Jesús Aísa Vicén

Wireless Real-Time Communication in Tunnel-like Environments using Wireless Mesh Networks: The WICKPro Protocol

ISSN 2254-7606

ISSN 2254-7606

Tesis Doctoral

WIRELESS REAL-TIME COMMUNICATION IN TUNNEL-LIKE ENVIRONMENTS USING WIRELESS MESH NETWORKS: THE WICKPRO PROTOCOL

Autor

Jesús Aísa Vicén

Director/es

Villarroel Salcedo, José Luis

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2016

PhD Thesis

Wireless Real-Time Communication in Tunnel-like Environments using Wireless Mesh Networks: The WICKPro Protocol

Jesús Aísa Vicén

Supervisor:

José Luis Villarroel Salcedo

Robotics, Perception and Real Time Group (RoPeRT)
Departamento de Informática e Ingeniería de Sistemas (DIIS)
Escuela de Ingeniería y Arquitectura (EINA)
Universidad de Zaragoza (UZ)

December 2016

PhD Thesis

Wireless Real-Time Communication in Tunnel-like Environments using Wireless Mesh Networks: The WICKPro Protocol

Jesús Aísa Vicén

Submitted to *Departamento de Informática e Ingeniería de Sistemas* in partial fulfillment of the requirements for the degree of Doctor of Philosophy

at

Universidad de Zaragoza

December 2016

A mi padre Diego
A mi madre María Jesús
A mi hermano Diego

"Yo soy yo y mi circunstancia,
y si no la salvo a ella no me salvo yo"

José Ortega y Gasset (1883-1955), filósofo y ensayista español

Agradecimientos

Con estas líneas me gustaría agradecer todo el apoyo recibido durante la realización de esta tesis doctoral.

A mi director José Luis por darme la oportunidad de iniciarme en el mundo de la investigación, por sus buenos consejos a lo largo de estos años y por su predisposición a aceptar nuevas ideas. Sin duda, me ha ayudado mucho a mejorar y reforzar mis competencias profesionales y personales.

A mis maestros y profesores que desde la guardería hasta la universidad han velado por mi aprendizaje y mi educación. Son muchos años de alumno hasta la finalización de esta tesis y muchos docentes han contribuido con su esfuerzo a ello.

A Luís Almeida por gestionar las dos estancias de investigación en Suecia y Portugal, así como por el interés mostrado en los dos artículos que hemos podido compartir. Fue un placer y fue muy enriquecedor, técnica y personalmente. Por supuesto, gracias también al *Wireless Communications Group* de la MDH de Västerås (Suecia) y al *Networked Systems Group* de la Universidad de Oporto (Portugal) por su gran acogida. Especialmente, gracias a Hossein por haber podido trabajar juntos y por el conocimiento que con ello he adquirido. Me siento afortunado por haber realizado estas dos estancias de investigación.

Al Grupo de Tecnologías en Entornos hostiles por el apoyo mostrado y por haber compartido proyectos y experiencias. Gracias asimismo al Grupo de Robótica por su ayuda cuando la he necesitado. Mi gratitud a todos los compañeros y amigos de estos dos grupos de investigación, con los que he podido convivir y compartir muchos momentos durante la realización de esta tesis, tanto en los laboratorios de la Universidad de Zaragoza en el Parque Tecnológico WALQA como en el Laboratorio de Robótica Móvil en la EINA.

A mis amigos de Huesca (los de "toda la vida"), a mis amigos de Teleco (los de "la carrera") y a mis amigos de Ingeniería Sin Fronteras Aragón (los "soñadores"). Gracias a todos y cada uno de vosotros, además de a otras personas no clasificables en estos grupos pero que también considero mis amigos. Gracias por los buenos momentos, los viajes, las aventuras, las charradas, los chistes, las locuras... Habéis contribuido enormemente a mi enriquecimiento como persona. Sin vosotros no sabría que existen tantas cosas en el mundo ni tantos tipos de personas.

También creo necesario elevar un gran gracias a la música y al baile porque han sido parte importante en la gestión emocional durante esta tesis doctoral.

Gracias a mi familia que siempre me ha apoyado de manera incondicional. Gracias a mis cinco primos hermanos por esa relación especial que tenemos. Gracias a Ana por su sensibilidad y su escucha, gracias a Fran por su serenidad, gracias a Carlos por su aura, gracias a Fernando Leiva por su clarividencia y gracias a Fernando Latorre por su bondad. Gracias también a mi tía Anita y mi tía María Cruz por preocuparse siempre por mí. Gracias a todos por vuestro cariño y generosidad.

Se acaban las palabras para expresar la inmensa gratitud que siento hacia mi padre Diego y mi madre María Jesús, que siempre han antepuesto sus hijos a ellos mismos. Siempre habéis estado, estáis y estaréis en mi memoria y en mi corazón. Apoyándome y educándome en el respeto. Inculcándome los valores del esfuerzo y de la superación. Gracias también a mi hermano Diego que ha sido otro padre más en muchas partes de mi vida, velando por mí y ayudándome en mis decisiones. Gracias a mi cuñada Carol por su apoyo y su preocupación. Y gracias a mis sobrinos Diego y Mario que personifican la alegría, la ilusión y la inocencia como nadie.

Abstract

Industrial applications have been shifting towards wireless networks in recent years because they present several advantages compared with their wired counterparts: lower deployment cost, mobility support, installation in places where cables may be problematic, and easier reconfiguration. These industrial wireless networks usually must provide real-time communication to meet application requirements. Examples of wireless real-time communication for industrial applications can be found in factory automation and process control, where Radio Frequency wireless communication technologies have been employed to support flexible real-time communication with simple deployment. Likewise, industry is also interested in real-time communication in underground environments, since there are several activities that are carried out in scenarios such as tunnels and mines, including mining, surveillance, intervention, and rescue operations.

Wireless Mesh Networks (WMNs) are promising enablers to achieve wireless real-time communication because they provide a wireless backbone comprised by dedicated routers that is utilized by mobile terminals. However, WMNs also present several challenges: wireless multi-hopping causes inter-flow and intra-flow interferences, and wireless propagation suffers shadowing and multi-path fading.

The IEEE 802.11 standard has been widely used in WMNs due to its low cost and the operation in unlicensed frequency bands. The downside is that its Medium Access Control (MAC) protocol is non-deterministic, and that its communications suffer from the hidden and exposed terminal problems.

This PhD thesis focuses on real-time communication in tunnel-like environments by using WMNs. Particularly, we develop a MAC and network protocol on top of the IEEE 802.11 standard to provide real-time capabilities, so-called WIreless Chain networK Protocol (WICKPro). Two WICKPro versions are designed to provide Firm Real-Time (FRT) or Soft Real-Time (SRT) traffic support: FRT-WICKPro and SRT-WICKPro. We also propose a hand-off algorithm dubbed Double-Threshold Hand-off (DoTHa) to manage mobility in SRT-WICKPro.

WICKPro employs a token-passing scheme to solve the inter-flow and intra-flow interferences as well as the hidden and exposed terminal problems, since this scheme does not allow two nodes to transmit at the same time. This is a reasonable solution for small-scale networks where spatial reuse is impossible or limited. The non-deterministic nature of IEEE 802.11 is faced by combining the token-passing mechanism with a polling approach based on a global cyclic packet schedule. As usual in cyclic scheduling, the hyper-period is divided into minor cycles. FRT-WICKPro triggers the token synchronously and fulfills strictly minor cycles, whereas SRT-WICKPro carries out asynchronous token-passing and lets minor cycles be overrun, thereby decoupling the theoretic and the actual minor cycles. Finally, DoTHa deals with shadowing and multi-path fading. Shadowing is addressed by providing the opportunity of triggering hand-off in the connected and transitional regions of a link, while multi-path fading is neglected for hand-off purposes by smoothing the received signal power.

We tested our proposals in laboratory and field experiments, as well as in simulation. As a case study, we carried out the tele-operation of a mobile robot within two confined environments: the corridors of a building and the Somport tunnel. The Somport tunnel is an old out-of-service railway tunnel that connects Spain and France through the Central Pyrenees. Although autonomous robots are becoming more and more important, technology is not mature enough to manage highly dynamic environments such as reconfigurable manufacturing systems, or to make life-and-death decisions, e.g., after a disaster with radioactivity contamination. Applications that can benefit from mobile robot tele-operation include real-time monitoring and the use of robotized machinery, for example, dumper trucks and tunneling machines, which could be remotely operated to avoid endangering human lives.

Resumen

En los últimos años, las redes inalámbricas se están utilizando cada vez más en entornos industriales debido a sus ventajas respecto a redes cableadas: menor coste de instalación, soporte de movilidad, instalación en lugares donde los cables pueden ser problemáticos y mayor facilidad de reconfiguración. Estas redes inalámbricas normalmente deben proporcionar comunicación en tiempo real para satisfacer los requerimientos de las aplicaciones. Podemos encontrar ejemplos de comunicación en tiempo real con redes inalámbricas para entornos industriales en el campo de la automatización industrial y en el control de procesos, donde redes inalámbricas de radiofrecuencia han sido utilizadas para posibilitar comunicación en tiempo real con un despliegue sencillo. Asimismo, la industria también está interesada en comunicaciones en tiempo real en entornos subterráneos, puesto que existen diversas actividades que se llevan acabo en escenarios tales como túneles y minas, incluyendo operaciones de minería, vigilancia, intervención y rescate.

Las redes inalámbricas malladas (*Wireless Mesh Networks*, WMNs) representan una solución prometedora para conseguir comunicación en tiempo real en entornos inalámbricas, dado que proporcionan una red troncal inalámbrica formada por encaminadores (*routers*) que es utilizada por terminales móviles. Sin embargo, las WMNs también presentan algunos retos: la naturaleza multisalto de estas redes causa interferencias entre flujos e interferencias de un flujo consigo mismo, además de que la propagación inalámbrica sufre *shadowing* y propagación multicamino.

El estándar IEEE 802.11 ha sido ampliamente utilizado en redes WMNs debido a su bajo coste y la operación en bandas frecuenciales sin licencia. El problema es que su protocolo de acceso al medio (*Medium Access Control*, MAC) no es determinista y que sus comunicaciones sufren los problemas del terminal oculto y expuesto.

Esta tesis doctoral se centra en el soporte de comunicaciones en tiempo real en entornos tipo túnel utilizando redes WMNs. Con este objetivo, desarrollamos un protocolo MAC y de nivel de red denominado *WIreless Chain networK Protocol* (WICKPro) que funciona sobre IEEE 802.11. Más concretamente, en este trabajo diseñamos dos versiones de este protocolo para proporcionar soporte de tráfico de tiempo real firme (*Firm Real-Time*, FRT) y de tiempo real no estricto (*Soft Real-Time*, SRT): FRT-WICKPro y SRT-WICKPro. Asimismo, proponemos un algoritmo de *hand-off* conocido como *Double-Threshold Hand-off* (DoTHa) para el manejo de la movilidad en SRT-WICKPro.

WICKPro utiliza un esquema de paso de testigo para solventar las interferencias entre flujos y de un flujo consigo mismo, así como los problemas del terminal oculto y expuesto, dado que este esquema no permite que dos nodos transmitan al mismo tiempo. Esta solución es razonable para redes pequeñas donde el reúso espacial es imposible o limitado. Para tratar la naturaleza no determinista de IEEE 802.11, combinamos el esquema de paso de testigo con una planificación cíclica global. Como es habitual en planificación cíclica, el hiperperiodo es dividido en un conjunto de ciclos secundarios. FRT-WICKPro inicia el paso de testigo de forma síncrona para satisfacer estrictamente dichos ciclos secundarios, mientras que SRT-WICKPro implementa un paso de testigo asíncrono y permite sobrepasar los ciclos secundarios, por lo que desacopla los ciclos secundarios reales de los teóricos. Finalmente, DoTHa lidia con el *shadowing* y la propagación multicamino. Para abordar el *shadowing*, DoTHa permite llevar a cabo el proceso de *hand-off* en la región conectada y en la región de transición de un enlace, mientras que la propagación multicamino es ignorada para el proceso de *hand-off* porque la potencia recibida es promediada.

Nuestras propuestas fueron validadas en experimentos de laboratorio y de campo, así como en simulación. Como un estudio de caso, llevamos a cabo la teleoperación de un robot móvil en dos entornos confinados: los pasillos de un edificio y el túnel del Somport. El túnel del Somport es un antiguo túnel ferroviario fuera de servicio que conecta España y Francia por los Pirineos Centrales. Aunque los robots autónomos son cada vez más importantes, la tecnología no está suficientemente madura para manejar entornos con alto dinamismo como sistemas de fabricación reconfigurables, o para realizar decisiones de vida o muerte, por ejemplo después de un desastre con contaminación radiactiva. Las aplicaciones que pueden beneficiarse de la teleoperación de robots móviles incluyen la monitorización en tiempo real y el uso de maquinaria robotizada, por ejemplo camiones *dumper* y máquinas tuneladoras, que podrían ser operadas remotamente para evitar poner en peligro vidas humanas.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The vision of future factories anticipates human-machine interaction through mobile devices that enhance relevant production information [EFFRA 13]. Indeed, industrial applications have been shifting towards wireless networks in recent years because they present several advantages compared with their wired counterparts: lower deployment cost, mobility support, installation in places where cables may be problematic, and easier reconfiguration [Galloway 13]. These industrial wireless networks usually must provide real-time communication to meet application requirements. Examples of wireless real-time communication for industrial applications can be found in factory automation and process control, where Radio Frequency (RF) wireless communication technologies have been employed to support flexible real-time communication with simple deployment [Willig 08, Da Xu 14].

Industry is also interested in real-time communication in underground and confined environments such as tunnels and mines. There are several activities that are carried out in these scenarios, including mining, surveillance, intervention, and rescue operations. In these environments, communication is essential in both normal and emergency situations [Yarkan 09]. When exploiting an underground environment, day–to–day operations benefit from communication to increase the safety and productivity though remote monitoring, control operations and voice conversations. In emergency conditions, for example in case of accident, communication is vital because it allows the coordination and the location of the workers.

With this in mind, the objective of this PhD thesis is to develop a protocol for wireless multi-hop networks with chain topologies that supports real-time communication. We have a special interest in chain or linear topologies due to their usefulness in tunnel and mines, as well as in infrastructure monitoring, as mentioned below. For this task, Wireless Mesh Networks (WMNs) and IEEE 802.11 wireless cards are employed.

1.2 Problem Statement

In this section, we provide a brief note about computer networks and real-time communication, and introduce WMNs and the IEEE 802.11 standard, along with their advantages and drawbacks.

1.2.1 Computer Networks and Real-time Communication

Telecommunication has been part of our lives at least since smoke signals and drums were employed to communicate over a distance. Telecommunication involves the exchange of information between two or more entities by using technology. Usual entities that take part in this process are human beings and machines. Nowadays, telecommunication, or simply communication, involves many different technologies such as telephone, radio, television and the Internet. The path traveled during last centuries has been full of innovations and revolutions. One of them occurred in the 1970s and 1980s, when the fields of computer science and data communications were merged and as a result the technology, products and companies of the now-combined computer-communications industry were profoundly changed [Stallings 07]. These early computer networks provided best-effort service where the network did not provide any guarantees to the application. However, some applications require not only that the information is conveyed from the source to the destination (logical correctness), but also that the information transmission is carried out in a bounded time (temporal correctness). This is called real-time communication.

1.2.2 Wireless Mesh Networks

WMNs are promising enablers to achieve wireless real-time communication, given that they provide a wireless backbone comprised by dedicated nodes (mesh routers) which is utilized by mobile terminals (mesh clients) to communicate with each other, or access the wired Internet if some routers are set up as gateways [Akyildiz 09]. This architecture is more reliable than other solutions where all nodes can move freely, like in Mobile Ad-Hoc Networks (MANETs). It should be noted that we use the terms mesh router, router and Access Point (AP) interchangeably, as well as the terms mesh client, client, mobile terminal and terminal. The term node is used to refer to both routers and clients.

As commented, a special feature of WMNs is that routers are connected by wireless interfaces, unlike in other wireless networks such as cellular and IEEE 802.11 networks, where base stations and APs are connected by wired infrastructure, respectively. Actually, we can meet some exceptions, e.g., the use of dedicated radio-links to connect base stations in the case of cellular networks where the terrain makes costly wire installation. Anyway, the bottom line is that these wireless networks support one-hop wireless communication while WMNs provide multi-hop wireless networking. And this is positive because wireless multi-hopping helps reduce deployment cost and reconfiguration time.

WMNs are not only employed in applications where routers are fixed, e.g., company or home networking, but also to spontaneous networking [Akyildiz 09]. In this case, the network is specially deployed for a specific purpose, for instance in emergency situation or infrastructure maintenance. An application of the latter is to equip an Unmanned Aerial Vehicle (UAV) with a video camera and send live video of the inspected infrastructure to a remote operator. If the infrastructure to monitor is very large, as in the case of high chimneys, electrical poles, large deposits or long pipelines, it is possible to use a team of UAVs [Pinto 16]. The work in [Pinto 16] deploys a chain network composed by a team of UAVs where the furthest of them is a sensor-UAV that monitors an infrastructure in an outdoor environment. The rest of the UAVs are the relays between the ground station and the sensor-UAV. The deployment maximizes the end-to-end throughput by analyzing the Packet Delivery Ratio (PDR) as a function of distance in every one-hop link. A similar approach is carried out in [Rizzo 13], where a multi-robot team is deployed in an underground tunnel taking into account the link quality. In this work, the base station and the robots also form a linear topology. It is worth noting that in [Pinto 16] and [Rizzo 13], communication is carried out through IEEE 802.11 cards.

However, WMNs present several challenges. As they support communication with several wireless hops, there is potential interference between links carrying different data flows (**inter-flow interference**) and between links supporting the same data flow (**intra-flow interference**). Moreover, unlike in wired networks, signal propagation suffers reflection, diffraction and scattering from obstacles, which in turn will cause path loss variation through **shadowing** and **multi-path fading** [Goldsmith 05].

1.2.3 IEEE 802.11

The IEEE 802.11 standard is a set of physical and Medium Access Control (MAC) layer specifications for Wireless Local Area Networks (WLANs), whose first version [IEEE802.11 97] was published in 1997. Popularly known as Wi-Fi, it has been widely used by the scientific community for three main reasons: (i) it is an standard; (ii) IEEE 802.11 cards represent low-cost Commercial Off-The-Shelf (COTS) technology; and (iii) instead of expensive, licensed spectrum, IEEE 802.11 systems operate in unlicensed bands such as the Industrial, Scientific and Medical (ISM) bands, e.g., 902-928 MHz, 2.4-2.5 GHz, 5.725-5.825 GHz [Tanenbaum 11]. However, unlicensed bands can be used at the same time by devices of different IEEE 802.11 networks as well as by devices of other technologies such as cordless phones and microwave ovens. For this reason, the IEEE 802.11 standard utilizes modulation techniques that tolerate interference, for example, IEEE 802.11b employs Direct-Sequence Spread Spectrum (DSSS) and IEEE 802.11g implements Orthogonal Frequency-Division Multiplexing (OFDM).

Another popular standard is IEEE 802.15.4, which consumes less energy than IEEE 802.11 at the expense of using lower bit rate and achieving lower transmission distance. Therefore, we conclude that IEEE 802.11 is more appropriate for communication in large environments such as tunnels.

1.2.3.1 Network Architecture

IEEE 802.11 networks can operate in infrastructure or ad-hoc mode. In infrastructure mode, each client is associated with an AP in a way that only sends and receives frames via this AP. APs are in turn connected to another network, such as an intranet or the Internet. Moreover, several APs may be connected together, usually by a wired network. In ad-hoc mode, there is no AP and clients can communicate directly. This mode presents therefore a more flexible network topology that can be used to implement wireless multi-hop networks.

1.2.3.2 Medium Access Control

Original IEEE 802.11 [IEEE802.11 97] considers two operation modes: Point Coordination Function (PCF) and Distributed Coordination Function (DCF).

- **PCF**. The AP coordinates the channel access in its cell, thus it is only available in infrastructure mode. The AP defines a contention period, where DCF is actually used, and a contention-free period. In the contention-free period, the AP polls all clients that have requested contention-free services to grant them access to the channel. Although this mechanism could provide bounded-delay communication, it is usually not possible to prevent other devices of nearby networks to transmit competing traffic, or even devices of other technologies. This could explain why PCF has not actually been implemented in practice, given that it is an optional feature.

- **DCF**. In this mode, there is no central control, so each terminal acts independently, either in infrastructure or in ad-hoc mode. For this reason, the IEEE 802.11 standard includes a MAC mechanism that allows to share the wireless medium between uncoordinated clients, e.g., Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA employs channel sensing before sending and exponential back-off after collisions. To illustrate the working of CSMA/CA, let us suppose that a node has a frame to send [Kurose 12]:

  1. The node waits until it senses the channel continuously idle during a time known as DCF InterFrame Space (DIFS). At that moment, if the node has not used the channel recently and the channel was idle since the node started listening, the node transmits the frame and goes on to step 4 to wait for a confirmation packet; otherwise proceeds to step 2 to defer its transmission. This strategy avoids collisions and gives other nodes the opportunity to capture the channel, for example, when a node sends several packet consecutively.
  2. The node selects a random back-off time using binary exponential back-off and counts down this value when the channel is sensed idle. While the channel is sensed busy, the counter value remains frozen. When the counter reaches zero, CSMA/CA continues with step 3.
  3. The node transmits the frame and proceeds to step 4, unless the transmission is broadcast, in which case the transmission of this frame has finished. In this situation, if the node has another frame to send, it comes back to step 1.
  4. The node waits for receiving an ACKnowledgment (ACK) frame. For this reason, the frame destination node sends an ACK frame after the successful frame reception and after sensing the channel idle during the so-called Short InterFrame Space (SIFS) time; SIFS is shorter than DIFS to prioritize confirmation frames. On the one hand, if the ACK is properly received by the sender, this node knows that its frame was correctly received and resets the back-off algorithm. If the node has another frame to send, it comes back to step 1. On the other hand, if the ACK is not received by the sender within a time-out, the transmitting node increases the random back-off (up to a maximum value) and proceeds to step 1 in order to retransmit the frame. This process is followed until the (configurable) maximum number of retransmissions is reached, when the frame is dropped and the back-off algorithm is reset. In this situation, the node comes back to step 1 if it has more frames to send.

CSMA/CA is similar to Carrier Sense Multiple Access with Collision Detection (CSMA/CD)[1], but there are two main differences due to the fact that CSMA/CA cannot detect collisions [Tanenbaum 11]. First, in CSMA/CA, when a node wants to transmit a frame, it starts with a random back-off (except in the case that it has not used the channel recently and the channel is idle). This wait is worthwhile because the entire frame is transmitted even in presence of collisions. Second, the lack of an ACK frame is interpreted as a collision. Thus, as commented above, every time an ACK frame is missed, the random back-off is increased. It is noteworthy that there are two main reasons why IEEE 802.11 cards cannot detect collisions [Kurose 12]: (i) the existence of the hidden terminal problem (explained below); and (ii) the use of half-duplex radios that are unable to send and receive at the same time. Full-duplex radios with collision detection would be costly because in WLAN the received signal has typically much less power than the transmitted signal.

In conclusion, even though CSMA/CA is effective under conditions of light load, it introduces timeliness limitations when the network traffic increases because collisions occur. The bottom line is that CSMA/CA is a **random access MAC scheme**.

An amendment to the IEEE 802.11 standard that defines a set of Quality of Service (QoS) enhancements was published in 2005 [IEEE802.11e 05], so-called IEEE 802.11e. IEEE 802.11e defines a new coordination function called Hybrid Coordination Function (HCF), which replaces DCF and PCF. HCF, in turn, provides two access methods: HCF Controlled Channel Access (HCCA) and Enhanced Distributed Channel Access (EDCA). HCCA also has a coordination point as PCF, while EDCA is similar to DCF in this sense, so the latter is more interesting in wireless multi-hop networks. EDCA includes traffic categories in a way that the higher the traffic priority, the shorter the average time a node has to wait before transmitting a packet of that traffic category. However, although prioritizing traffic is interesting, the MAC scheme is still contention-based.

1.2.3.3 Hidden and Exposed Terminal Problems

In general, not all nodes are within radio range of each other in wireless networks. This even happens in the case of one-hop 802.11 networks in infrastructure mode. In this situation, all clients can communication with the AP, but all of them are not within radio range of each other. In wireless multi-hop networks, such as WMNs, all nodes cannot hear each other, by definition, as routers are used to extend the coverage area. This, together with the use of CSMA/CA, causes the well-known **hidden and exposed terminal problems** [Tanenbaum 11]. The former causes collisions while the latter wastes transmission opportunities.

To address the hidden terminal problem, IEEE 802.11 includes an optional handshake mechanism that is based on a request frame called Request To Send (RTS) and a clearance frame dubbed Clear To Send (CTS). However, the effectiveness of the RTS/CTS mechanism has been shown to be limited in wireless multi-hop networks [Xu 02].

[1]CSMA/CD is the MAC schemed used by IEEE 802.3, popularly known as Ethernet, the most common type of wired local area network.

1.3 The Proposed Approach

1.3.1 Network Description

We consider a WMN comprising N_R routers and N_C clients. Routers form a chain in a way that they have high quality links with their one-hop neighbor routers, and each client is exclusively attached to one router at each instant of time (Fig. 1.1). The latter simplifies the mobility management and the routing algorithm, as there is only one possible route between any two nodes.

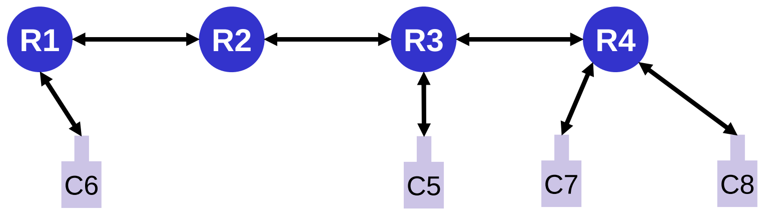


Figure 1.1: WMN with four routers (R1-R4) and four clients (C5-C8)

For the sake of simplicity too, we made the following assumptions:

- before network start-up, the topology is known to all nodes. This can be easily dropped in practice;
- the first router in the chain is the token master (R1 in Fig. 1.1). This node synchronizes the whole traffic scheduling and its updates when nodes join or leave or when communication requirements change. Here we consider fixed schedules known by all nodes. In case of failure, this node should be dynamically replaced;
- there is one radio channel, which is shared by all nodes;
- spatial reuse is not applied.

The network actually presents a tree topology because, from the graph theory perspective, the network is a connected graph that has v vertices and $v-1$ edges. If we only consider router-to-router communication, routers form a chain network, while if we focus on router-to-client communication, clients present a star topology with its serving router.

1.3.2 WICKPro

The design of communication protocols in WMNs has been tackled from different perspectives depending on several issues such as network size, communication requirements, number of channels in the network and number of radio interfaces per device [Akyildiz 09]. Proposals within the Real-time community that aims at supporting real-time traffic usually adapt priority-based or cyclic schedulers to a wireless multi-hop environment. Moreover, when proposals must be implemented in real hardware, two commonly used deterministic MAC schemes are Time Division Multiple Access (TDMA) and token passing. With this in mind, we developed a MAC and network protocol on top of the IEEE 802.11 standard to provide real-time capabilities, so-called WIreless Chain networK Protocol (WICKPro). This protocol implements a token-passing scheme and a cyclic packet scheduler, supports real-time traffic with firm or soft criticality, and operates in WMNs with chain topologies.

WICKPro addresses the problems mentioned above related to IEEE 802.11 and wireless multi-hop networks with the strategies shown in Table 1.1. We propose the use of a token-passing scheme to solve the inter-flow and intra-flow interferences as well as the hidden and exposed terminal problems, since this scheme does not allow two nodes to transmit at the same time. This is a reasonable solution for small-scale networks where spatial reuse is impossible or limited. To face the non-deterministic nature of IEEE 802.11, we combine the token-passing mechanism with a polling approach based on a cyclic packet schedule. This cyclic packet schedule is calculated with global information and defines the whole packet scheduling in the network, in a way that a node can only send a data packet if it holds the token and the transmission is explicitly determined by the cyclic packet schedule. Finally, a hand-off algorithm is developed to manage mobility in presence of shadowing and multi-path fading. Shadowing is taken into account in the hand-off procedure by triggering hand-off in various environmental conditions. For this task, we propose a novel hand-off approach dubbed as Double-Threshold Hand-off (DoTHa) algorithm that considers double threshold level and double hysteresis margin. Although multi-path fading is combated by IEEE 802.11 at physical layer, signal variability is still presented thus DoTHa averages the received signal power to avoid undesired effects such as the ping-pong effect, i.e., situations where a client switches back and forth between two or more routers.

Table 1.1: Problem statement and proposed solutions

Issue	Proposed solution
Inter-flow and intra-flow interferences	Token passing
Hidden and exposed terminal problems	Token passing
Non-deterministic MAC sublayer	Token passing + Cyclic packet scheduler
Shadowing and multi-path fading	Hand-off algorithm

We tested our proposals in laboratory and field experiments, as well as in simulation. As a case study, we carried out the tele-operation of a mobile robot within two confined environments: the corridors of a building and the Somport tunnel. The Somport tunnel is an old out-of-service railway tunnel that connects Spain and France through the Central Pyrenees. Although autonomous robots are becoming more and more important, technology is not mature enough to manage highly dynamic environments such as reconfigurable manufacturing systems [Lindhorst 13], or to make life-and-death decisions, e.g., after a disaster with radioactivity contamination. Applications that can benefit from mobile robot tele-operation include real-time monitoring and the use of robotized machinery, to name a few. In turn, examples of robotized industrial machinery are dumper trucks and tunneling machines, which could be remotely operated to avoid endangering human lives.

1.4 Contributions of the Thesis

The publications related to this PhD thesis are the following:

- [Aisa 10] J. Aisa and Jose L. Villarroel. *WICKPro: A Hard Real-Time protocol for Wireless Mesh Networks with chain topologies.* In European Wireless (EW) Conference, pages 163-170, Lucca, Italy, Apr 2010.

- [Aisa 11] J. Aisa and Jose L. Villarroel. *The WICKPro protocol with the Packet Delivery Ratio metric.* Computer Communications, vol. 34, no. 17, pages 2047-2056, 2011.

- [Aisa 15] J. Aisa and J.L. Villarroel. *Supporting Firm Real-Time Traffic in Fault-Tolerant Real-Time Systems based on Cyclic Scheduling - The WICKPro Protocol.* In IEEE International Symposium on Industrial Embedded Systems (SIES), pages 1-10, June 2015.

- [Aisa 16a] J. Aisa, H. Fotouhi, J.L. Villarroel and L. Almeida. *Soft Real-time Traffic Communication in Loaded Wireless Mesh Networks.* In IEEE World Conference on Factory Communication Systems (WFCS), pages 1-8, May 2016.

- [Aisa 16b] J. Aisa, H. Fotouhi, L. Almeida and J.L. Villarroel. *DoTHa - A Double-threshold Hand-off Algorithm for Managing Mobility in Wireless Mesh Networks.* In IEEE Conference on Emerging Technologies and Factory Automation (ETFA), September 2016.

The main contributions of this PhD thesis are as follows:

- **WICKPro**. We develop the WICKPro protocol to provide real-time capabilities in error-free WMNs with chain topologies [Aisa 10, Aisa 11]. WICKPro implements a cyclic packet scheduler along with a token-passing scheme, where the hyper-period is divided into minor cycles, as usual in cyclic scheduling. WICKPro is validated in laboratory experiments.

- **FRT-WICKPro**. We design a WICKPro version that supports Firm Real-Time (FRT) traffic in error-prone networks, dubbed FRT-WICKPro [Aisa 11, Aisa 15]. FRT-WICKPro triggers the token synchronously in a way that minor cycles are strictly fulfilled. FRT traffic support is achieved due to an off-line cyclic packet scheduler, which reserves time for packet transmissions and retransmissions, and an on-line packet scheduler that drops packets before congestion appears. The required retransmission time to satisfy a minimum percentage of data packet delivery is calculated based on the set of supported data flows, the mean PDR between one-hop neighbors and a memoryless packet loss model. For this reason, we introduce a method to calculate the PDR metric in WICKPro. We test FRT-WICKPro in laboratory and field experiments, as well as in simulation.

- **SRT-WICKPro**. We also propose and test a WICKPro version for supporting Soft Real-Time (SRT) traffic in error-prone networks, so-called SRT-WICKPro [Aisa 16a]. SRT-WICKPro carries out asynchronous token-passing and lets minor cycles be overrun, thereby decoupling the theoretic and the actual minor cycles. This protocol does not allocate explicitly time for retransmissions in the cyclic schedule, and allows the response time to grow beyond the deadline employed by the scheduler because SRT applications still find some value in these packets. This strategy increases the throughput, although also the average delay.

- **DoTHa**. We develop the DoTHa hand-off algorithm and implement it in SRT-WICKPro to manage client mobility [Aisa 16b]. We take advantage of the token rotations to continuously collect Received Signal Strength Indicator (RSSI) measurements and carry out on-the-fly hand-off. This probing scheme does not require extra overhead because the token packet is periodically passed to all nodes. Moreover, DoTHa can trigger hand-off in the connected and transitional regions of a link, thereby considering different levels of link quality. DoTHa and SRT-WICKPro are tested in simulation and laboratory experiments, as well as in field experiments in the corridors of a building and in the Somport tunnel, where a mobile robot is tele-operated using a WMN with chain topology.

1.5 Structure of the Thesis

This first chapter presented the motivation and the scope of this work. The rest of this dissertation is organized as follows.

Chapter 2 provides preliminary information about real-time communication, QoS metrics and wireless propagation.

Chapter 3 presents proposals that support real-time communication in wireless networks. We review the literature with special emphasis on protocols that support FRT or SRT traffic, employ IEEE 802.11 and provide wireless multi-hop networking.

Chapter 4 describes the basic working of the WICKPro protocol: protocol states, protocol packets, connection establishment and termination, network layer and MAC sublayer (medium access, error control, packet scheduler). Laboratory experiments are carried out to show the behavior of the proposed protocol in error-free scenarios.

Chapter 5 introduces a fault-tolerant framework to support FRT traffic in the WICKPro protocol. Basically, the packet scheduler is modified to allocate time for packet retransmissions. A method for calculating the PDR is also designed and incorporated to FRT-WICKPro. FRT-WICKPro is verified in simulation, laboratory and field experiments in the corridors of a building.

Chapter 6 presents another version of WICKPro that supports SRT traffic. The working of SRT-WICKPro is evaluated in error-prone scenarios through simulation and laboratory experiments.

Chapter 7 provides related work in mobility management, and shows the DoTHa hand-off algorithm as well as its working along with SRT-WICKPro. DoTHa and SRT-WICKPro are tested in the corridors of a building and in the Somport tunnel.

Chapter 8 concludes the dissertation and proposes research lines for future work.

Finally, Appendix A details the packets employed by WICKPro to implement the functionalities presented in Chapters 4-7, and Appendix B provides photographs of the hardware and the scenarios involved in the laboratory and field experiments of this PhD thesis.

2 Background

This chapter introduces basic knowledge, assumptions and terminology useful for this PhD thesis. The information is specially focused on real-time communication, QoS metrics and wireless propagation.

2.1 Real-Time Communication

In real-time computer systems, the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced [Kopetz 11]. Real-time communication systems manage therefore time-sensitive traffic that, in addition, can suffer from packet losses. For these reasons, it is essential to carry out a schedulability analysis at design time to verify the fulfillment of message deadlines. In this section, we introduce the real-time traffic model employed in this PhD thesis, as well as some ideas about how to design a fault-tolerant real-time system.

2.1.1 Real-Time Traffic Model

We consider time-triggered communication because it is well suited for control applications that require periodic transmissions such as tele-control. We assume a set of N_{MH} multi-hop flows $MH = \{MH_1, ..., MH_{N_{MH}}\}$. Every MH_i, in turn, is partitioned in HC_i one-hop flows $OH_{ij} = \{OH_{i1}, OH_{i2}, ..., OH_{iHC_i}\}$ where HC_i is the hop count of the route between the source node and the destination node of flow i. These one-hop flows have precedent relationships, i.e., OH_{i1} must be transmitted before OH_{i2}, OH_{i2} before OH_{i3}, and so on. The total number of one-hop flows in the network is $N_{OH} = \sum_{i=1}^{N_{MH}} HC_i$. Table 2.1 summarizes this nomenclature.

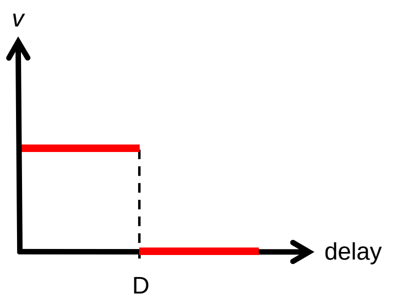
Table 2.1: Real-time traffic model notation. General parameters

Parameter	Meaning
MH	Multi-hop flow
OH	One-hop flow
N$_{MH}$	Number of multi-hop flows in the network
N$_{OH}$	Number of one-hop flows in the network

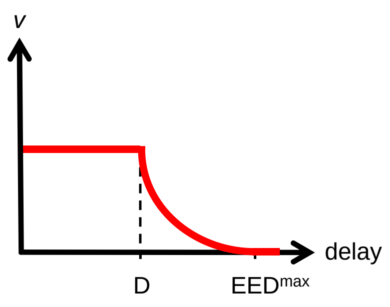
Table 2.2: Real-time traffic model notation. Multi-hop flow features

Parameter	Meaning
src	Source node of a multi-hop flow
dst	Destination node of a multi-hop flow
HC	Hop count between source and destination, or number of one-hop flows that has a multi-hop flow
C	Transmission time of a packet, including transmission-related overheads
P	Period of packet generation. If the traffic is sporadic, this is actually the packets minimum inter-arrival time
D	Relative deadline
F	Fault recovery time or retransmission time
v	Utility function that weights the value of a delivered packet depending on its delay. For SRT traffic, we define EED^{max} as the maximum End-to-End Delay that makes a packet be useful for the application
DDR	Data Delivery Ratio: minimum percentage of data packets that must be delivered from source to destination
L	Criticality level
R	Worst-case response time: maximum difference between the packet arrival time at destination node and the packet generation time at source node
B	Worst-case blocking time: delay produced in a multi-hop flow due to the transmissions of other flows different than the current flow
J	Release jitter: deviation from exact periodic release, i.e., the worst-case time a packet can spend waiting to be released after being generated

The parameters that characterize a multi-hop flow are shown in Table 2.2. We assume implicit deadline (D=P) and consider that the criticality level is firm or soft[1]. Fig. 2.1 depicts utility functions v of traffic with firm and soft criticality level [Buttazzo 11]. As shown, applications support FRT traffic if a packet received after its deadline is useless but it does not cause any damage, whereas traffic is SRT if a packet received with a delay higher than its deadline has still some value for the application. Specifically, packets are worthless if their delay is higher than the so-called EED^{max} value. Thus, we can state that D is the deadline for the packet scheduler whereas EED^{max} is the deadline for the SRT application.



(a) Firm real-time traffic



(b) Soft real-time traffic

Figure 2.1: Example of utility functions for real-time traffic

[1]It is considered to be hard in Chapter 4 because the network is considered error-free, but this constraint is relaxed when packet losses are taken into account.

We also assume that the set of supportable data flows is known beforehand because in this way we can analyze its schedulability previously. In particular, we define four data packet types when implementing WICKPro, so we can support up to four different data flow types simultaneously, although it could be increased if required. Let us illustrate this with the following example. Assume that WICKPro can support up to 20 voice conversations whenever they employ the same audio codec, since they have the same communication requirements and belong therefore to the same data type. However, WICKPro cannot support at the same time five voice conversation that have different communication features. It should also be noted that this assumption helps measure the PDR because WICKPro computes this metric individually for every packet size.

2.1.2 Fault-tolerant Approach

Fault-tolerant real-time systems are designed to operate properly in case of error, although possibly with lower performance. Some proposals in the literature carry out a schedulability test based on a fault model to check if the system is schedulable or not. Conversely, the framework presented in [Burns 99] provides probabilistic scheduling guarantees. It is also based on a fault model but calculates the probability of all deadlines being met during a given period of time.

We employ the probabilistic scheduling guarantees paradigm to support FRT traffic in FRT-WICKPro. In our framework, a fault is an erroneous packet reception and, although packet losses can be counterbalanced by retransmissions, this can jeopardize the deadlines of the supported traffic and it is therefore necessary to take into account retransmissions in the schedulability analysis.

2.2 Quantifying Quality of Service

In real-time communication, applications require metrics to evaluate the QoS level that the network is providing, whereas the network and link layers employ metrics to quantify the link quality and meet the QoS requirements of the applications. Thus, in this section we show performance metrics that measure the QoS level provided by the network and metrics that characterize link quality.

2.2.1 Performance Metrics

Performance metrics are usually measured on an end-to-end basis.

Throughput The number of bits per second that can be transmitted in practice over a link or set of links is known as throughput [Peterson 11]. Particularly, real-time applications demand an end-to-end throughput fulfillment. Note that the theoretic number of bits per second that can be transmitted on a link is called bit or transmission rate (R_b).

Delay The delay or latency corresponds to how long it takes a message to travel from one place of a network to another [Peterson 11]. As with throughput, delay can be defined in a single link or an end-to-end path. In the latter case, we can talk about End-to-End Delay (EED). Real-time applications requires that packets are delivered in a bounded time.

Data Delivery Ratio (DDR) The DDR is the percentage of data packets delivered from source to destination while satisfying timing constraints, and it is independently measured for every data flow. Real-time applications sometimes demand a minimum DDR fulfillment, but this requirement is actually equivalent to the end-to-end throughput. It should be noted that packet retransmission can increase the obtained DDR.

2.2.2 Link-layer and Physical-layer Metrics

Link-layer and physical-layer metrics are normally measured on a link-by-link basis. Link quality estimation is a critical issue in wireless networks that support real-time communication. Several mechanisms benefit from link quality knowledge, namely traffic scheduling, load-balancing, power-control, transmission rate selection, mobility management and routing. Next we present four typical metrics used in IEEE 802.11 networks [Vlavianos 08].

Received Signal Strength Indicator (RSSI) The RSSI is a dimensionless[2] measurement that represents the signal strength of received packets. It is measured only during the reception of a packet preamble, which is transmitted at the lowest rate of the standard in question. This, along with the fact that the RSSI cannot capture interferences, makes RSSI ineffective in accurately characterizing the link quality, especially at high transmission rates. However, it has been successfully used as a stand-alone metric [Tardioli 15, Fotouhi 14] and in combination with other metrics [Baccour 10]. There are two reasons for this: (i) the RSSI works properly for mechanisms such as mobility management under certain conditions, for example low external traffic; and (ii) the RSSI is available in commercial IEEE 802.11 cards, as well as in other wireless technologies.

Signal-to-Interference-plus-Noise Ratio (SINR) The SINR represents the extent to which the power of the received signal exceeds the sum of noise plus interference at the receiver. The SINR is an accurate predictor of link quality but it is hard (if not impossible) to measure the exact SINR with IEEE 802.11 cards.

Packet Delivery Ratio (PDR) The PDR is the ratio of the correctly received packets at the receiver to the total number of packets sent by the sender. It is influenced by interference and represents a good metric for characterizing link quality at a coarse-grained level. However, it is highly dependent on the packet size and the transmission rate, and cannot be accurately estimated by means of the RSSI. It should be noted that PDR is sometimes referred as a network-layer metric because it is directly measured by the routing algorithm, or sometimes as a performance metric because it is used to evaluate the QoS level, in which case it is typically measured on

[2]RSSI can also be expressed in Watts (usually dBm) by transforming the signal strength percentage into dBm values through a conversion function that is chipset-dependent. We employ Atheros chipsets whose RSSI measurement ranges from 0 to 60, and can be converted to dBm by subtracting 95, thus RSSI in dBm ranges from -95 dBm to -35 dBm.

an end-to-end basis. This said, in this work we consider the PDR as a link quality metric measured in each link, while we employ the DDR as a performance metric that is computed for every data flow on an end-to-end basis.

Bit Error Rate (BER) The BER provides a fine-grained indication of the quality of a link but it depends on the transmission rate. Other drawbacks are that repeated computations of this metric are required over extended periods of time and that one needs to ensure that outliers do not result in biased BER results. Moreover, it can give a bad estimation if the channel is very hostile because assuming that the lost packets have all their bits in error can distort the computed BER.

2.3 Wireless Channel Characteristics

We assume a path loss model with three contributions [Goldsmith 05]: (i) mean path loss, which is produced by dissipation of the power radiated by the transmitter as well as effects of the propagation channel, and is the same at a given transmit-receive distance; (ii) shadowing, which is caused by obstacles between the transmitter and receiver that absorb power; and (iii) multi-path fading, which is the variation due to the constructive and destructive addition of multi-path signal components. All these problems are also increased due to environment dynamism and terminal mobility. Moreover, the wireless medium is shared thus prone to interferences, and presents frequent packet losses and even periods of no communication. Conversely, these phenomena are not produced in wired networks and therefore the PDR in wireless networks is in general considerably lower than in wired networks. Particularly, the BER is usually at least two or three orders of magnitude higher in wireless than in wired networks.

2.3.1 Wireless Link Regions

Using a high-level description, a wireless link is usually classified in three different regions: connected, transitional and disconnected [Zuniga 04]. In the connected region, links are stable with high quality that results in PDR close to 100%. Conversely, the transitional region is characterized by the presence of unreliable and unstable links, where the PDR varies between 0 to 100%. In the disconnected region, the PDR is close to 0% and thus there is no wireless connectivity.

An example of variation of PDR with respect to RSSI is shown in Fig. 2.2. It is important to note that the values where links change from one region to another depend on the employed modulation and codification schemes, as well as the wireless chipset and the environment. In our experiments with RSSI measurements, we employed Ubiquiti Networks SRC 802.11 a/b/g PCMCIA cards based on the Atheros 5004 chipset. When using 802.11a at 6 Mbps, the boundary from the connected to the transitional region is −70 dBm, based on empirical studies. The threshold from the transitional to the disconnected region is −94 dBm, with a tolerance of +/-1 dB, based on the sensitivity of the card data sheet [Ubiquiti 16].

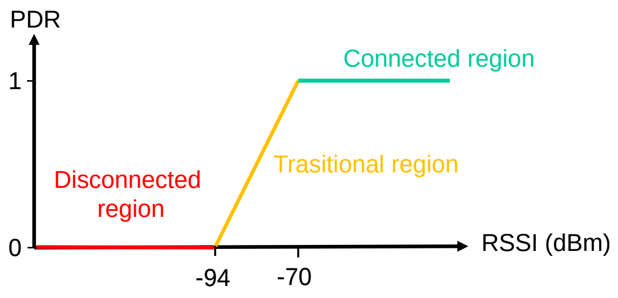


Figure 2.2: Example of transitional region based on PDR with respect to RSSI

2.3.2 Packet Loss Models

There is a great number of packet loss models that try to characterize accurately the wireless medium. Packet loss models provide the fault model required for designing a fault-tolerant communication system. Therefore, we incorporate a packet loss model in FRT-WICKPro to support FRT traffic, but we also employ packet loss models in this PhD thesis to evaluate WICKPro in error-prone scenarios. Next we introduce two simple packet loss models that have been widely used in the literature:

- **Memoryless packet loss model**. In this model, packet losses follow an independent and identically distributed model, thus this model is only characterized by the PDR value.

- **Bursty packet loss model**. In this work, we consider the Gilbert-Elliott Model, specifically the simple Gilbert Model that takes into account a two-state Markov chain: in the good state all packets are properly received whereas in the bad state all packets are lost. As observed in Fig. 2.3, the transition probability from good to bad state is p and the transition probability from bad to good state is q. In this way, the mean Packet Loss Rate (PLR)[3] is $p/(p+q)$ and the Average Burst Length (ABL) is $1/q$.

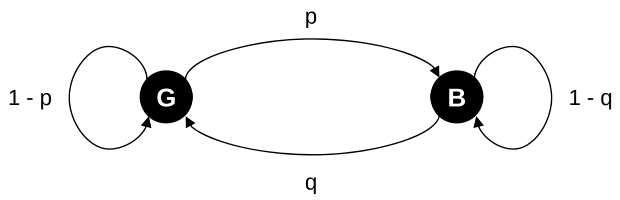


Figure 2.3: Simple Gilbert Model for modeling bursty packet losses

[3]PLR is the complement of PDR such that PLR = 1 - PDR.

3 Related Work in Real-time Communication

In this chapter, we review and discuss proposals for real-time communication in RF wireless networks. To provide real-time communication, it is essential to determine the order and the moment in which traffic must be transmitted. This is called traffic scheduling, and we place special emphasis on it. Traffic scheduling is implemented at the link layer, through the MAC scheme and the local queuing policies, and at the network layer. Indeed, as traffic scheduling (or simply scheduling) may be split into link and packet scheduling [Jayachandran 10], we use this division to classify the presented approaches. Specifically, link scheduling relates to determine a feasible set of links on which to transmit packets without interference, whereas packet scheduling refers to define which packets should be transmitted at any given time in each link.

We are mainly interested on schemes that support FRT or SRT traffic, employ IEEE 802.11 technology and provide wireless multi-hop networking. Moreover, for the sake of clarity, it should be highlighted that we are not interested in the following approaches: (i) routing algorithms, since there is only one possible route between any two nodes in our network, as stated in Section 1.3.1; (ii) protocols that consider a traffic model different than the presented in Section 2.1.1, for example, bursty traffic models; and (iii) proposals that satisfy a single QoS constraint, e.g., maximize the throughput or minimize the end-to-end delay.

3.1 Protocols based on Fixed Link Scheduling

When all network nodes are fixed, some protocols propose to assign a predefined resource (frequency, time slot, etc.) to every link in the network. These proposals analyze the interference pattern and maximize the channel reuse in space. This approach has been widely used in the design of radio-links, and it is now applied to the routers of a WMN with chain topology by Ripple [Cheng 06], RMP [Guo 07] and TDS [Hou 06]. These approaches consider IEEE 802.11 cards and has only been simulated in an error-free scenario.

Ripple Ripple [Cheng 06] is a token-passing protocol that achieves a channel utilization of 1/3. The token is managed in a way that each node has a fixed time to transmit, like a slot in TDMA. Ripple is simulated considering a one-hop interference model, but a two-hop interference model would be more realistic, since it is typical to consider that the interference range is more than twice the transmission range when using IEEE 802.11 [Guo 07]. In such a case, a node would achieve a channel utilization of 1/4. Let us illustrate this with the example of Fig. 3.1, where there are six routers (R1-R6). Transmissions of R5 interfere the receptions of R3 but they do not disturb the receptions of R2, thus R1 and R5 can transmit in the same slot. As the token handles these slots, it is necessary to maintain certain synchronization between nodes to avoid interferences.

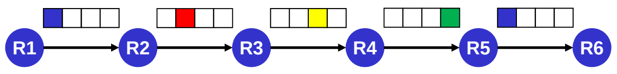


Figure 3.1: Four-slot allocation scheme used by Ripple and TDS in a chain network

Radio-Matching Protocol (RMP) In [Guo 07], RMP is introduced, a protocol that adopts spatial reuse and where a node achieves a channel utilization of 1/3. RMP is simulated considering a two-hop interference model. Like Ripple, it is also based on fixed slots but it organizes the slots in such a way that it represents an improvement on Ripple. Fig. 3.2 shows how RMP allocates slots in a chain network with 3 slots and six routers. In RMP, a node mounts two radios which use non-interfering channels so that it really achieves a channel utilization of 2/3. However, for a fair comparison, we only consider the unidirectional throughput, which is 1/3. Instead of using a token, RMP starts working with all the nodes in idle state except the first node in the chain. When the first node sends a packet to the last node, RMP assigns a slot to each node in the chain. As with Ripple, it is necessary to maintain certain synchronization between nodes to avoid interference.

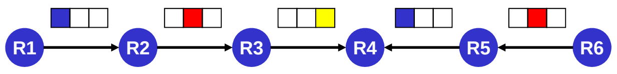


Figure 3.2: Three-slot allocation scheme used by RMP in a chain network

Token-based Distributed Scheduling (TDS) [Hou 06] presents TDS, a token-passing protocol that adopts spatial reuse and where a node achieves a channel utilization of 1/4. TDS is simulated considering a two-hop interference model. Like Ripple and RMP, it is based on fixed slots and organizes the slots in the same way as Ripple. Likewise, the token manages the start and the end of each slot. TDS provides proportional throughput allocation but it is quite limited. For example, in a four-node chain network it is possible to allocate a proportion equal to 1:2:1:1, i.e., node two has a slot twice as long as nodes one, three and four. If there are more subnets in the chain network, all of them must have the same allocation scheme to avoid interference. As with Ripple and RMP, it is necessary to maintain certain synchronization between nodes to avoid interference.

Discussion Ripple, TDS and RMP, each in its own way, implement a fixed TDMA-like scheduling. This scheme presents several advantages:

- high channel utilization due to their fixed link scheduling with spatial reuse;
- fairness;

- fixed airtime delay, which depends on the slot size and the number of hops of the end-to-end communication.

The throughput depends on the relation between the packet size and the slot size. For this reason, Ripple is enhanced to manage different data packet sizes [Nguyen 09].

Nevertheless, these static TDMA protocols also have several drawbacks:

- They do not adapt to the network load. TDS provides a service differentiation but this is quite limited, while Enhanced Ripple changes the slot duration to accommodate different packet sizes. These solutions alleviates this problem, but the link scheduling is still rigid because the spatial reuse pattern must be respected.

- No packet scheduler is considered to carry out a schedulability analysis taking into account the supported data flows, so they cannot guarantee real-time communication.

- Synchronization is required to avoid interference. These approaches have not been implemented in real environments and it is therefore unclear how clock synchronization will be achieved in such cases. Moreover, clock synchronization is still more difficult to attain in confined areas due to the lack of a central clock to synchronize all nodes, e.g., via Global Positioning System (GPS). In this situation, synchronization must be distributed and achieved through the exchange of clock information between nodes, using for example the Precision Time Protocol (PTP) described in IEEE 1588.

- Communication with clients is not considered.

3.2 Protocols based on Dynamic Link Scheduling

We now deal with approaches that change the link schedule depending on the network topology and the supported traffic. Some of the presented proposals do not consider a coordinated packet schedule, whereas others implement a priority-based or a cyclic packet scheduler.

3.2.1 Uncoordinated or Non-existent Packet Scheduling

In wireless networks, some protocols define queuing policies in every node but they do not carry out a coordinated working. Or sometimes they do not define explicitly any packet scheduler. This is the case of some dynamic TDMA slot assignment protocols that pay mainly attention on the following three types of mechanisms: (i) off-line link scheduling or full slot allocation, avoiding hidden and exposed terminal problems while maximizing spatial reuse; (ii) on-line link scheduling or rerouting, which consists on changing part of the link scheduling to improve the QoS provided by the network; and (iii) QoS adaptation, i.e., adjustment of the application requirements to adapt to the QoS provided by the network at a given moment. QoS adaptation is employed with the idea that the decrease of the end-to-end link quality is temporal. This kind of approaches are usually design to support SRT traffic, e.g., audio and video streaming.

Light-Weight TDMA MAC (LiT MAC) An interesting multi-hop TDMA protocol that has been tested in a real-world scenario using IEEE 802.11 cards is LiT MAC [Sevani 14]. It has a root node which receives all the global state information (link quality, node failure, etc.) in a way that it calculates the slot allocation and the packet routes, and sends this information to the remaining nodes. Spatial reuse is taken into account based on RSSI, while clock synchronization is achieved in a distributed way through the exchange of clock information between nodes. LiT MAC achieves synchronization with an average error of about 10 μs at seven hops of distance by using standard IEEE 802.11 hardware and Linux-based operating systems. However, flows are allocated in slots based on customized rules where the traffic features are not considered, thus no real-time packet scheduling is actually taken into account. Instead, this work is focused on achieving distributed synchronization and integrating LiT MAC with a routing algorithm while applying spatial reuse in space.

3.2.2 Priority-based Packet Scheduling

In this section, we introduce two protocols that implement arbitrary Fixed Priorities (FP), namely RT-WMP and ISRA, and one that employs implicit Earliest Deadline First (EDF). The three protocols support FRT traffic, however, ISRA carries out a fault-tolerant schedulability analysis where time for retransmissions is allocated, whereas the protocol based on implicit EDF and RT-WMP accomplish a schedulability analysis in an error-free scenario.

Protocol based on Implicit EDF (Implicit-EDF) In [Facchinetti 05], it is presented a TDMA protocol that implements the implicit EDF algorithm. In implicit EDF, all nodes execute in parallel an EDF queue with all the supported flows in the network. All local EDF schedulers work as a single global EDF algorithm, thus tight clock synchronization is required to avoid collisions. A consensus algorithm is employed to report on scheduling updates, for example after packet losses or topology changes. The consensus must be designed carefully because the number of consensus messages can become significant. Link quality is characterized by a binary metric that indicates the existence of connectivity or not, which causes that the network topology is inaccuracy. This protocol has been successfully simulated to verify its performance. However, a real implementation is challenging due to its synchronization requirements.

Real-Time Wireless Multi-hop Protocol (RT-WMP) Specifically designed for MANETs, RT-WMP [Tardioli 15] implements a token-passing scheme and a global FP schedule. Every message is given a priority level in the [0, 127] range, where 127 is the highest priority value. Messages with the same priority are stored in First In, First Out (FIFO) order. RT-WMP is implemented over IEEE 802.11 and works in three phases. In the first phase, the token is passed through all nodes in a way that nodes fill the token with the priority of the message they want to send, so that the Most Priority Message (MPM) is figured out. In the second phase, the token is sent directly to the node with the MPM whereas in the third phase the multi-hop message is actually transmitted. It should be noted that the first and the second phases are necessary because the information about message priorities is distributed throughout the network. But since these phases are presented, RT-WMP takes advantage to collect RSSI information for routing and topology changes; however, this information exchange may be reduced in WMNs.

Integrated Scheduling and Retransmission Approach (ISRA) A protocol based on IEEE 802.11e called ISRA is introduced in [Demarch 07], where packets are dispatched based on an arbitrary FP scheduler. ISRA modifies the polling scheme employed by the coordinated operation mode of IEEE 802.11e, namely HCCA. Specifically, ISRA calculates the additional number of retries that must be performed to achieve a specific DDR, assuming a memoryless packet loss model. Afterward, the required extra time is scheduled to enable FRT traffic support. This algorithm is centralized because the decision whether a frame should be scheduled for transmission or retransmission is taken entirely by the AP. It is therefore designed for one-hop wireless networks based on a single AP, which is its main drawback, together with the fact that it has only been simulated.

3.2.3 Cyclic Packet Scheduling

Unlike the idea of cyclic link scheduling that has been widely exploited by TDMA protocols that assign slots within a TDMA frame cyclically, few cyclic packet schedulers have been employed in wireless networks. Next we present a token-passing protocol (WTRP) and a TDMA protocol (IsoMAC) that precisely implement a cyclic packet scheduler.

Wireless Token Ring Protocol (WTRP) One of the first token-passing protocols for IEEE 802.11 wireless networks is WTRP [Ergen 04]. WTRP provides communication with bounded latency and reserved bandwidth while nodes maintain a logical ring topology. For this task, it defines a maximum time that nodes can transmit when they hold the token, as well as a maximum number of nodes per ring. In this way, a maximum Token Rotation Time (TRT) is enforced and real-time guarantees can be achieved. To increase the number of nodes in the network, it is possible to define several rings that work in non-interfering channels. However, WTRP does not implement a real-time packet scheduler where traffic is dispatched based on its real-time features.

Isochronous MAC (IsoMAC) [Trsek 11] presents IsoMAC, a protocol designed to support SRT traffic in one-hop wireless networks where the different APs are connected by wired links. It employs a TDMA round on top of IEEE 802.11 to schedule periodic and aperiodic traffic. Particularly, IsoMAC calculates a cyclic schedule for the periodic traffic, whereas a polling server is used for the aperiodic traffic. The TDMA round is divided into a scheduled phase for both periodic and aperiodic real-time traffic, and a contention phase for best-effort traffic and management information, such as messages for clock synchronization. The slots of the contention phase can also be used to accommodate retransmissions of lost real-time data frames. Although mobility is not explicitly managed in a way that mobile nodes select dynamically their serving AP, the scheduler takes into account the interference between the different APs, thus mobility support would be readily achieved. It should be noted that its packet scheduling is explained in detail in [Toscano 10].

3.3 Discussion

To provide an overall perspective, Table 3.1 shows a qualitative comparison of the reviewed protocols that we expand below. Since the revised proposals are representative, our conclusions are quite general, however, it could certainly be possible to find counterexamples for our statements.

Link Scheduler and Mobility

The main conclusion is that computing a link schedule is complex because the wireless medium is shared among all nodes and consequently links cannot be scheduled separately due to interference. The link scheduler can be centralized, decentralized or distributed, and can calculate fixed or dynamic schedules. However, mobility management requires the use of dynamic link schedules. More details about mobility strategies are presented in Section 7.3.

Packet Scheduler

As commented in [Jayachandran 10], local schedulability of deadlines does not necessarily guarantee that there exists a feasible global scheduling of all the packets. Therefore, it is desirable that the packet scheduler carries out a coordinated scheduling in a centralized, decentralized or distributed way. We now revisit some of the presented proposals to address this issue. First, LiT MAC uses a **centralized** algorithm which runs in a root node. The root node holds global state information, calculates the packet schedule and disseminates it to all nodes. We must recall that LiT MAC does not actually incorporate a real-time packet scheduler but it provides an illustrative example of how centralized algorithms work. Second, RT-WMP uses a **decentralized** approach where nodes collect global state information to be able to calculate a global packet schedule. A decentralized algorithm is also used by the Implicit-EDF protocol. Third, there are proposals that implement a **distributed** packet scheduler. For example, a distributed EDF algorithm is introduced in [Jayachandran 10] for wireless multi-hop networks. However, this protocol assumes a bursty traffic model and minimizes the end-to-end delay, thus it is not applicable directly to face our problem.

In small networks, decentralized algorithms are a good solution since the global state information to hold is reduced. Centralized algorithms can also be implemented but they are less flexible than decentralized solutions and present single point of failure. Distributed algorithms are, of course, very interesting because they require to maintain less information and are more easily scalable. However, they are in general more complex to develop.

It is also interesting to analyze the use of decentralized **cyclic** and **priority-based** packet schedulers in wireless multi-hop networks. First of all, we study the required control information to implement every strategy. As commented earlier, RT-WMP implements a priority-based packet scheduler and employs token packet exchanges to know the most priority message in the network at a given moment. Once this is known, the token is sent to the node that must send the data packet. Instead, WTRP passes the token periodically to every node and allows them to transmit for a maximum time, if required. Otherwise, nodes just pass the token to their

Table 3.1: Qualitative comparison of the reviewed protocols. All schemes are implemented using IEEE 802.11

Protocol	Link scheduler	Packet scheduler	Medium access	Real-time support	Fault-tolerant design	Mobility	Synchronization	Network type
Ripple	Fixed	Not defined	TDMA	No	No	No	Yes	WMN (chain)
RMP	Fixed	Not defined	TDMA	No	No	No	Yes	WMN (chain)
TDS	Fixed	Not defined	TDMA	No	No	No	Yes	WMN (chain)
LiT MAC	Dynamic	Not defined	TDMA	No	No	No	Yes	WMN
Implicit-EDF	Dynamic	Priority-based (EDF)	TDMA	FRT traffic	No	Yes	Yes	MANET
RT-WMP	Dynamic	Priority-based (FP)	Token passing	FRT traffic	No	Yes	No	MANET
ISRA	Dynamic	Priority-based (FP)	Polling	FRT traffic	Yes	No	No	Single AP
WTRP	Dynamic	Cyclic	Token passing	SRT traffic	No	Yes	No	WMN (ring)
IsoMAC	Dynamic	Cyclic	TDMA	SRT traffic	No	Yes	Yes	Based on APs

successor node. Thus, cyclic schedulers employ lower control traffic than priority-based schedulers if they are implemented together with token-passing schemes. In TDMA protocols, the conclusion should be the same because information is also distributed, in which case priority-based packet schedulers must update the global information every packet loss[1] or even before every packet transmission, whereas cyclic packet schedulers only need update the global information when there are changes in the communication requirements. Drawing a parallel with monoprocessor real-time systems, this overhead is equivalent to the context switch.

Moreover, cyclic schedulers fit properly to schedule periodic traffic. However, they present several drawbacks [Buttazzo 11]. Aperiodic traffic is not easily handled, cyclic schedules are fragile during overload conditions, the scheduling computation is not always obvious, and the complete scheduling must be recalculated when there are topology or application changes.

Medium Access: TDMA vs Token-passing Schemes

Controlled-access MAC mechanisms are desirable to provide real-time guarantees because they provide a conflict-free medium access and network resources can therefore be reserved. We focus specially on token-passing and TDMA schemes because they provide controlled access to the wireless medium and are simpler to implement than other conflict-free schemes such as Code Division Multiple Access (CDMA).

- **Implementation of link and packet schedulers**. The implementation of both link and packet schedulers are dependent on the MAC protocol. Some proposals such as RT-WMP does not actually differentiate between link and packet scheduler. This is usual in token-passing protocols because only one node can transmit at a given moment. Conversely, TDMA protocols usually calculate an interference-free slots assignment and the packet schedule is then restricted to use this slot allocation.

- **Flexibility**. TDMA mechanisms employ implicit control to define transmission instants whereas token-passing schemes provide explicit indication through the token packet. Token-passing protocols are therefore more flexible than TDMA approaches, which typically use predefined tight schedules. If a slot is not used, it remains idle. Conversely, in token-passing protocols such as WTRP, nodes can transmit until a maximum time (similarly to the slot duration), but they pass the token to their successor node if they do not have any packet to transmit, thus only part of the maximum token holding time is wasted. The price for this is the amount of required control information, as analyzed below.

- **Retransmission management**. Due to its flexibility, packet losses are easily handled by token-passing protocols that can retransmit packets when necessary. In any case, it is necessary to consider that immediate retransmissions may cause a domino effect on the other scheduled transmissions. TDMA protocols address retransmissions in different ways. Some of them retransmit lost packets in the next allocated slot. In this case, if a data packet is lost

[1]Implicit-EDF tries to avoid this information exchange by tightly synchronizing all nodes, but information control is also required when packet losses are produced to report scheduling updates.

within a multi-hop flow, the subsequent nodes of the multi-hop flow will not have anything to send in their allocated slots. To address this problem, CR-TDMA [Lee 14] proposes to allocate packets from other flows in these idle slots. Other TDMA protocols such as RT-WiFi [Wei 13] employ in-slot retransmission, i.e., a slot has a duration enough to accommodate a packet transmission and a retransmission, as well as two ACK packets. If this time is scheduled in every slot, more packet deadlines will be satisfied at the expense of throughput decrease. Actually, this problem may happen anyway when TDMA slots are larger than the time required to transmit the respective packets in error-free conditions, reducing efficiency. A third type of strategy consists of using common slots to accommodate retransmissions. An approach for one-hop wireless networks is presented in IsoMAC.

- **Synchronization**. Typically, TDMA schemes require clock synchronization, whereas token-passing protocols neglect the use of synchronization. However, some TDMA protocols do not use clock synchronization, such as [Oliveira 15], but they are relatively slow in tracking dynamic topologies or dynamic communication requirements. Specifically, [Oliveira 15] carries out loose synchronization based on packet receptions. As already commented in Section 3.1, synchronization in confined environments is more challenging because it must be distributed due to the lack of a central clock.

- **Control information**. Token-passing protocols usually require more control information than TDMA protocols because they carry out explicit control by using the token packet. TDMA protocols employ control slots for slot reservation and synchronization, as well as guard time within the slots. As behind every problem lies an opportunity, some token-passing schemes takes advantage of the token transfers to measure the link quality and implement global priority, e.g., RT-WMP.

- **Performance**. A typical advantage of TDMA protocols is the possibility to reuse the channel in space, increasing the total capacity. Throughput depends strongly on several factors such as the employed protocols, network topology, supported data flows, control information and link quality.

Real-time Guarantees: FRT vs SRT Traffic Support

Protocols that provide SRT traffic capabilities usually implement a real-time packet scheduler, carry out a schedulability analysis in an error-free scenario and assume that there will be a small degradation in a real environment due to packet losses (IsoMAC). This strategy is also carried out in some protocols that support FRT traffic (RT-WMP and the Implicit-EDF protocol). However, when supporting FRT traffic, quantifying the occasional deadline misses is critical for evaluating the QoS provided by the network [Liu 06]. For this reason, it is desirable that schedulability analysis for FRT traffic carries out a fault-tolerant design in which some QoS metric must be satisfied, for example a DDR value close to 100%, and where packet losses are considered. A typical approach is to calculate the required time that must be scheduled for fault recovery according to a fault model (ISRA). In this way, probabilistic scheduling guarantees can be provided, as mentioned in Section 2.1.2.

WICKPro Features

Finally, we present WICKPro features as well as the reasons behind these choices, extending the comments in Section 1.3.2. WICKPro implements a **token-passing scheme** to avoid using clock synchronization. This scheme also prevents inter-flow and intra-flow interferences as well as the hidden and exposed terminal problems. As only one node can transmit at a given moment, it is feasible to find a set of links to transmit without interference and link schedule is not therefore explicitly considered. Thus, token-passing protocols are a sound solution for small-scale networks where spatial reuse is impossible or limited. Nevertheless, in Chapter 4, we also compare WICKPro with TDMA protocols that apply spatial reuse in a WMN with chain topology, namely Ripple, RMP and TDS.

WICKPro incorporates a **cyclic packet scheduler** where all nodes maintain global state information to enable a decentralized implementation. As stated in Section 2.1.1, we are only interested in scheduling periodic traffic, thus cyclic packet schedulers represent a good solution. To evaluate the required control information in cyclic and priority-based schedulers, in Chapter 5 we carry out a comparison between WICKPro and a token-passing protocol that implements a priority-based packet scheduler, i.e., RT-WMP. Likewise, Table 3.2 compares WICKPro with some of the discussed protocols depending on their packet scheduler and MAC protocol.

Table 3.2: Relation between packet scheduler and MAC scheme in WICKPro and some of the presented protocols

Packet scheduler	MAC scheme	Protocol
Priority-based	Token-passing	RT-WMP
Cyclic	Token-passing	WICKPro
Priority-based	TDMA	Implicit-EDF protocol
Cyclic	TDMA	IsoMAC

Given that traffic is periodic, there will be a global period (so-called the hyper-period or major cycle) in the overall traffic pattern, thus WICKPro tries to find a scheduling for this major cycle. As common in cyclic scheduling, the major cycle is divided into various minor cycles of the same duration. Scheduling calculation is based on the well-known **cyclic executive** [Baker 89] used in monoprocessor real-time systems. This algorithm provides each node with a transmission time that depends on its supported traffic. Conversely, token-passing protocols that implement cyclic packet schedulers such as WTRP allocate the same maximum transmission time to every node. This upper bound makes this mechanism less flexible than the cyclic executive. A similar idea was exploited previously by the Timed-token protocol [Malcolm 94] in wired networks, where a maximum TRT is also defined but a different maximum transmission time is assigned to each node. The main drawback of the Timed-token protocol is that the allocated transmission time is usually lower than the transmission time of a single data packet [Zhang 04] and therefore packet fragmentation will happen. Packet fragmentation is employed to meet data packet deadlines, however, the overhead introduced by packet fragmentation is somewhat higher in IEEE 802.11 networks than in wired networks. The heart of the matter is that a node has the same time reserved to transmit in all the token rotations, both in WTRP and in the Timed-token protocol. To increase the schedulability of a set of data flows, the cyclic executive does not impose this constraint. Nevertheless, the complexity of the scheduling calculation is increased. For this reason, as usual in cyclic executives, we analyze previously the traffic features of the supportable flows and simplify the scheduling calculation, for example, by making harmonic the periods of the flows.

WICKPro comes in two flavors to support FRT or SRT traffic. A comparison can be seen in Table 3.3. On the one hand, **FRT-WICKPro** reserves time for retransmissions in the cyclic scheduling to satisfy a target DDR. This reservation is calculated based on the set of supported data flows, the mean PDR between one-hop neighbors and a memoryless packet loss model. It employs coarse-grained synchronization based on packet receptions to trigger the token synchronously and fulfill strictly minor cycles, as well as synchronize the supported flows. On the other hand, **SRT-WICKPro** neglects synchronization and lets minor cycles be overrun, thereby decoupling the theoretic and the actual minor cycles. It is totally asynchronous and time for retransmissions is not explicitly considered. As it is simpler, we choose SRT-WICKPro to manage mobility together with the DoTHa hand-off algorithm by using RSSI measurements.

Table 3.3: Main features of FRT-WICKPro and SRT-WICKPro

Feature	**FRT-WICKPro**	**SRT-WICKPro**
Supported Traffic	FRT	SRT
Synchronization	Coarse-grained	No
Minor cycle fulfillment	Yes	No
Reservation for retransmissions	Yes	No
PDR computation	Yes	No
Mobility support	No	Yes

4 The Wireless Chain Network Protocol

In this chapter, we show the basic working of the WICKPro protocol in an error-free WMN that supports periodic traffic. This said, error control strategies are implemented to manage the reduced packet losses that are encountered in laboratory experiments and that cause, for example, token packet duplication. As we consider that there is no packet loss, the objective is to provide Hard Real-Time (HRT) traffic support thus all packets must be delivered from source to destination. This constraint is relaxed in Chapters 5 and 6 because error-prone networks are then taken into account. In these chapters, the goal will be to support FRT and SRT traffic, respectively.

To describe WICKPro, we use the TCP/IP 5-layer reference model [Tanenbaum 11] that defines the following layers: application, transport, network, link and physical. Particularly, WICKPro has functionalities that belong to the MAC and network layers. As usual in the IEEE 802 standards, the link layer is divided into the Logical Link Control (LLC) and MAC sublayers. However, we do not describe the LLC sublayer because it remains unchanged with respect to IEEE 802.11. It should also be noted that no transport protocol work together with WICKPro, only the application layer.

4.1 Relevant Publications

The work presented in this chapter was mainly published in the following papers:

- [Aisa 10] J. Aisa and Jose L. Villarroel. *WICKPro: A Hard Real-Time protocol for Wireless Mesh Networks with chain topologies.* In European Wireless (EW) Conference, pages 163-170, Lucca, Italy, Apr 2010.

- [Aisa 11] J. Aisa and Jose L. Villarroel. *The WICKPro protocol with the Packet Delivery Ratio metric.* Computer Communications, vol. 34, no. 17, pages 2047-2056, 2011.

4.2 Protocol States and Global State Information

WICKPro works in four states: inactive, topology discovery, PDR calculation and active. The process is shown in Algorithm 4.1. Nodes are initialized in the inactive state and subsequently they enter in the topology discovery state to search the network topology. Then, nodes compute the PDR between one-hop neighbors in the PDR calculation state whenever this option is enabled. In any case, either after the PDR computation or after the topology discovery if the PDR calculation is not enabled, nodes enter in the active state where two procedures can be invoked: an admission phase to accept new multi-hop flows, and a tear-down phase to remove currently supported multi-hop flows. If the network is broken, WICKPro comes back to the topology discovery state.

Algorithm 4.1 WICKPro protocol states

```
1: procedure WICKPro State
2:     $State \leftarrow Inactive$
3:
4:     while True do
5:         repeat
6:             $State \leftarrow TopologyDiscovery$
7:         until All Nodes Reached
8:
9:         while (PDR Calculation is Enabled) AND (PDR is Not Calculated) do
10:            $State \leftarrow PdrCalculation$
11:        end while
12:
13:        while Network is Connected do
14:            $State \leftarrow Active$
15:        end while
16:    end while
17: end procedure
```

The information that is held globally in all nodes is the following:

- Network topology. This is known before the network start-up and is maintained by means of the mobility management procedure as long as mobility is enabled, and this happens when SRT-WICKPro is used along with the DoTHa hand-off algorithm.

- Multi-hop flow information. On the one hand, as WICKPro assumes that there are four types of supportable data flows, the communication requirements of these flow types are stored in every node before the network start-up. Depending on the traffic criticality, a different subset of the real-time parameters given in Table 2.2 is saved. On the other hand, the requirements of the currently supported flows are also stored in every node. Actually, it is only stored the source, the destination, the type of the flow and the flow label (or flow number) that unequivocally identifies every data flow. This information is shared in the admission phase and removed in the tear-down phase.

- PDR. The mean PDR between one-hop neighbors is calculated before any communication can be established. As this is actually an optional feature, it is only enabled in FRT-WICKPro.

4.3 Control and Data Packets in WICKPro

WICKPro defines six control packets that can be classified in the following three subcategories: (i) The forward token-passing packets (regular token, establishment, removal and hand-off packets) pass the token according to the token path calculated by the scheduler, (ii) the backward token-passing packet (Negative ACKnowledgment (NACK) packet) carries out a token pass in the opposite direction of the token path, and (iii) the drop packet does not stand for a token pass. The regular token packet authorizes its holder to transmit, while the other forward and backward token packets have this feature but add extra functionalities. As explained below, the establishment packet is used to admit a new data flow, while the removal packet is employed to tear down a currently supported data flow. The hand-off packet executes a hand-off process, as mentioned in Chapter 7, the NACK packet requests data packet retransmissions and the drop packet acts as an ACK packet under certain circumstances (in this chapter and in FRT-WICKPro). It should be highlighted that we employ the terms token packet and forward token packet interchangeably. This definition may be confusing because the NACK packet also stands for a token pass, however, the forward token packets strictly follow the token path and they are therefore as a regular token packet in this sense.

Four different data packets are defined by WICKPro, namely Data1, Data2, Data3 and Data4. Every data packet represents a different type of data flow. Moreover, data and forward token packets are piggybacked when possible. More details about the packet definition and implementation can be found in Appendix A.

4.4 Connection Establishment and Termination

In the active state, new data flows can be accepted in the admission phase and current data flows can be removed in the tear-down phase. On the one hand, the admission phase is started by the application layer and has the following five steps:

1. The application layer defines the communication requirements.

2. The network layer receives these real-time requirements and calculates the route between the source and destination nodes.

3. The MAC sublayer carries out the local admission control by using the off-line packet scheduler, which searches a scheduling with the new flow and the current flows in a way that all of them satisfy their real-time requirements. If it is found, the new flow is accepted, otherwise it is rejected.

4. The global admission control that leads to the resource reservation is then accomplished as follows. Let us consider the network depicted in Fig. 4.1 where there are five routers (R1-R5) and two clients (C6-C7), so-called Scenario 4.1. As will be explained later in detail, every minor cycle is implemented as a token rotation that starts and ends in the token master (R1 in this case). If C7 wants to establish a new flow, this waits to receive the regular token packet and changes it for a flow establishment packet, as we can see in the minor cycle n of Fig. 4.2, where the resource reservation process is shown. It should be noted that in Fig. 4.2 the packet destination is indicated by an arrow that

starts in the packet itself. In the minor cycles n and $n+1$, all nodes must receive the flow establishment packet, calculate the new scheduling and accept the new flow (if a node accepts a new flow, the rest will also accept it because they share the same information). For this task, the flow establishment packet carries the flow label and the real-time requirements of the new flow: source, destination and type of the flow.

Sending the flow establishment packet in a whole minor cycle, $n+1$ in this case, is mandatory to make sure that all nodes receive the flow establishment packet. After this minor cycle, all nodes have accepted the new flow and have calculated the new scheduling, thus the resources are actually reserved.

5. The scheduling change is indicated by the token master in the minor cycle $n+2$ through the *NewScheduling* flag in the field *Notification* of the regular token packet.

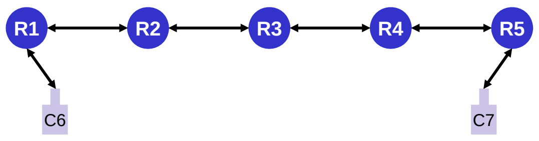


Figure 4.1: Scenario 4.1. WMN with five routers (R1-R5) and two clients (C6-C7)

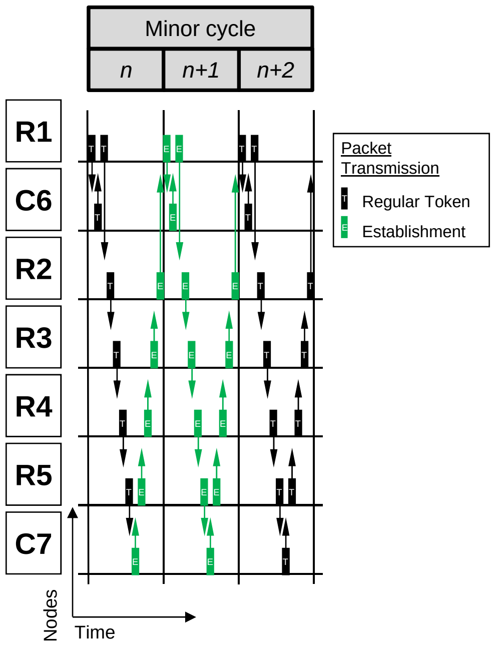


Figure 4.2: Resource reservation of WICKPro in three minor cycles during the admission phase in Scenario 4.1

On the other hand, the tear-down phase is similar to the admission phase but, instead of a flow establishment packet, a flow removal packet is used to release the reserved resources. The scheduling is also recalculated because this can produce some advantages such as energy saving.

Hence, in an error-free network, the maximum time to establish a new data flow corresponds to the duration of three minor cycles. However, in case of simultaneous requests, either flow establishment or removal announcement, only one of them can be managed at a time, because the same information must be held in all nodes. This serialization of requests is trivially handled by the token rotation and each request takes three minor cycles.

4.5 Network Layer

Two mechanisms are implemented in the network layer: the topology management procedure and the routing algorithm.

4.5.1 Topology Management

In the topology discovery state, WICKPro checks the connectivity based on the network topology that is known before the network start-up. Once in the active state, mobility is managed through a hand-off algorithm as shown in Chapter 7, whenever mobility is enabled. Likewise, in the active state, the network topology is recovered if the network is broken. The node that holds the token considers that the network is broken if the number of retransmissions is higher than a predefined value, whereas the other nodes wait until the time without receiving the token exceeds a maximum time.

4.5.2 Routing Algorithm

The routing algorithm is simple because, as described in Section 1.3.1, routes are unique. Furthermore, since all nodes know the network topology, they can calculate properly the route between any pair of nodes.

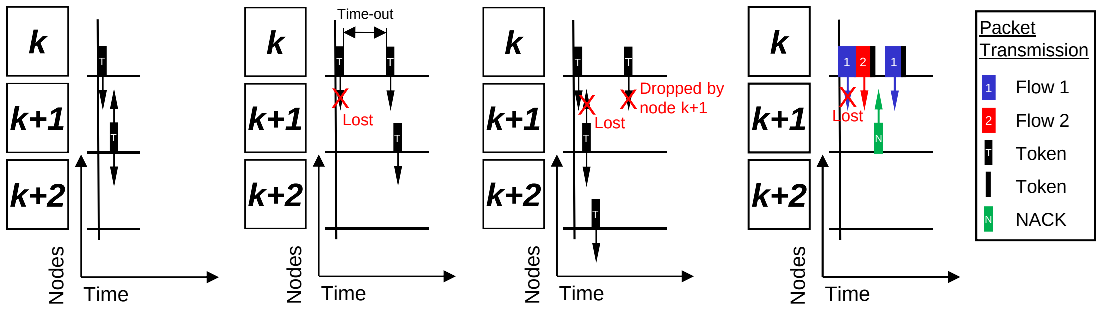
4.6 MAC Sublayer

4.6.1 Medium Access

WICKPro employes a token-passing access method to provide real-time guarantees. Since there is a single shared radio channel and spatial reuse is not applied, there is only one token in the network which all nodes pass on to one another. The node that holds the token is the only one that can transmit, so the hidden and exposed terminal problems, and inter-flow and intra-flow interferences are avoided. As mentioned earlier, this assumption is reasonable in small-scale networks because spatial reuse is limited. It should also be noted that this token-passing scheme is implemented on top of the MAC protocol of IEEE 802.11, i.e., CSMA/CA, which has to be taken into account, for example, when calculating the transmission time of every packet.

4.6.2 Error Control

Error control is required because packet losses will cause the loss and duplication of token and data packets. As its name suggests, token loss and duplication involve regular token, establishment, removal and hand-off packets, as well as piggyback packets since these also carry a forward token packet. Likewise, it applies to NACK packets because they also carry out a token pass, although NACK packets are not always mentioned in the explanations below. Besides the Forward Error Correction (FEC) codes provided by IEEE 802.11, an Automatic Repeat reQuest (ARQ) method based on software retransmissions is incorporated. This method substitutes to the hardware-level ACKs of IEEE 802.11, which is disabled due to the use of broadcast transmissions. Let us illustrate the solutions provided by WICKPro with the examples given in Fig. 4.3.



(a) Implicit ACK **(b)** Token loss **(c)** Token duplication **(d)** Data loss

Figure 4.3: Error control mechanisms of WICKPro

4.6.2.1 Token or Piggyback Packet Loss

WICKPro employs implicit ACK to handle token packet loss, so the node that transmits the token packet confirms receptions by forwarding transmissions, unlike explicit ACK where specific ACK packets are sent. This process is shown in Fig. 4.3a, where node k sends the token packet to node $k+1$ and this, in turn, sends the token to node $k+2$. As node k also receives the token packet sent by node $k+1$, node k acknowledges successfully its token packet transmission. It should be noted that implementing implicit ACK is feasible due to the use of omnidirectional antennas.

Now let us see how a token packet loss is managed. Fig. 4.3b illustrates this process, where node k sends a token packet to node $k+1$. If node k does not listen to any transmission from node $k+1$ in a time-out, node k presumes that the token packet did not arrive correctly and sends it again. Therefore, this is called timeout-based retransmission.

4.6.2.2 Token or Piggyback Packet Duplication

The token packet may be duplicated as shown in Fig. 4.3c. Node $k+1$ correctly receives a token packet sent by node k and sends another token packet to node $k+2$, but node k does not monitor it correctly. In this case, node k retransmits the token packet thereby producing a token duplication. To handle this situation, a field called *Serial* is introduced in the token packet (and also in the NACK packet). This field is increased every time a node transmits the token packet but it is not increased after a retransmission. Therefore, node $k+1$ ignores the retransmitted token packet sent by node k because the *Serial* of that packet is the same it received the last time. To mitigate the contention that happens if an implicit ACK is lost, we have introduced the possibility of making implicit ACK with any subsequent node in the token rotation.

4.6.2.3 Data Packet Loss

When a data packet is sent along with a forward token packet as a piggyback packet, timeout-based retransmission is employed. However, since nodes can transmit more than one data packet in a token holding, WICKPro contemplates another retransmission mechanism through a NACK packet. For this purpose, a field called *NumDataPacketsSent* is introduced in the forward token packets (but not in the NACK packet). In Fig. 4.3d, we can observe that node k sends two packets: a data packet of flow number 1, and a piggyback packet that contains a data packet of flow 2 and the forward token packet. As the data packet of flow 1 is lost, node *k+1* only receives one data packet but the field *NumDataPacketsSent* states that it should have received two data packets. Therefore, node *k+1* sends a NACK packet to node k indicating the correctly received packet and then node k retransmits the data packet of flow 1 that node *k+1* did not receive correctly. We must recall that more details about the control packet implementation can be found in Appendix A.1.

4.6.2.4 Data Packet Duplication

A data packet duplication can occur if a piggyback packet is duplicated, but since this also contains a forward token packet, it will be discarded due to its *Serial*. Moreover, data packets are numbered and thereby allowing nodes to recognize a data packet duplication.

4.7 MAC Sublayer - Off-line Packet Scheduler

WICKPro implements an off-line cyclic packet scheduler. It is off-line because flows are scheduled before they actually start transmission. And it is cyclic because the packet schedule is calculated for a hyper-period (or major cycle) where the overall traffic pattern is repeated, since the communication requirements are periodic. As common in cyclic scheduling, the major cycle (M) is divided into various minor cycles (m or MiC) of the same duration in a way that the major cycle is an integer multiple of the number of minor cycles. In every minor cycle, nodes can be allocated a different transmission time depending on its supported traffic.

4.7.1 Implementation within a Token-passing Scheme

Every minor cycle is implemented as a token rotation that starts and ends in the token master. This token-passing mechanism is combined with a polling approach based on a global cyclic packet schedule. In this way, the node that receives the token packet can transmit data packets if there are data packets ready to be sent and the packet scheduler authorizes these transmissions. In any event, whether a node has transmitted data packets or not, it sends the token packet to the next node in the token path. Moreover, in every minor cycle, all nodes must receive the token packet at least once, which increases the responsiveness in case of topology or communication requirements changes.

The token path may change from one minor cycle to another depending on the data packets to transmit. For this reason, we define some policies to minimize the token packet transmissions while all clients receive the token at least once every minor cycle[1]. For this reason, in every minor cycle the cyclic scheduler differentiates between clients that have one Token Holding Opportunity (THO) and those that have two THOs. If clients have one THO (like C7 in Fig. 4.1 given that R1 is the token master), they always receive the token once. If clients have two THOs (like C6 in Fig. 4.1), the scheduler considers five situations depending on if the client has to transmit data packets, receive or both, as can be seen in Table 4.1.

Table 4.1: Cases covered by the cyclic scheduler with clients that have two token holding opportunities per minor cycle

Case	Tx	Rx	Tokens received per client
1	No	No	Once, the first possible time (arbitrary choice)
2	No	Yes	Once, the first or the second possible time
3	Yes	No	Once, the first possible time
4	Yes	Yes (1st THO)	Once, the first possible time
5	Yes	Yes (2nd THO)	Once or twice

Let us illustrate the cases 1, 3 and 5 of Table 4.1 with two examples that consider the network topology depicted in Fig. 4.1. In the first example, there is no data flow in the network. Fig. 4.4 shows the cyclic scheduling and illustrates the case 1 where C6 only receives the token once. Particularly, we implemented the option depicted in Fig. 4.4a in which C6 receives the token packet in the first THO, but this choice is arbitrary. The other option where C6 receives the token in the second THO is shown in Fig. 4.4b.

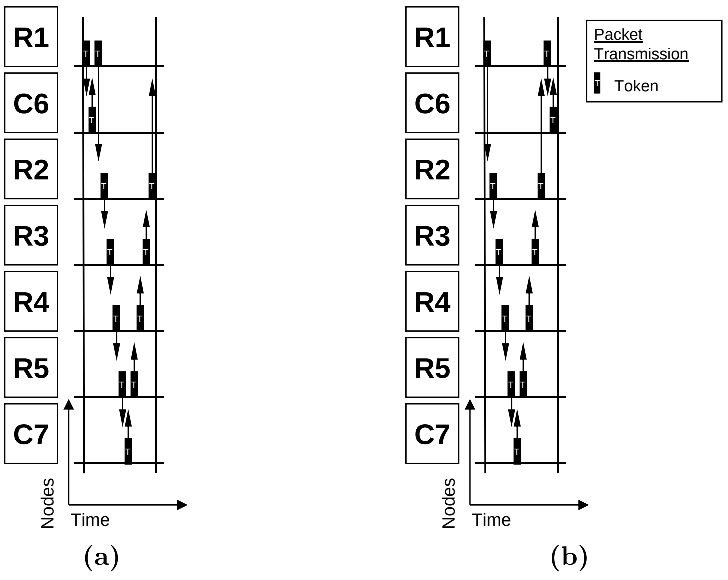


Figure 4.4: Cyclic scheduling without data flows in Scenario 4.1. (a) is the option implemented by WICKPro where C6 receives the token in the first THO, and (b) is another possible option in which C6 receives the token in the second THO

[1]This problem is similar to the traveling salesman problem, but there are some differences: sometimes WICKPro requires that a node is visited twice in a token rotation and the linear topology formed by routers may be used to simplify the problem.

In the second example, there are two multi-hop flows in the network whose features are given by Table 4.2 (we must recall that implicit deadline is assumed). The cyclic scheduling consists of two minor cycles, as observed in Fig. 4.5, where, in addition, the generation of every data packet is indicated by an arrow. In the second minor cycle, C6 must transmit one data packet but does not have to receive any packet, thus it receives the token in the first THO, as dictated by the case 3. Otherwise, if C6 received the token packet in the second THO, the complete multi-hop flow would not be transmitted in the current minor cycle. Finally, the case 5 is illustrated in the first minor cycle. C6 receives a data packet from C7 in the second THO but it must transmit in the first THO in order to let C7 receive its data packet in the current minor cycle; therefore, C6 must receive the token twice. It should be noted that in the case 5 it is also possible that a node can transmit and receive properly in the same minor cycle with a single token holding, depending on the scenario.

Table 4.2: Example of flow features in Scenario 4.1

Flow number	C (ms)	P (ms)	src	dst
1	2.09	26.58	C6	C7
2	2.09	53.16	C7	C6

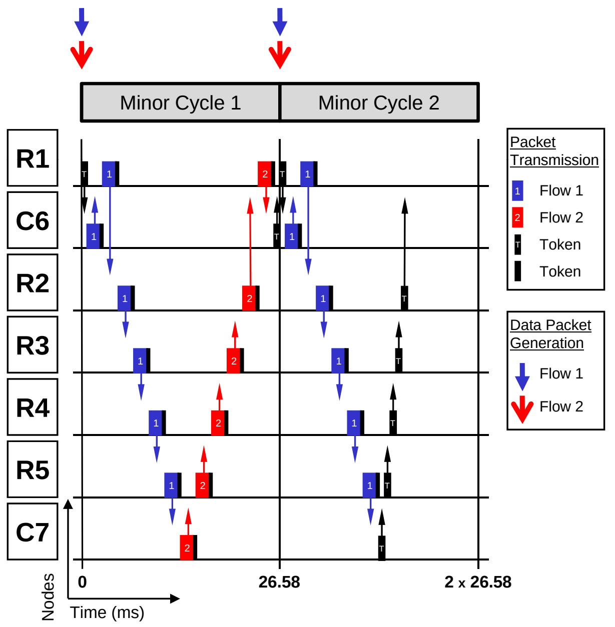


Figure 4.5: Example of cyclic scheduling in Scenario 4.1 while supporting the two data flows given by Table 4.2

4.7.2 Cyclic Scheduling Calculation

Scheduling calculation is based on the well-known **cyclic executive** [Baker 89] used in monoprocessor real-time systems. The algorithm is decentralized thanks to the global state information maintained by all nodes. As packet losses are not considered, the scheduler aims at providing HRT traffic support.

Network utilization The existence of a schedule requires that the network utilization is less than or equal to 1. This computation is not, however, obvious because it depends on the token path. For this reason, we calculate the worst-case network utilization in a minor cycle: all data packets are sent and all clients receive the token twice thus four tokens must be transmitted per client (this calculation could be refined in future by considering, for example, piggybacking). The network utilization U is expressed as follows:

$$U = U_{Data} + U_{Token} = \left[\sum_{i=1}^{N_{MH}} HC_i * \frac{C_i}{P_i} \right] + \left[\frac{(2*(N_R - 1) + 4 * N_C) * C_{Token}}{m} \right] \quad (4.1)$$

Although the parameters of Eq. (4.1) have already been defined previously, we show them again all together for the sake of clarity. N_{MH} is the number of multi-hop flows, HC_i is the hop count of flow i, C_i is the transmission time of a packet of flow i, P_i is the packet generation period of flow i, N_R is the number of routers, N_C is the number of clients, C_{Token} is the token transmission time and m is the minor cycle duration. As the transmission time of all the token packets is similar, we consider that C_{Token} corresponds to the transmission time of a regular token packet. It should be noted that we assume that the transmission time is the same in all the senders within a multi-hop flow, otherwise C_i and C_{Token} would be different in every transmitter node and Eq. (4.1) should be reformulated.

Multi-hop flow partition Every multi-hop flow MH_k is divided into HC_k one-hop flows $OH_{kj} = \{OH_{k1}, OH_{k2} ... OH_{kHC_k}\}$ with precedence relationships. The objective is therefore to schedule the N_{OH} one-hop flows that can be expressed as $OH_i = \{OH_1, OH_2, ..., OH_{N_{OH}}\}$, where $N_{OH} = \sum_{k=1}^{N_{MH}} HC_k$, as explained in Section 2.1.1.

Major and minor cycles Based on the communication requirements of the N_{OH} one-hop flows, the major cycle and the possible values of the minor cycle are computed. The major cycle is the least common multiple (lcm) of the periods of the supported one-hop flows:

$$M = lcm(P_i), i = 1...N_{OH} \quad (4.2)$$

As with the cyclic executive, the minor cycle duration must satisfy the following four requirements (gcd stands for greatest common divisor, TRT is the Token Rotation Time and $i = 1...N_{OH}$):

$$\begin{array}{ll} 1. & m \leq \min(D_i) \\ 2. & m \geq \max(C_i) + \mathrm{TRT} - C_{Token} \\ 3. & \exists k : M = km \\ 4. & \forall i : m + (m - \gcd(m, P_i)) \leq D_i \end{array} \quad (4.3)$$

The requirement 1 says that the minor cycle duration must be lower than or equal to the shortest deadline of the supported flows. The requirement 2 states that m must be higher than or equal to the longest transmission time plus the TRT minus the token transmission time, because this is the time required to carry out a token rotation and transmit the longest data packet by using piggybacking. Based on the requirement 3, the major cycle must be an integer multiple of the number of minor cycles. And the requirement 4, which includes the requirement 1, says that between every data packet generation and the corresponding deadline there must be a complete minor cycle, assuming that the data flows are started in phase.

After applying Eq. (4.1) and (4.3), we obtain several possible values than the minor cycles can have. If no value can satisfy these conditions, the set of data flows is not schedulable and the new flow is not therefore admitted.

Scheduling search The scheduling is searched by using an heuristic algorithm that works in the following steps:

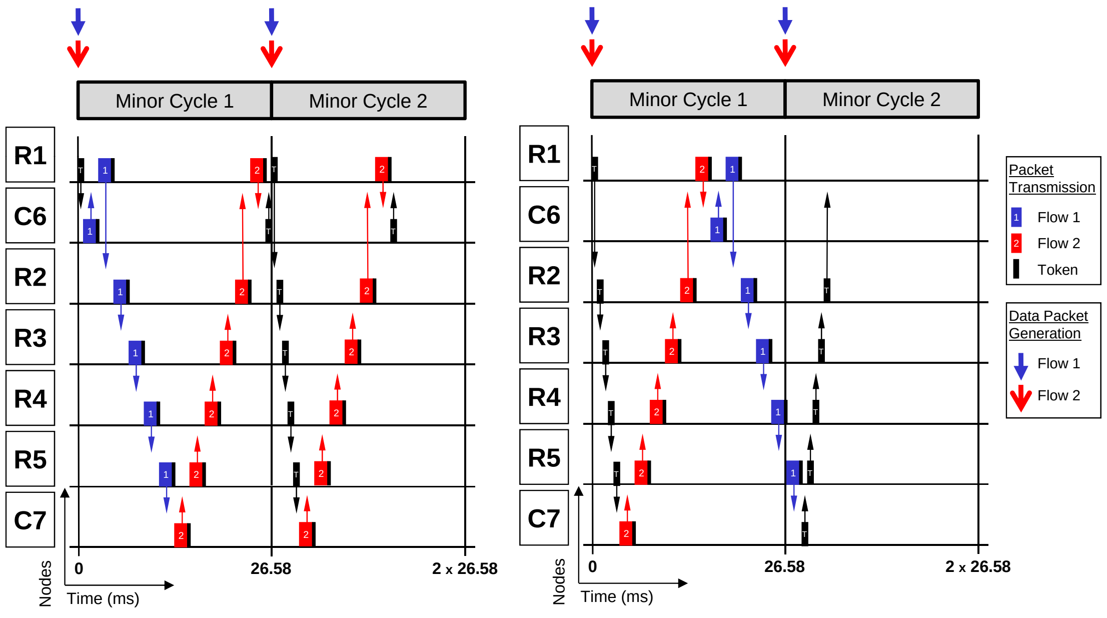
1. The highest possible m is selected for the heuristic search. The reason is that the network utilization is lower (the overhead produced by the token rotation is lower) and the complexity of the problem decreases when the number of minor cycles decreases. The number of minor cycles is $n_m = M/m$.

2. The number of transmissions (or executions) of every one-hop flow in the major cycle is $n_{e_i} = M/P_i$. These are the transmissions that are actually scheduled. The total number of transmissions to schedule is therefore $n_{total_e} = \sum_{i=1}^{N_{OH}} n_{e_i}$.

3. The possible minor cycles where every transmission can be scheduled is computed based on the next policy. A data packet can be sent in a minor cycle if it has arrived at the transmission buffer of the source node in the previous minor cycle and its deadline finishes after the minor cycle finishes, as in the cyclic executive. This also applies to all the packets of a multi-hop flow and consequently, for the scheduling algorithm, a data packet that arrives at the transmission buffer of a source node is also considered to arrive at the transmission buffer of the relay nodes involved in the end-to-end communication.

4. Every transmission is then scheduled one by one based on a Rate Monotonic Scheduling (RMS) algorithm that uses depth-first search. In this way, the lower the generation period, the higher its priority. One transmission is assigned to the first of its possible minor cycles whenever there is enough free time in that minor cycle for the transmission in question, otherwise the scheduler tries to allocate the transmission in the next feasible minor cycle. If it is not possible, the scheduler backtracks. If a scheduling is found, the algorithm will finish. However, if it is not possible to find a scheduling for the current m, this minor cycle duration is discarded and the algorithm comes back to the step (1), where the highest remainder m will be selected. As expected, if the packet scheduling is finally found for any of the possible minor cycle durations, the new flow is admitted. If not, it is rejected.

It should be noted that transmissions are scheduled based on rate-monotonic priorities, however, the scheduler calculates token rotations without taking into account these priorities, which lets reducing the number of token packets. This is illustrated in Fig. 4.6, which shows the same example than in Fig. 4.5 but where the periods of the flows are inverted: the period of flow 1 is 53.16 ms and the period of flow 2 is 26.58 ms. In this example, although flow 2 is scheduled before flow 1 because its period is lower, the scheduler organizes the transmissions considering that the token rotation starts in the master node (R1), as shown in Fig. 4.6a. Otherwise, the number of token packets is increased and the minor cycle is higher, thereby causing that the two flows are not actually schedulable. Fig. 4.6b depicts the first minor cycle of this latter implementation, where the number of THO is also higher. Having said all this, nodes still order their transmissions based on rate-monotonic priorities when they have to send more than one data packet.



(a) Option chosen (b) Option not selected

Figure 4.6: Example of cyclic scheduling in Scenario 4.1 where the period of flow 1 is 53.16 ms and the period of flow 2 is 26.58 ms (just the opposite than in Fig. 4.5). (a) is the option implemented by WICKPro, where the token rotation is calculated based on minimizing the number of token packets, while (b) is another possible option in which the token rotation is calculated based on rate-monotonic priorities

Minor cycle: to overrun or not to overrun? On the one hand, if all the transmissions within a minor cycle are completed before the minor cycle duration expires, the token master retains the token until the beginning of the next minor cycle. This is the option implemented in this chapter and in FRT-WICKPro, but not in SRT-WICKPro, where the next minor cycle is immediately triggered. For this reason, in this chapter and in FRT-WICKPro, if the token master is going to keep the token packet during a time higher than the stipulated time-out, the token master sends a drop packet to the node that is waiting for the implicit ACK to prevent this node from retransmitting the token packet to the master node. On the other hand, if transmissions are not completed before a minor cycle finishes, in this chapter and in SRT-WICKPro they are left in execution until completion. In the laboratory experiments of this chapter, this is not a problem because the network is almost error-free, whereas SRT-WICKPro carries out some strategies to avoid the possible domino effect. Conversely, FRT-WICKPro fulfills strictly minor cycles by using an on-line packet scheduler and packet synchronization. Finally, it should be noted that the theoretical minor cycle duration (m) is equivalent to the maximum TRT in other token-passing protocols.

Calculation complexity Our cyclic packet scheduler inherits some problems from the cyclic executive. Therefore, as stated in Section 2.1.1, we only define four data packet types in WICKPro. If we considered infinite types of data packets, the convergence of our cyclic scheduling would be much more complicated. For this reason, as usual in cyclic executives, we analyze previously the traffic features of the supportable flows and simplify the scheduling calculation by using the typical solutions applied in the cyclic executive. For example, if the major cycle is very large, the periods of the flows can be changed to be harmonic in order to reduce the major cycle duration and consequently the complexity of the scheduling.

The space of solutions is formed by all the ways of including the n_{total_e} transmissions in the major cycle, enclosed by the following equation as a function of the number of minor cycles (n_m) and the number of transmissions of every one-hop flow in the major cycle (n_{e_i}) [Yepez 06]:

$$\prod_{i=1}^{n_{total_e}} \binom{n_m}{n_{e_i}} = \frac{(n_m!)^{n_{total_e}}}{\prod_{i=1}^{n_{total_e}} n_{e_i} * (n_m - n_{e_i})!} \tag{4.4}$$

From this formula, we can infer that the more the number of transmissions and minor cycles, the more the complexity of the problem.

4.8 Evaluation

In this section, we assess WICKPro in an almost error-free scenario. In Test 4.1 (Section 4.8.1) and Test 4.2 (Section 4.8.2), we compare WICKPro with the IEEE 802.11 protocol and Ripple, TDS and RMP, the three TDMA protocols presented in Section 3.1 that apply spatial reuse in WMNs with chain topologies. Particularly, Test 4.1 and 4.2 evaluate the throughput with uniform and non-uniform traffic, respectively. Furthermore, Test 4.3 (Section 4.8.3) shows WICKPro performance in terms of throughput and delay when supporting real-time traffic. It should be noted that we did not implement the tests for the IEEE 802.11 protocol, Ripple, TDS and RMP, but we employed published results in the corresponding papers.

Experiment and comparison description To evaluate WICKPro, we carried out laboratory experiments where we employed five PCs equipped with D-Link Air Premier AG DWL-AG530 IEEE 802.11 cards that are based on Atheros chipsets. These cards used CSMA/CA without RTS/CTS and supported broadcast transmissions thus without hardware-level ACKs of IEEE 802.11. The operating system of the PCs was the Debian Linux with kernel version 2.6.26. WICKPro was implemented in C and executed as a user space process whose priority was set to real-time in order to improve WICKPro performance. Moreover, wireless cards were configured in ad-hoc mode by using a modified ath5k Linux driver developed in our research group.

In the three tests, WICKPro was tested in an almost error-free scenario and in a WMN without clients, thus letting a fair comparison with Ripple, TDS and RMP. However, a few errors occurred so that the error control mechanisms provided by

WICKPro could be tested. In Test 4.1 and 4.2, WICKPro was implemented over the IEEE 802.11b standard at 1 Mbps to compare its results with those of IEEE 802.11, Ripple and RMP. The results of TDS are based on simulations at 2 Mbps. This means that WICKPro is not compared directly with TDS, but we consider that its results are similar to those of Ripple because their approaches are similar. For a better understanding, we completed the results of the Tests 4.1 and 4.2 for six, seven and eight nodes with theoretical calculations. In Test 4.3, the IEEE 802.11a standard at 6 Mbps was employed. In addition, the data flows were established at the beginning of the three tests so that the admission phase of WICKPro was not used. The time-out was set to 10 ms because lower values did not work properly due to the used operating system.

Implementation on top of IEEE 802.11 The transmission time C of every packet must be computed before calculating the scheduling. Thus, since WICKPro is implemented over IEEE 802.11, it is important to study the IEEE 802.11 protocol. In [Jun 03] an analytical study was conducted to obtain the theoretical maximum throughput in error-free IEEE 802.11 networks that use the DCF operation mode. This analysis was classified based on different MAC schemes (CSMA/CA with or without RTS/CTS), standards (a, b or g), bit rates and packet sizes. We consider these values to be the upper bound of obtainable throughput. Particularly, when using CSMA/CA without RTS/CTS, without collisions and without hardware ACKs, the transmission time consists of three components [Jun 03]:

$$C_{802.11} = C_{DIFS} + C_{Back-off} + C_{Data} \tag{4.5}$$

C_{DIFS} stands for the DIFS duration and is the time that the channel must be sensed continuously idle before transmitting. If the node has not used the channel recently and the channel was idle since the node started listening, after this time the node can transmit, otherwise the node has to wait for the back-off counter to reach zero, which is precisely the time $C_{Back-off}$. In order to obtain the theoretical maximum throughput, in [Jun 03] collisions were neglected and $C_{Back-off}$ was set to be the expected back-off time. As there was no collision, the back-off time was uniformly distributed from 0 to the minimum contention time and the expected value was therefore the minimum contention time divided by two. In our case, as WICKPro implements a token-passing scheme, collisions are also avoided, except in a small percentage of times due to token packet duplication, and thus the formulas provided by [Jun 03] can be properly applied to WICKPro. Finally, C_{Data} is the transmission time itself, which depends on the bit rate and the packet size.

In the case of using IEEE 802.11b, the transmission time formula is as follows, where $C_{DIFS} = 50\ \mu s$ and $C_{Back-off} = 310\ \mu s$:

$$C_{802.11b}(\mu s) = 50 + 310 + \left(192 + \frac{8 * (28 + MSDU(bytes))}{R_b(Mbps)}\right) \tag{4.6}$$

As we can observe from Eq. (4.6), C_{Data} has three contributions: (i) physical layer (192 μs); (ii) the header of the link layer (28 bytes); and (iii) the MAC Service Data Unit (MSDU), i.e, the data itself, which includes WICKPro overhead and data since TCP, UDP and IP are not used.

When using IEEE 802.11a at 6 Mbps, the transmission time formula is as follows, being $C_{DIFS} = 34\ \mu s$ and $C_{Back-off} = 67.5\ \mu s$:

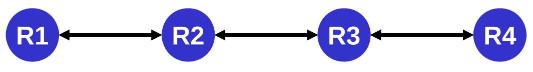
$$C_{802.11a}^{6Mbps}(\mu s) = 34 + 67.5 + \left(20 + 4 * \left\lceil \frac{16 + 6 + 8 * [28 + MSDU(bytes)]}{24} \right\rceil\right) \quad (4.7)$$

Moreover, WICKPro adds a transmission margin which is necessary because of the operating system overhead and the code execution time at each node. WICKPro used a margin of 0.7 ms because our tests indicated that this was a good margin. Therefore, the transmission time C is:

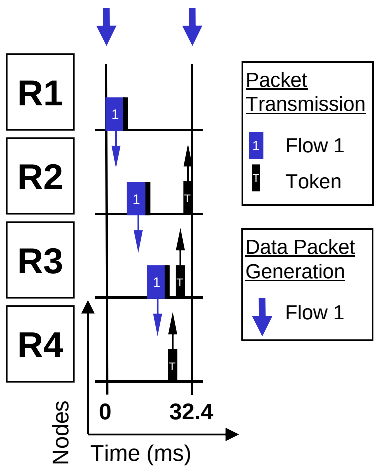
$$C(ms) = C_{802.11}(ms) + C_{process}(ms) = C_{802.11}(ms) + 0.7 \quad (4.8)$$

4.8.1 Test 4.1: Throughput with Uniform Traffic

In this test, there was only one flow that saturated the network and went from the first to the last node in the chain in order to measure the maximum end-to-end throughput. The data packet size was 1,000 bytes and the test duration was 500 seconds for comparing our results with those of Ripple and RMP. According to Eq. (4.6)[2], the transmission time of piggyback packets (1,009 bytes) and regular token packets (7 bytes) was 8.74 and 0.72 ms, respectively[3]. Thus, using Eq. (4.8) and rounding out the obtained values, WICKPro considered the transmission time C of piggyback and (regular) token packets equals 9.4 and 1.4 ms, respectively. According to these times, WICKPro defined only one minor cycle which lasted $10.8 * (N_R - 1)$ ms. In the case of four routers, the network topology and the packet scheduling are shown in Fig. 4.7.



(a) Network topology



(b) Packet scheduling

Figure 4.7: Chain network with four routers in Test 4.1

[2]In this case, we considered that the link layer header had a length of 14 bytes, instead of 28 bytes, due to the use of raw sockets and a driver developed by the research group that encapsulates 802.11 frames in Ethernet. However, in our later work we assumed the standard 802.11 MAC header size of 28 bytes.

[3]Although based on Appendix A the sizes of piggyback and regular token packets are, respectively, 1,011 and 9 bytes, at this point WICKPro had not defined the *Notification* field and the *FreeTimeMiC* field had a size of one byte instead of two bytes, so their fields had two bytes less.

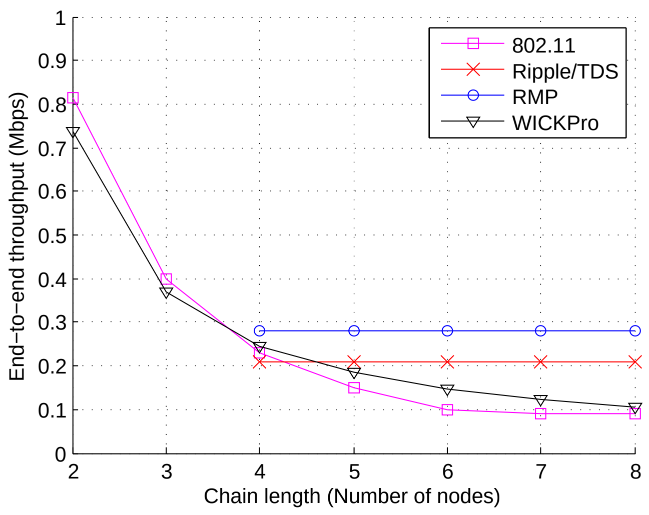


Figure 4.8: Throughput with uniform traffic in Test 4.1

Fig. 4.8 shows the end-to-end throughput attained by IEEE 802.11, Ripple, TDS, RMP and WICKPro:

- The end-to-end throughput of Ripple and TDS is 0.21 Mbps and that of RMP is 0.28 Mbps. As these experiments were simulated by using the 802.11b standard at 1 Mbps with RTS/CTS and with a packet size of 1,000 bytes, the theoretical maximum throughput is 0.815 Mbps according to [Jun 03]. Thus, the end-to-end throughput of Ripple and TDS should be 0.815 divided by 4, i.e., about 0.21 Mbps and that of RMP should be 0.815 divided by 3, i.e., about 0.28 Mbps. The simulation and theoretic values are therefore very similar.

- The IEEE 802.11 protocol obtains an end-to-end throughput of 0.815 Mbps when there are only two nodes in the chain because there is no packet collision. If the number of nodes increases, the end-to-end throughput decreases dramatically until about 0.1 Mbps. These results are extracted from the simulations realized in [Cheng 06]. In [Que 07] it is explained that in an IEEE 802.11 chain network with more than seven nodes, the end-to-end throughput tends to be about 1/7 of the theoretical maximum throughput, in this case 0.118 Mbps, which is quite similar to the simulated value.

- WICKPro obtains lower end-to-end throughput than Ripple, TDS and RMP when there are more than four nodes in the chain because the latter carry out spatial reuse. When there are four nodes, WICKPro obtains lower end-to-end throughput than RMP and higher than Ripple and TDS. Finally, WICKPro and the IEEE 802.11 protocol attains a similar end-to-end throughput.

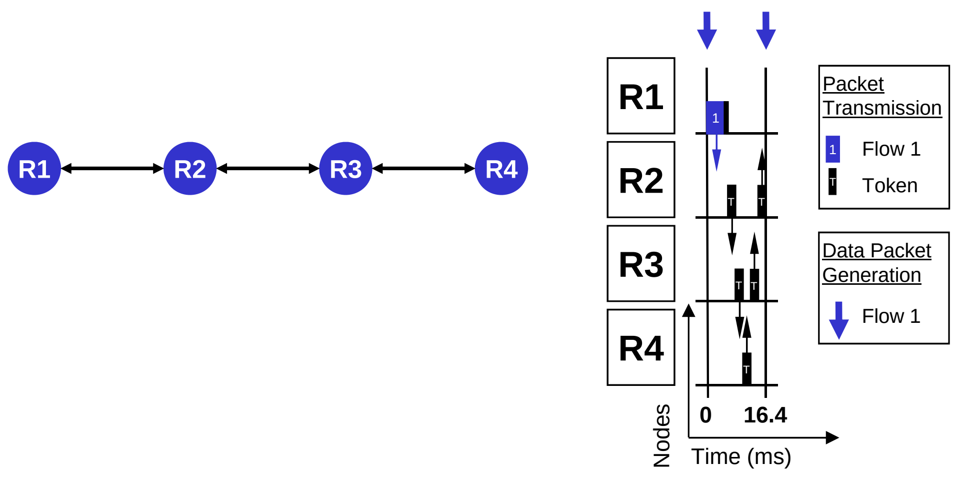
Table 4.3 shows the detailed results obtained by WICKPro in Test 4.1. The table includes the end-to-end throughput, the theoretical minor cycle duration, and the mean and the standard deviation measurements of the minor cycle for various chain lengths. The minor cycles where there were retransmissions, around the 0.03%, have been removed to study the nominal duration of the minor cycles. Surprisingly, the minor cycle standard deviation is very small and the minor cycle mean is similar to the minor cycle duration calculated without the temporal margin added to the transmission time of every packet (whose value was 0.7 ms). However, it is not a good idea to eliminate this temporal margin because it is sometimes useful to handle the indeterminism of the operating system and therefore it helps HRT traffic constraints to be met.

Table 4.3: Detailed results obtained by WICKPro in Test 4.1

Nodes	Throughput (Mbps)	MiC duration (ms)	MiC measure: mean ± std dev (ms)
2	0.737	10.8	9.4 ± 0.68
3	0.369	21.6	18.75 ± 0.14
4	0.246	32.4	28.08 ± 0.14
5	0.184	43.2	37.39 ± 0.15

4.8.2 Test 4.2: Throughput with Non-uniform Traffic

As mentioned in Section 3.1, Ripple, TDS and RMP use a fixed TDMA scheduling so that their scheduler always assigns each node the same throughput, independently of how much traffic it is supporting. In other words, these protocols do not adapt to the network load. For this reason, in this experiment we evaluated Ripple, TDS, RMP and WICKPro when supporting non-uniform traffic. In particular, we defined one single flow that went from the first to the second node in the chain and measured the throughput obtained by this flow as a function of the number of nodes in the chain. We considered the same transmission times of piggyback and token packets as in the first test, thus the minor cycle lasted $9.4 + 1.4 * (2 * (N_R - 1) - 1)$ ms. In the case of four routers, the network topology and the packet scheduling are shown in Fig. 4.9.



(a) Network topology **(b)** Packet scheduling

Figure 4.9: Chain network with four routers in Test 4.2

Fig. 4.10 shows the throughput attained by WICKPro, which decreases with the chain length. Ripple and TDS would obtain a throughput of 0.21 Mbps and RMP a throughput of 0.28 Mbps, as in the Test 4.1. The detailed results obtained by WICKPro are similar to those shown in Table 4.3, so they are not presented here. In this test, we wanted to highlight the flexibility of WICKPro with respect to Ripple, TDS and RMP.

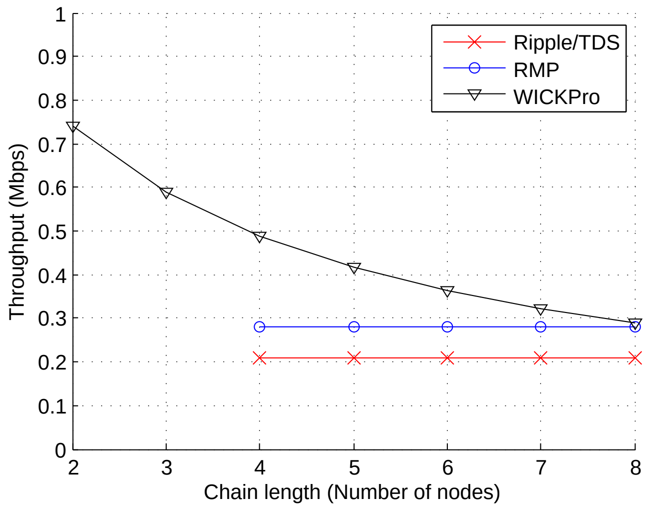


Figure 4.10: Throughput with a non-uniform traffic in Test 4.2

4.8.3 Test 4.3: Real-time Scheduling

In Test 4.3, we implemented a WMN with five routers that supported the five flows shown in Table 4.4. The transmission times are calculated according to Eq. (4.7) and (4.8). Note that the (regular) token transmission time is 0.85 ms due to the use of IEEE 802.11a at 6 Mbps.

Table 4.4: Communication requirements in Test 4.3

Flow number	Packet Size (bytes)	C (ms)	P (ms)	Source	Destination
1	1000	2.2	60	R4	R1
2	1000	2.2	60	R1	R5
3	500	1.5	90	R3	R1
4	500	1.5	90	R2	R5
5	100	1	120	R5	R1

Packet scheduling calculation WICKPro calculates the packet scheduling as described in Section 4.7.2. From Eq. (4.1), the network utilization is computed thus obtaining a constraint for the minor cycle duration:

$$\begin{aligned} &U = \left[\sum_{i=1}^{5} HC_i * \frac{C_i}{P_i}\right] + \left[\frac{(2*(5-1)+4*0)*0.85}{m}\right] \leq 1 \\ &U = 3*\frac{2.2}{60} + 4*\frac{2.2}{60} + 2*\frac{1.5}{90} + 3*\frac{1.5}{90} + 4*\frac{1}{120} + \frac{8*0.85}{m} \leq 1 \\ &m \geq 10.85\ ms \end{aligned} \tag{4.9}$$

The major cycle is then calculated using Eq. (4.2):

$$M = lcm(P_i) = lcm(60, 90, 120) = 360\ ms \tag{4.10}$$

The possible values of the minor cycle are given by Eq. (4.3), taking into account also the minor cycle constraint imposed by the network utilization formula:

$$\begin{aligned}
&1.\quad m \le \min(D_i) = \min(60, 90, 120) = 60\ ms \Rightarrow m \le 60\ ms \\
&2.\quad m \ge \max(C_i) + \text{TRT} - C_{Token} = \max(2.2, 1.5, 1) + 8 * 0.85 - 0.85 = 8.15\ ms \\
&\qquad\qquad \Rightarrow m \ge 8.15\ ms \\
&3.\quad \exists k : M = km \Rightarrow m = 22.5, 24, 30, 36, 40, 45, 60\ ms \\
&4.\quad \forall i : m + (m - \gcd(m, P_i)) \le D_i \Rightarrow m = 22.5, 24, 30, 36, 40, 60\ ms
\end{aligned} \tag{4.11}$$

Once the major cycle and the possible values of the minor cycle have been obtained, a scheduling is searched for the highest possible minor cycle duration ($m = 60\ ms$). Given that a scheduling is found, as shown in Fig. 4.11, the set of data flows is schedulable. As R1 is the token master, the token path is R1-R2-R3-R4-R5-R4-R3-R2-R1. The drop packet is used to prevent R2 from retransmitting a token packet to R1 in cases where R1 has to retain the token packet during a time higher than the stipulated time-out. This packet is not usually depicted in the cyclic scheduling because it is scheduled at run-time depending on the free time in every minor cycle. Furthermore, as shown in Fig. 4.11, WICKPro can assign a different token holding time to every node in every minor cycle.

Experiment We carried out a laboratory experiment that lasted 1,000 seconds and whose results are shown in Table 4.5. This table presents the time that the minor cycles should be theoretically used, and the mean and the standard deviation measurements of the six minor cycles. In this experiment, as in Test 4.1, the minor cycles with retransmissions have been removed to analyze the nominal duration of the minor cycles. These measurements show that the experiment was successful, given also that minor cycles without retransmissions were not overrun.

Table 4.5: Minor cycle measurements in Test 4.3

MiC number	Theoretical MiC utilization (ms)	MiC measure: mean ± std dev (ms)
1	26.9	16.43 ± 0.38
2	17.1	11.18 ± 0.16
3	26.9	16.14 ± 0.27
4	21.6	15.27 ± 0.15
5	19.4	12.35 ± 0.38
6	21.6	15.23 ± 0.14

4.9 Conclusions

In this chapter, we have presented the WICKPro protocol. WICKPro is a MAC and network protocol build on top of the IEEE 802.11 standard that can manage real-time traffic with hard deadlines in an error-free scenario by using a token-passing scheme and a cyclic packet scheduler. The scheduling complexity must be carefully analyzed given a set of data flows to ensure real-time capabilities.

In laboratory experiments, we have shown that WICKPro sacrifices end-to-end throughput to provide flexibility and HRT traffic support, which are two of the drawbacks of specific TDMA protocols for WMNs with chain topologies that implement spatial reuse, namely Ripple, TDS and RMP.

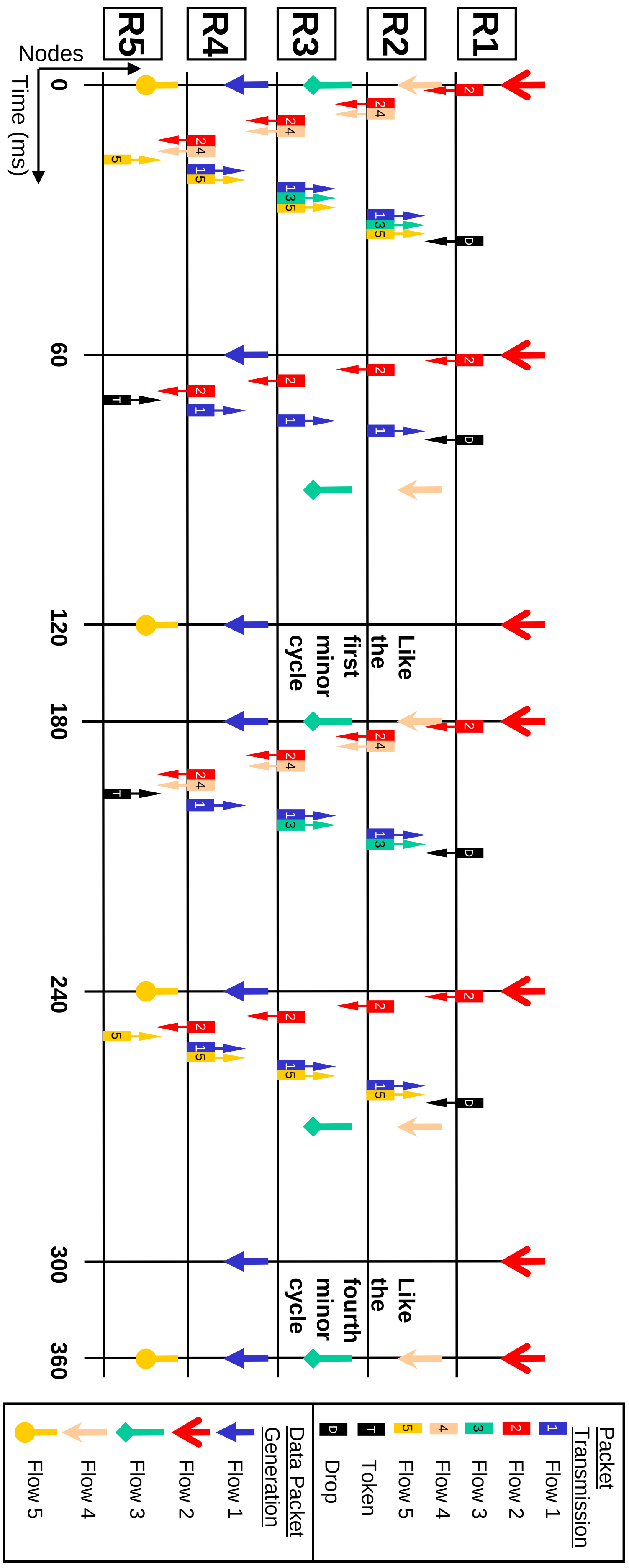


Figure 4.11: Packet scheduling calculated by WICKPro in Test 4.3. If the node that holds the token transmits some data packet, it sends the token packet as a piggyback packet along with the last data packet transmitted

5 WICKPro with Firm Real-Time Traffic Support

As WMNs are usually error-prone, the scheduling calculated by WICKPro will not be executed as expected and HRT traffic will not be supported. For this reason, in this chapter we aims at supporting FRT traffic by carrying out a fault-tolerant design that provides probabilistic scheduling guarantees, as explained in Section 2.1.2. In our framework, a fault is an erroneous packet reception and, although packet losses can be counterbalanced by retransmissions, this can jeopardize the deadlines of the supported traffic and it is therefore necessary to take into account retransmissions in the schedulability analysis.

FRT-WICKPro is based on both an off-line and an on-line packet scheduler. The off-line packet scheduler presented in Chapter 4 is enhanced to reserve time for packet transmissions and retransmissions. The time for retransmissions is calculated based on the set of supported data flows, the mean PDR between one-hop neighbors and a fault model, specifically a memoryless packet loss model. To compute the PDR, we design and incorporate to WICKPro a PDR calculation method. Moreover, an on-line packet scheduler is implemented with the help of a packet synchronization policy to drop packets when congestion appears, thereby implementing a synchronous token-passing scheme that fulfills strictly minor cycles. Next, we present the on-line packet scheduler (Section 5.2), the off-line packet scheduler enhancement (Section 5.3) and the PDR calculation method (Section 5.4), as well as simulation, laboratory and field experiments to evaluate FRT-WICKPro (Section 5.5). In Section 5.5, we also compare FRT-WICKPro with RT-WMP.

As FRT-WICKPro is designed to handle communication between routers, it does not manage client mobility. However, it could also be applied to WMNs where clients have reduced mobility and are always within the coverage area of the same serving router, e.g., a mobile robot in an assembly line.

5.1 Relevant Publications

The work presented in this chapter was mainly published in the following papers:

- [Aisa 11] J. Aisa and Jose L. Villarroel. *The WICKPro protocol with the Packet Delivery Ratio metric.* Computer Communications, vol. 34, no. 17, pages 2047-2056, 2011.

- [Aisa 15] J. Aisa and J.L. Villarroel. *Supporting Firm Real-Time Traffic in Fault-Tolerant Real-Time Systems based on Cyclic Scheduling - The WICKPro Protocol*. In IEEE International Symposium on Industrial Embedded Systems (SIES), pages 1-10, June 2015.

5.2 On-line Packet Scheduler

FRT-WICKPro carries out congestion control by using two congestion avoidance mechanisms: the admission control procedure explained in Chapter 4 and the on-line packet scheduler presented in this section. The on-line packet scheduler, along with the packet synchronization, allows FRT-WICKPro to fulfill strictly minor cycles. For this task, the on-line packet scheduler drops the packet transmissions that have not been transmitted when the minor cycle finishes, thereby letting the token master start the next minor cycle at the appropriate time.

Let us illustrate the working of this on-line packet scheduler with the WMN depicted in Fig. 5.1 where there are seven routers. In the so-called Test 5.1, this network supports two data flows with the features shown in Table 5.1. The transmission time is actually the piggyback transmission time and the retransmission time F is 6.27 ms because the time-out is 4.18 ms. Test 5.1 considers four periods of generation that yield four schedules where zero, one, two and three retransmissions are allocated, respectively (these calculations are detailed in Section 5.3.1). In Fig. 5.2, we can observe three examples of transmissions in a minor cycle that has free time to carry out one retransmission. In Fig. 5.2a, as there is no packet loss, we can see the time scheduled for packet transmissions (Sch_{Tx}) and the free time that can be used for packet retransmissions (Sch_{ReTx}). Fig. 5.2b shows a situation where there is just one packet loss and the minor cycle is therefore fulfilled. Conversely, Fig. 5.2c illustrates an example with two packet losses where the on-line packet scheduler is invoked to avoid R4 transmitting a packet of flow 2. The on-line packet scheduler is actually invoked by all nodes that have pending transmissions or ACKs, namely R2, R3, R4 and R5. In this way, the token master (R1) starts the next minor cycle at the appropriate time and the minor cycle is not overrun.

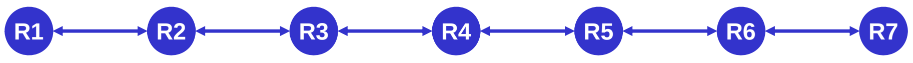
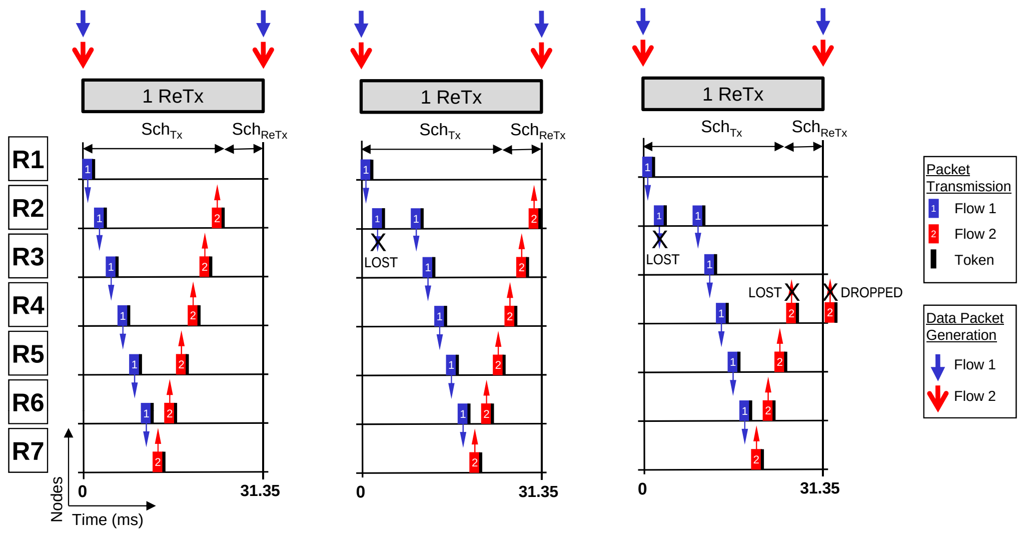


Figure 5.1: Scenario 5.1. WMN with seven routers employed in Tests 5.1 and 5.2

Table 5.1: Communication requirements in Test 5.1

Flow	Packet Size (bytes)	C (ms)	P (ms)	F (ms)	src	dst
1	1000	2.09	25.08, 31.35, 37.62 and 43.89	6.27	R1	R7
2	1000	2.09	25.08, 31.35, 37.62 and 43.89	6.27	R7	R1

The implementation of the on-line packet scheduler is performed as follows. When a node receives the token, it transmits a scheduled packet only if there is enough time in the current minor cycle, otherwise it waits for a new token reception meanwhile the token master triggers the next minor cycle. For this reason, this algorithm requires that all nodes have the same temporal reference. To this end, we define a field in the token and NACK packets called *FreeTimeMiC* that indicates



(a) No packt loss **(b)** One packt loss **(c)** Two packet losses (the on-line packet scheduler is invoked)

Figure 5.2: Three examples of transmissions in a minor cycle with one retransmission allocated in Test 5.1

the free time in the current minor cycle (more details about the WICKPro packets can be found in Appendix A). This field is initialized by the token master at the beginning of every minor cycle and decreased every time a transmission (retransmissions included) is carried out in the network. This synchronization through the *FreeTimeMiC* field is also employed to synchronize all the multi-hop flows in source nodes.

Computing systems that employ cyclic scheduling may be left in an inconsistent state if minor cycles cannot be overrun, for example if a task is aborted while updating some shared data [Buttazzo 11]. In wireless networks, packets are dropped when minor cycles are aborted, but this is not a problem because applications are designed to tolerate packet losses. The advantage of fulfilling minor cycles is that long data packet delays are avoided since the token packet is regenerated in the token master in case the token rotation is higher than the minor cycle. Otherwise, the token rotation time could increase considerably. Moreover, every minor cycle can be analyzed independently and schedulability analysis is therefore easier.

5.3 Off-line Packet Scheduler for FRT Support

Traffic Model We assume a set of N_{MH} multi-hop flows with firm criticality where each $MH_i = (src_i, dst_i, C_i, P_i, F_i, DDR_i)$. The DDR is the QoS metric to satisfy, i.e., it is the minimum percentage of data packets that must be delivered from source to destination[1]. The fault recovery time F is the worst-case retransmission time. To find out this value, we must analyze the two retransmission mechanisms

[1]As commented in Section 2.2, the DDR evaluates the QoS level provided by the network to every data flow, while the PDR measures the quality individually for each link. Thus, FRT-WICKPro measures the PDR in all the network links and calculates the DDR of every data flow based on this measured PDR.

implemented by WICKPro: one based on implicit ACK or time-out (TO ReTx), and the other on NACK (NACK ReTx). On the one hand, the worst-case NACK-based retransmission time (F_{NACK}) equals the time to transmit the NACK packet and the largest packet in the network:

$$F_{NACK} = C_{NACK} + C_{Max} \tag{5.1}$$

On the other hand, the worst-case timeout-based retransmission time (F_{TO}) corresponds to the time-out plus the maximum packet transmission time. The time-out in turn must be greater than or equal to the maximum packet transmission time. Hence, F_{TO} is greater than F_{NACK} because the maximum packet transmission time is greater than the NACK packet transmission time:

$$F_{TO} = TO + C_{Max} \geq 2C_{Max} > F_{NACK} \tag{5.2}$$

Packet Scheduling Algorithm The off-line cyclic packet scheduler calculates the free time necessary in every minor cycle to satisfy the target DDR. To this end, we take the next three decisions to design our framework:

- As explained earlier, minor cycles are always fulfilled in a way that the token master triggers minor cycles continuously even if it does not have the token packet. Thus, a synchronous token-passing scheme is implemented.

- The specification of the target DDR turns into the target Percentage of Minor Cycles satisfied (P-MiC) because it is simpler to manage. Particularly, the P-MiC is the probability to complete all the transmissions scheduled within a minor cycle. Thus, the off-line packet scheduler uses a worst-case scenario: all the data packets scheduled in a minor cycle will be considered lost if this minor cycle is exceeded, however, some of them will be actually delivered in real working conditions. For example, in Fig. 5.2c, both flow 1 and flow 2 will be considered lost in the scheduling calculation, but the flow 1 will be actually delivered in practice.

- The off-line packet scheduler only reserves multiples of F_{TO} so that the time margin for retransmissions will equal $k * F_{TO}$, where $k \geq 0$. Therefore, we use the terms retransmissions and TO ReTx interchangeably when talking about retransmission reservation.

The algorithm is similar to the procedure described in Section 4.7.2 but taking into account that there must be certain amount of time allocated for retransmissions:

1. The highest possible minor cycle duration is selected for the heuristic search.

2. The number of transmissions of every one-hop flow in the major cycle is obtained.

3. The possible minor cycles where every transmission can be scheduled is computed.

4. Every transmission is then scheduled one by one based on an RMS algorithm that uses depth-first search. One transmission is assigned to the first of its possible minor cycles as long as there is enough free time in that minor cycle for the transmission in question as well as enough free time for retransmissions. The required retransmission time is calculated as follows. Given the set of data and token transmissions scheduled in a minor cycle, the mean PDR between one-hop neighbors and a memoryless packet loss model, the P-MiC is calculated considering that there is no time allocated for retransmissions. If this value is lower than the target DDR, the same calculation is carried out considering one retransmission allocated, whenever there is enough time in the minor cycle for carrying out this retransmission. And so on, until the P-MiC is higher than or equal to the target DDR, or the minor cycle cannot allocate more retransmissions[2]. The scheduling search is like the algorithm of Section 4.7.2. Backtracking is used if a transmission cannot be scheduled in any minor cycle, and if it is not possible to find a scheduling for the current m, this minor cycle duration is discarded and the algorithm comes back to the step (1), where the highest remainder m will be selected.

Specific Solution We implement the packet scheduling algorithm for two cases: no node has more than one Transmission Per Token Holding (TPTH) and no node has more than two TPTH. If a node wanted to carry out more than two TPTH, packets could be merged into one or two packets. Let us illustrate both calculations with two examples, Test 5.1 and Test 5.2, respectively. Both tests are carried out in the Scenario 5.1 depicted in Fig. 5.1. Without loss of generality, the PDR is assumed to be the same for all the links and packet sizes, 97% to be precise. To better explain our framework, in Tests 5.1 and 5.2 we do not calculate the P-MiC given a set of multi-hop flows with its communication requirements. Instead, we consider several flows with the same communication requirements, except that the source and destination are different, in a way that the major cycle has only one minor cycle that is equal to the period of the multi-hop flows. In this situation, we calculate the P-MiC and the time consumed in four schedules where the period of the multi-hop flows is changed to let the minor cycle allocate zero, one, two and three retransmissions, respectively.

5.3.1 One Transmission per Token Holding

Definition of Test 5.1 As commented in Section 5.2, the network in Fig. 5.1 supports two data flows with the features shown in Table 5.1. Fig. 5.3 illustrates the scheduling calculated with zero and one retransmissions allocated where each multi-hop flow is partitioned in six one-hop flows[3]. The minimum minor cycle duration (without any retransmission allocated) is equal to the time to transmit six piggyback packets of flow number 1 and six piggyback packets of flow number 2, i.e, 25.08 (6*2.09 + 6*2.09) ms. The minor cycle duration with zero, one, two and three

[2]With this implementation, the stop condition is the quickest possible, given that it is necessary to compute the P-MiC with zero retransmissions to know the P-MiC with one retransmission, and so on. Otherwise, it would be better to calculate the number of retransmissions that can be carried out in a minor cycle and compute the P-MiC with this number of retransmissions.

[3]As stated in Chapter 4, the drop packet is not usually depicted in the cyclic scheduling because it is scheduled at run-time depending on the free time in every minor cycle.

retransmissions allocated is therefore 25.08, 31.35 (25.08 + 6.27), 37.62 (25.08 + 2*6.27) and 43.89 (25.08 + 3*6.27) ms, respectively.

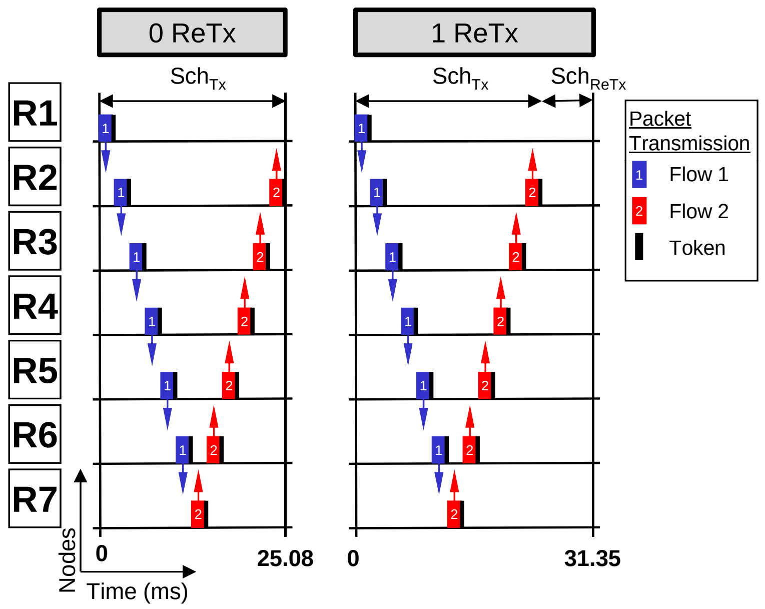
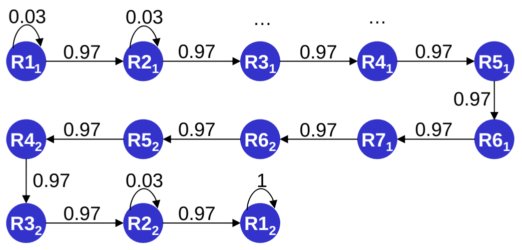


Figure 5.3: Cyclic packet scheduling in Test 5.1 with zero and one retransmissions allocated

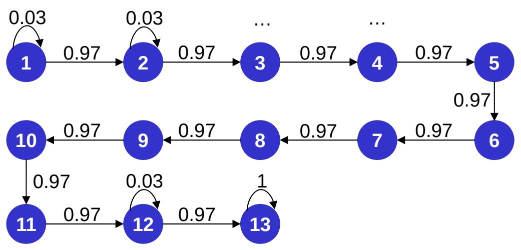
Scheduling Calculation of Test 5.1 To calculate the P-MiC, we define a homogeneous Discrete-Time Markov Chain (DTMC) [Bolch 05] for every minor cycle, one in this case. A DTMC can be used because transition probabilities are memoryless and do not depend on time. Every state in the chain represents the node that holds the token, as shown in Fig. 5.4, and transitions describe token pass. As the token only can be passed to the next node in the token path or stay in the current node if the token is lost, a token pass is characterized with a probability of success and failure. For simplicity, implicit ACKs are not included in the model but the loss of accuracy will be small whenever success probability is much higher than failure probability. In Fig. 5.4, the success and failure probabilities are respectively 0.97 and 0.03 (not all the failure probabilities are depicted for clarity). The number of states (n) in the chain is $2*(N_R-1)+1$, which is the necessary number to describe a minor cycle in which the token goes out and comes back to the master node. In Fig. 5.4, as the number of routers is 6, $n = 13$. It should be noted that this DTMC is an absorbing Markov chain because the last state has a transition to itself with probability 1, as the token master keeps the token until the next minor cycle begins.

Based on this DTMC, we obtain the one-step transition matrix Π that represents the transitions between any pair of states. Eq. (5.3) illustrates the general expression of Π that has the following features: (i) dimension is n x n, (ii) $0 \leq p_{ij} \leq 1$, and (iii) the elements in each row of the matrix sum up to 1 ($\sum_{j=1}^{n} p_{ij} = 1, \forall i \in n$). The 13 x 13 square matrix Π of Test 5.1 is shown in Eq. (5.4)[4].

[4]As this matrix can accommodate different PDR values, the algorithm is general in this sense. However, using the same PDR value allows us to validate our algorithm by using the formulas of a Binomial distribution (not included in this dissertation), because transmissions in a minor cycle are a sequence of success/failure experiments, or Bernoulli trials.



(a) State = Router that holds the token



(b) State = Number of state

Figure 5.4: DTMC in Test 5.1 where nodes have one TPTH. In (a), every state shows the router that holds the token (R1-R7) along with a subscript that indicates the number of times that the router has held the token in the current minor cycle, and in (b) every state is described with its state number

$$\Pi = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \tag{5.3}$$

$$\Pi = \begin{pmatrix}
0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0.97 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \tag{5.4}$$

The state vector *P(i)* denotes the probability of being in each state at step i, i.e., after i transmissions:

$$\begin{aligned} P(i) = &\{P_1(i), P_2(i), ..., P_n(i)\} \\ \text{where:} & \\ 0 \leq\ & P_m \leq 1 \\ \sum_{m=1}^{n} &P_m = 1 \end{aligned} \tag{5.5}$$

P(i) can be expressed as a function of the one-step transition matrix Π as shown in Eq. (5.6). This means that, given an initial state vector ($P(0)$), the DTMC evolves step by step according to Π. In this way, we can calculate the probability of transiting from a state S_0 at step 0 to a state S_k after i steps by defining appropriately the initial state ($P(0)$) and raising the matrix Π to the power of i.

$$P(i) = P(0) * \Pi^i \tag{5.6}$$

In Test 5.1, we are interested in calculating the probability that the token goes out and comes back to the master node with zero, one, two and three retransmissions. Eq. (5.7) exhibits the initial state of the DTMC, indicating that the token is in the state 1, in other words, in the token master. After *n - 1* transmissions, the token can reach the token master again, i.e., the DTMC can stay in the state n. As in Test 5.1 there are 13 states (n = 13), we show the transition matrix raised to the power of twelve (Π^{12}) and the state vector $P(12)$ in Eq. (5.8) and Eq. (5.9), respectively. Π^{12} describes all the transition probabilities between any pair of states after 12 transmissions, while $P(12)$ shows the probability of being in states 1 to 13 given that initially the DTMC was in the state 1. At this point, we select the probability of being in state 13: 69.38%.

$$P(0) = \{1,0,0,0,0,0,0,0,0,0,0,0,0\} \tag{5.7}$$

$$\Pi^{12} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0.0003&0.0045&0.0438&0.2575&0.6938 \\ 0&0&0&0&0&0&0&0&0&0.0003&0.0045&0.0438&0.9514 \\ 0&0&0&0&0&0&0&0&0&0&0.0003&0.0045&0.9952 \\ 0&0&0&0&0&0&0&0&0&0&0&0.0003&0.9997 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \tag{5.8}$$

$$P(12) = P(0) * \Pi^{12} = \{0,0,0,0,0,0,0,0,0.0003,0.0045,0.0438,0.2575,0.6938\} \tag{5.9}$$

If one, two and three retransmissions are permitted, the transition matrix must be raised to the power of thirteen, fourteen, and fifteen, respectively, because this is the number of possible transmissions in every case. Then, we select the probability that the token had passed from the state 1 to the state 13. For example, in the case of one retransmission, Eq. (5.10) and Eq. (5.11) describe Π^{13} and $P(13)$, in a way that the probability of being in state 13 after thirteen transmissions is 94.36%. Finally, we calculate that the P-MiC with zero, one, two and three retransmissions allocated will be 69.38, 94.36, 99.23 and 99.92% respectively, i.e, the less the offered load, the higher the P-MiC and the DDR.

$$\Pi^{13} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0.0005&0.0057&0.0502&0.9436 \\ 0&0&0&0&0&0&0&0&0&0&0.0004&0.0057&0.9938 \\ 0&0&0&0&0&0&0&0&0&0&0&0.0004&0.9995 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \tag{5.10}$$

$$P(13) = P(0) * \Pi^{13} = \{0,0,0,0,0,0,0,0,0,0.0005,0.0057,0.0502,0.9436\} \tag{5.11}$$

5.3.2 Two Transmissions per Token Holding

Definition of Test 5.2 The network in Fig. 5.1 supports four data flows with the features shown in Table 5.2. Moreover, the NACK transmission time is 0.75 ms. Fig. 5.5 illustrates the scheduling calculated without any retransmission allocated. The minimum minor cycle duration is equal to the time to transmit 12 data packets and 12 piggyback packets, i.e, 47.4 ms. This value is obtained because the time to transmit one data packet and one piggyback packet in the same token holding is 3.95 ms, as detailed below in Section 5.5.1.2. The minor cycle duration with zero, one, two and three retransmissions allocated is therefore 47.40, 53.67, 59.94 and 66.21 ms, respectively.

Table 5.2: Communication requirements in Test 5.2

Flow	Packet Size (bytes)	C (ms)	P (ms)	F (ms)	src	dst
1	1000	2.09	47.40, 53.67, 59.94 and 66.21	6.27	R1	R7
2	1000	2.09	47.40, 53.67, 59.94 and 66.21	6.27	R1	R7
3	1000	2.09	47.40, 53.67, 59.94 and 66.21	6.27	R7	R1
4	1000	2.09	47.40, 53.67, 59.94 and 66.21	6.27	R7	R1

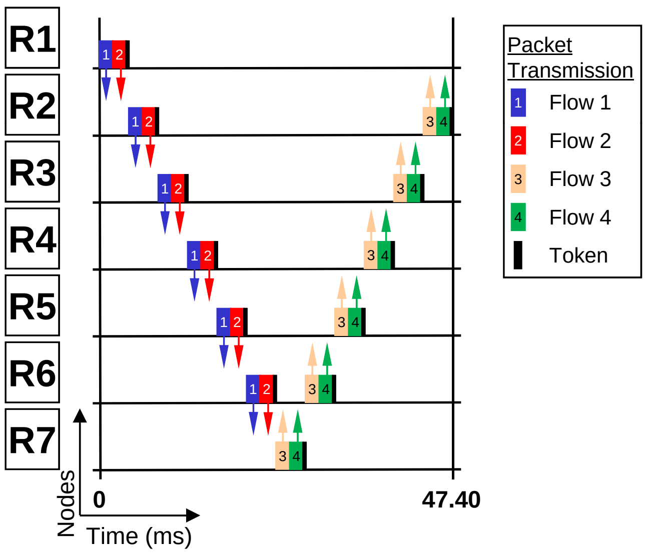


Figure 5.5: Cyclic packet scheduling in Test 5.2 without retransmissions allocated

Scheduling Calculation of Test 5.2 In this case, since all nodes carry out two TPTH, the calculation of the P-MiC must take into account the two retransmission mechanisms which have different duration and different number of packet transmissions[5]. This is a problem because a DTMC only considers the probability to transit from one state to another, but not the time spent in every state. In Table 5.3, we show the relationship between probability and time consumption with one TPTH and the unknown relationship with two TPTH. Note that PB stands for Piggy-Back. This table takes into account that the minor cycle is completed in exactly the number of transmissions indicated in the column *Tx*. It can be seen a problem with two TPTH when 26 transmissions are considered because the completion of the minor cycle may have been with two TO ReTx or with one NACK ReTx, which consume a different amount of time. Thus, the probability is obtained properly with the one-step transition matrix P but not the time consumed. One possible solution is to use Markov Chains with Costs and Rewards [Bolch 05] but the mathematics developed for these models focuses on average costs, e.g., the probability to transit from a state i to a state j in a specific number of steps (packet transmissions) and what is the average time to do it, whereas we are interested in the exact cost.

Hence, we propose an algorithm that uses a DTMC but does not use the one-step transition matrix. This DTMC is shown in Fig. 5.6 where only the transmissions in the first two routers are completely depicted for clarity. The rest of the DTMC is analogous to this part. PDR_{D1} is the PDR of the first data packet transmitted,

[5]For example, if one data packet has to be retransmitted, a TO ReTx only needs a correct packet reception (the packet to retransmit), whereas a NACK ReTx requires that two packets be correctly received (the NACK packet and the packet to retransmit).

Table 5.3: Relationship between probability and time consumption after a complete minor cycle using a DTMC in Test 5.1 (1 TPTH) and Test 5.2 (2 TPTH)

THTH	Tx	Prob.	Time consumed
1	12	Π^{12}	12 PB Tx
	13	Π^{13}	12 PB Tx + 1 TO ReTx
	14	Π^{14}	12 PB Tx + 2 TO ReTx
2	24	Π^{24}	12 Data Tx + 12 PB Tx
	25	Π^{25}	12 Data Tx + 12 PB Tx + 1 TO ReTx
	26	Π^{26}	12 Data Tx + 12 PB Tx + 2 TO ReTx? 12 Data Tx + 12 PB Tx + 1 NACK ReTx?

PDR_{PB1} is the PDR of the first data packet transmitted along with the token packet, PDR_{PB2} is the PDR of the second data packet transmitted along with the token packet and PDR_{NACK} is the PDR of the NACK packet. Let us illustrate this process considering that R1 holds the token packet and transmits two data packets. The first data packet is transmitted with a success probability of PDR_{D1} from state $R1_1$. If this packet is properly received, R1 transmits the second data packet along with the token packet with a success probability of PDR_{PB2} from state $R1_2$. This packet is retransmitted until R1 receives an implicit ACK packet. When the second data packet is properly received, R2 holds the token and can transmit its two data packets (state $R2_1$). Conversely, if R2 receives the piggyback packet with the second data packet but it did not receive the first data packet, R2 sends back a NACK packet with success probability PDR_{NACK} to R1 indicating that it only received the second data packet (state $R2_5$). This NACK packet also waits for an implicit ACK and therefore employs timeout-based retransmission. When R1 receives the NACK packet, it retransmits the first data packet with a success probability of PDR_{PB1} from state $R1_4$, using also timeout-based retransmission. When R2 receives the first data packet, it can transmit its two data packets (state $R2_1$).

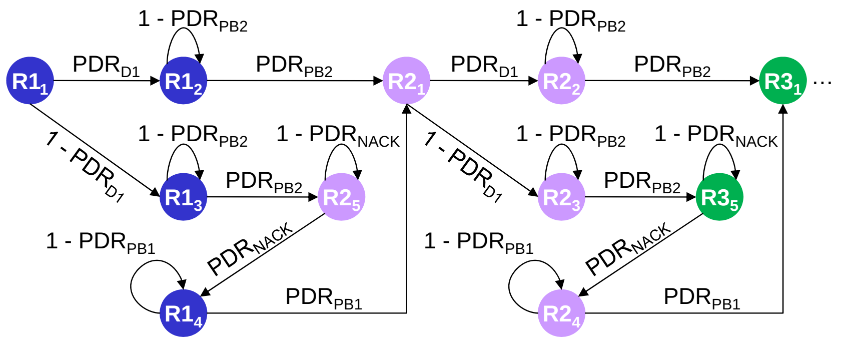


Figure 5.6: DTMC in Test 5.2 where nodes have two TPTH. R1 and R2 transmit two data packets in a way that the second data packet is actually a piggyback packet that contains the token packet. If required, considering that R1 holds the token packet, the first data packet is retransmitted based on NACK packet (state $R2_5$), while the second data packet is retransmitted based on the lack of implicit ACK after a time-out (state $R1_2$ and $R1_3$). If the first retransmission of the first data packet is lost, then the first data packet is also retransmitted based on time-out since it also contains the token packet (state $R1_4$)

Our algorithm works in two phases. Firstly, each link is analyzed independently. The off-line packet scheduler allocates up to 3 TO ReTx so that there are seven ways to successfully transmit the data packet and the piggyback packet in every link. The relationship between probability and time consumption for all of them are shown in Table 5.4.

Table 5.4: Relationship between probability and time consumption in a link with the proposed method in Test 5.2

#TO ReTx	#NACK ReTx	Time (ms)	Prob. (%)
0	0	3.95	94.0900
0	1	6.79	2.7380
1	0	10.22	2.8227
1	1	13.06	0.2464
2	0	16.49	0.0847
2	1	19.33	0.0148
3	0	22.76	0.0025

Secondly, the off-line packet scheduler calculates the probability and the time consumed in the completion of a minor cycle, i.e., when the token goes out and comes back to the master node. The sixteen ways to complete a minor cycle with up to 3 TO ReTx are illustrated in Table 5.5. For example, if 1 TO ReTx is allocated by the off-line packet scheduler, the minor cycle will be satisfied in the case of 0 retransmissions, 1 NACK ReTx, 2 NACK ReTx and 1 TO ReTx. Based on Table 5.5, in this example the P-MiC with zero, one, two and three retransmissions allocated will be 48.14, 84.97, 96.98 and 99.53%, respectively.

Table 5.5: Relationship between probability and time consumption in a minor cycle with the proposed method in Test 5.2

#TO ReTx	#NACK ReTx	Time (ms)	Cumulative Prob. (%)
0	**0**	**47.40**	**48.14**
0	1	50.24	64.95
0	2	53.08	67.64
1	**0**	**53.67**	**84.97**
0	3	55.92	85.24
1	1	56.51	92.30
0	4	58.76	92.31
1	2	59.35	93.60
2	**0**	**59.94**	**96.98**
0	5	61.60	96.99
1	3	62.19	97.13
2	1	62.78	98.71
0	6	64.44	98.71
1	4	65.03	98.72
2	2	65.62	99.05
3	**0**	**66.21**	**99.53**

It should be highlighted that this algorithm may be used with links with one

TPTH by taking into account only the TO ReTx in the calculation of those links. Thus, this algorithm may be generically employed with links with one and two TPTH as well as with different PDR for every link and packet size.

5.4 PDR Calculation Method

The PDR of the link between node i and node j (PDR$_{i,j}$) is defined as the ratio between the packets received by node j and the packets transmitted by node i. The key issue is therefore that the receiver knows how many packets were sent by the transmitter and that the transmitter knows how many packets were received correctly by the receiver during a specific period of time. To manage this, we consider three possible strategies. The first one is that the number of packets transmitted to measure the PDR (probe packets) is fixed in a period of time so that the receiver can compute PDR$_{i,j}$ properly and transmit it afterward. However, we have found this method too rigid for us because it would restrict the scheduling calculation. The second one is that the token packet has a field that indicates the number of retransmissions carried out within a token pass. On the one hand, in the case of using explicit ACK, this method may be very accurate, depending on how the PDR is specifically measured. On the other hand, if implicit ACK is used, it is inherently not so accurate. The main problem is that the token may be passed successfully but the implicit ACK may be lost, and vice versa, which generates maladjustments in the token-passing process because some retransmissions can be missed. Although this lack of precision might not be important depending on the scenario, it is clear that it exists.

For this reason, we propose a third strategy where the so-called probe packet is filled by all nodes with the number of packets transmitted and received in the last minor cycles. In this way, all nodes can calculate locally the PDR between any pair of nodes by properly computing these values, without the need to send explicitly the PDR. Concerning implementation, the PDR depends on the bit rate and the packet size. We have assumed that the transmission rate is fixed and the number of supportable flow types in the network is at most four. We now assume that there are three types of data packet sizes, thus the PDR for the four different packet sizes must be calculated, including the regular token packet. As the regular token packet and the other control packets have a very similar size, we only calculate the PDR for the regular token packet and suppose that the other packets have the same PDR. We take advantage of the cyclic packet scheduling that is known by all nodes and only the non-scheduled events in the cyclic packet scheduling are sent, i.e., the retransmissions and the non-scheduled receptions. More details about the implementation of probe packets can be found in Appendix A.4.

The PDR is computed in the PDR calculation state because the off-line packet scheduler must use this metric before admitting a new multi-hop flow. For this purpose, token rotations with probe packets, called probe rotations, are carried out where the probe packet size is changed to measure the PDR for the four different packet sizes. After a specific number of probe rotations, depending on the accuracy necessary, the mean PDR between one-hop neighbors is estimated. Moreover, in the active state, the PDR must be updated, but this is left for future research. In that case, WICKPro could take advantage of current data communications as well as carry out probe rotations when there was enough free time in a minor cycle, in order to keep updated all packet sizes in all links. Note that only the one-hop neighbor PDR is interesting for the off-line packet scheduler but the PDR between any pair of nodes could be obtained with this method.

5.5 Evaluation

In this section, we assess FRT-WICKPro in laboratory and field experiments (Sections 5.5.1 and 5.5.2, respectively). We developed FRT-WICKPro in C and performed all the experiments by using nodes consisting of minimal embedded and dedicated hardware (100x160 mm PcEngines ALIX3D3 board) and Atheros chipset-based wireless cards that ran the MaRTE OS operating system [Rivas 01]. Some photographs about the hardware employed can be found in Appendix B.1. Regarding the physical level configuration, the transmission frequency was 5.3 GHz (802.11a) at 6 Mbps using CSMA/CA without RTS/CTS, where hardware-level ACKs were disabled due to the use of broadcast transmissions. Moreover, we also carried out a simulation considering a busty packet loss model (Section 5.5.3) and a comparison between FRT-WICKPro and RT-WMP (Section 5.5.4).

5.5.1 Laboratory Experiments

Section 5.5.1.1 evaluates the PDR calculation method whereas Section 5.5.1.2 implements the Tests 5.1 and 5.2 to verify the correctness of the FRT traffic support mechanisms, specially the off-line and the on-line packet schedulers. The seven nodes used were very near to each other so that all the PDRs were close to 1. Consequently, to achieve that all the links and packet sizes had a PDR equal to 97%, we introduced deliberate errors in the software developed based on a random function.

5.5.1.1 PDR Calculation Method

We implemented the network depicted in Fig. 5.1 and conducted an experiment where the PDR calculation state was defined to last 40,000 minor cycles. In this way, 10,000 minor cycles were used to compute the PDR for each one of four packet sizes, whose lengths are 14, 100, 500 and 1,000 bytes in this case. The packet errors were introduced in the transmission module because it was easier to show that the method works properly. For this purpose, the transmitter node sends a packet whose source and destination addresses do not correspond to any node. Table 5.6 shows the PDR measured between R2 and R1. The combined PDR of the four packet sizes is 0.970748, which matches the value computed after all the executions of the random function. We have not shown all the cases checked because it is not relevant but the conclusion is that we can measure the PDR between one-hop neighbors appropriately.

5.5.1.2 FRT Traffic Support

First of all, we will present more details of the minor cycle duration calculation. As commented earlier, Tests 5.1 and 5.2 support multi-hop flows that have data

Table 5.6: Packet Delivery Ratio measured in a laboratory experiment

Packet Size (Bytes)	Packets Rx_1	Packets Tx_2	$PDR_{2,1}$
14	10299	10596	0.971970
100	10321	10616	0.972211
500	10297	10625	0.969129
1,000	10300	10622	0.969685

packets of 1,000 bytes (without headers). According to Eq. (5.12)[6], the nominal worst-case transmission time of data packets is 1.573 ms (1,005 bytes), the nominal worst-case transmission time of piggyback packets is 1.585 ms (1,015 bytes) and the nominal worst-case token transmission time is 0.249 ms (13 bytes). To compute the worst-case token transmission time, we considered the establishment packet instead of the regular token packet because it is the largest token packet (although there is no difference in practice due to their small difference). These values were rounded out to 1.58, 1.59 and 0.25 ms, respectively.

$$C^{6Mbps}_{802.11a}(\mu s) = 34 + 135 + \left(20 + 4 * \left\lceil \frac{16 + 6 + 8 * [28 + MSDU(bytes)]}{24} \right\rceil \right) \quad (5.12)$$

Moreover, two extra times must be added to the transmission time C:

$$C(ms) = C^{6Mbps}_{802.11a}(ms) + C_{log}(ms) + C_{process}(ms) \quad (5.13)$$

The first one is the time consumed by the log information added to every packet measuring 0.2 ms so that the previous nominal worst-case transmission times were converted to 1.78, 1.79 and 0.45 ms. The second time is a transmission margin that is necessary because of the operating system overhead and the code execution time at each node. This time must be taken into account once per token holding and it was empirically set to be 0.3 and 0.38 ms in the case of one and two TPTH, respectively. Thus, the transmission times of data, piggyback and token packets were 1.78, 2.09 (1.79 + 0.3) and 0.75 (0.45 + 0.3) ms, respectively, and the time to transmit consecutively one data packet and one piggyback packet was 3.95 ms (1.78 + 1.79 + 0.38). The time-out was defined to 4.18 ms to be able to monitor at least the next two packet transmissions before retransmitting. The retransmission time was therefore 6.27 ms. Finally, in the example with one TPTH, the minor cycle duration with zero, one, two and three retransmissions allocated was 25.08, 31.35, 37.62 and 43.89 ms, respectively, whereas in the example with two TPTH, the minor cycle duration with zero, one, two and three retransmissions allocated was 47.40, 53.67, 59.94 and 66.21 ms, respectively. It should be noted that, in these tests, the packet errors were introduced in the reception module because it was a more realistic scenario.

Every experiment in Tests 5.1 and 5.2 lasted 1,000 s. The P-MiC obtained in both tests is shown in Tables 5.7 and 5.8. The theoretical values (obtained in Section 5.3) and the measured values are very similar. There is a slight difference due to the erroneous implicit ACK monitoring which is more relevant when no retransmission

[6]This equation is very similar to Eq. (4.7). The only difference is that the back-off time is set to be the worst-case value (135 μs) instead of the mean value (67.5 μs).

is allocated. A solution could be to estimate this lack of precision and take it into account. In Tables 5.9 and 5.10 we can see the DDR obtained in both tests. Three issues should be pointed out based on these results. First, the DDR is always higher than or equal to the P-MiC. The reason is that the off-line packet scheduler assumes a worst-case scenario where all the scheduled transmission are considered to be lost if a minor cycle is overrun, but some transmissions are actually delivered, as stated in Section 5.3. Second, FRT-WICKPro does not provide fairness but it guarantees a minimum provision to every multi-hop flow. Third, as expected, the less the offered load (the more the free time), the more the P-MiC and the DDR become.

Table 5.7: Percentage of minor cycles satisfied in Test 5.1

	0 ReTx	**1 ReTx**	**2 ReTx**	**3 ReTx**
Theoretic values	69.38	94.36	99.23	99.92
Measured values	69.08	94.36	99.24	99.90

Table 5.8: Percentage of minor cycles satisfied in Test 5.2

	0 ReTx	**1 ReTx**	**2 ReTx**	**3 ReTx**
Theoretic values	48.14	84.97	96.98	99.53
Measured values	48.06	85.08	97.00	99.56

Table 5.9: Data Delivery Ratio (%) in Test 5.1

Flow	**0 ReTx**	**1 ReTx**	**2 ReTx**	**3 ReTx**
Flow 1	99.87	99.98	100	100
Flow 2	69.08	94.36	99.24	99.90

Table 5.10: Data Delivery Ratio (%) in Test 5.2

Flow	**0 ReTx**	**1 ReTx**	**2 ReTx**	**3 ReTx**
Flow 1	99.99	100	100	100
Flow 2	99.99	100	100	100
Flow 3	49.61	86.86	97.64	99.69
Flow 4	49.52	85.79	97.20	99.60

For a better understanding of FRT-WICKPro, Fig. 5.7 presents the histogram of the minor cycle duration measured in Test 5.1 when three retransmissions are allocated. We can observed ten events highlighted with legends from (I) to (X). The theoretic minor cycle duration with zero, one, two and three retransmissions allocated are drawn in discontinuous red lines (legends I, II, III and IV, respectively). The minor cycle measurements when there is exactly zero, one, two and three retransmissions are shown in (V), (VI), (VII) and (VIII), respectively. For example, legend (V) shows the minor cycles where there were 12 successful piggyback packet transmissions while legend (VI) depicts the minor cycles with 12 successful piggyback packet transmissions and one piggyback packet loss. Therefore, it can be easily observed that if the theoretic minor duration is set to be 31.35 ms (legend II), the minor cycles with zero (legend V) and one retransmission allocated (legend VI) will be smaller than this theoretic value and will not be overrun. We can see that the measured minor cycles are slightly lower than the theoretic minor cycle duration because the packet transmission time and the transmission margin taken into account have been calculated using the worst-case scenario. Legend (IX) shows the small percentage of times that there were 12 successful piggyback packet transmissions and one erroneous implicit ACK monitoring, which led to an useless retransmission because the token pass had actually been successful. A part of these minor cycles exceeded the theoretic minor cycle duration, which explains the slight difference between theoretic and measured values when no retransmission is allocated in Table 5.7. Moreover, legend (X) depicts the percentage of minor cycles overrun in this case. This value was 0.10% because the percentage of minor cycles satisfied was 99.90% in the case of allocating time for three retransmissions. It should also be highlighted that there are more values in the histogram besides the ten events highlighted, but they had very small occurrences; they were mainly produced due to the variability in the transmission time and the transmission margin, and the erroneous implicit ACK monitoring.

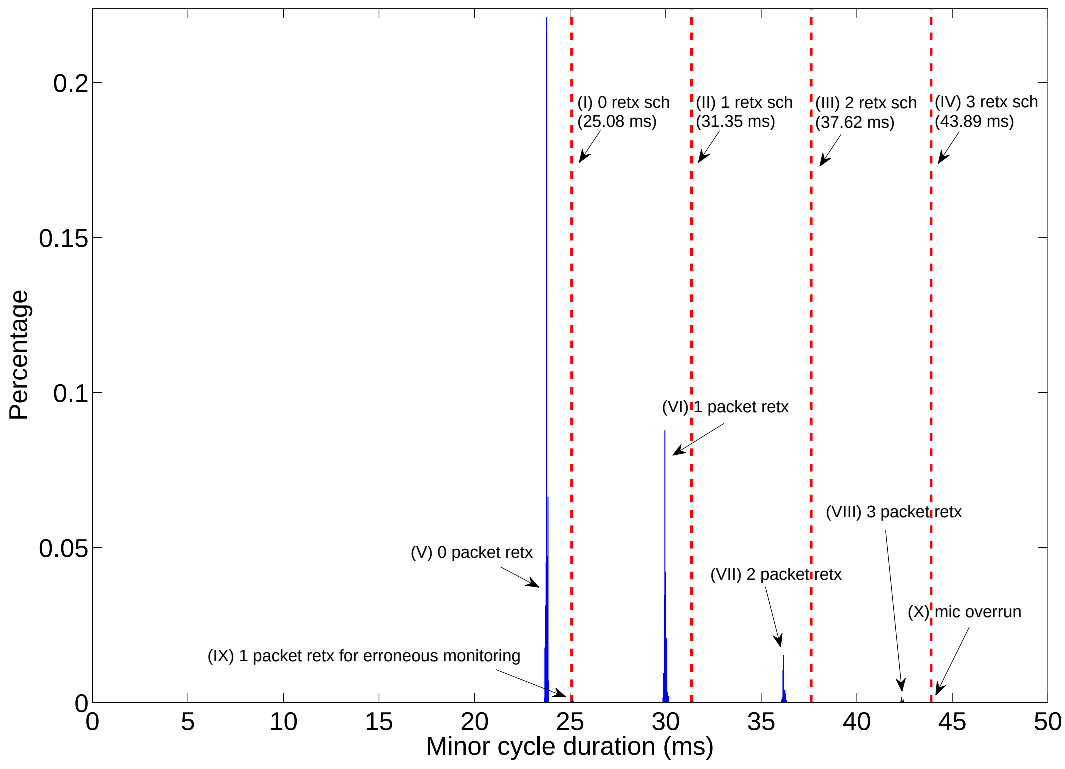


Figure 5.7: Histogram of the minor cycle duration measured in Test 5.1 when three retransmissions are allocated. The theoretic minor cycle duration with zero, one, two and three retransmissions allocated are drawn in discontinuous red lines (legends I, II, III and IV, respectively). The minor cycle measurements when there is exactly zero, one, two and three retransmissions are shown in (V), (VI), (VII) and (VIII), respectively. Legend (IX) shows the small percentage of times that there were 12 successful piggyback packet transmissions and one erroneous implicit ACK monitoring, and legend (X) depicts the percentage of minor cycles that was overrun

Hence, with these laboratory experiments, we have shown that FRT-WICKPro can support FRT traffic.

5.5.2 Field Experiment in the I3A Building

We also carried out a field experiment inside the corridors of the I3A[7] building with five routers that formed a WMN with chain topology as shown in Fig. 5.8. The experiment is equivalent to the one presented in Fig. 5.3 with no retransmission allocated, but in a network with five routers, i.e., there was one multi-hop flow from R1 to R5 and one multi-hop flow from R5 to R1.

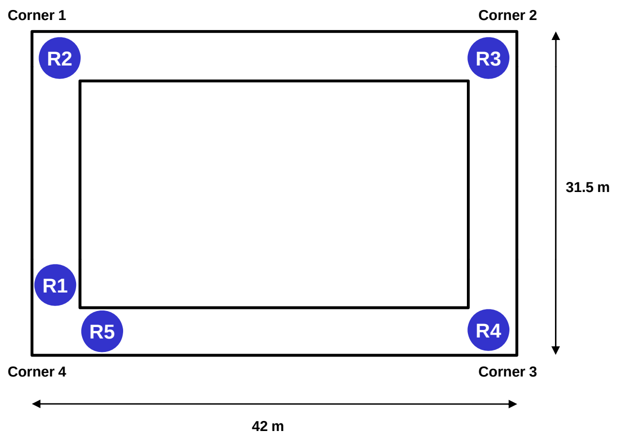


Figure 5.8: Simplified scenario of the field experiment using FRT-WICKPro

In the PDR calculation state, the mean PDR between one-hop neighbors is measured using 100,000 minor cycles, as can be seen in Table 5.11. Based on these measurements, as the packet scheduling did not have any free time for retransmissions, in the active state the expected P-MiC was 99.66%[8]. Fig. 5.9 shows the theoretical and the measured P-MiC in a experiment that lasted about 1,400 seconds. Note that the vertical axis only shows the percentage from 99 to 100%. We can observe that the measured value is very similar to the expected value. The error has a maximum of around 0.25% at the beginning of the experiment and remains below 0.1% from 150 seconds. The higher accuracy in the long-term behavior is explained by the law of large numbers since we use the mean PDR value as metric.

Table 5.11: Packet Delivery Ratio measured in the field experiment where Test 5.1 is implemented with five routers

Link	Packets Rx	Packets Tx	PDR
R1-R2	100016	100036	0.999800
R2-R3	100000	100000	1.000000
R3-R4	100004	100023	0.999810
R4-R5	100000	100000	1.000000
R5-R4	100000	100000	1.000000
R4-R3	100000	100263	0.997377
R3-R2	100000	100001	0.999990
R2-R1	100016	100053	0.999630

[7]The I3A is the Aragón Institute for Engineering Research, a research institute at Universidad de Zaragoza where the laboratory of mobile robotics is located.

[8]This value is trivially obtained by multiplying the PDR in every link: 0.999800 x 1.000000 x 0.999810 x 1.000000 x 1.000000 x 0.997377 x 0.999990 x 0.999630 = 0.996609.

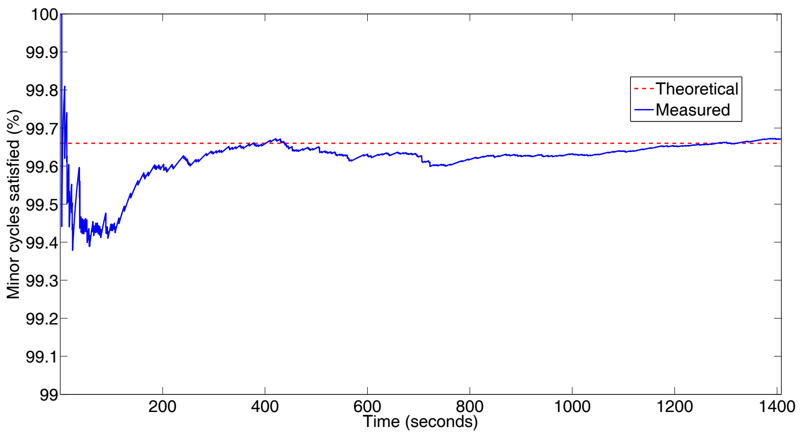


Figure 5.9: Minor cycles satisfied (%) as a function of time in the field experiment

The memoryless packet loss model is found to be quite accurate in this field experiment. We think that there are three main reasons for this behavior. First, the mean PDR is very high so that there are not many packet losses and consequently the possible error of using this model is low. Second, all nodes are fixed and multipath fading is therefore lower than in a network with mobile nodes. Third, interferences are reduced: external interference is non-existent, and there are neither intra-flow nor inter-flow interferences due to the employed token-passing scheme. The only interference is the one created because of erroneous implicit ACK monitoring, but this amount of interference is small, as shown earlier.

5.5.3 Simulation of FRT-WICKPro using a Bursty Channel

We simulated FRT-WICKPro in MATLAB to analyze the effect of a bursty packet loss model. For this purpose, we implemented WICKPro as a discrete-event system that calculates the cyclic packet scheduling and manages the token passing as WICKPro does. In our simulations, if a token packet is lost, it is considered to be lost for the receiver of the token and for the node that is monitoring the channel to acknowledge its previous token transmission. In other words, packet losses can occur but token duplication is not taken into account. This causes a slight loss of accuracy but it is only relevant when the network utilization is very high, as measured in Section 5.5.1.2. The bursty packet loss model employed in the simulation was the simple Gilbert Model explained in Section 2.3.2. All simulations were executed in 100,000 theoretic major cycles.

Table 5.12 shows the P-MiC and Table 5.13 the DDR obtained when simulating four packet loss models with PDR = 97%: one memoryless[9] and three bursty models with ABL equal to 1.11, 2 and 4. As expected, the higher the ABL, the higher the difference between the memoryless and the bursty channel is. This difference is also higher if the number of retransmissions allocated is lower, except if there is no retransmission allocated. Let us explain this behavior with the following reasoning: if there are two packet losses, there is a higher probability that they are concentrated in the same minor cycle if the packet loss pattern is bursty and, conversely, there is a higher probability that they are in different minor cycles if the channel is memoryless.

[9]We can verify that our WICKPro simulator works properly by comparing the theoretic and simulated values given in Tables 5.7 and 5.12 with a memoryless packet loss model.

For this reason, when there is no retransmission allocated, it is likely that the P-MiC will be higher if the channel has a higher ABL, since the number of packet losses is the same in the four packet models simulated (the mean PDR is the same), and packet losses are concentrated in less minor cycles when ABL is higher. It should be noted that it does not mind if the number of packet losses in a minor cycle is one or two, as the minor cycle will be overrun anyway when no retransmission is allocated. This said, there are some differences in our simulations when no retransmission is allocated but they are small and there is not a clear tendency due to the simulation resolution[10], so this behavior should be thoroughly studied in future. Conversely, when minor cycles can only allocate one retransmission, bursty losses are harmful because the probability that losses are together in the same minor cycles is higher. This effect disappears when the number of retransmissions is increased because minor cycles can tolerate more packet losses.

Table 5.12: Percentage of minor cycles satisfied in the simulation of Test 5.1 when PDR = 97% and four packet loss models are employed

ABL	0 ReTx	1 ReTx	2 ReTx	3 ReTx
Memoryless	69.49	94.40	99.25	99.92
1.11	69.18	94.42	99.22	99.91
2	69.53	91.53	97.93	99.49
4	69.44	84.22	91.87	95.94

Table 5.13: Data Delivery Ratio (%) obtained in the simulation of Test 5.1 when PDR = 97% and four packet loss models are employed

ABL	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
Memoryless	1	99.86	99.99	100.00	100.00
	2	69.49	94.40	99.25	99.92
1.11	1	99.86	99.99	100.00	100.00
	2	69.18	94.42	99.22	99.91
2	1	99.36	99.86	99.97	100.00
	2	69.53	91.53	97.93	99.49
4	1	96.41	98.32	99.24	99.62
	2	69.44	84.22	91.87	95.94

5.5.4 Comparison with RT-WMP in a Chain Network

In this section, we compare WICKPro with a version of RT-WMP enhanced for transmitting several packets in a single token holding [Tardioli 14]. In this paper, RT-WMP was simulated in an error-free WMN with chain topology that supported a bidirectional communication between the ends of the chain. Simulation was carried out for various values of network nodes and packet sizes, in a way that the network utilization was always 100%. The Maximum Transmission Unit (MTU) was 1,388

[10]These simulation results also supports the results of the field experiment presented in Section 5.5.2, where no retransmission is allocated and the measured results matches the theoretical results calculated with a memoryless packet loss model.

bytes, so that packets with greater sizes than the MTU required actually more than one packet transmission. It should be noted that if packet size is lower than the MTU, this version of RT-WMP [Tardioli 14] obtains the same performance than the previous standard version shown in [Tardioli 15]. Regarding the physical level, this simulation considered IEEE 802.11a at 6 Mbps.

For a fair comparison, we calculated the throughput achieved by FRT-WICKPro in the same conditions. As the scenario is error-free and the network utilization is 100%, we can use theoretical formulas to obtain the throughput in each situation, since the off-line cyclic scheduling calculated by the scheduler will be repeated every minor cycle. The aggregate throughput achieved by FRT-WICKPro is shown in Table 5.14, where the number of nodes (routers in FRT-WICKPro) ranges from 2 to 32, and the packet size from 256 bytes to 256 kilobytes. In this case, the aggregate throughput is obtained by summing the throughput of the two flows supported in the network[11].

Table 5.14: Aggregate throughput (Kbps) obtained by WICKPro in a WMN with chain topology as a function of the number of nodes and the packet size

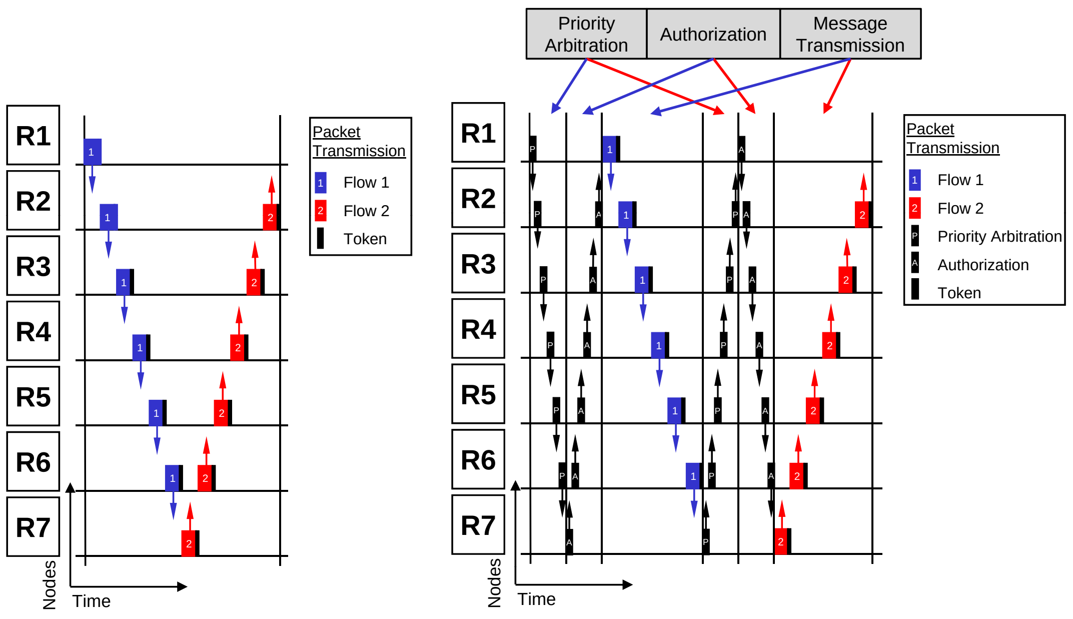
Nodes	256B	512B	1KB	2KB	4KB	8KB	32KB	256KB
2	2293.39	3311.24	4273.34	4662.49	5060.69	5190.56	5291.56	5331.10
3	1146.70	1655.62	2136.67	2331.25	2530.35	2595.28	2645.78	2665.55
4	764.46	1103.75	1424.45	1554.16	1686.90	1730.19	1763.85	1777.03
5	573.35	827.81	1068.34	1165.62	1265.17	1297.64	1322.89	1332.77
6	458.68	662.25	854.67	932.50	1012.14	1038.11	1058.31	1066.22
7	382.23	551.87	712.22	777.08	843.45	865.09	881.93	888.52
8	327.63	473.03	610.48	666.07	722.96	741.51	755.94	761.59
10	254.82	367.92	474.82	518.05	562.30	576.73	587.95	592.34
20	120.70	174.28	224.91	245.39	266.35	273.19	278.50	280.58
32	73.98	106.81	137.85	150.40	163.25	167.44	170.70	171.97

The comparison between Table 5.14 and the results attained by RT-WMP in [Tardioli 14] is exhibited in Table 5.15. In this table, the percentage of throughput improvement obtained by FRT-WICKPro is shown. Depending on the case, this improvement grows up to 278%, while the average improvement is 37.88%. This throughput improvement is achieved because FRT-WICKPro needs less token transfers to support these data flows, as illustrated in Fig. 5.10. In this figure, we can observe that RT-WMP, being decentralized and without global synchronization, must carry out two phases before the message transmission: the priority arbitration to know the highest priority packet and the authorization transmission for passing the token to the node with the highest priority packet. Conversely, FRT-WICKPro schedules its transmissions in advance thanks to its cyclic scheduler.

[11]For instance, the aggregate throughput in a seven-router network that supports two end-to-end flows with a packet size of 1,024 bytes is $2 * 1024 * 8/(12 * 1.917) = 712.22$ Kbps. As shown in Fig. 5.10a, in this case there is one minor cycle with 12 piggyback packets whose transmission time is 1.917 ms, based on Eq. (5.12) and with a transmission margin of 0.3 ms. The log margin of 0.2 ms is not considered for fair comparison. It should be highlighted that Test 5.1 without any retransmission allocated is a subset of these experiments, where the number of nodes is seven and the packet size is 1,000 bytes. In this situation, the throughput will be $2*1000*8/(12*2.09) = 637.96$ Kbps, where the piggyback packet transmission time is 2.09 ms because log information is sent along with the piggyback packet.

Table 5.15: Percentage of throughput improvement carried by WICKPro when compared with RT-WMP in a chain network

Nodes	256B	512B	1KB	2KB	4KB	8KB	32KB	256KB
2	47.16	44.08	18.61	33.37	24.61	22.81	21.39	20.72
3	37.39	30.94	14.95	30.77	23.37	22.17	21.17	20.90
4	38.21	25.62	14.36	30.47	23.27	22.01	20.90	21.05
5	40.02	26.77	14.66	31.17	22.38	21.72	20.88	21.29
6	42.59	28.50	15.56	32.44	20.60	22.50	21.00	21.46
7	45.95	31.12	17.04	33.27	23.55	22.36	21.29	21.74
8	49.66	33.62	18.71	35.02	24.29	23.23	21.32	21.95
10	59.15	40.25	22.81	35.97	27.71	22.87	23.19	22.66
20	139.83	98.27	60.15	48.55	30.36	27.15	29.10	28.90
32	278.42	197.37	123.09	76.12	46.44	32.19	39.85	41.87



(a) WICKPro scheduling **(b)** RT-WMP scheduling (worst-case scenario)

Figure 5.10: Example of scheduling carried out by WICKPro and RT-WMP in a chain network with seven routers

It should also be noted that RT-WMP takes advantage of this token rotation to share the RSSI between all network links, which is employed to manage mobility. However, FRT-WICKPro could handle mobility without exchanging so much information, given that it considers a network with fixed routers, unlike RT-WMP that takes into account that all nodes are mobile. Indeed, mobility is carried out in SRT-WICKPro with low overhead, as shown in Chapter 7. This scheme could also be applied to FRT-WICKPro but with higher complexity due to the packet synchronization and the strict minor cycle fulfillment. Hence, the comparison presented in this section is realistic.

Finally, as Table 5.14 also gives an indication of the FRT-WICKPro scalability, we show a graphical representation of these results in Fig. 5.11 when packet size equals 256 B, 1 KB and 256 KB. In this image we can notice that FRT-WICKPro is not scalable, as expected, due to the employed token-passing scheme without spatial reuse. However, we must recall that scalability was not a design specification, since WICKPro was developed to work in small-scale WMNs.

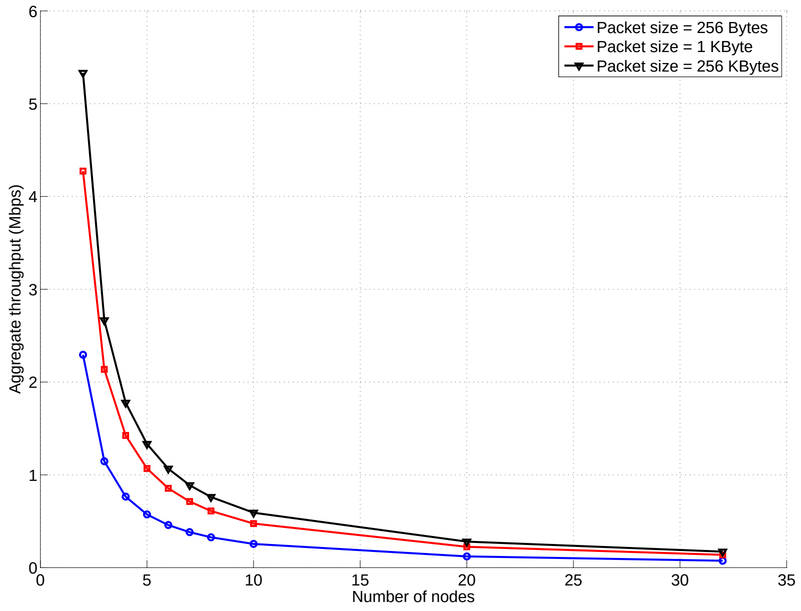


Figure 5.11: Aggregate throughput (Mbps) as a function of the number of nodes using FRT-WICKPro on top of IEEE 802.11a at 6 Mbps when packet size = 256 B, 1 KB and 256 KB

5.6 Conclusions

Packet retransmission is necessary in error-prone networks to support FRT traffic. However, these retransmissions can jeopardize the deadlines of the supported traffic and a fault-tolerant perspective becomes therefore essential. We have shown that FRT-WICKPro achieves FRT traffic support due to its off-line cyclic packet scheduler, which reserves time for packet transmissions and retransmissions, and its on-line packet scheduler that drop packets before congestion appears. Another important feature is that resource reservation is possible thanks to the employed token-passing scheme. We have presented an analytical framework for FRT traffic support and have evaluated FRT-WICKPro in laboratory and field experiments.

The time margin for retransmissions is calculated based on the set of supported data flows, the mean PDR between one-hop neighbors and a memoryless packet loss model. A field experiment has shown that this model may be used in certain scenarios as a good approximation. A MATLAB simulation was conducted to evaluate the loss of accuracy when using a memoryless packet loss model in a bursty channel. We showed that the inaccuracy increases when the ABL is higher and the number of retransmissions allocated is lower, except when there is no retransmission allocated, in which case both results are similar.

We also carried out a comparison with RT-WMP in an error-free scenario, where the throughput gain was 37.88% on average thanks to the lower overhead provided by FRT-WICKPro.

6 WICKPro with Soft Real-Time Traffic Support

There are many industrial applications that require SRT traffic support, for example, multimedia-based robotic applications such as surveillance and tele-control, possibly complemented with a low-level local safety subsystem. For this reason, we also developed a version of WICKPro to support SRT traffic. SRT-WICKPro employs the off-line cyclic packet scheduler presented in Chapter 4. Therefore, it does not allocate explicitly time for retransmissions and calculates the scheduling of a set of flows in an error-free scenario assuming that there will be a small degradation in a real environment due to packet losses. As SRT-WICKPro neglects using any synchronization policy, it is totally asynchronous and lets minor cycles be overrun. This, together with the fact that data flows will not be synchronized due to this lack of synchronization, may cause release jitter. However, this issue is not so critical with SRT traffic because the maximum delay tolerated by the application is usually increased beyond the deadline value. Thus, due to the implemented strategies, SRT-WICKPro neglects computing the PDR and using the on-line packet scheduler introduced in Chapter 5, unlike FRT-WICKPro.

This chapter presents SRT-WICKPro as follows. Section 6.2 details the real-time model assumed by SRT-WICKPro, Section 6.3 presents its off-line packet scheduler, Section 6.4 gives a preliminary formal analysis and Section 6.5 evaluates the performance of SRT-WICKPro, where a mobile robot tele-operation is carried out in simulation and laboratory experiments.

6.1 Relevant Publications

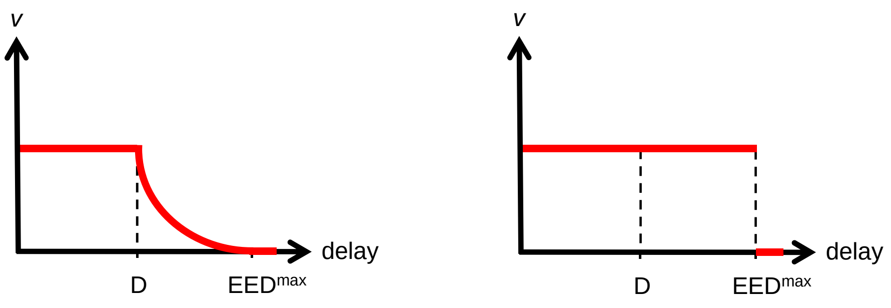
The work presented in this chapter was mainly published in the following paper:

- [Aisa 16a] J. Aisa, H. Fotouhi, J.L. Villarroel and L. Almeida. *Soft Real-time Traffic Communication in Loaded Wireless Mesh Networks.* In IEEE World Conference on Factory Communication Systems (WFCS), pages 1-8, May 2016.

6.2 SRT Traffic Model in WICKPro

SRT-WICKPro considers the traffic model with soft criticality explained in Section 2.1.1. Fig. 6.1 shows a generic utility function and the utility function employed by

WICKPro. As we can observe, EED^{max} is the maximum End-to-End Delay that makes a packet being useful for the application. This value is usually higher than D in SRT applications, since these still give some value to packets whose delays are higher than their deadlines. As the utility function weights every delivered packet depending on its delay, the objective should be to maximize the cumulative value of every multi-hop flow, given by the sum of the value of all the delivered packets.



(a) Generic utility function **(b)** Utility function of WICKPro

Figure 6.1: Utility functions for soft real-time traffic

The utility function depicted in Fig. 6.1a could be used by a voice application that transmits packets across a communication network. In this kind of applications, it is hard to follow a conversation when the time between speaking in one side and listening on the other side differs more than 300 ms [Peterson 11]. Therefore, packets with delays higher than 300 ms can be dropped ($EED^{max} = 300\ ms$). Moreover, the lower the end-to-end delay of voice packets, the more likely the application users would perceive an improvement, so that these packets should be more valuable.

A similar behavior is found in control applications, e.g., robot tele-operation, where there is usually an operator who sends control commands to a mobile robot that transmits feedback information to the operator, such as laser scan data or video information. If the delay in the feedback loop is higher than a specific value, the system can become unstable and thus these packets would be useless. SRT-WICKPro considers that improvements in the system performance for any delay lower than EED^{max} may be neglected, as shown in Fig. 6.1b. With this assumption, we can maximize the cumulative value of every multi-hop flow by maximizing its percentage of delivered packets within the EED^{max} deadline, i.e., its DDR. This strategy was also used during a robot tele-operation in [Lindhorst 13] by using the IsoMAC protocol.

6.3 Off-line Packet Scheduler for SRT Support

FRT-WICKPro vs SRT-WICKPro First of all, we compare the off-line packet scheduler of SRT-WICKPro with the one presented in Chapter 5. Let us illustrate this comparison in the Scenario 6.1, whose network is given in Fig. 6.2 and whose data flows are described in Table 6.1. Moreover, the token transmission time was 0.75 ms and the time-out was 4.18 ms. As both flows had the same communication requirements, except that the source and destination were different, the major cycle had only one minor cycle that was equal to the period of both multi-hop flows. Although this table gives four values for P and six for EED^{max}, we now focus on the case $P = 32.85ms$ and $EED^{max} = P$, which provided free time for exactly one piggy-back packet retransmission in every minor cycle[1].

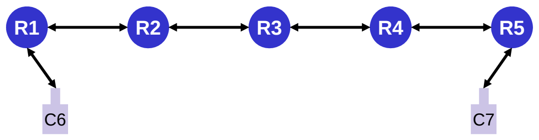


Figure 6.2: WMN with five routers and two clients in Scenario 6.1

Table 6.1: Communication requirements in Scenario 6.1

Flow	Size (bytes)	C (ms)	P (ms)	EED^{max} (ms)	src	dst
1	1000	2.09	26.58, 32.85, 39.12 and 45.39	P, 1.1P, 1.3P, 1.5P, 2P and inf	C6	C7
2	1000	2.09	26.58, 32.85, 39.12 and 45.39	P, 1.1P, 1.3P, 1.5P, 2P and inf	C7	C6

Fig. 6.3 and 6.4 show examples of transmissions carried out in Scenario 6.1 by FRT-WICKPro and SRT-WICKPro, respectively. We assume that in the first minor cycle of Fig. 6.3 (from 0 to 32.85 ms), there is no packet loss, so that all the scheduled transmissions are carried out, after which the token master keeps the token until the next minor cycle begins. In the second minor cycle (from 32.85 to 2x32.85 ms), there are two packet losses which causes that all the scheduled transmissions cannot be performed before the second minor cycle finishes. Specifically, when the packet of flow 1 transmitted by R4 is lost, this is retransmitted after a time-out. However, R4 transmits a packet of flow 2 that is not received properly, but this is not retransmitted because, when the second minor cycle finishes, all nodes invoke the on-line packet scheduler for dropping all packets that have not been delivered. In this way, the second minor cycle is not overrun and the token master starts the third minor cycle at the appropriate time.

SRT-WICKPro calculates the same cyclic scheduling than FRT-WICKPro, but the token master keeps the token circulating continuously as in common token-passing protocols (asynchronous token-passing). For this reason, unlike in FRT-WICKPro, the actual minor cycle duration differs from the theoretical minor cycle value. Likewise, due to the lack of synchronization, the generation of the different data flows do not occur at the same time, as shown in Fig. 6.4. In this example, we also assume that there is no packet loss from 0 to 32.85 ms, thus all the scheduled transmissions in the first minor cycle are properly carried out. After all these transmissions, the first minor cycle finishes and the token master starts the second minor cycle, however, only the token is transmitted since any data packet is ready to be sent. Therefore, the first and the second minor cycle are shorter than the theoretical minor cycle value. In the third minor cycle, the are two packet losses so that SRT-WICKPro carries out two retransmissions and this minor cycle becomes larger than the theoretic minor cycle. The downside of this scheme is that can produce a domino effect that can jeopardize the deadlines of the supported flows. In general,

[1]The cyclic scheduling calculated by WICKPro is shown in the first minor cycle of Fig. 6.4. Therefore, the shortest theoretic minor cycle duration was the time to transmit six packets of flow 1, six packets of flow 2 and two token packets. In this case, minor cycle duration = $6{*}C_1 + 6{*}C_2 + 2{*}C_{Token} = 6{*}2.09 + 6{*}2.09 + 2{*}0.75 = 26.58$ ms. If there was free time for one piggy-back packet retransmission, minor cycle duration = $6{*}C_1 + 6{*}C_2 + 2{*}C_{Token}$ + Time-out + $C_1 = 6{*}2.09 + 6{*}2.09 + 2{*}0.75 + 4.18 + 2.09 = 32.85$ ms.

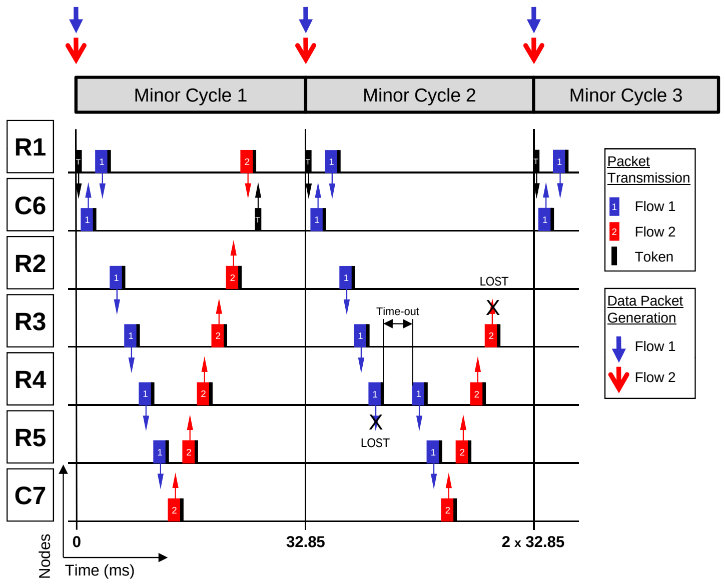


Figure 6.3: An example of FRT-WICKPro in Scenario 6.1 with two packet losses

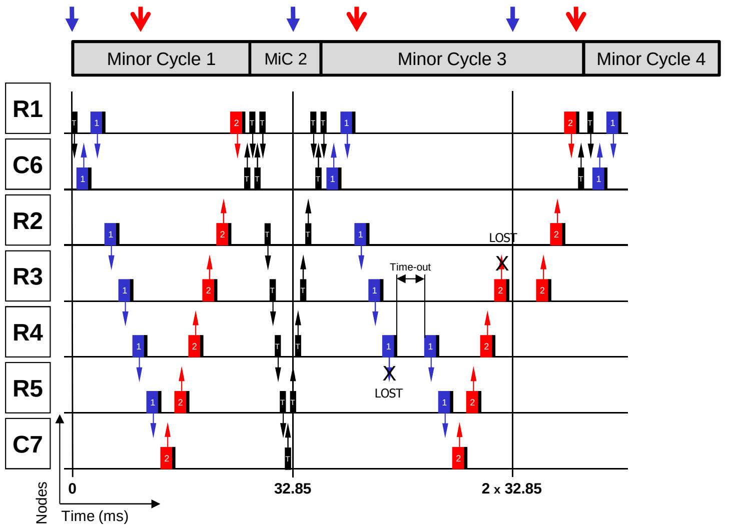


Figure 6.4: An example of SRT-WICKPro in Scenario 6.1 with two packet losses

SRT-WICKPro transmits data packets until their delay is higher than EED^{max}. At that point, the data packet is dropped and only the token packet is transmitted. As commented in Chapter 4, if the number of retransmissions is high, all nodes execute the topology discovery algorithm to restart the network. This decision is driven by a maximum number of retransmissions in the node that holds the token, and by a maximum time without receiving the token packet in the other nodes.

In summary, in FRT-WICKPro the token is synchronously released by the token master with the periodic minor cycles and thus every minor cycle is independent of each other. Conversely, SRT-WICKPro carries out asynchronous token-passing, thereby decoupling the theoretic and the actual minor cycles.

Details on SRT-WICKPro Data flows will suffer release jitter because the supported traffic is periodic and SRT-WICKPro is an asynchronous token-passing protocol. The lack of a synchronization policy causes two issues: (i) data flows are not synchronized with the minor cycle beginning thereby causing release jitter, and (ii) minor cycles can be overrun and thus release jitter would be produced even if data flows were synchronized with minor cycles. Release jitter will be therefore random and this may lead to poor performance in terms of DDR, so that SRT-WICKPro provides response times that can grow beyond D, specifically until EED^{max}. We can state that D is the deadline for the packet scheduler whereas EED^{max} is the deadline for the application. Because of this limitation, this new WICKPro version is suited for SRT traffic, only. Conversely, aperiodic traffic could be efficiently supported due to the asynchronous feature of the protocol.

Regarding network congestion, it is possible that source nodes have more than one packet to transmit of the same data flow, given that EED^{max} is higher than P. However, this is not the case for relay nodes due to the token-passing scheme implemented by WICKPro.

6.4 Preliminary Formal Analysis

Next we provide a preliminary formal analysis of SRT-WICKPro latencies, initially in the absence of errors and then accounting for these phenomena, in which we do not take into account external interferences.

6.4.1 Schedulability Analysis in Error-free Scenario

A set of data flows will be accepted if a feasible cyclic scheduling can be found in an error-free scenario, where a necessary condition is that the network utilization is no larger than 100%. It should be highlighted that the scheduler considers $D{=}P$.

6.4.2 DDR Test in Error-free Scenario

Although the scheduler finds a feasible schedule for the current set of data flows, there is no guarantee that the DDR will reach 100% even in error-free networks due to the asynchronous nature of the protocol. For this reason, we define the following test. Given a set of data flows, the DDR of the flow i will be 100% in an error-free scenario if the following condition is satisfied:

$$R_i = J_i + B_i + \sum_{j=1}^{HC_i} C_{ij} \leq EED_i^{max} \tag{6.1}$$

R_i is the worst-case response time among all packets of flow i. This, in turn, is the difference between the packet arrival time at destination node and the packet generation time at source node. R_i can be expressed as the sum of the release jitter of flow i (J_i), the blocking time (B_i), and the end-to-end transmission time ($\sum_{j=1}^{HC_i} C_{ij}$). J_i is the deviation from exact periodic release caused by the minor cycle duration variations and the lack of synchronization of the data flows. B_i is the worst-case delay in relay nodes, i.e., the time that the wireless medium is used by other flows different than flow i during the end-to-end transmission, which may include flows scheduled in the same minor cycle. Finally, HC_i is the hop count from *src* to *dst* of flow i, and C_{ij} is the transmission time of flow i in the hop j, which is equal to C_i because all nodes are equal.

6.4.3 Maximum DDR

When packet losses are considered, it is of interest to know the maximum DDR that may be attained whatever the EED^{max} is. Thus, we define DDR^{max} as the maximum DDR that may be achieved in an error-prone network modeled by a memoryless packet loss model. This upper bound takes into account that no packet is dropped, i.e., $EED^{max} = infinite$. DDR^{max} can be expressed as follows:

$$DDR^{max} = \frac{M}{\sum_{i=1}^{N_{MH}} ETT_i},\ DDR^{max} \in [0, 100]\,\% \tag{6.2}$$

M is the major cycle duration and ETT_i [Akyildiz 09] is the Expected Transmission Time of flow i. In turn, we define ETT_i as shown in (6.3):

$$ETT_i = \sum_{j=1}^{HC_i} ETT_{i,j} = \sum_{j=1}^{HC_i} (C_i + (ETX_{i,j} - 1)(\mathit{timeout} + C_i)) \tag{6.3}$$

$ETX_{i,j}$ [Akyildiz 09] is the Expected Transmission Count of flow i in the link formed by the relay number j (node x) and its destination node (node y). ETT_i is computed taking into account that the first transmission lasts C_i and that the subsequent transmissions consume a time equal to *timeout* plus C_i. As WICKPro uses implicit ACK, $ETX_{i,j} = 1/PDR_{x,y}{}^2$, where $PDR_{x,y}$ is the PDR of the link composed by a transmitter node x and a receiver node y.

6.5 Evaluation

We assessed SRT-WICKPro in simulation and laboratory experiments where we implemented Scenarios 6.1 and 6.2 (Sections 6.5.1 and 6.5.2, respectively). In the laboratory tests, we set the transmission frequency to 5.3 GHz (802.11a) at 6 Mbps using CSMA/CA without RTS/CTS. All transmissions were broadcast, so that the hardware-level ACKs of 802.11 were not employed. In the simulation evaluations, we used this configuration to calculate the transmission time. All simulation experiments were executed in 100,000 theoretic major cycles while each laboratory experiment was conducted in 1,000 seconds.

In Scenarios 6.1 and 6.2, P and EED^{max} were changed to analyze their effect in the DDR. P modified the network load while EED^{max} varied the maximum deadline accepted by the application[3]. Moreover, a simple congestion avoidance algorithm was implemented at source nodes to examine its importance.

[2]Since the packet loss model is memoryless, each attempt to transmit a packet can be considered a Bernoulli trial.

[3]Although application requirements are fixed, we changed the EED^{max} to analyze the behavior of SRT-WICKPro. Moreover, this approach is also useful for delay-adaptive applications such as voice applications because they carry out on-line adjustments of the EED^{max} depending on the network quality: if the network provides a good quality, they reduce EED^{max} to provide better user experience, whereas if the network quality gets worse, EED^{max} is augmented to increase the DDR, at the expense of higher delays. In short, they are continually maximizing the cumulative value of every flow based on a utility function similar to that of Fig. 6.1a.

6.5.1 Scenario 6.1 - 7-node WMN with Chain Topology

We performed extensive simulations in MATLAB to evaluate SRT-WICKPro in Scenario 6.1. This simulator is the same as the used in FRT-WICKPro in Chapter 5, where packet losses can occur but token duplication is not taken into account. Furthermore, all data flows are synchronized, although this could be easily changed, if needed.

Scenario 6.1 was described in Section 6.3 through the network depicted in Fig. 6.2 and the communication requirements given by Table 6.1. Moreover, the token transmission time was 0.75 ms and the time-out was 4.18 ms. Next we will present more details of these communication requirements. The transmission time was computed by using Eq. (5.12), where C corresponds to a piggy-back packet transmission that includes log information and transmission margin as explained in Section 5.5.1.2. As already mentioned, both flows had the same period and therefore the major cycle only had one minor cycle whose value was the period of the multi-hop flows. In this experiment, we carried out four tests where the period P and the theoretic minor cycle duration were modified. In the first test, the theoretic minor cycle duration was the shortest possible, i.e., 26.58 ms, which was the time to transmit six packets of flow 1, six packets of flow 2 and two token packets. In the other three tests, the theoretic minor cycle duration was the minimum minor cycle duration plus the time to retransmit one, two and three piggy-back packets, respectively, i.e., 32.85 (26.58 + 4.18 + 2.09), 39.12 (26.58 + 2*4.18 + 2*2.09) and 45.39 ms (26.58 + 3*4.18 + 3*2.09). Thus, the network utilization of the four tests was 100, 80.91, 67.94 and 58.56%, respectively. Moreover, for every one of the four minor cycle durations, we carried out six experiments where EED^{max} was set to be P multiplied by 1, 1.1, 1.3, 1.5, 2 and infinite. In turn, for every one of these 24 experiments, the simulations were conducted using a memoryless packet loss model with a PDR equal to 100% (error-free scenario), 99%, 97% and 95%. Without loss of generality, this PDR was assumed to be the same for all links and packet sizes. These 96 experiments were simulated with and without a Congestion Control (CC) algorithm, thus the experiment number was finally 192. This CC algorithm only allows source nodes to have one packet to be sent in their transmission buffer. This means that source nodes drop packets of the flows they generate if their delay is higher than P. Conversely, if source nodes neglect using any CC algorithm, these can have all the data packets whose delay is not higher than EED^{max} in their transmission buffer.

Next we present the obtained results when PDR = 100% and PDR = 97%. The results when PDR = 99% and PDR = 95% are analogous to those of PDR = 97%. It should be noted that nodes have a buffer that can allocate up to 20 packets, which is only relevant in the case of not using deadline (EED^{max} = infinite). Moreover, when $EED^{max} = P$, the attained results are the same whether or not the CC algorithm is employed, except for the logical variability of simulations.

6.5.1.1 PDR = 100%

DDR Table 6.2 shows that the DDR = 100% when PDR = 100%, both with and without CC algorithm. It should be highlighted that these results are obtained because all data flows are synchronized in our simulations, which will not be the case in real experiments.

Table 6.2: DDR (%) in Scenario 6.1 when PDR = 100% with and without CC

EED^{max}	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	100	100	100	100
	2	100	100	100	100
1.1P	1	100	100	100	100
	2	100	100	100	100
1.3P	1	100	100	100	100
	2	100	100	100	100
1.5P	1	100	100	100	100
	2	100	100	100	100
2P	1	100	100	100	100
	2	100	100	100	100
Infinite	1	100	100	100	100
	2	100	100	100	100

Delay The mean and the standard deviation of the end-to-end delay with and without CC is presented in Table 6.3. If no retransmission is allocated, the mean delay is fixed (the variance is 0), since all flows are synchronized. In a real experiment, the delay value will be likely different due to the lack of synchronization in the data flow generation and the minor cycles. In the case of one, two and three retransmissions allocated, the results are quite similar because the delay depends on the release jitter caused by the irregular token arrivals, which depends slightly on the amount of free time in the theoretic minor cycle, as long as this time is higher than the time to pass the token from C6 to C7, or vice versa ($6 * 0.75 = 4.5$ ms).

Table 6.3: Mean ± std dev delay (ms) in Scenario 6.1 when PDR = 100% with and without CC

EED^{max}	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	13.29	21.22 ± 6.76	21.24 ± 6.76	21.22 ± 6.76
	2	25.83	22.37 ± 6.76	22.38 ± 6.76	22.37 ± 6.76
1.1P	1	13.29	21.22 ± 6.76	21.24 ± 6.76	21.22 ± 6.76
	2	25.83	22.37 ± 6.76	22.38 ± 6.76	22.37 ± 6.76
1.3P	1	13.29	21.22 ± 6.76	21.24 ± 6.76	21.22 ± 6.76
	2	25.83	22.37 ± 6.76	22.38 ± 6.76	22.37 ± 6.76
1.5P	1	13.29	21.22 ± 6.76	21.24 ± 6.76	21.22 ± 6.76
	2	25.83	22.37 ± 6.76	22.38 ± 6.76	22.37 ± 6.76
2P	1	13.29	21.22 ± 6.76	21.24 ± 6.76	21.22 ± 6.76
	2	25.83	22.37 ± 6.76	22.38 ± 6.76	22.37 ± 6.76
Infinite	1	13.29	21.22 ± 6.76	21.24 ± 6.76	21.22 ± 6.76
	2	25.83	22.37 ± 6.76	22.38 ± 6.76	22.37 ± 6.76

DDR test in error-free scenario The DDR test in Eq. (6.1) shows that DDR will always be 100% in an error-free scenario, which matches the simulation results shown previously in Table 6.2. When no retransmission is allocated, as there is no packet loss, the theoretic cyclic scheduling will be strictly carried out in every minor cycle. Therefore, DDR will always be 100%[4]. When one retransmission is allocated, the response time R_i is 31.08 ms for both flows, which is lower than 32.85 ms, i.e., the EED^{max} in the most restrictive case ($EED^{max} = P$). Thus, Eq. (6.1) yields a DDR of 100%. This calculation is explained in Fig. 6.5 for the case of flow 1, where we can see the worst-case scenario for a packet of flow 1 in a minor cycle n: the release jitter is J_1 (18.54 ms), and the end-to-end transmission time is $6 \times C_1$ (12.54 ms). The blocking time B_1 is zero since there is no competing traffic.

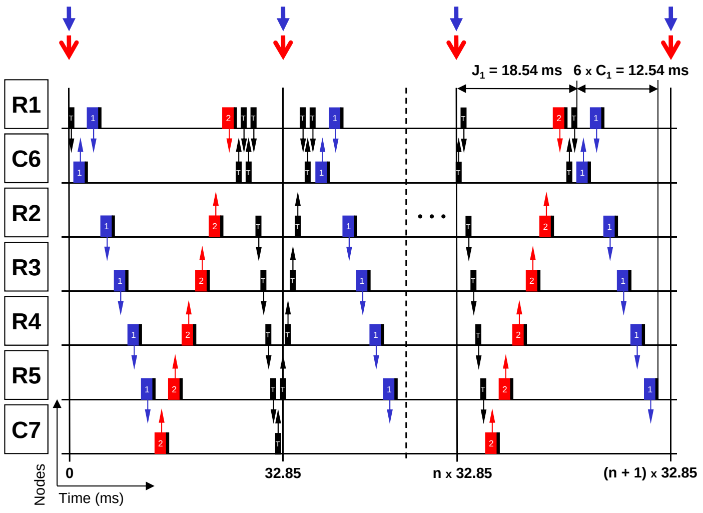


Figure 6.5: DDR test in Scenario 6.1 for flow 1 in an error-free network where one retransmission is possible every theoretic minor cycle

6.5.1.2 PDR = 97%

Tables 6.4 and 6.5 show the DDR achieved when PDR = 97% with and without CC mechanism, whereas the mean and the standard deviation delay is presented in Tables 6.6 and 6.7.

Congestion control On the one hand, when the no retransmission is allocated (network utilization = 100%), the strategy with CC achieves higher DDR (Table 6.4) than the strategy without CC (Table 6.5). Regarding the delay, when no retransmission is allocated and CC is not employed (Table 6.7), the mean and the standard deviation delay are increased considerably when EED^{max} is increased, particularly when EED^{max} = infinite. When using CC, the mean and the standard deviation delay are also increased, but slowly, as depicted in Table 6.6. On the other hand, if there is one or more retransmissions allocated, both cases are similar in terms of DDR and delay. Thus, as usual, CC is useful when the network utilization is high.

$EED^{max} = P$ vs $EED^{max} = 2P$ In Tables 6.4 and 6.5, it is shown that the DDR is increased when EED^{max} is augmented. When no retransmission is allocated and CC is employed, the overall DDR is increased by 42%[5] when EED^{max} is changed from P to $2P$ (Table 6.4). Moreover, unlike the case with $EED^{max} = 2P$

[4]This result is obtained in simulation because the data flows are synchronized. In a real experiment, the DDR will not be actually 100%, but a deeper analysis is left for future work.

[5]It is increased from 64.10% ((95.32 + 32.89) / 2) to 91.04% ((91.04 + 91.03) / 2).

Table 6.4: DDR (%) in Scenario 6.1 with CC when PDR = 97%

EED$^{\text{max}}$	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	95.32	84.85	95.80	99.23
	2	32.89	80.17	95.24	99.14
1.1P	1	67.30	92.76	98.66	99.85
	2	67.69	91.24	98.59	99.84
1.3P	1	81.51	98.35	99.88	99.99
	2	79.07	97.97	99.86	99.99
1.5P	1	88.24	99.40	99.97	100
	2	88.24	99.32	99.96	100
2P	1	91.04	99.55	99.98	100
	2	91.03	99.51	99.97	100
Infinite	1	90.96	99.55	99.97	100
	2	90.96	99.45	99.97	100

Table 6.5: DDR (%) in Scenario 6.1 without CC when PDR = 97%

EED$^{\text{max}}$	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	95.30	84.85	95.79	99.20
	2	33.19	80.17	95.28	99.23
1.1P	1	67.05	92.68	98.68	99.85
	2	67.16	91.23	98.54	99.83
1.3P	1	67.42	98.19	99.86	99.99
	2	66.42	97.77	99.88	99.99
1.5P	1	67.70	99.56	99.99	100
	2	66.91	99.56	99.99	100
2P	1	67.45	99.99	100	100
	2	66.74	99.99	100	100
Infinite	1	91.06	100	100	100
	2	91.06	100	100	100

that satisfies network fairness, the case with $EED^{max} = P$ fails in fairness[6]. The main reason is that the scheduling is not symmetric (as shown in the first minor cycle in Fig. 6.4) and the case $EED^{max} = 2P$ adds some degree of freedom which improves fairness, given that both flows have the same features (C, P, and v). Actually, the case EED^{max} = *1.1P* already achieves fairness. Regarding the case of three retransmissions, the cases $EED^{max} = P$ and $EED^{max} = 2P$ obtain similar results when EED^{max} is changed from P to $2P$ (99.20% vs 100%).

We also have to keep in mind that the higher the EED^{max}, the higher the average delay of the delivered packets, although in our scenario this is only relevant when the network utilization is 100%. Actually, as the delay of the delivered packets is always lower than the corresponding EED^{max}, the mean delay is not an important constraint given the utility function employed by SRT-WICKPro. However, delay analysis is useful to illustrate the behavior of the protocol.

[6]When $EED^{max} = 2P$, in Tables 6.4 and 6.5 both flows obtains similar results, while when $EED^{max} = P$, in Table 6.5 the DDR of flow 1 is 95.30% and the DDR of flow 2 is 33.19%. Similar values are obtained in Table 6.4.

Table 6.6: Mean ± std dev delay (ms) in Scenario 6.1 with CC when PDR = 97%

EED^{max}	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	16.19 ± 3.22	20.48 ± 6.13	22.89 ± 7.55	23.77 ± 8.19
	2	25.13 ± 3.02	23.10 ± 6.60	24.52 ± 7.65	25.02 ± 8.22
1.1P	1	21.03 ± 5.30	22.63 ± 7.17	23.71 ± 8.06	23.92 ± 8.37
	2	19.97 ± 5.80	23.75 ± 7.36	24.85 ± 8.11	25.10 ± 8.37
1.3P	1	24.27 ± 5.80	24.09 ± 8.10	23.98 ± 8.41	24.02 ± 8.45
	2	24.21 ± 6.85	25.27 ± 8.19	25.22 ± 8.39	25.17 ± 8.47
1.5P	1	26.09 ± 7.67	24.38 ± 8.42	24.06 ± 8.44	24.00 ± 8.45
	2	26.37 ± 7.70	25.72 ± 8.52	25.26 ± 8.50	25.19 ± 8.44
2P	1	27.02 ± 8.15	24.41 ± 8.45	24.05 ± 8.45	24.05 ± 8.45
	2	26.89 ± 8.13	25.73 ± 8.58	25.24 ± 8.48	25.17 ± 8.47
Infinite	1	27.02 ± 8.14	24.50 ± 8.50	24.09 ± 8.47	24.01 ± 8.48
	2	26.99 ± 8.16	25.72 ± 8.60	25.25 ± 8.47	25.18 ± 8.47

Table 6.7: Mean ± std dev delay (ms) in Scenario 6.1 without CC when PDR = 97%

EED^{max}	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	16.18 ± 3.22	20.48 ± 6.13	22.90 ± 7.56	23.79 ± 8.19
	2	25.14 ± 2.99	23.10 ± 6.60	24.50 ± 7.65	24.99 ± 8.22
1.1P	1	21.10 ± 5.26	22.62 ± 7.17	23.73 ± 8.05	23.95 ± 8.38
	2	19.94 ± 5.78	23.77 ± 7.36	24.86 ± 8.12	25.17 ± 8.39
1.3P	1	26.31 ± 5.21	24.20 ± 8.11	24.01 ± 8.38	23.96 ± 8.42
	2	25.40 ± 5.80	25.33 ± 8.20	25.17 ± 8.43	25.23 ± 8.47
1.5P	1	31.67 ± 5.25	24.83 ± 8.65	24.09 ± 8.48	24.07 ± 8.47
	2	30.67 ± 5.81	25.90 ± 8.65	25.25 ± 8.50	25.15 ± 8.47
2P	1	45.01 ± 5.25	24.97 ± 8.91	24.11 ± 8.46	24.02 ± 8.45
	2	43.92 ± 5.79	26.10 ± 8.91	25.25 ± 8.53	25.20 ± 8.48
Infinite	1	583.51 ± 24.73	25.20 ± 9.07	24.10 ± 8.47	24.00 ± 8.42
	2	583.65 ± 24.40	26.09 ± 8.96	25.24 ± 8.52	25.21 ± 8.50

$EED^{max} = 2P$ vs $EED^{max} = infinite$ The DDR attained in these cases is very similar. There is only a sizeable difference in Table 6.5 when no retransmission is allocated, where the DDR is increased from 67.1% ((67.45 + 66.74) / 2) to 91.06%. This comparison highlights the marginal increase in the DDR provided by EED^{max} values higher than $2P$.

DDR^{max} We now calculate the DDR^{max} by employing Eq. (6.2) and compare this theoretic value with the simulation value when CC is not enabled. Using Eq. (6.2), when no retransmission is allocated, the DDR^{max} calculated is 90.99%. In this case, the major cycle M is 26.58 ms and $\sum_{i=1}^{N_{MH}} ETT_i$ is 29.21 ms. This value is very similar to the DDR given by Table 6.5 when EED^{max} is infinite, i.e., 91.06%. When one retransmission is allocated, Eq. (6.2) yields a DDR^{max} of 100%, as M is 32.85 ms and $\sum_{i=1}^{N_{MH}} ETT_i$ is 29.21 ms. When two and three retransmissions are allocated, Eq. (6.2) also yields a DDR^{max} of 100%. The same results are shown in Table 6.5 when EED^{max} is infinite.

The presented results correspond to the case of PDR = 97%. If PDR = 95%, the DDR^{max} obtained by Eq. (6.2) is 85.58% while the simulation gives 85.69%, which are very similar. If PDR = 100%, the theoretic and simulated DDR^{max} = 100%.

FRT-WICKPro vs SRT-WICKPro for FRT traffic support Table 6.8 shows the results obtained by FRT-WICKPro in Scenario 6.1 when PDR = 97% and FRT traffic is supported ($EED^{max} = P$). On the one hand, we verify that the simulated and the measured values are quite similar. The measured values were obtained in a real experiment using the embedded hardware employed in Chapter 5. On the other hand, we can compare FRT-WICKPro and SRT-WICKPro in the case of supporting FRT traffic. SRT-WICKPro supports FRT traffic in Tables 6.5 and 6.4 when $EED^{max} = P$. This comparison shows that FRT-WICKPro achieves higher DDR than SRT-WICKPro. In particular, when there is no retransmission allocated, the DDR obtained by FRT-WICKPro is about 30%[7] higher than the attained by SRT-WICKPro. In general, the lower the network utilization, the more similar both WICKPro versions are. This points out that FRT traffic may be better supported when synchronization is used, specially in high load conditions.

Table 6.8: DDR (%) in Scenario 6.1 using FRT-WICKPro when PDR = 97% and $EED^{max} = P$ (FRT traffic support)

Flow	Value	0 ReTx	1 ReTx	2 ReTx	3 ReTx
1	Measured	99.77	99.98	100	100
	Simulated	99.79	99.99	100	100
2	Measured	67.23	93.50	99.07	99.89
	Simulated	67.34	93.55	99.06	99.88

Bursty packet loss model We also simulated Scenario 6.1 using the simple Gilbert Model for the case $EED^{max} = 2P$. Table 6.9 shows the DDR when simulating four packet loss models with PDR equals 97%: one memoryless and three bursty models with ABL equal to 1.11, 2 and 4. Differences are smaller than the presented by FRT-WICKPro in Chapter 5[8], thus SRT-WICKPro is less sensible to bursty channels than FRT-WICKPro in the sense that it achieves more similar results to those of a memoryless packet loss model. The main reason is that SRT-WICKPro compensates losses in one minor cycle with other minor cycles, whereas FRT-WICKPro treats every minor cycle independently.

6.5.2 Scenario 6.2 - Robot Tele-operation

In this scenario, we carried out laboratory experiments in which a Pioneer 3-AT robot was tele-operated. We developed SRT-WICKPro in C and tested it using Linux and the Robot Operating System (ROS) framework [Quigley 09]. For

[7]It is increased from 64.25% ((95.30 + 33.19) / 2) to 83.57% ((99.79 + 67.34) / 2).

[8]Chapter 5 describes a simulation of FRT-WICKPro in bursty channels that is very similar to the presented in this section. The only difference is that Test 5.1 considers a network with seven routers while in this section there are five routers and two clients. Simulations of FRT-WICKPro in Scenario 6.1 are not therefore presented because their results are very similar to the presented in Test 5.1 through Tables 5.12 and 5.13.

Table 6.9: DDR (%) achieved by SRT-WICKPro without CC in Scenario 6.1 when PDR = 97%, $EED^{max} = 2P$ and four packet loss models are employed

ABL	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
Memoryless	1	67.45	99.99	100.00	100.00
	2	66.74	99.99	100.00	100.00
1.11	1	67.51	100.00	100.00	100.00
	2	66.74	100.00	100.00	100.00
2	1	68.90	99.92	100.00	100.00
	2	66.74	99.91	100.00	100.00
4	1	69.16	98.40	99.84	99.97
	2	65.69	97.93	99.78	99.98

the ROS integration, we took advantage of the ROS version of RT-WMP [ROS-RTWMP 16]. The network was composed of three routers (R1-R3) and one client (C4) as illustrated in Fig. 6.6. C4 was the mobile robot and R1 was the base station where the tele-operator was located. We configured the base station as a router because it was not mobile. Moreover, there were two multi-hop flows: (i) the tele-operation commands from R1 to C4 (flow 1), and (ii) the laser scan data from C4 to R1 (flow 2). The tele-operation commands were generated by a PlayStation3 joystick [ROS-PS3 16] and the laser scan data by a SICK LMS2xx laser [ROS-SICK-LMS2xx 16]. MATLAB simulations were also conducted to analyze the behavior of SRT-WICKPro in the robot tele-operation.

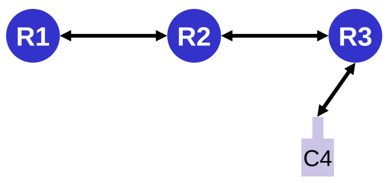


Figure 6.6: WMN employed in the mobile robot tele-operation (Scenario 6.2)

We measured the generation period of both flows as can be seen in Fig. 6.7. On the one hand, the laser data period was never lower than 101 ms and its mean was about 106 ms so that we set the period to 100 ms. On the other hand, the tele-operation command period was more complex to characterize because its generation period decreases if there is not any joystick movement. It is actually a sporadic flow with a mean of 17.15 ms and a median of 11.24 ms. We chose a minimum inter-arrival time of 10 ms because it has less computational cost to calculate the cyclic scheduling if all flows have harmonic periods. It should be noted that 1.8% of the packets arrived with a period lower than 10 ms.

The complete flow features are given by Table 6.10. Moreover, the token transmission time was 0.35 ms and the time-out was 10 ms. Table 6.11 details the computation of these transmission times, where the token packet is the largest forward token packet (the establishment packet). For this calculation, WICKPro used the following formula:

$$C(\mu s) = C_{802.11a}^{6Mbps}(\mu s) + C_{process}(\mu s) + C_{log}(\mu s) \qquad (6.4)$$

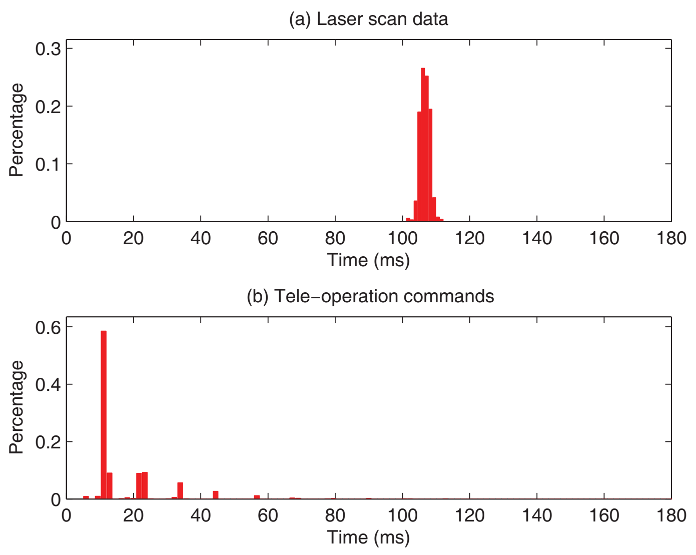


Figure 6.7: Histogram measurement in Scenario 6.2 with the generation period of (a) laser scan data, and (b) tele-operation commands

Table 6.10: Communication requirements in Scenario 6.2

Flow	Size (Bytes)	C (ms)	P (ms)	EED^{max}	src	dst
1	192	0.7	10	P, 1.1P, 1.3P, 1.5P, 2P and inf	R1	C4
2	802	1.48	100	P, 1.1P, 1.3P, 1.5P, 2P and inf	C4	R1

Table 6.11: Details of transmission time computation in Scenario 6.2

Type	Size (Bytes)	$C_{802.11a}^{6Mbps}(\mu s)$	$C_{process}(\mu s)$	$C_{log}(\mu s)$	$C(\mu s)$
Data1	192	500	150	50	700
Data2	802	1280	150	50	1480
Token	13	250	50	50	350

Comparing the token transmission time and the time-out presented above with those of Scenario 6.1, we can observe two issues: (i) the token has a lower transmission time, as less log information is sent with the token packet and lower process time is required, and (ii) the time-out is higher, while in Scenario 6.2 we used a non real-time operating system (Linux), but considered the MaRTE real-time operating system [Rivas 01] in Scenario 6.1 (similarly to Chapter 5).

The theoretic cyclic scheduling calculated for this scenario is shown in Fig. 6.8. The major and the minor cycles lasted 100 ms and 10 ms, respectively, and consequently there were 10 minor cycles. The network utilization of the minor cycle 1 (from 0 to 10 ms) was 65.40%, whereas the network utilization of the minor cycles ranging from 2 to 10 was 31.50%. We must recall that Fig. 6.8 depicts the theoretic cyclic schedule, in a way that this schedule will differ from the actual schedule at run-time, even if there is no packet loss, due to the asynchronous token-passing scheme implemented by SRT-WICKPro.

Next Section 6.5.2.1 presents simulation and laboratory experiments using synthetic traffic with the features provided by Table 6.10, and Section 6.5.2.2 describes laboratory experiments where the robot tele-operation was actually carried out.

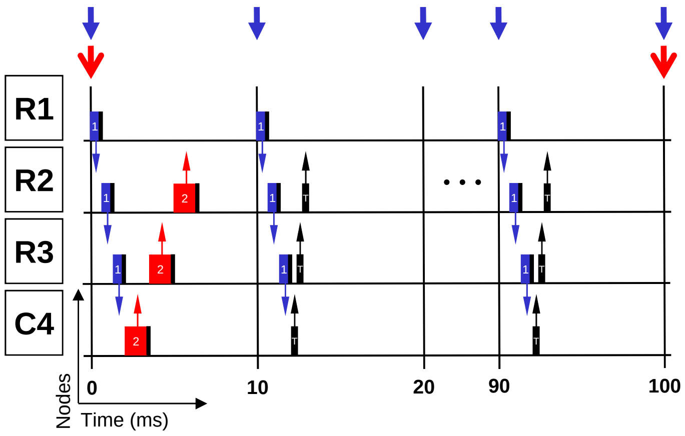


Figure 6.8: Theoretic cyclic schedule of the mobile robot tele-operation carried out in Scenario 6.2

6.5.2.1 MATLAB Simulation and Real Implementation Using Linux

As mentioned above, before testing the robot tele-operation using ROS, we carried out MATLAB simulations and laboratory experiments. In the latter experiments, we used Linux and synthetic data flows generated by external threads with the features given by Table 6.10. Furthermore, deliberate errors were introduced by software to achieve the desired link qualities. Every test of the laboratory experiments lasted 1,000 seconds whereas each simulation had an equivalent duration of 10,000 seconds.

Data Delivery Ratio Table 6.12 shows the DDR obtained for both types of experiments assuming a memoryless packet loss model with PDR equal to 97% and 95% (when PDR was 100%, the DDR was always 100%).

Table 6.12: DDR (%) in Scenario 6.2 with and without CC by using MATLAB simulations and laboratory experiments with synthetic data flows

EED^{max}	**Flow**	**PDR = 0.97**			**PDR = 0.95**		
		MAT^{CC}	MAT	Linux	MAT^{CC}	MAT	Linux
P	1	69.79	69.81	69.27	58.16	58.16	57.81
	2	99.94	99.92	99.96	98.95	98.91	98.84
1.1P	1	73.17	73.11	72.58	61.99	61.64	61.29
	2	99.96	99.98	99.87	99.37	99.56	99.45
1.3P	1	77.59	79.40	80.30	66.84	67.83	68.11
	2	99.96	100	99.96	99.44	99.93	99.94
1.5P	1	81.43	88.47	87.02	71.27	77.36	76.29
	2	99.97	100	100	99.45	99.99	99.97
2P	1	83.00	93.68	93.15	74.29	85.45	84.65
	2	99.95	100	100	99.39	100	100
Infinite	1	84.28	100	100	76.92	100	100
	2	99.97	100	100	99.41	100	100

In MATLAB, we simulated both SRT-WICKPro with and without congestion control (MAT^{CC} and MAT in Table 6.12, respectively). It can be seen that, in Scenario 6.2, SRT-WICKPro with congestion control attains lower DDR because the network utilization is about 35% in average and, as shown in Scenario 6.1, this strategy is better when network utilization is high, since network congestion is higher. Moreover, the difference between the version without and with CC is higher when the EED^{max} is higher (MAT vs MAT^{CC}). The reason is that packets are dropped at source nodes when using the congestion control algorithm, but they could be delivered properly with a delay lower than EED^{max}, specially when EED^{max} is higher. Thus, a congestion control algorithm may increase the network performance but it must be smart to adapt to every particular situation.

When comparing SRT-WICKPro without congestion control in MATLAB and Linux, results are very similar, however there is a slight difference due to the following reasons: (i) token duplication is ignored in the simulation, (ii) the transmission time varies in the real experiments due to the back-off mechanism in IEEE 802.11, (iii) the use of a non-real-time operating system in the real experiments, which causes some timing imprecision, and (iv) the lack of global clock information for computing data packet delay in the real experiments. This latter issue concerns how to calculate the accumulated delay of a data packet which has passed through several nodes if there is no synchronization in the network. For this purpose, WICKPro has a field in the log information of data packets that indicates the accumulated delay. This field is filled at every node by adding the process time as well as the theoretical transmission time, but this is not constant, and the delay computation is therefore an approximation. This situation is different from MATLAB simulation where all times are deterministic.

Delay The delay histogram measured in Scenario 6.2 for PDR = 97% when using Linux and SRT-WICKPro without CC is shown in Fig. 6.9 and 6.10 in the case of $EED^{max} = P$ and $EED^{max} = 2P$, respectively. As expected, the higher the EED^{max}, the higher the mean delay. This is also verified in Table 6.13, where the mean and standard deviation of the measured delays are given. It is important to highlight that the delay distribution of the laser scan data (flow 2) does not change too much when EED^{max} is increased from P to $2P$, since the DDR of flow 2 is 99.96% when $EED^{max} = P$ and 100% when $EED^{max} = 2P$.

Table 6.13: Mean ± std dev delay (ms) measured in Scenario 6.2 without CC when PDR = 97%

Flow	$\mathbf{EED^{max} = P}$	$\mathbf{EED^{max} = 2P}$
1	4.15 ± 2.28	7.31 ± 4.80
2	28.59 ± 17.04	29.19 ± 17.76

Bursty packet loss model Scenario 6.2 was also simulated using the simple Gilbert Model for the case $EED^{max} = 2P$. Table 6.14 illustrates the DDR when simulating four packet loss models with PDR equals 97%: one memoryless and three bursty models with ABL equal to 1.11, 2 and 4. Surprisingly, the four cases exhibit a similar behavior. The explanation lies in the fact that the channel estimation considers a saturated channel (continuous transmission in every link), but this is not the case in practice. For this reason, the memory effect disappears, as is thoroughly analyzed in [Gómez 15]. For a deeper understanding, we can observe that the

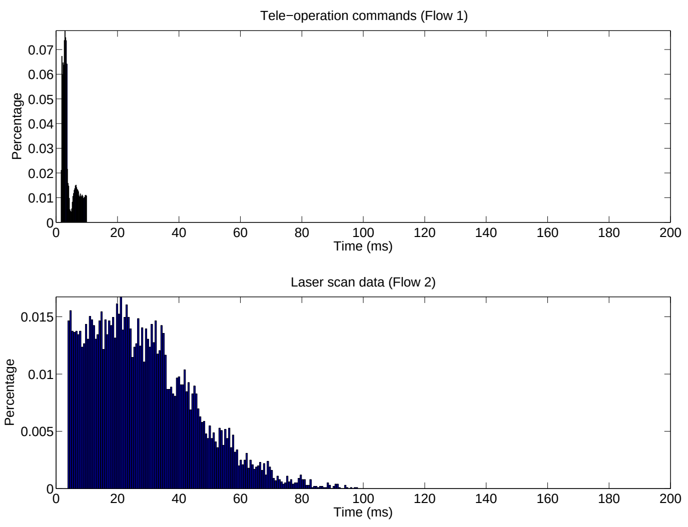


Figure 6.9: Delay histogram of flows 1 and 2 measured in Scenario 6.2 using Linux without CC when PDR = 97% and $EED^{max} = P$ ($EED_1^{max} = 10$ ms and $EED_2^{max} = 100$ ms)

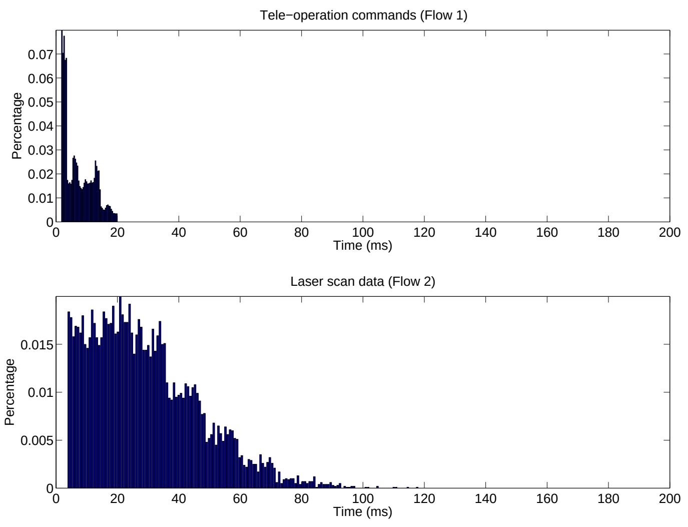


Figure 6.10: Delay histogram of flows 1 and 2 measured in Scenario 6.2 using Linux without CC when PDR = 97% and $EED^{max} = 2P$ ($EED_1^{max} = 20$ ms and $EED_2^{max} = 200$ ms)

maximum ABL is 4 but the time-out (10 ms) is higher than four times the maximum packet transmission time (1.48 ms), and the burst effect is therefore totally lost in our simulations[9]. Actually, as shown in Table 6.14, the DDR is higher if the channel has higher ABL. The reason is that the PDR is the same in the four experiments, thus the higher the ABL, the lower p must be to yield the same PDR value[10]. Therefore, the resulting PDR is lower because p is lower and q does not affect, as the memory effect disappears.

[9]This is not the case in Scenario 6.1 where the time-out was 4.18 ms and the maximum packet transmission time was 2.09 ms.

[10]As ABL $= 1/q$, the higher the ABL, the lower q. And PLR $= p/(p+q)$, so that the lower q, the lower p must be to yield the same PLR.

Table 6.14: DDR (%) in Scenario 6.2 without CC when PDR = 97%, $EED^{max} = 2P$ and four packet loss models are employed

Flow	Memoryless	ABL = 1.11	ABL = 2	ABL = 4
1	93.68	93.65	93.88	94.49
2	100	100	100	100

DDR test in error-free scenario As commented previously, when the PDR was 100%, the DDR was always 100%. For this reason, we carry out the DDR test in the so-called Scenario 6.2b where the periods of the flows are changed. The communication requirements in this case are shown in Table 6.15, where the period of flow 2 continues to be ten times the period of flow 1. We only analyze the first minor cycle because it represents the worst-case scenario, as illustrated in Fig. 6.8. It should be noted that the cyclic packet schedules of Scenario 6.2 and 6.2b are the same, expect for the minor cycle duration.

Table 6.15: Communication requirements in Scenario 6.2b to apply the DDR test

Flow	Size (Bytes)	C (ms)	P (ms)	EED^{max}	src	dst
1	192	0.7	6.54, 18.02, 29.5 and 40.98	P, 1.1P, 1.3P, 1.5P, 2P and inf	R1	C4
2	802	1.48	65.4, 180.2, 295 and 409.8	P, 1.1P, 1.3P, 1.5P, 2P and inf inf	C4	R1

When no retransmission is allocated, the minor cycle duration is 6.54 ms, i.e., the time required to transmit three packets of flow 1 (0.7 ms) and three packets of flow 2 (1.48 ms). In this situation, the DDR test in Eq. (6.1) does not always return 100% for flow 1 in an error-free scenario. Specifically, R_1 is 7.59[11] ms and EED_1^{max} is 6.54, 7.19 and 8.50 ms for the cases EED_1^{max} equal to P_1, $1.1P_1$ and $1.3P_1$, respectively. Thus, DDR for flow 1 will not be 100% when $EED_1^{max} = P_1$ or $EED_1^{max} = 1.1P_1$. This matches the simulation results, as shown in Table 6.16, where we obtained a DDR equal to 98.99% and 99.33% in these cases. When EED_1^{max} is higher than $1.1P_1$, as well as for flow 2, simulation and Eq. (6.1) yield a DDR of 100%.

When one, two and three retransmissions are allocated, the theoretic and simulation DDR is 100% in all cases.

6.5.2.2 Real Implementation Using ROS

Finally, we conducted the real tele-operation using ROS. The achieved DDR can be seen in Table 6.17 when considering a memoryless packet loss model (deliberate errors were introduced by software). The results are similar to those of simulation and real implementation using synthetic data (see Table 6.12), although there are some differences due to the following two issues. First, the real periods of flow 1 and 2 last about 11 and 106 ms, respectively, while the synthetic flow periods are 10 and 100 ms. This higher separation between packets explains that the DDR is higher when ROS is used and EED^{max} is high enough (mostly $EED^{max} > P$). Second, flow 1 is actually sporadic traffic in a way that approximately 1.8% of the packets arrives with a period lower than 10 ms, based on the flow characterization previously

[11]This is the time to transmit three token packets, three packets of flow 2 and three packets of flow 1: $3 * 0.35 + 3 * 1.48 + 3 * 0.7 = 7.59$ ms.

Table 6.16: DDR (%) in Scenario 6.2b without CC when PDR = 100%

$\mathbf{EED^{max}}$	Flow	0 ReTx	1 ReTx	2 ReTx	3 ReTx
P	1	98.99	100	100	100
	2	100	100	100	100
1.1P	1	99.33	100	100	100
	2	100	100	100	100
1.3P	1	100	100	100	100
	2	100	100	100	100
1.5P	1	100	100	100	100
	2	100	100	100	100
2P	1	100	100	100	100
	2	100	100	100	100
Infinite	1	100	100	100	100
	2	100	100	100	100

presented. For this reason, the DDR is lower in the experiments that employ ROS and EED^{max} is small (mainly $EED^{max} = P$), as the lower separation between packets has more influence with shorter deadlines. It is specially noteworthy that when PDR = 100%, unlike simulation and real implementation using synthetic data, the DDR of flow 1 is not always 100%. In short, the real periods are usually higher than the synthetic periods, however, the real flow 1 is generated with a lower period in a small percentage of times, which is more relevant for the DDR when EED^{max} is small.

Table 6.17: DDR (%) in Scenario 6.2 without CC using ROS

$\mathbf{EED^{max}}$	Flow	PDR = 1	PDR = 0.97	PDR = 0.95
P	1	99.97	68.99	57.25
	2	100	99.95	99.99
1.1P	1	99.99	72.53	61.66
	2	100	99.87	99.67
1.3P	1	99.97	80.84	69.57
	2	100	99.99	99.93
1.5P	1	99.98	88.40	77.94
	2	100	100	100
2P	1	100	93.59	86.41
	2	100	100	100
Infinite	1	100	100	100
	2	100	100	100

These results become useful for the design of a robot tele-operation application. For instance, in our network, choosing $EED^{max} = 2P$ could match the application requirements and the protocol would provide a good DDR, but this depends on the link quality. The application could also decimate the tele-operation packets at source node, e.g., taking one packet every five [Tardioli 14]. However, we did not use this option to obtain a more loaded network.

The delay histogram measured in Scenario 6.2 for PDR = 97% when using ROS and SRT-WICKPro without CC is shown in Fig. 6.11 in the case $EED^{max} = 2P$.

When comparing this figure with Fig. 6.10, which shows the results of the same experiment but using Linux instead of ROS, we can see that flow 1 presents a similar delay histogram while the histogram of flow 2 is different, specifically Fig. 6.10 exhibits a more uniform distribution. Regarding the mean and standard deviation of the measured delays, flows 1 and 2 achieve 7.11 ± 4.75 ms and 18.94 ± 16.56 ms, respectively. Thus, flow 1 presents similar values to those of Table 6.13 (when $EED^{max} = 2P$) and flow 2 attains a lower mean delay. The latter is explained because the real flow 1, being sporadic traffic, generates less packets than the synthetic flow 1 and therefore the real flow 2 is sent with less interference of flow 1, which results in a lower average delay.

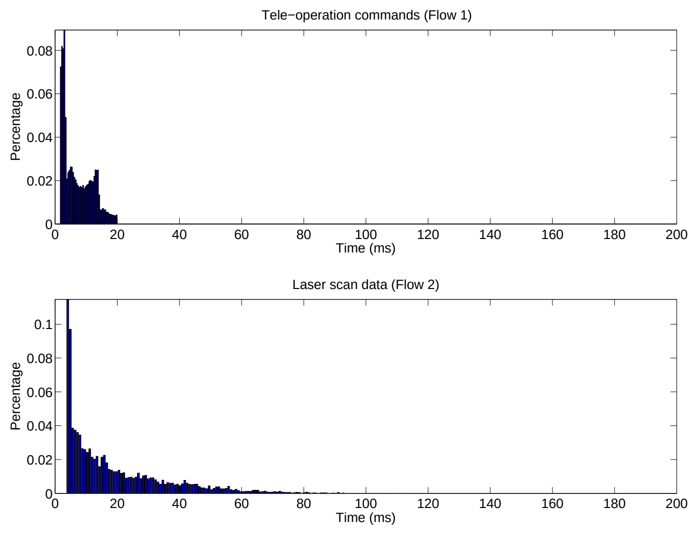


Figure 6.11: Delay histogram of flows 1 and 2 measured in Scenario 6.2 using ROS without CC when PDR = 97% and $EED^{max} = 2P$ ($EED_1^{max} = 20$ ms and $EED_2^{max} = 200$ ms)

6.6 Conclusions

This chapter presented and evaluated SRT-WICKPro, which manages periodic traffic with soft criticality by using an asynchronous token-passing scheme and a cyclic packet scheduler. In particular, we adapted the WICKPro protocol to support applications that find some value in packets with delays higher than their deadline, considered equal to the packets minimum inter-arrival time (P). Thus, the response time of the protocol can grow until EED^{max}, i.e., a packets delivery window, and the throughput may be significantly increased. We tested two different scenarios in simulation and laboratory experiments. The main conclusion is that the higher the network utilization and the packet losses, the higher the EED^{max} must be to achieve the same throughput. However, if EED^{max} is increased, the average packet delay will also be incremented. In a given scenario, the DDR increased by 42% when EED^{max} was changed from P to $2P$, considering SRT-WICKPro with congestion control in a saturated network. Moreover, we showed that congestion control may increase network performance in our protocol, too.

As a case study, we presented a robot tele-operation in a WMN with chain topology. The complete process from simulation to real experiment was detailed.

7 Mobility Management

In this chapter, the Double-Threshold Hand-off (DoTHa) algorithm is presented. DoTHa is designed and implemented within SRT-WICKPro to let this protocol manage mobility. In this way, clients can move while supporting real-time communication. We choose SRT-WICKPro because it is simpler than FRT-WICKPro and enough to support SRT traffic. Section 7.2 introduces the problem statement, Section 7.3 shows the related work in mobility management, Section 7.4 presents DoTHa, Section 7.5 details the features of the mobile robot tele-operation that we employ as case study, Section 7.6 carries out a performance comparison between DoTHa and the hand-off mechanisms described in Algorithms 7.1 and 7.2, and Sections 7.7 and 7.8 show the tele-operation of a mobile robot in the corridors of the I3A building and in the Somport tunnel, respectively.

Mobile robot tele-operation can be used, for example, in real-time monitoring and robotized machinery. The latter refers to operate remotely robotized machinery such as dumper trucks and tunneling machines to avoid endangering human lives. This is specially important in situations where technology for autonomous robots is not mature enough, as happens in reconfigurable manufacturing systems [Lindhorst 13] and to make life-and-death decisions, e.g., after a disaster with radioactivity contamination.

7.1 Relevant Publications

The work presented in this chapter was mainly published in the following paper:

- [Aisa 16b] J. Aisa, H. Fotouhi, L. Almeida and J.L. Villarroel. *DoTHa - A Double-threshold Hand-off Algorithm for Managing Mobility in Wireless Mesh Networks.* In IEEE Conference on Emerging Technologies and Factory Automation (ETFA), September 2016.

7.2 Problem Statement

Hand-off must select the best router in range when the client moves. One of the major issues with a hand-off process is the ping-pong effect, a situation where a client switches back and forth between two or more routers, adding instability to the connection and potential packet losses. Moreover, ping-pong effect causes delay variation in the end-to-end communication due to the route change. Thus, DoTHa aims at providing seamless connectivity while avoiding the ping-pong effect. This is a complex task due to the presence of shadowing and multi-path fading, as explained in Section 2.3.

We assume that clients store RSSI measurements of the packets received from all routers. Thus, a client i collects the following measurements:

$$RSSI_i = \{RSSI_{1i}, RSSI_{2i}, ..., RSSI_{N_R i}\} \tag{7.1}$$

$RSSI_{ji}$ shows the RSSI at client i from router j. Thus, the current RSSI (considering that client i is attached to router r), and the maximum RSSI at client i are respectively defined as:

$$RSSI_i^{current} = RSSI_{ri} \tag{7.2}$$

$$RSSI_i^{max} = \max_{1 \leq j \leq N_R} \{RSSI_{ji}\} \tag{7.3}$$

7.3 Related Work in Mobility Management

Mobility management in WMNs has been mainly addressed by (i) localization algorithms to estimate the position of mobile nodes, and (ii) hand-off algorithms to manage changing point of attachment [Xie 08]. We are specially interested in hand-off mechanisms because they can provide good performance while avoiding localizing mobile nodes.

7.3.1 Hand-off Basics

Hand-off (or hand-over) is a process in which a mobile node attached to a backbone network through a router changes the attachment point to another network router. The challenge is to select the best router in range as the mobile node moves. Hand-off mechanism targets both MAC sublayer when getting access to a mesh router through a single hop, and network layer when getting access to other nodes in multi-hop, through the routers of the mesh network.

Hand-off can be divided in three phases [Benedetto 13]:

- Trigger. Hand-off is started when the channel conditions degrade to an unacceptable quality.
- Search. Once hand-off is triggered, the search for a new router is started, where scanning can be passive or active. In passive scanning, routers send beacon frames periodically to let mobile nodes measure the signal strength. In active scanning, the mobile node transmits broadcast probe frames and routers send back probe responses frames to the mobile node. Scanning typically causes connection interruption, specially when single-channel devices are employed in multi-channel networks.
- Execution. When the new router is selected, the hand-off is executed and the mobile node becomes associated with the new router. Typically, this phase introduces a delay lower than the search phase: tens of milliseconds versus hundreds of milliseconds.

7.3.2 Hand-off Mechanisms

We classify hand-off schemes into gateway-based, probing-based, and prediction-based solutions:

Gateway-based solution This approach considers WMNs with static mesh routers acting as Gateways, providing wired access to the Internet [Li 14, Lakshmi 15].

Probing-based solution It focuses on periodic probing to assess link quality and obtain faster hand-offs thus reducing hand-off delay [Ramani 05, Collotta 15]. In [Ramani 05], authors implement a hand-off algorithm on top of IEEE 802.11 called SyncScan, which allows mobile nodes to regularly switch to other channels and record the signal strength of these other channels. However, this regular monitoring requires an accurate time synchronization between neighbor nodes.

Prediction-based solution Some schemes of this category take advantage of location information, predicting mobile nodes position and estimating the best router in the future, facilitating the hand-off process [Shin 04, Almulla 14]. In [Almulla 14], a GPS is employed to estimate location of nodes in vehicular networks. In particular, each mobile node acquires its location and movement direction from a GPS receiver, and selects the APs with higher probability of being located in its future path as potential candidates. Thus, the hand-off delay will reduce drastically.

Discussion Gateway-based solutions are more costly as they require Internet connection and a fixed wired backbone. Periodic probing solutions imply more network overhead due to frequent beacon transmissions. Prediction-based solutions are either hardware-based that add extra cost, or software-based that are inaccurate for mobile networks with high link dynamics. In this PhD thesis, we propose the DoTHa hand-off algorithm within the WICKPro protocol reconciling probing with token-passing. DoTHa employs the RSSI for hand-off purposes because RSSI-based hand-off algorithms work properly and RSSI is simple to measure, as stated in 2.2.2. Our scheme presents the benefit of carrying out probing without consuming extra overhead, as it takes advantage of the data and control packets transmitted in the network to collect RSSI measurements, continuously.

7.3.3 RSSI-based Hand-off Mechanisms

Being simple and effective, we put special emphasis on RSSI-based hand-off algorithms. We distinguish two main classes of procedures based on single and double hysteresis margin.

7.3.3.1 Single Hysteresis-based Hand-off

In this method, there is one single threshold and one single hysteresis margin for assessing link quality [Fotouhi 15], whose basic functionality is summarized in Algorithm 7.1. The mobile node starts the hand-off process when the RSSI of the link with its current serving router drops below a fixed lower threshold (T_l), and stops searching for a new router when the highest RSSI measurement of all neighbor routers is higher than or equal to a fixed higher threshold (T_h), which is defined as $T_h = T_l + HM$, where HM stands for the Hysteresis Margin. This margin allows reducing the possibility of ping-pong effect as it can compensate RSSI fluctuations. For the same reason, RSSI measurements are averaged before comparing to the threshold levels. It should be noted that some approaches in the literature employ a relative higher threshold ($T_h = RSSI_i^{current} + HM$).

Algorithm 7.1 Single hysteresis-based algorithm

```
1: function EXECUTE HAND-OFF
2:     if $RSSI_i^{current} \leq T_l$ then
3:         if $RSSI_i^{max} \geq T_l + HM$ then
4:             return $True$
5:         end if
6:     end if
7:     return $False$
8: end function
```

7.3.3.2 Double Hysteresis-based Hand-off

This scheme considers two hysteresis margins to increase decision accuracy [Bisti 11, Benedetto 13]. As explained in Algorithm 7.2, this method always carries out a hand-off if there is a router with enough higher RSSI than the serving router. On the one hand, if $RSSI_i^{current}$ is higher than or equal to a fixed threshold level $\beta = -70$ dBm, hand-off is performed if the RSSI of a neighbor router is higher than or equal to $RSSI_i^{current} + HM^G$, where HM^G is the hysteresis margin in the connected region (good link quality). On the other hand, if $RSSI_i^{current}$ is lower than β, hand-off is carried out if the RSSI of a neighbor router is higher than or equal to $RSSI_i^{current} + HM^B$ and the PDR is higher than or equal to a packet delivery threshold P_H[1]. HM^B is the hysteresis margin in the transitional region (bad link quality). It should be noted that this algorithm employs two relative higher thresholds, and defines $HM^G = 6$ dB and $HM^B = 3$ dB.

Algorithm 7.2 Double hysteresis-based algorithm [Bisti 11]

```
1: function EXECUTE HAND-OFF
2:     if $RSSI_i^{current} \geq \beta$ then
3:         if $RSSI_i^{max} \geq RSSI_i^{current} + HM^G$ then
4:             return $True$
5:         end if
6:     else
7:         if $RSSI_i^{max} \geq RSSI_i^{current} + HM^B$ AND $PDR \geq P_H$ then
8:             return $True$
9:         end if
10:    end if
11:    return $False$
12: end function
```

[1]The original work in [Bisti 11] used the Packet Loss Rate but we adapted it to the Packet Delivery Ratio for consistency with our framework.

7.3.3.3 Discussion

If hand-off was always carried out in the connected region, using Algorithm 7.1 would be an acceptable solution, however, this is impossible to guarantee. The actual deployment and environmental dynamics prohibit persistent connected regions in all wireless links. Thus, transitional regions are expected in different wireless links. For instance, it is typical in factories that machinery and mobile robots obstruct the communication between routers and clients, thereby creating shadowing and multi-path fading.

In case a client suffers shadowing for a long period of time, RSSI would significantly decrease in all links between routers and the corresponding client in a way that these links would enter the transitional region and network disconnections would happen. In this situation, the hand-off process should use a relative higher threshold to let a client perform hand-off if the RSSI of a router is higher than the RSSI of its serving router, considering a predefined hysteresis margin. Although this is the proposal carried out in Algorithm 7.2, this mechanism uses a relative higher threshold in both the connected and the transitional regions. The problem here is that a relative higher threshold usually produces more ping-pong effect than a fixed higher threshold.

Hence, we propose an algorithm dubbed as DoTHa that uses two higher threshold levels: a fixed higher threshold in the connected region with average link quality, and a relative higher threshold in the connected region with low link quality and in the transitional region. We claim that a relative higher threshold is only worthwhile when all links provide low quality, because this lets select the router with the best link amongst all the low quality links. Nevertheless, in the connected region with good link quality, this is not necessary because the serving router is providing a good connection. This way we can handle hand-off in different environmental conditions.

7.4 The Double-Threshold Hand-off Algorithm

We propose a simple, light and reliable algorithm where each client takes hand-off decisions based on its local RSSI measurements collected from neighbor routers. As it is implemented on SRT-WICKPro, all nodes are involved in the token pass and clients benefit from this to store RSSI values locally, related to the links between the mobile node and the routers in the transmission range, thus supporting on-the-fly hand-off. As commented in Section 7.3.2, this scheme has the feature of carrying out proactive probing without consuming extra overhead, as it takes advantage of the data and control packets[2] transmitted in the network to collect RSSI measurements of the links between routers and clients, continuously. Since at least a full token rotation is carried out in each minor cycle, at least one new RSSI measurement is available every minor cycle.

[2]All control packets can be used to read its RSSI value, namely regular token, establishment, removal, hand-off and NACK packets. We must recall that the drop is not employed in SRT-WICKPro.

7.4.1 Description

The DoTHa procedure is described in Fig. 7.1 and Algorithm 7.3. DoTHa defines three different states for hand-off initiation with respect to link features. These states are determined by a fixed lower threshold that separates the high and average link quality in the connected region T_l^G, and by another fixed lower threshold that splits the average and low link quality in the connected region T_l^B. The states and the hand-off conditions are the following:

1. No hand-off in the connected region with high link quality.
2. Hand-off in the connected region with average link quality. A hand-off is carried out if the highest RSSI of neighbor routers is more than T_l^G plus HM^G, i.e., a fixed higher threshold is used.
3. Hand-off in the connected region with low link quality and in the transitional region. A hand-off is performed if the highest RSSI of neighbor routers is more than the current RSSI plus HM^B, thereby employing a relative higher threshold. It should be noted that, since we intend to avoid that links enter the transitional region due to its higher packet losses, the relative higher threshold is already defined in the connected region with low link quality to make hand-off more likely before entering the transitional region.

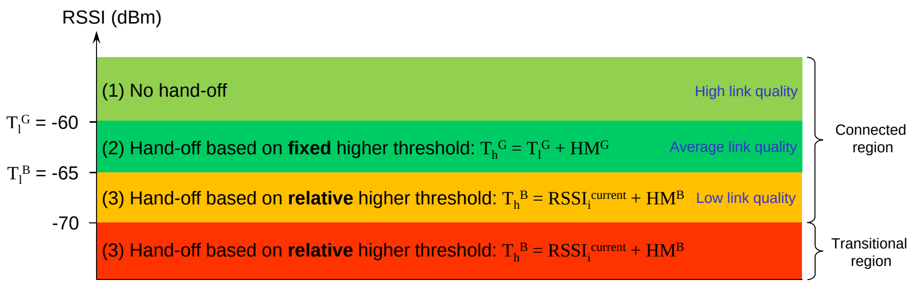

RSSI (dBm)
T_l^G = -60
T_l^B = -65
-70
(1) No hand-off
High link quality
(2) Hand-off based on **fixed** higher threshold: $T_h^G = T_l^G + HM^G$
Average link quality
(3) Hand-off based on **relative** higher threshold: $T_h^B = RSSI_i^{current} + HM^B$
Low link quality
(3) Hand-off based on **relative** higher threshold: $T_h^B = RSSI_i^{current} + HM^B$
Connected region
Transitional region

Figure 7.1: Hand-off states, threshold levels and hysteresis margins in DoTHa

Algorithm 7.3 DoTHa algorithm

```
1: function Execute Hand-off
2:     if $T_l^B < RSSI_i^{current} \leq T_l^G$ then
3:         if $RSSI_i^{max} \geq T_l^G + HM^G$ then
4:             return $True$
5:         end if
6:     else if $RSSI_i^{current} \leq T_l^B$ then
7:         if $RSSI_i^{max} \geq RSSI_i^{current} + HM^B$ then
8:             return $True$
9:         end if
10:    end if
11:    return $False$
12: end function
```

In our experiments, we selected the following values: $T_l^G = -60$ dBm, $T_l^B = -65$ dBm, $HM^G = 5$ dB and $HM^B = 5$ dB.

RSSI is smoothed before being compared with the threshold levels to compensate its variations. The objective is to remove variations caused by multi-path fading but preserving variations introduced by shadowing. In this way, multi-path fading is neglected for hand-off purposes, which is logical because RSSI variations due to multi-path fading are likely to change faster than adaptations can be made. For this reason, multi-path fading is usually mitigated at physical layer by using redundancy in time, frequency or space. Finally, it must be emphasized that the RSSI smoothness along with the hysteresis margin try to avoid the ping-pong effect.

7.4.2 Executing the Hand-off

After showing how DoTHa triggers hand-off and monitors RSSI, in this section we present how hand-off is indicated within the WICKPro protocol. Consider that a client k wants to execute a hand-off. When client k receives the regular token packet in minor cycle i, this client changes the regular token packet for a hand-off packet where its WICKPro address and the WICKPro address of the new router are announced[3]. This hand-off packet is maintained during minor cycles i and $i+1$, which ensures that all nodes are properly informed. In minor cycle $i+2$, the hand-off process is concluded and the token master replaces the hand-off packet with a regular token packet that sets the *NewScheduling* flag in the field *Notification*. Thus, the new cyclic scheduling considering the hand-off process will initiate. It should be noted that this process is similar to the admission phase shown in Fig. 4.2.

This procedure provides continuous network accessibility to the client involved in the hand-off process and therefore the hand-off delay[4] is zero. The reason is that the client continues receiving the token and transmitting data packets if ready and if scheduled by the cyclic scheduling. In case of simultaneous hand-off requests, only one of them is managed at a time to enforce consistency of topological information in all nodes. This serialization of requests is trivially handled by the token rotation and each request takes three minor cycles.

As a hand-off packet only has two bytes more than a regular token packet (see Appendix A.1), the hand-off indication does not consume almost extra information. Thus, the implementation of DoTHa in SRT-WICKPro yields a light hand-off algorithm, given that the RSSI is also monitored without requiring further overhead.

7.4.3 Widespread Use of DoTHa

DoTHa is generic and can therefore be implemented with other wireless protocols whenever mobile nodes can monitor the RSSI. In DoTHa, hand-off is triggered based on the RSSI of the serving router and, once triggered, the protocol uses the RSSI of neighbor routers. DoTHa carries out probing with the help of token rotations, but other active or passive scanning methods could also be used, as well as multi-channel scanning, if needed. For this reason, DoTHa could be applied to wireless industrial automation protocols such as ISA-100.11a and WirelessHART. However, integration should be carefully designed to guarantee interoperability between standard and DoTHa-enabled nodes. Likewise, hand-off indication should also be implemented.

[3]All nodes are assigned a 1-byte address called WICKPro address or simply address.

[4]Hand-off delay is defined as the time where the network is inaccessibility for the mobile node involved in the hand-off process.

7.5 Case Study: Mobile Robot Tele-operation

We carried out the tele-operation of a mobile robot as a case study to evaluate the DoTHa algorithm. For this purpose, we deployed a WMN with chain topology composed of three routers (R1-R3) and one client (C4). C4 was the mobile robot while R1 was the token master and the base station where the tele-operator was located. The base station was configured as a router because it was not mobile. We defined two multi-hop flows: (i) the tele-operation commands from R1 to C4 (flow 1), and (ii) the laser scan data from C4 to R1 (flow 2). The features of the real data flows has already been studied in Chapter 6 and their traffic model is depicted again in Table 7.1, in which we chose $EED^{max} = 2P$. Likewise, the token transmission time was 0.35 ms and the time-out was 10 ms.

Table 7.1: Flow features of the mobile robot tele-operation

Flow	Size (Bytes)	C (ms)	P (ms)	EED$^{\mathbf{max}}$	src	dst
1	192	0.7	10	20	R1	C4
2	802	1.48	100	200	C4	R1

In these experiments, we used four laptops running the Linux operating system and SRT-WICKPro, where one of these laptops was placed on top of the mobile robot. Each laptop was equipped with one Atheros chipset-based wireless card whose RSSI measurement ranges from 0 to 60. As commented in Section 2.2.2, this value can be converted to dBm by subtracting 95, thus RSSI in dBm ranges from −35 dBm at 100% to −95 dBm at 0%. In all the experiments, wireless cards were configured at 6 Mbps using CSMA/CA without RTS/CTS. Moreover, all the transmissions were broadcast thus without hardware-level ACKs of IEEE 802.11.

Scenario Description

We tested the robot tele-operation in four scenarios whose main features are shown in Table 7.2. In Scenarios 7.1 and 7.2, the client was actually a person moving with a laptop, thus we used synthetic data flows with the properties given in Table 7.1 to emulate the robot tele-operation. The transmission frequency was set to 5.3 GHz (IEEE 802.11a) in Scenarios 7.1, 7.2 and 7.3 to avoid external interferences, while was set to 2.412 GHz (IEEE 802.11g) in Scenario 7.4 because external interferences were non-existent and the propagation at that frequency in the Somport tunnel has been thoroughly studied in our research group [Rizzo 15]. More information about the hardware involved in the robot tele-operation can be found in Appendix B.2.

Table 7.2: Scenarios where the robot tele-operation was tested

Scenario	Place	Flows	Tx Power (dBm)	Details
7.1	I3A building	Synthetic	20	
7.2	I3A building	Synthetic	20	C4 carries an attenuator of 20 dB
7.3	I3A building	Real	20	C4 carries an attenuator of 20 dB
7.4	Somport tunnel	Real	1	R3 does not carry external antenna

SRT-WICKPro Scheduling

To ensure the schedulability in an error-free network with any topology, we calculate the scheduling in the worst-case scenario, i.e., when the hop count of the data flows is the largest possible. Fig. 7.2 and 7.3 show the network topology and the theoretical cyclic scheduling in this case. As mentioned in Chapter 6, Fig. 7.3 depicts the theoretic cyclic schedule, in a way that this schedule will differ from the actual schedule at run-time, even if there is no packet loss, due to the asynchronous token-passing scheme implemented by SRT-WICKPro.

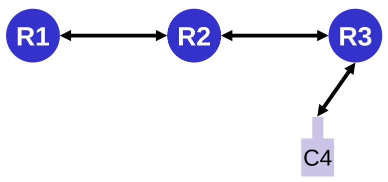


Figure 7.2: Network topology in the worst-case scenario with three routers and one client in the mobile robot tele-operation

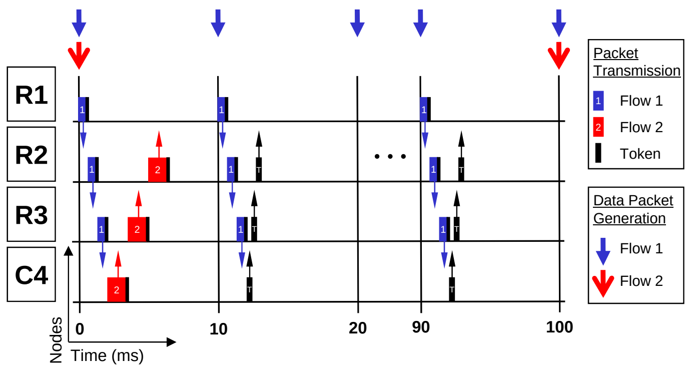


Figure 7.3: Theoretical cyclic schedule of the mobile robot tele-operation considering the worst-case network topology

RSSI smoothness

The RSSI was smoothed with two different filters: (i) Simple Moving Average (SMA) with window size = 50, i.e., the RSSI was averaged over the last 50 received samples, and (ii) Exponentially Weighted Moving Average (EWMA) with $\alpha = 1/256$, which is defined as follows:

$$RSSI[n] = \alpha RSSI_{sample} + (1 - \alpha)RSSI[n - 1] \tag{7.4}$$

where α is the constant smoothing factor between 0 and 1, n is the number of received samples and $RSSI_{sample}$ is the last received RSSI sample.

Performance metrics

We computed the DDR and the number of hand-offs in order to evaluate the implemented hand-off algorithms. The DDR metric gives the percentage of received packets at application layer with a delay lower than EED^{max}, whereas the number of hand-offs identifies the existence of ping-pong effect. Although hand-off algorithms are usually evaluated by computing losses at physical layer through the PDR, we show the DDR because we can compare these values with experiments in Chapter 6. Moreover, the PDR provided similar results than DDR in our experiments.

7.6 Experimental Comparison

We compared DoTHa with the Algorithms 7.1 and 7.2. For this purpose, the three procedures were implemented in WICKPro and field experiments were carried out in Scenarios 7.1 and 7.2. We tested Algorithm 7.1 with HM = 5 dB, while selecting three different T_l values, namely -60, -65, and -70 dBm. For Algorithm 7.2, we chose the same values as proposed by its authors: $\beta = -70$ dBm, $HM^G = 6$ dB, $HM^B = 3$ dB. In this algorithm, packet losses were neglected due to the absence of external interferences. We performed experiments with these five strategies in Scenarios 7.1 and 7.2 by using the two aforementioned filters. SMA 50 filter did not provide an adequate smoothed RSSI for the selected HM and the client speed, which caused ping-pong effect, while EWMA 1/256 filter smoothed RSSI properly.

Fig. 7.4 depicts the corridors of the I3A building where Scenarios 7.1, 7.2 and 7.3 were implemented. In this scenario, we did not place a router in the corner 4 to create a situation where the probability of shadowing effect increases. For instance, when C4 travels from R1 toward R3, and crosses the corner 4, there will be non-line-of-sight with its current router (R1), and line-of-sight with a new distant router (R3). This is a realistic situation that occurs in different environments since it is impossible to place routers in all corners. Even if it were possible, environmental dynamics would cause shadowing effect. In Scenario 7.1, the transmission power of the four nodes was 20 dBm, providing a situation where all hand-offs were carried out in the connected region. In Scenarios 7.2 and 7.3, the transmission power of the four nodes was also 20 dBm, but we placed an attenuator of 20 dB in C4. In this way, we created more link dynamics and the hand-off between R1 and R3 could take place in the transitional region.

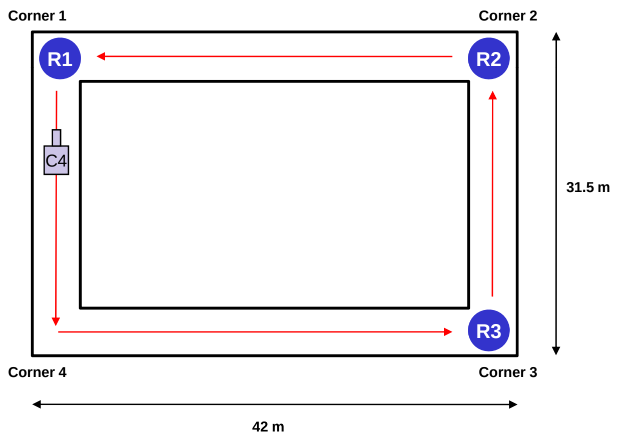


Figure 7.4: Simplified scenario of the mobile robot tele-operation in the corridors of the I3A building (Scenarios 7.1, 7.2 and 7.3)

Next we present the obtained results in Scenarios 7.1 and 7.2 to compare DoTHa with the Algorithms 7.1 and 7.2. In each experiment of these scenarios, C4 took six laps, three in each direction, starting from R1. For this reason, the expected number of hand-offs is 18, i.e., three hand-offs in every of the six laps.

7.6.1 Scenario 7.1

Data Delivery Ratio

The DDR obtained by the five strategies in Scenario 7.1 is shown in Table 7.3. In all cases, using SMA 50 filter or EWMA 1/256 filter for RSSI smoothness, the DDR was close to 100%, specifically higher than 99.97%.

Table 7.3: DDR of flow 1 (%) & DDR of flow 2 (%) in Scenario 7.1

Strategy	SMA 50	EWMA 1/256
Algorithm 7.1 ($T_l = -60$ dBm)	99.9705 & 100	99.9980 & 100
Algorithm 7.1 ($T_l = -65$ dBm)	99.9851 & 100	99.9732 & 100
Algorithm 7.1 ($T_l = -70$ dBm)	99.9961 & 100	99.9890 & 100
Algorithm 7.2	99.9981 & 100	99.9978 & 100
DoTHa	99.9976 & 100	99.9918 & 100

Number of hand-offs

Table 7.4 shows the number of hand-offs. On the one hand, when using EWMA 1/256 filter, the RSSI was properly smoothed and no ping-pong effect was produced. Although the expected number of hand-offs is 18, in some strategies there were 16 hand-offs. This situation happened when C4 changed the movement direction after three laps, which stands for no router switching. Specifically, when C4 was moving from R2 to R1 while it was attached to R2, the RSSI did not decrease enough to change to R1, even when C4 arrived at R1. We can verify this behavior in Fig. 7.5 when time was about 300 seconds, where the RSSI and the selected router as a function of time are depicted when using DoTHa. At that moment, C4 changed its direction and traveled again toward R2 in a way that no hand-off process was produced. Thus, there were two hand-offs less compared with the situation where client travels in the same direction. This was not the case for the Algorithm 7.2 because it always uses relative higher thresholds.

Table 7.4: Number of hand-offs in Scenario 7.1

Strategy	SMA 50	EWMA 1/256
Algorithm 7.1 ($T_l = -60$ dBm)	18	16
Algorithm 7.1 ($T_l = -65$ dBm)	18	16
Algorithm 7.1 ($T_l = -70$ dBm)	16	16
Algorithm 7.2	38	18
DoTHa	18	16

On the other hand, when using SMA 50 filter, only the Algorithm 7.2 presented ping-pong effect due to the use of relative higher thresholds. Specifically, it carried out double number of hand-offs than the other strategies. It should be noted that Algorithm 7.1 with $T_l = -70$ dBm presented 16 hand-offs because the RSSI did not decrease below -70 dBm when we changed the direction of rotation after three laps.

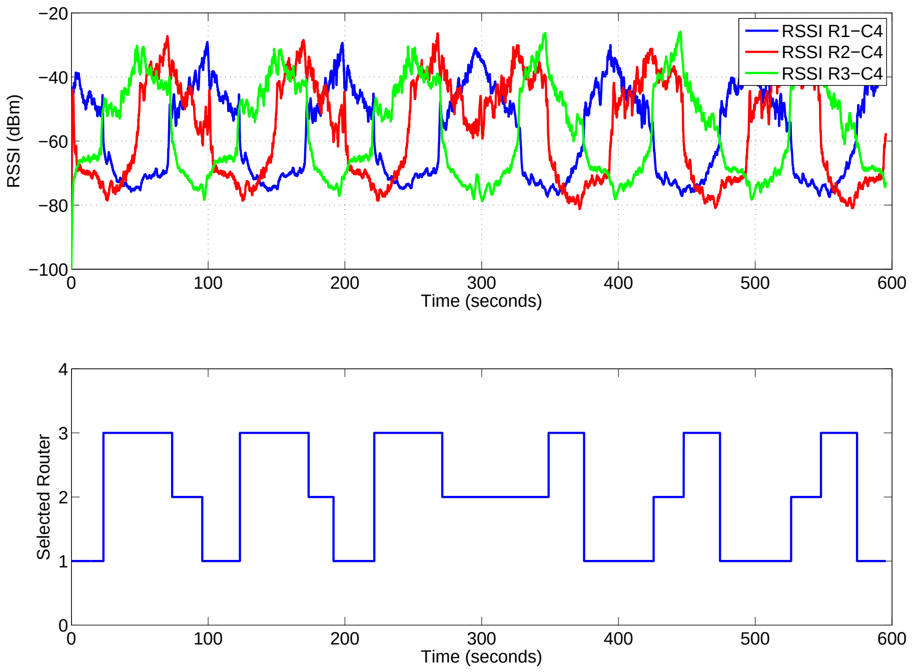


Figure 7.5: RSSI (dBm) and selected router as a function of time when using DoTHa and EWMA 1/256 filter in Scenario 7.1

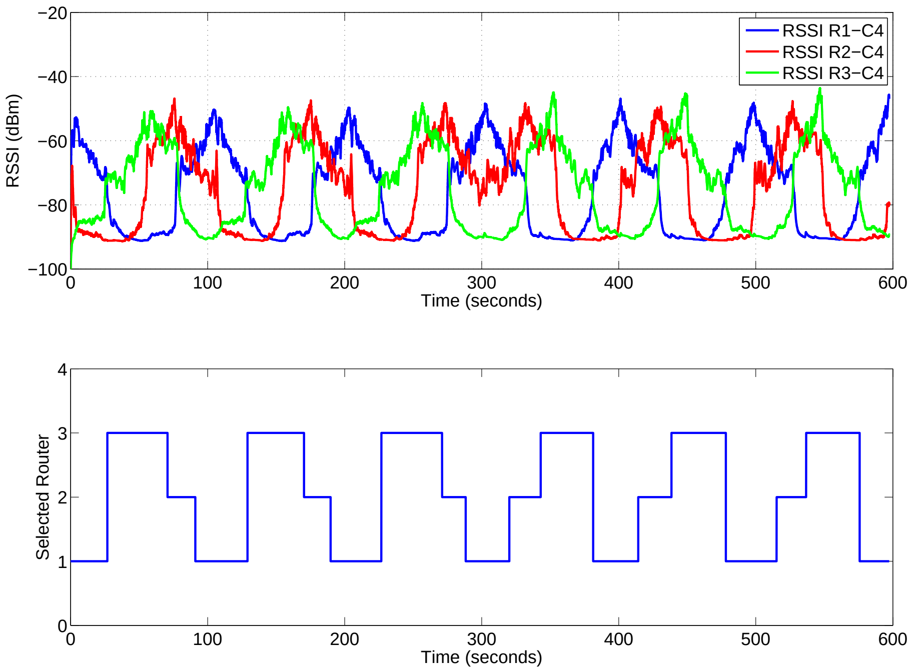


Figure 7.6: RSSI (dBm) and selected router as a function of time when using DoTHa and EWMA 1/256 filter in Scenario 7.2

Conclusions of Scenario 7.1

With a wise router deployment, all strategies achieve satisfactory DDR values. However, with a poorer smoothing filter, e.g., SMA 50 filter, Algorithm 7.2 experienced ping-pong effect. Although selecting an appropriate smoothing filter is desirable, it must be done considering the scenario, the HM value and the client speed.

7.6.2 Scenario 7.2

Data Delivery Ratio

Table 7.5 depicts the DDR obtained in Scenario 7.2. Algorithm 7.1 presented disconnections by showing DDR values between 88% and 94%. We experienced disconnections as the link between the serving router and the client entered the transitional region, and subsequently in the disconnected region, while no router provided an RSSI above $T_l + HM$, and thus hand-off was not possible. Surprisingly, Algorithm 7.1 with $T_l = -70$ dBm and SMA 50 filter did not present disconnections, thereby obtaining a DDR close to 100%. This was produced since the smoothed RSSI still presented some variability, as well as T_l and T_h were lower than in other cases. In this way, RSSI of the serving router fell below -70 dBm while RSSI of other router was higher that -65 dBm, making hand-off feasible. The price for this was ping-pong effect — see Table 7.6. DoTHa and Algorithm 7.2 achieved a DDR close to 100% because there was no disconnection in the communication thanks to the relative higher threshold that both strategies use when link quality is low.

Table 7.5: DDR of flow 1 (%) & DDR of flow 2 (%) in Scenario 7.2

Strategy	SMA 50	EWMA 1/256
Algorithm 7.1 ($\mathrm{T_l} = -60$ dBm)	92.3577 & 93.0578	88.0659 & 88.4918
Algorithm 7.1 ($\mathrm{T_l} = -65$ dBm)	93.7543 & 93.9454	91.8223 & 92.9382
Algorithm 7.1 ($\mathrm{T_l} = -70$ dBm)	99.8701 & 100	93.2166 & 93.6097
Algorithm 7.2	99.9816 & 100	99.9512 & 100
DoTHa	99.9561 & 100	99.9805 & 100

Number of hand-offs

When using SMA 50 filter, all strategies showed ping-pong effect, as observed in Table 7.6, which revealed that the RSSI smoothness was not effective. Algorithm 7.2 presented higher ping-pong effect than DoTHa, while Algorithm 7.1 presented the lowest ping-pong effect due to its fixed higher threshold. The best result was obtained for $\mathrm{T_l} = -65$ dBm, where ping-pong effect was almost nonexistent (only once). When using EWMA 1/256 filter, only Algorithm 7.2 encountered ping-pong effect, although it was much less compared with the experiments with SMA 50 filter.

Table 7.6: Number of hand-offs in Scenario 7.2

Strategy	SMA 50	EWMA 1/256
Algorithm 7.1 ($\mathrm{T_l} = -60$ dBm)	30	17
Algorithm 7.1 ($\mathrm{T_l} = -65$ dBm)	20	18
Algorithm 7.1 ($\mathrm{T_l} = -70$ dBm)	29	18
Algorithm 7.2	58	20
DoTHa	46	18

Fig. 7.6 shows the RSSI and the selected router as a function of time when using DoTHa and EWMA 1/256 filter in Scenario 7.2. Comparing these results with the same experiment in Scenario 7.1 (Fig. 7.5), we can notice that the RSSI is roughly

20 dBm lower due to the attenuator employed in C4. The number of hand-offs was 16 in Scenario 7.1 and 18 in Scenario 7.2 because of the lower RSSI received by C4 in Scenario 7.2, which caused two hand-offs at about 300 seconds. In both cases, no ping-pong effect was produced.

Conclusions of Scenario 7.2

We can see that Algorithm 7.1 does not work properly if hand-off must be carried out in the transitional region because it presents disconnections, while Algorithm 7.2 and DoTHa provide DDR close to 100% thanks to the relative higher threshold employed in low quality links. Algorithm 7.2 presents slightly higher ping-pong effect than DoTHa.

7.6.3 Final Remarks

We have shown that DoTHa works better than Algorithm 7.1 and Algorithm 7.2 when considering Scenarios 7.1 and 7.2. Although DoTHa and Algorithm 7.2 obtain similar DDR values, DoTHa presents lower ping-pong effect. We claim that it is also important to obtain an efficient hand-off process when the RSSI smoothness is inadequate, because this depends strongly on the scenario, the HM and the client speed. Moreover, if the fading is very deep, using a very strong average filter may not be the best idea since the smoothed RSSI may lose the tendency of the raw RSSI, and the hand-off may be initiated late.

7.7 Scenario 7.3 - Robot Tele-operation in the I3A Building

We carried out the tele-operation of a Pioneer 3-AT robot in the corridors of the I3A building (Fig. 7.4) by using the DoTHa algorithm with EWMA 1/256 filter for the RSSI. We also employed SRT-WICKPro, Linux and the ROS framework. As in Section 6.5.2, the tele-operation commands were generated by a PlayStation3 joystick and the laser scan data by a SICK LMS2xx laser.

The difference between Scenarios 7.2 and 7.3 is that in Scenario 7.3 a mobile robot was employed as client, whereas in Scenario 7.2 the client was a person moving with a laptop. This in turn caused that these scenarios presented two different features. First, in Scenario 7.2 synthetic data flows were generated with the properties shown in Table 7.1, thus the communication features were similar but not the same, given that Table 7.1 provides an approximate model of the real traffic. Second, the robot speed in Scenario 7.3 was lower than the client speed in Scenario 7.2. In particular, the average robot speed was about 0.44 m/s (1.59 km/h), with the experiment lasting about 1900 seconds, whereas the average speed in the previous experiments in Section 7.6 was roughly 1.4 m/s (5.04 km/h), where every experiment was executed in 600 seconds. Likewise, the robot path was less uniform than in the previous experiments, e.g., it could be stopped for some instants based on the operator commands.

7.7.1 Fault-tolerant Application

We developed a fault-tolerant tele-operation application to handle packet losses and avoid intermittent movement of the mobile robot. This application runs in the mobile robot laptop and provides an interface between WICKPro and the electric motors by sending the received tele-operation commands to the motors only if there is some continuity in the reception of these packets. Its finite-state machine is shown in Fig. 7.7, where we define two types of states: $S_i = \{S_0, S_1, ..., S_{K-1}\}$ and $S'_i = \{S'_1, S'_2, ..., S'_K\}$. In S_i, the tele-operation packets received by the mobile robot are ignored (state output = NO), whereas in S'_i the received tele-operation commands are sent to the motors (state output = YES). To change from one state to another, the application counts the number of received packets in a burst, so-called the Burst Size of Delivered Packets (BSDP). Specifically, the application must receive a tele-operation packet every period P, otherwise a tele-operation packet is considered to be missed. In S_i and S'_i, BSDP = i. BSDP ranges from 0 to K, in a way that it is neither decreased in S_0 nor increased in S'_K. The application starts from S_0 and, if BSDP becomes K, the application goes into the state S'_K and starts sending the received commands to the motors. In this state, if BSDP reaches 0, the application goes from S'_K to S_0 and the tele-operation packets stop being transferred to the motors. In our implementation, we chose $K = 12$.

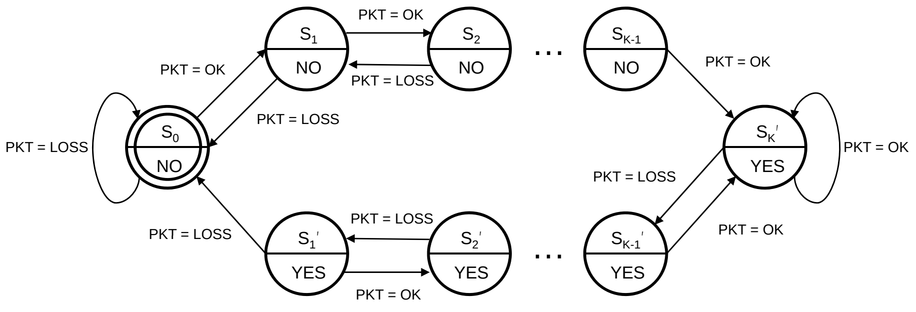


Figure 7.7: Finite-state machine of the fault-tolerant tele-operation application

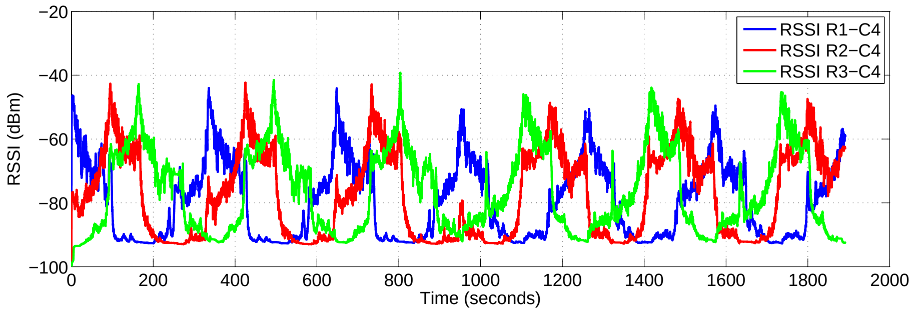
7.7.2 Results and Discussion

The robot tele-operation worked properly and its results are depicted in Table 7.7. As the robot speed was lower than in the experiments of Scenario 7.2, the RSSI smoothness was ineffective and ping-pong effect appeared, resulting in 36 hand-offs instead of 18. This fact is noticed in Fig. 7.8, where the selected router is switched back and forth several times when C4 moves away from its serving router and approaches another router.

The DDR was close to 100%, but it was lower than in Scenario 7.2 due to the client speed. The lower speed in Scenario 7.3 caused that the robot was more in regions where multi-path fading was very deep, in a way that more packets were lost at application layer. We must recall that this effect is usually mitigated at physical layer by using redundancy in time, frequency or space.

Table 7.7: DDR (flows 1 & 2) and number of hand-offs in the mobile robot tele-operation in the I3A building

DDR (%)	Number of hand-offs
99.59 & 99.75	36



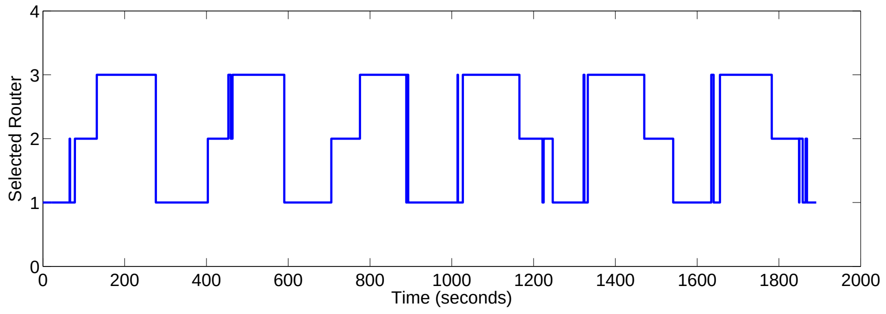


Figure 7.8: RSSI (dBm) and selected router as a function of time in the mobile robot tele-operation in the I3A building

7.8 Scenario 7.4 - Robot Tele-operation in the Somport Tunnel

We conducted another robot tele-operation in a confined and underground environment, particularly in the Somport tunnel. The Somport tunnel is an old out-of-service railway tunnel located in Canfranc (Huesca, Spain) that connects Spain and France through the Central Pyrenees. It is currently used as an auxiliary tunnel of the road tunnel and its map is depicted in Fig. 7.9, showing that it is 7.7 km long and has a horseshoe-shape cross section, approximately 5 m high and 4.65 m wide. It also has 17 lateral galleries that precisely connects the road tunnel and the auxiliary tunnel, each more than 100 m long and of the same height as the tunnel. More information about the robot tele-operation in the Somport Tunnel is provided in Appendix B.3.

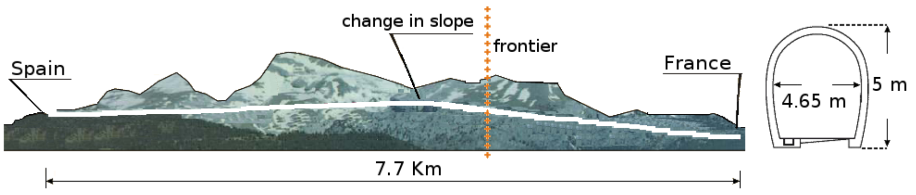


Figure 7.9: The Somport tunnel

Propagation in tunnel-like environments has special features due to the shape of the environment. To show this behavior, we present RSSI measurements within the Somport tunnel in Fig. 7.10 that exhibit a decreasing sinc-like propagation when the receiver is far enough from the transmitter. This phenomenon is only found under certain configurations in which both the transmitter and receiver are placed at specific positions in the cross section of the tunnel. These measurements are extracted from [Rizzo 15], where a detailed analysis can be found to justify this behavior and the high propagation distance achieved with standard 802.11 cards.

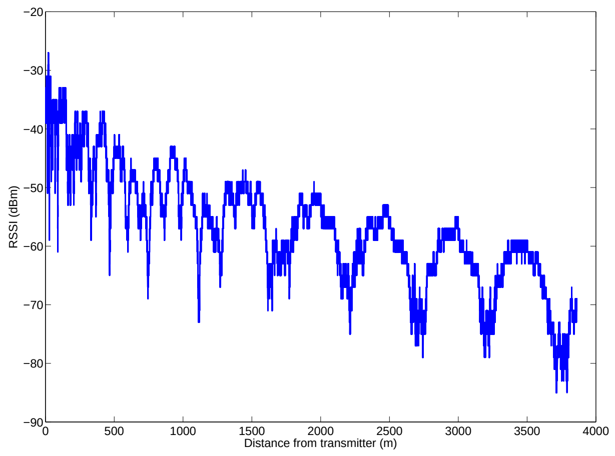


Figure 7.10: Smoothed RSSI measurements within the Somport tunnel under a specific configuration that achieves well-defined periodic fadings when the receiver is far enough from the transmitter (decreasing sinc-like propagation)

7.8.1 Experiment Description

In this experiment, we employed the same hardware and software than in the robot tele-operation carried out in the I3A building (Scenario 7.3), except that R3 had an Atheros-based 802.11 card without external antenna. R1, R2, R3 and C4 were configured with a transmission power of 1 dBm, however R3 provided lower transmission power due to the lack of external antenna. Regarding the software, we employed Linux, ROS, the fault-tolerant tele-operation application used in Scenario 7.3, SRT-WICKPro, and DoTHa with EWMA 1/256 filter for the RSSI. The simplified scenario of the experiment is shown in Fig. 7.11, where the Pioneer 3-AT robot (C4) traveled from the base station (R1) to the end of the gallery 14 and came back to the base station.

7.8.2 Results and Discussion

Table 7.8 depicts the DDR and the number of hand-offs during the robot tele-operation in the Somport tunnel. This experiment lasted about 1,740 seconds in which the mobile robot was successfully tele-operated.

The evolution of the smoothed RSSI received by C4 and its serving router at a time are shown in Fig. 7.12. Let us highlight the following four features in this figure. First, RSSI was initialized to −100 dBm, so this was the value when no packet had been monitored from a specific router. Second, the robot was stopped

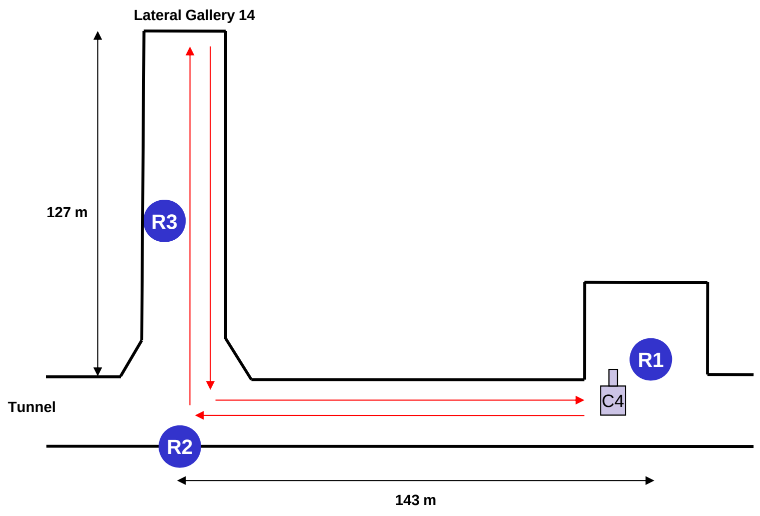


Figure 7.11: Upper view of the simplified scenario in the experiment within the Somport tunnel, where the mobile robot (C4) traveled from the base station (R1) to the end of the gallery 14 and came back to the base station

Table 7.8: DDR (flows 1 & 2) and number of hand-offs in the Somport tunnel experiment

DDR (%)	Number of hand-offs
97.45 & 98.57	4

about 300 seconds at the beginning of the experiment due to configuration issues such as waiting for the laser initialization. Third, in general R3 provided a weaker link than R1 and R2, thus C4 selected R3 as its serving router during a small percentage of time, particularly only during 16.4 seconds (time = 912 seconds in Fig. 7.12). Fourth, the RSSI received by C4 did not exhibit a decreasing sinc-like shape for two reasons: C4 was not far enough from the transmitters and we did not develop any configuration to achieve this phenomenon.

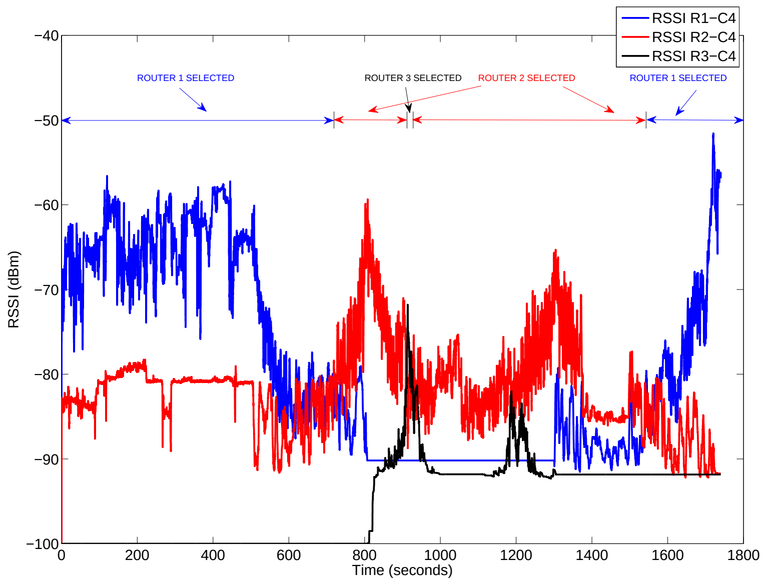


Figure 7.12: Smoothed RSSI and selected router in the Somport tunnel experiment

The delay histogram of flows 1 and 2 is shown in Fig. 7.13. Comparing this histogram with the histogram of Fig. 6.11, where the same robot tele-operation was carried out but in a laboratory experiment, we can appreciate some differences. Regarding the mean and standard deviation of the measured delays, flows 1 and 2 achieve 2.72 ± 2.60 ms and 14.69 ± 20.07 ms, respectively, while the same flows attain 7.11 ± 4.75 ms and 18.94 ± 16.56 ms in Fig. 6.11. There are two main reasons for obtaining a lower mean delay and a different distribution in this experiment. First, this experiment considers mobility in a way that the hop count was one, two or three depending on the selected router by C4, while the hop count in Chapter 6 was always three because mobility was neglected. Second, Fig. 6.11 shows an experiment where deliberate errors were introduced to achieve a PDR = 97% according to a memoryless packet loss model, whereas packet errors likely followed a bursty pattern in the Somport tunnel experiment. This fact is brought to light by analyzing the raw RSSI measured by C4 (Fig. 7.14), where there were deep fadings that leaded the received power of the serving router to almost the disconnected region.

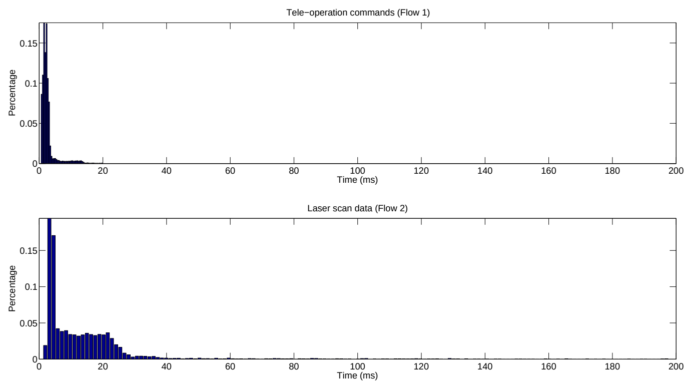


Figure 7.13: Delay histogram of flows 1 and 2 in the Somport tunnel experiment

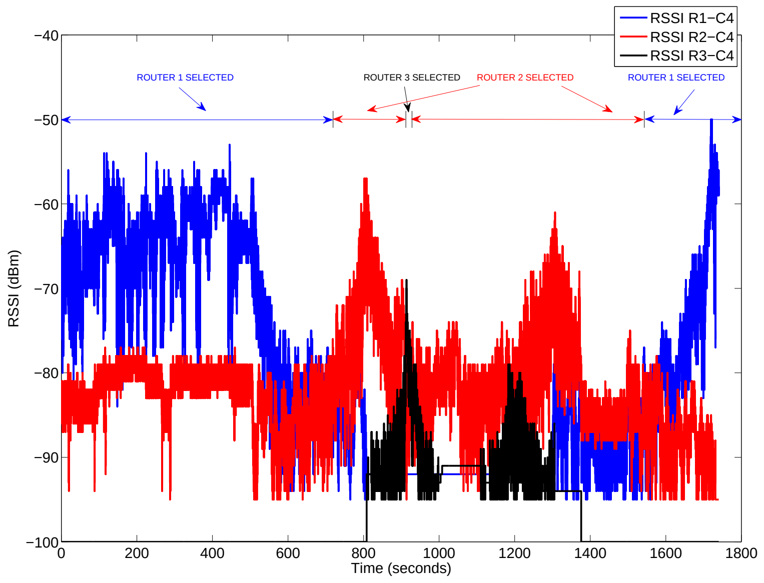


Figure 7.14: Raw RSSI and selected router in the Somport tunnel experiment

7.9 Conclusions

We designed the DoTHa hand-off algorithm to tackle mobility support in WMNs along with SRT-WICKPro. DoTHa considers double threshold level and double hysteresis margin in a way that the assumption of double threshold levels provides the opportunity of carrying out hand-off in the connected and transitional regions of a link. We tested DoTHa during a mobile robot tele-operation in two indoor environments, namely the corridors of the I3A building and the Somport tunnel, where real-time communication was supported by SRT-WICKPro and mobility by DoTHa. We showed that DoTHa handled mobility successfully when hand-off was carried out in the connected and transitional regions of a wireless link.

In other set of experiments in the I3A building, we compared DoTHa with two hand-off algorithms based on single and double hysteresis margin. The results revealed that DoTHa achieves DDR close to 100% whereas the single hysteresis-based hand-off suffers from frequent disconnections when hand-off is carried out in the transitional region, dropping DDR to 88%. The double hysteresis-based hand-off shows higher ping-pong effect than DoTHa, doubling the number of hand-offs in some scenarios.

Furthermore, we showed how token-passing protocols can be exploited to measure RSSI. We implemented a probing scheme that does not require extra overhead because the token packet is periodically passed to all nodes, particularly at least once per minor cycle.

8 Conclusions

8.1 Conclusions

Industrial real-time communication in tunnel-like environments has become a necessity due to the increasing number of activities carried out in such environments. This PhD thesis has focused on real-time communication in tunnel-like environments by using Wireless Mesh Networks with chain topologies and, as a result, the WICKPro protocol has been developed. This protocol comes in two flavors to support FRT or SRT traffic, while mobility is managed in SRT-WICKPro through the DoTHa hand-off algorithm. Next we detail the contributions of this PhD dissertation:

WICKPro The WICKPro protocol was presented in Chapter 4 in an error-free scenario. WICKPro is a MAC and network protocol build on top of the IEEE 802.11 standard that can manage real-time traffic with hard deadlines in an error-free scenario by using a token-passing scheme and a cyclic packet scheduler. The scheduling complexity must be carefully analyzed given a set of data flows to ensure real-time capabilities. WICKPro was evaluated in laboratory experiments and compared with three TDMA protocols for WMNs with chain topologies that implement spatial reuse, namely Ripple, TDS and RMP. The results showed that WICKPro sacrifices end-to-end throughput to provide flexibility and HRT traffic support.

FRT-WICKPro Chapter 5 introduced the WICKPro version to support FRT traffic in error-prone networks. FRT-WICKPro achieves FRT traffic support due to its off-line cyclic packet scheduler, which reserves time for packet transmissions and retransmissions, and its on-line packet scheduler that drops packets before congestion appears. This time reservation is possible thanks to the employed token-passing scheme. To implement the on-line packet scheduler, packet synchronization is required to let all nodes have the same temporal reference to finish minor cycles at the same time. This packet synchronization is also used to synchronize the data flow generation.

One of the most critical issues is the computation of the time margin for retransmissions. To this end, a fault-tolerant analysis is essential because packet retransmissions can jeopardize the deadlines of the supported traffic. In this protocol, the retransmission time calculation is based on the set of supported data flows, the mean PDR between one-hop neighbors and a memoryless packet loss model. Because of the complexity of this calculation, we compute the cyclic scheduling when nodes have one or two data packet transmissions per token holding, otherwise data packets have to be merged. A field experiment showed that this memoryless packet loss model may be used in certain scenarios as a good approximation, however, MATLAB simulations pointed out that the percentage of minor cycles satisfied can be overestimated if the loss pattern is bursty.

FRT-WICKPro was evaluated in laboratory and field experiments that validated the presented analytical framework for FRT traffic support. We also carried out a comparison with a priority-based token-passing protocol (RT-WMP) in an error-free network, where the throughput gain was 37.88% on average due to the lower overhead of FRT-WICKPro.

SRT-WICKPro The version of WICKPro that supports SRT traffic in error-prone networks is detailed in Chapter 6. SRT-WICKPro is an asynchronous token-passing protocol that implements a cyclic packet scheduler but does not allocate explicitly time for retransmissions. The response time of SRT-WICKPro can grow until the so-called maximum end-to-end delay EED^{max}, which is usually higher than the packet deadline (D) in SRT applications, considered equal to the packets minimum inter-arrival time (P).

We carried out MATLAB simulations and laboratory experiments using Linux and ROS, where we tele-operated a mobile robot using a WMN with chain topology. For this purpose, the complete tele-operation process from simulation to real experiment was detailed. The main conclusion of these experiments is that the higher the network utilization and the packet losses, the higher the EED^{max} must be to achieve the same throughput. However, if EED^{max} is increased, the average packet delay will also be incremented. In a given scenario, the DDR increased by 42% when EED^{max} was changed from P to $2P$, considering SRT-WICKPro with congestion control in a saturated network. Moreover, we showed that congestion control may increase network performance in our protocol.

FRT-WICKPro vs SRT-WICKPro Both versions of WICKPro exploits cyclic packet schedules but in different ways. FRT-WICKPro does not let minor cycles be overrun, which requires the use of an on-line packet scheduler and packet synchronization. This effort is worthwhile because it enables that minor cycles are analyzed independently and the design of an analytical framework is therefore simpler. The framework developed for FRT-WICKPro allocates explicitly time for retransmissions with the objective of supporting FRT traffic. To calculate the packet scheduling considering packet losses, FRT-WICKPro computes the PDR metric before the network start-up. Conversely, SRT-WICKPro supports SRT traffic, lets minor cycles be overrun and does not carry out on-line packet scheduling, packet synchronization, time reservation for retransmissions and PDR computation. In WICKPro, network restart is driven by a maximum number of retransmissions in the node that holds the token, and by a maximum time without receiving the token packet in the other nodes. FRT-WICKPro also employs another token recovery mechanism because the token is regenerated every time a minor cycle is overrun, thus token recovery will require likely more time in SRT-WICKPro than in FRT-WICKPro. We chose SRT-WICKPro to support mobility because it is simpler than FRT-WICKPro and enough to support SRT traffic.

DoTHa The DoTHa hand-off algorithm that manages mobility in WMNs along with SRT-WICKPro was presented in Chapter 7. DoTHa considers double threshold level and double hysteresis margin in a way that the assumption of double threshold levels provides the opportunity of carrying out hand-off in the connected and transitional regions of a link. We tested DoTHa during a mobile robot tele-operation in two indoor environments, namely the corridors of the I3A building and the Somport tunnel, where real-time communication was supported by SRT-WICKPro and mobility by DoTHa. We showed that DoTHa handled mobility successfully when hand-off was carried out in the connected and transitional regions of a wireless link.

In other set of experiments in the I3A building, we compared DoTHa with two hand-off algorithms based on single and double hysteresis margin. The results revealed that DoTHa achieves DDR close to 100% whereas the single hysteresis-based hand-off suffers from frequent disconnections when hand-off is carried out in the transitional region, dropping DDR to 88%. The double hysteresis-based hand-off shows higher ping-pong effect than DoTHa, doubling the number of hand-offs in some scenarios.

Furthermore, we showed how token-passing protocols can be exploited to measure RSSI. We implemented a probing scheme that does not require extra overhead because the token packet is periodically passed to all nodes, particularly at least once per minor cycle.

Final Remarks In short, the main purpose of this PhD thesis was to develop wireless multi-hop protocols with real-time capabilities or, in other words, with a deterministic behavior. We conclude that WICKPro presents a good trade-off between complexity and efficiency in small-scale networks in which spatial reuse is impossible or limited. This statement is supported by the tests carried out to compare WICKPro with TDMA protocols and RT-WMP.

The use of IEEE 802.11 cards has been fully satisfactory, however, real-time behavior could be jeopardized because of external interferences, specially in places with an elevated number of interfering IEEE 802.11 networks. Even though the IEEE 802.11 standard mitigates this problem by operating in several channels and using low transmission power, interferences from external IEEE 802.11 networks may become uncontrollable in crowded places. Conversely, if the environment is partially or totally controlled, as in the case of factories and tunnels where an authority can restrict access, real-time communication with IEEE 802.11 technology appears feasible from the point of view of external traffic.

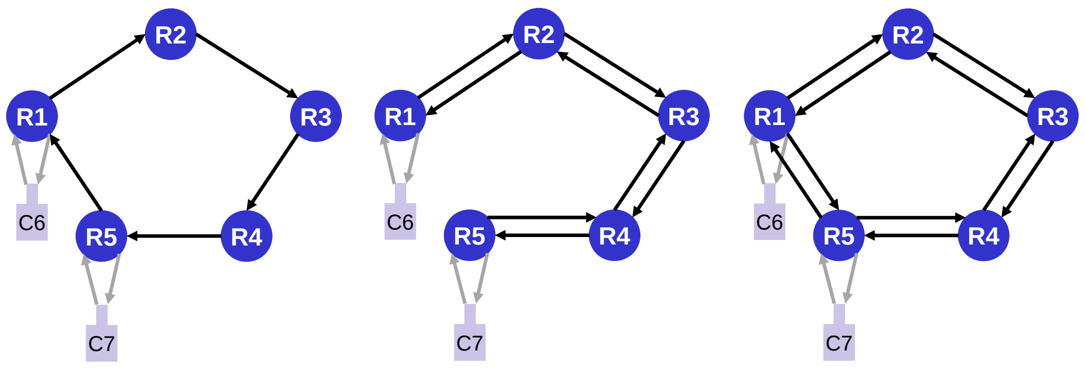
8.2 Future Work

In this section, we detail several ideas that arise logically as a continuation of the presented work.

8.2.1 Generalization of the Network Topology

Although using the WICKPro protocol is in itself useful in tunnel-like environments, adapting WICKPro to work in arbitrary topologies is a very interesting topic. For this task, the topology management procedure, the routing algorithm and the packet scheduler should be modified. First, mesh routers would not always form a chain and therefore mesh routers should be statically configured or provided with self-configuration capabilities. Second, the routing algorithm would be changed to include a more sophisticated metric than the hop count. Third, the packet scheduler should manage arbitrary topologies. The token path would be calculated based on the network topology and the set of supported data flows. Another option would be to build a logical ring to pass the token packet.

Next we detail the option of modifying WICKPro to work in ring networks because it could be easily adapted. Let us illustrate this with the example of Fig. 8.1 where token-ring and token-chain schemes are implemented in a WMN with five routers and two clients. In Fig. 8.1a, we can observe that R1 and R5 can communicate directly in a ring network, whereas, in a chain network, the token must come back from R5 to R1 by retracing the path, as shown in Fig. 8.1b. Therefore, WICKPro could be readily modified to support ring topologies, as depicted in Fig. 8.1c. Unlike protocols such as WTRP that always carry out token rotations in the same direction, this approach could define token rotations in both directions, which reduces the hop count and the delay of the end-to-end communication.



(a) Token ring **(b)** Token chain (WICKPro) **(c)** Token ring (WICKPro)

Figure 8.1: Example of token ring and token chain rotations that shows their similarity

8.2.2 Increasing the Protocol Scalability

WICKPro would be used in large-scale networks by using multiple channels and applying spatial reuse. A possible design is depicted in Fig. 8.2. In this architecture, each sub-network consists of three routers that share a radio channel and a token packet. Routers R3, R5 and R7 belongs to two sub-networks and consequently they must be able to transmit concurrently in two different channels. Interferences between sub-networks with the same working channel are avoided assuming that routers are separated by a distance equal to the transmission range and the interference range is 2.2 times the transmission range, as typical in IEEE 802.11 networks. It should also be noted that a framework should be developed to provide real-time communication.

If we considered more than three channels, each sub-network would consist of two routers that share a radio channel and a token packet.

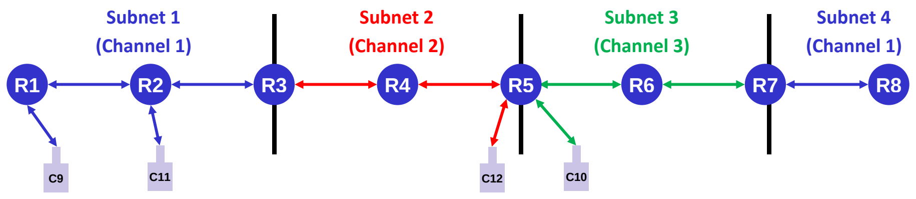


Figure 8.2: Future network architecture with spatial reuse and multiple channels

8.2.3 Improvements in FRT-WICKPro

FRT traffic support is not an easy task, specially in wireless multi-hop networks due to the problems stated in Section 2.3. For this reason, we made several assumptions that could, however, be relaxed in future. For instance, the analytical framework would consider the following: (1) the fulfillment of other metrics different than the DDR such as the (m,k)-firm constraint, (2) the use of a bursty packet loss model, and (3) the "solidarity" between minor cycles. The latter relates to let minor cycles be overrun and try to compensate the overrun of one minor cycle with the subsequent minor cycles, as SRT-WICKPro does. One option would be to carry out a feasibility analysis in the major cycle as a whole. In this case, the burstiness of wireless channels would be managed better and the DDR would be higher. The downside is the complexity increase.

Other interesting issue would be the mobility management with FRT traffic support, although this is really challenging due to the hostility of the wireless medium and the need to satisfy a QoS metric such as the DDR at the same time that mobility is being supported. This would also involve the PDR computation update meanwhile communications are being supported.

8.2.4 Improvements in SRT-WICKPro

In this work, we developed a preliminary formal analysis in SRT-WICKPro, so a complete framework would be desirable. This would likely lead to implement a congestion control algorithm to maximize the performance of the protocol. The use of packet synchronization would also help increase the obtained performance. Likewise, using more complex utility functions would be interesting in certain applications.

8.2.5 Improvements in WICKPro

Merging both WICKPro versions would be of interest to support FRT and SRT traffic simultaneously, as well as the incorporation of best-effort and aperiodic traffic support.

For a better working of WICKPro in places with other IEEE 802.11 networks, we would develop strategies to mitigate the effect of external traffic.

The use of more advanced IEEE 802.11 versions would provide higher performance. For example, we would use higher bit rates in our current IEEE 802.11 cards and also employ the newest IEEE 802.11 standards, namely IEEE 802.11n and IEEE 802.11ac.

Regarding the comparison of WICKPro with other protocols, it would be interesting the comparison between WICKPro and TDMA protocols for WMNs with chain topologies in an error-prone scenario (Chapter 4 showed this comparison but in an error-free scenario). It would be also constructive to compare WICKPro and IEEE 802.11p, the standard for wireless access in vehicular environments. Actually, the adaptation of WICKPro to vehicular environments would be a natural step given the topology of vehicular networks, probably including a network architecture that provides scalability as shown in Fig. 8.2.

8.2.6 Improvements in the DoTHa Hand-off Algorithm

We would carry out the following four improvements in DoTHa: (1) improve RSSI smoothing, e.g., by using trend estimators; (2) tune the hysteresis margin value more accurately, since increasing the hysteresis margin avoids ping-pong effect, but enlarges hand-off latency; (3) consider more metrics besides RSSI measurements to achieve more accurate hand-off decisions, for example, we could evaluate the PDR to take into account external interferences; and (4) carry out a specific design for tunnel-like environments considering the decreasing sinc-like propagation.

8.2.7 Simulation in Standard Platforms

We implemented a discrete-event system in MATLAB to simulate the WICKPro protocol. Nevertheless, it would be interesting to employ a standard simulator to compare easier WICKPro with other protocols. This would also enable to use a standard mobility framework to design mobility-related strategies, namely hand-off algorithms.

Simulation is one of the workhorses in communication protocol design because of the accuracy when simulating critical issues such as channel and mobility. At present, one of the most widely used network simulators is OMNET++ [OMNet++ 16], although there are other simulators available. Likewise, other authors employ MATLAB-OMNET++ co-simulation to increase the simulation accuracy [Zhang 10].

8.3 Conclusiones (in Spanish)

Las comunicaciones industriales en tiempo real en entornos tipo túnel han llegado a ser una necesidad debido al creciente número de actividades llevadas a cabo en dichos ambientes. Esta tesis doctoral se ha centrado en comunicaciones en tiempo real en entornos tipo túnel utilizando en redes WMN con topología en cadena y, como resultado, se ha desarrollado el protocolo WICKPro. En concreto, se han desarrollado dos versiones de este protocolo para soportar tráfico FRT o SRT. Para el manejo de la movilidad en SRT-WICKPro se ha diseñado el algoritmo de *hand-off* DoTHa. A continuación detallamos las contribuciones de esta tesis doctoral:

WICKPro El protocolo WICKPro se presentó en el Capítulo 4 en un escenario libre de errores. WICKPro es un protocolo MAC y de nivel de red diseñado sobre el estándar IEEE 802.11 que soporta tráfico de tiempo real crítico en un escenario sin errores utilizando un esquema de paso de testigo y un planificador de paquetes cíclico. La complejidad de la planificación debe ser analizada cuidadosamente para un conjunto dado de flujos de datos con el fin de asegurar las funcionalidades de tiempo real. WICKPro se evaluó en experimentos de laboratorio y se comparó con tres protocolos TDMA que implementan reúso espacial y que fueron especialmente diseñados para redes WMN con topología en cadena: Ripple, TDS y RMP. Los resultados mostraron que WICKPro sacrifica *throughput* extremo a extremo para proporcionar flexibilidad y soporte de tráfico HRT.

FRT-WICKPro En el Capítulo 5 se presentó la versión de WICKPro para soporte de tráfico FRT en redes propensas a errores. FRT-WICKPro consigue soportar tráfico FRT gracias a su planificador de paquetes cíclico *off-line*, que reserva tiempo para transmisiones y retransmisiones de paquetes, y a su planificador de paquetes *on-line*, que descarta paquetes para evitar la congestión en la red. Esta reserva de tiempo es posible gracias al esquema de paso de testigo empleado. Para implementar el planificador de paquetes *on-line*, se requiere cierto nivel de sincronismo para que todos los nodos tengan la misma referencia temporal y terminen los ciclos secundarios al mismo tiempo. Dicha sincronización también es utilizada para sincronizar la generación de los flujos de datos.

Uno de los aspectos más críticos es el cálculo del margen temporal para retransmisiones, puesto que las retransmisiones de paquetes pueden poner en peligro los plazos de entrega del tráfico soportado. Por ello, es esencial realizar un análisis tolerante a fallos. En este protocolo, el cálculo del tiempo para retransmisiones depende de los flujos de datos soportados, la PDR media entre los vecinos a un salto y un modelo de pérdida de paquetes sin memoria. Debido a la complejidad de este cálculo, obtenemos la planificación cíclica cuando los nodos tienen uno o dos paquetes a transmitir en cada posesión de testigo; en caso de tener más paquetes a transmitir, éstos deben ser unidos. El modelo de pérdida de paquetes sin memoria fue satisfactoriamente utilizado durante un experimento de campo, sin embargo, simulaciones en MATLAB señalaron que el porcentaje de ciclos secundarios no sobrepasados puede ser sobrestimado si el modelo de pérdidas es a ráfagas.

El marco analítico para el soporte de tráfico FRT fue validado en experimentos de laboratorio y de campo. Asimismo, comparamos FRT-WICKPro en una red libre de errores con un protocolo de paso de testigo que utiliza un planificador de paquetes basado en prioridades (RT-WMP), donde FRT-WICKPro logró una ganancia media en *throughput* del 37.88% debido a su menor información de control.

SRT-WICKPro La versión de WICKPro para soporte de tráfico SRT en redes propensas a errores fue descrita en el Capítulo 6. SRT-WICKPro es un protocolo de paso de testigo asíncrono que implementa un planificador de paquetes cíclico, pero no reserva explícitamente tiempo para retransmisiones. El tiempo de respuesta de SRT-WICKPro puede crecer hasta el denominado retardo máximo extremo a extremo EED^{max}, que suele ser mayor que el plazo de respuesta (D) en aplicaciones SRT, considerado igual al tiempo mínimo entre llegadas de paquetes (P).

Para testar SRT-WICKPro, realizamos simulaciones en MATLAB y experimentos de laboratorio utilizando Linux y ROS, en los cuales teleoperamos un robot móvil utilizando una red WMN con topología en cadena. Por ello, detallamos el proceso de teleoperación desde la simulación hasta los experimentos reales. La principal conclusión es que cuanto mayor es la utilización de la red y las pérdidas de paquetes, mayor debe ser EED^{max} para conseguir el mismo *throughput*. Sin embargo, al incrementar EED^{max}, el retardo medio de los paquetes también se ve incrementado. En un escenario concreto, la DDR se incrementó un 42% cuando EED^{max} fue aumentado de P a $2P$, considerando SRT-WICKPro con control de congestión en una red saturada. Además, mostramos que el control de congestión puede incrementar el rendimiento de nuestro protocolo.

FRT-WICKPro versus SRT-WICKPro Ambas versiones de WICKPro emplean planificaciones cíclicas de paquetes pero de diferentes maneras. FRT-WICKPro no permite que los ciclos secundarios sean sobrepasados, lo cual requiere utilizar un planificador de paquetes *on-line* y sincronismo entre nodos. Este esfuerzo merece la pena porque permite que los ciclos secundarios sean analizados independientemente, facilitando el diseño de un marco analítico. El marco desarrollado para FRT-WICKPro reserva explícitamente tiempo para retransmisiones con el objetivo de soportar tráfico FRT. Para calcular la planificación de paquetes teniendo en cuenta pérdidas de paquetes, FRT-WICKPro mide la PDR antes de que la red empiece a funcionar. Por el contrario, SRT-WICKPro soporta tráfico SRT, permite que los ciclos secundarios sean excedidos y no lleva a cabo planificación de paquetes *on-line*, sincronización entre nodos, reserva de tiempo para retransmisiones ni la medición de la PDR. En WICKPro, el reinicio de la red es activado si se sobrepasa un número máximo de retransmisiones en el nodo que tiene el testigo, o un tiempo máximo sin recibir el testigo en el resto de nodos. FRT-WICKPro también emplea otro método de recuperación del testigo porque el testigo es regenerado cada vez que un ciclo secundario es sobrepasado, por lo que la recuperación del testigo requerirá seguramente mayor tiempo en SRT-WICKPro que en FRT-WICKPro. SRT-WICKPro fue escogido para soportar movilidad porque es más simple que FRT-WICKPro y suficiente para soportar tráfico SRT.

DoTHa El algoritmo de *hand-off* DoTHa que gestiona movilidad en redes WMN junto con SRT-WICKPro fue presentado en el Capítulo 7. DoTHa considera doble umbral y doble margen de histéresis, de forma que la utilización de doble umbral permite realizar *hand-off* en la región conectada y en la región de transición de un enlace. DoTHa fue testado durante la teleoperación de un robot móvil en dos escenarios de interior, en concreto en los pasillos del edificio del I3A y el túnel del Somport, donde el soporte de comunicación en tiempo real fue realizado por medio de SRT-WICKPro y el de movilidad mediante el algoritmo DoTHa. En estos experimentos, mostramos que DoTHa manejó la movilidad satisfactoriamente cuando el *hand-off* fue realizado en la región conectada y en la región de transición de los enlaces inalámbricos. Además, realizamos otra serie de experimentos en el edificio del I3A donde comparamos DoTHa con otros dos algoritmos de *hand-off* basados en simple y doble margen de histéresis. Los resultados revelaron que DoTHa obtiene una DDR cercana al 100% mientras que el algoritmo basado en simple margen de histéresis sufre desconexiones frecuentes cuando el *hand-off* debe realizarse en la región de transición, descendiendo la DDR al 88%. Por su parte, el algoritmo basado en doble margen de histéresis produce mayor efecto ping-pong que DoTHa, doblando el número de *hand-offs* en algunos casos.

Además, mostramos cómo los protocolos de paso de testigo puede ser explotados para medir la RSSI. En particular, implementamos un esquema de medición continua (*probing*) que no requiere información de control adicional porque el testigo es pasado periódicamente a todos los nodos de la red, en concreto al menos una vez en cada ciclo secundario.

Reflexión Final En resumen, el objetivo principal de esta tesis doctoral era desarrollar protocolos para redes inalámbricas multisalto con requisitos de tiempo real, es decir, con un comportamiento determinista. Podemos concluir que WICKPro presenta un buen compromiso entre complejidad y eficiencia en redes de pequeña escala en las que el reúso espacial es imposible o limitado. Esta afirmación está apoyada por los experimentos realizados que compararan WICKPro con protocolos TDMA y con RT-WMP.

La utilización de tarjetas IEEE 802.11 ha sido plenamente satisfactoria, sin embargo, el comportamiento de tiempo real podría peligrar debido a inferencias externas, especialmente en lugares con un gran número de redes interferentes. Si bien este problema es mitigado por el propio estándar al operar en varios canales y utilizar baja potencia de transmisión, las interferencias de redes IEEE 802.11 externas pueden llegar a ser incontrolables en lugares muy concurridos. Por el contrario, si el ambiente está parcial o totalmente controlado, como puede ser el caso de fábricas y túneles donde una autoridad puede restringir el acceso, la comunicación en tiempo real con redes IEEE 802.11 parece factible desde el punto de vista del tráfico externo.

8.4 Trabajo Futuro (in Spanish)

En esta secion detallamos algunas ideas que surgen de forma lógica como continuación del trabajo presentado.

8.4.1 Generalización de la Topología de Red

Aunque el protocolo WICKPro es útil en sí mismo en entornos tipo túnel, adaptar WICKPro para que trabaje en cualquier topología es un tema muy interesante. Para ello, el procedimiento de gestión de la topología, el algoritmo de encaminamiento y el planificador de paquetes deberían ser modificados. En primer lugar, los *routers* podrían no tener una topología en cadena y, por tanto, deberían ser configurados estáticamente o ser dotados de funcionalidades de autoconfiguración. En segundo lugar, el algoritmo de encaminamiento podría cambiarse para incluir una métrica más sofisticada que el número de saltos. Y en tercer lugar, el planificador de paquetes debería manejar topologías aleatorias. El camino de rotación del testigo podría calcularse en función de la topología de la red y de los flujos soportados. Otra opción podría ser construir un anillo lógico para pasar el testigo.

A continuación detallamos la opción de modificar WICKPro para que trabaje con topologías en anillo, puesto que puede ser fácilmente adaptado. Vamos a ilustrar esto con el ejemplo de la Fig. 8.1, donde los esquemas de paso de testigo en anillo y en cadena son implementados en una red WMN con cinco *routers* y dos clientes. En la Fig. 8.1a, podemos observar que R1 y R5 pueden comunicarse directamente en una red en anillo, mientras que, en una red en cadena, el testigo debe volver de R5 a R1 deshaciendo el camino, como se muestra en la Fig. 8.1b. Por tanto, WICKPro podría ser adapado para soportar topologías en anillo de forma relativamente sencilla, como se ilustra en la Fig. 8.1c. Al contrario que protocolos como WTRP que siempre llevan a cabo rotaciones en la misma dirección, esta propuesta podría definir rotaciones de testigo en ambas direcciones, lo cual reduce el número de saltos y el retardo extremo a extremo.

8.4.2 Incrementando la Escalabilidad del Protocolo

WICKPro podría utilizarse en redes de gran tamaño utilizando múltiples canales y aplicando reúso espacial. Un posible diseño puede verse en la Fig. 8.2. En esta arquitectura, casa subred consiste en tres *routers* que comparten un canal radio y un paquete de testigo. Los *routers* R3, R5 y R7 pertenecen a dos subredes y, por tanto, deben ser capaces de transmitir concurrentemente en dos canales diferentes. Las interferencias entre subredes con el mismo canal son evitadas asumiendo que los *routers* están separados una distancia igual al rango de transmisión y que el rango de interferencia es 2.2 veces el rango de transmisión, como es típico en redes IEEE 802.11. También debemos destacar que sería necesario desarrollar un marco analítico para proporcionar comunicación en tiempo real.

Si consideráramos más de tres canales, cada subred podría consistir en dos *routers* que compartirían un canal radio y un paquete de testigo.

8.4.3 Mejoras en FRT-WICKPro

Soportar tráfico FRT no es una tarea sencilla, especialmente en redes inalámbricas multisalto debido a los problemas mencionados en la Sección 2.3. Por esta razón, realizamos ciertos supuestos que podrían ser relajados en el futuro. Por ejemplo, el marco analítico podría considerar lo siguiente: (1) el cumplimiento de otra métrica diferente a la DDR como la métrica (m,k)-firm, (2) la utilización de un modelo de pérdida de paquetes a ráfagas, y (3) la "solidaridad" entre ciclos secundarios. Esto último se refiere a permitir sobrepasar los ciclos secundarios para intentar compensar el incumplimiento de un ciclo secundario con los ciclos secundarios subsiguientes, como hace SRT-WICKPro. Una opción podría ser realizar un análisis de planificabilidad teniendo en cuenta el hiperperiodo globalmente como un todo. En este caso, el comportamiento a ráfagas de los canales inalámbricos podría ser manejado mejor y la DDR podría ser mayor. La contrapartida es que la complejidad aumentaría.

Otro tema interesante podría ser la gestión de la movilidad con soporte de tráfico FRT, aunque esto es realmente desafiante debido a la hostilidad del medio inalámbrico y a tener que asegurar el cumplimiento de una métrica de calidad de servicio como la DDR mientras se soporta la movilidad. Este asunto involucraría además la actualización de la PDR al mismo tiempo que la red está soportando comunicaciones.

8.4.4 Mejoras en SRT-WICKPro

En este trabajo hemos presentamos un análisis formal preliminar de SRT-WICKPro, así que sería deseable desarrollar un marco analítico completo. Esto seguramente nos llevaría a implementar un mecanismo de control de congestión para maximizar el rendimiento del protocolo. Dicho rendimiento podría incrementarse también con el uso de sincronismo entre nodos. Asimismo, la utilización de funciones de utilidad más complejas podría ser interesante en ciertas aplicaciones.

8.4.5 Mejoras en WICKPro

Unir ambas versiones de WICKPro podría ser interesante para soportar tráfico FRT y SRT simultáneamente, así como la incorporación de tráfico de mejor esfuerzo (*best-effort*) y tráfico aperiódico.

Para mejorar el funcionamiento de WICKPro en lugares con otras redes IEEE 802.11, podríamos desarrollar estrategias para mitigar el efecto de tráfico externo.

El uso de versiones más avanzadas de IEEE 802.11 podría proporcionar mayor rendimiento. Por ejemplo, podríamos utilizar mayores tasas de bit en nuestras actuales tarjetas IEEE 802.11, así como emplear los estándares IEEE 802.11 más recientes, tales como IEEE 802.11n y IEEE 802.11ac.

Respecto a la comparación de WICKPro con otros protocolos, podría ser interesante una comparación en escenarios propensos a errores entre WICKPro y protocolos TDMA en redes WMN con topología en cadena (en el Capítulo 4 se realizó esta comparación en una red libre de errores). También podría ser constructivo comparar WICKPro y IEEE 802.11p, el estándar para accesso inalámbrico en redes vehiculares. De hecho, la adaptación de WICKPro a ambientes vehiculares podría ser también un paso natural dada la topología de dichas redes vehiculares, probablemente incluyendo una arquitectura de red que proporcionara escalabilidad, como la mostrada en la Fig. 8.2.

8.4.6 Mejoras en el Algoritmo de *Hand-off* DoTHa

En el algoritmo DoTHa, podríamos trabajar en los siguientes cuatro aspectos: (1) mejorar el suavizado de la RSSI, por ejemplo, utilizando estimadores de tendencia; (2) ajustar el valor del margen de histéresis de forma más precisa, ya que incrementar el margen de histéresis evita el efecto ping-pong pero alarga la latencia del *hand-off*; (3) considerar más métricas aparte de la RSSI para conseguir más precisión en las decisiones de *hand-off*, por ejemplo, podríamos evaluar la PDR para tener en cuenta las interferencias externas; y (4) realizar un diseño específico para entornos tipo túnel considerando la propagación decreciente tipo sinc.

8.4.7 Simulación en Plataformas Estándar

Aunque en este trabajo ya hemos simulado WICKPro en MATLAB como un sistema de eventos discreto, podría ser también de interés utilizar simuladores estándar para comparar WICKPro con otros protocolos más fácilmente. Esto permitiría además la utilización de bibliotecas estándar de movilidad para diseñar estrategias de *hand-off*.

La simulación es uno de los caballos de batalla en el diseño de protocolo de comunicaciones debido a la exactitud en la simulación de aspectos críticos como el canal y la movilidad. Uno de los simuladores más ampliamente utilizados actualmente es OMNET++ [OMNet++ 16], aunque hay otros simuladores disponibles. Asimismo, otros autores emplean cosimulación entre MATLAB y OMNET++ para incrementar la exactitud en las simulaciones [Zhang 10].

A Packet Definition in WICKPro

This appendix describes the packets employed by WICKPro to implement the functionalities presented in Chapters 4-7. WICKPro distinguishes four categories of packets, as depicted in Table A.1: control, data, piggyback and probe packets. Likewise, WICKPro defines ten types of individual packets: Regular Token, Flow Establishment, Flow Removal, Hand-off, NACK, Drop, Data1, Data2, Data3 and Data4. Moreover, piggyback packets are created by joining one of the four forward token-passing packets and one of the four data packets. The probe packets are actually a special type of piggyback packets. The size of every packet type is summarized in Table A.2. It should also be noted that all nodes in the network are assigned a 1-byte address called WICKPro address or simply address.

Table A.1: WICKPro packets

Category	Subcategory	Name	Function
Control	Forward token-passing	Regular token	Regular token pass
		Establishment	Admit a new data flow
		Removal	Tear down a currently supported data flow
		Hand-off	Carry out a hand-off process
	Backward token-passing	NACK	Request data packet retransmissions
		Drop	ACK packet used under certain circumstances
Data		Data1-Data4	Carry data packets
Piggyback		Piggyback	Forward token-passing packet + data packet
Probe		Probe1-Probe5	Measure the PDR

A.1 Control Packets

WICKPro defines six control packets that can be classified in the following three subcategories: (i) The forward token-passing packets (regular token, establishment, removal and hand-off packets) pass the token according to the token path calculated by the scheduler, (ii) the backward token-passing packet (NACK packet) carries out a token pass in the opposite direction of the token path, and (iii) the drop packet does not stand for a token pass. The regular token packet authorizes its holder to transmit, while the other forward and backward token packets have this feature but add extra functionalities, as shown in Table A.1. It should be highlighted that we employ the terms token packet and forward token packet interchangeably.

Table A.2: Size of WICKPro packets

Category	Name	Size (Bytes)
Control	Regular token	9
	Establishment	13
	Removal	10
	Hand-off	11
	NACK	9 + 2*NumPacketsReceived
	Drop	5
Data	Data1-Data4	5 + Data
Piggyback	Piggyback + Regular token	11 + Data
	Piggyback + Establishment	15 + Data
	Piggyback + Removal	12 + Data
	Piggyback + Hand-off	13 + Data
Probe	Probe1-Probe5	10 + 5*NumTxRxNotSch + DummyData

Regular Token Packet

The fields of the regular token packet are depicted in Table A.3:

- *Src* is the source address of the packet.
- *Dst* is the destination address of the packet.
- *Type* is the packet type.
- *Serial* serves to handle token packet loss and duplication and it is therefore incremented after a token packet transmission but remains unchanged after a retransmission.
- *NumMiC* determines the number of minor cycle and is only modified by the token master.
- *Notification* is used by the token master to report about changes in the network: (i) changes in the scheduling due to admission of a new data flow, removal of a supported data flow or a hand-off process, and (ii) changes in the states of the protocol (inactive, topology discovery, PDR calculation and active, as stated in Section 4.2).
- *FreeTimeMiC* indicates the free time in the current minor cycle, which is initialized by the token master at the beginning of every minor cycle and decreased every time a transmission (retransmissions included) is carried out in the network. Note that it is only employed in FRT-WICKPro.
- *NumDataPktsSent* is the number of data packets sent from the source node of the token packet to the destination node within a token pass, i.e., in a single token holding.

More information can be found in Section 4.6.2 about the *Serial* and *NumDataPktsSent* fields, and in Section 5.2 about the *FreeTimeMiC* field. Moreover, it should be noted that the first three fields are actually common to all WICKPro packets: *Src*, *Dst* and *Type*.

Table A.3: Regular token packet in WICKPro. The size of each field is indicated below (in bytes)

Src	Dst	Type	Serial	NumMiC	Notification	FreeTimeMiC	NumDataPktsSent
1	1	1	1	1	1	2	1

Flow Establishment Packet

Table A.4 shows the establishment packet fields. It adds four fields to the fields of the regular token packet:

- *Label* specifies the label of the new flow whose admission has been requested. It is selected among the unused labels by the source node that requests the admission. After the flow is admitted, all nodes know the relationship between this label and the source, destination and type of the flow.
- *FlowSrc* is the source address of the new flow.
- *FlowDst* is the destination address of the new flow.
- *FlowType* is the flow type of the new flow. The flow type is equivalent to the data type: Data1, Data2, Data3 or Data4.

Table A.4: Establishment packet in WICKPro. The size of each field is indicated below (in bytes)

Src	Dst	Type	Serial	NumMiC	Notification	FreeTimeMiC	NumDataPktsSent
1	1	1	1	1	1	2	1
Label	FlowSrc	FlowDst	FlowType				
1	1	1	1				

Flow Removal Packet

Table A.5 exhibits the removal packet fields. It adds one extra field to the fields of the regular token packet, particularly a field called *Label* that gives the label of the flow that must be removed.

Table A.5: Removal packet in WICKPro. Field size is indicated below (in bytes)

Src	Dst	Type	Serial	NumMiC	Notification	FreeTimeMiC	NumDataPktsSent	Label
1	1	1	1	1	1	2	1	1

Hand-off Packet

The hand-off packet has two fields more than the regular token packet, as shown in Table A.6:

- *Client* defines the mobile node involved in the hand-off process.
- *Router* indicates the new serving router of the client in question.

It should be noted that this packet corresponds to the implementation of the DoTHa hand-off algorithm in SRT-WICKPro.

Table A.6: Hand-off packet in WICKPro. Field size is indicated below (in bytes)

Src	Dst	Type	Serial	NumMiC	Notification	FreeTimeMiC	NumDataPktsSent	Client	Router
1	1	1	1	1	1	2	1	1	1

NACK Packet

The NACK packet has all the fields of the regular token packet except the *Num-DataPktsSent* field (Table A.7). Instead of this field, it defines the *NumDataPktsRx* field, which indicates the number of data packets that a node received correctly after the reception of a token packet. The rest of the fields details these data packets by means of their label (*Label*) and number of data packet (*Num*). More information can be found in Section 4.6.2 about the working of the NACK packet to request data packet retransmissions.

Table A.7: NACK packet in WICKPro. Field size is indicated below (in bytes)

Src	Dst	Type	Serial	NumMiC	Notification	FreeTimeMiC
1	1	1	1	1	1	2
NumDataPktsRx	Label(0)	Num(0)	...		Label(NumDataPktsRx-1)	Num(NumDataPktsRx-1)
1	1	1			1	1

Drop Packet

The drop packet is implemented by error-free WICKPro (Chapter 4) and FRT-WICKPro (Chapter 5) because they fulfill strictly minor cycles. Particularly, it is only sent by the token master in the following situation. If a node k sends the token packet to the master node and this is going to keep the token packet during a time higher than the stipulated time-out, the token master sends a drop packet to node k to prevent node k from retransmitting the token packet to the master node. An example is shown in Fig. 4.11. The drop packet fields can be seen in Table A.8.

Table A.8: Drop packet in WICKPro. Field size is indicated below (in bytes)

Src	Dst	Type	Serial	NumMiC
1	1	1	1	1

A.2 Data Packets

As mentioned during this dissertation, WICKPro defines four data packet types, so-called Data1, Data2, Data3 and Data4. Every type has different communication requirements characterized by a real-time traffic model using the notation provided in Table 2.2. Table A.9 depicts the fields of a data packet in WICKPro:

- *Src* is the source address of the data packet.
- *Dst* is the destination address of the data packet.
- *Type* is the packet type: Data1, Data2, Data3 or Data4.

- *Label* identifies the flow to which the data packet belongs.
- *Num* is the number of the data packet.
- *Data* is the data itself, whose maximum size is given by the Maximum Transmission Unit (MTU).

Table A.9: Data packet in WICKPro. Field size is indicated below (in bytes)

Src	Dst	Type	Label	Num	Data
1	1	1	1	1	1...MTU

A.3 Piggyback Packets

Piggyback packets are composed of one forward token packets and one data packet. Its structure can be found in Table A.10:

- *Src* is the source address of the piggyback packet.
- *Dst* is the destination address of the piggyback packet.
- *DataType* is the data packet type: Data1, Data2, Data3 or Data4.
- *TokenType* is the forward token packet type: regular token, establishment, removal or hand-off.
- *TokenPacket* is the forward token packet, whose length is the size of the regular token, establishment, removal or hand-off packet (9/13/10/11 bytes, respectively), minus the fields *Src*, *Dst* and *Type* that has already been included in the piggyback packet: 6/10/7/8 bytes.
- *Label* identifies the flow to which the data packet belongs.
- *Num* is the number of the data packet.
- *Data* is the data itself, whose maximum size is given by the MTU.

Table A.10: Piggyback packet in WICKPro. The size of each field is indicated below (in bytes)

Src	Dst	DataType	TokenType	TokenPacket	Label	Num	Data
1	1	0.5	0.5	6/10/7/8	1	1	1...MTU

A.4 Probe Packets

The probe packets are a special type of piggyback packets. In this way, when the PDR is computed, the field *Label* ranges from 0 to 63 for data flows, and from 64 to 68 for the five probe packets of the four data packets and the token packet. As all the control packets have a very similar size, we only calculate the PDR for the regular token packet and suppose that the other packets have the same PDR. More details about the PDR calculation can be found in Section 5.4.

The common header of probe packets is shown in Table A.11, which matches the header of a piggyback packet with regular token. The probe packet is filled with the number of packets transmitted and received in the last minor cycles. However, we take advantage of the cyclic packet scheduling that is known by all nodes and only the non-scheduled events in the cyclic packet scheduling are sent, i.e., the retransmissions and the non-scheduled receptions. The specific fields of probe packets are depicted in Table A.12:

- *Label* identifies the type to which the probe packet represents. It is a number from 64 to 68 that represents the four data packets and the token packet, respectively.
- *Num* is the number of the probe packet.
- *DummyData* is the synthetic data used to define a packet of the desired size.
- *NumTxRxNotSch* is the number of retransmissions and non-scheduled receptions that contains the probe packet. These events are defined by the fields *MiC* (minor cycle number), *Tx* (source node), *Rx* (destination node), *Type* (packet type) and *IsTx* (indicates whether the non-scheduled event was a transmission or a reception).

Table A.11: Common fields of probe packets in WICKPro, which matches the header of piggyback packets with regular token. The size of each field is indicated below (in bytes)

Src	Dst	DataType	TokenType	Serial	NumMiC	Notification	FreeTimeMiC	NumDataPktsSent
1	1	0.5	0.5	1	1	1	2	1

Table A.12: Specific fields of probe packets in WICKPro. Field size is indicated below (in bytes)

Label	Num	DummyData			
1	1	0...MTU			
NumTxRxNotSch	MiC(0)	Tx(0)	Rx(0)	Type(0)	IsTx(0)
1	1	1	1	1	1
...	MiC(NumTxRxNotSch-1)	Tx (NumTxRxNotSch-1)	Rx(NumTxRxNotSch-1)	Type(NumTxRxNotSch-1)	IsTx(NumTxRxNotSch-1)
	1	1	1	1	1

B Field Experiments

In this appendix we detail the hardware and the scenarios where the laboratory and field experiments of this PhD thesis were carried out.

B.1 Experiments in the I3A Building with Embedded Hardware

In Fig. B.1, we can see two of the five PcEngines ALIX3D3 boards employed in the laboratory and field experiments of Chapter 5. The five embedded nodes were fixed and ran FRT-WICKPro in MaRTE OS. In the laboratory experiments, nodes were powered over Ethernet, whereas in the field experiment nodes were battery-powered, as shown in Fig. B.1.

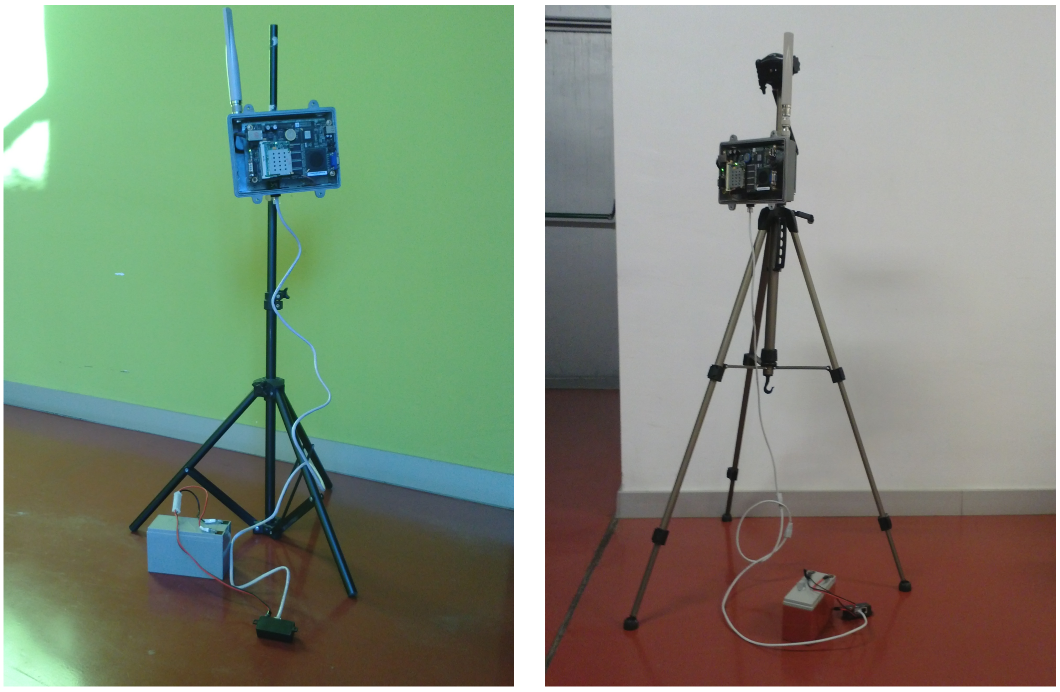


Figure B.1: Embedded nodes employed in Chapter 5

B.2 Robot Tele-operation in the I3A Building

Fig. B.2 depicts the four laptops involved in the mobile robot tele-operation of Chapter 7 that was carried out in the corridors of the I3A Building. These laptops employed Linux, SRT-WICKPro, and DoTHa. Fig. B.2d shows the laptop over a chair with casters that was used as a client in the experiments with synthetic traffic. It should be noted that the same hardware was used in Chapter 6 during the mobile robot tele-operation, but in that experiment the DoTHa algorithm was not implemented because mobility management was neglected.

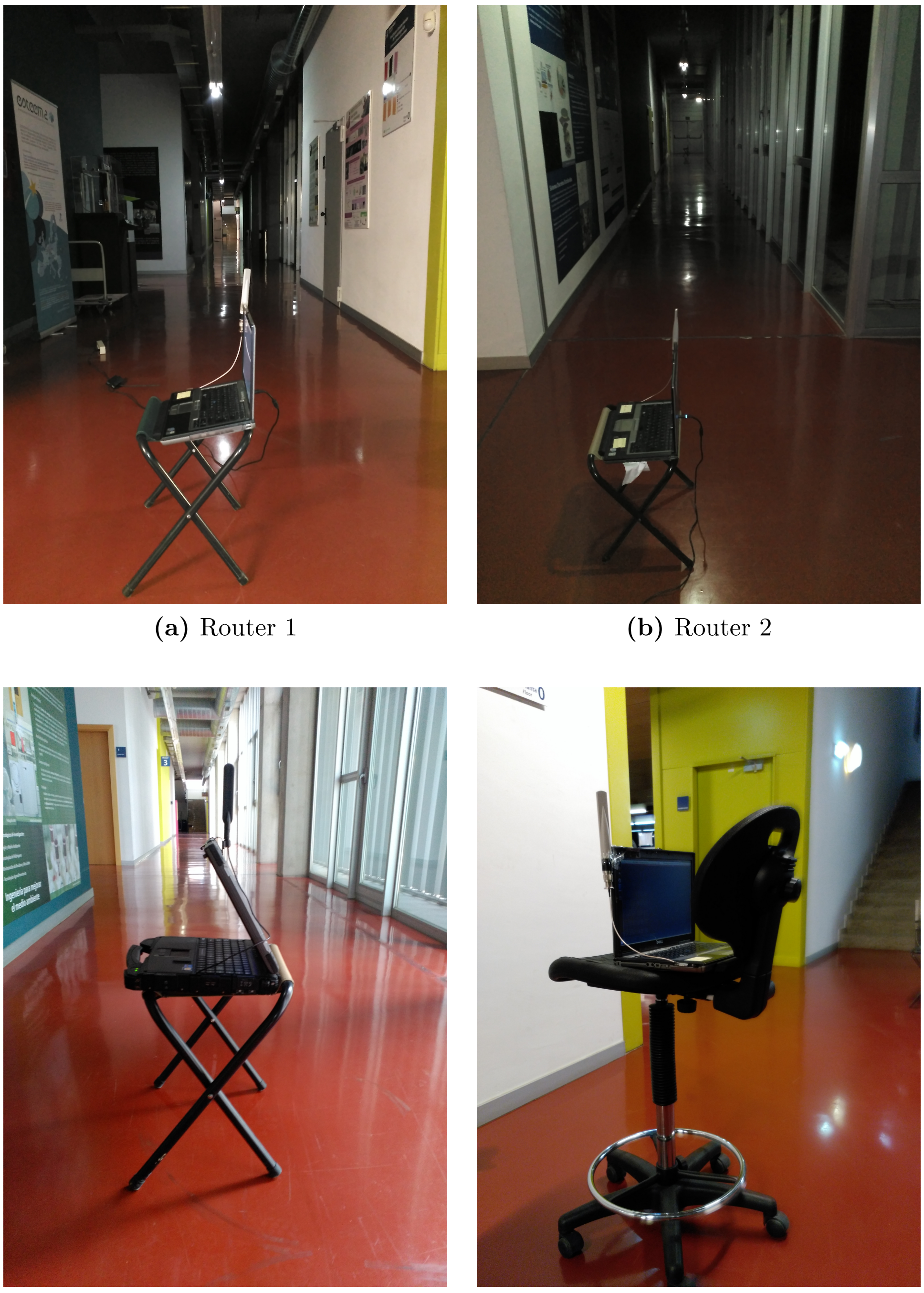
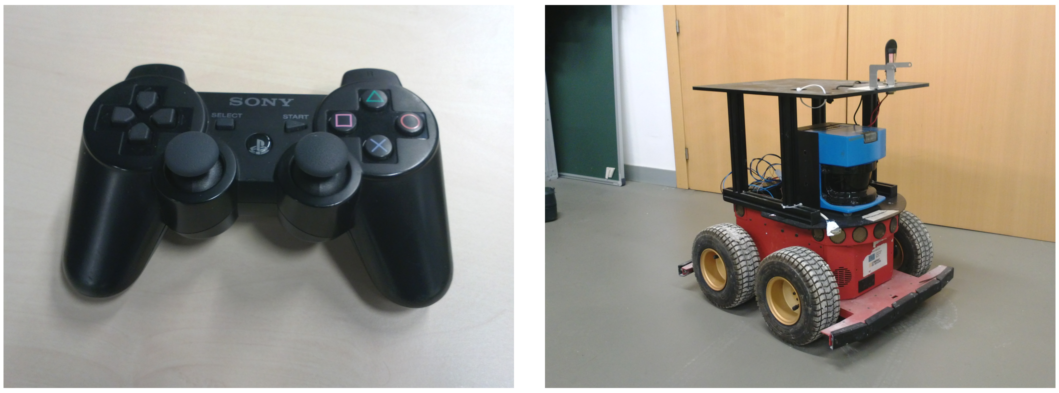

(a) Router 1 **(b)** Router 2 **(c)** Router 3 **(d)** Client 4 when using synthetic traffic

Figure B.2: Laptops used during the robot tele-operation in the I3A Building

The PS3 joystick used by the operator in the real robot tele-operation and the the tele-operated Pioneer 3-AT robot can be seen in Fig. B.3. The connection between the Router 1 (Fig. B.2a) and the PS3 joystick was handled by Bluetooth, whereas the Client 4 (Fig. B.2d) was connected to the Pioneer 3-AT robot via USB. Moreover, the ROS framework was also employed in the real robot tele-operation.



(a) PS3 joystick **(b)** The tele-operated Pioneer 3-AT robot

Figure B.3: Joystick and mobile robot involved in the real robot tele-operation within the I3A Building

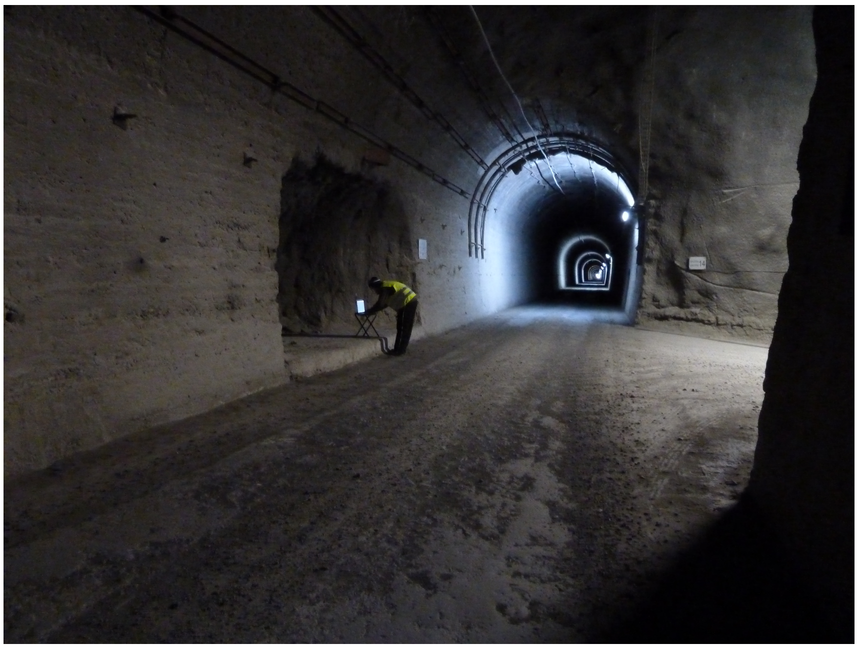
B.3 Robot Tele-operation in the Somport Tunnel

We took advantage of the hardware and software employed in the robot tele-operation in the I3A building and carried out an experiment in the Somport Tunnel. The Somport tunnel entrance along with the team involved in the Somport tunnel experiment are shown in Fig. B.4.

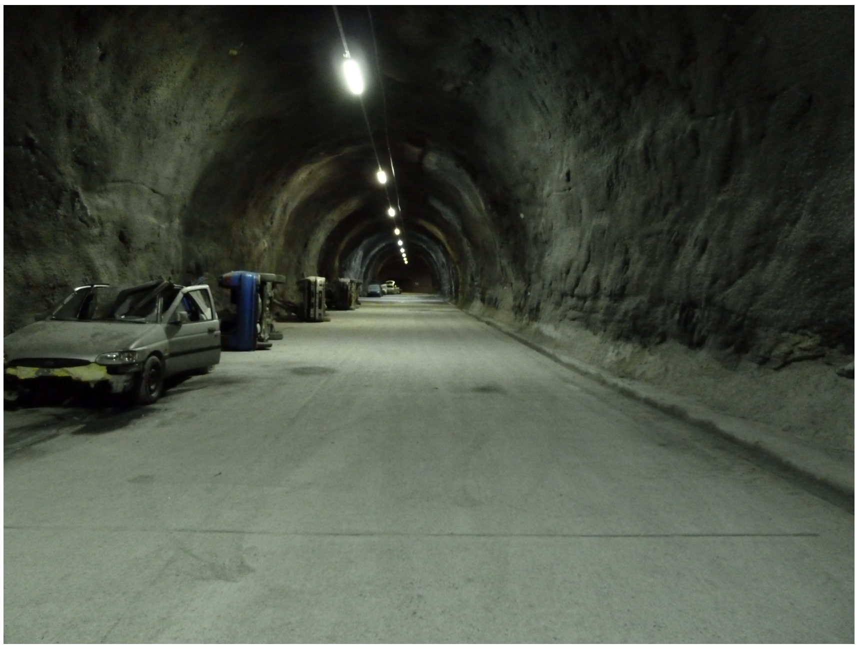
Figure B.4: The Somport tunnel entrance and the team involved in the Somport tunnel experiment

The Gallery 14 and the main section of the Somport tunnel at the intersection with Gallery 14 are depicted in Fig. B.5. In this figure, we can also observed the

Router 2 at the intersection between the main section and the Gallery 14, particularly in a hole in the wall. The hostile environment can be noticed in the previous photographs. The floor is covered with a fine layer of rock dust that makes more difficult the robot movement. The tunnel provides poor lighting and the temperature is low during the whole year, with a mean value of around 8 degrees Celsius, and the humidity and cold air streams from the Pyrenees traversing the tunnel turn the thermal sensation even lower. Moreover, communication presents special features, as shown in Section 7.8.



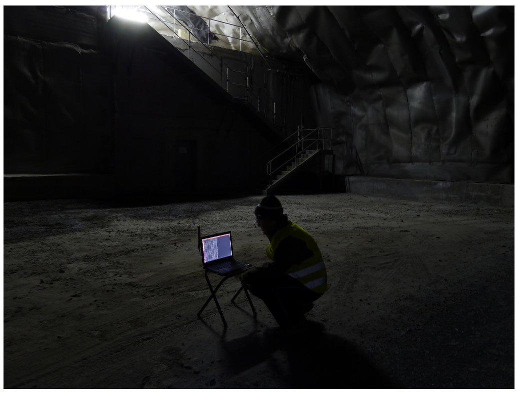
(a) The Somport tunnel at the intersection with Gallery 14 where Router 2 was located



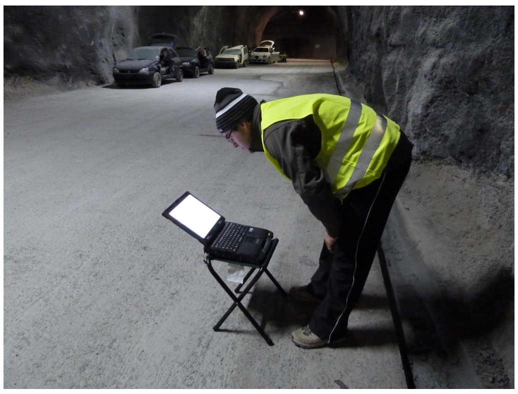
(b) Gallery 14

Figure B.5: The Gallery 14 of the Somport tunnel and surroundings

The four nodes involved in the robot tele-operation are shown in Fig. B.6.


(a) Router 1

(b) Router 2


(c) Router 3

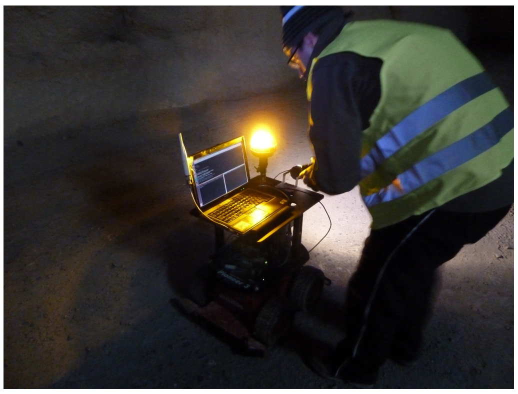

(d) Client 4

Figure B.6: The laptops, robot and PS3 joystick employed during the robot tele-operation in the Somport tunnel

Bibliography

[Aisa 10] J. Aisa & Jose L. Villarroel. *WICKPro: A Hard Real-Time protocol for Wireless Mesh Networks with chain topologies.* In European Wireless (EW) Conference, pages 163–170, Lucca, Italy, Apr 2010.

[Aisa 11] J. Aisa & Jose L. Villarroel. *The WICKPro protocol with the Packet Delivery Ratio metric.* Computer Communications, vol. 34, no. 17, pages 2047–2056, 2011.

[Aisa 15] J. Aisa & J.L. Villarroel. *Supporting Firm Real-Time Traffic in Fault-Tolerant Real-Time Systems based on Cyclic Scheduling - The WICKPro Protocol.* In IEEE International Symposium on Industrial Embedded Systems (SIES), pages 1–10, June 2015.

[Aisa 16a] J. Aisa, H. Fotouhi, J.L. Villarroel & L. Almeida. *Soft Real-time Traffic Communication in Loaded Wireless Mesh Networks.* In IEEE World Conference on Factory Communication Systems (WFCS), pages 1–8, May 2016.

[Aisa 16b] J. Aisa, H. Fotouhi, L. Almeida & J.L. Villarroel. *DoTHa - A Double-threshold Hand-off Algorithm for Managing Mobility in Wireless Mesh Networks.* In IEEE Conference on Emerging Technologies and Factory Automation (ETFA), page accepted, September 2016.

[Akyildiz 09] I. Akyildiz & X. Wang. Wireless mesh networks. Advanced Texts in Communications and Networking. Wiley, 2009.

[Almulla 14] Mohammed Almulla, Yikun Wang, Azzedine Boukerche & Zhenxia Zhang. *Design of a fast location-based hand-off scheme for IEEE 802.11 vehicular networks.* IEEE Transactions on Vehicular Technology, vol. 63, no. 8, pages 3853–3866, 2014.

[Baccour 10] Nouha Baccour, Anis Koubâa, Habib Youssef, Maissa Ben Jamâa, Denis do Rosário, Mário Alves & Leandro B. Becker. *F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks.* In Proceedings of the 7th European Conference on Wireless Sensor Networks, EWSN'10, pages 240–255, Berlin, Heidelberg, 2010. Springer-Verlag.

[Baker 89] T.P. Baker & Alan Shaw. *The cyclic executive model and Ada.* Real-Time Systems, Springer Netherlands, vol. 1, no. 1, pages S7 – S25, june 1989.

[Benedetto 13] C. Benedetto, E. Mingozzi & C. Vallati. *A handoff algorithm based on link quality prediction for mass transit wireless mesh networks.* In IEEE Symposium on Computers and Communications (ISCC), pages 969–974, July 2013.

[Bisti 11] L. Bisti, L. Lenzini, E. Mingozzi & C. Vallati. *Efficient handoff for mass transit connectivity using ieee 802.11.* In Informatica Quantitativa (Infq), June 2011.

[Bolch 05] Gunter Bolch, Stefan Greiner, Hermann de Meer & Kishor Shridharbhai Trivedi. Queueing networks and markov chains. Wiley-Interscience, 2005.

[Burns 99] A. Burns, S. Punnekkat, L. Strigini & D.R. Wright. *Probabilistic scheduling guarantees for fault-tolerant real-time systems.* In Dependable Computing for Critical Applications 7, 1999, pages 361–378, Nov 1999.

[Buttazzo 11] Giorgio C. Buttazzo. Hard real-time computing systems: Predictable scheduling algorithms and applications. Springer Publishing Company Inc., 3rd edition, 2011.

[Cheng 06] Ray-Guang Cheng, Cun-Yi Wang, Li-Hung Liao & Jen-Shun Yang. *Ripple: a wireless token-passing protocol for multi-hop wireless mesh networks.* Communications Letters, IEEE, vol. 10, no. 2, pages 123 –125, feb. 2006.

[Collotta 15] Mario Collotta. *FLBA: A fuzzy algorithm for load balancing in IEEE 802.11 networks.* Journal of Network and Computer Applications, vol. 53, pages 183–192, 2015.

[Da Xu 14] Li Da Xu, Wu He & Shancang Li. *Internet of things in industries: a survey.* IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pages 2233–2243, 2014.

[Demarch 07] D.D. Demarch & L.B. Becker. *An Integrated Scheduling and Retransmission Proposal for Firm Real-Time Traffic in IEEE 802.11e.* In Real-Time Systems, 2007. ECRTS '07. 19th Euromicro Conference on, pages 146–158, July 2007.

[EFFRA 13] European Factories of the Future Research Association EFFRA. *Factories of the future 2020: multiannual roadmap for the contractual PPP under horizon 2020.* http://www.effra.eu/attachments/article/129/Factories%20of%20the%20Future%202020%20Roadmap.pdf, 2013.

[Ergen 04] M. Ergen, Duke Lee, Raja Sengupta & P. Varaiya. *WTRP - wireless token ring protocol.* Vehicular Technology, IEEE Transactions on, vol. 53, no. 6, pages 1863 – 1881, nov. 2004.

[Facchinetti 05] Tullio Facchinetti, Giorgio Buttazzo & Luis Almeida. *Dynamic Resource Reservation and Connectivity Tracking to Support Real-time Communication Among Mobile Units.* EURASIP J. Wirel. Commun. Netw., vol. 2005, no. 5, pages 712–730, October 2005.

[Fotouhi 14] H. Fotouhi, M. Alves, M. Zuniga Zamalloa & A. Koubaa. *Reliable and Fast Hand-Offs in Low-Power Wireless Networks.* Mobile Computing, IEEE Transactions on, vol. 13, no. 11, pages 2620–2633, Nov 2014.

[Fotouhi 15] Hossein Fotouhi, Daniel Moreira & Mário Alves. *mRPL: Boosting mobility in the Internet of Things.* Ad Hoc Networks, vol. 26, pages 17 – 35, 2015.

[Galloway 13] B. Galloway & G.P. Hancke. *Introduction to Industrial Control Networks.* Communications Surveys Tutorials, IEEE, vol. 15, no. 2, pages 860–880, Second 2013.

[Goldsmith 05] Andrea Goldsmith. Wireless communications. Cambridge university press, 2005.

[Gómez 15] David Gómez, Ramón Agüero, Marta García-Arranz & Luis Muñoz. *On the Use of Hidden Markov Processes and Auto-regressive Filters to Incorporate Indoor Bursty Wireless Channels into Network Simulation Platforms.* Wirel. Netw., vol. 21, no. 7, pages 2137–2154, October 2015.

[Guo 07] Da-Ren Guo, Kuochen Wang & Lung-Sheng Lee. *Efficient Spatial Reuse in Multi-Radio, Multi-Hop Wireless Mesh Networks.* In Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th, pages 1076 –1080, 22-25 2007.

[Hou 06] Ting-Chao Hou, Chien-Yi Wang & Ming-Chieh Chan. *A token-based distributed scheduling for mesh networks with chain topologies.* In Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on, volume 1, page 6 pp., 18-20 2006.

[IEEE802.11 97]	IEEE802.11. *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* IEEE Std 802.11-1997, pages i–445, 1997.
[IEEE802.11e 05]	IEEE802.11e. *IEEE Standard for Information technology–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements.* IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003), pages 1–212, Nov 2005.
[Jayachandran 10]	P. Jayachandran & M. Andrews. *Minimizing End-to-End Delay in Wireless Networks Using a Coordinated EDF Schedule.* In IEEE INFOCOM, pages 1–9, March 2010.
[Jun 03]	Jangeun Jun, P. Peddabachagari & M. Sichitiu. *Theoretical maximum throughput of IEEE 802.11 and its applications.* In IEEE International Symposium on Network Computing and Applications, pages 249 – 256, 16-18 2003.
[Kopetz 11]	Hermann Kopetz. Real-time systems - design principles for distributed embedded applications. Real-Time Systems Series. Springer, 2011.
[Kurose 12]	James F. Kurose & Keith W. Ross. Computer networking: A top-down approach (6th edition). Pearson, 6th edition, 2012.
[Lakshmi 15]	L Rajya Lakshmi, Vinay J Ribeiro & BN Jain. *PRIME: A partial path establishment based handover management technique for QoS support in WiMAX based wireless mesh networks.* Computer Networks, vol. 83, pages 217–234, 2015.
[Lee 14]	Jong-Kwan Lee, Hong-Jun Noh & Jaesung Lim. *TDMA-Based Cooperative MAC Protocol for Multi-Hop Relaying Networks.* Communications Letters, IEEE, vol. 18, no. 3, pages 435–438, March 2014.
[Li 14]	Haopeng Li & Jiang Xie. *GaS: A Gateway Scheduling-Based Beamforming Scheme for Handoff Support in Wireless Mesh Networks.* IEEE Transactions on Vehicular Technology, vol. 63, no. 9, pages 4560–4573, 2014.
[Lindhorst 13]	T. Lindhorst, G. Lukas & E. Nett. *Wireless Mesh Network infrastructure for industrial applications - A case study of*

tele-operated mobile robots. In IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–8, Sept 2013.

[Liu 06] D. Liu, X.S. Hu, M.D. Lemmon & Qiang Ling. *Firm real-time system scheduling based on a novel QoS constraint.* Computers, IEEE Transactions on, vol. 55, no. 3, pages 320–333, 2006.

[Malcolm 94] Nicholas Malcolm & Wei Zhao. *The timed-token protocol for real-time communications.* Computer, vol. 27, no. 1, pages 35–41, 1994.

[Nguyen 09] Q. V. Nguyen & R. G. Cheng. *Enhanced Ripple (E-Ripple) protocol for chain-based multihop wireless networks.* In IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops (WoWMoM), pages 1–7, June 2009.

[Oliveira 15] L. Oliveira, L. Almeida & P. Lima. *Multi-hop routing within TDMA slots for teams of cooperating robots.* In IEEE World Conference on Factory Communication Systems (WFCS), pages 1–8, May 2015.

[OMNet++ 16] OMNet++. *Official Site for OMNet++.* http://www.omnetpp.org/, 2016.

[Peterson 11] Larry L. Peterson & Bruce S. Davie. Computer networks: A systems approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.

[Pinto 16] L. Pinto, A. Moreira, L. Almeida & A. Rowe. *Aerial multi-hop network characterisation using COTS multi-rotors.* In 2016 IEEE World Conference on Factory Communication Systems (WFCS), pages 1–4, May 2016.

[Que 07] Can Que, Xinming Zhang, Yongzhen Liu & Yanbin Huang. *Performance Analysis of Chain Topology in IEEE 802.11 Multi-hop Ad hoc Networks.* In Convergence Information Technology, 2007. International Conference on, pages 1880 –1883, 21-23 2007.

[Quigley 09] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler & Andrew Y. Ng. *ROS: an open-source Robot Operating System.* In ICRA Workshop on Open Source Software, 2009.

[Ramani 05] Ishwar Ramani & Stefan Savage. *SyncScan: practical fast handoff for 802.11 infrastructure networks.* In IEEE INFOCOM, pages 675–684, March 2005.

[Rivas 01] Mario Aldea Rivas & Michael Gonzalez Harbour. *MaRTE OS: An Ada Kernel for Real-Time Embedded Applications.* In Ada-Europe, pages 305–316, 2001.

[Rizzo 13] C. Rizzo, D. Tardioli, D. Sicignano, L. Riazuelo, J. L. Villarroel & L. Montano. *Signal Based Deployment Planning for Robot Teams in Tunnel-like Fading Environments.* The International Journal of Robotics Research, vol. 32, no. 12, page 1381–1397, 2013.

[Rizzo 15] Carlos Rizzo. *Propagation, Localization and Navigation in Tunnel-like Environments.* PhD thesis, Universidad de Zaragoza, 2015.

[ROS-PS3 16] ROS-PS3. *Playstation 3 joystick driver for ROS.* http://wiki.ros.org/ps3joy, 2016.

[ROS-RTWMP 16] ROS-RTWMP. *ROS version of the RT-WMP protocol.* http://wiki.ros.org/ros-rt-wmp, 2016.

[ROS-SICK-LMS2xx 16] ROS-SICK-LMS2xx. *SICK LMS2xx laser driver for ROS.* http://wiki.ros.org/sicktoolbox_wrapper, 2016.

[Sevani 14] V. Sevani, B. Raman & P. Joshi. *Implementation-Based Evaluation of a Full-Fledged Multihop TDMA-MAC for WiFi Mesh Networks.* Mobile Computing, IEEE Transactions on, vol. 13, no. 2, pages 392–406, Feb 2014.

[Shin 04] Minho Shin, Arunesh Mishra & William A Arbaugh. *Improving the latency of 802.11 hand-offs using neighbor graphs.* In ACM conference on Mobile systems, applications, and services, pages 70–83. ACM, 2004.

[Stallings 07] William Stallings. Data and computer communications (8th ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2007.

[Tanenbaum 11] Andrew S. Tanenbaum & David J. Wetherall. Computer networks. Prentice Hall, 5th edition, 2011.

[Tardioli 14] D. Tardioli. *A wireless communication protocol for distributed robotics applications.* In IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pages 253–260, May 2014.

[Tardioli 15] Danilo Tardioli, Domenico Sicignano & José Luis Villarroel. *A wireless multi-hop protocol for real-time applications.* Computer Communications, vol. 55, pages 4 – 21, 2015.

[Toscano 10] E. Toscano, F. Misenti & L. Lo Bello. *A traffic scheduler for real-time wireless communication in adaptable industrial automation systems.* In IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–8, Sept 2010.

[Trsek 11] H. Trsek, L. Wisniewski, E. Toscano & L. L. Bello. *A flexible approach for real-time wireless communications in adaptable industrial automation systems.* In IEEE Conference on Emerging Technologies Factory Automation (ETFA), pages 1–4, Sept 2011.

[Ubiquiti 16] Ubiquiti. *SuperRange Cardbus Datasheet.* https://www.streakwave.com/mmSWAVE1/Video/src_datasheet.pdf, 2016.

[Vlavianos 08] A. Vlavianos, L.K. Law, I. Broustis, S.V. Krishnamurthy & M. Faloutsos. *Assessing link quality in IEEE 802.11 Wireless Networks: Which is the right metric?* In Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on, pages 1 –6, 15-18 2008.

[Wei 13] Yi-Hung Wei, Quan Leng, Song Han, A. K. Mok, Wenlong Zhang & M. Tomizuka. *RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications.* In IEEE Real-Time Systems Symposium (RTSS), pages 140–149, Dec 2013.

[Willig 08] Andreas Willig. *Recent and emerging topics in wireless industrial communications: A selection.* IEEE Transactions on Industrial Informatics, vol. 4, no. 2, pages 102–124, 2008.

[Xie 08] Jiang Xie & Xudong Wang. *A survey of mobility management in hybrid wireless mesh networks.* IEEE Network, vol. 22, no. 6, pages 34–40, 2008.

[Xu 02] Kaixin Xu, M. Gerla & Sang Bae. *How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks.* In Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE, volume 1, pages 72–76 vol.1, Nov 2002.

[Yarkan 09] S. Yarkan, S. Guzelgoz, H. Arslan & R. R. Murphy. *Underground Mine Communications: A Survey.* IEEE Communications Surveys Tutorials, vol. 11, no. 3, pages 125–142, rd 2009.

[Yepez 06] J. Yepez, J. Guardia, M. Velasco, J. Ayza, R. Castane, P. Marti & J. M. Fuertes. *CICLIC: A Tool to Generate Feasible Cyclic Schedules.* In 2006 IEEE International Workshop on Factory Communication Systems, pages 399–404, 2006.

[Zhang 04] Sijing Zhang, Alan Burns, Jing Chen & E. Stewart Lee. *Hard Real-Time Communication with the Timed Token Protocol: Current State and Challenging Problems.* Real-Time Syst., vol. 27, no. 3, pages 271–295, 2004.

[Zhang 10] Z. Zhang, Z. Lu, Q. Chen, X. Yan & L. R. Zheng. *COSMO: CO-Simulation with MATLAB and OMNeT++ for Indoor Wireless Networks.* In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pages 1–6, Dec 2010.

[Zuniga 04] Marco Zuniga & Bhaskar Krishnamachari. *Analyzing the transitional region in low power wireless links.* In IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON), pages 517–526. IEEE, 2004.

Acronyms

ABL	Average Burst Length
ACK	ACKnowledgment
AP	Access Point
ARQ	Automatic Repeat reQuest
BER	Bit Error Rate
CC	Congestion Control
CDMA	Code Division Multiple Access
COTS	Commercial Off-The-Shelf
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear To Send
DCF	Distributed Coordination Function
DIFS	DCF InterFrame Space
DDR	Data Delivery Ratio
DSSS	Direct-Sequence Spread Spectrum
DoTHa	Double-Threshold Hand-off
DTMC	Discrete-Time Markov Chain
EDCA	Enhanced Distributed Channel Access
EDF	Earliest Deadline First
EED	End-to-End Delay

EWMA Exponentially Weighted Moving Average

FEC Forward Error Correction

FIFO First In, First Out

FRT Firm Real-Time

FP Fixed Priorities

GPS Global Positioning System

HCCA HCF Controlled Channel Access

HCF Hybrid Coordination Function

HRT Hard Real-Time

ISM Industrial, Scientific and Medical

IsoMAC Isochronous MAC

ISRA Integrated Scheduling and Retransmission Approach

LiT MAC Light-Weight TDMA MAC

LLC Logical Link Control

MANET Mobile Ad-Hoc Network

MAC Medium Access Control

NACK Negative ACKnowledgment

OFDM Orthogonal Frequency-Division Multiplexing

PLR Packet Loss Rate

PCF Point Coordination Function

PDR Packet Delivery Ratio

PTP Precision Time Protocol

P-MiC Percentage of Minor Cycles satisfied

QoS Quality of Service

RF Radio Frequency

ROS Robot Operating System

RMP Radio-Matching Protocol

RSSI Received Signal Strength Indicator

RT-WMP Real-Time Wireless Multi-hop Protocol

SINR	Signal-to-Interference-plus-Noise Ratio
RMS	Rate Monotonic Scheduling
RTS	Request To Send
SIFS	Short InterFrame Space
SMA	Simple Moving Average
SRT	Soft Real-Time
THO	Token Holding Opportunity
TDS	Token-based Distributed Scheduling
TDMA	Time Division Multiple Access
TPTH	Transmission Per Token Holding
TRT	Token Rotation Time
UAV	Unmanned Aerial Vehicle
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WICKPro	WIreless Chain networK Protocol
WTRP	Wireless Token Ring Protocol