *TSN-enqueue* process is then used to enqueue the generated message to its corresponding queue according to the traffic profile of the TSN-enabled node.

As illustrated in Figure 5.12, the egress TAS performs a *TSN-dequeue* process to check the gate status of every queue at an egress port. Therefore, this process is used to send out the messages from a queue to the *MAC* layer when the gate is opened. For instance, TAS lets a TT message be dequeued and sent out directly to the attached link if its queue is opened according to the generated GCL. Deterministic message transmission and QoS are guaranteed provided that the required time to deliver a message is equal or less than the time when the gate remains open. The gate has to be opened to transmit the entire message and to maintain its period pattern.

As mentioned earlier, TSN-enabled nodes depend on the precision of the global time for generating time-based services (i.e. IEEE 802.1Qbv functions). The wireless TSN nodes, which are integrated with the wired TSN domain, should also operate as a time-based homogeneous hybrid system. To achieve that, the *TSN-sync-time* module in Figure 5.12 considers on-line the factors that affect inversely the precision of the global time. These factors include clock drift, asynchronous packet transmission and non-deterministic processing. Therefore, the *TSN-sync-time* module extends the role of an 802.1AS clock, which does not consider these factors, to improve precision of the time synchronization. Further details about the extended 802.1AS protocol to consider the mentioned factors are shown in Chapter 6.

In the following, the *TSN-sync-time* module shows how to incorporate BMCA, peer-delay and synchronization mechanisms to synchronize the time of the TSN-enabled node with its grandmaster clock. Figure 5.14 depicts the state transition diagram of **modelling the *TSN-sync-time* module** to achieve the time synchronization with the grandmaster clock [208]. The states of the diagram are summarized as follows:

- *Init* state: it is a forced state, which is used to initialize all variables and forward the control to the unforced *Idle* state.
- *Idle* state: it generates an "*Announce-generate*" interrupt to dispatch an *Announce* message for the BMCA algorithm. The *TSN-sync-time* process model waits in the *Idle* state until the arrival of a gPTP message and then the process model moves the control to the *Busy* state by a *Service-start* interrupt.
- *Busy* state: it identifies the incoming gPTP message. For instance, if the message is an *Announce* message, the BMCA algorithm will be invoked. Similarly, if it is a *SYNC* message, the local time of the node will be synchronized with the grandmaster clock. The P2P path delay is measured when the received message is a *Delay_Res* message at the instant of processing the gPTP message in the *Busy* state. When the *Service-complete-Queue-empty* interrupt is triggered, the *TSN-sync-time* process moves the control to the *Idle* state if there are no gPTP messages in the queue. Otherwise, the *Service-complete-!Queue-empty* interrupt is triggered to let the process model wait in

the *Busy* state until processing all messages in the queue is finished, where !*Queue-empty* indicates that the queue is not empty.

- *Master-slave* state: it is a forced state, which is used to generate *Announce* and *SYNC* messages. Therefore, it serves the slave and the grandmaster events. Transition to the state is performed by a *Master-service-complete* interrupt. When finishing the service, the system model moves the control to the *Idle* or the *Busy* state based on the status of the queue. The *Queue-empty* interrupt is triggered to move the control to the *Idle* state or the !*Queue-Empty* interrupt is triggered to move the control to the *Busy* state.

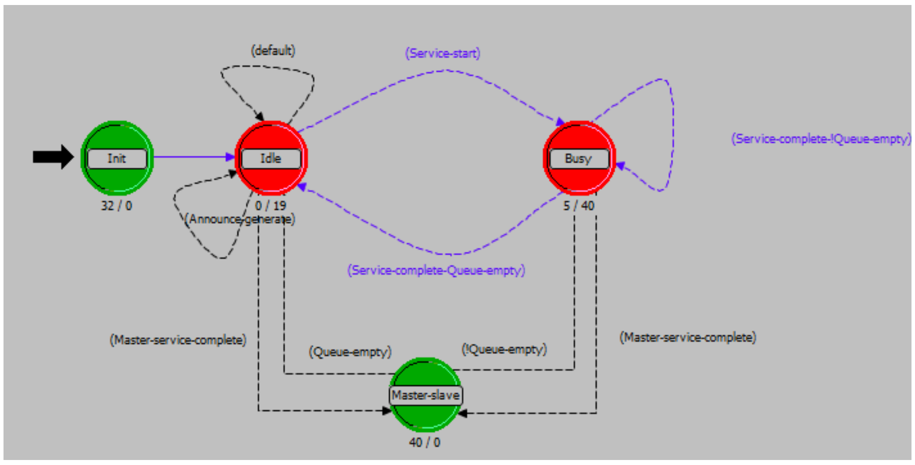


FIGURE 5.14: Process model of the *TSN-sync-time* module at a TSN-enabled end node [208].

The BMCA algorithm firstly initializes in every TSN-enabled node an *Announce* message to distribute its clock properties to all nodes in the TSN hybrid domain. The exchanged properties are used to build the master-slave hierarchy. Figure 5.15 depicts the format of the *Announce* message that is transmitted through the wired/wireless TSN domains. The fields, which are contained in the *Announce* message, are as follows:

- Message type (4 bit), Version PTP (4 bit) and Domain number (8 bit): they identify the type of the message (i.e. *Announce*, *SYNC*, *Delay_Req* or *Delay_Res*), the version of the PTP protocol (i.e. gPTP) and the TSN domain in which the TSN-enabled node is located (i.e. wired or wireless domain).

- Source-port Identity (80 bit): it identifies the port from which the message is transmitted.

- Priority 1 field (8 bit): the lowest value is the highest priority. Normally, the priority of a slave node is set to be 255 and it is 128 for the grandmaster node. It is manually configured depending on factors such as the location. For example, a node that is in a centralized position is typically preferred. Moreover, a wired node takes a higher priority than a wireless one due to the stability.

- Priority 2 field (8 bit): it is used to identify the last updated grandmaster and to specify the grandmaster clock and its redundant backup if they have the same properties.

- Clock class (8 bit): it is a list of class states. For instance, a node's clock that complies to Universal Coordinated Time (UTC) has a higher class than one that is free-running and manually set.

- Clock accuracy (8 bit): this is a list of a precision ranges for UTC (e.g., 25-100 ns).
- Clock variance (16 bit): this is a complex statistic for a modular log that represents the oscillator's jitter and its variation through the period of the *SYNC* messages.
- Time-source (8 bit): it specifies the type of time source that is used in the TSN-enabled node (e.g. GPS, NTP).
- Path trace Type Length Value (TLV) (96 bit): it determines the nodes that an *Announce* message passes through.
- StepsRemoved (16 bit): it is used at the receiver to verify the correctness of the received *Announce* message.

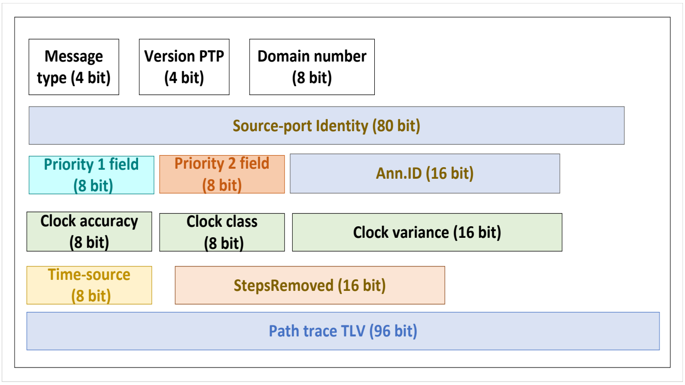


FIGURE 5.15: Format of an *Announce* message transmitted in TSN wired/wireless domains.

To simplify the procedure of selecting the grandmaster clock, the priority is used as a standard to differentiate between the clocks. Each TSN-enabled node in a TSN hybrid domain has two fields (called node's dataset) used during the selection of a grandmaster clock. The first field (i.e. clock priority) is used to represent the local dataset, whereas the second field (i.e. parent priority) is used to represent the priority of the best grandmaster clock found so far. The dataset of the node is continuously updated whenever an *Announce* message arrives.

In the initialise phase of the BMCA algorithm, the parent dataset of a TSN-enabled node is set to its local clock and the clock's status is in the master state. After that, each node sends out its *Announce* message to all neighbouring nodes in the TSN hybrid domain. The information obtained is utilized to build the hierarchy of slaves and masters. Whenever a node receives an *Announce* message, it compares its dataset with the dataset of the message sender. Thus, the state (i.e. master or slave) of the node is determined. Further, the state of the parent dataset is updated accordingly. For instance, the state of the receiver clock is identified to be a slave state and its parent priority is updated to the priority of the sender if the dataset of the sender is better than the local and parent datasets of the receiver. In contrast, if the receiver does not receive an *Announce* message with information better than the local dataset, the receiver remains as a master clock. It is considered as the grandmaster clock at the end of the *Announce* message transmission time.

To understand the procedure of the BMCA algorithm, Figure 5.16 depicts *Announce* messages that are dispatched from three TSN-enabled nodes in a TSN hybrid domain. The priorities (i.e. parent priority and clock priority) of the first clock *A* (coloured in brown) are

100 and 100, the priorities of the second clock *B* (coloured in blue) are 110 and 110 and the priorities of the third clock *C* (coloured in green) are 120 and 120. At the initialization, each clock is set to be in the master state. In the beginning, clocks *A*, *B* and *C* broadcast their *Announce* messages. Each message in Figure 5.16 is numbered by '1', '2' or '3' to indicate the arrival order of the messages. At the end of the broadcasting period, each node receives two *Announce* messages that contain the properties of the neighbour clocks. At the instant that clock *B* receives the first *Announce* message originated by *A*, it compares its dataset with the dataset of *A*. It then identifies itself as a slave clock and updates its parent priority to be 100 instead of 110 (i.e. the master clock of *B* is updated to be *A*). The reason is that *A* has better properties than *B*. On the other hand, upon arrival of the second *Announce* message from the clock *C*, *B* recognizes that its dataset is better than *C*. Therefore, no updating will occur on the dataset of *B* and it remains as a slave clock. Similarly, when the clock *C* receives the first *Announce* message from *B*, it updates its parent priority to be 110 instead of 120 and identifies itself as a slave clock. The second *Announce* message from *A* updates the dataset again to be 100 instead of 110. Clock *A* recognizes that it is the best available clock in the domain because its dataset is better than the dataset of the two *Announce* messages that are received from clocks *B* and *C*. Thereby, *A* does not change its master state and it is considered as a grandmaster clock for the TSN hybrid domain.

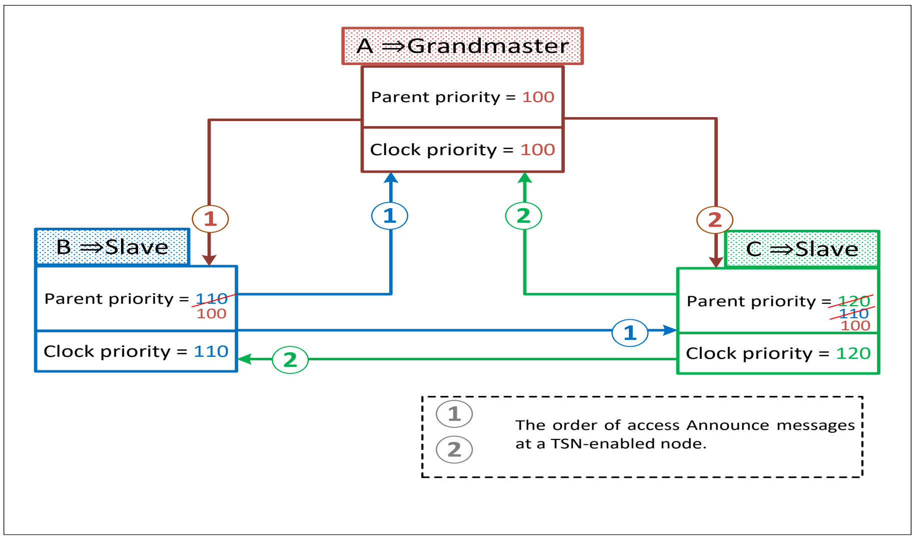

A ⇒Grandmaster
Parent priority = 100
Clock priority = 100
1
1
2
2
B ⇒Slave
Parent priority = ~~110~~ 100
Clock priority = 110
C ⇒Slave
Parent priority = ~~120~~ ~~110~~ 100
Clock priority = 120
1
2
1
2
The order of access Announce messages at a TSN-enabled node.

FIGURE 5.16: Procedure of selecting the grandmaster clock in a TSN hybrid domain using the BMCA algorithm.

When the BMCA is implemented and a grandmaster is detected, each slave clock initializes an interval named *Announce-receipt-timeout* which is normally 2-10 times the *Announce* interval. For instance, if the *Announce* interval is 2 seconds, the *Announce-receipt-timeout* interval is set to be 4 seconds at the minimum. The *Announce-receipt-timeout* interval is used to check the availability of the current grandmaster. If no message arrives from the grandmaster within this interval, the slave clock generates an *Announce* message and implements the BMCA in order to select a new grandmaster clock.

It is worth mentioning that the grandmaster in our system model will be mostly selected as a wired clock because the wired communication is more stable and does not suffer from interference and channel overlapping as in wireless communication. Therefore, the priority attributes of a node clock are user-defined to have a higher priority in the wired nodes than in the wireless nodes.

After establishment of the master-slave hierarchy, the grandmaster sends periodic $SYNC$ messages containing its own local time in the *Original-timestamp* field. Figure 5.18 depicts the format of the $SYNC$ message that is transmitted through the wired/wireless TSN domains. The $SYNC$ message is transmitted through TSN-enabled bridge nodes until arriving at the slave clocks. The target of the $SYNC$ message is to synchronize the slave clocks with their grandmaster. *Follow-Up-SYNC* messages are sent if a two-step process is used. Every TSN-enabled node receives a $SYNC$ message and the *TSN-sync-time* module executes the P2P mechanism that consists of request/response message transmission between the recipient node and its TSN neighbour node that forwards that $SYNC$ message, as shown in Figure 5.17.

When a TSN-enabled node transmits a *Delay_Req* message, the P2P mechanism begins and records the time the message was transmitted (i.e. t_1). The neighbour node receives that *Delay_Req* message at t_2 and immediately generates a *Delay_Res* message to store t_2 and the time at which the *Delay_Res* message is sent (i.e. t_3) in *Request-receipt-timestamp* and *Response-timestamp* fields, respectively. Figures 5.19 and 5.20 depict the format of the *Delay_Req* and *Delay_Res* messages that are transmitted through the wired/wireless TSN domains, respectively. The *Follow-Up-Delay-Res* message is sent if a two-step process is used.

When a TSN-enabled node receives the *Delay_Res* message and stores the time at which the *Delay_Res* message arrives (i.e. t_4), then the recipient node has all required four timestamps used for the P2P mechanism. Thereby, the path delay that is computed by the P2P mechanism is added to an aggregated path delay, which represents the delay along the whole route from the grandmaster until a slave clock. The aggregated value is continuously updated in the *Correction-field* of the $SYNC$ message that passes through every TSN-enabled node until the slave node.

At the instant of receiving the $SYNC$ message, the slave clock is synchronized with the grandmaster clock. To achieve that, the grandmaster time included in the *'Original-timestamp'* is compensated with the aggregated path delay in the *Correction-field*. Chapter 6 contains further details about how the *Correction-field* is considered in the deterministic processing time (i.e. residence time) in the intermediate nodes. Moreover, the *CSRO-field* is also addressed in Chapter 6 to consider the problem of clock drift.

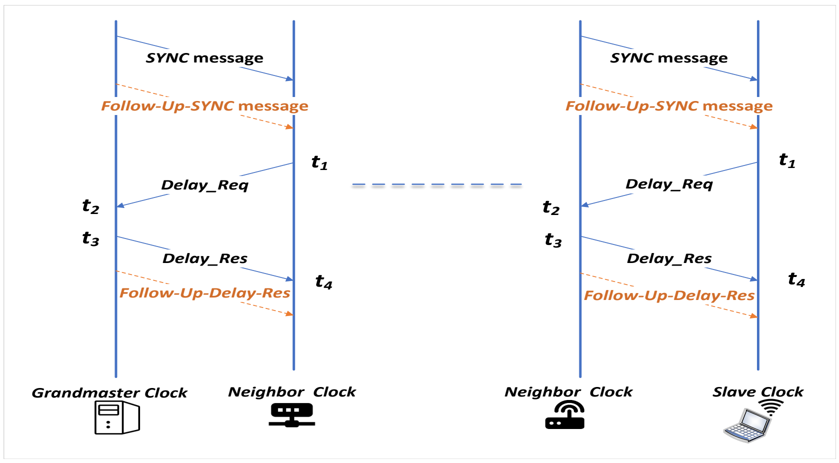


FIGURE 5.17: Message passing gPTP scheme in the TSN hybrid domain.

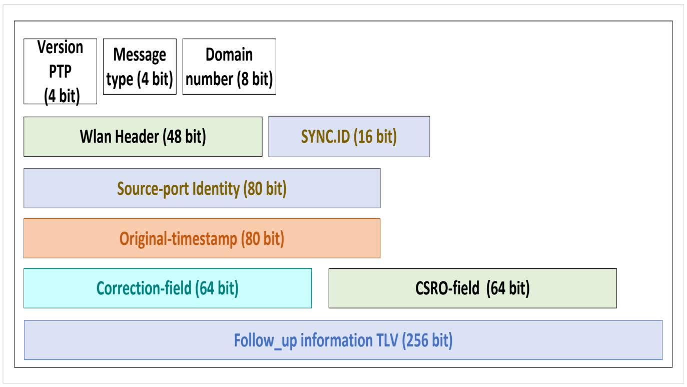

Version PTP (4 bit)
Message type (4 bit)
Domain number (8 bit)
Wlan Header (48 bit)
SYNC.ID (16 bit)
Source-port Identity (80 bit)
Original-timestamp (80 bit)
Correction-field (64 bit)
CSRO-field (64 bit)
Follow_up information TLV (256 bit)

FIGURE 5.18: Format of a *SYNC* message transmitted in the TSN wired/wireless domains.

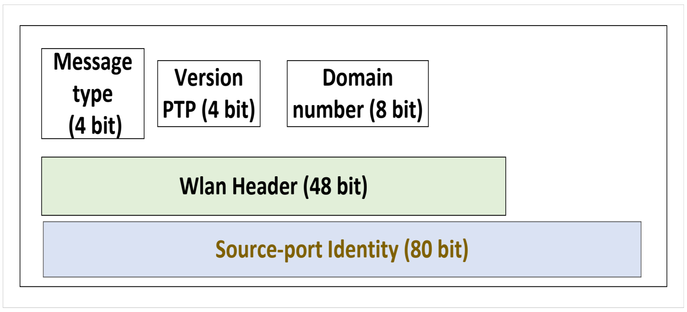


FIGURE 5.19: Format of a *Delay_Req* message transmitted in the TSN wired/wireless domains.

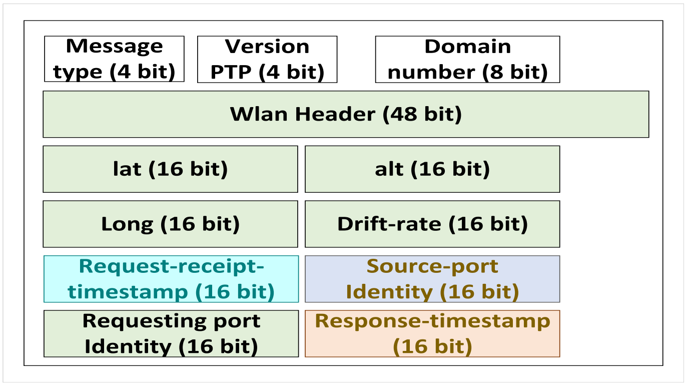


FIGURE 5.20: Format of a *Delay_Res* message transmitted in the TSN wired/wireless domains.

MAC-interface Module

The *MAC* module and higher modules use it to communicate and share data. It is the same as in the standard interface for both wired and wireless nodes.

MAC Module

The main target of the *MAC* module is to record all timestamps including the sending time of the outgoing messages and receiving time of the incoming messages.

At the grandmaster when a *SYNC* message is initiated, the *MAC* module inserts the local time in the *Original-timestamp* at the time of sending the message through the *port-tx*. To generate a *Delay_Res* message, the *MAC* module encapsulates the time-stamp of receiving the *Delay_Req* message at the *port-rx* with the time-stamp of sending the *Delay_Res* message at the *port-tx* and inserts them in the transmitted *Delay_Res* message. It is worth noting that the grandmaster does not generate *Delay_Req* messages because it does not receive *SYNC* messages in its gPTP domain.

On the other hand at the slave node, the *MAC* module synchronizes the node's clock to the time of the grandmaster clock, which is included in the received *SYNC* message. Moreover, the slave node periodically sends *Delay_Req* messages to its adjacent node (i.e. the nodes that forwarded the *SYNC* message) in order to implement the P2P mechanism. To generate a *Delay_Req* message, the *MAC* module encapsulates the time-stamp of sending the *Delay_Req* message at the *port-tx* and inserts the associated time-stamp in the transmitted *Delay_Req* message. On receiving a *Delay_Res* message, the *MAC* module uses the stored and the included timestamps in the received message to compute the P2P path delay.

To prevent the latency margin caused by creating time stamps at upper layers (such as the layer between the MAC and the Logical Link Control (LLC) sub-layer), the time-stamping takes place close to the physical layer of a TSN-enabled node. Consequently, the physical layer time-stamping occurs either when a message is sent from the *MAC* module towards the *port-tx* transmitter unit or when a message is sent from the *port-rx* receiver unit towards the *MAC* module. Since the *MAC* module is directly connected with the transmitter and receiver units, the latencies of ingoing and outgoing message are neglected.

5.3.4 Modelling of a TSN-enabled Ethernet/wireless intermediate node

For modelling a TSN-enabled Ethernet/wireless intermediate node, the OPNET Ethernet standard switch model and OPNET wireless standard model (i.e. access points, bridges and wireless routers) are modified in a way that they support subsets of non-time-based and time-based properties of the hybrid TSN domain. The regular intermediate node forwards the incoming message either as unicast traffic by specifying the MAC destination address or as broadcast traffic by sending copies of the message from all egress ports. However, the TSN-enabled bridge node comprises two different process modules, namely the *TSN-main-bridge* module and the *MAC* module.

TSN-main-bridge Module

The TSN-enabled intermediate node operates through a *TSN-main-bridge* module and six child modules. The state transition diagram of the main module consists of an *Init* state to initialize all ports and build a port mapping table to map each connected port to a MAC address. A *Spawn* state is used to initialize all six child modules, and an *Idle* state to invoke a child module according to the type of the incoming message. Therefore, each child module can be described as follows:

- *Bridge-protocol-entity* module: it is used in the TSN-enabled intermediate nodes to run the Spanning Tree Protocol (STP) upon receiving a Bridge Protocol Data Unit (BPDU) [209].
- *LACP* module: this process uses the Link Aggregation Control Protocol (LACP) [210] to identify and store the operational and state modes for each enabled port in the intermediate node.

- *MSTP* module: it executes the Multiple Spanning Tree Protocol (MSTP) to create multiple spanning instances for each Virtual LAN throughout a local area network.

- *PVST* module: it runs Per VLAN Spanning Tree (PVST) to allow a TSN-enabled intermediate node to have multiple spanning trees per Virtual LAN.

- *Bridge-MAC-relay* module: as indicated in the relevant traffic generation profile, it first receives the incoming message and then transmits it to the destination port. According to Subsection 5.2.3, FRER introduces different functions. Therefore, this module calls firstly the **message identification function** to identify the incoming message. As mentioned earlier, only TT traffic is supported, therefore this function needs to find out the incoming TT message to which the stream belongs to. The TT message *seq-num* is then retrieved by executing the **decoding sequencing function**, and **the sequence recovery function** to examine the *seq-num* of the received TT message in relation to the *seq-num* of the previous TT message of the stream to which the received message belongs. If the received message's *seq-num* is higher by one than the previous message's *seq-num*, the **encoding sequencing function** is called and the *TSN-enqueue* module is invoked. Otherwise, the message with the wrong *seq-num* will be skipped. Figure 5.21 depicts the flow of the functions that are offered by the *Bridge-MAC-relay* module at a TSN-enabled intermediate node.

- *TSN-bridge-sync-time* module: the TSN-enabled intermediate node plays the role of an 802.1AS boundary clock. This means that the *TSN-bridge-sync-time* module in Figure 5.21 executes all clock synchronization methods including BCMA, P2P measurement and synchronization mechanism once receiving *Announce*, *Delay_Req*/*Delay_Res* and *SYNC* messages, respectively. The logic of the synchronization methods in a TSN-enabled bridge node is the same as in a TSN-enabled end node. To be more specific, if a TSN-enabled bridge node is identified as a grandmaster clock, it sends periodically *Announce* and *SYNC* messages. In contrast, if a TSN-enabled bridge node is identified as a slave clock, the P2P and the synchronization mechanisms are executed at the instant of receiving *Delay_Req*/*Delay_Res* and *SYNC* messages, respectively.

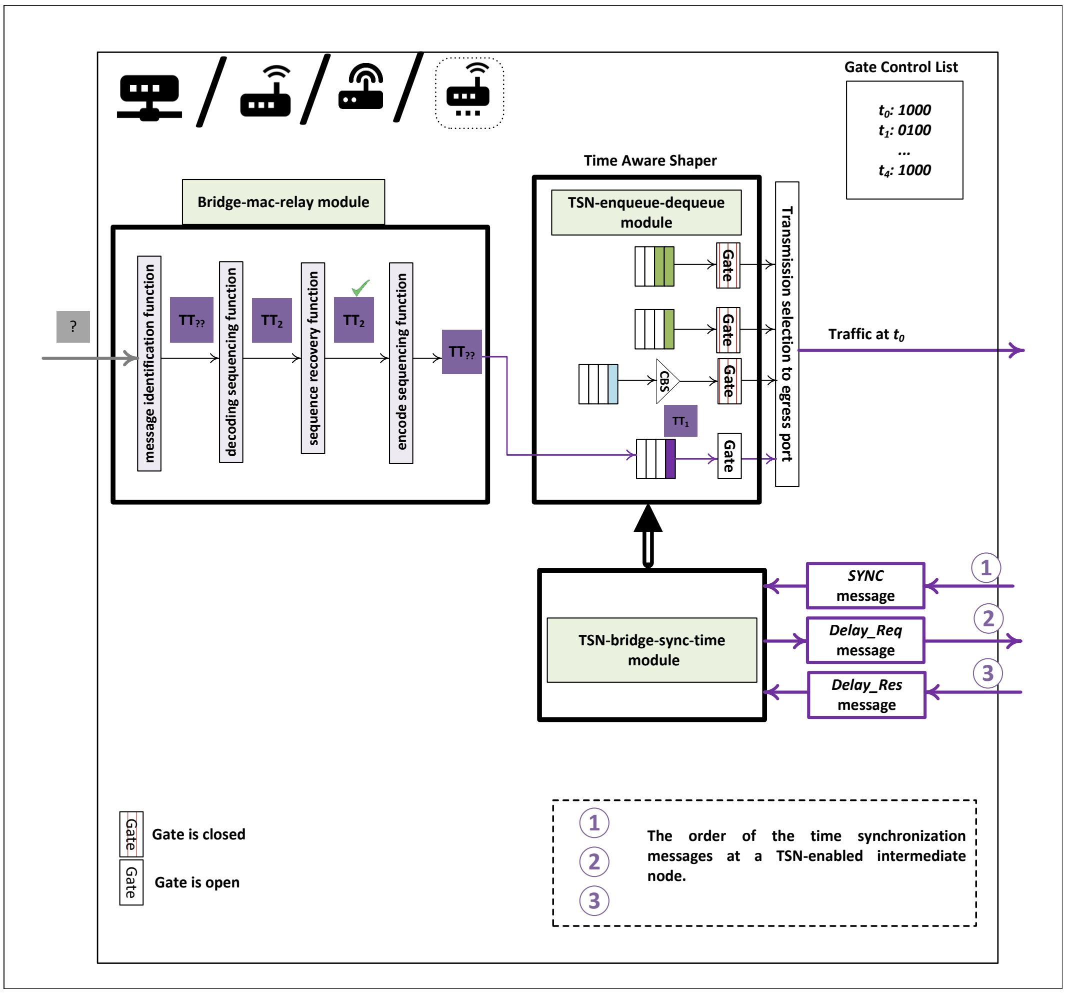

Gate Control List
t_0: 1000
t_1: 0100
...
t_4: 1000
Time Aware Shaper
Bridge-mac-relay module
TSN-enqueue-dequeue module
message identification function
decoding sequencing function
sequence recovery function
encode sequencing function
Transmission selection to egress port
Traffic at t_0
?
$TT_{??}$
TT_2
TT_1
CBS
Gate
TSN-bridge-sync-time module
SYNC message
Delay_Req message
Delay_Res message
1
2
3
Gate is closed
Gate is open
The order of the time synchronization messages at a TSN-enabled intermediate node.

FIGURE 5.21: Flow of the services that are offered by the *TSN-main-bridge* module at a TSN-enabled intermediate node.

As shown in Figure 5.22, **modelling the** *TSN-bridge-sync-time* **module** of a TSN-enabled bridge node has the same states as in the TSN-enabled end node with an additional *Port-QoS-deq* state. The *Port-QoS-deq* state is invoked if the neighbouring nodes implement quality of service functions via output port dequeuing.

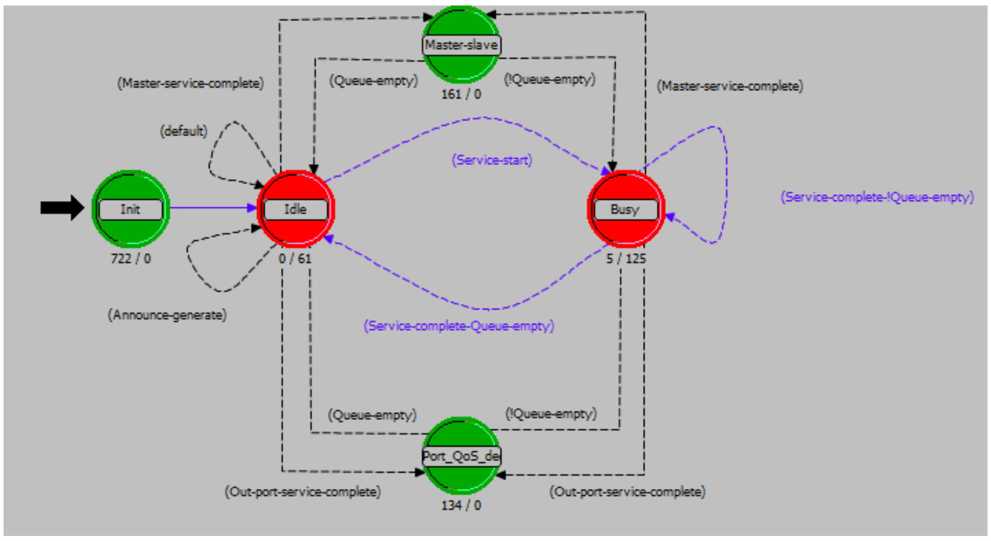


FIGURE 5.22: Process model of the *TSN-bridge-sync-time* module at a TSN-enabled intermediate node [208].

MAC Module

The *MAC* module of a TSN-enabled intermediate node operates in the same way as the *MAC* module of a TSN-enabled end node. To be more specific, if a TSN-enabled intermediate node is identified as a grandmaster, the TSN-enabled intermediate node transits all its output ports to the master state, and then it starts sending periodically *Announce* and *SYNC* messages within its hybrid TSN domain. A TSN-enabled bridge node that is identified as a grandmaster does not perform the P2P mechanism. Therefore, it does not receive *SYNC* or transmit *Delay_Req* messages. In contrast, if a TSN-enabled intermediate node is identified as a slave clock, at most one of the ports transits to the slave state to receive the *SYNC* messages from the grandmaster, and the rest of the ports are transited to the master state to forward the received *SYNC* messages. A *MAC* module, which is associated with a port in the slave state, records and performs all time-stamps related to the P2P mechanism. These time-stamps are then used to measure the P2P path delay.

Chapter 6

Time Synchronization for Improved Precision in an Asymmetric TSN Hybrid Network Using Extended IEEE802.1AS

In safety-critical distributed systems, hybrid networks are gaining importance, and clock synchronization is a crucial service to provide safety-relevant services that span both wired and wireless subsystems. The IEEE 802.1AS protocol does not account for some delays that are present in hybrid wireless networks, including random delays (channel access and transmission jitter) and deterministic delays (residence time, propagation, transmission, and receiving). In time-aware hybrid networks, asymmetric delays are one of the key causes of the synchronization process' inaccuracy. The synchronization precision is affected by the asymmetric delays in such a way that the degree of precision depends on the accuracy of the calculated path delay. In the TSN-enabled hybrid system based on the 802.1AS synchronization protocol, all slave nodes in a hybrid domain share a global time base that is realized using the robust synchronization mechanism. The standard 802.1AS synchronization protocol is designed mainly for Ethernet networks and assumes that the communication links between nodes are symmetric. Therefore, determining the path delay is simple by calculating the mean delay value for *Delay_Req* and *Delay_Res* messages based on Eq. 3.1. On the contrary, the path delay obtained in Eq. 3.1 is not an optimal solution in asymmetric TSN hybrid networks.

Several variations like variable data rate, time-stamping errors, and the mobility of the nodes are frequently changing for the same node and thus affect significantly the path delays and synchronization precision. To make it worse, the synchronization process with a reference clock can be affected by the continuously changing drifts of the clocks that depend on several factors like the clock class and clock accuracy.

Based on the above, this chapter proposes an extension of the standard 802.1AS protocol to improve the precision of the time synchronization in a TSN hybrid network. The correctness and applicability of the extended IEEE802.1AS clock synchronization protocol are studied through several test scenarios. Based on an example hybrid network layout, simulation scenarios are run in order to generate experimental results that show the validity and viability of implementing TSN services over wired and wireless networks.

The rest of the chapter is structured as follows: Section 6.1 discusses the clock drift in a TSN-enabled node. Section 6.2 presents modelling the delay of the *Delay_Res* message. A path deviation delay filter test to skip outlier path delays is discussed in Section 6.3. Section 6.4 presents the hybrid simulation models for the clock synchronization process. The optimization of 802.1AS in a TSN hybrid domain is shown in Section 6.5. Finally, Section 6.6 illustrates the simulation setup and evaluation of the improved 802.1AS protocol.

6.1 The Frequency Drift in a TSN-enabled Node

In electronic nodes especially in telecommunications, the frequency drift, which is traditionally measured in Hz/s, is normally unintended and means that the oscillator deviates from its specified frequency. This relates to different reasons including the node ageing [212], temperature variation [213] that vary the piezoelectric effect in a crystal oscillator and voltage regulation problems on the oscillator. The mentioned reasons are variable and one or more of them could cause changeable effects on a node and more or less on another one. For example, a slave clock may suffer from an increase in the temperature while the grandmaster clock has a problem with its voltage regulator. This leads to frequency drift in the slave clock ,which differs from its reference clock. Frequency stability can be measured as the absence or level of the frequency drift. Besides, the frequency drift may cause the drift of a radio transmitter to an adjacent channel if multiple channels are used to schedule the wireless data transmission, which in turn causes unintended signal interference. For this reason, frequency allocation regulations are used to specify the allowed tolerance for such oscillators in TSN-enabled wireless nodes. For example, a Temperature-Compensated Voltage-Controlled Crystal Oscillator (TCVCXO) is normally used for frequency modulation.

The frequency drift of a clock can occur either in a linear or non-linear way. In the linear drift, the drift rate is constant while the drift rate varies in the non-linear drift. In the real world, the non-linear drift is more realistic according to the changeable environmental and technical factors such as temperature and voltage fluctuations. The extension of the standard 802.1AS protocol solves this problem by computing a frequency ratio of the neighbour TSN nodes, known as Neighbour Rate Ratio (NRR) [208]. The NRR value is computed in each TSN-enabled intermediate node, which is equal to the frequency ratio with the neighbour clock. On the other side, the Cumulative Scaled Rate Offset (CSRO) [208], which is carried by the *SYNC* message, is an aggregated value by adding the system's NRR values. As a result, the *SYNC* messages are utilized to employ CSRO to sync an oscillator's frequency (syntonization) with the grandmaster clock in a TSN-enabled node. Due to the periodic sending of the *SYNC* message, the deviation of the non-linear drift is computed regularly every time the *SYNC* message is sent. Therefore, selecting the appropriate *SYNC* interval affects the accuracy of the synchronization process significantly.

6.2 Modelling the Timing of Delay Response Messages

This section describes techniques for accurately measuring the path delay value for the P2P mechanism. The delay of the *Delay_Res* message (i.e. *Response_delay*) represents the actual path delay, because it represents the transmission delay of the *SYNC* message. Thus, it must be accurately measured.

The slave node adjusts its local time with the grandmaster and the offset value to the grandmaster should be minimized. Therefore, if the P2P mechanism is incorrect, it leads to a bias or deviation between the local clock and the grandmaster clock. When communication delays are asymmetric, the difference of the propagation and transmission delays has to be determined for the different directions but identifying this difference in the P2P mechanism (upload and download) is not mentioned in the standard protocol. This degrades the synchronization precision for dynamic wireless environments, where the data rate is frequently changed as a result of channel conditions, mobility or the characteristics of the service. If we assume that the grandmaster is located in the wired TSN domain, the synchronization messages will be transmitted through the wired and wireless TSN bridges until arriving at the TSN-enabled wireless end nodes. Several broadband technologies such as mobile WiMAX and LTE would change in the data rate. Changes can be caused by the mobility of the TSN end nodes (e.g. sensors). Slow moving does not affect the synchronization process, but if a mobile node increases its speed, this leads to fluctuation in its data rate and makes it unstable due to changes in the quality of the used wireless links. The wireless link is a shared resource for multiple users who want to send data at that link. Thus, the allocated time varies for each scheduled interval. Conditions in the link such as channel fading and errors affect also the date rate. Moreover, wireless technologies use different asymmetric duplexing mechanisms. For example, asymmetric traffic can occur for specific services (e.g., web server) where higher bandwidth is required for downloading than uploading. These asymmetric characteristics cause varying time offsets between the grandmaster and its slave clocks.

When integrating the synchronization process in TSN hybrid networks, which contain different node models, random mobility, dynamic data rates and asymmetric characteristics, it becomes necessary to propose a model to deal with the mentioned boundary conditions.

6.2.1 Symmetric Degree Ratio (SDR)

In order to measure the ratio of delays for *Delay_Req* and *Delay_Res* messages during the P2P mechanism, the Symmetric Degree Ratio (SDR) is introduced as shown in Eq. 6.1. If its value is unity or very close to unity, this means symmetric delays for *Delay_Req* and *Delay_Res* messages. Thereby, considering and identifying all the differences in the P2P traffic, the SDR value has to be approximately one. Therefore, the SDR ratio gives an indication of the symmetric P2P delay mechanism. If the SDR value deviates from unity, it means we have to consider asymmetric environmental characteristics in the wireless TSN network. The SDR ratio of a TSN node can be calculated as:

$$SDR = \frac{Request\_delay}{Response\_delay} \tag{6.1}$$

Where, *Request_delay* and *Response_delay* are the times needed to transmit *Delay_Req* and *Delay_Res* messages, respectively. According to Figure 5.17, these delays are defined as follows.

$$Response\_delay = t_4 - t_3 \tag{6.2}$$

$$Request\_delay = t_2 - t_1 \tag{6.3}$$

Where, t_1 is the egress time-stamp and t_2 is the ingress time-stamp of the sender and receiver clocks when sending and receiving the *Delay_Req* message, respectively. Similarly, t_3 is the egress time-stamp and t_4 is the ingress time-stamp of the sender and receiver clocks when sending and receiving the *Delay_Res* message, respectively. To avoid unpredicted delays and time-stamping errors as mentioned in Section 3.6.6, the time-stamping process is operated at the MAC layer, which eliminates the need to know the delay from higher layers and gives a more accurate reading when sending messages. The difference in the transmission time equals the absolute value of subtracting Equations 6.4 and 6.5.

$$Request\_trans\_time = \frac{Delay\_Req\_size}{Data\_rate} \tag{6.4}$$

$$Reply\_trans\_time = \frac{Delay\_Rep\_size}{Data\_rate} \tag{6.5}$$

$Data\_rate$ in Equations 6.4 and 6.5 refers to the date-rate used to send the $Delay\_Req$ and $Delay\_Res$ messages, respectively.

According to the potential movement of some TSN wireless end nodes, the distance between them is subject to be changed. Therefore, the propagation delay for $Delay\_Req$ and $Delay\_Res$ messages is calculated using Equations 6.6 and 6.7, respectively.

$$Request\_propagation\_time = \frac{Dist\_req}{speed\_of\_light} \tag{6.6}$$

$$Reply\_propagation\_time = \frac{Dist\_rep}{speed\_of\_light} \tag{6.7}$$

Where, $Dist\_req$ and $Dist\_rep$ are computed based on the distances between the message sender and the message receiver for $Delay\_Req$ and $Delay\_Res$ messages, respectively. The difference in the propagation time equals the absolute value of subtracting Equations 6.6 and 6.7.

The time difference equals the sum of the difference in the transmission time and the difference in the propagation time. Eq. 6.8 shows the consideration of the time difference ($time\_dif$) in the SDR ratio.

$$SDR = \frac{Response\_delay}{Request\_delay + time\_dif} \tag{6.8}$$

$time\_dif$ deals with delays required for the transmission on the physical layer and the traffic propagation in the medium. Even if the time difference is computed and considered in the SDR ratio, in some scenarios, the SDR ratio deviates from unity as a result of highly dynamic changes in the size and the topology of the network, besides the random access delays. Eq. 6.9 shows how to compensate the time difference for $Delay\_Req$ and $Delay\_Res$ messages.

$$t_4 - t_3 = (t_2 - t_1) + time\_dif \tag{6.9}$$

Dividing Eq. 6.9 by $(t4 - t3)$, and by using Eq. 6.1, the SDR ratio can be expressed as in Eq. 6.10.

$$SDR = 1 - \frac{time\_dif}{t_4 - t_3} \tag{6.10}$$

6.2.2 Neighbour Rate Ratio (NRR)

The NRR value is used to compensate for the drift time of a local clock and support TSN-enabled nodes with different drift rates. Therefore, Equations 6.11 and 6.12 show the expressions of the timing offset and the normalized frequency offset (the clock drift factor) between a node clock and its neighbour clock.

$$t_2 - t_1 = Request\_delay - Offset - Drift \tag{6.11}$$

$$t_4 - t_3 = Response\_delay + Offset + Drift \tag{6.12}$$

$$Where, Drift = NRR * (current\_time - last\_sync\_time) \tag{6.13}$$

The $current\_time$ represents the local time of a node once it receives the $SYNC$ message. In contrast, the $last\_sync\_time$ is the local time of the node since the last time it received the *SYNC* message. NRR represents the clock drift factor between the current node, which receives the *SYNC* message and its neighbour that has sent the *SYNC* message.

Equations 6.11 and 6.12 can be reformulated as shown in the following equations.

$$t_2 = t_1 + Request\_delay - Offset - NRR * (t_1 - last\_sync\_time) \tag{6.14}$$

$$t_4 = t_3 + Response\_delay + Offset + NRR * (t_4 - last\_sync\_time) \tag{6.15}$$

The value of NRR can be computed via two subsequent *Delay_Res* messages that have been received at a TSN-enabled node, as shown in Figure 6.1. Since NRRs are continuously measured through the P2P mechanism, if a network encounters some changes (e.g. reconfiguration of nodes or switching to another grandmaster), there is no need to explicitly request a measurement of NRRs.

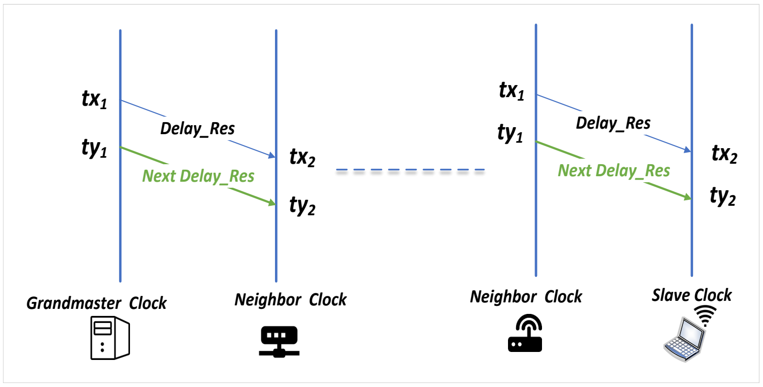


FIGURE 6.1: Two consequent *Delay_Res* messages.

Similarly to Eq. 6.15, the expressions of tx_2 and ty_2 can be shown as follows:

$$tx_2 = tx_1 + Response\_delay + Offset + NRR * (tx_2 - last\_sync\_time) \tag{6.16}$$

$$ty_2 = ty_1 + Response\_delay + Offset + NRR * (ty_2 - last\_sync\_time) \tag{6.17}$$

Where, tx_2 and tx_1 are time-stamps of a *Delay_Res* message at the receiving node and the sending neighbouring node, respectively. Similarly, ty_2 and ty_1 are the time-stamps of the same *Delay_Res* message. The neighbouring node is where the *SYNC* message is sent or forwarded.

By using Equations 6.16 and 6.17, the expression of NRR can be expressed using Eq. 6.18:

$$NRR = 1 - \frac{ty_1 - tx_1}{ty_2 - tx_2} \tag{6.18}$$

It is worth noting that if the neighbouring TSN nodes have the same drift rate, then the measured NRR value is unity, which may lead to an inaccurate computation of the synchronized time due to the actual frequency offset between the slave and the grandmaster clocks. Therefore, the CSRO is used to accumulate the values of NRR during the transmission of the *SYNC* message between the grandmaster and the slave clocks. If there are multiple TSN intermediate nodes between the grandmaster and a slave node, each intermediate node computes its NRR and adds it to the accumulative CSRO value. The CSRO value is then loaded into the *CSRO-field* of the *SYNC* message as shown in Figure 5.18 that shows the *CSRO-field* in the *SYNC* message format. Even with NRR unity values between neighbouring nodes, the respective frequency ratio will be obtained by the aggregated CSRO values.

6.2.3 Response Delay Value

After formulating the SDR and NRR equations from the previous sections, it is time to express the *Response_delay* formula as shown in this section.

Equations 6.14 and 6.15 are used to formulate Eq. 6.19 as follows:

$$(t_2 + t_4) - (t_1 + t_3) = Request\_delay + Response\_delay + NRR * (t_4 - t_1) \tag{6.19}$$

Dividing Eq. 6.19 by the *Response_delay* value, the *Response_delay* value is finally obtained using Eq. 6.20:

$$Response\_delay = \frac{(t_2 - t_1) + (t_4 - t_3) - NRR * (t_4 - t_1)}{SDR + 1} \tag{6.20}$$

Substituting the outputs of Eq. 6.10 and Eq. 6.18 in Eq. 6.20, an accurate *Response_delay* value (the actual path delay of P2P mechanism) is obtained. Thereafter, the measured value is used as a path delay instead of the mean path delay in the hybrid TSN systems. The residence time of the *SYNC* message in every TSN-enabled intermediate node is also computed and added besides the *Response_delay* value in the *Correction-field* of the *SYNC* message.

In the symmetric scenario (ideal scenario), the value of NRR and SDR are 0 and 1, respectively. According to Eq. 6.20, the *Response_delay* value in the symmetric scenario is the same as in Eq. 3.1.

Eq. 3.1 is used normally in symmetric frameworks such as wired communications, but it is not feasible in dynamic wireless environments because NRR and SDR values mostly are not 0 and 1, respectively. Regarding the previous analysis, the synchronization precision in asymmetric wireless networking depends on computing accurate values for time and frequency offsets, which ensures that the slave time nearly operates as the grandmaster time.

In general, once a TSN-enabled intermediate node receives a *SYNC* message from the grandmaster, the improved 802.1AS protocol uses the P2P mechanism to compute NRR (i.e. the frequency drift) and an accurate path delay value (i.e. the *Response_delay*) with the TSN neighbouring nodes. The recipient node adds the computed NRR to the aggregated CSRO value, then the added value is inserted into the *CSRO-field* of the *SYNC* message. At the same time, the recipient node adds the summation of its computed *Response_delay* value and the *SYNC*'s residence time into the *Correction-field*. Thereafter, the *SYNC* message is forwarded to the next TSN-enabled node. By combining the values in the *CSRO* and *Correction-fields* with the original grandmaster time that was previously stored in the *Original-timestamp* field of the *SYNC* message, the local time at the TSN-enabled end node can be measured when it gets the *SYNC* message.

Thereby, the slave clock of a TSN end node, which is used for time-stamping the gPTP messages in a hybrid TSN domain, is expressed as follows:

$$Slave\_clock = Timestamp.of.SYNC + Agg.drift + Agg.path\_delay + Agg.residence\_time. \tag{6.21}$$

As stated in Eq. 6.21 and illustrated in Figure 6.2, the time of the slave clock on the left hand side is the summation of the timestamp of the *SYNC* message transmission instant captured by the grandmaster clock (*Timestamp.of.SYNC*), the total drift time (i.e. *Agg.drift*), the aggregated path delays (i.e. *Agg.path_delay*) and the aggregated residence times (i.e. *Agg.residence_time*) on the right hand side.

The computed $Agg.drift$ at the slave clock is defined as a result of the frequency drift between the grandmaster and the slave clock. It is computed using the following equation:

$$Agg.drift = CSRO * (current\_time - last\_sync\_time) \tag{6.22}$$

The CSRO value is obtained from the $CSRO\text{-}field$, the compensated times (i.e. $Agg.residence\_time$ and $Agg.path\_delay$) are obtained from the $Correction\text{-}field$ and $Timestamp.of.SYNC$ is obtained from the $Original\text{-}timestamp$ field.

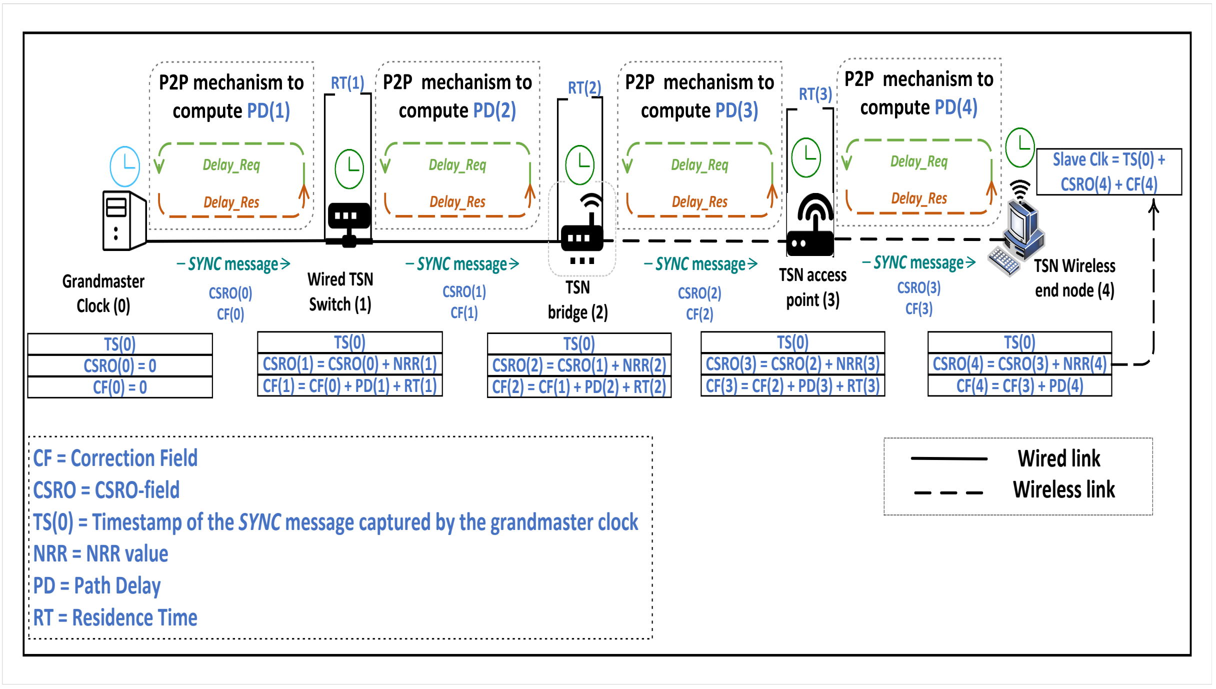

P2P mechanism to compute PD(1)
RT(1)
P2P mechanism to compute PD(2)
RT(2)
P2P mechanism to compute PD(3)
RT(3)
P2P mechanism to compute PD(4)
Delay_Req
Delay_Res
Slave Clk = TS(0) + CSRO(4) + CF(4)
– SYNC message→
CSRO(0) CF(0)
CSRO(1) CF(1)
CSRO(2) CF(2)
CSRO(3) CF(3)
Grandmaster Clock (0)
Wired TSN Switch (1)
TSN bridge (2)
TSN access point (3)
TSN Wireless end node (4)
TS(0) | CSRO(0) = 0 | CF(0) = 0
TS(0) | CSRO(1) = CSRO(0) + NRR(1) | CF(1) = CF(0) + PD(1) + RT(1)
TS(0) | CSRO(2) = CSRO(1) + NRR(2) | CF(2) = CF(1) + PD(2) + RT(2)
TS(0) | CSRO(3) = CSRO(2) + NRR(3) | CF(3) = CF(2) + PD(3) + RT(3)
TS(0) | CSRO(4) = CSRO(3) + NRR(4) | CF(4) = CF(3) + PD(4)
CF = Correction Field
CSRO = CSRO-field
TS(0) = Timestamp of the SYNC message captured by the grandmaster clock
NRR = NRR value
PD = Path Delay
RT = Residence Time
Wired link
Wireless link

FIGURE 6.2: Diagram illustrating how the slave clock is synchronized with its grandmaster using the improved IEEE 802.1AS protocol.

6.3 Path Deviation Delay Filter Test

In the previous section, the *Response_delay* value (i.e. the computed path delay) is calculated to accurately determined SDR and NRR values. It should be noted that the value of SDR is deceptive, because the SDR ratio might give diverged values for *Request_delay* (in the numerator) and *Response_delay* (in the denominator) but an acceptable SDR ratio, especially when dealing with a time-sensitive environment, where the deadline conditions play a critical role. Therefore, to ensure acceptable values for *Response_delay* and to ensure that it is not an outlier value; a filter called Path Deviation Delay (PDD) filter is used to compare it with *Request_delay* values stored in the TSN-enabled intermediate node.

After receiving the *SYNC* message and exchanging P2P messages at the TSN-enabled intermediate node, the *Response_delay* value is computed according to the measured NRR and SDR values. In this filter, the previously stored *Response_delay* values are utilized in a table called Path Delay Table (PDT). The idea is to observe the standard deviation of the values in PDT with and without the computed *Response_delay* value. If the *Response_delay* value causes a decrease or increase in the deviation by more than 30%, the *Response_delay* value is then considered as an outlier value and is excluded because it does not satisfy the TSN application requirements [9] and it might be an indication of traffic congestion or the sender of the *Response_delay* is out of coverage area. At this moment, the node keeps its time until another $SYNC$ message with a successful filter test is coming from a TSN neighbour node. If the $Response\_delay$ value is close to the previous values, the $Response\_delay$ can be used to update the time of the TSN-enabled intermediate clock. It is then inserted into the $Correction\text{-}field$ for the next node until reaching the TSN-enabled end node as shown in Figure 6.3. This filter contributes to removing the effects of random delays. It is worth noting that discarding several consecutive synchronization messages reduces the synchronization precision, thus the filter may not work perfectly in very high congested dynamic environments.

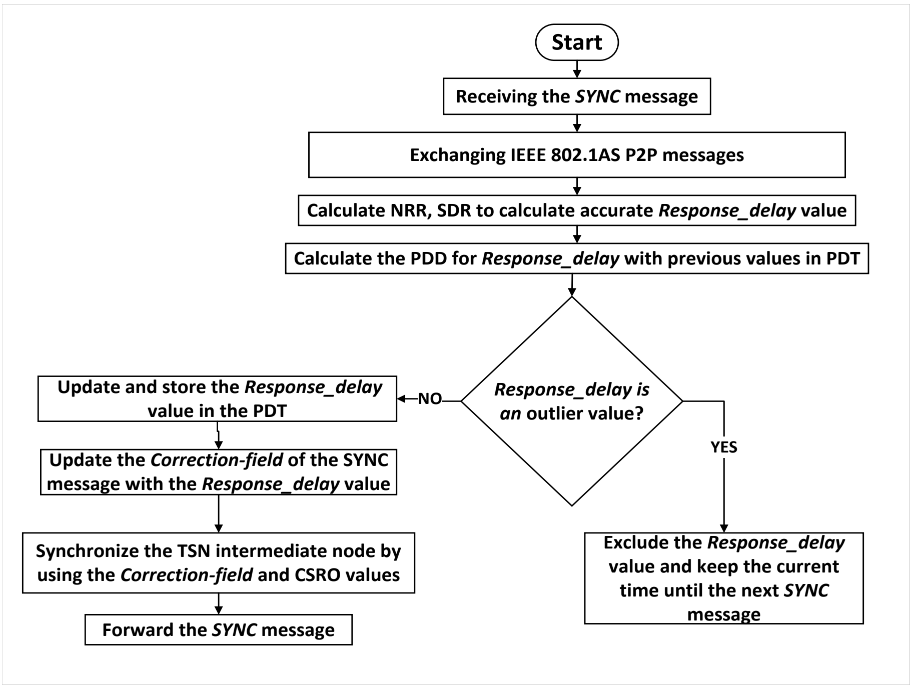

Start
Receiving the SYNC message
Exchanging IEEE 802.1AS P2P messages
Calculate NRR, SDR to calculate accurate Response_delay value
Calculate the PDD for Response_delay with previous values in PDT
Response_delay is an outlier value?
NO
YES
Update and store the Response_delay value in the PDT
Update the Correction-field of the SYNC message with the Response_delay value
Synchronize the TSN intermediate node by using the Correction-field and CSRO values
Forward the SYNC message
Exclude the Response_delay value and keep the current time until the next SYNC message

FIGURE 6.3: PDD filter test

6.4 Hybrid Network Simulation Models for Clock Synchronization Process

In this section, the simulation model and environment used to evaluate the effectiveness of the proposed synchronization protocol are described. Wireless nodes and bridges' TSN synchronization capabilities are combined and tested with already-existing wired TSN models [208]. These wired TSN models have been implemented in the OPNET framework, where the standard node models were identified to be wired TSN-enabled nodes in a wired TSN environment. We summarize the models of the wireless TSN-enabled nodes (i.e. wireless end nodes and bridge models) as shown below.

- Wireless TSN-enabled End node: All packets belonging to the IEEE 802.1AS protocol are based on the WLAN format. Therefore, the existing standard wireless node model is used for modelling the TSN wireless end node.
- Wireless TSN-enabled Bridge: The functionalities of the boundary clock as proposed in the IEEE 802.1AS protocol are applied in the wireless TSN bridge. Thus, the existing bridge model in OPNET is used for modelling the TSN wireless bridges.

By utilizing the proposed IEEE 802.1AS protocol, all models (including Ethernet and wireless models) incorporate the synchronization process and the BMCA algorithm. These models are then re-defined to support the IEEE 802.1Qbv and IEEE 802.1CB protocols [203], which schedule TT streams using TAS and send replicas of the TT messages over redundant routes. It is worth mentioning, broadcasting an *Announce* message in the wireless environment to all neighbours makes a neighbour receive more than one copy of the same message by other nodes located with the sender in the same domain. Therefore, BMCA solves this problem by using an identification in the *Announce* message (i.e. $Ann.ID$), see Figure 5.15. If a node receives an *Announce* message, it keeps the $Ann.ID$ and in the future another *Announce* message with the same ID is simply dropped. After finishing the BMCA algorithm and selection of the grandmaster, $SYNC$ and P2P messages will be sent periodically. Because of their periodicity, their time sensitivity and their importance to keep all slaves synchronized with their reference time, $SYNC$ and P2P messages will be scheduled as TT traffic and inserted as high priority messages in the GCL of each node they pass through.

6.5 Optimization of 802.1AS in a TSN Hybrid Domain

The key procedures of the extended 802.1AS in TSN hybrid systems adhere to the same principles as the standard protocol but there are several optimizations:

- The extended 802.1AS facilitates the interconnection of wired/wireless networking technologies within a single TSN hybrid domain. For this reason, the timing information transmitted through the wireless TSN domain is generalized. This means that the extended protocol conforms to the management control and the format of the transmitted messages. In the contrast, standard 802.1AS is limited to the IEEE 802.3 standard. The major modification is associated with the TSN hybrid domain that combines and unifies the functionalities of the wireless and wired TSN domains rather than supporting one single gPTP domain.

- The extended 802.1AS enables the TSN-enabled node to measure the residence time and the asymmetric path delay to prevent the outlier path delay values and non-deterministic P2P from degrading the performance of the time synchronization process in a wireless environment.

- The extended 802.1AS protocol is designed to support the reliability of the clock synchronization process in the TSN wireless domain. To achieve that, the extended 802.1AS protocol introduces multiple redundant synchronization trees which are used to forward the timing information (i.e. $SYNC$ messages). For example, Figure 6.4 shows two synchronization trees and the timing information can be sent over these trees at the same time. Therefore, in case one of these trees fails, TSN nodes continue to receive the timing information from their grandmaster through the redundant synchronization tree. This procedure serves for protecting the synchronization process and keeping it deterministic. Each synchronization tree is distinguished by the identification of the $SYNC$ message (i.e. $SYNC.ID$), see Figure 5.18. On one hand, if multiple copies of timing information come from the same synchronization tree, the TSN-enabled node synchronizes its time according to the first arriving copy and eliminates the others. On the other hand, if multiple copies of timing information come from different synchronization trees, the TSN-enabled node synchronizes its time according to the first arriving copy and forwards each arrived copy.

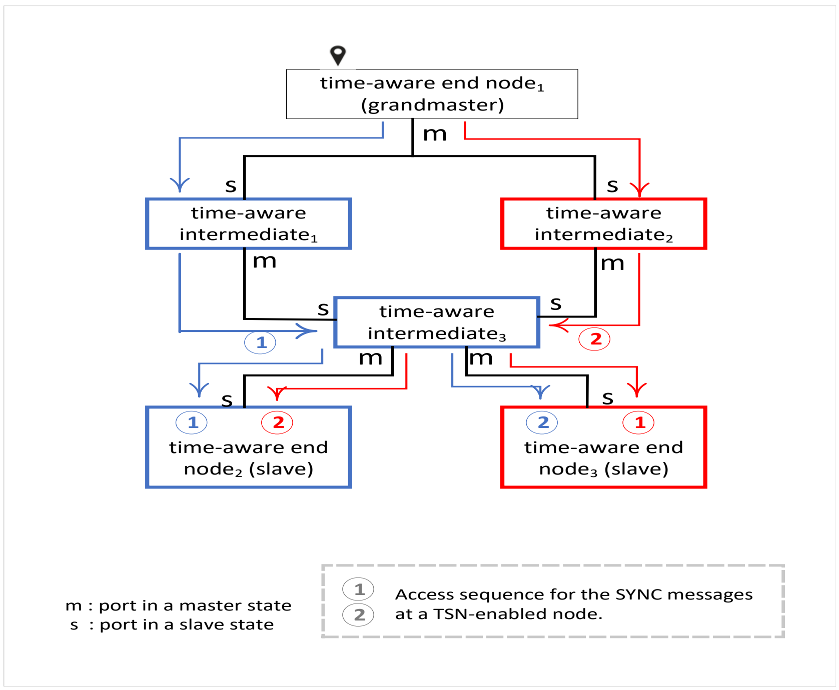

time-aware end node₁ (grandmaster)
m
s
time-aware intermediate₁
m
s
time-aware intermediate₂
m
s
time-aware intermediate₃
s
m
m
1
2
s
1
2
time-aware end node₂ (slave)
s
2
1
time-aware end node₃ (slave)
m : port in a master state
s : port in a slave state
1
2
Access sequence for the SYNC messages at a TSN-enabled node.

FIGURE 6.4: Wireless TSN domain with two redundant synchronization trees.

6.6 Evaluation of the Improved IEEE 802.1AS Protocol

In the hybrid TSN simulation framework, different TSN-enabled nodes will be used including wired and mobile wireless nodes, Ethernet relays and wireless bridges. The improved IEEE 802.1AS was implemented using the OPNET simulation environment, which runs on a PC (Intel i5) with 24 GB of memory.

6.6.1 Simulation Setup

To evaluate the behaviour of the proposed protocol, our grid hybrid modelling approach mentioned in Chapter 4 is used. As illustrated in Figure 4.4, two wireless TSN bridges connect a group of TSN wireless routers and wireless access points. Each access point is connected with mobile and fixed wireless TSN-enabled end nodes. On the other side of the bridges, TSN wired relays are connected with a HMI and monitoring application stations. Data traffic of every TSN end node is sent to a CCU. Then, the CCU forwards the traffic to the HMI and the monitoring application. After that, the HMI sends the processed data back to the CCUs.

6.6.2 Time-Aware Hybrid Network Simulation

The BMCA algorithm is executed in our experimental network for all the TSN-enabled systems. In [208], the authors show the configuration parameters for clock synchronization and how the BMCA algorithm works to select the grandmaster. In the simulation, the HMI is selected to be the grandmaster according to the highest priority and it is set to the OPNET simulation time (i.e. *op_sim_time*()), the adjusted slave times are then utilized as precise times for scheduling other time-sensitive events. It is worth noting that all parameters are user-configurable and can be changed depending on the network specifications.

The proposed protocol is evaluated with different scenarios, including different mobile speeds, different oscillation drifts between the grandmaster and the slave clocks, increasing number of hops between them and different asymmetry ratios. The simulation results of both the proposed and the standard protocols are compared and a detailed analysis is presented. The oscillation drifts are measured in parts per million (ppm) which indicates how much the crystal's frequency deviates from the nominal value [212], [213]. The asymmetry ratio is defined as the ratio in the data rate between the wireless uplinks and downlinks.

Figure 6.5 shows the computed synchronization error in nanoseconds (ns) according to the variation in ppm between the selected grandmaster (i.e. HMI) and wireless TSN mobile host $h17$. The synchronization error is calculated by referring to the OPNET simulation time (i.e. $op\_sim\_time()$). Figure 6.5 shows how the variation in ppm between the grandmaster and the slave clocks affects the synchronization precision. It shows a linear increase in the synchronization error when increasing ppm variation, thus the proposed protocol considers the variation in the frequency by calculating the accumulated value of CSRO at the slave clock. The CSRO value gives the actual deviation in the frequency, which should be considered in the synchronization process. It is obvious from the results of the clock drift synchronization error that it equals 0 in case of no clock-drift or when both node clocks are operating with the same frequency drifts as a result of the same frequencies.

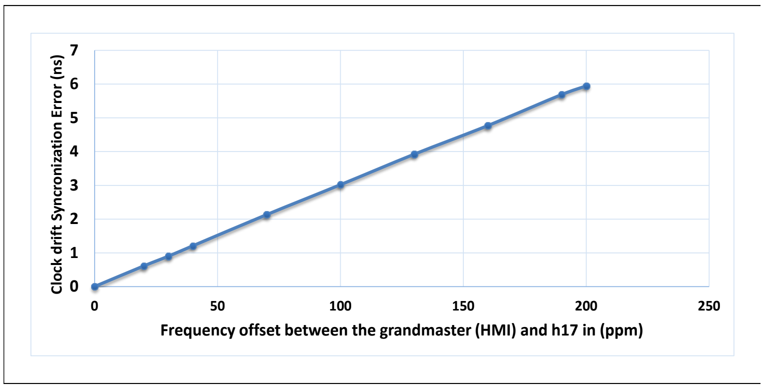


FIGURE 6.5: Clock drift synchronization error between the grandmaster (HMI) and $h17$ (fixed speed = 20 m/s) with variable frequency offsets in (ppm), uplink and downlink = 24 Mbps, asymmetry ratio = 1.

In the following scenarios, ppms of wireless/wired end nodes, wireless bridge and wired router are fixed to 1000, 500 and 200 respectively. Figure 6.6 shows the synchronization error that is produced between HMI and $h17$ at different mobile speeds in random directions. The speed of $h17$ affects the synchronization error because the transmission time for the synchronization messages varies depending on the speed, topology and direction of the node that receives these messages.

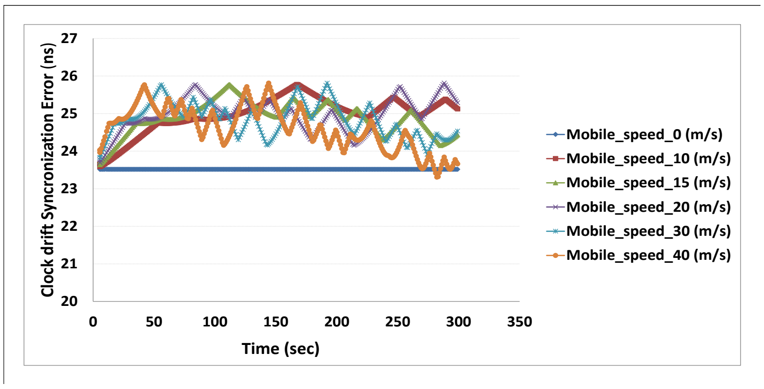


FIGURE 6.6: Clock synchronization error between grandmaster (HMI) and $h17$ for different mobile speeds, uplink and downlink = 24 Mbps, asymmetry ratio = 1.

Figure 6.7 shows the measured synchronization error in $h17$ according to the asymmetry data ratio between $h17$ and its access point (i.e. $ap5$). As shown, the synchronization error for the standard algorithm increases as the asymmetry ratio between the uplink and the downlink increases. In contrast, the proposed IEEE 802.1AS provides accurate synchronization for dynamically changing asymmetric wireless systems compared to the standard protocol. The proposed correction model shows that the average synchronization error is around 0.645 microseconds in average. This is due to the wireless data rate awareness in the proposed correction model, which enables accurate path delay calculation for dynamic changes in uplinks and downlinks and dynamic mobile network topologies.

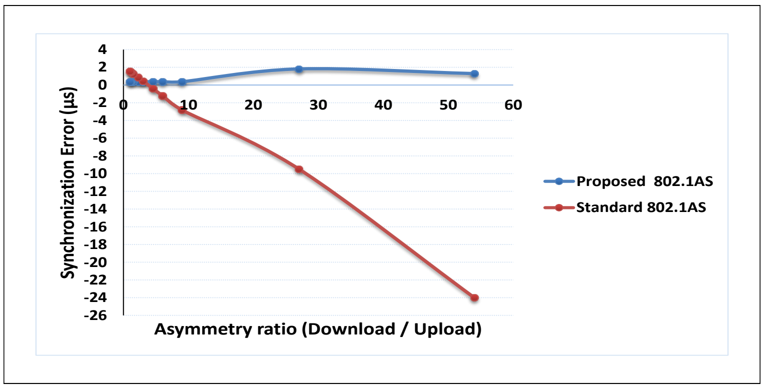


FIGURE 6.7: The synchronization error for the standard and the proposed IEEE 802.1AS protocols when increasing the asymmetry ratios (downlink = 54 Mbps, uplink 1 - 54 Mbps) between $h17$ (speed = 20 m/s) and $ap5$.

Figure 6.8 shows the calculated synchronization error according to the number of intermediate hops between HMI and a selected TSN-enabled node. The synchronization error for the standard and the proposed protocols increases as the number of hops increases. This is as expected, but the proposed correction model shows a smaller synchronization error. The proposed correction model could be utilized to minimize the upward increase in the synchronization error by selecting the best path out of different paths between the grandmaster and a slave end node.

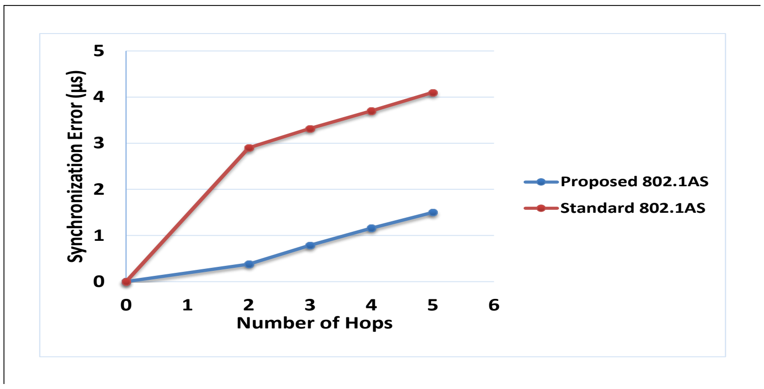
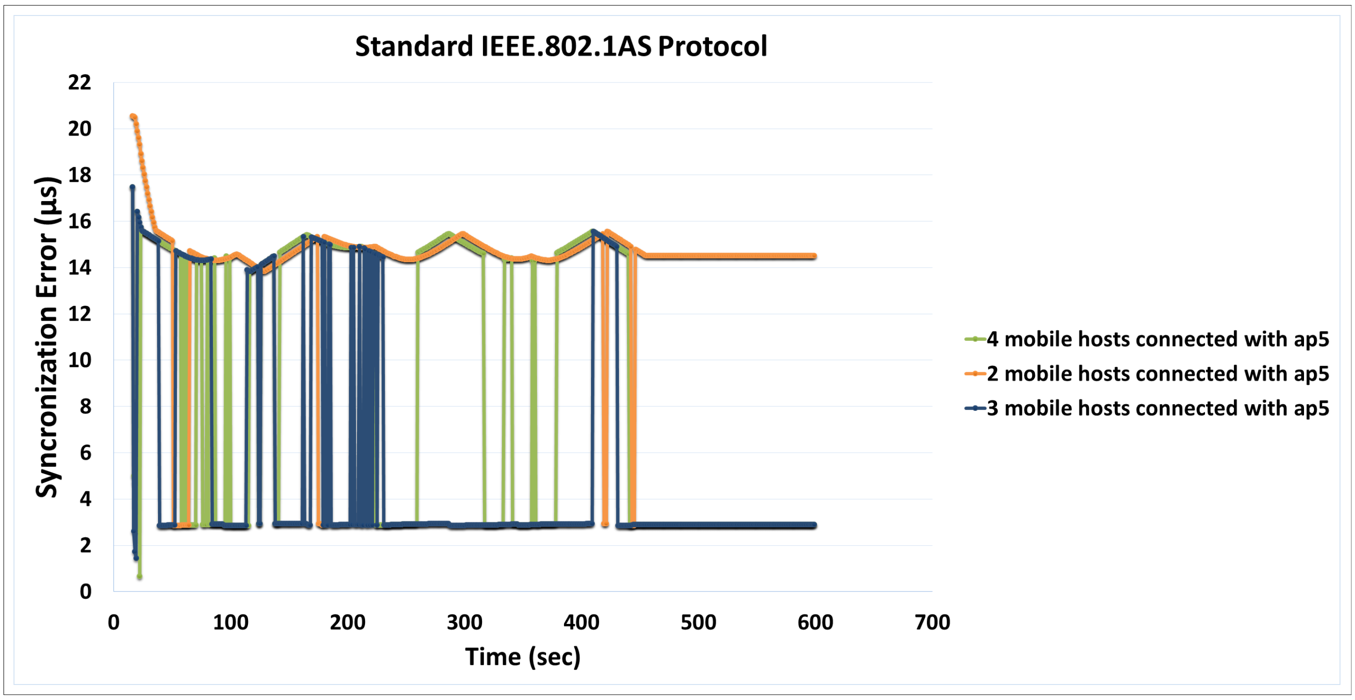
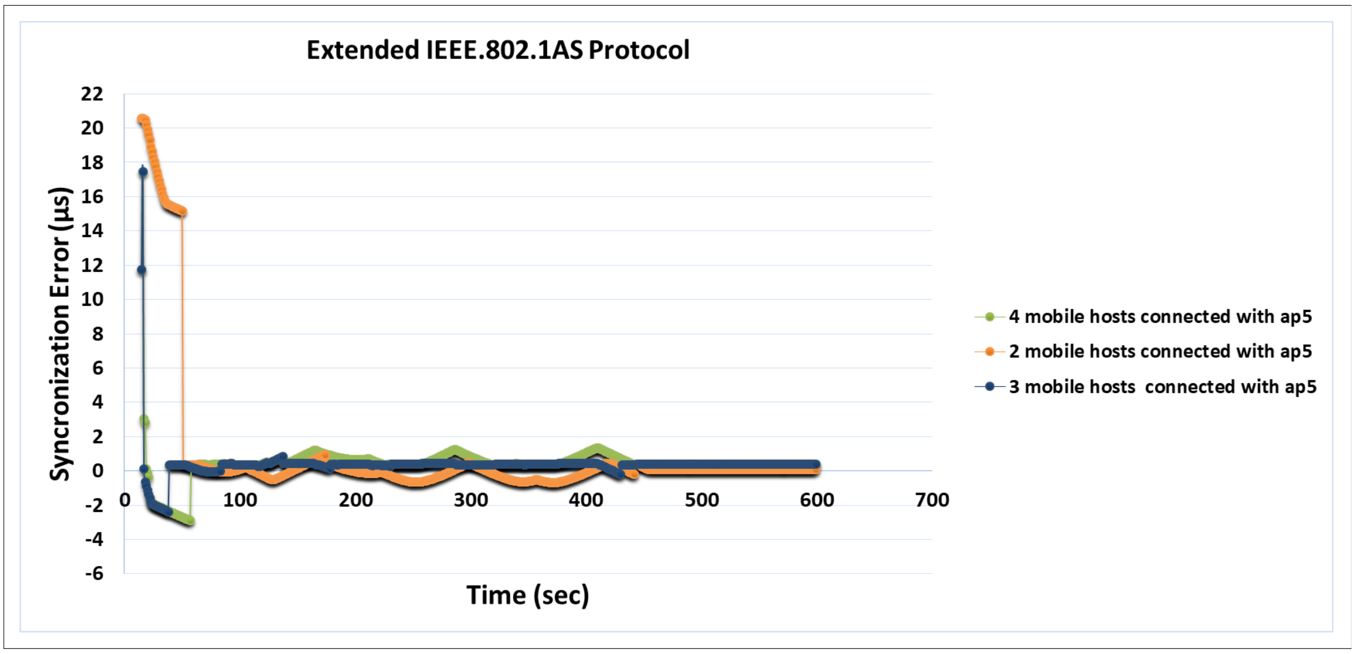


FIGURE 6.8: The synchronization error for the standard and the proposed IEEE 802.1AS protocols when increasing asymmetry ratios (downlink = 54 Mbps, uplink 1 - 54 Mbps) between $h17$ (speed = 20 m/s) and $ap5$.

Another scenario to evaluate the proposed protocol and compare it with the standard protocol is illustrated in Figure 6.9. It shows the synchronization error for $h17$ in case of increasing the number of mobile wireless TSN-enabled nodes (i.e. hosts) that are connected with its access point (i.e. $ap5$). The actual synchronization precision in $h17$ is not correctly computed by the standard IEEE 802.1AS protocol. Significantly more accurate time synchronization is achieved by the proposed IEEE 802.1AS protocol by incorporating deterministic delays and considering the oscillation factor with the grandmaster, besides eliminating asymmetry and outlier values in the path delay values. Figure 6.9a shows that in the standard protocol, the synchronization error stabilizes at the range of 2.899 to 14.522 microseconds when increasing the number of mobile hosts that are located in the same local network. This relates to the asymmetric behaviour of the standard protocol. On the other hand, at the beginning of the simulation, Figure 6.9b shows positive and negative synchronization errors. The reason is that the compensated time upon receiving the synchronization message at the end node is less or more than op_sim_time, respectively. But over time, every end node receives more synchronization messages. Therefore, the compensated time is computed accurately and the synchronization error stabilizes to be in the range 0.0558 to 0.5251 microseconds although the number of the mobile hosts increases.



(A) Standard Protocol.



(B) Proposed Protocol.

FIGURE 6.9: The synchronization error for $h17$ by using the standard (A) and the proposed (B) protocols with a variable number of wireless TSN-enabled nodes (range speed 0 - 30 m/s) connected with $ap5$, uplink and downlink = 24 Mbps, asymmetry ratio = 1.

Chapter 7

Task and Message Scheduling Algorithm for Hybrid TSN Systems

7.1 Introduction

As mention in Chapter 1, systems with wireless networks (e.g., WLAN, WSN and ad hoc networks) have attracted considerable attention due to their scalability and flexibility. Therefore, there is a trend to apply them in conjunction with TSN technology [77], [202] in various industrial systems such as robotics, vehicular applications and machine control systems. These systems need a high level of temporal predictability in data transmission especially for real-time applications that consist of periodic tasks and TT messages used to send information between the interconnected tasks.

In Chapter 5, the TSN standards are implemented in wireless nodes besides the TSN-based wired nodes to fulfil the requirements of time-sensitive applications in a hybrid environment. The TAS feature is used to support scheduled traffic, where each TSN node has a certain number of queues dedicated to transmitting different types of traffic according to a GCL list. The GCL typically allocates a higher priority to TT streams than others like AV and BE streams. In this chapter we are mapping GCL to wireless components to specify the egress time (i.e. the gate is open) to transmit TT messages over a wireless medium. GCL guarantees that the allocated time slot for a TT message will not be occupied by any other message.

The scheduling of TT messages and task scheduling problem in wireless environments is NP-complete. Increasing the number of computation tasks and TT messages leads to an exponential increase in the number of possible task schedules. Therefore, finding a valid task scheduling solution, which is defined as a case of distributing all tasks to potential wireless hosts, is considered a challenge to fulfil the deadline restrictions, especially in an environment that suffers from surrounding signal interference. Thus, it becomes necessary to reduce the search space that to find a feasible solution.

The majority of prior research to find a global feasible solution for real-time TT networks considered the task scheduling constraints with fixed routing as input to their algorithms. Other research considered the routing possibilities [43], which in turn leads to better solutions that satisfy the requirements of real-time systems. The authors in [214], [215] use ILP-based algorithms, which are quite slow and not scalable to address large real-time systems, with a focus on joint routing and scheduling for the TT schedule computation. Others use the interference as a metric to find a feasible task schedule [216]. In order to combine the scheduling, interference and routing constraints into one adaptive algorithm, a list-based task scheduling algorithm named Task and Message Scheduling (TMS) algorithm is proposed to generate a feasible task and message scheduling solution in $WirelessTSN$ networks.

In contrast to state-of-art task scheduling solutions, TMS generates port-specific GCLs imposing routing, interference and scheduling constraints at the same time. It transforms the mentioned constraints into one set of constraints and solves the task scheduling problem considering precedence constraints of TT tasks in a single step. The basic objective of TMS is to meet TT traffic deadlines while minimizing TT transmission makespan and overhead. The makespan is determined by the time for the execution of a leaf task, which is a task without message forwarding. Additionally, it gives the option to distribute safety-critical applications across all available hosts rather than just one, helping to prevent host overload on particular hosts. This approach is beneficial for mission-critical applications (e.g. Unmanned Aerial Vehicles (UAVs)) that operate in wireless systems and require considerable amounts of computational power.

Since the wireless networks use a shared medium, the interference as a result of sending TT messages at the same time leads to decreased capacity and reliability of the network [153]. This is one of the main problems in finding a global task and message scheduling solution for hard real-time applications. The proposed algorithm solves this problem via scheduling the TT messages, which use wireless physical links, into several time slots. In each time slot, the TT messages can be transmitted at the same time without causing undesired mutual interference. Clearly, it is often effective to find an optimal route of minimum latency for each TT stream, and in each time slot only a subset of TT streams can be scheduled.

In order to integrate TSN wireless nodes into an accurate time-synchronized network, we use our extended IEEE 802.1AS protocol from Chapter 6 to establish a precise global time [27]. A single channel is used to send and receive the TT messages and the time is slotted into fixed-size time slots. Each TT message is fragmented according to the time slot size along the wireless route and each fragmented message is assigned to a specific time slot as long as it is transmitted through its path. The proposed algorithm guarantees using the spatial distribution of the communication activities that all TT messages in each slot can be transmitted simultaneously without causing undesired mutual interference.

Section 3.8 analyzes the physical interference and protocol models in [153]. If no other node is simultaneously sending its data inside the interference range of the receiver (r), then the data transmission between sender (s) and receiver (r) is successful, according to the protocol model. In the physical model, data transmission between s and r is successful if the SINR value at r is greater than a predetermined communication threshold value, or β, whose value relies on the channel characteristics and the communication system standard. Because it offers a somewhat accurate depiction of a wireless environment, the physical interference model is used in this work. The physical model considers the channel effects like SINR conditions, ambient noise power and path loss, which leads to higher network throughput and reliability.

In order to have a strong foundation for comparison, we also created the Minimum traffic Load Task Scheduling (MLTS) algorithm [21], [22], which chooses the routes with the least amount of traffic load to transfer the messages throughout the task scheduling process. Another algorithm is the Min-Min Task Scheduling (MMTS) algorithm [20] which does not consider the traffic cost due to the message transmissions. MMTS is a traditional heuristic algorithm, which first sorts the tasks according to the top-level values. The algorithm proceeds by assigning the task to the wireless host that produces the minimum completion time. The same procedure is repeated by MMTS until all tasks are scheduled.

Several experimental tests with different topologies, traffic loads and variable interference environments are implemented. The simulation results show significant improvement in the reliability, network resource-saving and makespan of the proposed algorithm as compared with the MLTS and MMTS algorithms. All notations used in this chapter are shown in Table 7.1.

The rest of this chapter is organized as follows. The task and message scheduling problem of TMS is formulated in the next section. Section 7.3 defines the time slot message scheduling model. The proposed algorithm is described in detail in Section 7.4. Finally, Section 7.5 discusses the experiments and the evaluation.

7.2 Problem Formulation of TMS

To start the task and message scheduling process, firstly we assume that all wireless TSN-enabled nodes are fixed and spread randomly in the Euclidean area and their clocks are synchronized with the grandmaster clock. One available channel is shared for all bidirectional wireless links.

The wireless systems normally decompose a real-time application into several tasks that are scheduled on several wireless hosts to be executed. TT messages are used to send data between the periodic tasks. The task and message scheduling problem is a critical issue in $WirelessTSN$ networks because the message transmission cost is integrated with the task execution time. Hence, it is imperative to choose the most suitable host for each task to run on and to make sure that all tasks are completed in hosts before their deadlines, which is the fundamental objective of the TMS algorithm.

As mentioned in Section 3.11, there are three types of data traffic transmitted over the wireless paths. These types are (i) BE streams, (ii) AV streams and (iii) TT streams. BE and AV streams do not require deadline guarantees. To avoid the interference of BE and AV streams with TT streams, the proposed algorithm employs a GCL which is defined for each port of a wireless TSN node. The proposed algorithm considers only TT messages, while other types of data are scheduled when no TT messages are transmitted over the wireless paths.

TMS considers our hybrid modelling approach from Chapter 4 including the architecture graphs (i.e. the grid and ring graphs) and the application graph as inputs to schedule a set of tasks T to the available wireless hosts for each task t, whereas the task and message scheduling model addresses the following constraints:

1. Resource scheduling constraint: Each computational task t is executed only on one wireless host h.
2. Path-attribute constraint: To eliminate routing loops, each TT message is allowed to pass a wireless TSN-enabled intermediate node (i.e. a wireless router or an access point) only once toward the receiver host.
3. Physical interference constraint: The dedicated time slot for each TT message uses a certain path only if the interference induced by other messages does not violate the SINR threshold value.
4. Precedence constraint: Each task t is ready to be executed at time t_{rt} after the arrival of all messages M_t sent from its parent tasks. The ready time of task t is equal to the last arrival time. It is expressed as shown in Eq. 7.1:

$$t_{\text{rt}} = max_{\text{m} \in \text{M}_t}(m_{\text{arrival}}) \tag{7.1}$$

TABLE 7.1: Notations used in Chapter 7

Symbol	The description of the symbol	Domain
T	Set of all computational tasks in the application graph	$T \in G_{App}$
M_{TT}	Set of all TT messages in the application graph	$M_{\mathrm{TT}} \in G_{App}$
t_{rt}	Ready time of the task t to be executed	$t \in T$
t_{et}	Execution time of the task t	
t_{dl}	Deadline of the task t	
$t.AvailableHosts$	Available hosts to allocate the task t	
P_t	Period of task t	
M_t	Set of TT messages from the sending task t	
m_{sz}	Message size	$m \in M_{\mathrm{TT}}$
m_{e2eD}	Message end to end delay	
m_{comm}	Message communication cost	
m_{IT}	Injection time of TT message into the medium	
m_{arrival}	Arrival time of the message at the receiver	
Fr_m	Total number of fragments of a TT message m	
fr_m	A fragment of a TT message m	
$l_{\mathrm{i}}(fr_m)$	l_i is the path used by a fragment of a TT message m	
r_m	A tested route for a TT message m	
$r^*{}_m$	Selected route for a TT message m	
$parent(t)_m$	Parent of task t that sends the TT message m	$parent(t) \in T$
$m_{parent(t)}$	Message that comes from that parent	
n	Time slot number	
slt_{d}	Time slot duration	
$Slt(n)$	Set of paths in the n^{th} time slot	
L	Set of wireless communication paths	$L \in G_{Arc}(N, L)$
$I_{\mathrm{Slt}}(Slt(n), b)$	Total interference induced on b by all senders in $Slt(n)$	$l_i \in L$
$I_{\mathrm{Slt}}(Slt(x, n), b)$	Interference induced on b by sender x in $Slt(n)$	
$P_{tr}(a)$	Transmission power of node a	$a \in N$
$P_{rcv}(a, b)$	Reception power at node b from node a	$a, b \in N$

m_{arrival} is defined as the instant of time when a message m from a parent task arrives at task t. It is defined as:

$$m_{\text{arrival}} = m_{\text{IT}} + m_{\text{e2eD}} \tag{7.2}$$

Where, m_{IT} and m_{e2eD} denote the injection time and the transmission time of the message m, respectively.

5. Message period constraint: TMS considers the period of the interconnected tasks. Therefore, TMS dedicates time slots to transmit each message by taking into account the period of other messages that share the same path.

6. Deadline constraint: The instant of time for the host to finish executing the task t after receiving all messages must be less than the deadline for that task.

$$t_{\text{rt}} + t_{\text{et}} \leq t_{\text{dl}} \tag{7.3}$$

Where, t_{et} and t_{dl} refer to the execution time and the deadline of a task t, respectively.

7.3 Time Slot Message Scheduling Model

Signal interference would be induced in the case of two or more simultaneous transmissions. Selecting the interference model has a vital impact on the efficiency of the message scheduling. To capture the aspects of TT wireless systems, the time slot message scheduling in the TMS adopts the physical interference model [153] because it aims to maximize the amount of successfully received data and considers channel effects like path loss, shadowing, fading and the received power. The physical model operates to send multiple messages simultaneously using fixed duration time slots in order to maximize the amount of successfully received data and prevent the influence of the induced interference as possible.

Assume $d(a,b)$ to be the length (i.e. the transmission distance) of the path (a,b), where (a,b) represents the path between two wireless nodes a and b. The transmit power of node a is denoted by $P_{tr}(a)$. The received power $P_{rcv}(a,b)$ at node b of the transmitted signal from the node a is:

$$P_{rcv}(a,b) = \frac{P_{tr}(a)}{d(a,b)^{\alpha}} \tag{7.4}$$

α is the path loss, where the path loss increases with the fourth power of the transmission distance [217].

In the physical interference model, the message that uses the path (a,b) will be received correctly at node b if and only if the following condition is satisfied:

$$\frac{P_{rcv}\left(a,b\right)}{N + \sum_{p \in Paths} P_{rcv}\left(c,b\right)} \geq \beta \tag{7.5}$$

Where $P_{rcv}(a,b)$ represents the received power on b from a, $P_{rcv}(c,b)$ represents the interference on b by c. The ambient noise power is N, $Paths$ is a subset of wireless paths used by other communication messages simultaneously with path (a,b) to the same receiver b. β is the SINR threshold value, which ensures that the message will be successfully received at b. Thus, path (a,b) is inserted in a specific time slot with other $Paths$ if the computed value on the left side of Inequality 7.5 is equal to or more than β. Otherwise, the message that uses path (a,b) uses the next time slot that satisfies this condition.

Let the interference $P_{rcv}(x,b)$ on b by a sender x be expressed as $I_{\text{Slt}}(Slt(x,n),b)$, then the total interference on b by all other senders (i.e. not the sender a) in the n^{th} time slot ($Slt(n)$) is expressed by:

$$I_{\text{Slt}}(Slt(n),b) = \sum_{x \in Slt(n)} I_{\text{Slt}}(Slt(x,n),b) \tag{7.6}$$

$Slt(n)$ is considered as a feasible time slot for a message that uses path (a,b) if the total interference on b by other senders in $Slt(n)$ keeps the value on the left side of Inequality 7.5 higher than β.

7.4 Task and Message Scheduling Algorithm

In this section, we present the TMS algorithm to solve the problem of message scheduling and scheduling the task on several wireless TSN hosts. It aims to find a feasible global solution to minimize the makespan of TT traffic in a *WirelessTSN* network by taking into consideration the constraints that are mentioned in Section 7.2. TMS, in general, assigns a host for each task gradually based on the selection of routes that deliver messages as early as possible. TMS is described as follows:

TMS algorithm (General overview): For each unscheduled task t, TMS computes firstly its top-level cost tl_t. Eq. 7.7 defines a path length $pl(t_a,t)$ from t_a to task t, where t_a is considered as the root task (a task has no incoming messages) for the unscheduled task t (the child task). $pl(t_a,t)$ is the sum of the costs of vertices (i.e. t_{et}) and edges (i.e. m_{comm}) of a path from task t_a to t. tl_t in Eq. 7.8 represents the longest path found from t_a to t in G_{App}.

$$pl_{(t_a,t)} = \sum_{t \in path(t_a,t)} t_{\text{et}} + \sum_{m \in path(t_a,t)} m_{\text{comm}} \tag{7.7}$$

$$tl_t = max(pl_{(t_a,t)}) \quad \forall t_a \in parent(t) \tag{7.8}$$

For instance, if we consider in Figure 4.2 that the costs of the edges and vertices of G_{App} are 20 and 200, respectively, then tl_{t_3} will be equal to 0 (t_3 is a root task and has no incoming messages). The longest path to t_0 is (t_3 -> t_2 -> t_1 -> t_0), thus tl_{t_0} = 200+20+200+20+200+20 = 660. After that R-TMS sorts all unscheduled tasks in ascending order according to their top-level values.

After that, TMS sorts the unscheduled tasks from lowest to highest top-level values (line 3 of TMS algorithm (General overview)). The Task and Message Scheduling Procedure (Algorithm 1) will be repeated until finishing the scheduling all sorted tasks, then the makespan of all tasks is computed (lines 4-7). The final makespan is the instant of time when all computational tasks are executed.

TMS algorithm (General overview)

```
1: $makespan \leftarrow 0$
2: compute $top\_level$ ($tl$) of each unscheduled_task $t$
3: $T_{\text{sorted}} \leftarrow \text{sort}_{\text{tasks}}$(Sorting tasks on $tl$ ascendant order)
4: for $t \in T_{\text{sorted}} unscheduled$ do
5:     Call Algorithm 1 Task and Message Scheduling Procedure (task $t$)
6: end for
7: return global makespan
```

Pseudo-code of TMS procedure for scheduling all computation tasks of G_{App}.

Task and Message Scheduling Procedure (Algorithm 1): For each unscheduled task t, the incoming TT messages are determined first. If task t waits for TT messages from parent tasks (constraint # 4), TMS will firstly schedule all the previous tasks (lines 1-5 of Algorithm 1). If all the parent tasks are already scheduled then TMS works on assigning that task on one of its available hosts ($t.AvailableHosts$).

TMS initializes t_{rt} and m_{arrival} to 0. In order to select the best available host $h \in t.AvailableHosts$ for the unscheduled task t, TMS works firstly to find all routes (i.e. R) between the sender host ($parent(t)_m$) and the receiver host (h) while taking into consideration constraint # 2 and constraint # 5, see lines 6-18 of Algorithm 1.

Equation 7.9 is used in lines 13-17 to compute the number of times (i.e. $rep_{parent(t)}$) a parent task (i.e. $parent(t)$) should be repeated before starting to send its message to the child task t. P_t and $P_{parent(t)}$ denote the period of task t and its parent task, respectively. $Ceil(\frac{P_t}{P_{parent(t)}})$ method returns the smallest integer no smaller than the fraction of t and $parent(t)$ periods.

$$rep_{parent(t)} = \begin{cases} \frac{P_t}{P_{parent(t)}}, & \text{if } mod(P_t, P_{parent(t)}) = 0, P_t \geq P_{parent(t)} \\ Ceil(\frac{P_t}{P_{parent(t)}}), & \text{if } mod(P_t, P_{parent(t)}) \neq 0, P_t > P_{parent(t)} \\ 1, & \text{if } P_t < P_{parent(t)} \end{cases} \tag{7.9}$$

For each route $r_m \in R$ a **Message Link Scheduler (Algorithm 2)** is applied. Algorithm 2 is used to compute the message end-to-end delay (i.e. m_{e2eD}) for each route r_m after scheduling the message on time-slotted paths. The scheduler begins to schedule the message at the n^{th} time slot ($Slt(n)$), where $n = \frac{m_{\text{IT}}}{slt_{\text{d}}}$, m_{IT} denotes the message injection time from the sender host and slt_{d} denotes the time slot duration. TMS fragments the message transmission time (i.e. m_{comm}) into a number of fragments Fr_m according to the time slot duration time slt_{d} (lines 1-2 of Algorithm 2). For example, if a message m needs 180 ms to reach the recipient task t and slt_{d} = 60 ms, then the number of fragments is 180/60 = 3 fragments. For each path l_i belonging to the route r_m , TMS applies the feasible time slot algorithm (Algorithm 3) for all fragmented messages Fr_m on that path l_i. The process is repeated and n increases starting from the first fragment of the first path l_1 until the last fragment of the last path l_h in the route r_m, where r_m is denoted as $r_m = \{l_1, l_2, l_3, \ldots, l_h\}$ (lines 3-13).

Algorithm 1 Task and Message Scheduling Procedure(task t)

```
Input: $G_{App}$, $G_{Arc}$
Output: makespan ($t$)
 1: if task $t$ is unscheduled and waits TT messages from its parent tasks then
 2:   for $m \in M_t$ do
 3:     Call Task and Message Scheduling Procedure ($parent(t)_m$)
 4:   end for
 5: else if $all\_parent(t).scheduled$ then
 6:   $RT \leftarrow 0$
 7:   for $h \in t.AvailableHosts$ do
 8:     for $m \in M_t$ do
 9:       $t_{\text{rt}} \leftarrow 0$
10:       $m_{\text{arrival}} \leftarrow 0$
11:       $R = routes(parent(t)_m, h)$
12:       if $P_t > P_{parent(t)}$ then
13:         $rep_{parent(t)} = \frac{P_t}{P_{parent(t)}}$
14:       else
15:         $rep_{parent(t)} = 1$
16:       end if
17:       $m_{\text{comm}} = \frac{m_{parent(t)}}{BW}$
18:       for $r_m \in R$ do
19:         $m_{\text{IT}} \leftarrow Find(parent(t)_{\text{et}} + P_{parent(t)} * (rep_{parent(t)} - 1))$
20:         Call Algorithm 2 Message Link Scheduler ($r_m$, $m_{\text{IT}}$, $m_{\text{comm}}$)
21:         $arrival \leftarrow m_{\text{IT}} + m_{\text{e2eD}}$
22:         if $arrival + t_{\text{et}} > t_{\text{dl}}$ then
23:           invalid route, go to the next route
24:         end if
25:         if $arrival < m_{\text{arrival}}$ then
26:           $m_{\text{arrival}} \leftarrow arrival$
27:           $r^*_m \leftarrow r_m$
28:         end if
29:       end for
30:     end for
31:     $RT \leftarrow max(RT, m_{\text{arrival}})$
32:     if $t_{\text{rt}} > RT$ then
33:       $t_{\text{rt}} \leftarrow RT$
34:       $t.runs\_on \leftarrow h$
35:       $makespan \leftarrow max(makespan, t_{\text{rt}} + t_{\text{et}})$
36:     end if
37:   end for
38:   return $makespan$
39: end if
```

Pseudo-code of Algorithm 1 used to schedule a computation task to an available host $h \in t.AvailableHosts$.

Algorithm 2 Message Link Scheduler ($r_m, m_{\text{IT}}, m_{\text{comm}}$)

```
Input: $r_m, m_{\text{IT}}, m_{\text{comm}}, slt_{\text{d}}$
Output: $m_{\text{e2eD}}$ (Scheduling the message on time-slotted paths)
    Initialize : $n = \frac{m_{\text{IT}}}{slt_{\text{d}}}$, $m_{\text{e2eD}} = 0$
 1: Fragment $m_{\text{comm}}$ according to time slotsize ($slt_{\text{d}}$)
 2: $Fr_m = \frac{m_{\text{comm}}}{slt_{\text{d}}}$
 3: for each path $l_i \in r_m$ do
 4:     for each fragment $fr_m \in Fr_m$ do
 5:         Call Algorithm 3 Feasible Time Slot ($l_i(fr_m), n$)
 6:         B = mod ($m_{\text{comm}}, slt_{\text{d}}$)
 7:         if B == 0 then
 8:             $m_{\text{e2eD}} = (n)* slt_{\text{d}} + slt_{\text{d}}$
 9:         else
10:             $m_{\text{e2eD}} = (n)* slt_{\text{d}}$ + B
11:         end if
12:     end for
13: end for
14: return $m_{\text{e2eD}}$
```

Pseudo-code of Algorithm 2 used to schedule a TT message on time slotted wireless paths of a certain route and computation of end-to-end delay.

Feasible Time Slot (Algorithm 3): It adopts the physical interference model to find a feasible time slot used to transmit every $fr_m \in Fr_m$ through its path $l_{\text{i}}(fr_m)$. Algorithm 3 checks two cases for $Slt(n)$, where n is the current time slot number. In the first case $Slt(n)$ is empty, $l_{\text{i}}(fr_m)$ will be inserted into the empty slot and its slot number (n) will be returned to Algorithm 2 (lines 2-5 of Algorithm 3). In the second case $Slt(n)$ is not empty and Algorithm 3 computes the interference on $l_{\text{i}}(fr_m)$ by every path $l_j \in Slt(n)$ that has already been scheduled in that slot. At this moment, the SINR value is computed according to the total induced interference in that slot [4]. If the computed SINR value is less than β, the algorithm checks the next slot until finding a feasible slot (the computed SINR value is more than β), thus, $l_{\text{i}}(fr_m)$ will be inserted into that slot and its n will be returned to Algorithm 2 (lines 6-21). At the end of the algorithm, all paths in each slot can be used to transmit their fragments successfully at the same time, so that the influence of the interference is prevented as much as possible (constraint # 3).

Algorithm 3 Feasible Time Slot ($l_i(fr_m)$, n)

```
Input: $l_i(fr_m)$, $n$ (current time slot), $P_{tr}$, $\alpha$, $N$, $\beta$
Output: $n^*$ (next time slot)
  Initialize : find_slot = false
 1: while find_slot == false do
 2:   if $Slt(n)$ == null then
 3:     find_slot = true
 4:     $n^* = n$
 5:     return $n^*$
 6:   else
 7:     for each sender $x \in Slt(n)$ do
 8:       compute $I_{\text{slt}}(Slt\{x\}, l_i)$
 9:       $I$ += $I_{\text{slt}}$
10:       compute $P_{rcv} = \frac{P_{tr}}{{length_{\text{l}_i}}^{\alpha}}$
11:       compute SINR = $\frac{P_{rcv}}{(N+I)}$
12:     end for
13:     if SINR $\geqslant \beta$ ($Slt(n)$ is feasible for path $l_i(fr_m)$) then
14:       Add $l_i(fr_m)$ on $Slt(n)$
15:       find_slot = true
16:       $n^* = n$
17:       return $n^*$
18:     else
19:       increment $n$
20:       Continue
21:     end if
22:   end if
23: end while
```

Pseudo-code of Algorithm 3 used to check the feasibility of the time slot of each fragment's wireless path. The algorithm returns the last incremented slot number.

In Algorithm 2: After calling Algorithm 3 and finding $Slt(n)$ of every $l_{\text{i}}(fr_m)$ in the route r_m, the last value of n is used to compute m_{e2eD} of the route r_m (lines 3-13 of Algorithm 2).

In Algorithm 1: After calling algorithm 2, TMS finds the route that forwards a message with the minimum m_{e2eD} and takes into consideration the constraints # 5 and # 6 from all possible routes (lines 19-30 of Algorithm 1). If the minimum m_{e2eD} violates constraint # 6 then the algorithm looks to the next best route (lines 23-25). The route that has the minimum arrival time m_{arrival} will be selected and r^*_m is updated accordingly (lines 26-29).

After selecting the best route which leads to the least delay for every incoming message to the recipient host h (lines 9-31 of Algorithm 1), the unscheduled task start time t_{rt} at the available host h is continuously updated until receiving all incoming messages (line 32). The task and message scheduling procedure is repeated for all available hosts in $t.AvailableHosts$, then the unscheduled task t is assigned to host h that leads to the minimum t_{rt} (lines 32-34). The makespan value will be updated (line 35) until scheduling all tasks T in the hyper-period $HP.G_{App}$. In the end, the makespan is computed to represent the time that is required to schedule all tasks of the system model within one complete execution of G_{App}.

$HP.G_{App}$ is defined as the Least Common Multiple (LCM) of the periods P of all tasks as shown in Eq. 7.10. $HP.G_{App}$ is the smallest interval of time after which the periodic patterns of all tasks are repeated.

$$HP.G_{App} = LCM(P_0, P_1, P_2, P_3...., P_T) \tag{7.10}$$

$$rep_t = \frac{HP.G_{App}}{P_t} \tag{7.11}$$

rep_t in Eq. 7.11 is the number of times task t will be repeated within one complete execution of G_{App}.

For example, if all tasks have the same period (e.g. 200 ms), the computed $HP.G_{App}$ is equal to 200. Therefore, every task should be repeated one time until starting the next complete execution of G_{App}.

7.4.1 Example

The state-of-the-art MLTS and MMTS algorithms have been created, as was described at the beginning of this chapter, in order to effectively compare with TMS. The main difference between TMS and MLTS is that TMS selects the routes which lead to the minimum message arrival time after scheduling the message on time-slotted paths. It computes the message delay caused by scheduling the message on a route after continuous updating and inserting all previous messages on time-slotted paths that make up this route. This gives an accurate value for the time of arrival of the message. MLTS does not search for the available routes of sending the message in terms of which one leads to the least delay, but it searches for a route that leads to the least load and has the least congestion traffic.This in turn leads to choosing longer routes that may not meet deadline constraints and consume more energy than in TMS.

MMTS does not consider the congestion cost or message arrival latency due to the message transmissions. MMTS is a heuristic algorithm, which firstly computes the top-level of all tasks, and then it sorts the task with the minimum top-level has the highest priority. The algorithm proceeds by scheduling the sorted task to the host that produces the minimum completion time, where the completion time is computed by adding the task execution time to the ready time of that host. Afterwards, the ready time of the host will be updated. The same procedure is repeated for the next task until having scheduled all tasks.

MMTS is not an optimal algorithm for real-time applications because it does not consider the latencies of the message transmissions and the congestion traffic when it addresses task scheduling solutions. In other words, MMTS's solutions are initiated before starting the implementation of the tasks on the proposed hosts, therefore, the delay resulting from the signal interference and the continuous updating of scheduling the messages into their time slots are not considered. Thereby, this may result in a delay in the scheduling of all tasks, in turn, MMTS increases the likelihood of failure to schedule all tasks within their deadlines.

To illustrate the idea of how TMS, MLTS and MMTS work, a simple application graph as shown in Figure 7.1 and the gird architecture graph (see Figure 4.4) are used as inputs. Figure 7.1 consists of three tasks that are connected using three interconnected messages. The bandwidth of the wireless links is set to be 2 units per millisecond, and it is used to compute m_{comm} of each TT message. For example, if message size m_{sz} of message m_{10} is 6, then m_{comm} of m_{10} will be $6/2 = 3$ ms. t_{et} is the execution time for all tasks and it is set to 4 ms. $P_0 = P_1 = P_2 = 40$ ms. Thereby, according to Eq. 7.10, $HP.G_{App}$ for this graph is 40 ms. t_0,

t_1 and t_2 will be repeated one time within once complete execution of G_{App} according to Eq. 7.11. P_{tr}, N, α and β values are set to be 100 mW, $1x10^{-10}$mW, 4 and 10 dB respectively.

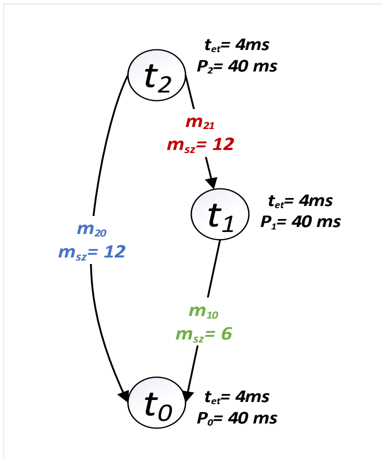


FIGURE 7.1: A simple application graph consists of 3 tasks and 3 messages.

The Gantt charts in Figure 7.2 depict the TT message transmission schedules of TMS, MLTS and MMTS algorithms, respectively. TT messages as shown in Table 7.2 are with different transmission times. The periods of tasks are the same and equal to 40 ms. According to Eq. 7.9 each parent task sends its periodic TT message to its child task every 4 ms with the parent task execution time $rep_{parent(t)}$ = 1, which means that the parent task sends its message after the first execution. If $rep_{parent(t)}$ = 2 then the parent task sends its message after the second execution, which is after 40 + 4 = 44 ms. TT messages are fragmented according to slt_{d} (slt_{d} is set to 2 ms). For example, m_{comm} of message m_{20} is 6 ms and it will be fragmented into 6/2 = 3 fragments.

TABLE 7.2: Transmitted TT messages between periodic tasks.

Message (m)	m_{comm}	slt_{d}	F_m
m_{10}	3	2	2
m_{21}	6	2	3
m_{20}	6	2	3

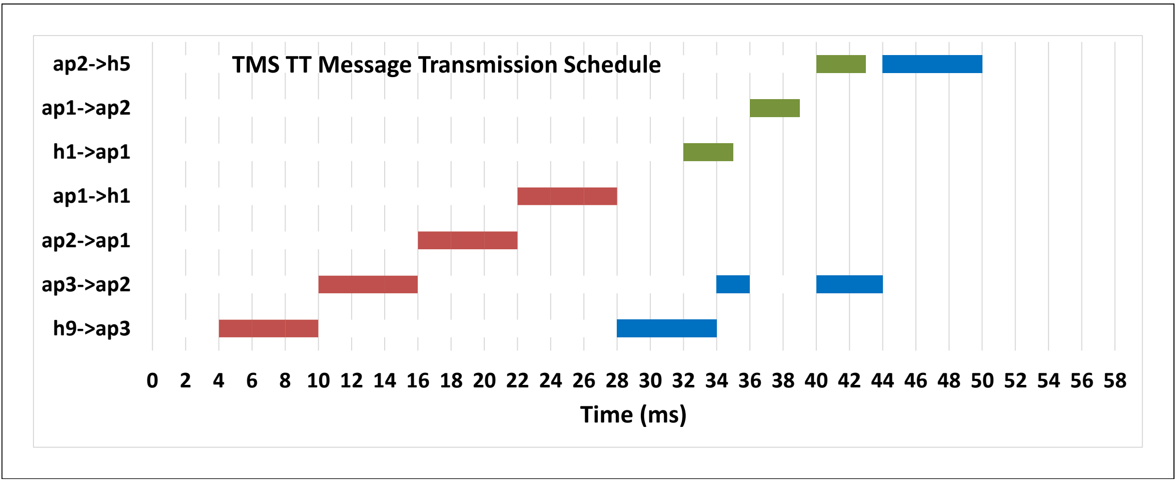
Figure 7.2a shows scheduling the TT messages using the TMS algorithm. As shown, two TT messages are sent from host $h9$ and one TT message is sent from $h1$. The route of each message can be found by keeping track of each coloured message from the sender host until reaching the receiving host. These routes are selected according to the minimum m_{e2eD} according to the TMS procedure. m_{10} and m_{21} messages are directly scheduled without delaying their fragments because either no message uses the time slot (slot is empty) or the induced interference by other messages, which use other paths at the same time slot, does not violate β (the computed SINR value is more than β). For instance, m_{21} uses all its paths in the range of time slots $4 - 28$ without delaying its fragments because no other messages use their paths in the same time slots. m_{10} uses also path ($ap2$ -> $h5$) in time slots $40 - 43$ without delaying its fragments because the interference caused by using m_{20} on its path ($ap3$ -> $ap2$) simultaneously in the same time slot does not degrade the SINR value to be less than β in the Inequality 7.5.

The induced interference by other messages, which use other paths simultaneously may make the sending infeasible. For example, the first fragment of message m_{20} starts using the path ($ap3$ -> $ap2$) at time slot $34 - 36$, and the second fragment is delayed until time slots $40 - 44$. The reason is that the induced interference of sending message m_{10} on the path

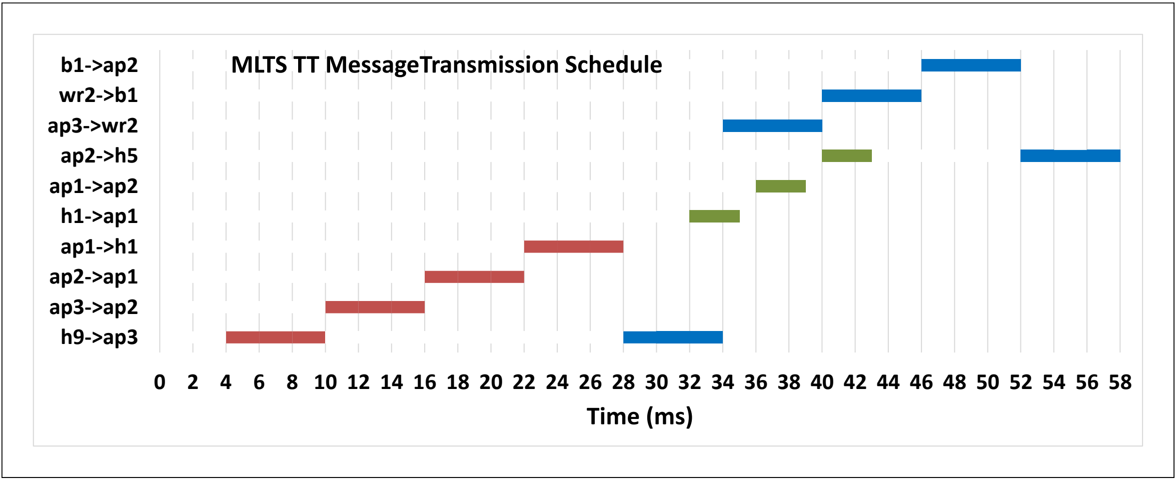
($ap1$ -> $ap2$) in timeslots $36-39$ makes the path ($ap3$ –> $ap2$) infeasible for sending m_{20} in timeslots $36-40$. It is worth mentioning that a message has to use a path at the beginning of the next time slot even if the message can be scheduled before the end of the previous time slot. In this example, message m_{10} uses the path ($h1$ -> $ap1$) in time slots $32-35$, whereas it starts using the next path ($ap1$ -> $ap2$) at the beginning of the next time slot $36-39$. It is shown that all messages require 50 ms to be scheduled. After 50 ms, all incoming messages are scheduled and received by the child task (i.e. t_0). Thereby, the total makespan is 50 + 4 = **54 ms**, where 4 ms is the execution time of task t_0.

The MLTS algorithm uses the same time slot message scheduling model as TMS, but as we see in Figure 7.2b, m_{20} is scheduled on a different route which is $\{h9, ap3, wr2, b1, ap2, h5\}$ rather than the route $\{h9, ap3, ap2, h5\}$ which is used by the TMS algorithm. The reason is that m_{21} is scheduled on the route $\{h9, ap3, ap2, ap1, h1\}$, thereby the path ($ap3$ -> $ap2$) has a traffic load from message m_{21}. Therefore, MLTS changes the route in which m_{20} is scheduled to another route that has less traffic load, which leads to selecting a longer route that consumes, in turn, more time and energy than the route from the TMS algorithm. In the end, the makespan as a result of scheduling all messages and executing task t_0 equals 58 + 4 = **62 ms**. The increase of the global makespan may lead to missing a deadlines of some scheduled tasks, thereby, the global task scheduling solution by MLTS may be considered as an infeasible solution.

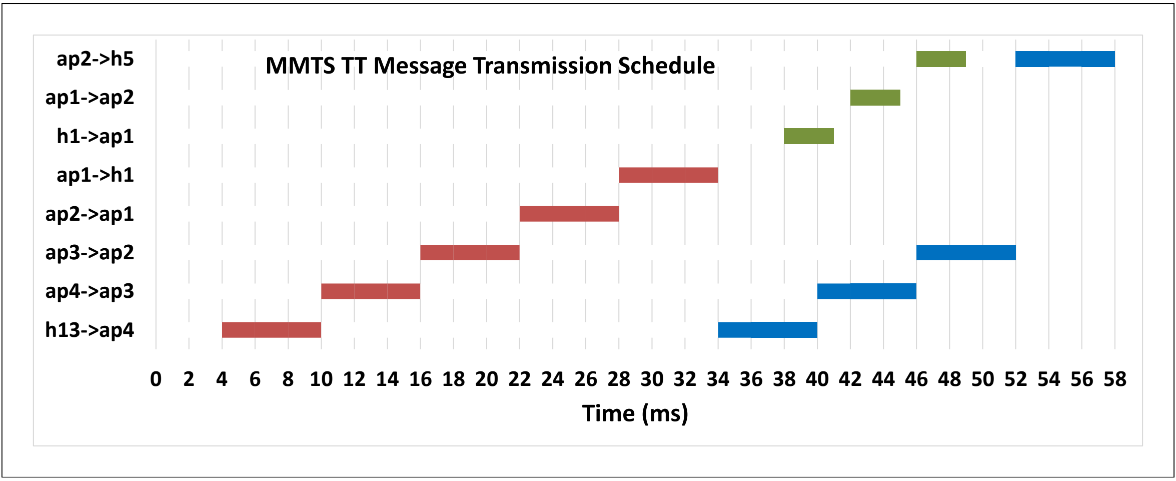
TMS and MLTS aim to select nearby hosts for assigning tasks. For instance, host $h9$ is assigned to execute task t_2. Similarly, $h1$ and $h5$ are assigned to execute t_1 and t_0, respectively. Therefore, the proposed solutions by TMS and MLTS select $h1$, $h5$ and $h9$ which are close to each other (see Figure 4.4). On the contrary, the prepared solutions in MMTS, which selects the hosts that have the minimum ready time to execute tasks, may make the selected hosts relatively distant, depending on the size of the system model. This is evident from Figure 7.2c, where hosts $h13$, $h1$ and $h5$ are assigned to execute tasks t_2, t_1 and t_0, respectively. For instance, the sender $h13$ sends message m_{21} through a longer route to the receiver $h1$. In this simple example, the global makespan is **62 ms**. However, with the increase in the number of tasks and the transmitted messages, the proposed solutions by MMTS are more complex and require more time than TMS and MLTS to implement all tasks, which adversely leads to missing the task deadlines in real-time applications.



(A) TT message transmission schedule of TMS.



(B) TT message transmission schedule of MLTS.



(C) TT message transmission schedule of MMTS.

FIGURE 7.2: TT message scheduling with dedicated time slots for TMS, MLTS and MMTS algorithms.

7.5 Evaluation of TMS Algorithm

This section illustrates the experimental setup with several sets of simulation experiments used for evaluating the proposed algorithm in a *WirelessTSN* framework. The experiments run on a ThinkPad laptop (Intel i5 CPU) with 24GB of memory. The SNAP library [218] is used to build random input system models. The simulation results are then compared using different parameters.

7.5.1 Simulation Setup

To evaluate the TMS algorithm, MLTS and MMTS algorithms are implemented using the same procedures as TMS (i.e. implementation of the time slot message scheduling model considering the mentioned constraints in Section 7.2). The evaluation procedure is done by using three sets of simulation experiments. The first experiment set shows the makespan, consumed energy and failure rate obtained by using two types of network topologies (i.e. the grid and ring topologies in Figures 4.4, 4.5, respectively). The second experiment set shows the miss deadline ratio as a result of increasing the task deadlines. The last experiment is used to represent the impact of the interference on the makespan, consumed energy and failure rate obtained by changing the values of the path loss and the communication threshold in the mentioned grid topology. Table 7.3 summarizes the simulation setup of the simulation experiments.

TABLE 7.3: The simulation setup of the simulation experiments to evaluate TMS algorithm.

Parameter	**Value**
Simulation area.	1000 x 1000 meters.
Distance between hosts and their access points (i.e. gateways).	Randomly chosen in a range of [1, 30] meter.
Execution time (ms) and energy cost (joule) of routing a message are subject to be changed according to each wireless router.	Determined from a range of [20, 25].
Execution time (ms) and energy cost (joule) of executing a task are subject to be changed according to each wireless host.	Determined from a range of [200, 250].
The full energy of any router or host.	100,000 joule.
Wireless bandwidth.	2 units per ms.
Time slot duration (i.e. slt_d).	2 ms.
Ambient noise power (i.e. N).	10^{-10} mW.
Transmission power (i.e. P_{tr}).	100 mW.
Number of simulation tests.	20 times, then the result is averaged.

The application graphs are generated as random forest fire, directed graphs [218]. The set of hosts that are used to process the computational tasks are selected randomly from a set of available hosts in the system.

7.5.2 Experimental Results

This section shows a comparison of the TMS algorithm with MLTS and MMTS algorithms by applying three sets of simulation experiments.

Network Topology

We consider our hybrid modelling approach from Chapter 4 to evaluate the proposed algorithm. Each network topology has ten TSN-enabled routers (access points and wireless routers) and each access point is connected with four wireless TSN-enabled hosts. Each router in the grid topology is connected with two or three wireless routers, while the routers in the ring topology are connected to form a circular loop. The grid topology has more routes and connections, while the ring topology is more popular in several industrial fields.

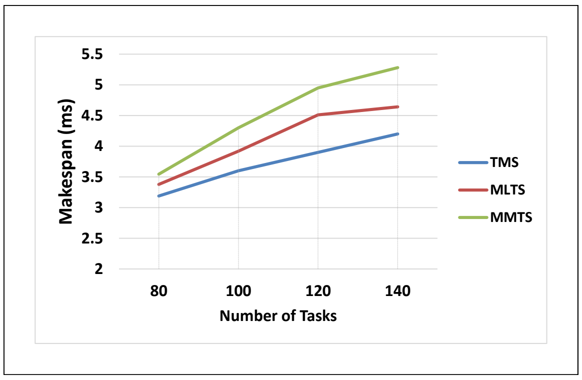
To show the impact of using different network structures, Figures 7.3 and 7.4 illustrate the average makespan, average consumed energy and average failure rate for 20 different cases in TMS, MLTS and MMTS. Figure 7.3 is applied in a grid topology, whereas Figure 7.4 is applied in a ring topology. The comparison of the algorithms is made by increasing the number of tasks to be scheduled. As can be seen in the table below, as the number of tasks increases, so do the number of messages sent.

Number of tasks	Number of accompanying messages
80	200
100	250
120	300
140	350

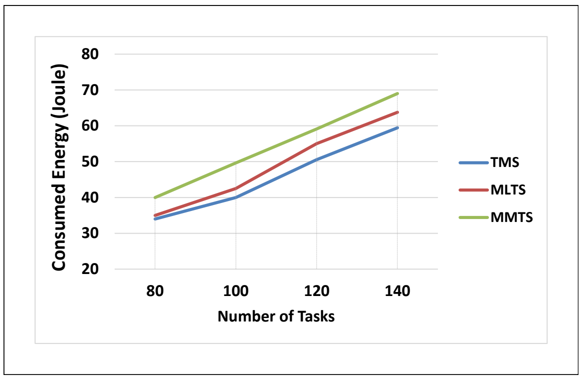
TMS improves the makespan, consumed energy and failure rate compared to MLTS and MMTS. The reason is that TMS achieves the minimum makespan because of considering the minimum end-to-end delays of message transmissions, which in turn, leads to better utilization of the channel bandwidth. The routes chosen by TMS are mostly the shortest routes to send or forward messages, which reduces the energy required to transmit these messages. The involvement of fewer routers also reduces the failure rate of the proposed scheduling solution. On the other hand, MLTS depends on the principle of adopting the least congested routes, which may lead to positive results in small networks but when increasing the complexity of the networks, it becomes difficult to choose the best route to send messages especially when the task deadline is not allowed to be missed. Moreover, less congested routes are mostly longer routes and require more energy and time, which negatively affects the failure rate in applying the MLTS algorithm.

MMTS as we mentioned earlier selects off-line solutions, this often affects the makespan, consumed energy and failure rate, especially when the complexity of the networks increases. In contrast, the previous two algorithms (TMS and MLTS) assign a host for each task gradually, so that after scheduling a task, assigning a host for the next task considers the deadline for that task until all tasks are completed.

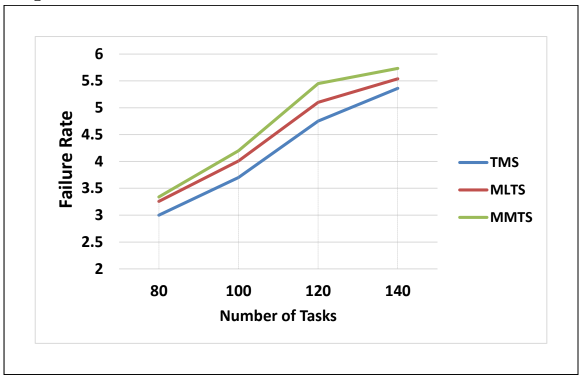
It is clear that the makespan, consumed energy and failure rate increase when the ring structure is used compared to the grid structure for all algorithms. The reason is that the ring structure has fewer routes, therefore, the best route is chosen with fewer routes available. In addition, using the same route by several messages leads to higher utilization of the network resources and imposes in the physical interference model the need to stretch the time when scheduling messages on a particular route. It is worth noting that the size of the application graph does not affect the makespan significantly if there are enough resources in the architecture graph.



(A) Impact of applying the compared algorithms on the makespan.



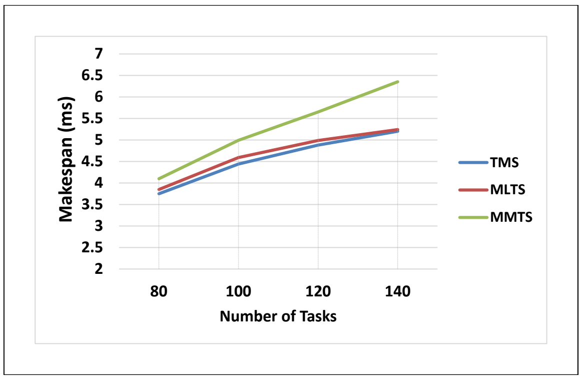
(B) Impact of applying the compared algorithms on the consumed energy.



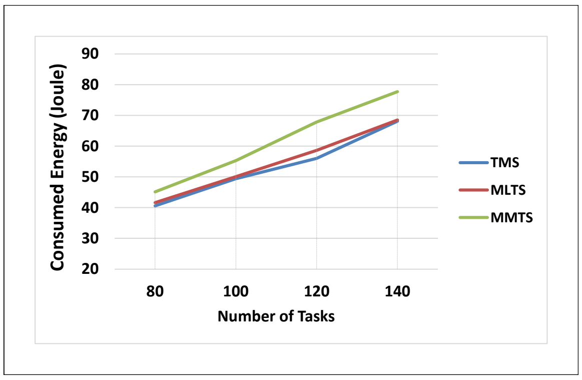
(C) Impact of applying the compared algorithms on the failure rate.

FIGURE 7.3: Impact of applying TMS, MLTS and MMTS on the makespan, consumed energy and failure rate when increasing the number of tasks in the **grid** topology.

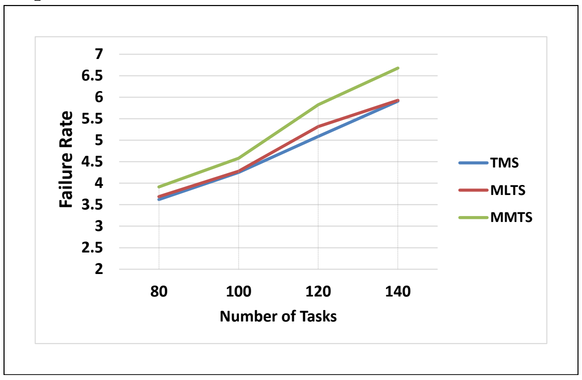
Tables 7.4 and 7.5 illustrate the CPU execution time required for all compared algorithms in the grid and ring topologies, respectively. The time it takes to implement all algorithms increases as the number of tasks increases since more tasks require more CPU time to be scheduled. Compared to MMTS, TMS and MLTS take longer to reach the final solution. The reason is that MMTS prepares a solution that schedules all tasks before starting the implementation, and this does not require much time compared to TMS and MLTS algorithms, which are looking into the possibility of scheduling each task gradually in all available routes until finding the final solution. Checking the traffic load on all available routes and then choosing the one that is the least congested route adds additional effort to MLTS. Therefore, MLTS consumes a notable CPU time, especially when increasing the number of tasks that in turn leads to an increase in the congestion on the routes. Consequently, the more congestion on the routes, the more CPU time MLTS takes to determine which is the least congested one. While TMS computes the time for sending messages on the available routes, choosing the route to send a message depends on the fastest route from among all available routes.



(A) Impact of applying the compared algorithms on the makespan.



(B) Impact of applying the compared algorithms on the consumed energy.



(C) Impact of applying the compared algorithms on the failure rate.

FIGURE 7.4: Impact of applying TMS, MLTS and MMTS on the makespan, consumed energy and failure rate when increasing the number of tasks in the **ring** topology.

TABLE 7.4: The execution time (in seconds) for TMS, MLTS and MMTS in a **grid** topology when increasing number of tasks/graph.

	Increase the number of tasks/graph.			
Algorithm	T **= 80**	T **= 100**	T **= 120**	T **= 140**
TMS	3.5	6.3	8	11
MLTS	36	51	89	111
MMTS	1.5	2.1	2.6	4

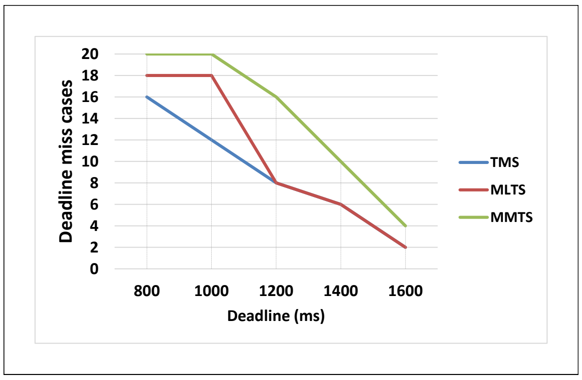
TABLE 7.5: The execution time (in seconds) for TMS, MLTS and MMTS in a **ring** topology when increasing number of tasks/graph.

	Increase the number of tasks/graph.			
Algorithm	T **= 80**	T **= 100**	T **= 120**	T **= 140**
TMS	5	8.8	11	13
MLTS	14	23	27	38
MMTS	1.6	2.3	3.6	5

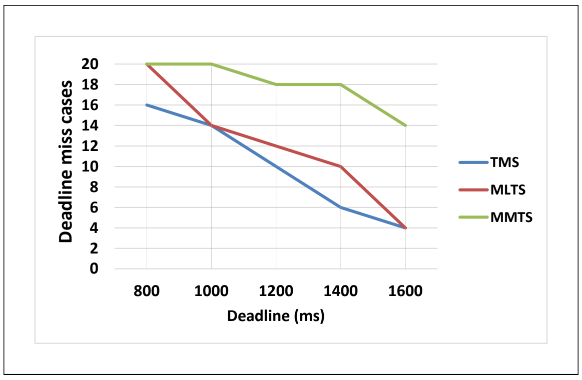
Deadline miss ratio

Figure 7.5 shows the deadline miss ratio of 20 test cases for TMS, MLTS and MMTS in the grid and ring topologies. The miss ratio is computed by increasing the task deadlines in the range of 800-1600 ms. The deadline miss ratio of TMS is less than that of MLTS and MMTS by 15.3% and 37.1% in the grid topology, respectively. Similarly, the deadline miss ratio of TMS is less than that of MLTS and MMTS by 16.6% and 44.4% in the ring topology, respectively.

TMS outperforms MLTS in the deadline miss ratio. The reason is that TMS mainly depends on selecting the fastest routes to transmit messages and tasks are scheduled accordingly. MLTS adopts the least congested routes, which often require more time to send the messages and cause in turn to increase the deadline miss ratio. On the other hand, MMTS schedules the tasks to hosts that have the minimum ready time. However, the cost of delivering messages between hosts is not taken into consideration, which clearly affects adversely on the deadline miss ratio. The grid topology shows better results than the ring topology for all compared algorithms, the reason is that the ring topology uses a limited number of routes and the high utilization of these routes leads to increased transmission time and violates the deadline constraints.



(A) Deadline miss cases of the compared algorithms in the **grid** topology when increasing of task deadlines.



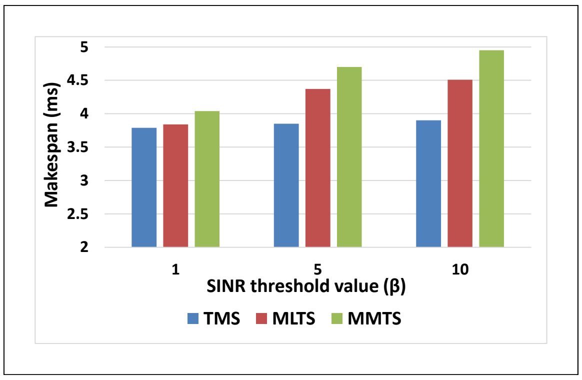
(B) Deadline miss cases of the compared algorithms in the **ring** topology when increasing of task deadlines.

FIGURE 7.5: Deadline miss cases of TMS, MLTS and MMTS in the grid and ring topologies when increasing of task deadlines.

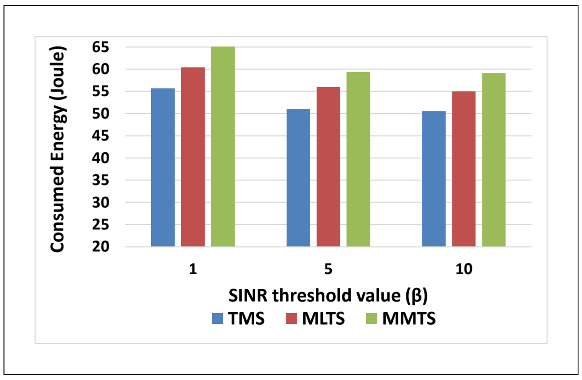
Wireless Interference Parameters (α, β)

To evaluate TMS, MLTS and MMTS when considering the interference metric, Figures 7.6 and 7.7 compare the average makespan, consumed energy and failure rate for the compared algorithms. The grid graph and an application graph of 120 tasks and 300 messages are inserted as inputs. Firstly, Figure 7.6 shows that TMS reduces the makespan, consumed energy and failure rate by 8.93%, 8.09% and 6.41% compared to MLTS when the path loss (i.e. α) = 4 and the communication threshold (i.e. β) increases to be 1, 5 and 10. Similarly, TMS reduces the makespan, consumed energy and failure rate by 15.13%, 14.55% and 12.53% compared to MMTS. Secondly, Figure 7.7 shows that TMS reduces the makespan, consumed energy and failure rate by 10.9%, 10.42% and 6.77% compared to MLTS when β = 10 and α increases to be 4, 7 and 10. Similarly, TMS reduces the makespan, consumed energy and failure rate by 19.08%, 14.05% and 11.9% compared to MMTS.

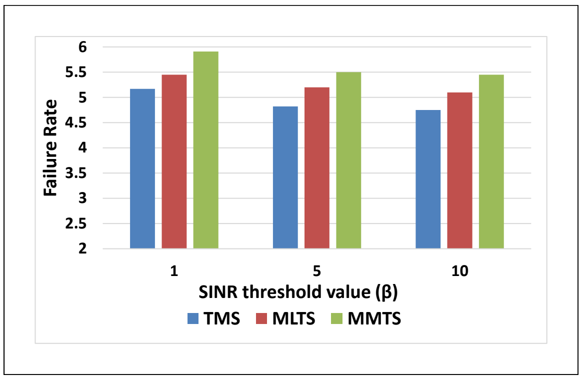
Figure 7.6 shows that increasing the value of β makes it more difficult for the computed SINR to satisfy the physical interference condition, which in turn increases the number of time slots that are required to send all messages, thereby, the average makespan for all algorithms will be increased. On the other side, Figure 7.7 shows that increasing the value of α reduces the average makespan because higher α induces less interference on every wireless path. On the contrary, we notice that with the increase of β value, the consumed energy and the failure rate will be decreased. The reason is that when increasing the value of β, the signal strength of the sender node should be higher than any potential signal interference. Therefore, the signal interference when increasing the value of the interference condition has no significant effect on the failure ratio or the possibility of retransmitting the failed messages that require extra energy. Figure 7.7 shows also that the path loss value, the consumed energy and the failure rate will be slightly increased.



(A) Makespan of TMS, MLTS and MMTS when increasing β value.

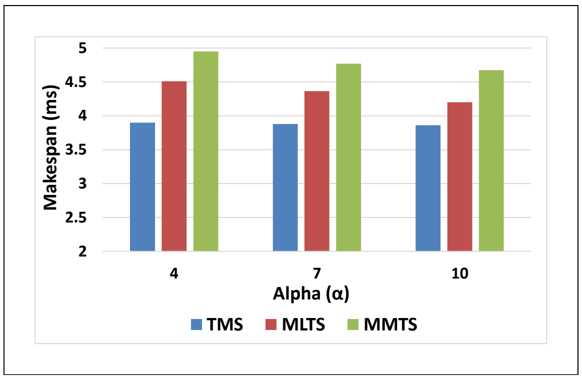


(B) Consumed energy of TMS, MLTS and MMTS when increasing β value.

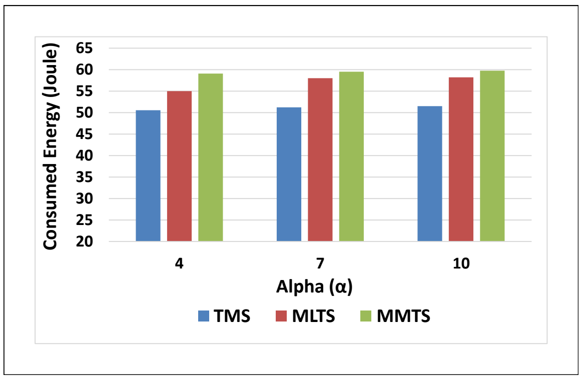


(C) Failure rate of TMS, MLTS and MMTS when increasing β value.

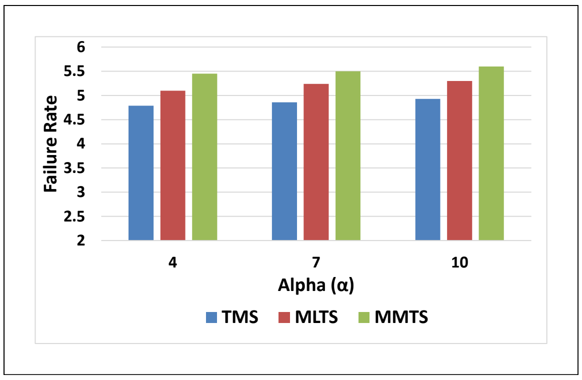
FIGURE 7.6: Makespan, consumed energy and failure rate of TMS, MLTS and MMTS in the **grid** topology when increasing β value, $\alpha = 4$.



(A) Makespan of TMS, MLTS and MMTS when increasing α value.



(B) Consumed energy of TMS, MLTS and MMTS when increasing α value.



(C) Failure rate of TMS, MLTS and MMTS when increasing α value.

FIGURE 7.7: Makespan, consumed energy and failure rate of TMS, MLTS and MMTS in the **grid** topology when increasing α value, $\beta = 10$.

Chapter 8

Reliable Task and Message Scheduling Algorithms for Hybrid TSN Systems

8.1 Introduction

Our proposed TMS algorithm in the previous chapter aims to solve the task and message scheduling problem with the least possible makespan, ensuring that all tasks meet their deadlines. TMS takes into account constraints such as the signal interference, but it considers the system as fault-free which is unrealistic for most wireless technologies. Therefore, the task scheduling to hosts, scheduling the transmitted messages, and simultaneously balancing the task load in the context of unreliable wireless networks are urgent and significant issues.

The task and message scheduling problem in $WirelessTSN$ networks involves additional challenges compared to traditional distributed systems. Distribution of sensing tasks among hosts should take into consideration the consumed energy to prolong the network lifetime, besides guaranteeing that all tasks finished before their deadlines. The reason is that draining of the remaining energy leads some nodes to become overloaded, and then tasks fail to be executed. The problem of task and message scheduling becomes more difficult when these tasks must satisfy precedence requirements [28], meaning that a task does not start its execution until all incoming messages from parent tasks have been received.

Moreover, $WirelessTSN$ nodes or paths are subject to failures. In order to prevent a failure from degrading the network performance, the reliability in the presence of faults becomes a necessity for the continuity and functionality of $WirelessTSN$ networks. Reliability becomes even more important when adopting wireless technologies to be used in hard real-time systems (e.g. the battlefield surveillance) where the transmitted data is critical and it should arrive with stringent deadlines [4].

Time-triggered wireless networks are gaining importance due to the increasing demand for reliable and deterministic wireless infrastructures in real-time and safety-critical systems. All contributing network nodes should firstly share a precise global time base. Therefore, we extended the IEEE 802.1AS protocol in prior work [27] to provide a precise time synchronization process in a TSN hybrid framework.

Recently, some state-of-the-art algorithms have adopted means of task scheduling in wireless environments to reduce the task completion time, the energy consumption or to establish a reliable task scheduling framework. This chapter offers two algorithms that seek to take into account all of the restrictions listed in Section 7.2 for reliably scheduling tasks and messages. The first reliable algorithm is called Reliable Task and Message Scheduling

(R-TMS) algorithm [29], which optimizes the system reliability when scheduling periodic tasks and scheduling their messages. Thus, R-TMS provides the following contributions:

- Like TMS, R-TMS schedules tasks gradually until all tasks are completed. The difference is that TMS selects the routes to send messages according to the minimum latencies. On the contrary, R-TMS establishes a cost function for optimization of the task scheduling, which includes the message latency, energy consumption and failure rate for selecting hosts on which the tasks can run.

- Employment of FRER [16] in *WirelessTSN* networks. In this approach, every message is replicated and sent by the sender over more than one redundant route. The receiver accepts one copy and eliminates the other one. Therefore, FRER avoids the message loss as a result of failures.

- A physical interference model is employed as part of the message scheduling model to prevent the influence of an interference on message transmissions and achieve timeliness [28]. The interference model in R-TMS is used more widely to accommodate duplicate messages, which makes message scheduling more complicated and requires more time.

- A reliability model is also incorporated in R-TMS to determine the overall system's reliability. Firstly, the reliability of each transmitted message between tasks [219] is computed by establishing a Reliability Block Diagram (RBD). Second, conditional precedence restrictions are used for the sequential tasks, which compute each task's reliability using the reliability of the incoming communication messages as an input. The conditional restriction specifies whether the incoming message from a sending task is essential to execute the receiving task or is substitutable with other messages from other senders to start the execution of that task. The sink task, which is a leaf task without forwarding messages, is ultimately responsible for the overall system's reliability.

The solution of R-TMS is a multi-objective optimization, which aims to reduce the task completion time, save energy and minimize the failure rate. Nonetheless, its major disadvantage is that the computational complexity is growing exponentially when increasing the size of the system models. In contrast, stochastic heuristic optimization algorithms e.g., GA [89], [90], PSO [91], [92] can find a solution with higher efficiency. PSO typically outperforms other evolutionary-based optimization algorithms in terms of the quality of the solutions and success rate while offering low processing time and fast convergence [220]. In addition, the simplicity, ease to use and low computational complexity of PSO make it suitable for resource-limited wireless environments [221]. Since the task scheduling problem for *WirelessTSN* networks is a discrete search space problem, where the goal is to schedule a set of available tasks to certain hosts that will satisfy specific constraints, the discrete version of PSO (i.e. DPSO) is employed in this thesis.

Based on DPSO, we introduce a second reliable algorithm called Optimized Reliable Task and Message Scheduling algorithm (OR-TMS) [30], which also uses FRER and the physical interference model. OR-TMS is optimized by the following further features:

- The DPSO algorithm [222] is applied as a multi-objective optimization. The solution is optimized by considering properties including the global makespan, total energy consumption and total failure rates.

- OR-TMS balances the network load by using a load balance mechanism during the execution of DPSO.

- We conduct extensive tests to compare OR-TMS and R-TMS with other algorithms. The simulation results show the effectiveness and the feasibility of OR-TMS.
- A metascheduler based on OR-TMS is presented to provide a predictable adaptation mechanism, ensuring that in the event of run-time events, transitions to a previously verified schedule are made. Therefore, the proposed metascheduler pre-computes schedules at design time and forms a Multi-Schedule Graph (MSG). It uses a reconvergence mechanism for repeated schedules to address the state-space explosion problem in the MSG.

To the best of our knowledge, no work has been done to design a task and message scheduling algorithm that works to find a feasible solution, which considers the mentioned contributions as in R-TMS and OR-TMS algorithms.

The rest of this chapter is organized as follows. The next section presents the problem formulation of R-TMS. Section 8.3 presents time slot message scheduling over redundant routes. The reliability model of the Wireless TSN system is discussed in Section 8.4. The R-TMS algorithm is demonstrated in Section 8.5. The experimental results of using R-TMS are shown in Section 8.6. Section 8.7 presents the problem formulation of OR-TMS. Section 8.8 discusses the DPSO algorithm. The task and message scheduling process in OR-TMS is explained in Section 8.9. Section 8.10 shows a trace example of using DPSO-based OR-TMS. Section 8.11 discusses the experiments and results as a result of OR-TMS implementation. Finally, an OR-TMS based metascheduling technique is discussed and evaluated in Section 8.12.

8.2 Problem Formulation of R-TMS

We consider the hybrid modelling approach mentioned in Chapter 4 as an input to R-TMS. The proposed algorithm aims to minimize the completion time, energy consumption, and failure rate of each task until all tasks are done while gradually scheduling T tasks to available hosts. The algorithm also aims to improve the global system reliability. In other words, the algorithm ensures that the tasks do not fail due to a potential failure of components or links. To do so, the messages of the sender (i.e. parent) tasks are transmitted through redundant disjoint routes. In real-time applications, there exist dependant tasks that should be reliably executed under constraints of deadline, energy, bandwidth and processing capability of wireless networks. Therefore, it is required to optimize the selection of the hosts to execute the consecutive tasks.

To explain the task and message scheduling process, we introduce the following points. The notations used for R-TMS are shown in Table 8.1:

- The R-TMS algorithm considers the constraints of the TMS algorithm as mentioned in Section 7.2.
- We consider $T = \{t_1, t_2, t_3, \ldots, t_n\}$ as a sorted set of n unscheduled tasks and $H = \{h1, h2, h3, \ldots, hm\}$ as a changeable set of m available hosts on which each task (i.e. $t \in T$) can run. For instance, t_1 can be run on a host that belongs to a set (e.g. $H_{t_1} = \{h1, h3, h6\}$), whereas t_2 can be run on a host that belongs to a different set (e.g. $H_{t_2} = \{h1, h4, h8\}$). R-TMS aims to schedule each task $t \in T$ to the best available host that belongs to its available host set.
- FRER is used to send x copies ($m^{c1}, m^{c2}, \ldots, m^{cx}$) of message m through disjoint redundant routes $RR(m)_h$ from all senders to a tested available host h on which a task t can be run.

- Computing the energy consumption and the failure rate of routing message m over the disjoint redundant routes:

  The energy consumption of routing the message m (i.e. en_m) is the sum of the consumed energy on each router $r \in RR(m)_h$, as shown in Eq. 8.1.

$$en_m = \sum_{r=1}^{RR(m)_h} en_r \qquad \forall r \in RR(m)_h \tag{8.1}$$

  The failure rate of routing the message m (i.e. F_m) is shown in Eq. 8.2:

$$F_m = \sum_{r=1}^{RR(m)_h} \eta_r \times et_r \qquad \forall r \in RR(m)_h \tag{8.2}$$

  Where, en_r and et_r are the consumed energy and the execution time on every router $r \in RR(m)_h$, respectively. The time slot message scheduling, which will be discussed later, shows how to compute en_r and et_r values according to the message interference analysis. It is important to note that the execution time and energy cost of message routing might vary based on the route the message takes, they are denoted as $et[R]$ and $en[R]$ parameters in Chapter 4. η_r denotes the failure rate of r, which denotes the number of failures of router r over a specified time (e.g. per 6 months) [223].

- When the time slot message scheduling has been completed, an RBD diagram is applied to compute the reliability of message m (i.e. Rl_m). Section 8.4 shows more details about computing the reliability of the message that depends on the reliability of the TSN-enabled nodes and the reliability of network paths that the message passes through.

- Finding the time of arrival of all incoming messages at task t and computing the failure rate and the consumed energy of executing task t at host $h \in H$ is done as follows:

  According to the definition of the message arrival time, the arrival time denotes the last instant of time when all incoming messages have arrived at the receiver task t at host h. The arrival value at host h is defined as:

$$arrival = max(m_{\text{arrival}}) \qquad \forall m \in M_t \tag{8.3}$$

  The consumed energy and the failure rate after arrival of all messages and executing task t at host h are computed as follows:

$$en_t = \sum_{m=1}^{M_t} en_m + en_{h.t} \qquad \forall m \in M_t \tag{8.4}$$

$$F_t = \sum_{m=1}^{M_t} F_m + F_{h.t} \qquad \forall m \in M_t \tag{8.5}$$

  Where $en_{h.t}$ and $F_{h.t}$ denote the consumed energy and the failure rate of executing task t at host h, respectively.

- A conditional precedence restriction is then applied to compute the task reliability Rl_t. Section 8.4 shows more details about computing the reliability of the task that depends on the reliability of incoming messages and their conditional cases (i.e. essential or substitutable).

TABLE 8.1: Notations used in Chapter 8

Symbol	The description of the symbol	Domain
$RR(m)_h$	The disjoint redundant routes for message m from a sender to host h	$h \in G_{Arc}$
$rt(m)$	A route in $RR(m)_h$	$m \in G_{App}$
$en_{r.m}$	The consumed energy of routing message m on router r	$r \in G_{Arc}$
$et_{r.m}$	The routing time of message m on router r	$m \in G_{App}$
$en_{h.t}$	The consumed energy of executing task t on host h	$h \in G_{Arc}$
$et_{h.t}$	The execution time of task t on host h	
$F_{h.t}$	The failure rate of executing task t on host h	$t \in G_{App}$
en_r	The consumed energy on router r	$r \in G_{Arc}$
et_r	The execution time on router r.	
en_m	The consumed energy of routing message m.	$m \in G_{App}$
F_m	The failure rate of routing message m.	
en_t	The consumed energy of executing task t on available hosts after arrival of all messages.	$t \in G_{App}$
F_t	The failure rate of executing task t on available hosts after arrival of all messages.	
$EN = (en_{r.m})_{mxr}$	The consumed energy matrix.	
$ET = (et_{r.m})_{mxr}$	The execution time matrix.	

- According to the above descriptions, the task scheduling cost *task_cost* of scheduling task t at host h is formulated as shown in Eq. 8.6:

$$task\_cost = \theta * arrival + \gamma * en_{\mathrm{t}} + \delta * F_{\mathrm{t}} \tag{8.6}$$

  Where θ, γ and δ are weighting parameters, *task_cost* value is computed for each $h \in H$, then task t is finally scheduled to the host h that results in the minimum *task_cost* value. The process is repeated for every computational task until scheduling all tasks in T.

- Finally, the reliability of the system is considered as the reliability of the sink task (i.e. leaf task) where no forwarding messages will be sent in the wireless TSN-enabled system model.

8.3 Message Scheduling Over Redundant Routes

To determine the cost of scheduling a task to a host $h \in H$, a message scheduler is applied for scheduling all incoming messages over redundant routes. Therefore, FRER should be first applied to determine how to extract these disjoint redundant routes.

8.3.1 Frame Replication and Elimination for Reliability (FRER)

Many TT wireless task scheduling techniques assume that the network structure is fault while the messages are being exchanged. In practice, the wireless networks may be subject to node or link failures. Since delays need to be bounded, the redundancy becomes a necessity to mitigate the faulty behaviours. Recovery from faulty behaviour is crucial, especially in safety-critical systems where failure is more likely to cause severe, irreparable harm and significant financial losses, as well as put people's lives in peril. Therefore, the permanent failures can be alleviated by using FRER spatial redundancy [16], which sends replicas of every transmitted message over redundant routes and eliminates redundant replicas at the receiver.

FRER features need to be considered during the process of the message scheduling because the redundancy increases the search space and the schedule that meets the restrictions imposed on $WirelessTSN$ becomes more complex. Therefore, the replicas of the communication messages are forwarded over the most disjoint routes, where the most disjoint routes are defined as the shortest routes that do not share common wireless relays between them. Thus, considering the shortest disjoint routes increases the likelihood of messages arriving within the target time and it reduces the research space.

FRER in the proposed algorithm runs first the Shortest Path First (SPF) routing algorithm to find routes starting from the shortest route. FRER prunes the edges of the shortest route from the graph to find the available paths on the remaining graph. If pruning leaves leads to no paths (i.e. no disjoint routes from the sender to the receiver), then FRER excludes this route and repeats the pruning process on the next shortest route. When pruning of a route provides a certain number of K disjoint routes, the proposed algorithm uses them as paths to send copies of a message.

To illustrate how to extract the most disjoint routes, the application graph in Figure 8.1 shows tasks: t_3, t_2 and t_1 are already run on hosts $h1$, $h12$ and $h8$, respectively. t_0 can run on one of the hosts $\{h9, h16, h20\}$. The grid architecture graph in Figure 4.4 shows different redundant routes: route1 = $\{h1, ap1, ap2, ap3, h9\}$, route2 = $\{h1, ap1, wr1, b1, wr2, ap3, h9\}$ and route3 = $\{h1, ap1, wr1, b1, ap2, ap3, h9\}$ that can be used to transmit a message m_3 from task t_3 that is scheduled at host $h1$ to t_0 at a potential host (e.g. $h9$). The wireless TSN nodes indicated in blue represent either a host or a gateway through which the message passes while it is being sent from a sender to the recipient, so they are excluded when choosing the most disjoint routes. Route1 and route2 are considered as disjoint routes (i.e. $RR(m_3)_{h1->h9}$) between the sender host $h1$ and the receiver $h9$ because they do not have common wireless intermediate nodes. Route3 is excluded from the disjoint route set because it has a common wireless intermediate node (i.e. $ap2$) with route1 and two intermediate nodes (i.e. $wr1$ and $b1$) with route2. As soon as the receiver $h9$ receives message replicas, it forwards only one copy and eliminates the others. Since FRER adds extra delay and requires more network resources (i.e. bandwidth), we make the assumption that a message will arrive at its destination at the time from the longest route in the disjoint route set in order to prevent fluctuation in the arrival latency.

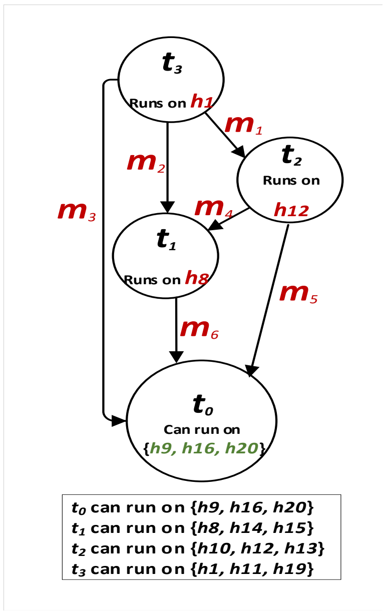

t_3
Runs on h1
m_1
m_2
m_3
t_2
Runs on h12
m_4
t_1
Runs on h8
m_5
m_6
t_0
Can run on {h9, h16, h20}
t_0 can run on {h9, h16, h20}
t_1 can run on {h8, h14, h15}
t_2 can run on {h10, h12, h13}
t_3 can run on {h1, h11, h19}

FIGURE 8.1: A application graph showing different available hosts on which each task can run.

8.3.2 Time Slot Message Scheduling Over Disjoint Redundant Routes

Time slot message scheduling is an extension to the scheduling introduced in TMS. As described, message scheduling means assigning each message to a set of time slots in which it will be transmitted using the dedicated paths. The message scheduling adopts the physical interference model and computes en_r and et_r of each $r \in RR(m)_h$ used by a transmitted message. The implemented model aims to transmit multiple messages simultaneously over redundant routes using conflict-free time slots in order to increase the likelihood that the data arrives successfully and avoid the influence of the induced interference.

8.3.3 Example

In Figure 8.1, when task t_3 at sender host $h1$ starts to transmit message m_3 to task t_0, we assume that the available host set on which t_0 can run is $H_{t_0} = \{h9, h16, h20\}$. Firstly, R-TMS starts to check the task cost (*task_cost*) at every $h \in H_{t_0}$ (e.g. $h9$). Thus, route1 = $\{h1, ap1, ap2, ap3, h9\}$ and route2 = $\{h1, ap1, wr1, b1, wr2, ap3, h9\} \in RR(m_3)_{h1->h9}$ are extracted by FRER, where $\eta = 0.01$ for all network nodes. Table 8.2 displays the interference analysis on paths $l_j \notin$ route1 that have been scheduled in their time slots and used by other communication messages like m_1 and m_2.

We assume that the consumed energy $en_{r.m}$ and the routing execution time $et_{r.m}$ of m_1, m_2 and m_3 in intermediate nodes ($ap1$, $ap2$, $ap3$) $\in$ route1 are shown in the following matrixes, the units of measure of energy and time are 'joule' and 'millisecond', respectively. We assume the consumed energy $en_{h9.t_0}$ and the processing execution time $et_{h9.t_0}$ of executing task t_0 at host $h9$ are equal to 14 joule and 11 ms, respectively.

$$\mathbf{EN} = \left(\begin{array}{cccc} ap1 & ap2 & ap3 & \\ 2 & 3 & 1 & m_1 \\ 3 & 1 & 3 & m_2 \\ 3 & 1 & 2 & m_3 \end{array}\right) \mathbf{ET} = \left(\begin{array}{cccc} ap1 & ap2 & ap3 & \\ 4 & 3 & 4 & m_1 \\ 2 & 4 & 3 & m_2 \\ 1 & 4 & 4 & m_3 \end{array}\right)$$

TABLE 8.2: Time slot message scheduling and communication message interference analysis.

Time-slot1	Time-slot2	Time-slot3	Time-slot4	Time-slot5	Time-slot6	Time-slot7
m_1, m_2 messages have been scheduled on paths l_j's $\notin$ $route1$						
$(h1-ap1)_{m_1}$	$(ap1-ap2)_{m_1}$ $(h1-ap1)_{m_2}$	$(ap2-ap3)_{m_1}$ $(ap1-ap2)_{m_2}$	$(ap3-h12)_{m_1}$	$(ap2-h8)_{m_2}$		
Interference analysis to schedule message m_3 on the paths l_i's $\in$ $route1$						
		$(h1-ap1)_{m_3}$	$(ap1-ap2)_{m_3}$		$(ap2-ap3)_{m_3}$	$(ap3-h9)_{m_3}$

The physical interference model allows message m_3 to be transmitted through its path l_i = $(h1-ap1)$ *in* route1 until time-slot3. The reason is the induced interference of transmitting message m_1 in time-slot1, and the interference of transmitting messages m_1 and m_2 in time-slot2, which does not satisfy the condition in Eq. 7.5. In this case, the consumed energy at $ap1$ (i.e. en_{ap1}) as a result of receiving messages m_1 and m_2 in time-slot1 and time-slot2, respectively is added to the consumed energy due to arrival of the message m_3 in time-slot3. The energy consumption en_{ap1} is thus 2+3+3 = 8 joule, and the time consumption et_{ap1} at $ap1$ is in total 4+2+1 = 7 ms.

If the physical interference model allows message m_3 to travel through path l_i = $(ap1-ap2) \in$ route1 in time-slot4, the values of en_{ap2} and et_{ap2} will increase as a result of transmitting message m_1 through path $(ap1-ap2)$ in time-slot2 and message m_2 through path $(ap1-ap2)$ in time-slot3. Therefore, en_{ap2} becomes 3 + 1 + 1 = 5 joule, and et_{ap2} becomes 3 + 4 + 4 = 11 ms in time-slot4.

Transmitting the message m_3 through the path $(ap2-ap3)$ in time-slot5 causes the SINR at $ap3$ to be below the SINR threshold (i.e. β). Therefore, R-TMS considers time-slot5 as an infeasible slot for sending message m_3 and looks at the next slot (i.e. time-slot6).

Resulting from sending message m_1 through the path $(ap2-ap3)$ in time-slot3, en_{ap3} and et_{ap3} values increase to be equal to 1+2 = 3 joule and 4+4 = 8 ms, respectively in time-slot6. Similarly, m_3 is transmitted through path l_i = $(ap3-h9)$ in time-slot7. In the end, every transmitted message is scheduled on a feasible time slot, en_r and et_r of $r \in$ route1 are estimated.

As soon as the interference analysis is completed, the value of en_{m_3} as a result of using route1 is equal to en_{ap1} + en_{ap2} + en_{ap3} = 8+5+3 = 16 joule (using Eq. 8.1), and F_{m_3} is equal to η*(et_{ap1} + et_{ap2} + et_{ap3}) = 0.01*(7+11+8) = 0.26 (using Eq. 8.2). en_{m_3} and F_{m_3} values are then updated by using the interference analysis on the other redundant route route2 = $\{h1, ap1, wr1, b1, wr2, ap3, h9\}$. At the time of receiving all messages at $h9$, Equations 8.3, 8.4, and 8.5 are used to compute $arrival$, en_{t_0} and F_{t_0}, respectively. Where, the consumed energy of execution t_0 at host $h9$ (i.e. $en_{h9.t_0}$) is equal to 14 joule, and the failure rate (i.e. $F_{h9.t_0}$) is equal to η*$et_{h9.t_0}$ = 0.11.

Task_cost value of scheduling task t_0 at host $h9$ is computed by Eq. 8.6. The process is repeated to compute the *task_cost* of all available hosts (i.e. $h16$ and $h20$). t_0 will be scheduled to the host that results in the minimum *task_cost* value. The task and message scheduling process is applied in the same way to schedule all tasks in the system model.

8.4 Reliability Model of the Wireless TSN System

R-TMS is an extension of the TMS algorithm to maximize the reliability of *WirelessTSN* during the task and message scheduling process. Therefore, a novel reliability analysis model is applied. The reliability of real-time tasks, which depend on the reliability of communication messages that are replicated and delivered between them, is the foundation of the system's reliability. The reliability of the root tasks that have no incoming messages equals the reliability of their hosts. For example, Figure 8.1 shows that task t_3 has no incoming messages, therefore its reliability equals the reliability of host $h1$ on which it runs.

8.4.1 Reliability of Communication Message Transmission

After scheduling message m over all disjoint redundant routes, the reliability of the wireless TSN-enabled node n, where $n \in RR(m)_h$ is defined as follows:

$$Rl_n = e^{-\eta_n * et_n} \tag{8.7}$$

To determine the reliability of each incoming message Rl_m, R-TMS models the reliability of the nodes Rl_n and the reliability of the links Rl_l, which are used in $RR(m)_h$ as a series and parallel blocks B, where $b \in (N \cup L) \in B$. For instance, Figure 8.2 shows the RBD of message m_3. Each block is connected in series if the failure of that block causes a failure of the transmission of the message m_3. On the contrary, if the message is delivered successfully at the receiver $h9$ provided that at least one block out of the set of blocks operates correctly, this set of blocks is connected in parallel.

According to the following formulation, the reliability of the series blocks [219] equals the product of the reliability of each block:

$$Rl_m(series) = \prod_{b=1}^{B} Rl_{b_i} \tag{8.8}$$

The reliability of the parallel blocks [219] is equal to the complement of products of failure probabilities for all parallel blocks and it is formulated as follows:

$$Rl_m(parallel) = 1 - \prod_{b=1}^{B} (1 - Rl_{b_i}) \tag{8.9}$$

The final form of the message reliability is shown in Eq. 8.10:

$$Rl_m = Rl_m(series) * Rl_m(parallel) \tag{8.10}$$

According to the described model, the reliability of message m_3 in Figure 8.2 is equal to:

Rl_{m_3}=$Rl_{l_1}Rl_{ap1}$[1-(1-$Rl_{l_2}Rl_{ap2}Rl_{l_3}$)(1-$Rl_{l_5}Rl_{wr1}Rl_{l_6}Rl_{b1}Rl_{l_7}Rl_{wr2}Rl_{l_8}$)]$Rl_{ap3}Rl_{l_4}Rl_{h9}$

Where, wr, ap and $h \in N$. $l \in E_l$. E_l and $N \in G_{Arc}$.

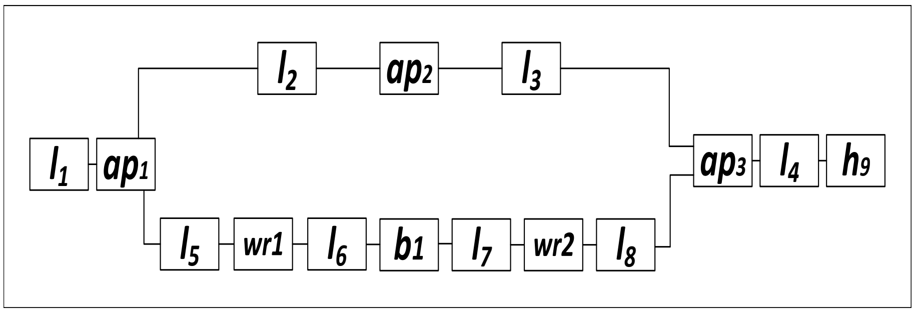


FIGURE 8.2: Reliability model of message m_3.

8.4.2 Reliability of Real-time Tasks

After computing the reliability of all incoming messages Rl_m where $m \in G_{App}$, it is time to calculate the reliability of the receiving task t (i.e. Rl_t) that receives the incoming messages. In the same way as calculating the reliability of communication messages, we assume the sender (parent) tasks and their forwarding messages as blocks. The principle of series and parallel systems is based on conditional precedence restrictions. The essential parent tasks and their forwarded messages are connected in series, whereas the substitutable tasks and their forwarded messages are connected in parallel.

For instance, Figure 8.3 illustrates the block diagram of the task t_0 reliability model. We consider that t_0 can start its execution only after receiving the message from task t_3 and messages from either t_1 or t_2. Hence, Rl_{t_0} at host $h9$ is formulated as follows:

$Rl_{t_0} = Rl_{t_3} Rl_{m_3} [1 - (1 - Rl_{t_1} Rl_{m_6})(1 - Rl_{t_2} Rl_{m_5})]$

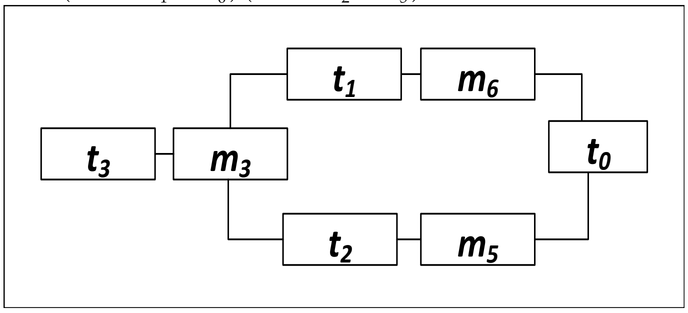


FIGURE 8.3: Reliability model of task t_0.

Where the reliability of the incoming messages is computed as introduced in the previous sub-section.

8.4.3 Reliability of Wireless TSN System

Once the reliability of all real-time tasks is computed, the system reliability is finally the reliability of the leaf task (i.e. t_0 in Figure 8.1).

$$Rl_s = Rl_t \quad \iff \quad t = leaf\_task \tag{8.11}$$

8.5 Reliable Task and Message Scheduling (R-TMS) Algorithm

R-TMS aims to find a reliable global task and message scheduling solution by considering the points mentioned in Section 8.2. Figure 8.4 shows the flow chart of the R-TMS algorithm, which is described as follows:

R-TMS procedure (General overview): For each unscheduled task $t \in T$, R-TMS computes its top-level cost (i.e. tl_t). The top-level cost is the sum of costs of the vertices (i.e. t_{et}) and the edges (i.e. m_{comm}) in the longest path from the root task (where the root task has no incoming messages) to task t in G_{App}.

The R-TMS Task Scheduling (Algorithm 1) will be repeated and the reliability of every task computed until scheduling all sorted unscheduled tasks. The final global task scheduling solution is the instant of time when the leaf task is executed and its reliability value is returned.

```
R-TMS procedure (General overview)
Input: $G_{App}, G_{Arc}$
Output: global reliability
   Initialize $\eta_n$ randomly, where $n \in G_{Arc}$. Initialize $slt_d$ period in (ms).
1: Global Reliability $\leftarrow$ 0
2: Compute top-level ($tl$) of each unscheduled-task ($t$)
3: $T_{\text{sorted}} \leftarrow$ sort$_{\text{tasks}}$(depending on $tl$ ascending order)
4: for $t \in T_{\text{sorted}}$ do
5:     Call Algorithm 1: R-TMS Task Scheduling ($t$)
6: end for
7: return global reliability ($Rl_{\text{s}}$).
```

Pseudo-code of R-TMS procedure for scheduling all tasks, which returns the global reliability.

R-TMS Task Scheduling (Algorithm 1): R-TMS schedules a task t on one of the available hosts H_t. Therefore, it assigns *Min_task_cost* as the minimum cost value of scheduling task t at host $h \in H_t$, and it is initialized to be 0. In order to find the best available host h to be assigned to task t, the algorithm proceeds as follows. Firstly, R-TMS finds for each incoming message m all available routes between the sender host and host h and extracts the disjoint redundant routes (i.e. $RR(m)_h$) by using FRER. The reliability of every root task t_{root} is equal to the reliability of its assigned host as mentioned earlier.

For each redundant route $rt(m) \in RR(m)_h$, a message scheduler (Algorithm 2) is applied. Algorithm 2 is used to compute m_{arrival} on $rt(m)$ after scheduling m on fixed duration time slots and calculating en_r and et_r where $r \in rt(m)$ by using the message interference analysis (Algorithm 3).

After scheduling message m on all $rt(m) \in RR(m)_h$, Rl_n values are computed depending on the determined et_n values in Algorithm 3, where $n \in RR(m)_h$. Further, Equations 8.1 and 8.2 are used to calculate en_m and F_m values of message m, respectively.

The *arrival* value is continuously updated until all incoming messages have arrived. If ($arrival$ + t_{et}) exceeds the t_{dl} value, where t_{dl} is defined as the maximum allowed time by which all sender tasks provide their messages and execution of task t is finished, Algorithm 2 looks at the next available host h.

The reliability of m (i.e. Rl_m) that is determined by the routes $RR(m)_h$ is computed depending on the RBD. The conditional precedence restriction is then employed to update the reliability of the task t (i.e. Rl_t) until receiving all its incoming messages. At this moment, the consumed energy en_t, failure rate F_t and maximum message arrival time *arrival* of scheduling t at host h are used as inputs to find the task scheduling cost on host h (i.e. *task_cost*). Finally, t will be assigned to the host $h \in H_t$ that results in the minimum *task_cost* value.

The process continues until scheduling all unscheduled tasks T in a G_{App}. The global reliability is computed when the leaf task is executed and its reliability value is returned.

Algorithm 1 R-TMS Task Scheduling(task t)

```
Input: unscheduled task $t$
Output: Reliability of the scheduled task $t$
1: $Min\_task\_cost \leftarrow 0$
2: for $h \in H_t$ do
3:   if $t$ is $t_{\text{root}}$ then
4:     $Rl_t = Rl_h$
5:   end if
6:   $Rl_t, Rl_t(serial), Rl_t(parallel) \leftarrow 0$
7:   for $m \in M_{\text{t}}$ do
8:     Find the redundant routes ($RR(m)_h$)
9:     for $rt(m) \in RR(m)_h$ do
10:      Call Algorithm 2: Message Scheduler
11:      $m_{\text{arrival}} \leftarrow m_{\text{IT}} + m_{\text{e2eD}}$
12:      Update $arrival = max(m_{\text{arrival}})$.
13:      if $(arrival + t_{\text{et}}) > t_{\text{dl}}$ then
14:        Invalid $h$, go to the next $h$.
15:      end if
16:    end for
17:    Calculate $Rl_n$, for each $n \in RR(m)_h$ using Eq. 8.7.
18:    Calculate $Rl_m$ using RBD.
19:    Calculate $en_m$ and $F_m$ using Eqs. (8.1, 8.2).
20:    if $m$ is essential then
21:      Update $Rl_t$, $Rl_t(serial)$ for task $t$ at host $h$.
22:    else
23:      Update $Rl_t$, $Rl_t(parallel)$ for task $t$ at host $h$.
24:    end if
25:  end for
26:  Find ($t_{\text{rt}}$)
27:  if $(t_{\text{rt}} + t_{\text{et}}) > t_{\text{dl}}$ then
28:    Invalid $h$, go to the next $h$.
29:  end if
30:  Calculate $en_{\text{t}}$, $F_{\text{t}}$ for task $t$ at host $h$ using Eqs. (8.4, 8.5).
31:  Compute $task\_cost$ value using Eq. 8.6.
32:  if $task\_cost < Min\_task\_cost$ then
33:    $Min\_task\_cost \leftarrow task\_cost$
34:    $t.runs\_on \leftarrow h$
35:  end if
36: end for
37: $Rl_{\text{s}} \leftarrow Rl_t \iff t = leaf\_task$
38: return $Rl_{\text{s}}$
```

Pseudo-code of Algorithm 1 used to schedule a task t on the best available host h and return the reliability of t at h.

Message Scheduler (Algorithm 2): It computes m_{e2eD} on each $rt(m)$ after scheduling m on feasible time slots. The scheduler starts at the n^{th} time slot where $n = \frac{m_{\text{IT}}}{slt_{\text{d}}}$, then R-TMS fragments message m into several fragments Fr_m according to the time slot duration slt_{d}. For each path l_i in the route $rt(m)$, R-TMS applies message interference analysis (Algorithm 3) to find a feasible time slot n for each $fr_m \in Fr_m$ on that path l_i. The process is repeated and n is incremented beginning from the first fragment of the first path l_1 (from the sender host) until the last fragment of the last path (i.e. l_h) in the route $rt(m)$, where $rt(m)$ is represented as $\{l_1, l_2, l_3, ..., l_h\}$.

The last incremented value of n is used to compute m_{e2eD} as a result of scheduling message m on route $rt(m)$.

Algorithm 2 Message Scheduler

```
Input: $rt(m)$, $m_{\text{IT}}$
Output: $m_{\text{e2eD}}$, scheduling the message $m$ on fixed duration time slots
    $Initialize : n = \frac{m_{\text{IT}}}{slt_{\text{d}}}$, $m_{\text{e2eD}} = 0$
1: $Fr_m \leftarrow$ Fragments of $m$ according to $slt_{\text{d}}$ value
2: for each path $l_i \in rt(m)$ do
3:     for each fragment $fr_m \in Fr_m$ do
4:         Call Algorithm 3: Message Interference Analysis
5:         $m_{\text{e2eD}} = (n + 1) * slt_{\text{d}}$
6:     end for
7: end for
8: return $m_{\text{e2eD}}$
```

Pseudo-code of Algorithm 2 used to schedule message m on conflict-free time slots on a route $rt(m) \in RR(m)_h$, after which it returns the message end-to-end delay (m_{e2eD}).

Message Interference Analysis (Algorithm 3): It adopts the physical interference model and finds a feasible time slot number (i.e. n) for every $fr_m \in Fr_m$. R-TMS checks two cases for the n^{th} time slot $Slt(n)$. In the first case $Slt(n)$ is empty, $l_i \in rt(m)$ will be inserted into the empty slot and its slot number (i.e. n) will be returned to Algorithm 2. en_r and et_r (where $r = l_i.receiver$) is equal to $en_{r.m}$ and $et_{r.m}$, respectively. In the second case $Slt(n)$ is not empty, and R-TMS computes the interference on r by every path l_j that has already been scheduled in that slot. For every path l_j, if its receiver router (i.e. $l_j.receiver$) is equal to r, then the consumed energy $en_{r.m^*}$ and the consumed time $et_{r.m^*}$ as a result of routing message m^* (m^* uses path l_j) will be added to the accumulated Sum_en_r and Sum_et_r values, respectively.

At this moment, R-TMS will compute the SINR value according to the total induced interference in $Slt(n)$ [28]. If the computed SINR is less than β, the algorithm checks the next time slot until finding a feasible slot (i.e. the computed SINR is more than β), then ($en_{r.m}$ + Sum_en_r) and ($et_{r.m}$ + Sum_et_r) values in the feasible slot will be added to en_r and et_r, respectively, l_i will be inserted in that slot and its n will be returned to Algorithm 2. At the end of the algorithm, all the paths in each time slot can be used to transmit their fragments successfully at the same time, so that the influence of the interference is prevented as much as possible.

Algorithm 3 **Message Interference Analysis**

```
Input: path $l_i$ of $fr_m$, $fr_m$, $n$ (current slot number)
Output: $n^*$ (next slot number)
    $Initialize$ : find_slot = false, $r = l_i.receiver$, {$r$ is the router to which the fragment $fr_m$ is sent.}
1:  while find_slot == false do
2:      if $Slt(n)$ == null then
3:          Insert $l_i$ on $Slt(n)$
4:          find_slot = true
5:          $en_r$ += $en_{r.m}$, $et_r$ += $et_{r.m}$
6:          $n^* = n$
7:          return $n^*$
8:      else
9:          for each path $l_j \in Slt(n)$ do
10:             Compute $I_{\text{slt}}(Slt\{l_j\}, l_i)$ {the interference on $l_i$ by all paths $l_j$'s in time-slot ($Slt$).}
11:             I += $I_{\text{slt}}$
12:             Update SINR value
13:             if $l_j.receiver$ == $r$ then
14:                 $Sum\_en_r$ += $en_{r.m^*}$, $Sum\_et_r$ += $et_{r.m^*}$, where $m \neq m^*$
15:             end if
16:         end for
17:         if SINR $\geqslant \beta$ then
18:             Insert $l_i$ on $Slt(n)$
19:             find_slot = true
20:             $en_r$ += $Sum\_en_r + en_{r.m}$, $et_r$ += $Sum\_et_r + et_{r.m}$
21:             $n^* = n$
22:             return $n^*$
23:         else
24:             Increment $n$
25:             Continue
26:         end if
27:     end if
28: end while
```

Pseudo-code of Algorithm 3 used to check the feasibility of the time slot of each fragment fr_m: it continuously updates en_r and et_r values, and then returns the feasible slot number n.

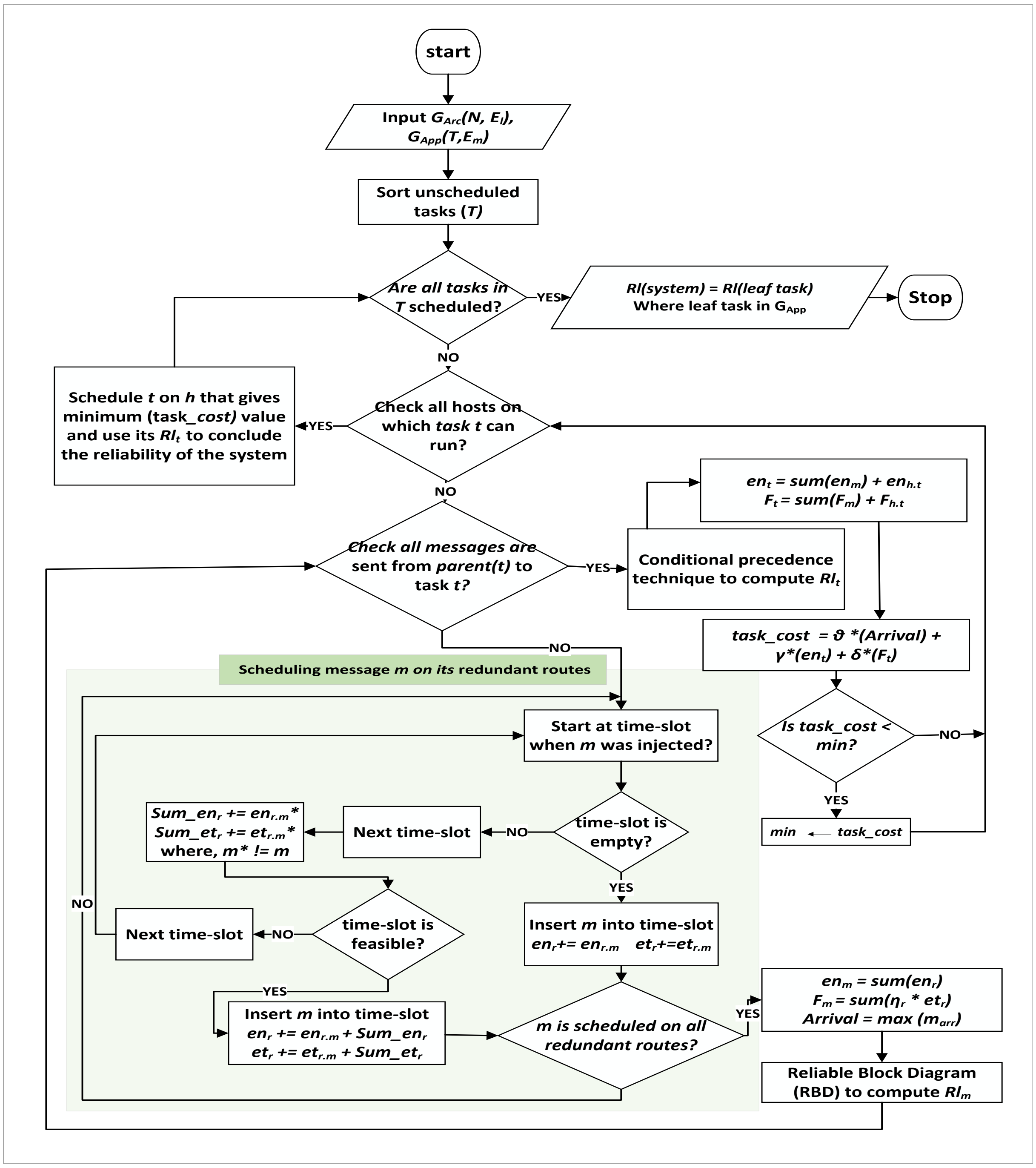

start
Input $G_{Arc}(N, E_l)$, $G_{App}(T,E_m)$
Sort unscheduled tasks (T)
Are all tasks in T scheduled?
YES
$Rl(system) = Rl(leaf\ task)$ Where leaf task in G_{App}
Stop
NO
Schedule t on h that gives minimum ($task\_cost$) value and use its Rl_t to conclude the reliability of the system
YES
Check all hosts on which task t can run?
NO
$en_t = sum(en_m) + en_{h.t}$
$F_t = sum(F_m) + F_{h.t}$
Check all messages are sent from $parent(t)$ to task t?
YES
Conditional precedence technique to compute Rl_t
NO
$task\_cost = \vartheta * (Arrival) + \gamma * (en_t) + \delta * (F_t)$
Scheduling message m on its redundant routes
Start at time-slot when m was injected?
Is $task\_cost <$ min?
NO
YES
$min \leftarrow task\_cost$
Sum_en_r += $en_{r.m}$*
Sum_et_r += $et_{r.m}$*
where, m^* != m
Next time-slot
NO
time-slot is empty?
YES
NO
Next time-slot
NO
time-slot is feasible?
Insert m into time-slot en_r+= $en_{r.m}$ et_r+=$et_{r.m}$
YES
Insert m into time-slot en_r += $en_{r.m}$ + Sum_en_r et_r += $et_{r.m}$ + Sum_et_r
m is scheduled on all redundant routes?
YES
$en_m = sum(en_r)$
$F_m = sum(\eta_r * et_r)$
$Arrival = max(m_{arr})$
Reliable Block Diagram (RBD) to compute Rl_m

FIGURE 8.4: R-TMS task and message scheduling model.

8.6 Simulation Setup to Evaluate R-TMS

In the experimental tests, the grid and ring network topologies in Figures 4.4 and 4.5 are applied, each topology contains 10 static wireless TSN-enabled nodes (i.e. wireless routers and access points), and each access point connects to four static wireless TSN hosts with 50 Mbps wireless data rate. Moreover, 20 different system models are used and generated as random Forest Fire directed graphs [218]. In our experimental setup, $\beta = 10$, $\alpha = 0.01$ and $N = 0$. The $et_{r.m}$ and $en_{r.m}$ values of routing the messages in each router r are selected randomly from a range of [20, 25], whereas $et_{h.t}$ and $en_{h.t}$ of executing the tasks in each host

h are selected randomly from a range of [200, 250]. The units of measure of energy and time are 'joule' and 'millisecond', respectively.

In addition, we assume that all tasks have the same deadline (i.e. 700 ms) and the available host set H_t to execute a computational task t is selected randomly. The 20 generated application graphs that have different inter-message dependency patterns are used to compute the general reliability and the total values are averaged.

8.6.1 Experimental Results

In Eq. 8.6, the values of weighting parameters (θ, γ and δ) are important. The selection of appropriate parameters can make the algorithm achieve better results. Therefore, we compare the reliability of the system by setting different values of weighting parameters. The values in Table 8.3 achieve better results in the computed reliability. These values are obtained by taking the average result of several tests.

TABLE 8.3: Test of several parameter settings.

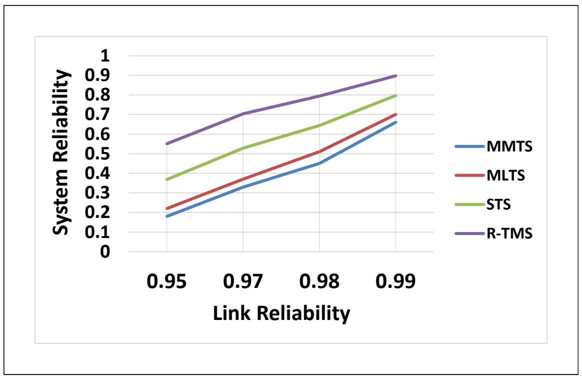
θ	γ	δ	Rl_s
0.45	0.1	0.45	0.8975
0.4	0.2	0.4	0.8944
0.6	0.1	0.3	0.8927
0.3	0.1	0.6	0.8920

In order to evaluate the system reliability and efficiency of R-TMS, we compare R-TMS with MMTS and MLTS algorithms that are mentioned in Chapter 7. We also contrast R-TMS with the Shortest Task Scheduling (STS) algorithm [24], [25], which primarily relies on choosing the shortest routes to deliver the periodic messages. According to the route selection mechanism, tasks are scheduled to available hosts in R-TMS, MLTS and STS algorithms.

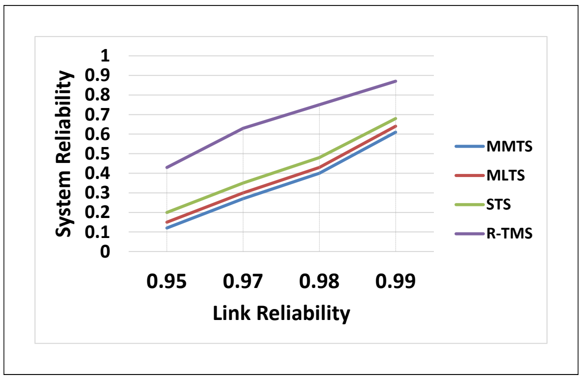
System Reliability Based on the Link Reliability

Figure 8.5 illustrates how the system reliability increases when increasing link reliability for all evaluated algorithms in our modelled grid and ring topologies. Figure 8.5a shows that the average system reliability of the generated solution by R-TMS is improved in the grid topology by 29%, 80% and 105% compared to the reliability of the generated solutions by STS, MLTS and MMTS, respectively. Figure 8.5b shows that R-TMS improves the system reliability in the ring topology by 69%, 101% and 130% compared to STS, MLTS and MMTS, respectively. The noticeable improvement of the system reliability in the grid topology compared to the ring is due to the fact that grid networks contain more routes and thus the link reliability has a greater impact.

The significant improvement of the system reliability in R-TMS is achieved by forwarding copies of messages over disjoint redundant routes. STS shows better reliability than MLTS because STS forwards one copy of a message over the shortest routes instead of choosing the least congested routes that may be longer and thus MLTS reduces its reliability as more nodes and links contribute to sending messages. MMTS shows the worst system reliability because it selects a prepared solution before starting the task scheduling process. The proposed solution schedules tasks to hosts that have the minimum completion time. MMTS's solution does not also consider the communication costs and the reliability of the nodes and links that the periodic messages pass through. It should be noted that R-TMS selects the fastest routes to forward messages. These routes may be the shortest as in STS or maybe the least congested ones as in MLTS. The shortest routes are not always the fastest due to the congestion that may cause delayed delivery of messages, so choosing less congested routes in some cases leads to faster delivery of messages, even if they are longer. Thus, R-TMS regards the transmission time as a priority due to the importance of the bounded time in real-time applications.



(A) Grid topology.



(B) Ring topology.

FIGURE 8.5: System reliability by applying R-TMS, STS, MLTS and MMTS for different link reliabilities, tasks = 10 and messages = 30.

Nevertheless, Table 8.4 shows the makespan, consumed energy and CPU time of R-TMS, STS, MLTS and MMTS algorithms. The grid topology is applied to schedule 80 tasks with 200 messages, and Table 8.5 shows the same comparison for the ring topology. As illustrated, the grid topology needs less CPU-time than the ring topology for MMTS, STS and R-TMS algorithms due to the multiplicity of options for sending messages, while the ring topology consumes more CPU time until it reaches a feasible solution. We notice that MLTS needs almost half the CPU-time for the ring topology compared to the grid topology because the search space to find the least congested routes is significantly smaller in the ring topologies.

R-TMS increases the makespan and consumed energy to generate its solutions compared to STS, MLTS and MMTS. The reason is that the duplication of messages in the R-TMS algorithm needs more time and energy until receiving all message copies. However, increasing the time in R-TMS does not affect its feasibility when it still permits the execution of tasks before their deadlines. In addition, γ parameter in Eq. 8.6 can be modified (i.e. γ is the highest weight) to give the consumed energy the highest priority when scheduling the tasks. Using the weight sum in Eq. 8.6 contributes to obtaining better results and controls the task and message scheduling. In other words, the weighting parameters are set based on the importance of the corresponding factors regarding the specifications of the network and the constraints of the applications. For example, the value of the θ parameter, which corresponds to the $arrival$, gets a higher weight in the case of applying hard real-time applications. Similarly, the weight of γ is higher than other weights for networks consisting of hosts with limited energy (e.g. sensors). So in general, the arrival time has been considered of greater importance because of dealing with fault-tolerant real-time systems.

STS shows better makespan than MLTS but not in all cases because of the potential congestion along the shortest routes. On the contrary, MLTS consumes the most CPU-time to find the least congested routes of all routing possibilities because of the larger search space, especially for complex system models. MMTS requires the least CPU-time because it prepares its solution at the beginning of executing the algorithm and it is not trying to find the best routes during the task scheduling process, but on the other hand, we notice that it consumes more makespan and more energy compared to MLTS and STS algorithms.

TABLE 8.4: Comparison of the completion-time, consumed energy and CPU-time for MMTS, MLTS, STS and R-TMS in the grid topology, tasks = 80, messages = 200.

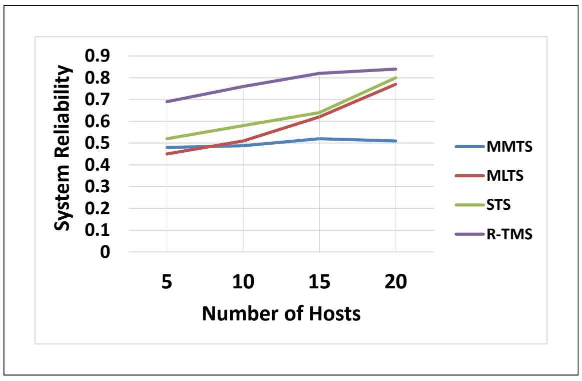
Algorithm	**Makespan**	**Consumed energy**	**CPU-time**
MMTS	3545	40050	1.5
MLTS	3378	35000	36.1
STS	3203	34102	3.5
R-TMS	5344.5	67670	10.39

TABLE 8.5: Comparison of the completion-time, consumed energy and CPU-time for MMTS, MLTS, STS and R-TMS in the ring topology, tasks = 80, messages = 200.

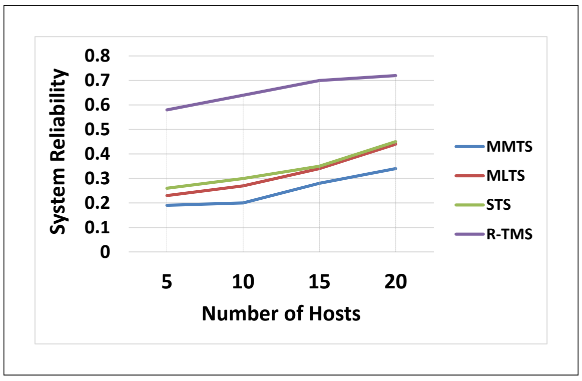
Algorithm	**Makespan**	**Consumed energy**	**CPU-time**
MMTS	4100	4516	1.6
MLTS	3853	4165	14
STS	3781	4083	5
R-TMS	7569	90235	15.29

System Reliability Based on the Number of Hosts

In the second experimental test, we study the impact of the number of available hosts on the system reliability. For this purpose, we consider that the link reliability is 0.99 for all algorithms. Figure 8.6a shows that the average system reliability of R-TMS in the grid topology improves considerably by 24%, 35% and 54% compared to STS, MLTS and MMTS, respectively. Figure 8.6b shows that R-TMS improves rapidly the system reliability in the ring topology to be 99%, 114% and 171%, compared to STS, MLTS and MMTS, respectively. We observe that the system reliability rises as the number of hosts used to schedule tasks for all algorithms increases. Moreover, there is a minor effect on the system reliability when using R-TMS and it is applied in the ring topology instead of the grid one compared with the other algorithms. The reason is that R-TMS optimizes the reliability in presence of variability of available hosts. To achieve optimal reliability, R-TMS distributes and balances the load of tasks according to the consumed energy, time and failure rate of several available hosts. In addition, applying FRER and scheduling different messages in their time slots with the minimum induced interference significantly contributes to improving the global system reliability.



(A) Grid topology.

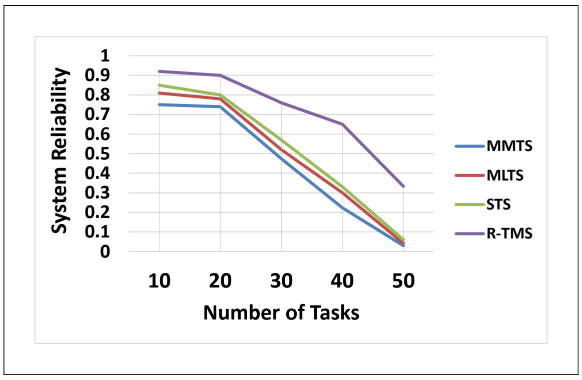


(B) Ring topology.

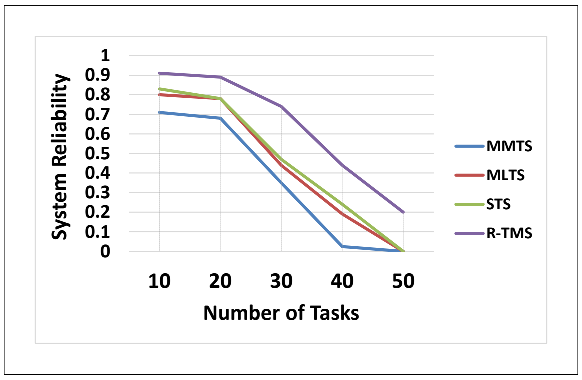
FIGURE 8.6: System reliability when applying R-TMS, STS, MLTS and MMTS for different numbers of hosts, tasks = 30, messages = 90 and link reliability = 0.99.

System Reliability Based on the Number of Tasks

As illustrated in Figure 8.7, due to the host availability and limited resources, the overall system reliability is decreasing when increasing the number of tasks (i.e. network utilization). The higher number of tasks has a greater impact on ring networks because they consume more network resources. Likewise, R-TMS gives better system reliability than STS, MLTS and MMTS when increasing the load (the number of tasks). The R-TMS algorithm shows a clear improvement in system reliability compared to others, especially with more tasks that need to be scheduled. Increasing the number of tasks makes it more difficult to perform all tasks reliably, especially for tasks that take into account precedence constraints, which means that the task is executed only after receiving all messages from its parent tasks. R-TMS solves this problem by implementing a reliability model that guarantees conditional precedence constraints to maintain system reliability even when tasks fail.



(A) Grid topology.



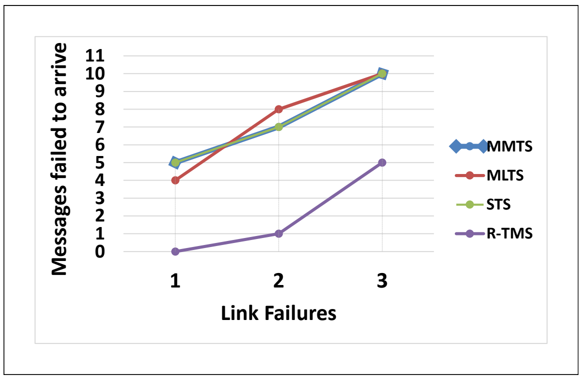
(B) Ring topology.

FIGURE 8.7: System reliability when applying R-TMS, STS, MLTS and MMTS for different numbers of tasks, with a link reliability = 0.99.

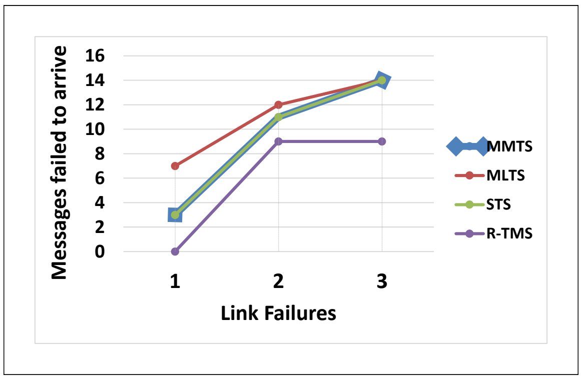
System Reliability Based on Link Failures

Figure 8.8 shows that increasing the number of injected link failures leads to more messages that fail to arrive at their receiving tasks for all compared algorithms. R-TMS outperforms STS, MLTS and MMTS with a lower number of failed messages because of multiple paths, which are used to send messages, reducing the possibility of message failures.

We note that the impact of the link failure is greater on ring networks, which is expected because the links in the ring topologies are used more than in the grid topologies and thus the probability of failure to deliver messages is significantly greater for all algorithms. The messages that failed to arrive in R-TMS remain less compared to other algorithms, as we explained earlier because replicas of a message are sent over redundant routes. However, the increase of the number of link failures makes sending of messages more complex for all algorithms. MMTS shows the same failure rate in sending messages compared to STS because MMTS uses the shortest routes to send messages to prepare the task scheduling solutions.



(A) Grid topology.



(B) Ring topology.

FIGURE 8.8: Number of messages failing to arrive by applying R-TMS, STS, MLTS and MMTS for different numbers of link failures, tasks = 8 and messages = 20.

8.7 Problem Formulation of OR-TMS

OR-TMS [30] uses the DPSO algorithm to find an optimized task and message scheduling solution with regard to a given measure of quality. The solution is evaluated by considering multiple objectives including the global makespan, total energy consumption and total failure rates of scheduling all tasks. In contrast, R-TMS schedules each task one after the other until all tasks are completed. OR-TMS has the advantage that it selects a solution that distributes the tasks in a way taking into consideration the constraints in an integrated manner for all tasks, while R-TMS [29] is a heuristic algorithm, which looks to schedule a current task considering only constraints of the previous tasks (i.e. parent tasks).

OR-TMS considers our hybrid modelling approach mentioned in Chapter 4 as inputs to schedule T tasks to sets of available hosts. In our work, the term "*task-scheduling-instance*" is used. It refers to distributing the tasks to potential hosts so that each instance has its own distribution of the tasks. To explain the *task-scheduling-instance* and determine its cost of using the network resources, we introduce the following points:

- The OR-TMS algorithm considers the task and message scheduling constraints of the TMS algorithm mentioned in Section 7.2.
- We consider T as unscheduled tasks and H_t as a changeable set of available hosts on which a task $t \in T$ can run. The cost of the *task-scheduling-instance* is determined when scheduling all tasks.
- FRER is used to send replicas of a message m through redundant routes $RR(m)_h$ from every parent task to a child task t that is scheduled on an available host $h \in H_t$.

- The consumed energy and the failure rate of routing a message m are computed.

  The energy consumption of routing a message m (i.e. en_m) is the sum of the consumed energy of routing m on each router $r \in RR(m)_h$.

$$en_m = \sum_{r=1}^{RR(m)_h} en_r \qquad \forall r \in RR(m)_h \tag{8.12}$$

  The failure rate of routing a message m (i.e. F_m) is shown in Eq. 8.13.

$$F_m = \sum_{r=1}^{RR(m)_h} \eta_r \times et_r \qquad \forall r \in RR(m)_h \tag{8.13}$$

  Where, en_r and et_r are the consumed energy and the execution time on a router $r \in RR(m)_h$, respectively. The time slot message scheduler in Section 8.3 uses a message interference analysis to compute en_r and et_r values during the message transmission. η_r denotes the failure rate of r.

- The arrival time of all incoming messages at task t is found, and the consumed energy and the failure rate of executing task t at host h are computed.

  The arrival time (i.e. *arrival*) denotes the instant of time when all incoming messages have arrived at the child task t at host h, where t's execution depends on the arrival of all parent messages. *Arrival* at host h is defined as:

$$arrival = max(m_{\text{arrival}}) \qquad \forall m \in M_t \tag{8.14}$$

  The consumed energy and the failure rate after the arrival of all messages and the completion of task t at host h are computed as follows:

$$en_t = \sum_{m=1}^{M_t} en_m + en_{h.t} \qquad \forall m \in M_t \tag{8.15}$$

$$F_t = \sum_{m=1}^{M_t} F_m + F_{h.t} \qquad \forall m \in M_t \tag{8.16}$$

  Where $en_{h.t}$ and $F_{h.t}$ denote the consumed energy and the failure rate of executing task t at host h, respectively. After executing all tasks (i.e. T) on the selected available hosts, a cost function for the *task-scheduling-instance* is formulated as shown in Eq. 8.17.

$$Cost = \theta * makespan + \gamma * en_T + \delta * F_T \tag{8.17}$$

  en_T and F_T denote the consumed energy and the failure rates as a result of scheduling all tasks T, respectively. The *makespan* is the instant of time at which all tasks are finished. θ, γ and δ are weighting parameters.

  The computed cost function value determines the cost of scheduling all tasks on the selected available hosts. Lower values indicate better instances. Finding the best possible scheduling instance depends mainly on selecting the best distribution of tasks to the available hosts, where each task has a different set of available hosts. Therefore, DPSO will be employed in the course of the OR-TMS algorithm.

8.8 Discrete Particle Swarm Optimization (DPSO)

8.8.1 PSO Definition

PSO is a heuristic and computational algorithm developed by Kennedy and Eberhart [222]. It optimizes a problem with regard to its computed fitness value. The most important features of PSO over other algorithms are ease of implementation and fast convergence to get a reasonable solution. PSO also preserves the diversity of the population to avoid premature convergence. Therefore, it is widely used in various experiments to solve optimization problems [92], [224].

PSO aims to solve an optimization problem by iteratively trying to improve a solution with regard to a fitness value. PSO consists of a swarm (i.e. population) of individual particles. Firstly, particles are initialized with random solutions and through many iterations, every particle moves in a direction to search for a better solution. Thus, each particle is defined by two variables the position and velocity. The position determines its location in the space and the velocity is used to determine its movement and its direction. During the iterations, every particle updates its position according to its velocity. Besides the previous variables, each particle has also two best values. The first best value is P_{best}, which the best local position value for each particle so far. This value is adjusted in accordance with the particle's position's fitness value. The second best value is G_{best} which is the best global position value for the swarm so far. G_{best} is checked in every iteration and is changed by P_{best} value of any particular particle at the iteration $k+1$ if the fitness value of P_{best} is better than the fitness value of G_{best} in the previous iteration k.

Following are the equations, which are used to update the parameters and determine the best values for each particle in the swarm.

$$V_i^{k+1} = w \cdot V_i^k + c_1 \cdot r_1 \cdot \left(P_{best_i}^k - X_i^k\right) + c_2 \cdot r_2 \cdot \left(G_{best}^k - X_i^k\right) \tag{8.18}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{8.19}$$

Equation 8.18 computes a new velocity V_i^{k+1} for each particle (p_i) at iteration $k+1$ according to its previous velocity V_i^k at iteration k. The position of the particle at which the best fitness value for that particle has been attained so far is continuously updated in P_{best}. The global best position at which the best fitness value has been achieved so far in the swarm is continuously updated in G_{best}. Eq. 8.19 updates the particle's position (i.e. X_i^{k+1}) at iteration $k+1$ by adding the new velocity resulting from Eq. 8.18 to its previous position at iteration k. r_1 and r_2 are random values, c_1 and c_2 are factors used to determine the acceleration and the pull for each particle in the swarm toward P_{best} and G_{best} values, w is an inertia weight and it is decreased in every iteration.

8.8.2 Mapping of a Task Scheduling Instance to DPSO

PSO was introduced as a continuous optimization problem, while the task scheduling in *WirelessTSN* is a discrete optimization problem. Energy savings, reliability, and timeliness objectives in *WirelessTSN* are defined as optimization problems that should be resolved during the task and message scheduling process. Thus, a discrete search space is introduced to the continuous PSO.

Inspired by the idea of PSO, the *task-scheduling-instance* mentioned in Section 8.7 is formulated and mapped to a particle's position and each particle's position has one dimension for all tasks T. Therefore, this section describes how Discrete PSO (DPSO) is formulated for the task and message scheduling problem. One of the key issues for a successful

DPSO representation is finding a suitable mapping between the *task-scheduling-instance* and DPSO particle's position. We setup the discrete search space of the dimension for the task scheduling problem. This dimension is a set of discrete sets of possible values located in $s = \{h_i : 1 \leq i \leq H_t\}$; such that H_t is the number of possible hosts on which task $t \in T$ can run. For example, the problem in Figure 8.9 is considered to map 5 tasks $\{t_0 : t_4\}$ onto the grid graph of our system model in Figure 4.4. Table 8.6 shows that each task t can be run on a host chosen from the set of hosts $\{H_{t_0} : H_{t_4}\}$. For example, task t_0 can be run on one of its available hosts $\{h6, h7, h9, h10, h14\}$.

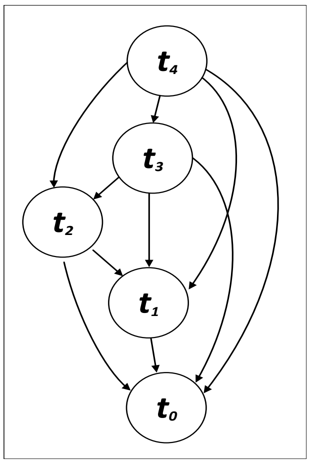


FIGURE 8.9: Application graph consisting of 5 tasks with 10 messages.

TABLE 8.6: Set of available hosts for each task $t \in T$.

Task number	Available set (H_t)
t_0	$\{h6, h7, h9, h10, h14\}$
t_1	$\{h4, h8, h13, h14, h17\}$
t_2	$\{h10, h13, h18, h20\}$
t_3	$\{h1, h6, h10, h15, h18\}$
t_4	$\{h2, h4, h9, h15\}$

Figure 8.10 illustrates a mapping of a possible task scheduling instance [6 4 13 10 2] to particle position coordinates in the DPSO domain. A vector of hosts is used to represent the particle's position, and each host $h \in H_t$ is internally stored as an integer value in the vector, with the index indicating the task t that each host is given during the DPSO process. For example, t_0 can be scheduled to host 6 because host 6 belongs to t_0's available set $\{6, 7, 9, 10, 14\}$ where t_0 can run. t_1 can be scheduled to host 4 because host 4 belongs to t_1's available set $\{4, 8, 13, 14, 17\}$. The same scenario is applied to assign a host to all other tasks $\{t_2 : t_4\}$ that run on different available sets. The algorithm starts randomly generating several possible instances depending on the size of the initial population of the DPSO. Thus, the population of the DPSO is represented as a two dimensional HxT matrix consisting of a defined number of particles, where each particle p_i is represented as a vector of T tasks and its scope is a T-dimensional discrete search space. At the end of the DPSO implementation, all particles are converged to a global position that represents an optimized instance (i.e. solution) for the task scheduling problem.

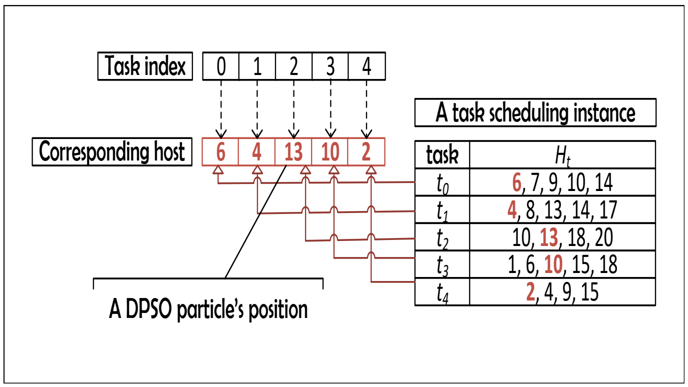


FIGURE 8.10: Mapping a task assignment instance to a DPSO particle.

A *load-balance-mechanism* should be taken into account for an effective task scheduling process to increase the network lifespan and decrease the likelihood of failing to map a *task-scheduling-instance* in the event that a host's energy is depleted. Using the *load-balance-mechanism*, the number of times the host is assigned to the particle of a task scheduling instance does not exceed the permissible limit as shown in the following equation:

$$Max\_ass_h = int(T/H) + 1 \tag{8.20}$$

Max_ass_h represents the maximum number of assignments of a host to an scheduling instance. For our example, Figure 8.9 shows 5 tasks (T = 5) and 20 available hosts (H = 20). According to Eq. 8.20, the output value of Max_ass_h equals 1, which means any host should be assigned only once to any possible instance. For example, instances such as [7 17 18 18 2] and [6 17 18 6 2] are not considered regarding Table 8.6 as possible instances because hosts 18 and 6 are used twice to assign tasks in the first and the second scheduling instances, respectively. Therefore, DPSO generates at the beginning different instances that satisfy the condition in Eq. 8.20. During the course of DPSO, if the new position of a particle does not satisfy the condition, DPSO rounds the not compliant host to the closest host that satisfies the mentioned condition. For example, host 18 in the instance [7 17 18 18 2] is used twice to schedule tasks t_2 and t_3, thus, DPSO in Figure 8.11 rounds the host, which is assigned to task t_3, to the closest available host (i.e. host 15). This process is implemented in every iteration for all particles until DPSO is terminated and a unified solution is obtained.

Increasing the number of tasks relating to the total number of hosts leads to an increase in the maximum number of assignments per host. For example, if T = 30 and H = 20, then the number of assignments of a host in any *task-scheduling-instance* could be a maximum of two.

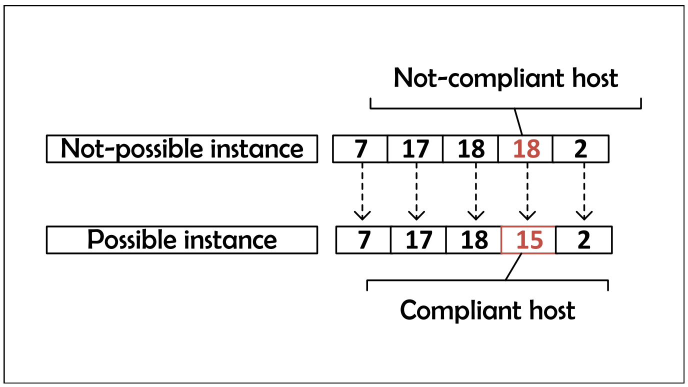


FIGURE 8.11: Rounding the not compliant host to the closest available host satisfying the load balance condition.

8.8.3 Task and Message Scheduling Process Based on DPSO

To explain how our DPSO-based algorithm works, DPSO starts with random *task-scheduling-instances* as a generated population. Each instance is evaluated by measuring its cost function value using Eq. 8.17. We conclude that Eq. 8.17 expresses the task and message scheduling problem as a Multi-objective Optimization Problem (MOP) by means of a weight sum of the system's completion time (i.e. makespan), total energy consumption and total failure rates. The solution and the cost function value are updated gradually until scheduling all transmitted messages and executing all tasks on their corresponding hosts. DPSO keeps updated G_{best} and P_{best} values during the course of its execution. If the cost of a particle's current place is less than that of previously visited positions, P_{best} is updated. If the cost of any particle's current position is less than the cost of the best global position so far, G_{best} of all the particles in the swarm is updated. According to Equations 8.18 and 8.19, these two revised values cause every particle to have a new velocity, which results in a new position. The random parameters in 8.18 prohibit a bias effect from either the global or the local best positions; as a result, these parameters regulate the process of the exploration and exploitation forced by the particle's new velocity randomly. Once all particles update their new positions, a new status of the DPSO population is established. The cost of the updated G_{best}, which represents the minimum cost found so far, is compared with previous cost function values. If the new cost function value is changed, then the whole process is repeated with a new iteration to determine new global and local best positions until the convergence condition is satisfied. In general, the convergence condition is met when G_{best} reaches a value that the particle cannot update and find a lower value, so the DPSO is terminated and the optimized global position (i.e. the final solution) is obtained.

The flow chart in Figure 8.12 gives an overview of OR-TMS by applying DPSO. Each particle firstly supposes a random position which represents a solution to schedule tasks on random possible hosts. Each particle's solution is evaluated in DPSO by calculating the cost function value $Cost(p_i)$ of the proposed solution. To do so, every task is not scheduled to the assigned host until all message replicas sent from all parents through redundant routes are received. The time slot message scheduler is used to schedule the message traffic using conflict-free time slots in order to avoid signal interference. To determine the $Cost(p_i)$ value, OR-TMS computes the task completion time, overall energy consumption, and failure rates of scheduling all tasks. After computing $Cost(p_i)$ of all particles, DPSO generates the best global solution G_{best} depending on the lower $Cost(p_i)$ value. DPSO then updates new positions and velocities in parallel for all particles. If the new global solution does not meet the terminal condition ($Cost(G_{best})$ is not converged), DPSO continues with the next iteration.

A technique known as the random method [225] is used to change a particle's velocity such that the new position of the particle is always inside a feasible area, as indicated in the following formulas:

$$\begin{aligned} &\text{if } X_i^{k+1} > X_{i.max}, \text{ then} \\ &\quad V_i^{k+1} = \text{rand}\left(0, X_{i.max} - X_i^k\right) \end{aligned} \tag{8.21}$$

$$\begin{aligned} &\text{if } X_i^{k+1} < X_{i.min}, \text{ then} \\ &\quad V_i^{k+1} = \text{rand}\left(0, X_i^k - X_{i.min}\right) \end{aligned} \tag{8.22}$$

Where, $X_{i.max}$ and $X_{i.min}$ stand for the highest and lowest host integer value for a particle i in the H domain.

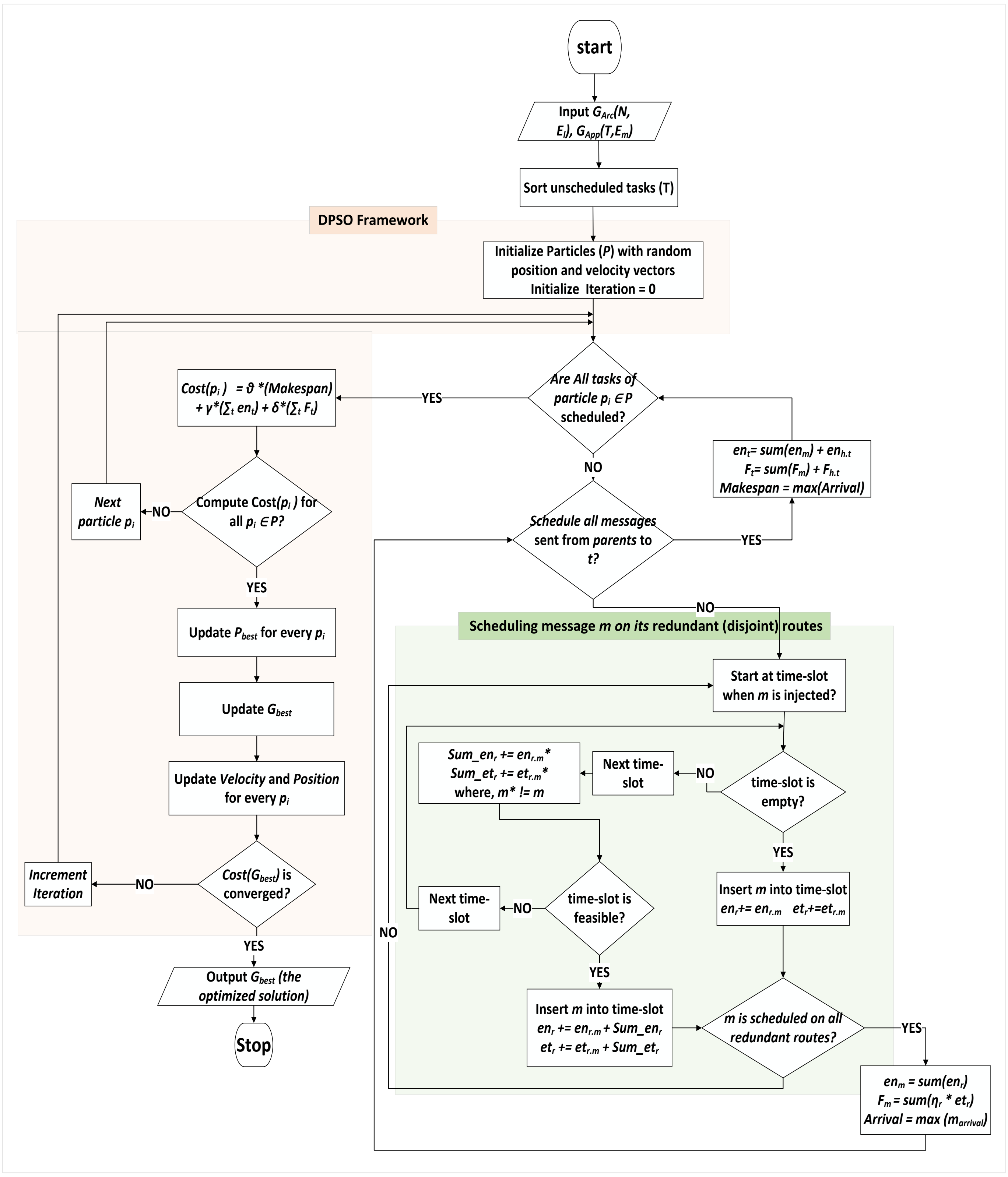

start
Input $G_{Arc}(N, E_l)$, $G_{App}(T,E_m)$
Sort unscheduled tasks (T)
DPSO Framework
Initialize Particles (P) with random position and velocity vectors
Initialize Iteration = 0
Are All tasks of particle $p_i \in P$ scheduled?
YES
NO
$Cost(p_i) = \vartheta * (Makespan) + \gamma * (\sum_t en_t) + \delta * (\sum_t F_t)$
Compute Cost(p_i) for all $p_i \in P$?
Next particle p_i
Update P_{best} for every p_i
Update G_{best}
Update Velocity and Position for every p_i
$Cost(G_{best})$ is converged?
Increment Iteration
Output G_{best} (the optimized solution)
Stop
Schedule all messages sent from parents to t?
$en_t = sum(en_m) + en_{h.t}$
$F_t = sum(F_m) + F_{h.t}$
Makespan = max(Arrival)
Scheduling message m on its redundant (disjoint) routes
Start at time-slot when m is injected?
time-slot is empty?
Next time-slot
Sum_en_r += $en_{r.m^*}$
Sum_et_r += $et_{r.m^*}$
where, m^* != m
Insert m into time-slot en_r+= $en_{r.m}$ et_r+=$et_{r.m}$
time-slot is feasible?
Insert m into time-slot en_r += $en_{r.m}$ + Sum_en_r et_r += $et_{r.m}$ + Sum_et_r
m is scheduled on all redundant routes?
$en_m = sum(en_r)$
$F_m = sum(\eta_r * et_r)$
Arrival = max ($m_{arrival}$)

FIGURE 8.12: OR-TMS task and message scheduling model.

Algorithm 1 shows the pseudo-code of the DPSO-based task and message scheduling procedure. The particle's position X_i represents a solution for scheduling tasks to random hosts. The cost function value of a position $Cost(X_i)$ is used to determine the particle's solution. The lower the value, the better the particle's location. In general, the convergence condition refers to finding a satisfactory solution. If the convergence condition is met, the solution (the global best position) is obtained and the run of DPSO is terminated.

Algorithm 1 DPSO-based Task and Message Scheduling Procedure

```
Input: Tasks, Particles
Output: $G_{best}$
1: for each particle $p_i$ do
2:   Initialize position $X_i$ randomly
3:   Initialize velocity $V_i$ randomly
4:   Initialize $P_{best_i}$ with a copy of $X_i$
5: end for
6: for each particle $p_i$ do
7:   Evaluate $Cost(P_{best_i})$
8:   Initialize $G_{best}$ with a copy of $P_{best_i}$ with the least cost
9: end for
10: while the convergence condition is not met do
11:   for each particle $p_i$ do
12:     Call Algorithm 2 OR-TMS(T)
13:     Evaluate $Cost(X_i)$ using Eq. 8.17
14:     if $Cost(X_i) < Cost(P_{best_i})$ then
15:       $P_{best_i} \leftarrow X_i$
16:     end if
17:     if $Cost(X_i) < Cost(G_{best})$ then
18:       $G_{best} \leftarrow X_i$
19:     end if
20:   end for
21:   if the convergence condition is met then
22:     Output $G_{best}$; Break
23:   else
24:     for each particle $p_i$ do
25:       Update $V_i$ according to Eq. 8.18
26:       Update $X_i$ according to Eq. 8.19
27:     end for
28:   end if
29: end while
```

DPSO-based task and message scheduling procedure.

8.9 Evaluation of a Task Scheduling Instance (A Particle's Position)

This section shows how OR-TMS employs DPSO (Algorithm 2) to evaluate a *task-scheduling-instance* (i.e. a particle's position) in the swarm. OR-TMS algorithm introduces a reliable task and message scheduling process to consider the assumptions mentioned in Section 8.1.

OR-TMS (Tasks T) (Algorithm 2): For each unscheduled task t, OR-TMS algorithm computes firstly its top-level cost tl_t using Equations 7.7 and 7.8. OR-TMS sorts all unscheduled tasks in ascending order according to their top-level values. Every sorted task applies Algorithm 3 until scheduling all tasks of a particle's position. At the instant of time when all tasks are executed within the hyper period time $HP.G_{App}$ [226], the consumed energy en_T, failure rates F_T and makespan of all tasks are computed, where the makespan is defined as the completion time of the execution of the leaf task (i.e. a task that has no forwarding messages).

Outputs of Algorithm 2 are used to compute the cost function value of the current instance, which is used to be compared with other instances in DPSO.

Algorithm 2 OR-TMS(T)

```
Input: $G_{App}$, $G_{Arc}$
Output: $F_T$, $en_T$ and $makespan$
1: Sort(tasks $T$)
2: for $t \in T_{\text{sorted}}$ do
3:     Call Algorithm 3: Task $t$ Scheduling
4:     $F_T$ += $F_t$
5:     $en_T$ += $en_t$
6:     $makespan = max(arrival)$
7: end for
8: return $F_T$, $en_T$ and $makespan$
```

Pseudo-code of Algorithm 2 used to schedule all tasks to their corresponding hosts, outputs are used to evaluate the current DPSO instance.

Task t Scheduling (Algorithm 3): It shows how to schedule task t to its corresponding host h in a particle's position. Algorithm 3 firstly finds for each incoming message m all available routes between the sender host of message m and host h, then it extracts the disjoint redundant routes $RR(m)_h$ by using FRER.

For each redundant route $rt(m) \in RR(m)_h$, a message scheduler (Algorithm 4) is applied. Algorithm 4 is used to compute m_{e2eD} as a result of using a route $rt(m)$ and scheduling m on conflict-free time slots. m_{e2eD} is then used to find the arrival time m_{arrival} of message m. The arrival time of all messages, the consumed energy en_m and the failure rate F_m as a result of sending message m are continuously updated until all messages that use their redundant routes have arrived at task t.

As discussed earlier, OR-TMS considers the period of the tasks. Equation 7.9 in Chapter 7 is used to compute the number of repetitions of a parent task to start injecting its message towards its child task.

If the updated *arrival* time and execution time of task t (i.e. t_{et}) violate the t_{dl} value, where t_{dl} is defined as the maximum allowed time by which all parent tasks provide their messages and execution of task t is completed, Algorithm3 considers the particle's position (i.e. the *task-scheduling-instance*) as an invalid solution.

At this moment, Equations 8.15 and 8.16 are used to compute the consumed energy en_t and the failure rate F_t of scheduling t to its corresponding host h in the particle's position.

Algorithm 3 Task *t* Scheduling

```
Input: unscheduled task $t$
Output: $arrival$, $en_t$ and $F_t$
 1: for $m \in M_t$ do
 2:     Find the redundant routes ($RR(m)_h$)
 3:     for $rt(m) \in RR(m)_h$ do
 4:         Call Algorithm 4 Message Scheduler
 5:         $m_{\text{arrival}} \leftarrow m_{\text{IT}} + m_{\text{e2eD}}$
 6:         Update $arrival = \max(m_{\text{arrival}})$
 7:         if $(arrival + t_{\text{et}}) > t_{\text{dl}}$ then
 8:             Invalid solution, break.
 9:         end if
10:         Update $en_m$
11:         Update $F_m$
12:     end for
13: end for
14: Find($t_{\text{rt}}$)
15: if $(t_{\text{rt}} + t_{\text{et}}) > t_{\text{dl}}$ then
16:     Invalid solution, break.
17: end if
18: Compute $en_t$, $F_t$ using Eqs. (8.15, 8.16).
19: return $arrival$, $en_t$ and $F_t$
```

Pseudo-code of Algorithm 3 used to schedule a task to its corresponding host.

Message Scheduler (Algorithm 4): The scheduler computes m_{e2eD} of routing message m on a redundant route $rt(m)$ after scheduling m on its feasible time slots. The scheduler starts at the time slot number $n = \frac{m_{\text{IT}}}{slt_{\text{d}}}$, where m_{IT} and slt_{d} are defined as the message injection time and the duration of the time slot, respectively. Algorithm 4 then fragments message m into several fragments Fr_m according to the duration of slt_{d}. For each path l_i in the route $rt(m)$, Algorithm 4 applies message interference analysis (Algorithm 5) to find a feasible time slot number (i.e. n) for each $fr_m \in Fr_m$ using that path l_i. The process is repeated and n is incremented beginning from the first fragment of the first path l_1 (from the sender host) until the last fragment of the last path l_h in the route $rt(m)$. The last incremented value of n is used to compute m_{e2eD} of routing message m on the route $rt(m)$. Equations 8.12 and 8.13 are used to return the values of en_m and F_m, respectively.

Algorithm 4 Message Scheduler

```
Input: $rt(m)$, $m_{\mathrm{IT}}$
Output: $m_{\mathrm{e2eD}}$, $en_m$ and $F_m$
    Initialize : $n = m_{\mathrm{IT}}/slt_{\mathrm{d}}$ , $m_{\mathrm{e2eD}} = 0$
1: $Fr_m$ = Fragments of $m$ according to $slt_{\mathrm{d}}$ value
2: for each link $l_i \in rt(m)$ do
3:     for each fragment $fr_m \in Fr_m$ do
4:         Call Algorithm 5: Message Interference Analysis
5:         $m_{\mathrm{e2eD}} = (n + 1) * slt_{\mathrm{d}}$
6:     end for
7: end for
8: Compute $en_m$, $F_m$ using Eqs. (8.12, 8.13).
9: return $m_{\mathrm{e2eD}}$, $en_m$ and $F_m$
```

Pseudo-code of Algorithm 4 used to schedule message m on conflict-free time slots on a route $rt(m) \in RR(m)_h$.

Message Interference Analysis (Algorithm 5): The proposed algorithm adopts the physical interference model to find a feasible time slot used to transmit every $fr_m \in Fr_m$. Algorithm 5 checks two cases for the time slot number n (i.e. $Slt(n)$), where n is the number of the current time slot. In the first case $Slt(n)$ is empty, $l_i \in rt(m)$ will be inserted into the empty slot and its slot number (i.e. n) will be returned to Algorithm 4. en_r and et_r (where $r = l_i.receiver$) are equal to $en_{r.m}$ and $et_{r.m}$, respectively. In the second case $Slt(n)$ is not empty, and Algorithm 5 computes the interference on $r \in rt(m)$ by every path $l_j \in Slt(n)$ that has already been scheduled in that slot. If the receiver router (i.e. $l_j.receiver$) of a path l_j is equal to router r, then the consumed energy $en_{r.m^*}$ and the consumed time $et_{r.m^*}$ as a result of routing message m^* (m^* uses path l_j) will be added to the accumulated Sum_en_r and Sum_et_r values, respectively. At this moment, the SINR value is computed according to the total induced interference in that slot [28]. If the computed SINR value is less than β, the algorithm checks the next slot until finding a feasible slot (i.e. on which the computed SINR value is more than β). Then ($en_{r.m} + Sum\_en_r$) and ($et_{r.m} + Sum\_et_r$) values in the feasible slot will be added to en_r and et_r, respectively. l_i will be inserted into that slot and its n will be returned to Algorithm 4. At the end of the process, all path in each slot can be used to successfully send their fragments at the same time, minimizing the impact of interference.

Algorithm 5 Message Interference Analysis

```
Input: path $l_i$ of $fr_m$, $fr_m$, $n$ (current slot number)
Output: $n^*$ (next slot number)
    Initialize : find_slot = false, $r = l_i.receiver$, {r is the router to which the fragment $fr_m$ is sent.}
 1: while find_slot == false do
 2:     if $Slt(n)$ == null then
 3:         Insert $l_i$ on $Slt(n)$
 4:         find_slot = true
 5:         $en_r$ += $en_{r.m}$, $et_r$ += $et_{r.m}$
 6:         $n^* = n$
 7:         return $n^*$
 8:     else
 9:         for each path $l_j \in Slt(n)$ do
10:             Compute $I_{\mathrm{slt}}(Slt\{l_j\}, l_i)$ {the interference on $l_i$ by all paths $l_j$'s in time slot number $n$.}
11:             $I$ += $I_{\mathrm{slt}}$
12:             Update SINR value
13:             if $l_j.receiver$ == $r$ then
14:                 $Sum\_en_r$ += $en_{r.m^*}$, $Sum\_et_r$ += $et_{r.m^*}$, where $m \neq m^*$
15:             end if
16:         end for
17:         if SINR $\geqslant \beta$ then
18:             Insert $l_i$ on $Slt(n)$
19:             find_slot = true
20:             $en_r$ += $Sum\_en_r + en_{r.m}$, $et_r$ += $Sum\_et_r + et_{r.m}$
21:             $n^* = n$
22:             return $n^*$
23:         else
24:             Increment $n$
25:             Continue
26:         end if
27:     end if
28: end while
```

Pseudo-code of Algorithm 5 used to check the feasibility and determine the number of time slots of each fragment fr_m.

8.10 Trace Particles Using DPSO

In this section, we show an example execution trace for the DPSO as depicted by mapping the application graph in Figure 8.1 to the architecture graph in Figure 4.4. For ease of use, each index in a particle vector is a task, and the value of the associated index is a host number to which the corresponding task is scheduled. A population of 10 particles ($p1, p2, \dots, p10$) is initialized as shown in Table 8.7. Particle velocities are initialized randomly in a range of [-3, 3]. Table 8.7 also shows the best local position P_{best} for each particle and the best global position G_{best} in each iteration. The particle's cost, which is the cost of the task scheduling, is shown at the cost column of each particle. For example, in the first iteration, $p1$'s position [7, 17, 18, 15, 2] suggests a solution to the task scheduling problem by scheduling t_0 to host $h7$, t_1 to host $h17$, t_2 to host $h18$, t_3 to host $h15$ and t_4 to host $h2$. In this iteration, the particle $p4$ = [10, 4, 13, 15, 2] has the least cost function value (cost = 1107). Therefore, it will be considered as G_{best}. A copy of G_{best} is updated throughout all particles. At the beginning the historical

P_{best} of each particle is its initial position, DPSO then updates the particle's velocity using Eq. 8.18. The updated velocity is used in Eq. 8.19 to get a new position for each particle. For instance, the velocity vector [7,-8,-10, 0, 1] of particle $p1$ in the 2nd iteration is added to its position [7, 17, 18, 15, 2] in the 1st iteration to give a new position equal to [14, 9, 8, 15, 3]. If the corresponding host of a task in the new position is not considered as an available host as shown in Table 8.6, we map the host number to the closest number on which that task can run; thus, we obtain the new position of $p1$ to be [14, 12, 5, 15, 2] in the 2nd iteration.

G_{best} and P_{best} are constantly updated for all particles at every iteration. For example, P_{best} of each particle in the 2nd iteration is updated if the new position gives less cost (better solution) such as for particles $(p1, p2, p3, p4, p5, p6, p8, p10)$, whereas $p7$ and $p9$ particles keep their previous best local position found so far. The cost function value is updated (i.e. minimized) from 1107 in the 1st iteration to be 944 in the 6th iteration at $p3$'s position [14, 12, 13, 15, 9]. The iteration number is increased and the process is repeated until obtaining results or a termination condition is met such as the particles convergence into a unified solution. We conclude that the cost function value (944) is a steady-state from the 5th iteration until iteration # 32, where all particles have the same solution for both G_{best} and P_{best}. This solution represents the best global position found so far, thereby the corresponding velocity of each particle equals zero.

8.11 Simulation Setup to Evaluate OR-TMS

20 variable system models are randomly generated as random forest fire directed graphs. These graphs have different dependency patterns and they are used to evaluate the behaviour of the OR-TMS algorithm compared it to other suggested algorithms.

In our experimental setup, the system model in Figures 4.4 and 4.5 is used as input for the compared algorithms. Each architecture graph in the system model includes 10 wireless TSN-enabled intermediate nodes (i.e. wireless routers and access points). Every access point creates a local network with 4 static wireless TSN-enabled hosts with 50 Mbps data rate.

We assume a node that is out of energy as a failed node. Moreover, some simulation parameters are set as follows: $\beta = 10$, $\alpha = 0.02$ and $N = 0$. The weighting parameters of Eq. 8.17 (i.e. θ, γ and δ) are set as 0.1, 0.45 and 0.45, respectively. Choosing appropriate weighting parameters is important to obtain better results. However, throughout the task and message scheduling process, the weight of each parameter is determined depending on the significance of its related factor. Therefore, the failure rate and the makespan have been considered of greater importance because of dealing with reliable and real-time applications. The initial value of w in Eq. 8.18 is set to be 0.9, c_1 and c_2 equal to 2, r_1 and r_2 are randomly selected from a range of [0, 1]. $et_{r.m}$ and $en_{r.m}$ costs of routing a message m are determined from a range of [20, 25]. Similarly, $et_{h.t}$ and $en_{h.t}$ costs of executing a task t in a host h are determined from a range of [200, 250]. The full energy of any router or host is 100,000. In the setup, the units of measure of time and energy are 'millisecond' and 'joule', respectively. Moreover, an available set H_t, from which to select a host for a task t, is generated randomly and it is standardized for all algorithms that will be evaluated.

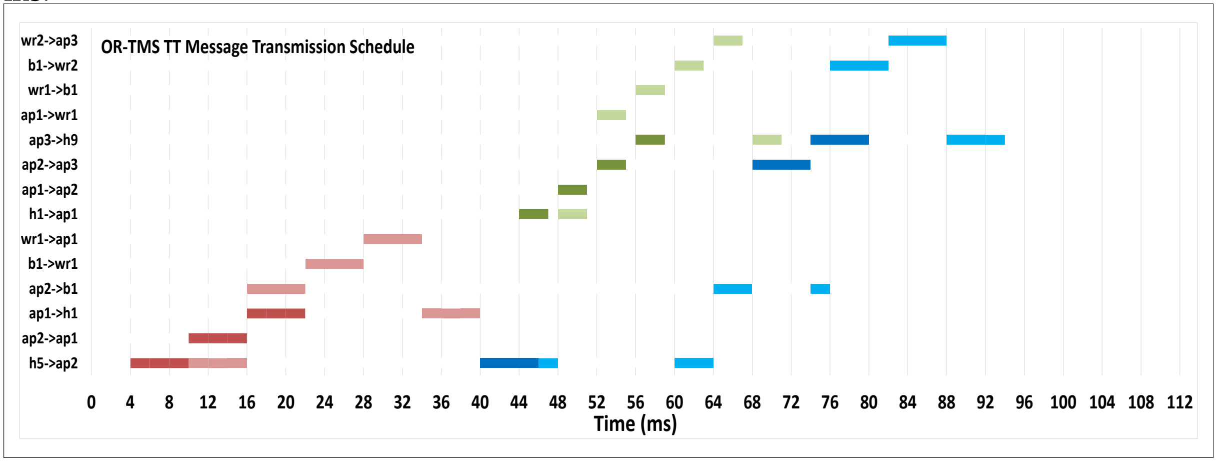
To evaluate the efficiency of OR-TMS, it is compared with our mentioned algorithm R-TMS in [29], Reliable Min-Min Task Scheduling algorithm (R-MMTS), and Reliable Shortest Task First Scheduling (R-STFS) algorithm [26]. All algorithms implement the physical interference model and FRER, which makes the comparison more fair. The main difference between OR-TMS and R-TMS lies in the fact that R-TMS does not support DPSO. In R-TMS, a host is selected for a listed task as soon as all the incoming messages, which are scheduled on time-slots and sent via redundant routes, have been received. R-TMS then moves to the next task until scheduling all tasks, while OR-TMS compares pre-selected solutions until the optimized solution is achieved by using DPSO. Meanwhile, to reveal the performance improvements gained by considering the communication cost as a result of time-slot message scheduling, we also compare OR-TMS with R-MMTS and R-STFS that do not consider the cost due to the message transmissions. R-MMTS is a traditional heuristic algorithm, which firstly computes the top-level of all tasks, and then it sorts the tasks depending on the task that has the minimum top-level as the priority. The algorithm then assigns the sorted tasks to the hosts that result in the shortest completion time, where the completion time is determined by adding the task execution time to the host's ready time. The same procedure is repeated for the next task until scheduling all tasks. R-STFS is a non-preemptive algorithm and assigns a task to a host based on the shortest execution time, where the host that finishes the task in less time gets the highest priority.

8.11.1 Message Scheduling of OR-TMS, R-TMS and R-MMTS Algorithms

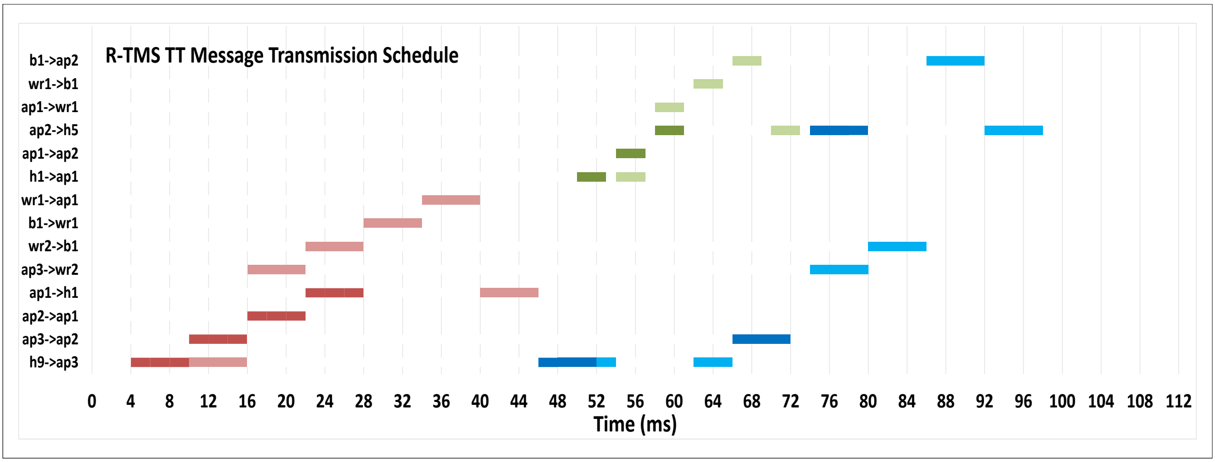
To show the message scheduling for three of the compared algorithms (i.e. OR-TMS, R-TMS and R-MMTS), we use the application graph mentioned in Figure 7.1. It is a simple graph that shows sending of only three messages m_{21}, m_{20} and m_{10} between three tasks (t_2, t_1 and t_0) so that we can show the scheduling of the messages on time slots in the simplest way possible.

Figures 8.13a, 8.13b, and **??** show TT message scheduling over their disjoint redundant routes for OR-TMS, R-WTA and R-MMTS algorithms, respectively. For example, Figure 8.13a shows that OR-TMS sends two replicas of m_{21} over two redundant routes: the first route = $\{h5, ap2, ap1, h1\}$ is used to send the first replica indicated by dark red color and the second route = $\{h5, ap2, b1, wr1, ap1, h1\}$ is used to send the second replica indicated by light red color. Messages m_{20} and m_{10} are sent over their redundant routes in the same way for all algorithms. We conclude that the replicas can be sent in parallel or even with other messages at the same time, provided that the induced interference does not affect the received signal strength (i.e. SINR) and makes it less than a threshold value (i.e. β). For example in OR-TMS, the first replica of message m_{20} (dark blue message) is sent over its path ($h5$ -> $ap2$) simultaneously with sending the first replica of message m_{10} (dark green message) that uses its path ($h1$ -> $ap1$) in the time slot $44 - 46$. The second replica of message m_{20} (light blue message) sends its first fragment in the time slot $46 - 48$ but it delays the rest of the fragments until time slots $60 - 64$ because of the induced interference by sending the first and the second replicas of message m_{10} making the time slots $48 - 60$ infeasible for sending message m_{20}. Subsection 7.4.1 also gives a more illustrative example that shows the scheduling of messages on time slots using the physical interference model.

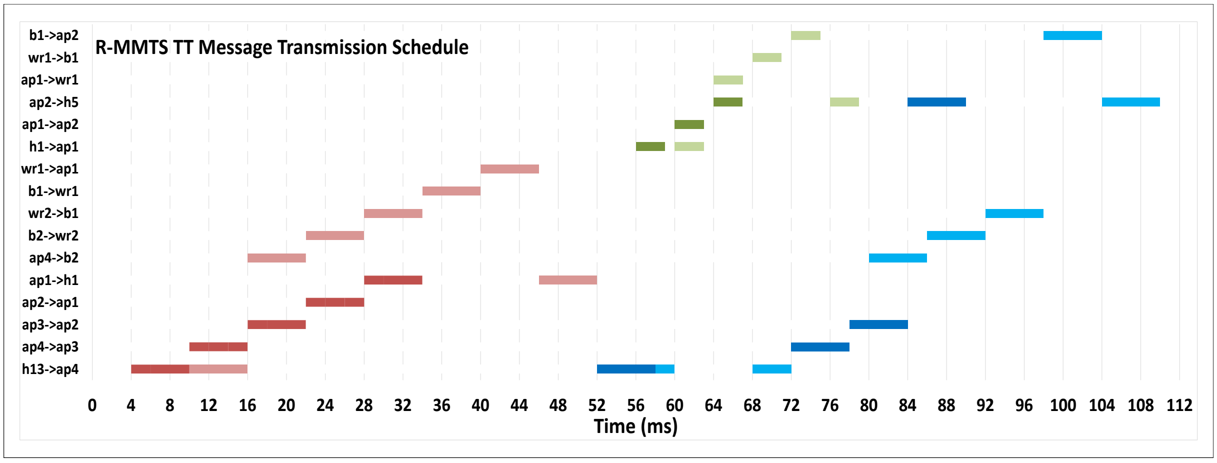
Sending replicas through redundant routes increases the reliability in the event of faults, but this causes an increase in the time required to schedule messages for all algorithms. Based on the mechanism of scheduling tasks and messages in R-TMS, it initially schedules task t_2 on host $h9$. The best hosts to schedule tasks t_1 and t_0 based on the previous task (i.e. t_2) and according to Eq. 8.6 are $h1$ and $h5$, respectively. If we assume that OR-TMS uses the same hosts to schedule the same tasks, we notice the ability of OR-TMS to take an overview of all tasks and find out the best host for each task based on Eq. 8.17. We conclude that only by rearranging the scheduling of the tasks on the hosts, we will find out that OR-TMS finds a better solution than R-TMS. The solution of OR-TMS results in scheduling tasks t_2, t_1 and t_0 on hosts $h5$, $h1$ and $h9$. The task and message scheduling solution of OR-TMS decreases the message scheduling time to be **94 ms** compared to **98ms** in the R-TMS algorithm. Finding more efficient solutions using OR-TMS is made possible by the growth in the number of tasks and the growth in the number of related messages in a more sophisticated system model. R-MMTS schedules tasks on hosts that have a lower completion time, while the cost of sending messages is not taken into account. We conclude that R-MMTS schedules t_2, t_1 and t_0 tasks on $h13$, $h1$ and $h5$ hosts, thus, it increases the message scheduling time to be **110 ms**.



(A) TT message transmission schedule of OR-TMS.



(B) TT message transmission schedule of R-TMS.



(C) TT message transmission schedule of R-MMTS.

FIGURE 8.13: TT message scheduling along dedicated time slots for OR-TMS, R-TMS and R-MMTS algorithms.

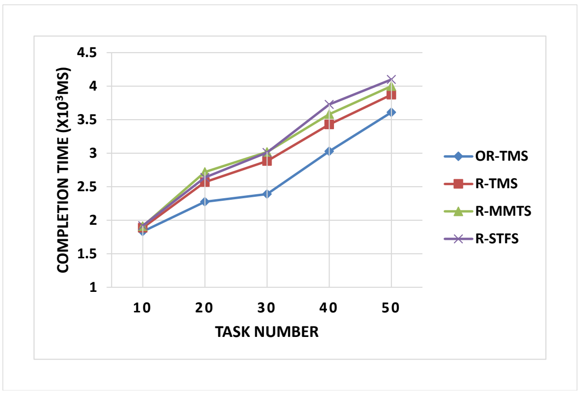
8.11.2 Experimental Results

Impact of Network Topology on Performance

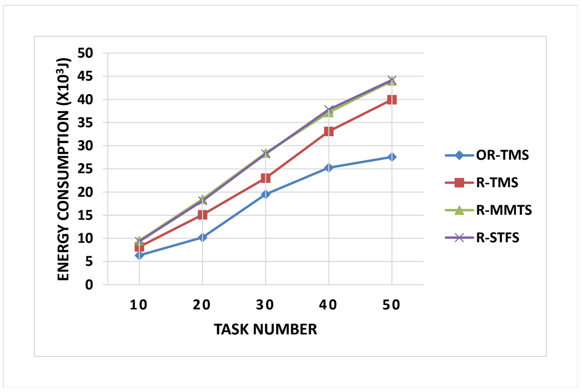
Firstly, we carry out a set of experiments to observe the impact of the network topology and the number of tasks on OR-TMS, R-TMS, R-MMTS and R-STFS. Figures 8.14 and 8.15 show the impact of increasing the number of tasks in the grid (see Figure 4.4) and ring (see Figure 4.5) topologies. The task number is increased from 10 to 50 tasks with a step size of 10.

The deadline miss ratio, task completion time, network energy consumption, failure rate, and cost (the lower the better) are growing for all compared algorithms due to the restricted network resources and node availability that results from an increase in task numbers. Relatively, the results in the grid topology are better than in the ring topology because the grid contains more routes that allow to reduce the congestion in some routes and make a balance in the distribution of the messages. In the grid topology, the messages can also be received more quickly using short routes, and multiple messages can be transmitted simultaneously. Additionally, a failure doesn't stop the transmission of the message.

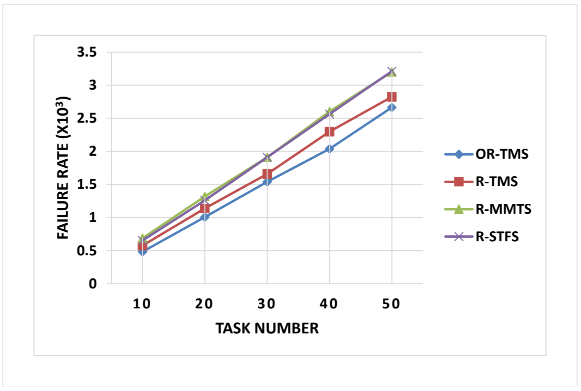
OR-TMS shows the best results because it continuously updates the best solution by updating the best local position gained by each particle, and the best global position gained by all particles. The updating of each particle's position depends mainly on the cost function value that results from scheduling all tasks. Moreover, OR-TMS applies a load balance mechanism which distributes a set of tasks over a set of hosts, with the aim of making the task and message scheduling process more efficient. Load balancing can reduce completion time, energy consumption, and avoid unevenly overloading some nodes while leaving other nodes idle. The same cost function is also used in R-TMS but in a different manner. The function in R-TMS is used by every task gradually to select a suitable host until finishing all tasks. R-TMS does not compare and make a comprehensive evaluation for different solutions as in OR-TMS. Moreover, the load balancing is not applied in R-TMS. Thus, the effectiveness of OR-TMS appears with the increase in the number of tasks. On the contrary, R-MMTS and R-STFS do not support a comprehensive evaluation during the task scheduling process regarding the energy, message routing, message arrival time, and workload balancing. R-MMTS selects the host that has the minimum completion time to execute a task, while R-STFS selects the shortest process for task execution. Therefore, R-MMTS and R-STFS cause unbalanced load in some nodes, which consumes more energy and causes a higher overall failure rates. They also do not benefit from the message routing during the message scheduling process. Figure 8.14 illustrates that R-MMTS and R-STFS have mostly similar performance in the grid topology, whereas R-MMTS deviates in producing less completion time than R-STFS when the task number increases. Figure 8.15 shows that when increasing the number of tasks in the ring topology, R-MMTS cannot schedule tasks appropriately, and the completion time, energy consumption, failure rates and deadline miss ratio of the system gets relatively high. As a result, as traffic loads increase, it performs the worst.



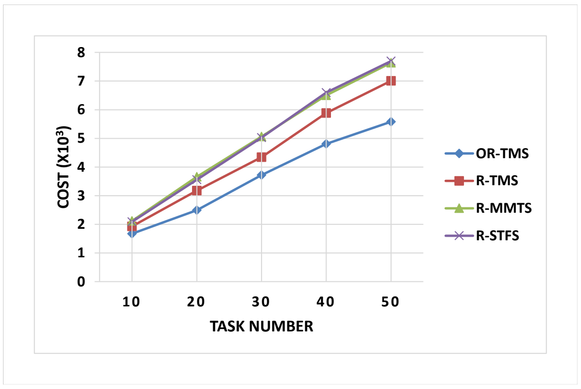
(A) System's completion time.



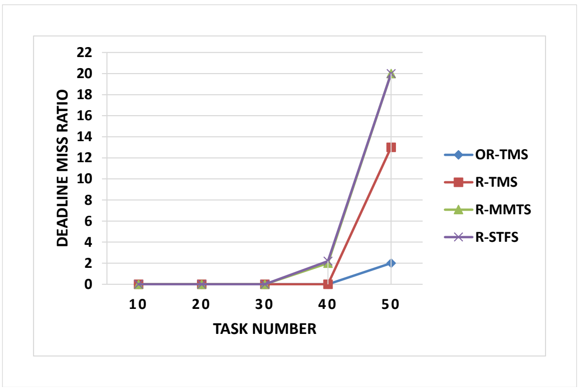
(B) Network energy consumption.



(C) Failure rate.

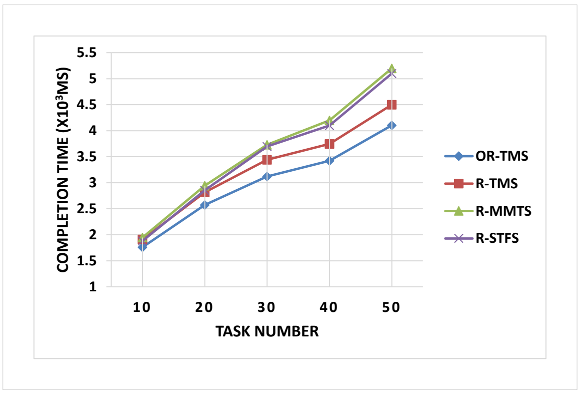


(D) Cost.

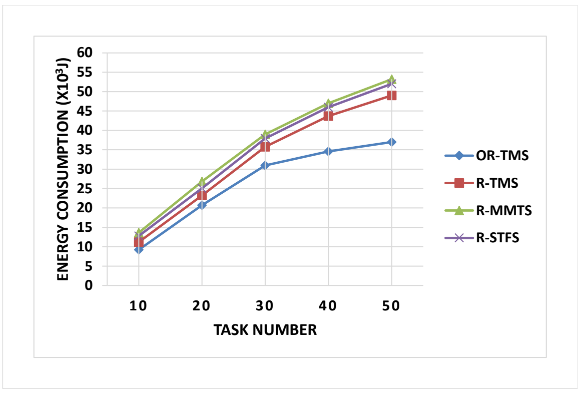


(E) Deadline miss ratio.

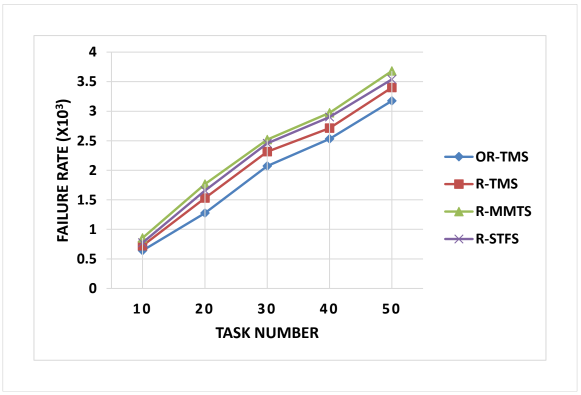
FIGURE 8.14: Impact of the grid topology on performance of OR-TMS, R-TMS, R-STFS and R-MMTS algorithms.



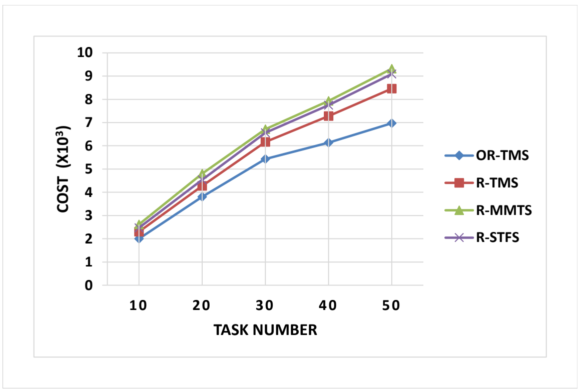
(A) System's completion time.



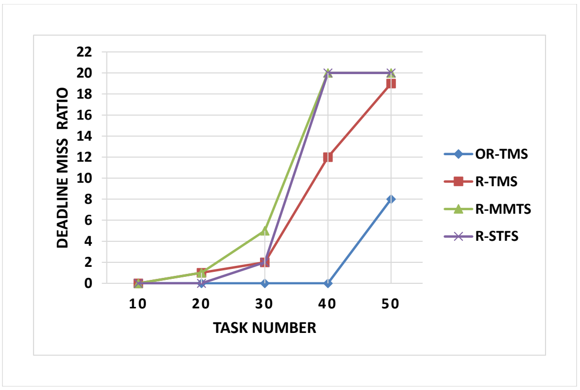
(B) Network energy consumption.



(C) Failure rate.



(D) Cost.



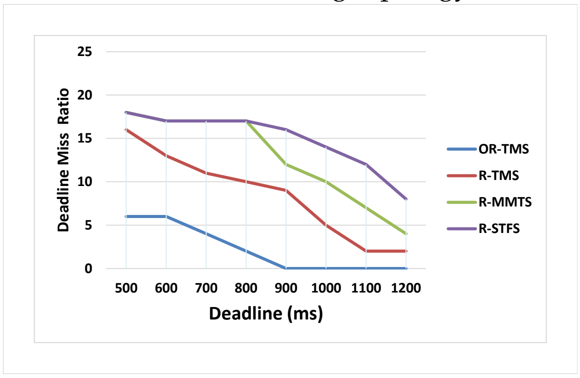
(E) Deadline miss ratio.

FIGURE 8.15: Impact of the ring topology on performance of OR-TMS, R-TMS, R-STFS and R-MMTS algorithms.

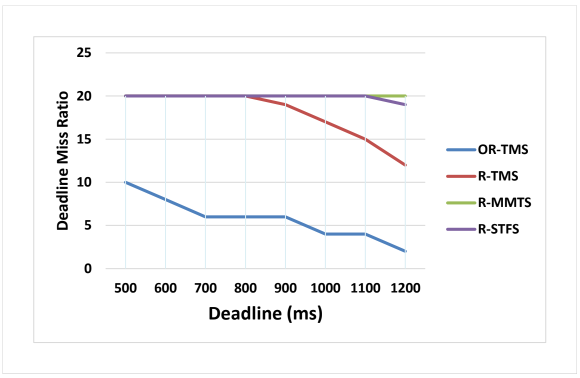
Impact of Deadline on the Task Scheduling Miss Ratio

Secondly, we carry out experiments as shown in Figure 8.16 to observe the impact of the task deadline. Here, the number of tasks is set as 30, the task deadline is increased from 500 ms to 1200 ms. Figure 8.16 shows that the deadline miss ratio of the compared algorithms is dropping when increasing the task deadline. OR-TMS adopts an optimized solution to execute the tasks and it makes a comprehensive evaluation of the proposed solutions. If one of the tasks in that solution misses its deadline, then this solution will be considered as an unsuccessful task scheduling and it is skipped when generating the best global solution. R-TMS adopts the miss deadline condition, but this condition is applied for each task individually. This means if a task in R-TMS is scheduled to a host, this scheduling may lead to an increase in the time required for the outgoing messages to arrive at the next dependent task, thus, the next task may miss its deadline. R-MMTS and R-STFS work without considering the deadline condition. Hence, OR-TMS shows the least deadline-miss ratio, which is 11.25% and 35% in the grid and ring topologies, respectively. R-TMS misses the deadlines by 42.5% and 89.37% in the grid and ring topologies, respectively. Similarly, R-MMTS misses the deadlines by 63.75% in the grid topology and 100% in the ring topology. R-STFS misses the deadlines by 74.73% and 99.3% in the grid and ring topologies, respectively. We observe better results in the grid topology than in the ring topology for all compared algorithms, because the grid topology has more routes to forward messages, which leads to a reduced possibility of miss the deadlines.

Figure 8.16 shows that the lower the deadline, the more difficult it is for all algorithms to find solutions that finish all tasks before the deadlines. We recognize the significance of considering the wireless TSN's limitations on simultaneous messaging or power consumption. The difficulty of finding solutions increases with the increase in the number of tasks and messages in a network with limited resources. However, we conclude that OR-TMS is able to find solutions for the 20 tested system models, despite the low task deadline. On the other hand, R-MMTS, R-STFS and R-TMS are unable to find solutions with a deadline of less than 800 ms in the ring topology.



(A) Grid topology.



(B) Ring topology.

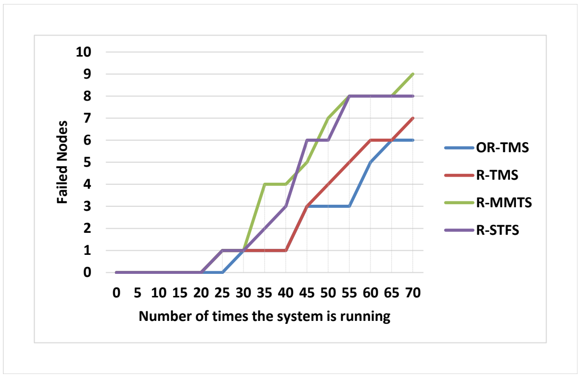
FIGURE 8.16: Impact of deadline on the task scheduling miss ratio of OR-TMS, R-TMS, R-STFS and R-MMTS algorithms.

Impact of the Runtime of the System on the Network Lifetime

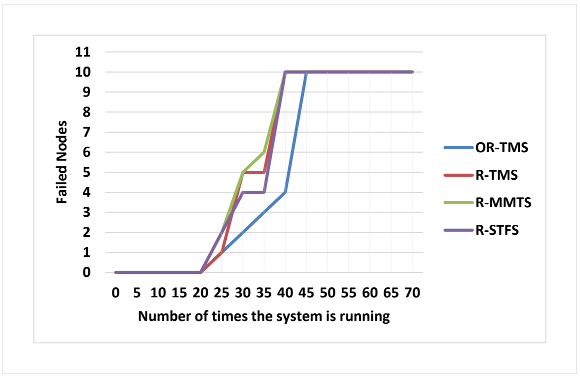
In order to study the impact of the number of times the system is running on the network lifetime, 30 tasks with 2000 ms deadline are tested from 0 to 70 times. The algorithms are used in grid and ring networks.

Figure 8.17 shows the number of failed nodes when increasing the number of times the system is running (after scheduling). When a node runs out of all energy, it is deemed to have failed. We observe that OR-TMS has the longest network lifetime because the number of the failed nodes is the least compared with the other algorithms. Similar to the previous experiments, OR-TMS consumes the least amount of energy because the messages are transmitted on the shortest routes, so it shows the best results. Moreover, the hosts, which are chosen to schedule tasks in the selected solution, are scheduled in a way that minimizes the resource consumption of the traversing messages.

In OR-TMS, the average failed nodes (i.e. the percentage of nodes that failed during an experiment) are 3.76% and 9.98% in the grid and ring topologies, respectively. The average failed nodes for R-TMS are 4.68% and 12.17% in the grid and ring topologies, respectively. Likewise, the average failed nodes for R-MMTS are 7.62% and 12.38% in the grid and ring topologies, respectively. For R-STFS, the average failed nodes are 6.95% and 11.73%. In the ring topology, all algorithms resulted in more failed nodes compared with the grid topology because the ring networks require more energy to schedule all tasks.



(A) Grid topology.



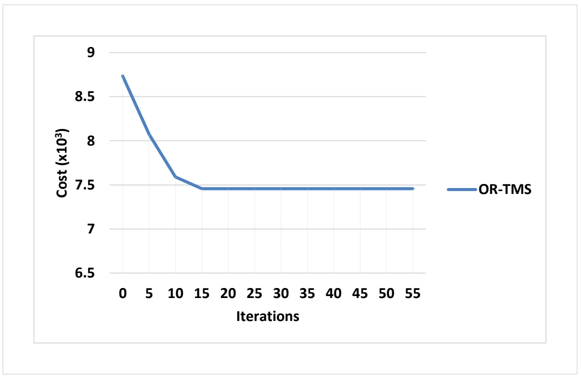
(B) Ring topology.

FIGURE 8.17: Impact of the number of running times on the network lifetime using OR-TMS, R-TMS, R-STFS and R-MMTS algorithms.

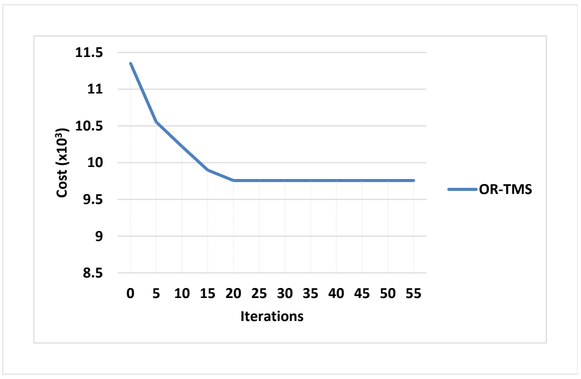
Convergence Rate of OR-TMS

Finally, the convergence rate of OR-TMS is investigated. The number of iterations the algorithm requires to progress toward a converged and uniformly optimized solution is known as the convergence rate. Therefore, 30, 40, and 50 tasks with 2000 ms deadline for each task are tested. The step size of the experiments is set as 5.

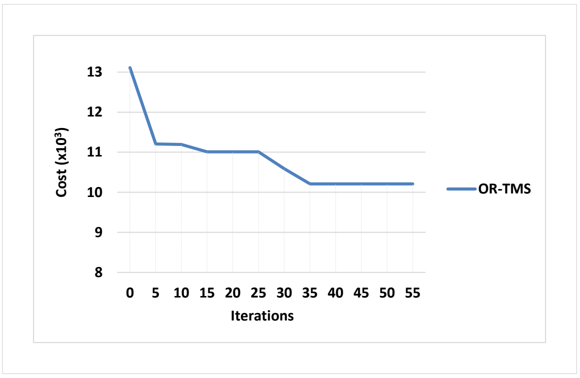
As shown in Figure 8.18 with a population of 10 particles, OR-TMS begins to converge after 15, 20, and 35 times to schedule 30, 40, and 50 tasks, respectively. We conclude that OR-TMS needs more iterations in an ascending trend when increasing the number of tasks that are required to be scheduled, besides more iterations as the size of the population increases.



(A) Tasks = 30.



(B) Tasks = 40.



(C) Tasks = 50.

FIGURE 8.18: Convergence rate of OR-TMS.

8.12 OR-TMS Based Metascheduler

In time-triggered systems, adaptation is performed using the metascheduling, which is used to ensure temporal predictability and fault tolerance by activating another schedule for fault recovery. Metascheduling approaches to *WirelessTSN* present difficulties in estimating and storing the resulting scheduling at design time. A large number of schedules causes a state space explosion problem that can be managed using a reconvergence method. In this section, a metascheduler is proposed to solve a new scheduling problem for each adaptation scenario using OR-TMS, resulting in a Multi-Schedule Graph (MSG) that combines the repeated schedules. The proposed metascheduler computes the MSG using the input models at design time. It repeatedly invokes OR-TMS to solve a scheduling problem constructed by the metascheduler. The resulting schedule is inserted into the MSG.

The hybrid modelling approach mentioned in Chapter 4 is given to the OR-TMS based algorithm, which generates a schedule that satisfies resource limitations and precedence constraints. The preference for choosing the schedule depends on quality metrics and input parameters that include the energy consumption, failure rates, and the time when all tasks are finished.

8.12.1 Metascheduler with Reconvergence of Repeated Schedules

The suggested metascheduler provides schedule adaptation in response to each failure event $fr \in Fr$. For example, as shown in Figure 8.19, for the case that a single-failure (e.g. fr_{n1}) or a two-failure (e.g. fr_{n1l1}) event occurs, a new schedule is generated to allow adapting to the failure event. The metascheduler calculates these new schedules, resulting in a MSG covering the occurrence of the specified failure events. The output MSG is used to determine the possibility of switching schedules in the event of a failure event during the operation of the *WirelessTSN*.

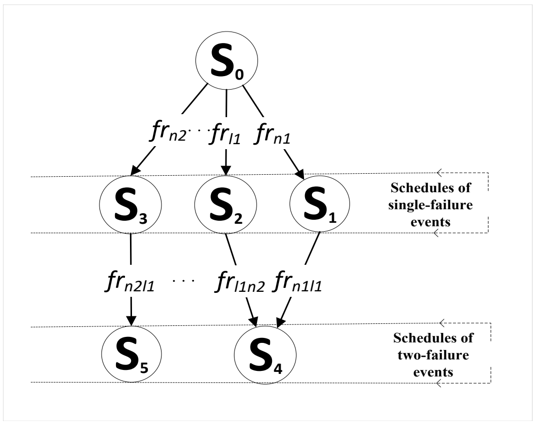


FIGURE 8.19: An example of a MSG with reconvergence of repeated schedules.

The metascheduler in Figure 8.20 is used to build an MSG by continually invoking the OR-TMS. To begin, the original hybrid system model (i.e. G_{Arc} and G_{App}) is utilized to create a base schedule S_0, which is then added as the MSG's root node. A set of failure events is calculated using the formula in Equation 8.23, where ${}^{Co}Comb_r$ is the number of failed component combinations, r is the number of failed components, and Co is the total number of components in the network, where a component represents a wireless node or link. For example, if the total number of components in the *WirelessTSN* is 20 (i.e., the total links and nodes are 20), and the number of failed components is 1 (i.e., r = 1), the number of failed component combinations is $\frac{20!}{1!(20-1)!}$ = 20. In the same way, with r = 2, the number of failed component combinations is 190. Two sets (Set_1 and Set_2) are built in this work, one for one failed component and the other for two failed component combinations. The existing sets are merged into a single event set to index all combinations beginning with Set_1, which was created from S_0, where all events at Set_1 are defined from S_0.

$${}^{Co}Comb_r = \frac{Co!}{r!(Co-r)!} \tag{8.23}$$

The initial failure event is picked and deleted from the set in the metascheduling process, and it is used to change the original G_{Arc} (i.e. G'_{Arc}) in the case of a failed link fr_l or a failed node event fr_n. The modified system model m' is then sent to OR-TMS to solve the new scheduling problem.

The metascheduler checks if the computed schedule S' is valid. The schedule is considered valid if all tasks are finished by their deadlines; otherwise, it is deemed invalid. The metascheduler calls the next event in the set if the schedule is invalid.

Each valid schedule is compared to the MSG's existing schedules. If the MSG contains a repeated schedule, the schedules are converged, and a new transition (i.e. edge) is constructed from the previous schedule (i.e. previous state). If the schedule does not exist, a new transition is produced from the previous schedule and it is added to the MSG. The next event is called by the metscheduler, and the metscheduling procedure is repeated. The metascheduler begins calling the first of the two-failure combination sets after processing all failures in the single-failure combination set. After calling all events, the MSG is completely formulated, and the number of valid, invalid, and stored schedules is calculated. Thus, converged schedules that exist on multiple schedule tree routes merge the transitions and then reduce the size of the MSG.

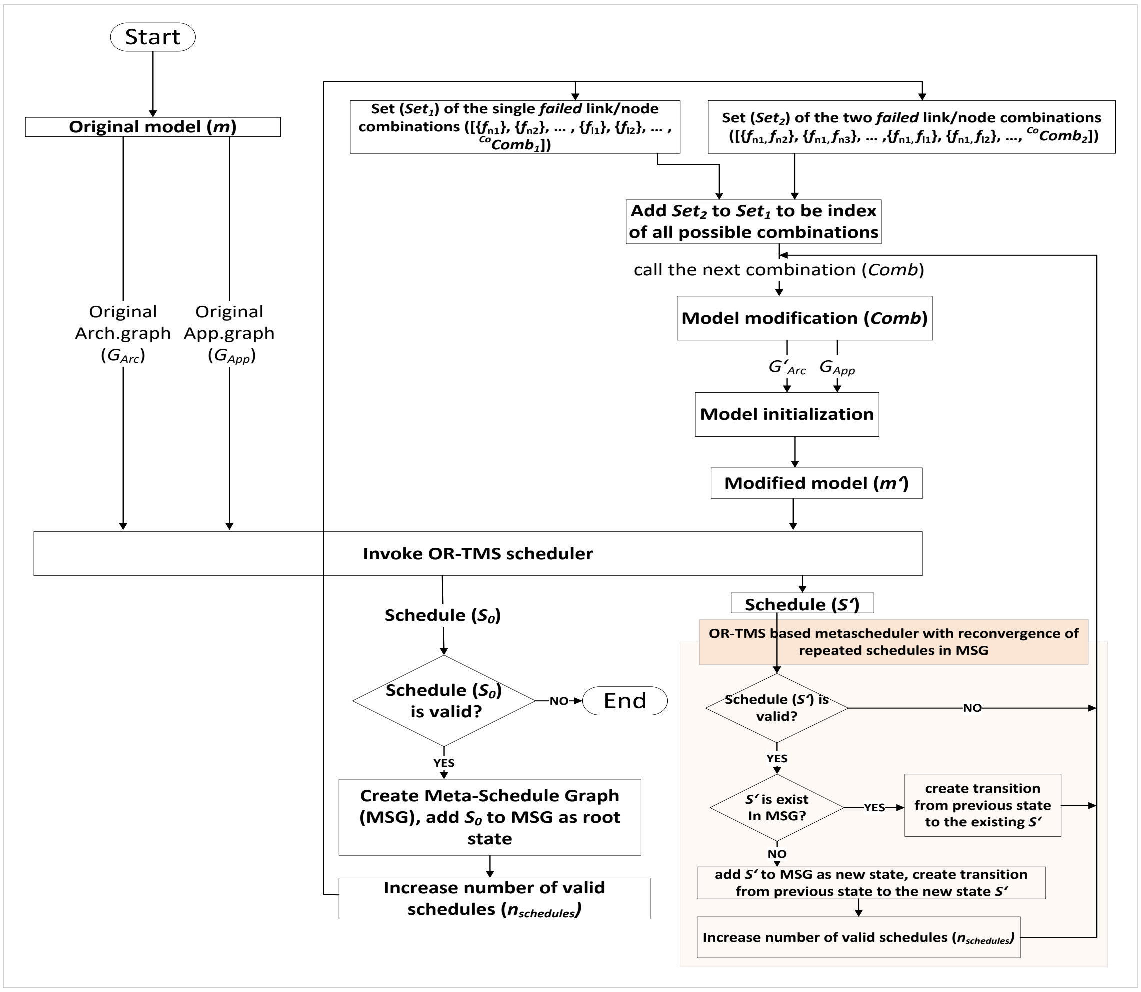

Start
Original model (m)
Original Arch.graph (G_{Arc})
Original App.graph (G_{App})
Set (Set_1) of the single failed link/node combinations ([{f_{n1}}, {f_{n2}}, … , {f_{l1}}, {f_{l2}}, … , ${}^{Co}Comb_1$])
Set (Set_2) of the two failed link/node combinations ([{f_{n1},f_{n2}}, {f_{n1},f_{n3}}, … ,{f_{n1},f_{l1}}, {f_{n1},f_{l2}}, …, ${}^{Co}Comb_2$])
Add Set_2 to Set_1 to be index of all possible combinations
call the next combination (Comb)
Model modification (Comb)
G'_{Arc} G_{App}
Model initialization
Modified model (m')
Invoke OR-TMS scheduler
Schedule (S_0)
Schedule (S_0) is valid?
NO
End
YES
Create Meta-Schedule Graph (MSG), add S_0 to MSG as root state
Increase number of valid schedules ($n_{schedules}$)
Schedule (S')
OR-TMS based metascheduler with reconvergence of repeated schedules in MSG
Schedule (S') is valid?
NO
YES
S' is exist In MSG?
YES
create transition from previous state to the existing S'
NO
add S' to MSG as new state, create transition from previous state to the new state S'
Increase number of valid schedules ($n_{schedules}$)

FIGURE 8.20: Flowchart of the OR-TMS based metascheduler.

8.12.2 Evaluation of the Proposed Metascheduler

The grid and ring network topologies in Figures 4.4 and 4.5 are applied in our experimental setup. The weighting parameters (θ, γ and δ) of Eq. 8.17 are set to 0.1, 0.45, and 0.45, respectively to give more importance to *makespan* when finding a schedule. The TT task deadline is 1500 ms, and the task periods are chosen at random from 100 and 200 ms. The communication message size is either 100 or 200 bytes, and the message takes either 3 or 6 ms to transmit. The task execution time is between [200, 250] ms, and the message routing time is between [20, 25] ms. The OR-TMS based metascheduler is compared against metaschedulers based on R-TMS, R-MMTS, and R-STFS algorithms to determine its efficiency.

Evaluation of the MSG size reduction in relation to increasing task numbers

This subsection shows the efficiency of the proposed OR-TMS based metascheduler in reconvergence of repeated schedules in the MSG. In the ring and grid topologies with a 1500 ms deadline, Figure 8.21 demonstrates the ratio of reducing the size of the MSG when increasing the number of tasks. To show how MSG removes repeated schedules, the reduction ratio is defined in Equation 8.24.

$$MSG\ size\ reduction\ ratio = \frac{1 - \ MSG\ size}{number\ of\ valid\ schedules} x100\% \tag{8.24}$$

When two failure events are evaluated, the reduction ratio is larger than when single failure events are considered for both topologies. In the case of two-failure events, the ratio jumped to 46% compared to single-failure events in the grid topology. Similarly, in the ring topology, the ratio increased to 38%. The reason for this is that as the number of failed component combinations grows, so does the size of the MSG. While the reduction ratio in the ring topology is greater than that in the grid topology, this is because of the structure of the rings and the reduced number of available paths as compared to the grids, which leads to the similarity of many valid schedules even in the presence of failures.

Figure 8.21 clearly demonstrates that when the number of tasks increases, the MSG size reduction ratio decreases. The reason for this is that as the number of tasks to be scheduled grows, it becomes increasingly difficult to create effective schedules.

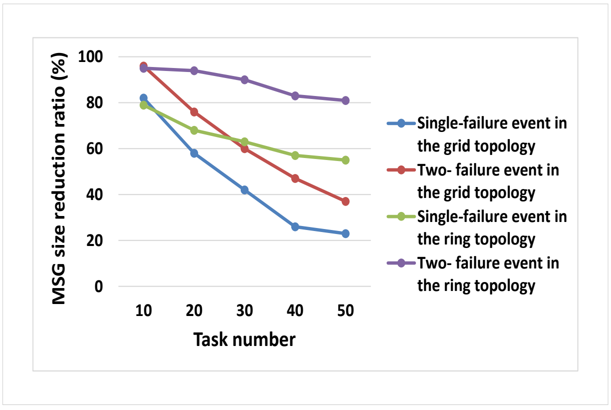


FIGURE 8.21: MSG size reduction ratio in relation to increasing task numbers.

Evaluation of the MSG size reduction in relation to increasing deadline values

For 30 tasks distributed in the ring and grid topologies, the MSG size reduction ratio is shown in Table 8.8 against the task deadlines in milliseconds. All tasks of the application model have the same deadline, which varies between 1100 and 1900 ms. The size of the MSG is greatly reduced when the ring and grid topology are re-converged for single and two-failure events. The MSG size is reduced while still keeping a valid schedule.

When the task deadline is fixed, the lowered ratio for two-failure events is found to be larger than for single-failure events. Similarly, ring topologies have a larger ratio than grid topologies. It is assumed that as deadlines increase, valid schedules increase as well, raising the MSG size reduction ratio. However, if an increase in deadlines leads in a considerable increase in MSG size compared to an increase in valid schedules, the MSG size reduction ratio will be smaller.

TABLE 8.8: MSG size reduction ratio in relation to increasing deadline values.

MSG size reduction ratio (%)	1100	1300	1500	1700	1900
Single-failures in grid	50%	50%	70%	67%	53%
Two-failures in grid	59%	65%	73%	72%	75%
Single-failures in ring	94%	84%	46%	65%	74%
Two-failures in ring	98%	98%	93%	94%	96%

The effect of task load on validity

We ran a series of tests to see how task load and network topology affect the metascheduler, which is based on OR-TMS, R-TMS, R-MMTS, and R-STFS. The validity of the grid and ring topologies in Figure 8.22 is affected by the number of tasks, where the validity formula is defined in Equation 8.25. The number of tasks has been increased from 10 to 50, with a step size of 10 and a deadline of 1500 ms.

$$Validity = \frac{number\ of\ valid\ schedules}{number\ of\ valid\ and\ invalid\ schedules} \tag{8.25}$$

Due to limited network resources and host availability, increasing the number of tasks reduces the task's capacity to be completed before its deadline, resulting in an increase in the number of invalid schedules and a decrease in the validity of all compared algorithms.

The task scheduling process to hosts is complex and depends on the number of messages flowing from the parent tasks. Moreover, differences in the structure of application models may affect communication costs by scheduling more than one task on the same host, which may lead to higher validity with a larger number of tasks. The R-MMTS in Figure 8.22a, for example, is observed to schedule 50 tasks with higher validity than 40 tasks. A similar behavior can be observed for R-TMS in Figure 8.22d.

When single-failure events are present, the OR-TMS based metascheduler outperforms those based on R-STFS, R-MMTS, and R-TMS by 44%, 49%, and 57%, respectively, in the grid topology scenarios. With two-failure events, it also demonstrates better validity than R-STFS, R-MMTS, and R-TMS by 40%, 41%, and 56%, respectively.

In the ring topology when single-failure events are present, a OR-TMS based metascheduler outperforms those based on R-STFS, R-MMTS, and R-TMS by 32%, 14%, and 24%, respectively. It also outperforms R-STFS, R-MMTS, and R-TMS in terms of validity, by 7%, 4%, and 18%, respectively, with two-failure events.

Because it continuously updates the solution by updating the best local position achieved by each particle and the best global position gained by all particles, the metascheduler based on OR-TMS produces better results. A similar cost algorithm is utilized in R-TMS, but it selects an appropriate host until all of the tasks are completed. R-TMS does not assess and evaluate different solutions in the same way that OR-TMS does. R-MMTS and R-STFS do not profit from message routing during the message scheduling process and cannot provide the individual advantage of nodes.

Due to its efficiency in regulating parameters and quickness in discovering solutions, the metascheduler based on OR-TMS provides high validity stability with one or two failure events in the ring or grid topologies besides its overall effectiveness. Although the metascheduler based on OR-TMS in Figure 8.22 has a smaller advantage in the ring topology than in the grid topology, this is because there are fewer valid solutions, particularly when there are several failures.

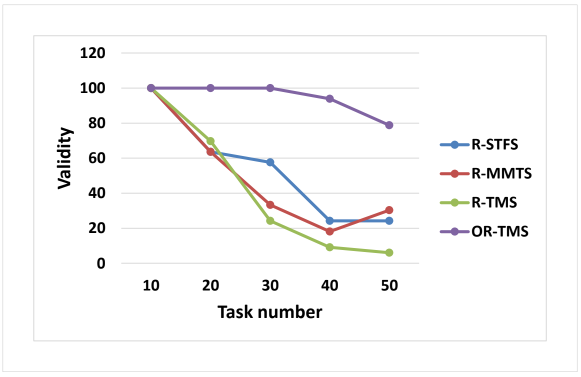
The effect of increasing deadline values on validity

In the grid and ring topologies, increasing the value of deadlines in Figure 8.23 has an impact on the validity. The number of tasks has been increased to 30, and the deadline has been extended from 1100 to 1900 ms with a 200 ms step size.

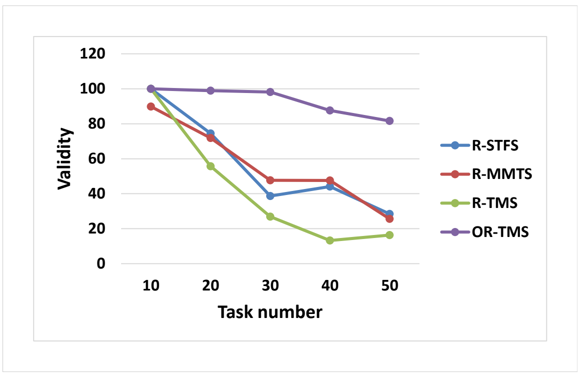
When single-failure events are present, the OR-TMS based metascheduler outperforms those based on R-STFS, R-MMTS, and R-TMS by 76%, 62%, and 62%, respectively, in the grid topology. With two-failure incidents, it also demonstrates 70%, 47%, and 65% better validity than R-STFS, R-MMTS, and R-TMS, respectively.

In the ring topology when single-failure events are present, the OR-TMS based metascheduler has better validity than those based on R-STFS, R-MMTS, and R-TMS by 38%, 22%, and 1%, respectively. With two-failure events, it also outperforms R-STFS and R-TMS by 6% and 4%, respectively, in terms of validity. When the deadline starts at 1500 ms, OR-TMS shows better validity than R-MMTS.

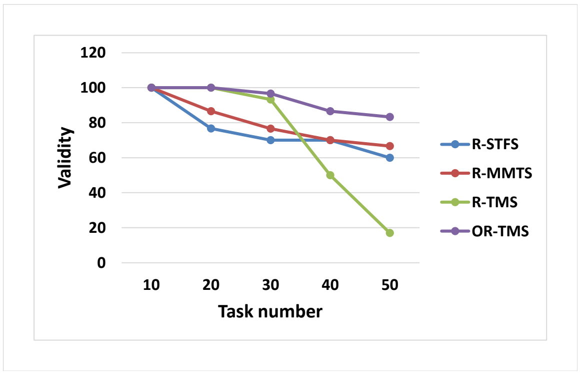
In the metascheduler based on OR-TMS, if a task in a solution misses its deadline, that solution is considered an unsuccessful task assignment and is skipped when the global best solution is constructed. The miss deadline clause is adopted by the R-TMS, although it applies to each task separately. This means that scheduling a task to a host in R-TMS may lengthen the time required for outgoing communications to reach the following child task, causing the next task to miss its deadline. The R-MMTS and R-STFS work without regard for the requirement of a deadline. As a result, despite the short deadlines for the tasks, we conclude that OR-TMS is capable of finding solutions.



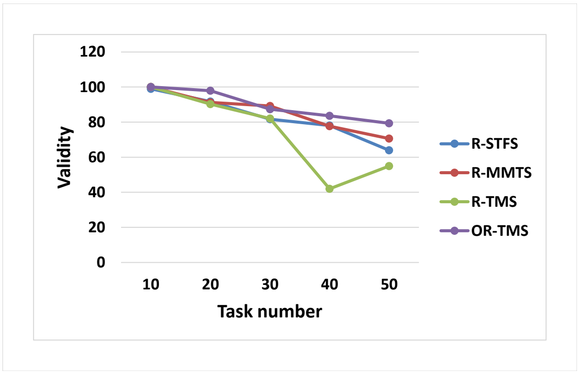
(A) Single failure/Gird topology.



(B) Two failure/Gird topology.

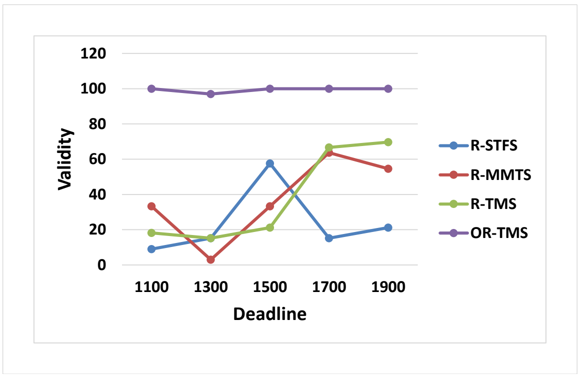


(C) Single failure/Ring topology.

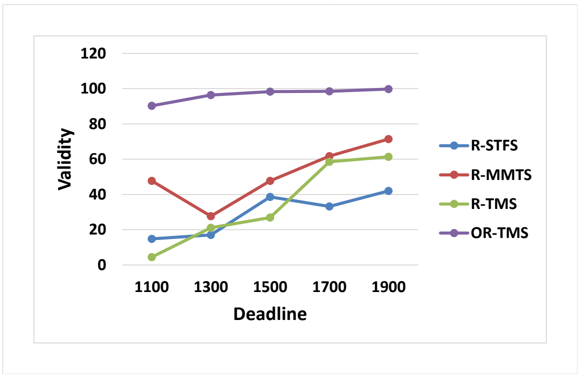


(D) Two failure/Ring topology.

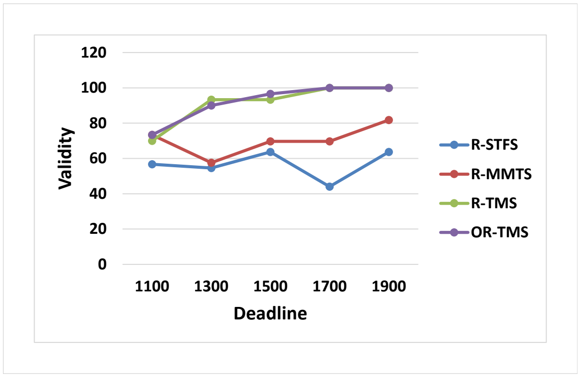
FIGURE 8.22: Effect of increasing task numbers on validity.



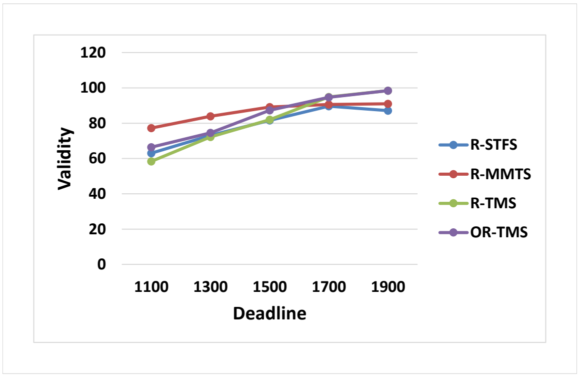
(A) Single failure/Gird topology.



(B) Two failure/Gird topology.



(C) Single failure/Ring topology.



(D) Two failure/Ring topology.

FIGURE 8.23: Effect of increasing deadline values on validity.

TABLE 8.7: A trace of DPSO-based OR-TMS algorithm

Iteration 1	Position					Best Local Position					Velocity					Cost
$p1$	7	17	18	15	2	7	17	18	15	2	3	-2	-3	-1	2	1231
$p2$	10	17	13	18	1	10	17	13	18	1	0	-2	-2	-1	-3	1210
$p3$	6	12	13	15	1	6	12	13	15	1	2	-1	1	0	-1	1120
$p4$	**10**	**4**	**13**	**15**	**2**	10	4	13	15	2	-2	-3	-2	3	-2	**1107**
$p5$	8	14	18	4	1	8	14	18	4	1	2	1	0	2	-2	1231
$p6$	8	14	20	6	9	8	14	20	6	9	3	3	2	-3	-3	1176
$p7$	6	14	5	15	1	6	14	5	15	1	0	-2	-1	3	-1	1167
$p8$	10	17	5	6	1	10	17	5	6	1	2	3	0	-3	-1	1170
$p9$	14	4	5	6	1	14	4	5	6	1	-2	-3	0	2	-2	1124
$p10$	6	17	20	15	1	6	17	20	15	1	0	2	-2	-2	-3	1233

Best Global Position at Iteration 1 = $\{10, 4, 13, 15, 2\}$

Iteration 2	Position					Best Local Position					Velocity					Cost
$p1$	14	12	5	15	2	14	12	5	15	2	7	-8	-10	0	1	1141
$p2$	10	12	13	15	1	10	12	13	15	1	0	-8	-1	-5	0	1046
$p3$	**14**	**12**	**13**	**15**	**1**	14	12	13	15	1	6	-2	-1	0	0	**1044**
$p4$	10	4	13	15	2	10	4	13	15	2	1	0	0	0	0	1107
$p5$	8	12	5	4	1	8	12	5	4	1	1	-3	-9	0	0	1068
$p6$	14	12	5	15	9	14	12	5	15	9	5	-3	-11	7	-1	1063
$p7$	8	4	20	15	1	6	14	5	15	1	2	-10	14	-2	0	1222
$p8$	10	12	18	15	1	10	12	18	15	1	0	-8	11	7	0	1144
$p9$	10	4	18	15	1	14	4	5	6	1	-4	2	12	11	0	1222
$p10$	10	12	5	15	1	10	12	5	15	1	4	-8	-11	1	0	1114

Best Global Position at Iteration 2 = $\{14, 12, 13, 15, 1\}$

Iteration 3	Position					Best Local Position					Velocity					Cost
$p1$	14	4	5	15	2	14	12	5	15	2	0	-7	0	0	0	1162
$p2$	14	4	13	10	1	10	12	13	15	1	3	-7	0	-4	0	1154
$p3$	14	12	13	15	1	14	12	13	15	1	0	-1	0	0	0	1044
$p4$	**14**	**12**	**13**	**15**	**2**	14	12	13	15	2	4	6	0	0	0	**1043**
$p5$	14	12	5	15	1	8	12	5	4	1	5	-2	0	9	0	1143
$p6$	14	12	5	15	2	14	12	5	15	9	0	-2	0	0	-7	1141
$p7$	14	12	13	15	1	14	12	13	15	1	4	7	-7	-1	0	1044
$p8$	14	4	20	15	1	10	12	18	15	1	3	-7	2	0	0	1200
$p9$	14	12	13	15	1	14	12	13	15	1	3	8	-5	1	0	1044
$p10$	10	4	5	15	1	10	12	5	15	1	0	-7	0	0	0	1167

Best Global Position at Iteration 3 = $\{14, 12, 13, 15, 2\}$

Iteration 4	Position					Best Local Position					Velocity					Cost
$p1$	14	4	5	15	2	14	12	5	15	2	0	2	2	0	0	1162
$p2$	14	4	13	10	1	10	12	13	15	1	0	2	0	1	0	1154
$p3$	14	12	13	15	1	14	12	13	15	1	0	0	0	0	0	1044
$p4$	14	17	13	15	2	14	12	13	15	2	0	5	0	0	0	1077
$p5$	14	12	5	15	1	8	12	5	15	1	0	-1	2	0	0	1143
$p6$	14	12	5	15	2	14	12	5	15	9	0	-1	2	0	0	1141
$p7$	14	13	5	15	1	14	12	13	15	1	0	2	-6	0	0	1088
$p8$	14	4	18	15	1	10	12	18	15	1	0	2	-2	0	0	1202
$p9$	14	13	5	15	1	14	12	13	15	1	0	2	-4	0	0	1088
$p10$	10	4	5	15	1	10	12	5	15	1	1	2	2	0	0	1167

Best Global Position at Iteration 4 = $\{14, 12, 13, 15, 2\}$

Iteration 5	Position					Best Local Position					Velocity					Cost
$p1$	14	12	13	15	2	14	12	13	15	2	0	8	9	0	0	1043
$p2$	7	12	13	15	1	10	12	13	15	1	-7	8	0	3	0	1117
$p3$	14	12	13	15	1	14	12	13	15	1	0	0	0	0	0	1044
$p4$	14	4	13	15	2	14	12	13	15	2	0	-10	0	0	0	1091
$p5$	10	12	13	15	1	10	12	13	15	1	-4	0	9	-3	0	1046
$p6$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	9	0	13	**944**
$p7$	14	12	13	15	1	14	12	13	15	1	0	-1	6	0	0	1044
$p8$	7	12	13	15	1	7	12	13	15	1	-7	8	-6	0	0	1117
$p9$	14	12	13	15	1	14	12	13	15	1	0	-1	6	0	0	1044
$p10$	14	12	13	15	1	14	12	13	15	1	4	8	9	0	0	1044

Best Global Position at Iteration 5 = $\{14, 12, 13, 15, 9\}$

Iteration 6	Position					Best Local Position					Velocity					Cost
$p1$	14	13	20	15	9	14	13	20	15	9	0	1	7	0	4	994
$p2$	10	13	18	15	9	10	12	13	15	1	3	1	0	2	5	1065
$p3$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	5	944
$p4$	14	17	13	15	9	14	17	13	15	9	0	11	0	0	4	993
$p5$	8	12	20	15	9	10	12	13	15	1	-1	0	7	-2	5	1132
$p6$	14	12	20	15	9	14	12	13	15	9	0	0	7	0	0	1052
$p7$	14	12	18	15	9	14	12	13	15	1	0	0	5	0	5	1056
$p8$	6	13	5	15	9	7	12	13	15	1	-1	1	-5	0	5	1119
$p9$	14	12	18	15	9	14	12	13	15	1	0	0	5	0	5	1056
$p10$	14	12	20	15	9	14	12	13	15	1	0	0	7	0	5	1052

Best Global Position at Iteration 6 = $\{14, 12, 13, 15, 9\}$

Iteration 32	Position					Best Local Position					Velocity					Cost
$p1$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p2$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p3$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p4$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p5$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p6$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p7$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p8$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p9$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
$p10$	**14**	**12**	**13**	**15**	**9**	14	12	13	15	9	0	0	0	0	0	**944**
Best Global Position at Iteration 32 = $\{14, 12, 13, 15, 9\}$																

Chapter 9

Conclusion and Future Directions

This chapter highlights the conclusion of the thesis and potential future work.

9.1 Conclusion

This thesis intends to map the TSN features to wireless nodes and hybrid systems due to the extensive deployment of hybrid systems including wirebound and wireless time-triggered networks and the key features that are introduced by the TSN task group for Ethernet-based infrastructure. A TSN simulation framework that includes simulation models of TSN switches and end nodes in wired and wireless TSN domains is used to assess the suggested models and algorithms. All incorporated TSN nodes share a global time using an improved synchronization protocol based on IEEE 802.1AS. The synchronized nodes distinguish different types of traffic (i.e. TT, AV and BE traffic) using time-aware shaping (IEEE 802.1Qbv), which gives higher priority for TT traffic at the output egress ports. To support the reliability of the system, TSN-enabled nodes replicate the data transmission (IEEE 8021.1CB) through redundant paths. A common system model is provided for all introduced protocols and presented algorithms to synchronize and schedule the services.

An improved version of the standard IEEE 802.1AS time protocol has been presented to address the challenges of wireless environments caused by deterministic and random delays. The asymmetric delays can be produced as a result of several factors like channel characteristics, noise, channel fading, variable data rate and using different wireless technologies. Therefore, by computing the changeable delays in dynamic TSN hybrid networks and considering the frequency drift between the slave and grandmaster clocks, a formula to calculate accurate values for the link delay is derived. This synchronized time is used for the scheduling and filtering in our simulation framework. In addition, the proposed protocol deals with the dynamic and asymmetric behavior of mobile hybrid networks by applying a PDD filter, which is used to exclude the outlier behavior in the wireless networks as a result of mobility and traffic congestion. As shown by the simulation results, the proposed protocol outperforms the standard IEEE 802.1AS and provides a significant gain in terms of synchronization precision.

To find a feasible task and message scheduling solution for $WirelessTSN$ networks, this thesis presents the TMS algorithm which considers link scheduling, routing and interference constraints in one algorithm. It supports multicast traffic between periodic and inter-flow-dependent computational tasks. TMS targets to find solutions that fulfill the task deadlines with the least possible makespan.

In order to evaluate TMS, several algorithms with several scheduling strategies are implemented. The simulation results show the impact of the network topology and the number of tasks on metrics like makespan, consumed energy and failure rate, besides, the impact of changing wireless interference parameters (i.e. α, β) on the measured metrics. Moreover, the results show how varying the deadline affects the deadline miss cases. TMS outperforms in all experimental tests, the reason is that TMS is a comprehensive and adaptive algorithm that selects message forwarding routes that avoid the effect of the signal interference and fulfill the period and deadline constraints simultaneously. In contrast, the other compared algorithms select routes either to avoid the congested traffic or use off-line solutions for task scheduling without considering the link scheduling during the message transmissions. This, in turn, leads to an increased load on some wireless links or missing the deadlines.

This thesis presents the R-TMS algorithm that aims to maximize the reliability and fulfill the deadlines of *WirelessTSN* networks. It supports the redundant FRER during the time slot message scheduling. Moreover, R-TMS employs a task cost definition, which depends on the consumed energy, failure rate and message arrival time of each task t at every available host h to find the best host that satisfies the cost definition. The reliability of network nodes is used as an input to the RBD diagram that computes the reliability of the communication messages. The conditional precedence restrictions are then used to compute the reliability of the tasks. In the end, the reliability of the system is considered as the reliability of the leaf task in the system. The experimental findings demonstrate that, as compared to state-of-the-art TT algorithms that do not enable redundant techniques for their task scheduling solutions, R-TMS greatly increases the system's reliability.

A novel optimized reliable task and message scheduling algorithm OR-TMS is also proposed for *WirelessTSN* networks. In this algorithm, each unscheduled task is firstly sorted according to the idea of the least top-level. The task completion time, failure rates and total energy consumption are considered as optimization goals. Therefore, a DPSO algorithm is constructed to represent all these goals as a multi-objective optimization problem and formulates them as a utility function used to improve a global optimized solution for the task and message scheduling problem. To evaluate the proposed algorithm, we conduct extensive simulation experiments to compare it with our prior R-TMS and other related algorithms. The simulation results show that OR-TMS leads to significant results in the network lifetime, task completion time, deadline miss cases and failure rates in different topologies. In the experiments, the results show also the impact of varying deadlines and the frequent use of network nodes on the deadline miss ratio and network lifetime, respectively. OR-TMS outperforms in all tests because of depending on a bio-inspired algorithm that iteratively tries to find the best possible solutions until the cost function value is converged.

A metascheduler based on OR-TMS that computes a meta schedule graph is provided. The proposed metascheduler controls the state-space explosion of the resulting multi-schedule graph because of increasing numbers of failure events by merging repeated schedules. Different scenarios are used to assess the proposed metascheduler under various failure events. The results show a reduction in the size of the multischedule graphs by applying inputs with different deadline values and different numbers of tasks. Furthermore, the validity provided by the proposed metascheduler, which is the ratio of valid to invalid schedules, is improved when compared to R-TMS, R-MMTS, and R-STFS algorithms.

On the contrary, the redundant routes used in R-TMS and OR-TMS algorithms tend to increase the time and the energy costs compared with fault-free systems. However, by choosing task scheduling solutions that meet their deadlines, it solves the issue of the lengthening task completion time. Hence, the increased time will not affect the efficiency of *WirelessTSN* networks.

9.2 Future Work

Many concepts and features can be added or extended in future work. Therefore, possible direction for future work can be summarized as follows:

- Selecting the best path through different paths between the grandmaster and slave clocks, or using multiple grandmaster clocks, can be considered. Furthermore, a TSN-enabled node can belong to multiple gPTP domains, where each gPTP domain executes the BMCA independently and each domain has its own grandmaster, where one acts as an active grandmaster, and the others are backup grandmasters. The backup grandmaster operates as a cold-standby where it does not dispatch SYNC messages or as a hot-standby where it operates with the active grandmaster simultaneously. Regardless of the operation mode, the backup grandmaster synchronizes to the active grandmaster.
- Consideration of multiple TT channels, where the wireless link can be used simultaneously in both directions, helps improve data delivery.
- Using primary/backup technique to support fault-tolerant task scheduling in *WirelessTSN* networks. FRER supports fault-tolerant message delivery, but the primary/backup technique is used to execute tasks in backup hosts in case the primary hosts fail.
- The task and message scheduling solutions in this thesis are only proposed for TT messages, while other types of traffic (i.e. AV and BE traffic) are sent when no TT traffic is scheduled. Therefore, as a next step, we can incorporate different types of traffic at the same network medium and study the impact of scheduling all traffic at the output egress ports on the end-to-end delay and deadline fulfilment.
- Several fault categories can be addressed (intermittent and transient faults) to evaluate the reliability of the system using reliable task and message scheduling algorithms.
- The TSN simulation framework can use a centralized configuration model to deploy configuration and management data. This facilitates the configuration of the nodes and avoids inconsistencies in the network design.
- The simulation framework and experimental setup can be extended to combine both simulated and physical TSN-enabled nodes in an actual network.

Bibliography

[1] A. AlQammaz, K. A. Darabkh, B. A. Sha'ar, O. Ghatasheh, *et al.*, "A framework for artificial intelligence assisted smart agriculture utilizing lorawan wireless sensor networks," in *International Workshop Soft Computing Applications*, Springer, 2018, pp. 408–421.

[2] A. Khalifeh, N. Bartolini, S. Silvestri, *et al.*, "Hybrid wireless sensor networks: A prototype," in *IFIP Conference on Human-Computer Interaction*, Springer, 2019, pp. 549–553.

[3] N Vikram, K. Harish, M. Nihaal, R. Umesh, A. Shetty, and A. Kumar, "A low cost home automation system using wi-fi based wireless sensor network incorporating internet of things (iot)," in *2017 IEEE 7th International Advance Computing Conference (IACC)*, IEEE, 2017, pp. 174–178.

[4] X. Zhu, X. Qin, and M. Qiu, "Qos-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters," *IEEE transactions on Computers*, vol. 60, no. 6, pp. 800–812, 2011.

[5] S. Mersch, T. Meyerhoff, L. Krüger, and A. Timm-Giel, "Coexistence of wireless avionics intra-communication networks," in *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, IEEE, 2018, pp. 18–23.

[6] T. Ringler, J Steiner, R. Belschner, and B. Hedenetz, "Increasing system safety for by-wire applications in vehicles by using a time triggered architecture," in *International Conference on Computer Safety, Reliability, and Security*, Springer, 1998, pp. 243–253.

[7] G. Heiner and T. Thurner, "Time-triggered architecture for safety-related distributed real-time systems in transportation systems," in *Digest of Papers. Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing (Cat. No. 98CB36224)*, IEEE, 1998, pp. 402–407.

[8] J. Wang, P. Wang, and X. Ma, "Adaptive event-triggered control for quadrotor aircraft with output constraints," *Aerospace Science and Technology*, vol. 105, p. 105 935, 2020.

[9] Institute of electrical and electronics engineers, time sensitive networking, *Time-sensitive networking task group*, `http://www.ieee802.org/1/pages/tsn.html`, Last accessed on 2021-03-16, IEEE, 2017.

[10] S. AS6802, "Time-triggered ethernet," *SAE International*, 2011.

[11] Institute of Electrical and Electronics Engineers, Inc., *802.1asrev - timing and synchronization for time-sensitive application in time-sensitive networking task group*, `http://www.ieee802.org/1/pages/802.1AS-rev.html`, Last accessed on 2021-03-16, IEEE, 2017.

[12] T. Neagoe, V. Cristea, and L. Banica, "Ntp versus ptp in com puter networks clock synchronization," in *2006 IEEE International Symposium on Industrial Electronics*, IEEE, vol. 1, 2006, pp. 317–362.

[13] E. Heidinger, F. Geyer, S. Schneele, and M. Paulitsch, "A performance study of audio video bridging in aeronautic ethernet networks," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, IEEE, 2012, pp. 67–75.

[14] A. S. Sethi and V. Y. Hnatyshin, *The practical OPNET user guide for computer network simulation*. Chapman and Hall/CRC, 2012.

[15] Institute of Electrical and Electronics Engineers, Inc., *802.1qbv - enhancements for scheduled traffic, in time-sensitive networking task group*, `http://www.ieee802.org/1/pages/802.1bv.html`, Last accessed on 2021-03-16, IEEE, 2016.

[16] "Institute of electrical and electronics engineers, inc. 802.1cb - frame replication and elimination for reliability." (), [Online]. Available: `http://www.ieee802.org/1/files/private/cb-drafts/d2/802-1CB-D2-9.pdf,lastaccessed(11.04.2021).`.

[17] J. Eidson and K. Lee, "Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," in *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*, Ieee, 2002, pp. 98–105.

[18] J. Jasperneite, K. Shehab, and K. Weber, "Enhancements to the time synchronization standard ieee-1588 for a system of cascaded bridges," in *IEEE International Workshop on Factory Communication Systems*, Citeseer, 2004, pp. 239–244.

[19] S. Lee, “An enhanced ieee 1588 time synchronization algorithm for asymmetric communication link using block burst transmission,” *IEEE communications letters*, vol. 12, no. 9, pp. 687–689, 2008.

[20] T Kokilavani, D. G. Amalarethinam, *et al.*, “Load balanced min-min algorithm for static meta-task scheduling in grid computing,” *International Journal of Computer Applications*, vol. 20, no. 2, pp. 43–49, 2011.

[21] D. M. Shila and T. Anjali, “Load aware traffic engineering for mesh networks,” *Computer Communications*, vol. 31, no. 7, pp. 1460–1469, 2008.

[22] L. T. Nguyen, R. Beuran, and Y. Shinoda, “A load-aware routing metric for wireless mesh networks,” in *2008 IEEE Symposium on Computers and Communications*, IEEE, 2008, pp. 429–435.

[23] F. Marini and B. Walczak, “Particle swarm optimization (pso). a tutorial,” *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015.

[24] R. Gomathi and J. M. L. Manickam, “Energy efficient shortest path routing protocol for underwater acoustic wireless sensor network,” *Wireless Personal Communications*, vol. 98, no. 1, pp. 843–856, 2018.

[25] M. Pahlevan, N. Tabassam, and R. Obermaisser, “Heuristic list scheduler for time triggered traffic in time sensitive networks,” *ACM Sigbed Review*, vol. 16, no. 1, pp. 15–20, 2019.

[26] T. Aladwani, “Types of task scheduling algorithms in cloud computing environment,” *Scheduling Problems-New Applications and Trends*, 2020.

[27] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, “Extended synchronization protocol based on ieee802. 1as for improved precision in dynamic and asymmetric tsn hybrid networks,” in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2020, pp. 1–8.

[28] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, “Time triggered scheduling algorithm for real-time wireless systems,” in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, IEEE, vol. 1, 2020, pp. 265–272.

[29] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, “A reliable job allocation scheduler for time-triggered wireless networks,” in *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)*, IEEE, 2021, pp. 1–10.

[30] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, “Reliable task allocation for time-triggered iot-wsn using discrete particle swarm optimization,” *IEEE Internet of Things Journal*, 2021.

[31] A. Saifullah, D. Gunatilaka, P. Tiwari, *et al.*, “Schedulability analysis under graph routing in wirelesshart networks,” in *2015 IEEE Real-Time Systems Symposium*, IEEE, 2015, pp. 165–174.

[32] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, “End-to-end communication delay analysis in industrial wireless networks,” *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1361–1374, 2014.

[33] H. Zhang, P. Soldati, and M. Johansson, “Performance bounds and latency-optimal scheduling for convergecast in wirelesshart networks,” *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2688–2696, 2013.

[34] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, “Adaptive real-time communication for wireless cyber-physical systems,” *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 2, pp. 1–29, 2017.

[35] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu, “Incorporating emergency alarms in reliable wireless process control,” in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, 2015, pp. 218–227.

[36] P. Suriyachai, J. Brown, and U. Roedig, “Time-critical data delivery in wireless sensor networks,” in *International Conference on Distributed Computing in Sensor Systems*, Springer, 2010, pp. 216–229.

[37] F. Dobslaw, T. Zhang, and M. Gidlund, “End-to-end reliability-aware scheduling for wireless sensor networks,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 758–767, 2014.

[38] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, “6tisch: Deterministic ip-enabled industrial internet (of things),” *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014.

[39] R. Brummet, D. Gunatilaka, D. Vyas, O. Chipara, and C. Lu, “A flexible retransmission policy for industrial wireless sensor actuator networks,” in *2018 IEEE International Conference on Industrial Internet (ICII)*, IEEE, 2018, pp. 79–88.

[40] M. Zhao, Y. Yang, and C. Wang, “Mobile data gathering with load balanced clustering and dual data uploading in wireless sensor networks,” *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 770–785, 2014.

[41] P. Neamatollahi, S. Abrishami, M. Naghibzadeh, M. H. Y. Moghaddam, and O. Younis, "Hierarchical clustering-task scheduling policy in cluster-based wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 1876–1886, 2017.

[42] F. Ren, J. Zhang, T. He, C. Lin, and S. K. D. Ren, "Ebrp: Energy-balanced routing protocol for data gathering in wireless sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 22, no. 12, pp. 2108–2125, 2011.

[43] S. Tyagi and N. Kumar, "A systematic review on clustering and routing techniques based upon leach protocol for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 623–645, 2013.

[44] W. Yu, Y. Huang, and A. Garcia-Ortiz, "An altruistic compression-scheduling scheme for cluster-based wireless sensor networks," in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2015, pp. 73–81. DOI: 10.1109/SAHCN.2015.7338293.

[45] Y. Liang and W. Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 12–18, 2010.

[46] Y. Wu, X.-Y. Li, Y. Li, and W. Lou, "Energy-efficient wake-up scheduling for data collection and aggregation," *IEEE Transactions on parallel and distributed systems*, vol. 21, no. 2, pp. 275–287, 2009.

[47] D. Ye and M. Zhang, "A self-adaptive sleep/wake-up scheduling approach for wireless sensor networks," *IEEE transactions on cybernetics*, vol. 48, no. 3, pp. 979–992, 2017.

[48] O. Chipara, C. Wu, C. Lu, and W. Griswold, "Interference-aware real-time flow scheduling for wireless sensor networks," in *2011 23rd Euromicro Conference on Real-Time Systems*, IEEE, 2011, pp. 67–77.

[49] A. Jamthe, A. Mishra, and D. P. Agrawal, "Scheduling schemes for interference suppression in healthcare sensor networks," in *2014 IEEE international conference on communications (ICC)*, IEEE, 2014, pp. 391–396.

[50] S. S. Nirjon, J. A. Stankovic, and K. Whitehouse, "Iaa: Interference aware anticipatory algorithm for scheduling and routing periodic real-time streams in wireless sensor networks," in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, IEEE, 2010, pp. 14–21.

[51] B. Fateh and M. Govindarasu, "Joint scheduling of tasks and messages for energy minimization in interference-aware real-time sensor networks," *IEEE transactions on mobile computing*, vol. 14, no. 1, pp. 86–98, 2013.

[52] G. Joshi, S. Jardosh, and P. Ranjan, "Bounds on dynamic modulation scaling for wireless sensor networks," in *2007 Third International Conference on Wireless Communication and Sensor Networks*, IEEE, 2007, pp. 13–16.

[53] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No. 98TH8379)*, IEEE, 1998, pp. 76–81.

[54] W. Chen, J. Sun, L. Zhang, X. Liu, and L. Hong, "An implementation of ieee 1588 protocol for ieee 802.11 wlan," *Wireless networks*, vol. 21, no. 6, pp. 2069–2085, 2015.

[55] R. Exel, "Clock synchronization in ieee 802.11 wireless lans using physical layer timestamps," in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, IEEE, 2012, pp. 1–6.

[56] A. Mahmood, G. Gaderer, H. Trsek, S. Schwalowsky, and N. Kerö, "Towards high accuracy in ieee 802.11 based clock synchronization using ptp," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2011, pp. 13–18.

[57] G. von Zengen, K. Garlichs, Y. Schrcöder, and L. C. Wolf, "A sub-microsecond clock synchronization protocol for wireless industrial monitoring and control networks," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, 2017, pp. 1266–1270.

[58] A. Elsts, S. Duquennoy, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "Microsecond-accuracy time synchronization using the ieee 802.15. 4 tsch protocol," in *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*, IEEE, 2016, pp. 156–164.

[59] A. Tinka, T. Watteyne, and K. Pister, "A decentralized scheduling algorithm for time synchronized channel hopping," in *International Conference on Ad Hoc Networks*, Springer, 2010, pp. 201–216.

[60] D. Shrestha, Z. Pang, and D. Dzung, "Precise clock synchronization in high performance wireless communication for time sensitive networking," *IEEE Access*, vol. 6, pp. 8944–8953, 2018.

[61] H. Cho, J. Jung, B. Cho, Y. Jin, S.-W. Lee, and Y. Baek, "Precision time synchronization using ieee 1588 for wireless sensor networks," in *2009 International Conference on Computational Science and Engineering*, IEEE, vol. 2, 2009, pp. 579–586.

[62] A. Mahmood, G. Gaderer, and P. Loschmidt, "Software support for clock synchronization over ieee 802.11 wireless lan with open source drivers," in *2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2010, pp. 61–66.

[63] A. Mahmood, R. Exel, and T. Sauter, "Delay and jitter characterization for software-based clock synchronization over wlan using ptp," *IEEE Transactions on industrial informatics*, vol. 10, no. 2, pp. 1198–1206, 2014.

[64] A. Mahmood, R. Exel, and T. Sauter, "Performance of ieee 802.11's timing advertisement against synctsf for wireless clock synchronization," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 370–379, 2016.

[65] M. Buevich, N. Rajagopal, and A. Rowe, "Hardware assisted clock synchronization for real-time sensor networks," in *2013 IEEE 34th Real-Time Systems Symposium*, IEEE, 2013, pp. 268–277.

[66] J. Chen, Y. Li, Y. Song, and H. Chen, "Hardware-assisted clock synchronization in ieee 802.11 wireless real-time application," 2007.

[67] J. Wu, L. Zhang, Y. Bai, and Y. Sun, "Cluster-based consensus time synchronization for wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1404–1413, 2014.

[68] Z. Wang, P. Zeng, M. Zhou, D. Li, and J. Wang, "Cluster-based maximum consensus time synchronization for industrial wireless sensor networks," *Sensors*, vol. 17, no. 1, p. 141, 2017.

[69] D. K. Lam, K. Yamaguchi, Y. Nagao, M. Kurosaki, and H. Ochi, "An improved precision time protocol for industrial wlan communication systems," in *2016 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, 2016, pp. 824–829.

[70] S. Schriegel, H. Trsek, and J. Jasperneite, "Enhancement for a clock synchronization protocol in heterogeneous networks," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2009, pp. 1–5.

[71] A. Mahmood and F. Ring, "Clock synchronization for ieee 802.11 based wired-wireless hybrid networks using ptp," in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, IEEE, 2012, pp. 1–6.

[72] G. M. Garner and H. Ryu, "Synchronization of audio/video bridging networks using ieee 802.1 as," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 140–147, 2011.

[73] M. Bauer, G. May, and V. Jain, "A wireless gateway approach enabling industrial real-time communication on the field level of factory automation," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, IEEE, 2014, pp. 1–8.

[74] Y. Quan and G. Liu, "Drifting clock model for network simulation in time synchronization," in *2008 3rd International Conference on Innovative Computing Information and Control*, IEEE, 2008, pp. 385–385.

[75] P. Loschmidt, R. Exel, and G. Gaderer, "Highly accurate timestamping for ethernet-based clock synchronization," *Journal of Computer Networks and Communications*, vol. 2012, 2012.

[76] A. Swidan, H. B. Abdelghany, R. Saifan, and Z. Zilic, "Mobility and direction aware ad-hoc on demand distance vector routing protocol," *Procedia Computer Science*, vol. 94, pp. 49–56, 2016.

[77] A. Mildner, "Time sensitive networking for wireless networks-a state of the art analysis," *Network*, vol. 33, 2019.

[78] L. Zhang, D. Goswami, R. Schneider, and S. Chakraborty, "Task-and network-level schedule co-synthesis of ethernet-based time-triggered systems," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2014, pp. 119–124.

[79] S. Amin and R. Obermaisser, "Time-triggered scheduling of query executions for active diagnosis in distributed real-time systems," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2017, pp. 1–9.

[80] S. Amin and R. Obermaisser, "A time-triggered scheduling algorithm for active diagnosis in heterogeneous distributed systems," in *2018 IEEE International Conference on Computational Science and Engineering (CSE)*, IEEE, 2018, pp. 44–55.

[81] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on industrial electronics*, vol. 56, no. 10, pp. 4258–4265, 2009.

[82] S. Hong, X. S. Hu, T. Gong, and S. Han, "On-line data link layer scheduling in wireless networked control systems," in *2015 27th Euromicro Conference on Real-Time Systems*, IEEE, 2015, pp. 57–66.

[83] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2502–2517, 2018.

[84] T. Zhang, T. Gong, Z. Yun, S. Han, Q. Deng, and X. S. Hu, "Fd-pas: A fully distributed packet scheduling framework for handling disturbances in real-time wireless networks," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2018, pp. 1–12.

[85] J. Song, S. Han, A. Mok, *et al.*, "Wirelesshart: Applying wireless technology in real-time industrial process control," in *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*, IEEE, 2008, pp. 377–386.

[86] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on networking*, vol. 12, no. 3, pp. 493–506, 2004.

[87] Y. Chen, H. Zhang, N. Fisher, G. Yin, *et al.*, "Probabilistic per-packet real-time guarantees for wireless networked sensing and control," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2133–2145, 2018.

[88] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Fully distributed packet scheduling framework for handling disturbances in lossy real-time wireless networks," *IEEE Transactions on Mobile Computing*, 2019.

[89] Y. Jin, J. Jin, A. Gluhak, K. Moessner, and M. Palaniswami, "An intelligent task allocation scheme for multihop wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 444–451, 2011.

[90] Y. Jin, S. Vural, A. Gluhak, and K. Moessner, "Dynamic task allocation in multi-hop multimedia wireless sensor networks with low mobility," *Sensors*, vol. 13, no. 10, pp. 13 998–14 028, 2013.

[91] J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun, "Task allocation for wireless sensor network using modified binary particle swarm optimization," *IEEE Sensors Journal*, vol. 14, no. 3, pp. 882–892, 2013.

[92] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A pso-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3236–3249, 2014.

[93] W. Yu, Y. Huang, and A. Garcia-Ortiz, "Distributed optimal on-line task allocation algorithm for wireless sensor networks," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 446–458, 2017.

[94] P. Neamatollahi, M. Naghibzadeh, S. Abrishami, and M.-H. Yaghmaee, "Distributed clustering-task scheduling for wireless sensor networks using dynamic hyper round policy," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 334–347, 2017.

[95] L. Dai, Z. Shen, T. Chen, and Y. Chang, "Analysis and modeling of task scheduling in wireless sensor network based on divisible load theory," *International Journal of Communication Systems*, vol. 27, no. 5, pp. 721–731, 2014.

[96] L. Dai, Y. Chang, and Z. Shen, "An optimal task scheduling algorithm in wireless sensor networks," *International Journal of Computers Communications & Control*, vol. 6, no. 1, pp. 101–112, 2011.

[97] T. Xie and X. Qin, "An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems," *IEEE Transactions on Computers*, vol. 57, no. 3, pp. 329–343, 2008.

[98] T. Gong, T. Zhang, X. S. Hu, Q. Deng, M. Lemmon, and S. Han, "Reliable dynamic packet scheduling over lossy real-time wireless networks," in *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[99] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," *Information Sciences*, vol. 319, pp. 113–131, 2015.

[100] F. F. Marshall, M. Mu'azu, I. Umoh, A. Salawudeen, B. Sadiq, and D. Ikpe, "A modified real-time fault-tolerant task allocation scheme for wireless sensor networks," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, vol. 4, no. 1, pp. 45–54, 2018.

[101] Q. Han, T. Wang, and G. Quan, "Enhanced fault-tolerant fixed-priority scheduling of hard real-time tasks on multi-core platforms," in *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, IEEE, 2015, pp. 21–30.

[102] M. Pandya and M. Malek, "Minimum achievable utilization for fault-tolerant processing of periodic tasks," *IEEE Transactions on Computers*, vol. 47, no. 10, pp. 1102–1112, 1998.

[103] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *IEEE Transactions on Computers*, vol. 53, no. 2, pp. 217–231, 2004.

[104] T. Wei, P. Mishra, K. Wu, and J. Zhou, "Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1386–1399, 2012.

[105] Q. Han, M. Fan, L. Niu, and G. Quan, "Energy minimization for fault tolerant scheduling of periodic fixed-priority applications on multiprocessor platforms," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2015, pp. 830–835.

[106] Y. Guo, D. Zhu, and H. Aydin, "Generalized standby-sparing techniques for energy-efficient fault tolerance in multiprocessor real-time systems," in *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, IEEE, 2013, pp. 62–71.

[107] H. Chetto and M. Chetto, "Some results of the earliest deadline scheduling algorithm," *IEEE Transactions on software engineering*, vol. 15, no. 10, pp. 1261–1269, 1989.

[108] K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 55–67, 1994.

[109] K. W. Tindell, A. Burns, and A. J. Wellings, "Allocating hard real-time tasks: An np-hard problem made easy," *Real-Time Systems*, vol. 4, no. 2, pp. 145–165, 1992.

[110] A. M. Terrasa Barrena, "Flexible real-time linux a new environment for flexible hard real-time systems," Ph.D. dissertation, Universitat Politècnica de València, 2012.

[111] J. M. Yabarrena, J. A. Herrera, and G. A. Caurin, "Embedded application development. a procedure based on vxworks© real-time operating system," SAE Technical Paper, Tech. Rep., 2010.

[112] M. A. Rivas and M. G. Harbour, "Marte os: An ada kernel for real-time embedded applications," in *International Conference on Reliable Software Technologies*, Springer, 2001, pp. 305–316.

[113] Z. Zhang, *Distributed real-time operating system (DRTOS) modeling in SpecC*. Iowa State University, 2006.

[114] K. M. Zuberi and K. G. Shin, "A causal message ordering scheme for distributed embedded real-time systems," in *Proceedings 15th Symposium on Reliable Distributed Systems*, IEEE, 1996, pp. 210–219.

[115] H. Kopetz, "Event-triggered versus time-triggered real-time systems," in *Operating Systems of the 90s and Beyond*, Springer, 1991, pp. 86–101.

[116] F. Scheler and W. Schröder-Preikschat, "Time-triggered vs. event-triggered: A matter of configuration?" In *ITG FA 6.2 Workshop on Model-Based Testing, GI/ITG Workshop on Non-Functional Properties of Embedded Systems, 13th GI/ITG Conference Measuring, Modelling, and Evaluation of Computer and Communications*, VDE, 2006, pp. 1–6.

[117] H. Kopetz, "Time-triggered real-time computing," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 59–70, 2002.

[118] A. Avizienis, J.-C. Laprie, B. Randell, *et al.*, *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[119] N. Xiong, Y. Yang, M. Cao, J. He, and L. Shu, "A survey on fault-tolerance in distributed network systems," in *2009 International Conference on Computational Science and Engineering*, IEEE, vol. 2, 2009, pp. 1065–1070.

[120] R. Obermaisser and P. Peti, "The fault assumptions in distributed integrated architectures," *SAE Transactions*, pp. 789–801, 2007.

[121] ——, "A fault hypothesis for integrated architectures," in *2006 International Workshop on Intelligent Solutions in Embedded Systems*, IEEE, 2006, pp. 1–18.

[122] H. Kopetz, "The fault hypothesis for the time-triggered architecture," in *Building the Information Society*, Springer, 2004, pp. 221–233.

[123] S. Osaki *et al.*, "Reliability evaluation of some fault-tolerant computer architectures," 1980.

[124] "Error detection in computer networks." (), [Online]. Available: https://www.geeksforgeeks.org/error-detection-in-computer-networks/,lastaccessed(08.05.2021)..

[125] "Error correcting codes - hamming codes." (), [Online]. Available: https://www.tutorialspoint.com/error-correcting-codes-hamming-codes,lastaccessed(08.05.2021)..

[126] "Error-detecting codes - checksums." (), [Online]. Available: https://www.tutorialspoint.com/error-detecting-codes-checksums,lastaccessed(08.05.2021)..

[127] "Understanding the cyclic redundancy check." (), [Online]. Available: https://www.cardinalpeak.com/blog/understanding-the-cyclic-redundancy-check/,lastaccessed(08.05.2021)..

[128] T. Mandel and J. Mache, "Investigating crc polynomials that correct burst errors.," in *ICWN*, 2009, pp. 632–637.

[129] P. Liu and Y. Wang, "Multiphase damage confinement system for databases," in *Research Directions in Data and Applications Security*, Springer, 2003, pp. 75–87.

[130] F. Jambon, "Error recovery representations in interactive system development," in *Third Annual ERCIM Workshop on "User Interfaces for All", Obernai, France*, 1997, pp. 177–182.

[131] "Fault treatment patterns." (), [Online]. Available: `https://www.oreilly.com/library/view/patterns-for-fault/9780470319796/ch08.html#:~:text=After%20an%20error%20is%20processed,cause%20the%20same%20error%20again.,lastaccessed(08.05.2021).`.

[132] M. Khayyambashi, H. Soltani, and M. Sadeghi, "The study of hardware redundancy techniques to provide a fault tolerant system,"

[133] S. A. Shernta and A. A. Tamtum, "Using triple modular redundant (tmr) technique in critical systems operation," in *Proceeding Book of First Conference for Engineering Sciences and Technology (CEST-2018)(Part 1)*, 2018.

[134] P. Balasubramanian, "Asic-based design of nmr system health monitor for mission/safety–critical applications," *SpringerPlus*, vol. 5, no. 1, pp. 1–16, 2016.

[135] M. Murakami, "Task-based dynamic fault tolerance for humanoid robots," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, vol. 3, 2006, pp. 2197–2202.

[136] "Fault tolerance." (), [Online]. Available: `https://tiagobluiz.com/2019/04/21/sec-3-fault-tolerance/.,lastaccessed(09.05.2021).`.

[137] T. Lehtonen *et al.*, "On fault tolerance methods for networks-on-chip," 2009.

[138] E. Dubrova, "Hardware redundancy," in *Fault-Tolerant Design*, Springer, 2013, pp. 55–86.

[139] J. Losq, "A highly efficient redundancy scheme: Self-purging redundancy," *IEEE Computer Architecture Letters*, vol. 25, no. 06, pp. 569–578, 1976.

[140] A. Sengupta and S. Bhadauria, "Bacterial foraging driven exploration of multi cycle fault tolerant datapath based on power-performance tradeoff in high level synthesis," *Expert Systems with Applications*, vol. 42, no. 10, pp. 4719–4732, 2015.

[141] A. Eghbal, P. M. Yaghini, H Pedram, and H. Zarandi, "Designing fault-tolerant network-on-chip router architecture," *International Journal of Electronics*, vol. 97, no. 10, pp. 1181–1192, 2010.

[142] L. Chen and A. Avizienis, "N-version programming: A fault-tolerance approach to reliability of software operation," in *Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8)*, vol. 1, 1978, pp. 3–9.

[143] J. H. Cho, H. Kim, S. Wang, *et al.*, "A novel method for providing precise time synchronization in a distributed control system using boundary clock," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2824–2829, 2009.

[144] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," 2010.

[145] "Combining ptp with ntp to get the best of both worlds." (2016), [Online]. Available: `https://www.redhat.com/en/blog/combining-ptp-ntp-get-best-both-worlds`.

[146] T. Instruments, "An-1728 ieee 1588 precision time protocol time synchronization performance," *Application Report SNLA098A*, vol. 10, 2013.

[147] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle, *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[148] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO wireless communications*. Cambridge university press, 2007.

[149] O. Edfors, M. Sandell, J. van de Beek, D. Landström, and F. Sjöberg, *An introduction to orthogonal frequency-division multiplexing*. Luleå tekniska universitet, 1997.

[150] J. So and N. H. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, 2004, pp. 222–233.

[151] B. Alawieh, Y. Zhang, C. Assi, and H. Mouftah, "Improving spatial reuse in multihop wireless networks-a survey," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 71–91, 2009.

[152] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005, pp. 73–87.

[153] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on information theory*, vol. 46, no. 2, pp. 388–404, 2000.

[154] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless networks*, vol. 11, no. 4, pp. 471–487, 2005.

[155] A. D. Gore and A. Karandikar, "Link scheduling algorithms for wireless mesh networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 258–273, 2010.

[156] O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer, "Complexity in geometric sinr," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, 2007, pp. 100–109.

[157] D. Yang, X. Fang, N. Li, and G. Xue, "A simple greedy algorithm for link scheduling with the physical interference model," in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, IEEE, 2009, pp. 1–6.

[158] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: The joint routing and scheduling problem," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 42–54.

[159] J. Grönkvist and A. Hansson, "Comparison between graph-based and interference-based stdma scheduling," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001, pp. 255–258.

[160] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke, "The ieee 802.11 universe," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 62–70, 2010.

[161] R. L. Peterson, D. E. Borth, and R. E. Ziemer, *An introduction to spread-spectrum communications*. Prentice-Hall, Inc., 1995.

[162] M. A. Youssef, A. Vasan, and R. E. Miller, "Specification and analysis of the dcf and pcf protocols in the 802.11 standard using systems of communicating machines," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, IEEE, 2002, pp. 132–141.

[163] M Molle and L. Kleinrock, "Virtual time csma: Why two clocks are better than one," *IEEE transactions on Communications*, vol. 33, no. 9, pp. 919–933, 1985.

[164] D. Tardioli, "Real time communications in wireless ad-hoc networks. the rt-wmp protocol," Ph.D. dissertation, Universidad de Zaragoza, 2010.

[165] S. S. Sawwashere and U. N. Sonali, "Rts/cts frame synchronization to minimize the hidden node problem in wireless network," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, pp. 570–575, 2014.

[166] G. Fokus-STEP, "Analyzing the rts/cts mechanism in the dfwmac media access protocol for wireless lans,"

[167] S. Ray, J. B. Carruthers, and D. Starobinski, "Rts/cts-induced congestion in ad hoc wireless lans," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, IEEE, vol. 3, 2003, pp. 1516–1521.

[168] S. Xu and T. Saadawi, "Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks?" *IEEE communications Magazine*, vol. 39, no. 6, pp. 130–137, 2001.

[169] M. Ali, U. Saif, A. Dunkels, *et al.*, "Medium access control issues in sensor networks," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 33–36, 2006.

[170] W.-Z. Song, R. Huang, B. Shirazi, and R. LaHusen, "Treemac: Localized tdma mac protocol for real-time high-data-rate sensor networks," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 750–765, 2009.

[171] F. Santos, L. Almeida, P. Pedreiras, L. S. Lopes, and T. Facchinetti, "An adaptive tdma protocol for soft real-time wireless communication among mobile autonomous agents," in *Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS*, Citeseer, vol. 2004, 2004, pp. 657–665.

[172] W. Shen, T. Zhang, M. Gidlund, and F. Dobslaw, "Sas-tdma: A source aware scheduling algorithm for real-time communication in industrial wireless sensor networks," *Wireless networks*, vol. 19, no. 6, pp. 1155–1170, 2013.

[173] J. Kim, J. Lim, C. Pelczar, and B. Jang, "Rrmac: A sensor network mac for real time and reliable packet transmission," in *2008 IEEE International Symposium on Consumer Electronics*, IEEE, 2008, pp. 1–4.

[174] T. Zheng and K. Ki-Il, "A survey on real-time mac protocols in wireless sensor networks," *Communications and Network*, vol. 2010, 2010.

[175] B. K. Singh and K. E. Tepe, "A novel real-time mac layer protocol for wireless sensor network applications," in *2009 IEEE International Conference on Electro/Information Technology*, IEEE, 2009, pp. 338–343.

[176] T. Watteyne, I. Augé-Blum, and S. Ubéda, "Dual-mode real-time mac protocol for wireless sensor networks: A validation/simulation approach," in *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, 2006, 2–es.

[177] T. Watteyne and I. Augé-Blum, "Proposition of a hard real-time mac protocol for wireless sensor networks," in *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, IEEE, 2005, pp. 533–536.

[178] C Caccamo and L. Y. Zhang, "The capacity of implicit edf in wireless sensor networks," in *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.*, IEEE, 2003, pp. 267–275.

[179] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.*, IEEE, 2002, pp. 39–48.

[180] A. Krohn, M. Beigl, C. Decker, T. Zimmer, *et al.*, "Tomac–real-time message ordering in wireless sensor networks using the mac layer," in *International Workshop on Networked Sensing Systems (INSS)*, 2005.

[181] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, J. García-Haro, P. Pavón-Mariño, and M. B. Delgado, "A wireless sensor networks mac protocol for real-time applications," *Personal and Ubiquitous Computing*, vol. 12, no. 2, pp. 111–122, 2008.

[182] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, J. García-Haro, P. Pavón-Mariño, and M. V. Bueno-Delgado, "A real-time mac protocol for wireless sensor networks: Virtual tdma for sensors (vts)," in *International Conference on Architecture of Computing Systems*, Springer, 2006, pp. 382–396.

[183] H. Li, P. Shenoy, and K. Ramamritham, "Scheduling messages with deadlines in multi-hop real-time sensor networks," in *11th IEEE Real Time and Embedded Technology and Applications Symposium*, IEEE, 2005, pp. 415–425.

[184] J. Chen, P. Zhu, and Z. Qi, "Pr-mac: Path-oriented real-time mac protocol for wireless sensor network," in *International Conference on Embedded Software and Systems*, Springer, 2007, pp. 530–539.

[185] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency mac for tree-based data gathering in sensor networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 7, pp. 863–875, 2007.

[186] J. A. Afonso, L. A. Rocha, H. R. Silva, and J. H. Correia, "Mac protocol for low-power real-time wireless sensing and actuation," in *2006 13th IEEE International Conference on Electronics, Circuits and Systems*, IEEE, 2006, pp. 1248–1251.

[187] E. Tovar and F. Vasques, "Real-time fieldbus communications using profibus networks," *IEEE transactions on Industrial Electronics*, vol. 46, no. 6, pp. 1241–1251, 1999.

[188] G. Prytz, "A performance analysis of ethercat and profinet irt," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, IEEE, 2008, pp. 408–415.

[189] J. L. Sobrinho and A. S. Krishnakumar, "Equb-ethernet quality of service using black bursts," in *Proceedings 23rd Annual Conference on Local Computer Networks. LCN'98 (Cat. No. 98TB100260)*, IEEE, 1998, pp. 286–296.

[190] N. Malcolm and W. Zhao, "The timed-token protocol for real-time communications," *Computer*, vol. 27, no. 1, pp. 35–41, 1994.

[191] D. Miorandi and S. Vitturi, "Analysis of master-slave protocols for real-time-industrial communications over ieee802. 11 wlans," in *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04. 2004*, IEEE, 2004, pp. 143–148.

[192] F. Hanssen and P. G. Jansen, *Real-time communication protocols: an overview*. Citeseer, 2003.

[193] A. TTTech Computertechnik, "Tttech computertechnik ag, schönbrunner straße, a-1040 vienna, austria," *TTP/C Protocol*, 1999.

[194] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (tte) design," in *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, IEEE, 2005, pp. 22–33.

[195] W. Steiner and G. Bauer, "Ttethernet: Time-triggered services for ethernet networks," in *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*, 2009, p. 1.

[196] J. Imtiaz, J. Jasperneite, and L. Han, "A performance study of ethernet audio video bridging (avb) for industrial real-time communication," in *2009 IEEE Conference on Emerging Technologies & Factory Automation*, IEEE, 2009, pp. 1–8.

[197] I. W. Group *et al.*, "Local and metropolitan area networks-virtual bridged local area networks," *IEEE Std 802.1 Q-1998*, 1999.

[198] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using ieee time-sensitive networks," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 86–94, 2016.

[199] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1 qbv time sensitive networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[200] I. of Electrical and I. Electronics Engineers, *P802.1qci – per-stream filtering and policing, in time-sensitive networking task group*, https://1.ieee802.org/tsn/802-1qci/, Last accessed on 2021-03-16, IEEE, 2017.

[201] Institute of Electrical and Electronics Engineers, Inc., *P802.1qcc – stream reservation protocol (srp) enhancements and performance improvements, draft 1.6*, https://www.ieee802.org/1/files/private/cc-drafts/d1/802-1Qcc-d1-6.pdf, IEEE, 2017.

[202] D. Cavalcanti, S. Bush, M. Illouz, G. Kronauer, A. Regev, and G. Venkatesan, "Wireless tsn-definitions use cases & standards roadmap," *Avnu Alliance*, pp. 1–16, 2020.

[203] M. Pahlevan, "Time sensitive networking for virtualized integrated real-time systems," 2020.

[204] T. Issariyakul and E. Hossain, "Introduction to network simulator 2 (ns2)," in *Introduction to network simulator NS2*, Springer, 2009, pp. 1–18.

[205] R. L. Patel, M. J. Pathak, and A. J. Nayak, "Survey on network simulators," *International Journal of Computer Applications*, vol. 975, p. 8887, 2018.

[206] "Scalable network technologies." (), [Online]. Available: https://www.scalable-networks.com/., lastaccessed(20.05.2021)..

[207] "Omnet++ discrete event simulator." (), [Online]. Available: https://omnetpp.org/.,lastaccessed(20.05.2021)..

[208] M. Pahlevan, B. Balakrishna, and R. Obermaisser, "Simulation framework for clock synchronization in time sensitive networking," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, IEEE, 2019, pp. 213–220.

[209] W. Wojdak, "Rapid spanning tree protocol: A new solution from an old technology," *Reprinted from CompactPCI Systems*, 2003.

[210] "Link aggregation." (), [Online]. Available: https://en.wikipedia.org/wiki/Link_aggregation..

[211] F. Cristian, "Probabilistic clock synchronization," *Distributed computing*, vol. 3, no. 3, pp. 146–158, 1989.

[212] C. W. Nicholls and P. Wu, *Method and system for correcting oscillator frequency drift*, US Patent 8,674,778, 2014.

[213] Y. Liu, K. Xu, and M. Wei, "A study on mems oscillators 'frequency drift of temperature," in *MATEC Web of Conferences*, EDP Sciences, vol. 189, 2018, p. 11 002.

[214] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, "Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 2017, pp. 1–6.

[215] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "Ilp-based joint routing and scheduling for time-triggered networks," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, 2017, pp. 8–17.

[216] R. E. Moraes, W. W. dos Reis, H. R. Rocha, and D. J. Coura, "Power-efficient and interference-free link scheduling algorithms for connected wireless sensor networks," *Wireless Networks*, pp. 1–20, 2019.

[217] S. Kurt and B. Tavli, "Path-loss modeling for wireless sensor networks: A review of models and comparative evaluations.," *IEEE Antennas and Propagation Magazine*, vol. 59, no. 1, pp. 18–37, 2017.

[218] "Snap library 4.0, user reference documentation." (2017), [Online]. Available: https://snap.stanford.edu/snap/doc/snapuser-ref/index.html.

[219] A. Damaso, N. Rosa, and P. Maciel, "Reliability of wireless sensor networks," *Sensors*, vol. 14, no. 9, pp. 15 760–15 785, 2014.

[220] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced engineering informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[221] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: A brief survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 2, pp. 262–267, 2010.

[222] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-International Conference on Neural Networks*, IEEE, vol. 4, 1995, pp. 1942–1948.

[223] *Annualized failure rate*, 2020. [Online]. Available: `https://sciencing.com/calculate-failure-rates-6403358.html`.

[224] W. Guo, N. Xiong, A. V. Vasilakos, G. Chen, and C. Yu, “Distributed k–connected fault–tolerant topology control algorithms with pso in future autonomic sensor systems,” *International Journal of Sensor Networks*, vol. 12, no. 1, pp. 53–62, 2012.

[225] S. Helwig, J. Branke, and S. Mostaghim, “Experimental analysis of bound handling techniques in particle swarm optimization,” *IEEE Transactions on Evolutionary computation*, vol. 17, no. 2, pp. 259–271, 2012.

[226] J. Goossens and C. Macq, “Limitation of the hyper-period in real-time periodic task set generation,” in *In Proceedings of the RTS Embedded System (RTS’01*, Citeseer, 2001.

[227] S. Lv, Y. Lu, and Y. Ji, “An enhanced ieee 1588 time synchronization for asymmetric communication link in packet transport network,” *IEEE Communications Letters*, vol. 14, no. 8, pp. 764–766, 2010.

[228] M. Pahlevan and R. Obermaisser, “Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks,” in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, vol. 1, 2018, pp. 337–344.