UNIVESITY OF MODENA AND REGGIO EMILIA

Department of Engineering "Enzo Ferrari"

Doctorate School in Information and Communication Technologies (ICT)

Cycle XXVIII

PhD Dissertation

Communication Networks for Public Protection and Disaster Relief Systems

Advisor:
Prof. MAURIZIO CASONI

Candidate:
CARLO A. GRAZIA

The Coordinator of the Doctorate:
Prof. GIORGIO M. VITETTA

The Director of the School:
Prof. GIORGIO M. VITETTA

Communication Networks for Public Protection and Disaster Relief Systems

Carlo Augusto Grazia

Abstract

Network-enabled services for Public Protection and Disaster Relief (PPDR) professionals are more than necessary in todays emergency situations. Under the extreme circumstances of an emergency, it is essential to have networks which support the required data throughput as well as high availability in spite of high traffic volumes, and minimize end-to-end delay. In the last decades, these issues have been steering a good portion of networking research in ICT world; PPDR networks, however, still represent a significant case study due to the challenges that they entail.

During the emergency operations that follow natural or man-made disasters, in fact, both commercial and dedicated terrestrial networks often fail to provide the necessary support. The reason is threefold: they simply get destroyed by the disaster, they cannot sustain the sudden surge of network demand, or they fail to deliver the necessary bandwidth and/or other QoS guarantees. The final goal is to provide field operators and people in distress with transparent accessibility, coverage guarantees and broadband performance, with both the technological and the algorithmic points of view properly considered.

The contribution of this work is to analyse the issues and the challenges belonging to the PPDR world, and to provide suitable solutions to the above mentioned problems. A key feature of the work is that the proposed solutions are not simply tailored for specific PPDR scenarios, but they can be easily deployed in whichever network environment, representing a possible ICT solution for a vast range of problems. The proposed solutions can all be placed at (or around) the networking level of the TCP-IP stack, designed with the cross-layering paradigm to boost the efficiency and to address resiliency, a vital feature for PPDR systems, with different technologies and environmental conditions.

This work presents and analyses the following proposals: (i) a middleware layer solution to control network congestion and fairness; (ii) a no-drop AQM algorithm that mitigates TCP issues related to RTT variations and congestion avoidance; (iii) a modular packet scheduler algorithm that provides a balance between high QoS guarantees and high throughput while still offering the possibility to boost one with respect to another in a fine-grained way, offering flexibility in PPDR systems.

All these solutions are tested and analysed through analytical and experimental evaluations in order to provide them with sound and robust support, by using both simulations and emulations techniques.

Keywords: emergency networks; cross-layering; middleware solutions; AQM; packet scheduling.

Contents

Chapter 1

Introduction

1.1 PPDR Communication Networks

Wireless communications are critical for public protection and disaster relief (PPDR) professionals during the emergency operations that follow natural or man-made disasters, scenarios in which both commercial and dedicated terrestrial networks often fail to provide the necessary support. The reason is threefold: they simply get destroyed by the disaster, they cannot sustain the sudden surge of network demand or they fail to deliver the necessary bandwidth and/or other QoS guarantees. From a technological point of view, LTE is expected to become the main wireless technology for broadband communication; a lot of studies have been devoted to assess its compliance for PPDR purposes and to find suitable architectural solutions able to meet mission-critical requirements. This approach is surely worthy, but it is based on the assumption that infrastructure-based terrestrial systems are reliable. As a consequence, in worst-case emergency scenarios appropriate guarantees can be provided only in the hypothesis of huge investment costs. Recent developments in satellite technologies are bringing the availability of non-terrestrial high performance channels, with better properties when comparing to LTE for what regards availability and reliability. On this basis, the best approach could be a network architecture based on the integration of satellite and LTE networks for both infrastructure-based and infrastructure-less scenarios. Throughout this integration we aim to provide field operators and people in distress with transparent accessibility, coverage guarantees and broadband performance when terrestrial infrastructures are lacking, and to expand

coverage, capacity and resilience otherwise. Moreover, on one side we analyse this integration from a technological and topological point of view while, on the other side, starting from the defined architecture we propose a centralized middleware and investigate its performance optimization from a service point of view.

1.2 Networking Algorithms

The networking layer of the TCP-IP stack can be introduced by presenting two of its main goals: (i) QoS provisioning and (ii) congestion control. In the Linux Kernel there is a specific networking layer, called queueing layer, in which each deployed queue faces one of these two tasks (i.e. this happens in generic internet nodes like gateways or routers). The first task, the QoS provisioning, is addressed by using a Packet Scheduler algorithm. A packet scheduler separates different traffic types in different queues and chooses the next packet to transmit according to specific QoS requirements. The second task, the congestion control, is addressed by using an Active Queue Manager (AQM). An AQM implements techniques to control the queue length (e.g. it drops packets) in order to avoid to exceed the queue limit and cause congestion on the network. In other words, a packet scheduler controls the order, while the AQM controls the amount of packets.

Packet scheduling has been deeply studied for lot of years by researchers in the computer science and telecommunications fields as an important solution that decides the order in which packets are sent over a link in order to provide QoS on a network. Recently, packet scheduling has become again a challenging topic due to the massive use of wireless technologies (e.g. WiFi, LTE, 4G/5G), with which is still an open issue to provide high QoS guarantees. A common solution is using a single, integrated scheduler that deals both with the QoS guarantees and the wireless link issues. Unfortunately, such an approach is little flexible and does not allow any of the existing high-quality schedulers for wired links to be used without modification. To address these issues, in this thesis we validate a modular architecture which permits the use, as they are, of existing packet schedulers for wired links over wireless links, and at the same time allows the flexibility to adapt to different channel conditions. We validate the effectiveness of the modular architecture by showing, through experimental

results, that this architecture enables us to get a new scheduler with the following features, just by combining existing schedulers: execution time and energy consumption close to a Deficit Round Robin, accurate fairness and delay guarantees, possibility to set, by changing one parameter, the desired trade-off between throughput-boosting level and granularity of the service guarantees. In particular, we show that this scheduler, which we named High-throughput Twin Fair scheduler (HFS), outperforms one of the most accurate and efficient integrated schedulers available in the literature.

The traditional role of Active Queue Management in IP networks was to complement the work of end-system protocols such as the Transmission Control Protocol (TCP) in congestion control so as to increase network utilization, and to limit packet loss and delay, avoiding congestion. Since its formal introduction to IP networks in 1993 as a viable complementary approach for congestion control, there has been a steady stream of research outputs with respect to Active Queue Management. This thesis attempts to define and present a novel AQM called PINK (Passive INverse feedbacK), a queue management algorithm designed to indirectly impose a certain resource allocation policy on defined sets of client hosts. PINK adds intelligence at intermediate nodes that connect client hosts to bottleneck links or to external networks in general, allowing these nodes to dynamically modify the TCP Acknowledgements (ACKs) segments passing through. This is made by setting TCP ACK advertised Receive Windows field (RCV.WNDs) to custom values, in order to enforce a specific bandwidth utilization upper bound. To compute new RCV.WND values, PINK needs only the number of active connections, the flows RTTs and the transmission channel bandwidth. It follows that PINK permits to impose a centralized bandwidth management without the cooperation of clients, which means that no modification or addition whatsoever to end hosts is needed. Furthermore, throughout simulations we demonstrate that PINK improves satellite emergency network performance where high RTTs are in place. At the same time, both for high RTT emergency networks and for general purpose networks, PINK allows to efficiently exploit channel throughput maintaining a low queuing delay, and guarantees optimal flow fairness without forcing any packet drop.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows.

In **Chapter 2**, we describe how an Emergency Network is deployed and how to mitigate, in a centralized way, important issues like the congestion control and the bandwidth management.

- In **Section 2.1**: (*i*) we define how the current PPDR systems are deployed and (*ii*) we propose an integration between the emerging LTE technology with the Satellite one as a backhaul technique.
- In **Section 2.2**: (*i*) we propose a novel centralized middleware for congestion control and (*ii*) we compare this solution with standard techniques.
- Finally, in **Section 2.3**: (*i*) a novel algorithm for bandwidth distribution is proposed, (*ii*) the algorithm is formalized throughout a mathematical formulation and (*iii*) integrated with the centralized middleware in order to boost the general performance achieved.

In **Chapter 3**, we address the congestion control problem through the use of AQM techniques moving from a "ad hoc" and centralized middleware to a general AQM suitable for internet nodes regardless the topology.

- In **Section 3.1**: (*i*) a novel AQM technique for rate management in cooperative networks is described and (*ii*) the algorithm is deployed and tested with the centralized middleware.
- In **Section 3.2**: (*i*) a smart no-drop AQM for emergency network is proposed, (*ii*) the solution is uncoupled from the middleware and studied for general PPDR networks and (*iii*) an analysis on satellite network is provided.
- Finally, in **Section 3.3**: (*i*) the no-drop AQM solution is deeply investigated and analysed though mathematical formulations, (*ii*) the algorithm is compared with general AQM solutions for general purpose networks and (*iii*) strong performance analysis are presented over several figures of merit (goodput, latency, RTT variation, fairness and energy consumption).

In **Chapter 4**, we addressed the trade-off between QoS guarantees and throughput with a novel packet scheduler for wireless links.

- In **Section 4.1**: (*i*) we define a modular architecture suitable for PPDR wireless networks and (*ii*) describe the benefits introduced by this flexible approach.
- In **Section 4.2**: (*i*) we describe a novel packet scheduler deployed on top of the modular architecture, (*ii*) we provide a formal analysis of the QoS bounds of the novel scheduler and (*iii*) we test this algorithm against the high-performance packet schedulers available in the literature.
- Finally, in **Section 4.3**: (*i*) we describe the novel test environment for packet scheduling testing, (*ii*) we provide a detailed overview about this test environment features and (*iii*) validate the tool through the analysis of several packet schedulers figures of merit.

Finally, in **Chapter 5** the conclusions are drawn and the main achievements of the thesis are summarized.

Chapter 2

Middleware for PPDR systems

In this Chapter we focus on the Emergency Network topic by defining and presenting the reference architecture. We start by defining an integration between the state-of-the-art LTE technology for Incident Area Network (IAN) coverage with a Satellite link as a network backhaul and, in this way, we can deal with major disaster events while boosting the coverage and the reliability of the emergency network itself. Once introduced the architecture, we define and present a Congestion Control Middleware Layer (C^2ML), a cooperative middleware suitable for centralized emergency networks organized with a hierarchical star topology. With this novel middleware it is possible to improve end users fairness while also improving QoS and reducing end-to-end latency. Together with C^2ML is also presented $DyBRA$, a novel algorithm for bandwidth distribution; the algorithm is integrated with the centralized middleware in order to boost the performance even in high dynamic environments in which clients experience different channel conditions during the emergency operations.

2.1 The Reference Architecture

Current technologies employed for PPDR purposes provide a rich set of voice-centric services that are of paramount importance for field operators, especially in the very early stages of the response; unfortunately, these systems are unable to sustain high-bandwidth data-oriented applications, for which there is an increasing demand from the PPDR community. Furthermore, due to the generalized lack of a common PPDR communication infrastructure between different PPDR entities (e.g. different police corps, fire department, ambulance service), operators often relies on commercial terrestrial networks for coordination and data-oriented communication [1].

To provide field operators with reliable, high performance communication channels, LTE is increasingly being chosen for next-generation public safety networks, for which both dedicated only and hybrid dedicated/commercial solutions have been proposed [2] [3]. These approaches are valuable for day-to-day routine activities and major planned events, because they can offer great improvements to network capacity and in consequence to the range of services and applications that PPDR users can employ. However, in most major incidents and disasters, terrestrial mobile networks are overloaded or their infrastructures are damaged and thus out of service [4]. Even the deployment of ad-hoc mobile networks backhauled by infrastructure-based facilities cannot therefore offer adequate guarantees in the latter, worst-case scenarios.

Nevertheless, LTE is indeed a technology that shows a great potential in unexpected emergency situations; for example, preliminary research on dynamic spectrum management protocols has shown [5] how LTE could exploit spectrum holes created by the sudden failure of terrestrial infrastructures through cognitive radio access techniques, providing increased transmission quality and coverage.

This Section proposes an infrastructure-less approach to provide high-bandwidth connectivity through deployable LTE base stations, backhauled by new-generation satellite systems. This way, it is possible to provide coverage in the majority of the cases in which terrestrial infrastructures are damaged or destroyed by a disaster, without requiring field operators and civilians to employ special equipment. In infrastructure-based scenarios, this proposal provides PPDR operators with extended coverage, higher broadband capacity and greater resilience.

Section 2.1.1 describes the deficiencies of systems currently used to offer communication in PPDR operations and presents, briefly, the features that are demanded by PPDR stakeholders for the future. Section 2.1.2 details our proposal through the analysis of network architecture, properties and features. Section 2.1.3 draws the main conclusions.

2.1.1 Current PPDR Systems

In the European Union, PPDR communication is currently performed by means of different technologies, depending on the country or even on the specific region: TETRA Release 1, TETRA Release 2, TETRAPOL, Analogue Professional Mobile Radio (PMR), Digital PMR (dPMR), Digital Mobile Radio (DMR) and satellites. Their main purposes are the provision of specific voice capabilities, such as Group Calls and Direct Mode Operation (DMO), and the facilitation of voice calls when terrestrial networks coverage is lacking.

TETRA Release 1, the Terrestrial Trunked Radio standard, was firstly introduced in 1995 by the European Telecommunications Standards Institute (ETSI) as a PMR two-way transceiver with Voice plus Data (V+D) capabilities. TETRA uses Time Division Multiple Access (TDMA) and a $\pi/4$ DQPSK modulation scheme, with four user channels on one radio carrier and 25 kHz spacing between carriers. Both point-to-point and point-to-multipoint transfers can be done. In addition to voice and dispatch services, digital data transmission is also included in the standard although at a very low data rate. The most important services provided by TETRA are specific voice capabilities, namely: wide area "all informed net" group calls, Direct Mode Operation (DMO) that allows direct communication between radio terminals independently from the network, high level voice encryption to meet the security needs of public safety organizations, an Emergency Call facility that gets through even if the system is congested and full-duplex voice for PABX and PSTN telephony communications. TETRA maximum data rate is 7.2 kbit/s, which can be increased up to 28.8 kbit/s if multi-slot packets are employed.

TETRA Release 2 introduced some enhancements to the first version of the standard: Enhanced Data Service (TEDS), Range Extension for Trunked Mode Operation and additional specific voice codecs; the physical layer remained the same. The biggest improvement of TETRA release 2 is the introduction of

the TEDS standard which, by utilizing different modulation schemes and by supporting the use of different RF channel bandwidths, allows for data rates up to 400 kbit/s. It also permits QoS negotiations in terms of throughput, delay, priority and reliability.

TETRAPOL is a digital, cellular trunked radio system for voice and narrowband data communication developed by private stakeholders; it has no relation with TETRA, although the offered functionalities are very similar. The physical network structure of TETRAPOL is similar to that of TETRA, but TETRAPOL uses Frequency Division Multiple Access (FDMA) with only one user channel per carrier, which limits the maximum data rate to 7.2 kbit/s.

PMR is an analogue technology available for both licensed and unlicensed spectrum use; Analogue PMR applications extend from low-cost walkie-talkies aimed at the consumer market to public safety and mission-critical systems. The reachable data rate is even lower than the aforementioned solutions, halting at 1 kbit/s.

dPMR utilizes digital transmission over the air interface allowing for minimal migration costs from Analogue PMRs. dPMR uses a 6.25 kHz FDMA radio with a 4FSK modulation scheme, providing good spectrum efficiency. dPMR data rate ranges between 1.2 kbit/s up to 9.6 kbit/s.

DMR has been produced in order to facilitate migration from analogue PMR to dPMR by allowing the simultaneous utilization of both technologies. This is realized through FM analogue modulation and selective calling based on traditional protocols for voice communication, and through 4FSK digital modulation for voice communication and data transmission. DMR is also capable of performing simulcast transmissions, in which the network repeaters are active on the same frequency at the same time. The DMR standard permits the use of a 12.5 KHz spaced radio carrier for the simultaneous employment of two TDMA channels. DMR use a modulation scheme called *constant envelope*, that allow a decrease in power consumption and facilitates spectrum efficiency features, providing a maximum data rate of 9.6 kbit/s.

Satellite communications are currently used only to facilitate voice calls and narrowband data transmissions in remote areas where terrestrial networks coverage is lacking or to provide field operators with a reliable communication medium. In its simplest form, the network architecture consists of three entities: the user segment (user terminals), the ground segment (gateways, network

control centers and satellite control centers) and the space segment (one or more satellite constellations). There is a large variation of achievable data rates, usually ranging from 2.4 kbit/s to tens of Mbit/s depending on the constellation type and the utilized frequency bands.

Although these technologies are commonly used in the PPDR domain, they exhibit well-understood deficiencies, especially in relation to the PPDR community demands for data-oriented applications. In particular, the key properties that these systems are unable to meet are:

- Data Rate: current PPDR technologies do not offer broadband capabilities; in fact, except with TETRA Release 2 and some satellite links, the achievable data rates are always less than 30 kbit/s. TETRA Release 2 allows for data rates up to 400 kbit/s whereas, for what regards satellite communications, there is a large variation of achievable data rates, usually ranging from 2.4 kbit/s to an aggregate of tens of Mbit/s depending on the constellation type and the utilized frequency bands. It is therefore impossible to exploit applications like video streaming, high-resolution image transmissions and data services in general [6]. For these, PPDR personnel often relies on commercial terrestrial networks [1].

- Availability: currently deployed infrastructures often fail to guarantee operation in major incidents and disaster scenarios. Even in the absence of damage to terrestrial infrastructures, their coverage is far from being ubiquitous, especially for data-intensive applications.

- Resilience: in addition to the aforementioned issues, the current infrastructure-based systems are not redundant and, in general, they are vulnerable to disasters and subsequent incidents (e.g. earthquakes often happen in series, posing a threat to terrestrial infrastructures even if the first event has left them partially or totally operational).

- Spectrum: to date, spectrum allocation is not harmonized between EU countries, affecting the interoperability of PPDR systems, especially in cross-border operations.

Between PPDR stakeholders, there is complete consensus on the paramount importance that voice services play in every disaster relief operations. It is also

clear that data and video services have already started to play an important role for PPDR users [1]. PPDR operators need networks that offer high availability, high reliability, high security and faster data transmissions; capabilities that map respectively into coverage, resilience, security and data rate properties.

2.1.2 Integration of Satellite and LTE

In general, in PPDR operations it is often essential to employ satellite technologies or cellular repeater stations to provide wireless network coverage to field operators and people in distress. Traditionally, PPDR agencies have had to choose between the two; instead, we show that they can be employed together. A model that encompasses both terrestrial and satellite technologies has the potential to provide field operators with a fast and reliable way to communicate in emergency scenarios, overcoming the typical issues faced today for what regards bandwidth-intensive applications. In order to specify a next-generation PPDR network architecture able to sustain the features and requirements that PPDR stakeholders are demanding, we propose an integration between LTE and satellite technologies, because it is the combination of the two that may offer the best benefits for PPDR operators and people in distress in critical disaster scenarios, by exploiting the benefits of both and at the same time mitigating each technology specific issue.

This approach has been chosen because satellites, today, are an almost ubiquitous mean to provide broadband connectivity; however, general-purpose User Equipment (UE) such as mobile phones and smartphones do not have, in general, components to access satellite systems. Furthermore, the propagation delay imposed by satellites would yield impractical the communication among field personnel. On the contrary, general-purpose UE is built to connect to commercial mobile networks, and it is already clear that LTE and LTE-Advanced are going to be the next key general-purpose technologies for mobile devices.

The potential benefits of a hybrid architecture composed by LTE base stations backhauled with satellite links may be summarized as:

- Easy access to the deployed network with general-purpose User Equipment (UE), through LTE.
- Almost ubiquitous external coverage, even in case of failure of terrestrial networks, through satellite.

- High bandwidth capabilities and reasonable latencies, especially with the newest Ka-band Medium Earth Orbit satellite constellations.
- Concurrent exploitation of existing terrestrial commercial networks, if they are still operating in the disaster area, by the addition of coverage, capacity and resilience.
- Possibility to bring coverage indoor and in tunnels, through UE bridges or via commercially available LTE picocells and femtocells.

Therefore, the key design requirements for the architecture have been defined as:

- Accessibility: the deployed network(s) must be easily accessed by general UE.
- Coverage: in absence of adverse external conditions, it must be possible to deploy Incident Area Network(s) where terrestrial coverage is disrupted or absent.
- Performance: it must provide broadband access, at least to PPDR operators.

The subsections that follow details our proposal.

2.1.2.1 System architecture

In disaster scenarios, a complete or partial destruction of already existing terrestrial networks must be expected but it is not "a priori" predictable; therefore, both infrastructure-based and especially infrastructure-less solutions for communication must be taken into account. In fact, base stations may not be the only network components affected by disasters, also aggregation channels and core infrastructures may be damaged. Our proposal aims at being a solution in the worst-case (i.e. when an infrastructure-less network must be deployed and represents the only source of coverage), while adding capacity and functionalities when an infrastructure-based LTE network is still totally or partially operational.

As the general concept, we consider LTE to be the access technology while satellite the backhaul one; i.e. satellites are used as backhaul means to convey

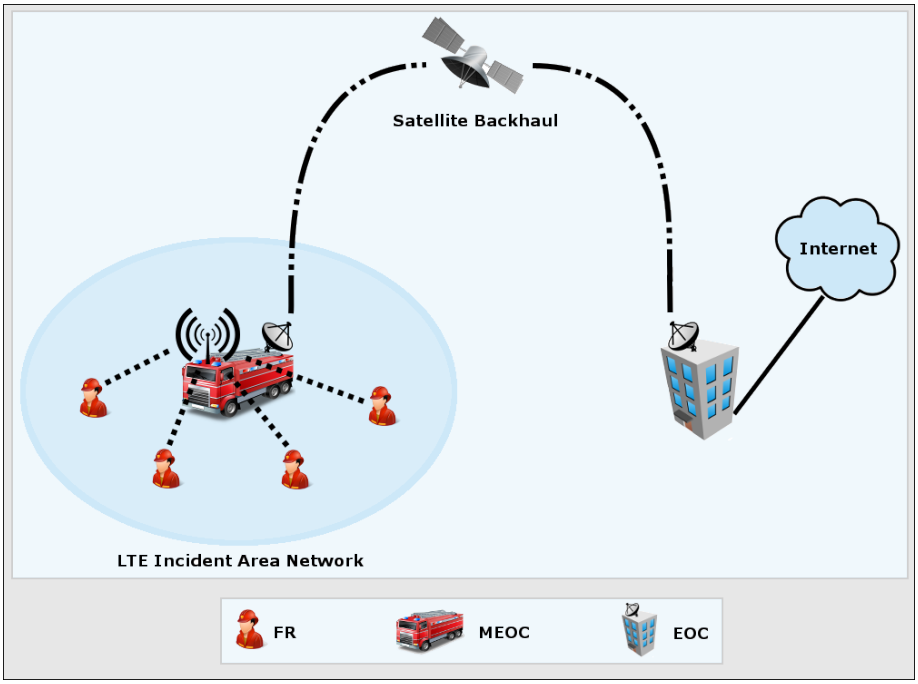


Figure 2.1: Single Reference Deployment

coverage through LTE base stations, as depicted in Figure 2.1. In this model, the Mobile Emergency Operations-control center (MEOC) provides First Responders (FRs) with a LTE Incident Area Network (IAN), thus representing a deployable (and mobile) LTE repeater station for field operators. A reliable satellite link serves as the backhaul medium between the MEOC and the Emergency Operations-control Center (EOC), which is non-mobile and represents the operations headquarters, in order for the former to be able to communicate with the latter independently from the geographical position.

We consider the Incident Area Network provided by the MEOC to be an "atomic" element with which to compose the ad-hoc infrastructure-less network topology that will be presented now.

The state of network access on the field can be fully described with three cases:

- Persons have no connectivity to MEOCs or terrestrial networks, and are therefore unable to communicate.
- Persons can have connectivity with either a MEOC or a terrestrial base station.

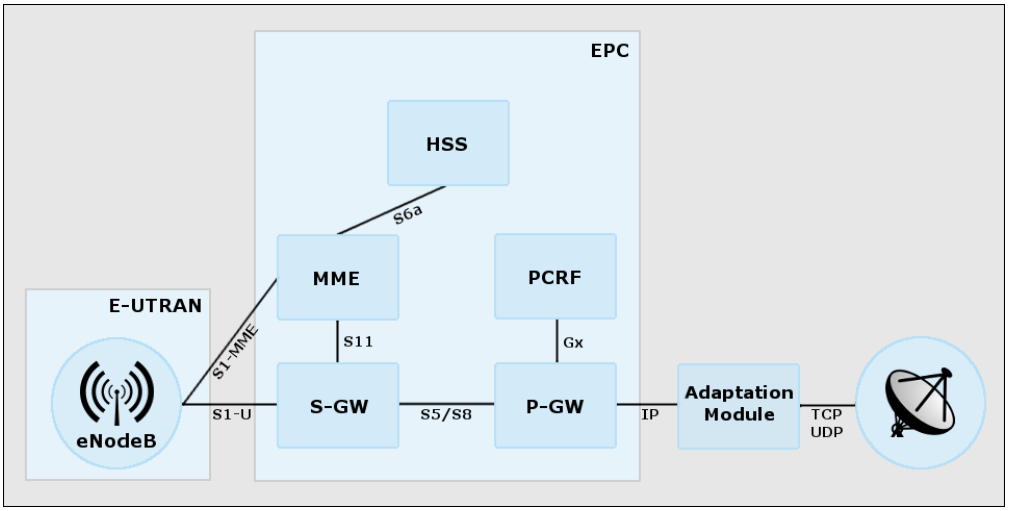


Figure 2.2: MEOC Network Architecture

- Persons can have connectivity with both a MEOC and a terrestrial base station.

These conditions are not necessarily fixed for the entire duration of the rescue operations: a person may shift between them at any time, for example if terrestrial congestion increases/decreases, if subsequent disasters damage network infrastructures or if MEOCs move. It is obvious that the first case must be always avoided.

Figure 2.2 shows the network architecture on a MEOC. By integrating both the E-UTRAN and the EPC subsystems, the MEOC can directly provide IP connectivity to UEs, and in case of necessity it can thus route data through its satellite link. Furthermore, this way no terrestrial core network is necessary, thus avoiding the presence of a single point of failure when a number of MEOCs are deployed.

2.1.2.2 Service Provisioning

People in distress may use their standard UE to ask for help, communicate their status and position, give a contribution to disaster recovery, and communicate with loved ones. If for whatever reason they are not able to rely on commercial terrestrial networks, they must connect to a MEOC when it provides them with

LTE coverage. If, on the other hand, FRs use the terrestrial commercial networks as today often happens, they will congest them even more, especially in the presence of prioritization mechanisms; therefore, this option must be used as a last resort, e.g. if they are deployed in an area covered by a MEOC and they subsequently move out of its coverage. Thus, connectivity (voice and text messaging at least) must be provided to people in distress that may be trapped, injured or dying and additional services (specific voice capabilities, text messaging and broadband data support) must be provided to First Responders. The latest LTE standard releases already contemplate the provision of services such as DMO and Group Calls [7]; in addition, external platforms such as the IP Multimedia Subsystem (IMS) or service enablers defined by the Open Mobile Alliance (OMA) can be employed to provide advanced voice services through LTE. For what regards broadband data support, the optimal solution is for FRs to have a dedicated channel (i.e. high priority) with the MEOC via LTE that provides broadband data capabilities; MEOCs network capacity in excess (i.e. low priority) can be therefore provided for people in distress. MEOCs have thus a twofold function: they provide a dedicated PPDR network for FRs and additional network capacity for people in distress. This prioritization can be realized through LTE bearers. A bearer is an IP packet flow that defines a specific quality of service (QoS) between a gateway and a UE. A user can be associated with multiple bearers; this has been done in order to provide different levels of QoS for different streams. For example, a FR might be engaged in an emergency voice (VoIP) call while at the same time performing a file upload. A VoIP bearer would provide the necessary QoS for the voice call, while a best-effort bearer would be suitable for the file transfer.

The same concept applies for different users, i.e. it is possible to assign by default dedicated bearers to FRs and best-effort bearers to civilians. This is possible because, as part of the procedure in which a UE connects to a LTE network, the UE is assigned an IP address by the P-GW and at least one default bearer is always established, the parameters of which are assigned on the basis of subscription data retrieved from the HSS. Bearers can be classified into two categories based on the nature of the QoS they provide: Minimum guaranteed bit rate (GBR) bearers and Non-GBR bearers. The former permanently allocates dedicated transmission resources, while the latter does not guarantee any particular data rate.

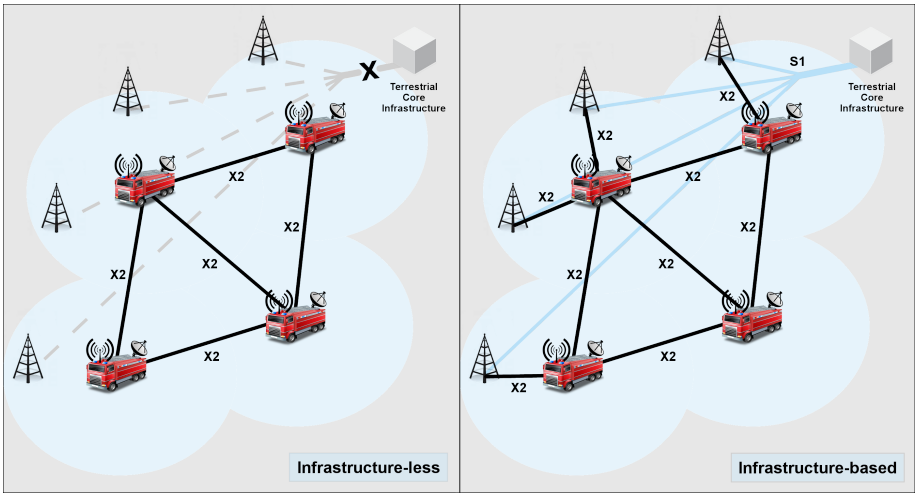


Figure 2.3: Infrastructure-based vs Infrastructure-less Topologies

2.1.2.3 Infrastructure-based and infrastructure-less topologies

Our proposal contemplates the possibility to extend infrastructure-based networks if terrestrial LTE infrastructures are still operational, and to deploy completely infrastructure-less networks otherwise, as shown in Figure 2.3. Normally, the X2 interface is established between one eNodeB and some of its neighbour eNodeBs in order to exchange signaling information. Its initialization starts with the identification of a suitable neighbour, a process that can be manually performed or automatically carried out by LTE Self-Organizing Network (SON) features. Specifically, the Automatic Neighbour Relation Function makes use of UEs to identify neighbor eNodeBs: an eNodeB may ask a UE to read the global cell identity from the broadcast information of another eNodeB for which the UE has identified the physical cell identity (PCI). Once a suitable neighbour has been identified, the initiating eNodeB can set up the Transport Network Layer (TNL) using the X2 IP address of this neighbour, either as retrieved from the network or locally configured. After the TNL has been set up, the initiating eNodeB must trigger the X2 Setup procedure, which enables an automatic exchange of application level configuration data relevant to the X2 interface. This automatic data exchange is also the core of another SON feature: the automatic self-configuration of the PCIs [8]. Once the X2 Setup procedure has been completed, the X2 interface is operational. This way, it is possible to deploy MEOCs that automatically attach to already existing terrestrial eNodeBs, thus extending the terrestrial infrastructure.

In the same way, it is also possible to automatically interconnect different MEOCs in an infrastructure-less topology. When a UE has the possibility to attach to different eNodeBs, such network topologies may be very useful to shift traffic from heavily congested MEOCs to less congested ones, thus augmenting the overall system efficiency, and to provide stronger resilience by the presence of different available paths in order to reach a remote host. An additional LTE eNodeB on each MEOC is an option that may be considered to provide a long-range channel to connect different MEOCs when they are not deployed near each other; without providing connectivity and capacity to users on the field, their range may be much more extended comparing to a typical LTE eNodeB used to provide coverage to people in distress and FRs. This way, inter-MEOC communication can be at least provided. In absence of these dedicated channels, or if MEOCs are deployed even farther, the satellite backhaul channels may be used among MEOCs, although this solution would add the time for two additional satellite hops to the overall transfer.

2.1.2.4 Handover

FRs are expected to move in the disaster area, and their specific movements cannot be predicted in advance. When civilians, on the other hand, are not able to rely on commercial terrestrial networks, they must be able to shift to a MEOC when it provides LTE coverage in their area; furthermore, if terrestrial networks are still operating, it may happen that people in distress try to communicate (e.g. asking for help) through the terrestrial links; however, they may be unable to do it successfully in consequence of network congestion or poor signal conditions (e.g. if trapped in buildings); in these cases, a MEOC in the surrounding can provide them with a better access. Therefore, transparent handover provision is an important requirement, especially in a hybrid network like this. There are two types of handover mechanisms in LTE: the S1 and X2 handover procedures. When a UE moves between one cell and another, handover through the X2 interface is triggered by default, unless there is no X2 interface established or the source eNodeB is configured to use S1 handover instead. Mobility over X2 can be classified according to its resilience to packet loss: the handover can be termed *seamless* if it minimizes the interruption time during the UE movement or *lossless* if it tolerates no loss of packets at all [8]. X2 handover should be performed if the UE is moving between terrestrial eNodeBs. When

moving between MEOCs cells or between a MEOC and a terrestrial eNodeB, a S1 handover procedure must be performed instead. This is because terrestrial infrastructure-based networks are usually realized through star topologies, i.e. there is one Core Network infrastructure for many eNodeBs, while infrastructure-less networks as those provided by the MEOCs contemplate both the E-UTRAN and EPC Core subsystems to be deployed on each node, as previously explained. In these cases, the source and target eNodeBs are served by different MME/SGW nodes, and S1 handover is required. These two different procedures are totally transparent for the user, because from the point of view of its UE their handling by the target eNodeB is exactly the same, regardless of the type of handover (S1 or X2) being used.

2.1.2.5 Spectrum Remarks and Interoperability with Legacy Technologies

In order to enable LTE deployment around the world the technology have been designed to support as many regulatory requirements as possible, and in consequence it is able to operate in a number of different frequency bands. While from a commercial standpoint most European LTE networks are being deployed in Bands 3 (1800 MHz), 7 (2600 MHz) and 20 (800 MHz), EU still lacks a harmonized frequency band dedicated to PPDR purposes. To date, the only accomplished initiatives on the allocation of LTE spectrum for PPDR have been independently pursued by few nations and have a very local scope. In fact, the most significative example in all EU is Greece, where Band 40 (2300 MHz) has been allocated for public safety networks in its five largest cities [9]. This lack of spectrum harmonization for PPDR purposes clearly represents an issue for cross-border operations and Pan-European service provision; furthermore, it may happen that equipment from one country is not able to work in some others. To help to overcome these issues the proposed architecture may allow for a local spectrum harmonization of the deployed IANs, also through on-demand (i.e. by MEOCs configuration) spectrum alignment with the frequency bands used by terrestrial networks in the specific geographical area of the disaster, feature that may ease accessibility, reduce signal interference and allow for cross-border interoperability. In alternative, under the hypothesis of proper EU regulatory enforcements, the hybrid spectrum management approach proposed in [3] may provide PPDR operators with sufficient guarantees.

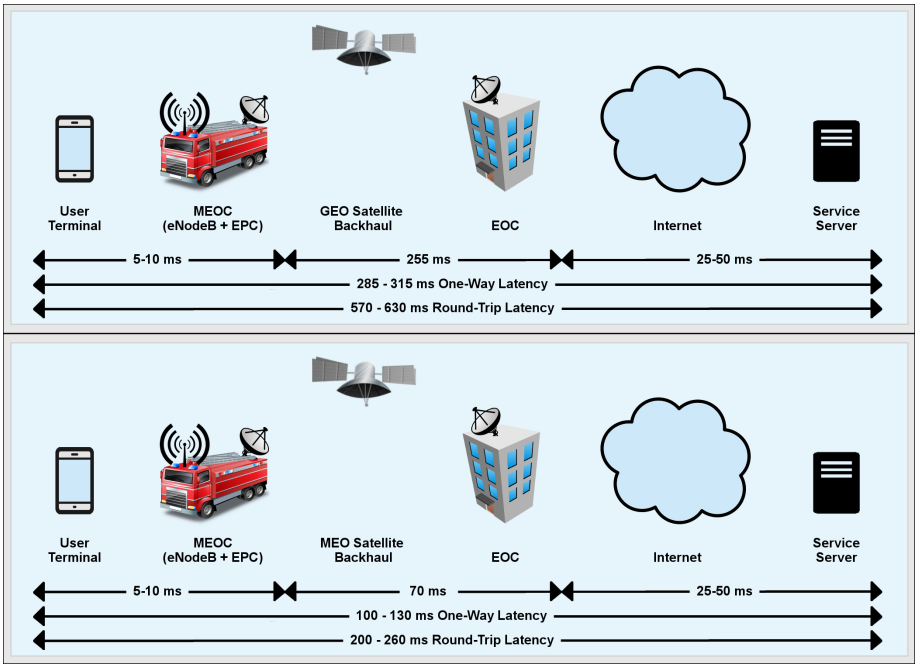


Figure 2.4: End to end latency comparison between GEO and MEO satellite backhauls

Because the legacy PPDR networks listed in Section 2.1.1 are already operational in a number of EU countries and are already under development in others, multi-mode PMR/LTE or TETRA/LTE UE may be adopted by FRs in order to allow the exploitation of existing PPDR infrastructures, where those are deployed.

LTE also supports interworking and mobility (handover) with networks using other technologies, namely GSM, UMTS, CDMA2000 and WiMAX. The S-GW acts as the mobility anchor for interworking with other 3GPP technologies such as GSM and UMTS, while the P-GW serves as an anchor allowing seamless mobility to non-3GPP networks such as CDMA2000 or WiMAX [8]. This allows for interoperability with legacy terrestrial commercial networks, if deployed and operational in the disaster area.

2.1.2.6 Networks range, Data Rate and QoS

In general, the range limit of wireless repeaters is primarily determined by cell size (standard base station, microcell, picocell or femtocell), system configuration, signal penetrability and expected number of users. Theoretically, the

coverage offered by common LTE base stations may range from about 15 km to 100 km [10]. In practice, these values vary. In urban/suburban areas, single cell coverage typically ranges between 0.5 km and 3.5 km; this is because both the propagation loss and the average user density are high. In rural areas, the nominal single cell coverage typically ranges between 25 km and 50 km; however, the actual coverage is easily affected by environmental conditions that rise the propagation loss (e.g. when obstacles as hills are present between eNodeBs and UE). Considering the peculiarities of a generic emergency situation, the main factors that hinder coverage of a deployed LTE cell are the possibility of high propagation losses (especially when the disaster strikes an urban area) and the certainty of a very high user density, both in the spatial and temporal domains. Therefore, we consider ranges between 0.5 km and 1.5 km in urban areas, between 1 km and 2 km in suburban areas, and between 1 km and 10 km in rural areas to be appropriate estimations for a deployable LTE eNodeB to be used in emergency scenarios. Satellites, on the other hand, are theoretically able to provide global coverage. The actual covered zones depend mainly on their orbit type:

- Geosynchronous Orbit (GEO) satellites are the solutions that offer the greatest coverage (a single one may cover around 42% of Earth's surface); they also have the advantage of remaining permanently in the same area of the sky, and thus ground-based antennas do not need to track them but they can remain fixed in one direction. Unfortunately, being that these satellites are very far from Earth (they orbit at 35.786 km above the Earth's equator), they exhibit poor latency and data rate performance, caused by a very high propagation delay.

- Medium Earth Orbit (MEO) satellites orbit at altitudes comprised between 2000 km and 35.768 km. A greater number of satellites is necessary to provide a coverage comparable to GEO solutions. The lower orbit, however, allow for better transmission performance.

- Low Earth Orbit (LEO) satellites orbit at altitudes comprised between 160 km and 2000 km. This solution can provide high bandwidth and low propagation delay. However, an even greater number of satellites must be put in orbit in order to offer a coverage that is comparable to GEO and MEO solutions.

For what regards data rate, LTE can theoretically reach an uplink value of 75 Mbit/s, while 326 Mbit/s can be reached in downlink with a 4x4 MIMO antenna without error control coding, value that descends to 226 Mbit/s with rate 5/6 error control coding. However, simulations and trials have showed [11] that realistic LTE average throughput values stop between 17 Mbit/s and 33 Mbit/s. The data rate provided by a satellite backhaul depends primarily on its orbit type; because propagation delay plays a key role in the realistically achievable data rates, GEO satellites usually offer the least performance. We consider as the reference for latest MEO systems the O3b solution [12]; it uses the Ka-band frequencies between 17.8 and 19.3 GHz in the downlink and between 27.6 and 29.1 GHz in the uplink. These satellites compose a MEO constellation at a height of 8063 km, in which a subset of them is kept for redundancy purposes. Each satellite circles the earth within 4 hours and, for a fixed point on Earth, a new satellite rises every 45 minutes. O3b uses a make-before-break mechanism to ensure seamless satellite handover, in which a ground terminal temporarily enjoys a simultaneous connection to two satellites. Each satellite possesses 12 spot beams, 2 of which are reserved for the link to the gateway while 10 are left for the terminals. Each spot beam can have a bandwidth of up to 216 Mhz and is able to deliver a data rate of up to 1.2 Gbit/s. To achieve such high data rates, the satellite dishes of terminals have to have a diameter that is between 3.5 m and 4.5 m, facilities that cannot be embedded in a UE but can be installed on a MEOC. There is also an increasing interest on the usage of Ka-band frequencies for future LEO constellations. L-3 Communications estimated that data rates up to 3.75 Gbit/s should be achievable between a LEO satellite and a ground station [13]. However, to date no LEO constellation able to offer broadband capacity is operative.

For QoS purposes, the overall latency plays a key role, especially when considering the usage of satellites as backhaul means. For the LTE access part, thanks to its IP-based architecture, the initial packet data connection typically takes around 50 ms, and between 10-15 ms of round-trip latency is needed for subsequent transfers. The propagation delay between the eNodeB and the UE does not pose any issue to delay-sensitive applications; in fact, even at a distance of 100 km, the eNodeB needs less than a millisecond to reach the UE. At that distance, however, performance would be significantly impaired by the propagation loss. It is worth to notice that once the connection has been established, the UE can move farther away and still maintain communication capabilities, as long

as both ends of the link can reach each other. On the contrary, a GEO satellite backhaul takes *at least* 500-550 ms for a Round Trip Time (RTT), whereas with a MEO solution this value can be reduced to an average of 200 ms [14], depending on the orbit height. LEO satellites typically need less than 100 ms for a RTT, also depending on the orbit height, which would make them the satellite solution that offers the best propagation delay; however, because of the data rate deficiency stated above, current LEO constellations may only be considered as part of the legacy PPDR systems described in Section 2.1.1. An estimation of the total average latency needed for a complete, remote connection made through the LTE subsystem backhauled by satellite is represented in Figure 2.4. In it, the GEO propagation delay is optimistic on purpose, resulting in a mean RTT of 600 ms with GEO satellites and of 230 ms with MEO satellites. Especially in the PPDR context, where stringent QoS guarantees are often needed, the total latency experienced by the user is a very important parameter.

The usage of satellites as backhaul mediums implies that the Transport Protocol in the network stack is another QoS key factor. More specifically, QoS is strongly affected when Transport Control Protocol (TCP) is employed, as it is often the case in PPDR operations where reliable, in-order and error-checked stream delivery is often desirable if not necessary. In fact, while also in PPDR operations UDP is preferred for general video streaming, a good number of applications require reliable data delivery [1]. It is very well known that the standard TCP protocol performs badly when employed on links with high propagation delay. Although a complete analysis of Transport-Level protocols on satellite links is out of the scope of this first Section (it will be deeply investigated in the following Sections), we note that the use of tailored solutions [15] has the potential to mitigate these issues. If GEO satellites are employed, very specific TCP protocols or TCP accelerators should be used; instead, with MEO and LEO solutions it is possible to maintain the usage of common TCP protocols (e.g. TCP Cubic) with a limited QoS degradation [14].

In the proposed architecture, the identified bottlenecks are therefore:

- The LTE access subsystem for what regards coverage.
- The LTE access subsystem for what regards data rate if it is backhauled by MEO satellites, while the situation may invert if GEO satellites are employed.

- The satellite backhaul subsystem for what regards latency QoS.

Overall, new-generation Ka-Band MEO satellite constellations offer a high-performance and valuable medium to backhaul LTE connections for PPDR purposes.

2.1.2.7 Indoor and Tunnel Network Provision

In PPDR operations, it is not infrequent to meet an indoor or tunnel scenario. In order to provide coverage in those cases, the proposed architecture allows for two different approaches. In general, the FRs UE should embed the possibility to act as repeaters, in order to extend network range and to create a redundant number of network paths available to the MEOCs, so as to advantage other FRs or people in distress.

The first option is to deploy picocells and femtocells inside buildings and tunnels, respectively; these may be directly connected to a MEOC, to other eNodeBs that in turn reach the MEOC, or to some FRs UE acting as a bridge. The second option is to create a network chain by deploying the necessary number of FRs that form a network path from the MEOC to inside the building or tunnel; this way, their UE may provide connectivity indoor and traffic may be routed outside through multiple hops.

2.1.3 Conclusions

To address the vulnerabilities of terrestrial infrastructure-based networks in major incidents and disaster scenarios, we proposed a hybrid network architecture that integrates LTE and satellite technologies for PPDR purposes. It is based on deployable mobile units that bring LTE coverage to the disaster area through a satellite backhaul. The architecture is designed in order to provide easy connectivity, extended coverage and high performance guarantees. Furthermore, it allows for both infrastructure-less and infrastructure-based service provision, without requiring extensive configuration. This way, existing dedicated and commercial terrestrial infrastructures can be leveraged, and at the same time sufficient reliability from unexpected events can be provided. We conclude by noting that the proposed architecture has the potential to permit interoperability with

legacy cellular technologies, to ease cross-border operations and to provide communication in disasters that include both outdoor and indoor scenarios. The following Sections provide several simulations with the ns-3 network simulator in order to find bottlenecks and optimization opportunities starting from this reference architecture.

2.2 C^2ML: Congestion Control Middleware Layer

Wireless communications technologies are critical for public protection and disaster relief (PPDR) professionals, who require services like basic voice communication and its extended features, such as Push-To-Talk, Group Calls and Direct Mode capabilities. For several years, data-intensive applications have also been strongly requested by the PPDR community during the emergency operations that follow natural or man-made disasters, scenarios in which commercial terrestrial networks often fail to provide the necessary support. The reason is threefold: they simply get disrupted by the disaster, they cannot sustain the sudden surge of network demand or they fail to deliver the necessary bandwidth and/or other QoS guarantees. For these scenarios satellites offer a reliable and almost ubiquitous communication channel and have been proposed as a backhaul solution also for TETRA networks [16]. The major issues that arise when trying to deliver services through satellite technologies or, in general, through high-delay links are related to the "reliable" transport-level protocol, TCP. This happens because of two phenomena: the high Round Trip Times (RTTs) and the often too large buffer sizes, which have a bad influence on its algorithms and related assumptions.

Standard TCP performs congestion control through the Van Jacobsons algorithm of Slow Start with Congestion Avoidance, coupled with exponential backoff [17]. When the delay is high, the Slow Start algorithm phase leads to an initial underutilization of the available capacity, while both link asymmetry and variable RTT further affect connection throughput when using the additive-increase/multiplicative-decrease Congestion Avoidance algorithm. Several TCP variants, as well as TCP accelerators [18], have been implemented so far to mitigate these problems, however, when several hosts are competing for the same link, fairly distribute the bandwidth among the flows still represents a challenging task. Intra-protocol fairness and inter-protocol friendliness properties play a crucial role in emergency networks, because priority flows should be carefully designed and chosen by the network administrator rather than imposed by congestion control algorithms.

At the same time, there is another figure of merit for the network: latency. It is a fundamental parameter for public protection and disaster recovery network purposes, and in this perspective our focus is at network and transport lever

rather than at application level. When TCP was designed, it employed the idea of pipe size and the correct observation that there was reasonable but not excessive buffering along the data path through which to send a window of packets. Buffering is necessary because, unless sources are synchronized, packets may arrive in bursts taking some time to get routed. The increasing number of links with high delay and the reduction of memory costs have in general led to an increase in buffer sizes, deployed to overcome TCP losses at bottleneck links. Unfortunately, this is seen by TCP as another pipe in series with the link; the protocol will thus try to fill it increasing the delay after the bottleneck capacity is reached. In fact, even in terrestrial networks, the existence of excessively large and nevertheless frequently full buffers may lead to an increase in overall delay. This phenomenon has been called "Bufferbloat" [19]. One of the main goals here, coupled with a low network latency, is to avoid expensive QoS operations in routers or wireless access points, because they should be fast to deploy and setup during real crisis or disasters (avoiding any delay due to configuration) and lightweight enough to be run without high-energy sources.

In this Section, we analyse standard TCP as well as protocols specifically designed to overcome issues on links with high delays, and investigate both their congestion control algorithms as well as the - sometimes excessive - buffering they require to provide good performance on this kind of channels. Results show that a "cooperative" transmission control protocol has the potential to mitigate the aforementioned issues. Therefore here we propose a middleware layer, called C^2ML, to optimize the transport layer performance over high delay links. Our simulations through the ns-3 network simulator show that C^2ML effectively improves throughput and fairness, leading also to a better network-level buffer usage and, as a consequence, to a remarkable reduction of the overall latency, avoiding buffers to be bloated by uncontrolled streams. Our proposal requires sender-side modifications only, along with a central gateway that manages bandwidth allocation for competing nodes over satellite links. In particular, in Section 2.2.1 we summarize related work. In Section 2.2.2 we describe our reference environment, along with its issues, while Section 2.2.3 describes C^2ML. Numerical results are reported in Section 2.2.4, and in Section 2.2.5 we summarize the conclusions.

2.2.1 Literature Overview

Active queue management (AQM) schemes, such as RED [20, 21], have been around for well over a decade and could potentially solve the aforementioned problem, also reducing queuing delay. RFC 2309 strongly recommends the adoption of AQM schemes in the network to improve the performance of the Internet. RED is implemented in a wide variety of network devices, both in hardware and software. Unfortunately, due to the fact that it needs a careful tuning of its parameters in relation to various network conditions, most network operators do not turn it on. In addition, it is designed to control the queue length which may implicitly affect delay. Recently, a new AQM scheme named CoDel [22] has been proposed to address the bufferbloat problem by directly controlling the latency, but its requirements consume excessive processing and infrastructural resources, yielding CoDel expensive to implement and operate, especially in hardware. These issues are reported in [23]. The entire class of solutions based on AQM schemes can not be comprehensive enough to definitely solve the bufferbloat problem, despite a reduction in queuing delay, especially in channels with high bandwidth-delay product (BDP); this is because they all become eventually prone to instability, which leads to lower performance [24].

When the bottleneck is known, another interesting approach is to cooperatively adapt the transmission rate between peers of the same bottleneck link. As an example, in a mobile wireless scenario the throughput is deteriorated by traffic congestion when multiple devices share one single access point. This happens because the traffic control mechanism implemented in TCP is not designed to coordinate them; instead, it is designed to control the traffic only in an end-to-end manner for each connection. A Cooperative Transmission Control Middleware [25] has been proposed to adjust the congestion control in such cases by exchanging information with broadcast messages among devices in the same wireless LAN. As a result, both communication throughput and bandwidth utilization fairness are improved. Anyway, the fact that the work is tailored over TCP Cubic [26] and smartphone devices in a wireless LAN environment makes it hard to adapt to emergency networks.

In order to overcome the problems mentioned in the previous section, a number of Transport Protocol variants have been proposed [27–31]. Floyd *et al.* [27] introduced the High Speed TCP algorithm (HSTCP), which considerably increases the efficiency of networks with high BDP. However, HSTCP does not

consider RTT fairness and friendliness with standard TCP flows. Kuzmanovic *et al.* merged HSTCP with TCP-LP [32] in a protocol [28] that aims to enable a simple two-class prioritization for low-priority background file transfer over high-speed networks, so as to use only the excess bandwidth for low-priority flows as compared to the fair-share of bit rate as targeted by other TCP variants. Kelly [29] proposed Scalable TCP (STCP), as an alternative to HSTCP, to solve the data-transfer efficiency problem in long fat networks. STCP uses multiplicative increase and multiplicative decrease strategy (MIMD) instead of additive increase and multiplicative decrease (AIMD) to adjust the congestion window for each coming RTT. The proposed modification scales well in many environments. However, STCP flow usually moves the network to a state of nearly constant congestion which is generally undesirable for most networks [33]. TCP-Peach [30] is based on the replacement of slow start and fast recovery algorithms with sudden start and rapid recovery procedures, which rely on the introduction of dummy segments to probe the bandwidth availability of the network. TCP-Peach requires all the routers along the connection to implement some priority mechanism at the IP layer, in order to discard dummy segments in presence of congestion. Caini and Firrincieli proposed Hybla [31] to solve the RTT-unfairness problem in heterogeneous networks composed by both low and high BDP channels, by removing the performance dependence on RTT. TCP Hybla exhibits excellent fairness and friendliness properties, and effectively enhances performance of connections with high RTTs. However, it tends to introduce inefficiency regarding channels full capacity utilization, thus not allowing connections to fully exploit the link potentials.

Another approach is the eXplicit Control Protocol (XCP) [34], a novel protocol for congestion control that outperforms TCP in conventional environments and remains efficient, fair, and stable as the link bandwidth or the round-trip delay increase. It generalizes the Explicit Congestion Notification (ECN) [35], informing senders about the degree of congestion. The protocol is stateless, as no router along the path should maintain a per-flow state, and it can thus scale to any number of flows. Unfortunately, each router over the entire end to end path plus the receiver should be XCP-aware. Furthermore, it has been shown [36] that the scheme fails to achieve max-min fairness and leads to under-utilization of the network in scenarios with multiple congested links, and it also presents oscillations in cases of large delays. Adaptive Congestion Protocol (ADP) [37] has been designed to outperform both TCP and XCP, even in the presence of

random packet losses, in traffic scenarios comprising of a small number of long flows and a large number of short flows. Unfortunately, it also forces routers over the entire end to end path to be ADP-aware. A separate work [38] focuses on max-min fair allocation of bandwidth with minimum and maximum rate constraints, but it is oriented to multirate, multicast scenarios. In [39] authors propose a scheme (REFWA) that allows satellite systems to automatically adapt to any change in the number of active TCP flows due to handover occurrence, free buffer size and BDP. The system efficiency is controlled by matching the aggregate traffic rate to the sum of the link capacity and total buffer size, while max-min fairness is achieved by allocating bandwidth among individual flows in proportion with their RTTs. An important point is that this scheme requires no modification to TCP implementations in the end systems. However, the proposed congestion control method is specifically designed for multi-hop Low Earth Orbit (LEO) satellite networks and it doesn't take into account Medium Earth Orbit (MEO) or Geostationary Earth Orbit (GEO) solutions, which are characterized by higher delay.

Being in the context of satellite-based emergency networks we focused on TCP Noordwijk (TCPN) [40], a new burst-based TCP variant for satellite links, which requires only sender-side modifications. Authors aim at satisfying three particular requirements, overcoming issues of standard congestion control algorithms over high BDP channels and consequently maximizing throughput:

- To optimize the transmission of short objects;
- To guarantee a fair behaviour with competing flows and efficient resource sharing in case of *large* transfers;
- To efficiently operate over all DVB-RCS DAMA schemes [41].

TCPN replaces the standard "window-based" transmission with a "burst-based" transmission. The transmission of packet bursts is characterized by two variables: burst size, which is the number of segments to send in one shot, and burst transmission interval, which is the time between two consecutive transmissions. Both variables are updated according to ACK-based measurements, i.e. *ACK Dispersion* and *RTT Variation*. TCPN has also some drawbacks: it works under the assumption of a controlled environment with well known characteristics and it does not guarantee a fair behaviour with competing flows and

the efficient resource sharing in case of *very* short transfers, resulting in flow oscillations. Also, its queue usage is very large.

Therefore, the proposed cooperative transmission control middleware with centralized congestion management will be based on the following key points:

- To be independent from the AQM system;
- To enhance transport layer performance in throughput, latency and fairness terms;
- To require sender side modifications only;
- To provide fairness among flows for both *large* and *short* transfers.

Moreover, it is thought to be easily adapted to different TCP algorithms and to different congestion control mechanisms, to make it as much flexible as possible.

2.2.2 Environment and background issues

As ground setting for our work we selected from the EU FP7 Project "Public Protection and Disaster Relief - Transformation Centre" the scenario depicted in Figure 2.5, which is composed by two hierarchical tiers. The top one (Tier 1 in the Figure) comprehends a Mobile Emergency Operations-control Center [42] (MEOC) that provides, through the connectivity guaranteed by a reliable satellite link, an ad-hoc network to the child tier (Tier 2 in the Figure); this is composed by other MEOCs that in turn create radio coverage for First Responders (FRs), deploying this way Incident Area Networks (IANs) in the mission field. The specific radio technologies shown in the Figure have been inserted only by way of example. Figure 2.6, instead, represents the logical high-level architecture of a generic tier.

It is important to highlight that our proposal can be generalized for every challenging scenario in which clients may use a central gateway to access a bottleneck link, as may be the case of a home wireless LAN connected to the Internet via ADSL, or a wired network in a building where the external connection is guaranteed by a satellite link; in cases where the bottleneck is due to a satellite channel our proposal allows to avoid the use of Performance Enhancement Proxies.

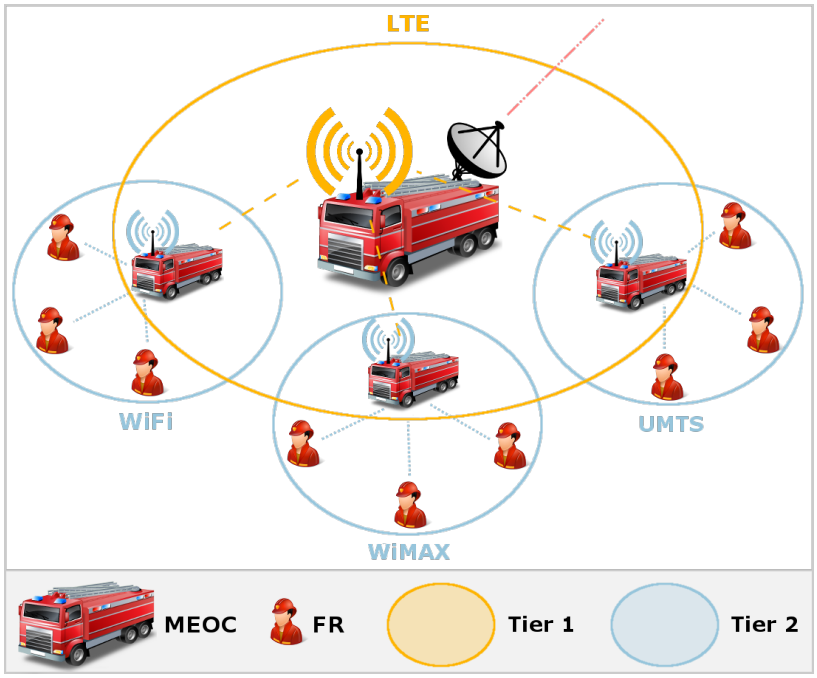


Figure 2.5: Emergency Network scenario

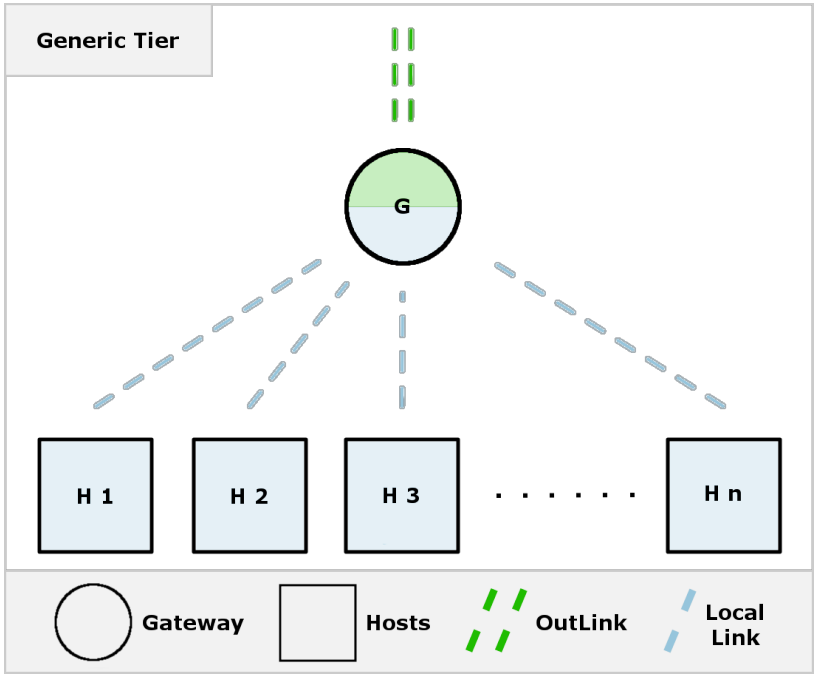


Figure 2.6: Generic Tier

2.2.2.1 Overview of the experimental setup

In order to assess performance we simulated the scenario shown in Figure 2.7. This scenario represents a network composed by one tier only, as it is simpler to analyse yet comprehensive enough to model more complex scenarios as the one depicted by Figure 2.5. In the simplest case, in fact, additional tiers can be modelled through additional routers between client hosts at the end of the tree hierarchy and the root gateway.

All tests have been performed on the ns-3 network simulator. The satellite link has been modelled as a point-to-point channel with a delay of [300-350] ms and a bandwidth in the range $[1, 4]$ Mbit/s (depending on the simulation). Regarding the gateway buffer size, two configurations have been introduced: the first, called *standard*, sets a buffer size equal to the BDP of the channel; the other, called *large*, sets instead a fixed buffer size of 4 MB. As claimed by the DVB-RCS standard [43], the link is dimensioned to work in quasi-error-free conditions (Bit Error Rate less than 10^{-10}); therefore, no error model has been added to the simulated link. In the simulated scenario, point-to-point links connect multiple local nodes to the gateway; each link is configured with the same bandwidth of the bottleneck link and a propagation delay of 0 ms, since we do not want to take into account the local network delay, focusing only on the bottleneck link. This does not invalidate the simulations with regards to reality, because the same results are obtained moving 10 ms of delay from the bottleneck to the clients (therefore having a 290 ms propagation delay over the bottleneck link and a 10 ms propagation delay over the client connections). The gateway is in turn connected to remote nodes with the aforementioned satellite link; each local node runs an application that transmits TCP segments to a certain remote node, where there is a sink application that receives them, and send related ACKs.

With regards to TCP/IP stack, we modelled the receivers not to send cumulative ACKs in case of TCP Noordwijk simulations (as its reference model demands), but let other TCP to send one ACK every two segments, offloading a bit the return channel. Other parameters, like initial congestion window and initial slow start threshold, are respectively 10 segments [44] and 5000 bytes. For TCP Noordwijk, we set a default burst size of 20 packets with a default transmission timer of 500 ms (as its reference model demands) but decreased them to 5 packets and 50 ms respectively when employing our proposal. Also, we enabled all optional TCP extensions available on ns-3 which could help in

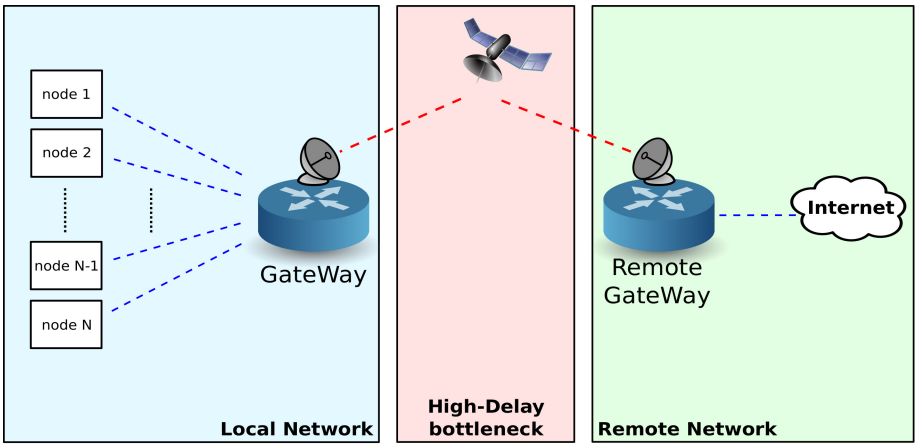


Figure 2.7: Simulations scenario

high-delay links, as the TCP window scaling and timestamps option, and we chose as MSS for all TCP variants a fixed value of 500 bytes. With this setup, the bottleneck is clearly the satellite connection.

2.2.2.2 TCP performance over satellite links

TCP, originally designed for terrestrial networks, suffers when applied to a satellite communication environment due to the different channel characteristics. In the following experiments we run nodes with the *large* buffer size and a bottleneck bandwidth of 1 Mbit/s.

In Figure 2.8 we show how TCP NewReno [45] performs in our experimental setup. There are three nodes, and each one have to transfer a 5 MB file to a remote host. To see how flows interact, we start each node with a 10 second delay from the previous one: Node 1 starts at the beginning of the experiment, Node 2 starts after 10 seconds of simulated time and finally Node 3 starts after 20 seconds of simulated time. We can see that the third flow is penalized, compared to the other two already saturating the channel. The third flow experiences an high number of losses in the initial phase of the transmission. Mixed with the high latency, up to two orders of magnitude larger than latency in terrestrial networks, this drastically slows down the TCP congestion window ($cWnd$) growth: this means that most of the typical TCP transfers (from dozen of kilobytes up to several megabytes) conclude before TCP reaches its optimal window. This implies a huge waste of potentially available network capacity.

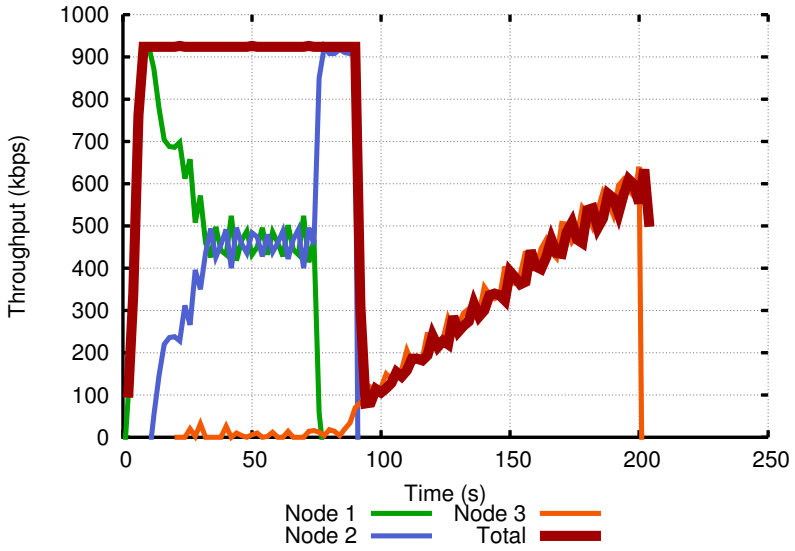


Figure 2.8: Three nodes, TCP NewReno, file transfer, *large* buffers

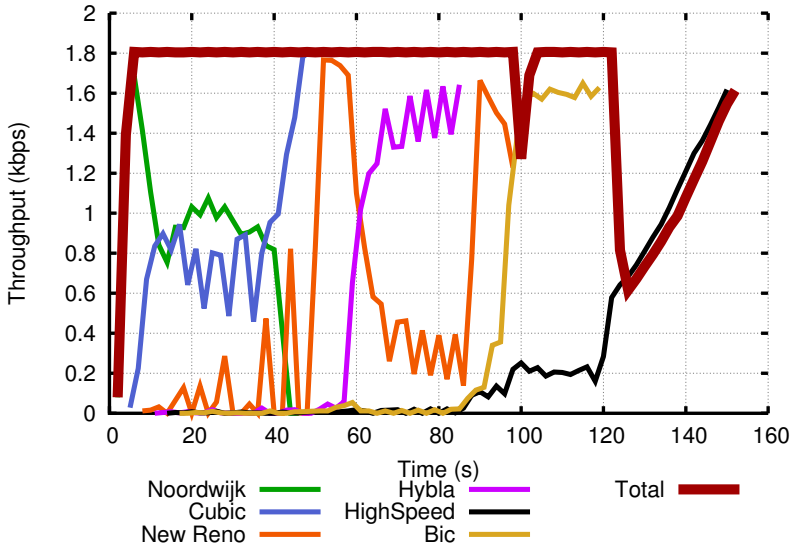


Figure 2.9: Friendliness between different TCP flows, *large* buffers

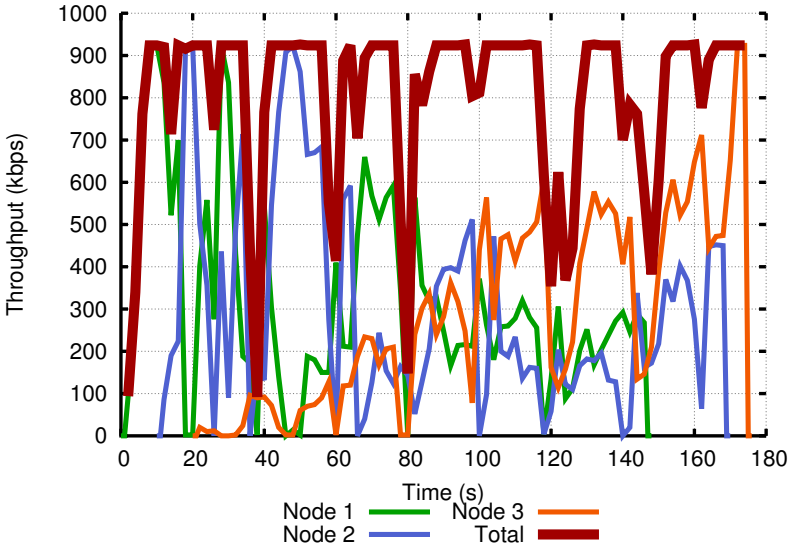


Figure 2.10: Three nodes, TCP Cubic, file transfer, *standard* buffers

In Figure 2.9 we show how six TCP flows interact. We raised the available bandwidth to 2 Mbit/s, with a delay of 300 ms and used for each node the following TCP versions: Noordwijk, Cubic, NewReno, Hybla, HighSpeed, and Bic; each node starts 3 seconds after the previous one. Besides, the same Figure shows how these six different TCP algorithms fail to achieve fairness balance. Aside from the well-known issues of TCP in high-delay links, the unfairness is partially due also to the AQM scheme used, Drop-Tail, which is the only AQM which can be employed when using TCP Noordiwjk (we will detail this statement later in the Section).

2.2.2.3 Buffer impact over TCP congestion controls

In this section, we want to show how buffers along the path influence the behaviour of some congestion controls with respect to throughput, while latency is investigated later in Section 2.2.4. TCP algorithms are all affected by losses in environments with high delays, as the recovery time can be very high (also with techniques like Fast Recovery or Fast Retransmit). We repeated the previous experiments to see the behaviour of the different congestion controls with *standard* buffer size, using Drop-Tail as queue management algorithm.

In Figure 2.10 we can see TCP Cubic performance, whose flows affected by queue-drops unfairly reduce their rate, allowing those not affected to continue

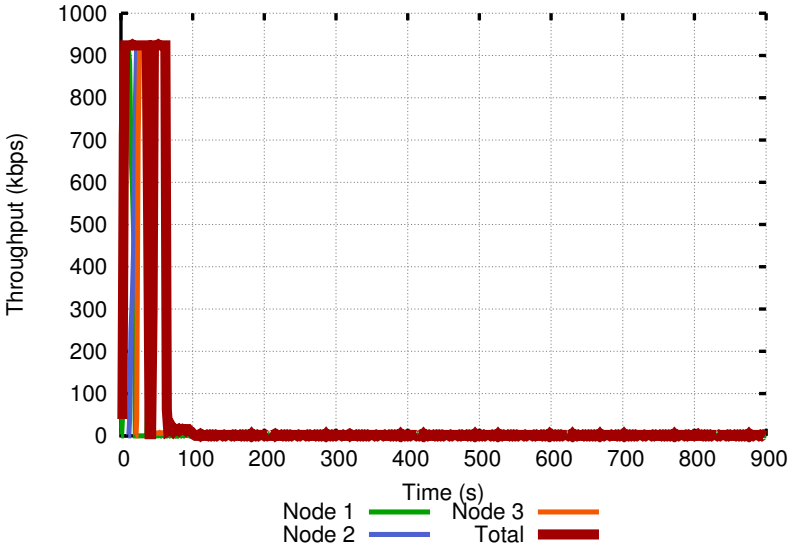


Figure 2.11: Three nodes, TCP Noordwijk, file transfer, *standard* buffers

and also to steal some portion of bandwidth, leading to a poor fairness among flows.

A worse situation holds for TCP Noordwijk, as we can see in Figure 2.11, because it assumes a well-defined environment with no losses, relying on queuing capability of the I-PEP [46] interface. Consequently, its behaviour in the reference model is undefined when losses arise. In particular, the protocol loses the timing information of ACKs due to retransmissions (or fast retransmissions) triggered by duplicated ACKs or RTO expirations, and thus it can not regulate itself or recover from multiple loss events. It is, for instance, the main reason which lead us to avoid the use of any AQM schemes in simulations: by early dropping packets (i.e. the RED or CoDeL case) TCP Noordwijk performance starts to worsen early in simulations, making the comparison unfair.

At this point, our goal is to introduce a system that, while maintaining an high overall throughput, helps both fairness and friendliness at the same time with a low buffer usage to minimize the overall latency.

2.2.3 C^2ML Design and Implementation

In this section we describe C^2ML: Congestion Control Middleware Layer. It is composed by a new network layer which manages the flows congestion control

and a protocol that distribute information about the status of the network. Firstly, we focus our discussion on the design and the key principles, and then we show its implementation. Source code, with related scripts, is available [47].

2.2.3.1 C^2ML Design

In the reference scenario a gateway provides the access to the bottleneck link and it is aware of its characteristics (e.g. channel bandwidth). In such an environment, a practical and effective way to reduce queuing delay and improve fairness and throughput is to use a cooperative adjustment of the transmission rate in order to match the bottleneck link capacity.

To distribute the congestion control management to the set of nodes and gateway, the addition of a layer (that we call Congestion Control Layer, C^2L) in the network stack of nodes and gateway is required. As a key principle, we allow only sender-side modifications, in order to allow remote hosts and routers to run without any modification. C^2L, which acts as middleware between the application and the transport layer, should intercept all socket creations and it can be seen as a logical channel between gateway and nodes. Therefore, in order to cooperatively adjust the transmission rate, a protocol is also required to regulate communications. Its focal point is the update of the active nodes rate when another node enter or exit the network. In fact, informing all other nodes about a new entry permits to avoid channel congestion thus leading to an improved fairness (as flows know their maximum allowed rate avoiding to compete with each other in a cooperative environment) and to a reduced buffer usage, as we will show. On the other hand, informing all other nodes about a leaving permits the increase of the remaining nodes maximum allowed rate, thus obtaining higher throughput for each one. We call this protocol C^2LP, Congestion Control Layer Protocol.

In the emergency network field, the resilience is a key factor of the network, so redundancy of gateways should be always considered. Anyway, if for any reasons C^2LP can not be run, the nodes must fall back to the old behaviour (and so, disabling C^2ML at all) in an application-transparent way.

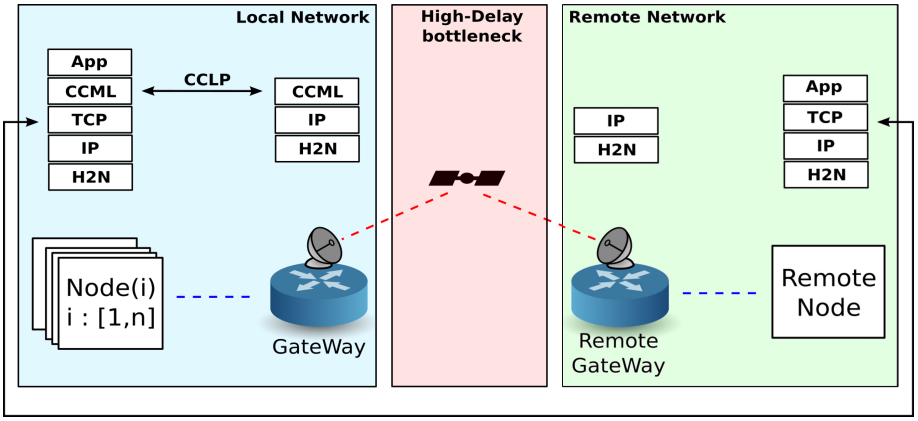


Figure 2.12: C^2ML scheme.

2.2.3.2 C^2ML Implementation

The scheme of C^2ML is depicted in Figure 2.12. Nodes establish a session with the gateway (implemented as a standard TCP session from nodes to gateway, in order to avoid middle-boxes interferences), where the following control messages can be exchanged:

- `CLIENT_HELLO`, used by a node to inform the gateway it wants to send data;
- `CLIENT_BYE`, used by a node to inform the gateway it has finished the transmission;
- `AVAIL_BW`, used by the gateway to inform nodes about their available bandwidth.

When a node joins the network, it opens a control connection with the gateway. Only when the first application socket is created it sends the `CLIENT_HELLO` packet, while the `CLIENT_BYE` packet is only sent when the last application socket is closed. This way, the bandwidth sharing among the flows within a node is relegated to the node itself, triggered the first time when an `AVAIL_BW` packet is received. Such packet contains a parameter with the node maximum allowed bandwidth; the node itself is responsible to divide the value among the sockets, by evenly splitting it or prioritizing some flows over the others depending on the flow type, but ultimately the node should not over-exceed the assigned bandwidth regardless the socket number. This paradigm shift (actually, the TCP

distributed congestion control works naturally on a per-connection rather than a per-node basis) is required from PPDR community, in order to avoid an excessive bandwidth usage from nodes, malicious or not, that open several parallel connections.

The gateway knows the total bandwidth (BW) of the bottleneck link. When it receives a `CLIENT_HELLO` packet (or a `CLIENT_BYE`), it increments (or decrements) the number of active clients, recalculates the bandwidth allowed for each node and then sends n `AVAIL_BW` packets to the n connected nodes. In the same way, if the gateway detects a variation of the bottleneck link bandwidth, it recalculates the bandwidth subdivision and sends the appropriate values to its n client hosts. A simple but effective algorithm for bandwidth subdivision is the *Unweighted Fair Budget* one: the gateway assigns at each node i

$$BW_i = \frac{BW}{n}$$

where n is the number of active nodes. Other bandwidth management algorithms could also be employed without affecting the entire system, but the following invariant property must be respected: the total budget assigned to nodes is less or equal to the total available bandwidth. Not doing so will lead to congestion resulting in throughput, fairness and perceived delay degradation. We also provided a dynamic algorithm, as an extension to C^2ML, that deals with nodes which are not able to fully exploit the assigned bandwidth in [48].

The run-time complexity of the algorithm for clients is $\mathcal{O}(1)$ when joining or leaving the network, while for the gateway it depends on the bandwidth management algorithm chosen. When *Unweighted Fair Budget* is used, the run-time complexity is $\mathcal{O}(1)$. The communication complexity worst case is $\mathcal{O}(n)$ `AVAIL_BW` messages (where n is the number of operative nodes in the network) sent from the gateway to node themselves when one of them joins or leaves the network. As the message content is the same for all nodes, the communication complexity can be reduced to $\mathcal{O}(1)$ if broadcasting techniques are employed.

2.2.3.3 Adapting existing congestion controls

We chose to adapt TCP Noordwijk (TCPN) to work with C^2ML because of its effectiveness over satellite links, but other TCP protocols could be adapted as well. Conceptually, C^2ML aims at avoiding congestion by giving each node

```
                                  ▷ An AVAIL_BW message is received from the gateway
procedure HandleMessage(Bandwidth bw)
    availBw ← bw
    ssThres ← availBw * rtt
    cWnd ← ssThres/2
end procedure

procedure AckRcv(SeqNum n)                          ▷ An ACK is received
    ssThres ← availBw * rtt
    Cubic::AckRcv(n)
    if cWnd > ssThres then
        cWnd ← ssThres
    end if
end procedure
```

Figure 2.13: Pseudocode of (client-side) TCP Cubic adaptations for C^2ML

the available bandwidth. This perfectly fits the scheme of TCPN which bases its transmission over a timer, i.e. TX_TMR, which when expires triggers the sending of a fixed numbers of segments called $BURST$. Thus, the node i sends over the link an amount of bits per second BW_i (assuming that TX_TMR is expressed in seconds and MSS, the Maximum Segment Size, is in bytes) equal to

$$BW_i = \frac{BURST_i \cdot (MSS \cdot 8)}{TX\_TMR_i}$$

If the bandwidth BW_i (in bit/s) is given by the gateway, each node can adjust the rate for the TCPN scheme in two ways: changing $BURST_i$ or TX_TMR_i to match the given bandwidth utilization:

$$BURST_i = \frac{BW_i \cdot TX\_TMR_i}{(MSS \cdot 8)}$$

$$TX\_TMR_i = \frac{(MSS \cdot 8) \cdot BURST_i}{BW_i}$$

Rate Adjustment and Rate Tracking algorithms, employed by TCPN to adjust $BURST$ and TX_TMR variables in reaction to channel conditions, are still present (in fact, the gateway know neither the network conditions beyond the bottleneck link nor the receiver status), but they are modified in order not to exceed the available bandwidth, with simple conditional controls.

A different situation holds for TCP algorithms which rely on the well-known

congestion window concept. Without any transmission timer, the best these protocols can do is to adjust the congestion window according to the RTT information and the gateway assigned bandwidth by doing:

$$cWnd = \left\lceil \frac{BW_i}{8} \right\rceil \cdot RTT \tag{2.1}$$

In presence of long RTTs and high bandwidth, the result from (2.1) is a large *cWnd*, independently of the TCP version adopted. This may lead to a bursty transmission, with a possible consequent performance degradation on the buffers side, especially when few nodes share a lot of bandwidth. However, such burstiness can be eliminated by employing packet spreading techniques [49, 50] or by limiting the resulting *cWnd* value. The latter approach does not exploit the available bandwidth, while the former requires accurate analysis for each TCP version. Since one of the key point of C^2ML is to be easily adapted to different TCP algorithms, a mixed version of these two proposals is employed: the resulting *cWnd* is limited by a constant of $\frac{1}{2}$, and the classic slow start algorithm is used as packet spreading technique (2.2).

$$\begin{aligned} ssThres &= \left\lceil \frac{BW_i}{8} \right\rceil \cdot RTT \\ cWnd &= \frac{ssThres}{2} \end{aligned} \tag{2.2}$$

This way, the *cWnd* reaches as fast as possible (in one RTT) the given rate while remaining friendly with others TCP, thanks to the exponential *cWnd* growth of the slow start algorithm (in one RTT it effectively doubles the *cWnd*). In fact, if no losses are detected in slow start, when the *cWnd* reaches the *ssThres* value, with an accurate RTT measurement, the bandwidth utilized is the same as the one assigned by the gateway (assuming the receiver window is large enough).

To avoid congestion episodes, another action to employ is to limit the congestion avoidance part of the algorithm, in order not to exceed the assigned *cWnd*, and to maintain an accurate *ssThres* value by recalculating it each time the RTT changes. It is worth to note that the congestion avoidance function is free to reduce the *cWnd*, not only because receiver status and conditions beyond the bottleneck are unknown, but also to maintain compatibility with nodes without

C^2ML support in the same local network (in this case the effectiveness is reduced, but allowing a *cWnd* reduction makes possible the gradual deployment of C^2ML).

A pseudo-code of such changes, employed to adapt TCP Cubic, is presented in Figure 2.13. The original *AckRcv* function, invoked each time a new ack arrives, updates *cWnd* on the basis of its rules, and is thus overwritten in our case. The modifications are conceptually the same for other TCP versions; in particular, in our experiments we adapted every TCP version changing only the call to the proper *AckRcv* function (e.g. $NewReno :: AckRcv$, $HighSpeed :: AckRcv$ and so on). If gateway can not be reached for a long period of time (e.g. the TCP session expires) nodes can automatically disable such extensions, in order to transparently fall back to the standard behaviour.

2.2.4 C^2ML Performance

We implemented C^2ML and the modifications to adapt various TCP congestion control algorithms (including Noordwijk, NewReno, Cubic, Hybla, HighSpeed, Bic) to work with C^2ML in the ns-3 network simulator [51]; we then simulated the setup described in Section 2.2.2 and then we ran an extensive packet-level simulation.

2.2.4.1 Intra-protocol fairness

The aim of this subsection is to prove the effectiveness of C^2ML in terms of general throughput and fairness of flows of the same TCP type, with same or different RTTs, over a bottleneck link of 1 Mbit/s with a delay of 300 ms. To achieve this, we performed several simulations enabling C^2ML with *standard* buffer size configuration. At first, we show in Figure 2.14 how three TCP NewReno flows perform over a large transfer, with the same RTT. Thanks to C^2ML the slow increase of TCP *cWnd* is avoided; thus, several RTT are saved and the data rate is close to the maximum value since the very beginning, reaching the best throughput performance earlier than the NewReno simulation presented in Section 2.2.2 (Figure 2.8), with C^2ML disabled. However, we can still observe little oscillations when other nodes join the network.

The typical NewReno behaviour is to privilege flows with short RTT, because of the faster growth of the window size. For instance, with an RTT of 1 ms the

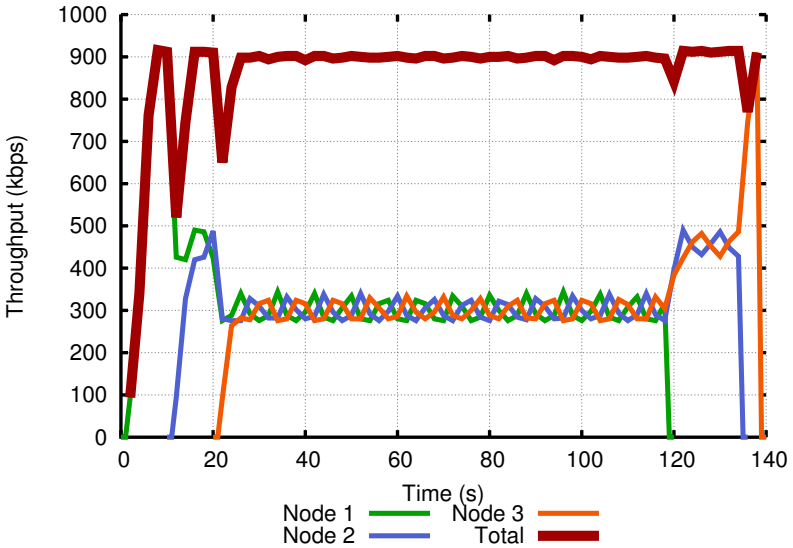


Figure 2.14: Three nodes, TCP NewReno and C^2ML, file transfer, *standard* buffers

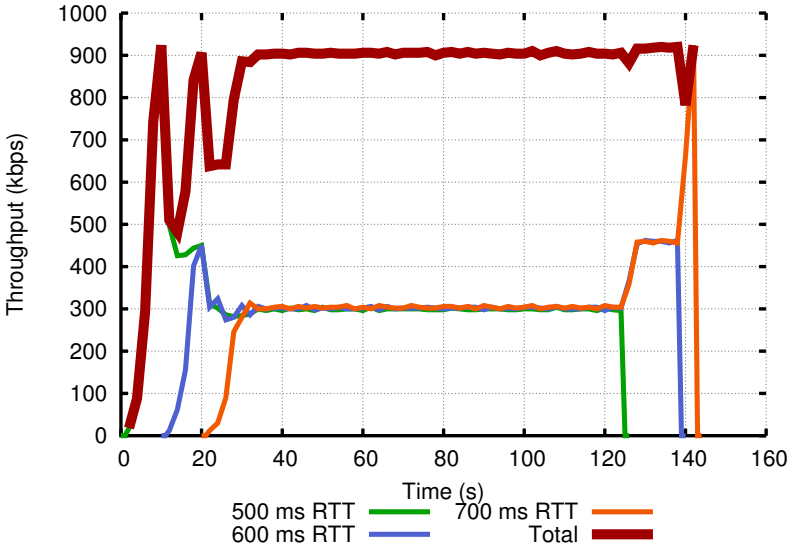


Figure 2.15: Three nodes, TCP NewReno and C^2ML, file transfer, *standard* buffers, with different RTT

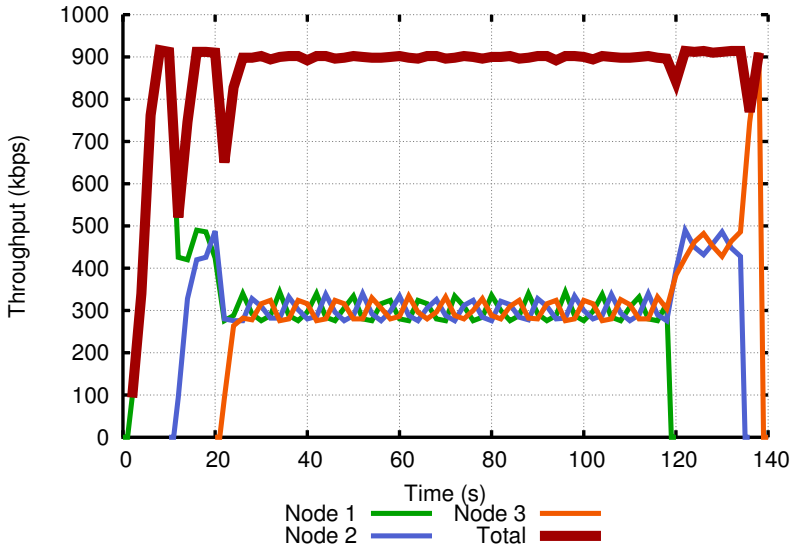


Figure 2.16: Three nodes, TCP Cubic and C^2ML, file transfer, *standard* buffers

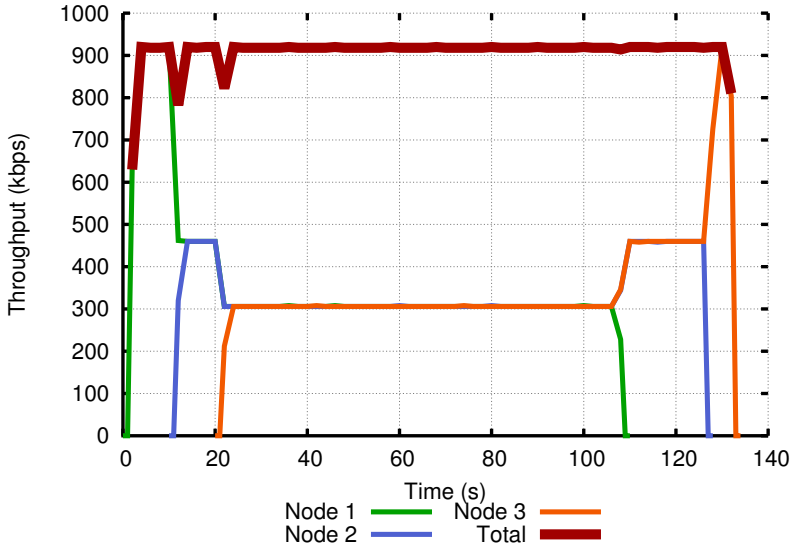


Figure 2.17: Three nodes, TCP Noordwijk C^2ML, file transfer, *standard* buffers

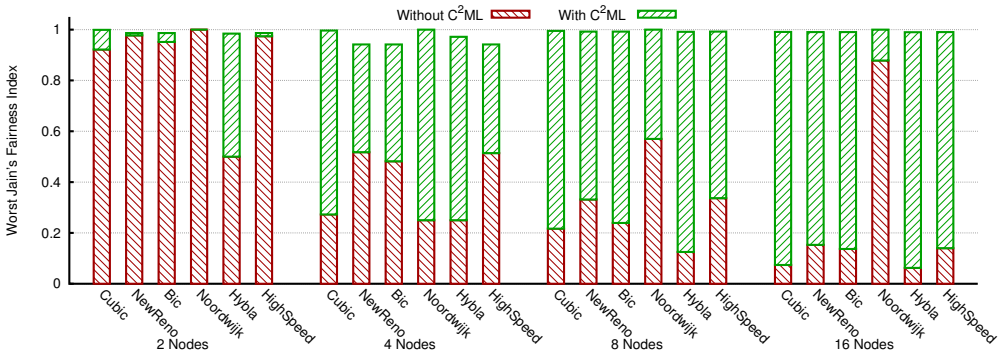


Figure 2.18: Worst Jain's Fairness Index for various TCP, *large* buffers

window can grow by 30 packets in 30 ms, while with an RTT of 100 ms the same increment is achieved in 3 seconds. By utilizing C^2ML the NewReno protocol becomes RTT-fair, as we show in Figure 2.15, where the throughput of three competing TCP NewReno flows with C^2ML and different RTTs is plotted. When C^2ML is employed, all flows (regardless of RTT) can set their proper rate. In general it helps any TCP protocol, which isn't RTT-fair by design, to become RTT-fair.

For an RTT-fair scheme such as TCP Cubic, C^2ML helps in terms of both throughput and fairness: in Figure 2.16 it is shown how flows self-regulate to the proper rate given by the gateway, and how throughput oscillations are decreased leading to better intra-protocol fairness. We have implemented and tested other congestion control algorithms, but we obtained the best results with a congestion control that employs a fine tuning of bandwidth utilization, like TCP Noordwijk. As we can see in Figure 2.17, there are not throughput oscillations with it, leading to a very high fairness among flows.

To further analyse the fairness, it can be defined in terms of different system parameters. For instance, it could be defined in terms of throughput if each flow is allocated the same amount of service over time, or it could be defined in terms of data rate if the objective is to allocate flows resources in order for all of them to achieve the same data rate. Assuming equal data rate requirements among flows, the instantaneous Jain's Fairness index JFi is defined in terms of the instantaneous data rate R_i as:

$$JFi(R_1, R_2, \ldots, R_n) = \frac{(\sum_{i=1}^{n} R_i)^2}{n \cdot \sum_{i=1}^{n} {R_i}^2}$$

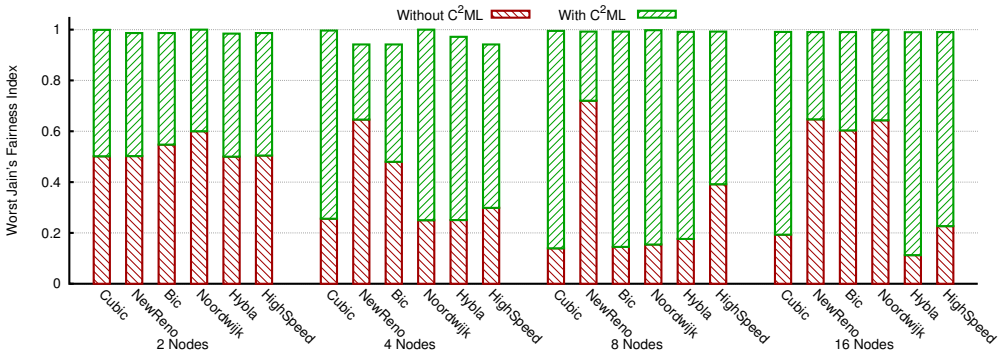


Figure 2.19: Worst Jain's Fairness Index for various TCP, *standard* buffers

where n is the number of active flows, R_i is the instantaneous data rate of flow i and JFi is a real number in the interval $[\frac{1}{n}, 1]$ with a maximum best-case value of 1 if the achieved rate is equal for all flows and a minimum worst-case value of $\frac{1}{n}$ if only one aggressive flow is filling the entire data rate of the system.

In Figure 2.18 we show the lowest JFi calculated over a series of 5 minutes run, where a defined number of flows, continuously backlogged, are sending data. This way it is possible to highlight a lower bound of fairness guaranteed for each configuration. For this test we used the *large* buffer configuration for each node and varied the number of the n active flows from 2 up to 16, performing two runs, one without C^2ML, and the other with C^2ML enabled. It can be seen that in all cases without C^2ML lower JFi values are obtained, which indicates that in some points there is a poor fairness among flows (in other words, one or more flows are stealing bandwidth to others). On the contrary, when C^2ML is employed the flows have a higher JFi, reaching in most cases a value near 1, which indicates a perfect balance of bandwidth usage among different flows.

Figure 2.19 follows the same setup but with flows in a *standard* buffer configuration. By moving from Figure 2.18 to Figure 2.19 it is possible to notice a decrease of JFi values without C^2ML; instead, with C^2ML values of JFi near one are reached in most cases. A remarkable increase of fairness performance can be highlighted when TCP Noordwijk with C^2ML is employed. Therefore, the use of C^2ML helps to achieve a proper bandwidth distribution resulting in a high intra-protocol fairness, even with the *standard* buffer size. In the next subsection the buffer usage and how C^2ML is able to reduce it will be investigated.

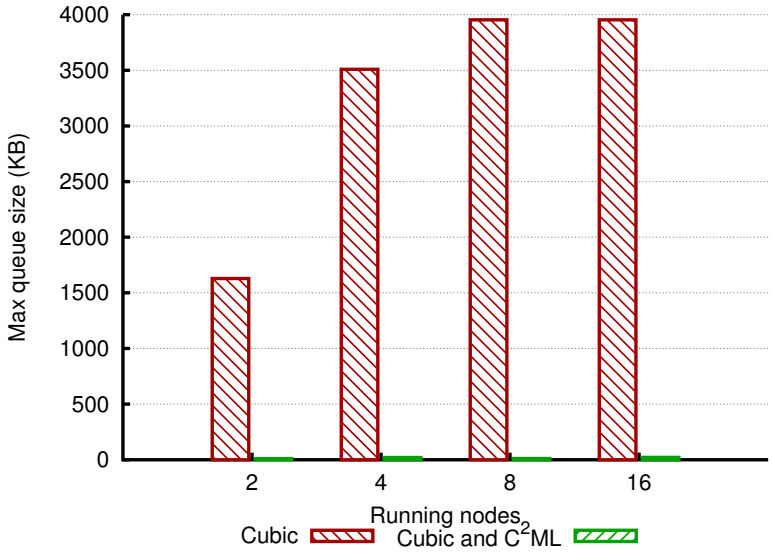


Figure 2.20: TCP Cubic max queue usage

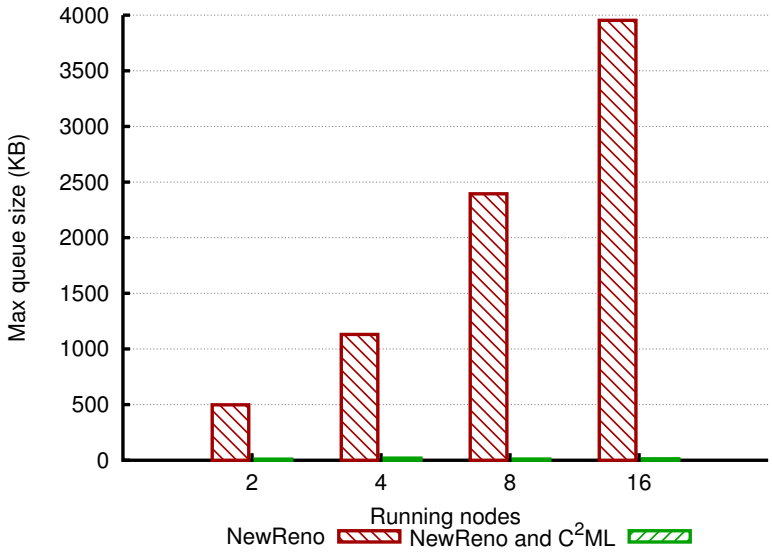


Figure 2.21: TCP NewReno max queue usage

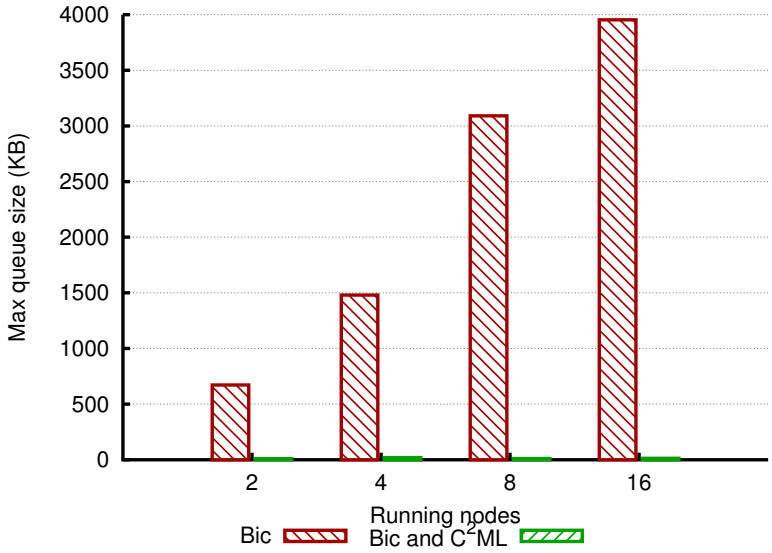


Figure 2.22: TCP Bic max queue usage

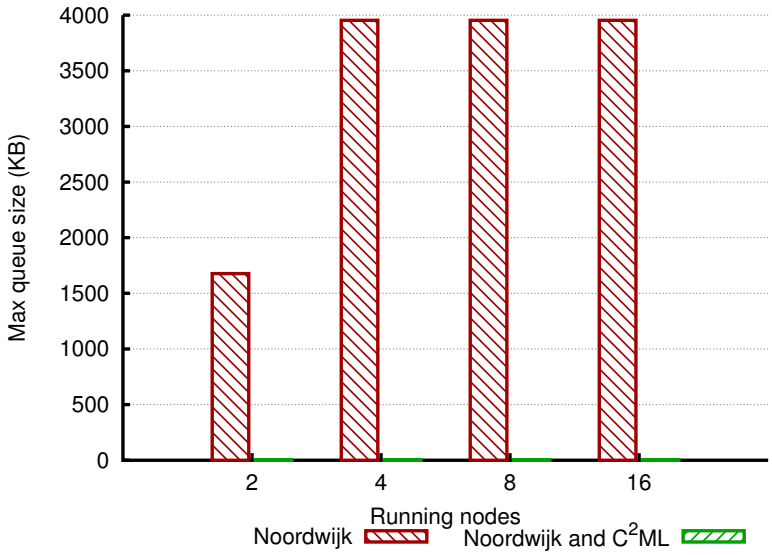


Figure 2.23: TCP Noordwijk max queue usage

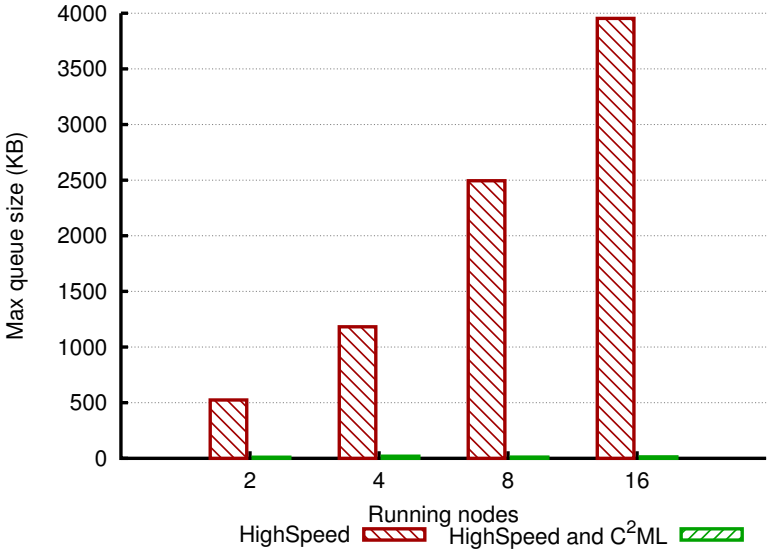


Figure 2.24: TCP HighSpeed max queue usage

2.2.4.2 Buffer usage analysis

In the following simulations the bottleneck link bandwidth has been raised to 2 Mbit/s and the number of active nodes varies from two up to sixteen; they start at the same time and they each perform a transfer of 5 MB to a remote node. We then analyse the buffer usage on the gateway. From Figure 2.20 to Figure 2.24, the maximum amount of bytes stored in the gateway queue is reported, comparing different TCP algorithms with and without C^2ML.

When C^2ML is disabled, with all tested TCP algorithms it is possible to observe a really high queue usage. This is not always an issue, but it increases the overall network latency as a function of the queue size. Contrarily, C^2ML has a less-than-logarithmic growth of used buffer space, a constant value for each simulation, because each node sets its rate to a constant value (as from Section 2.2.2), which never exceeds the bottleneck link total bandwidth.

A separate and thorough discussion has to be done for TCP Noordwijk; the growth is faster with respect to the other TCP algorithms, causing also several errors in the 16-nodes test, resulting in a poor utilization of resources. This trend explains the misbehaviour of TCPN when buffer size is limited to a standard size. In fact, because of the frequent packet drops due to the Tail-Drop queue algorithm, TCPN loses the timing information about the network, normally used

to regulate burst size and transmission timer. On the other hand, when TCPN is coupled with C^2ML, in the worst case the node i injects in the gateway queue an amount of data equal to $BURST_i$ bytes each time TX_TMR_i expires. If the C^2ML uses the Unweighted Fair Budget algorithm to assign the available bandwidth to the n connected nodes we have $BW_i = \frac{BW}{n}$, and so each node sets $BURST_i = BW_i \cdot TX\_TMR_i$ as its burst size. Without loss of generality, it can be assumed $TX\_TMR_i = TX\_TMR_j$ for all i, j (it is trivial to adjust burst size and transmission timer for all nodes maintaining the same ratio), which results in all nodes having the same burst size. At this point we can write the total amount of packets injected in the gateway queue (Q_USAGE) in the worst case, i.e. when all n nodes are transmitting at the same time, as:

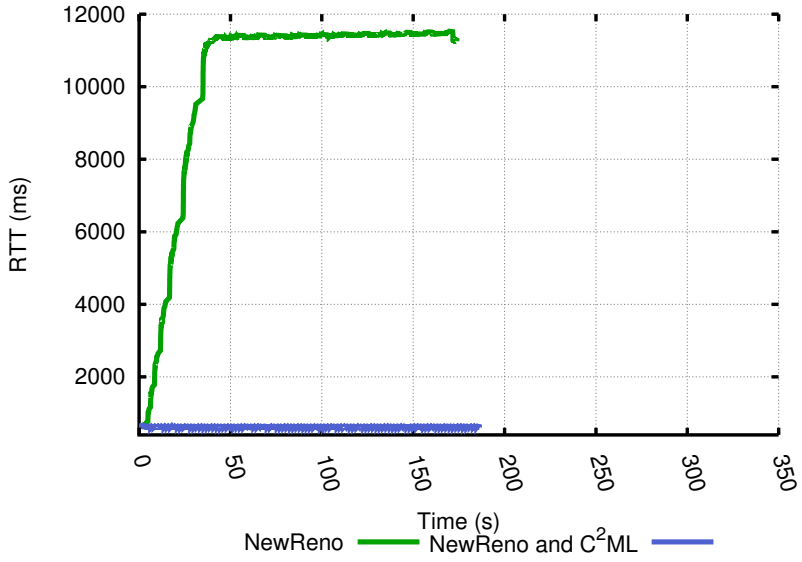
$$Q\_USAGE = n \cdot BURST_i$$

As a general note, nodes should have a low default transmission timer in order to have a small burst size. This helps not to over-exceed the receiver window and to maintain queue occupancy low. In the sixteen nodes case, thanks to C^2ML, each node sets its burst size equal to 3 segments (1500 bytes) and the transmission timer to 100 ms; this results in using at most $16 \cdot 1500 = 24000$ bytes of queue space on the gateway, an amount of data that is transmitted (at a 2 Mbit/s rate) in roughly 91 ms, leaving the buffer empty at the next transmission timer expiration.

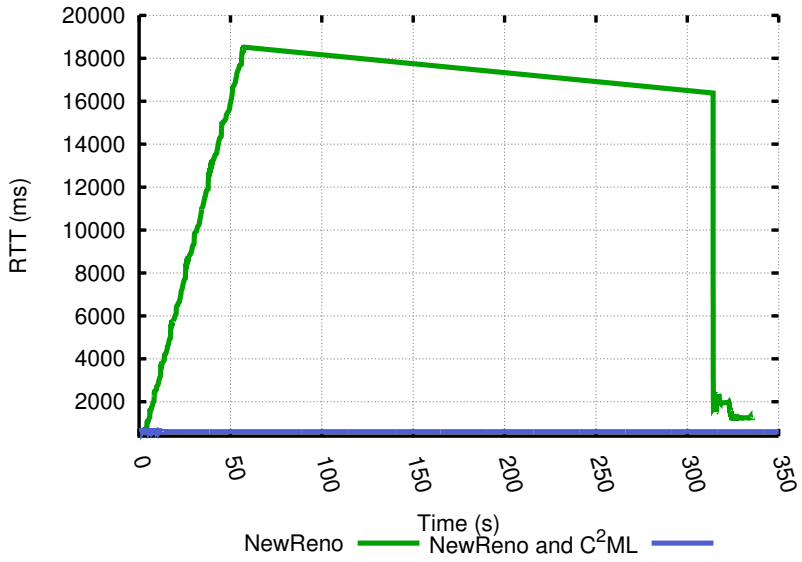
Without C^2ML, such a high buffer usage of TCP Noordwijk, as well as of other TCP algorithms, leads to performance degradation with respect to packet delay, which will be investigated in the next subsection.

2.2.4.3 Delay analysis

The queuing delay is an important factor for network performance. High queuing delays means a high packet transfer delay, and so an overall degradation of the Quality of Experience perceived by the users; this delay greatly depends on the queue length along the entire path. Here, we used the *large* buffer configuration for each node. Our aim is to show how C^2ML impacts over the RTT experienced by various TCP congestion control algorithms in a large buffer scenario, maintaining the same bandwidth and the same transmission delay of the previous section (i.e. 2 Mbit/s and 300 ms).



(a) 8 nodes



(b) 16 nodes

Figure 2.25: TCP NewReno RTT analysis

	TCP Cubic			TCP Cubic+C^2ML		
n	APD (s)	PDV (s)	End (s)	APD (s)	PDV (s)	End (s)
2	0.373	0.447	99	0.300	0.037	47
4	0.818	1.420	124	0.301	0.039	93
8	1.936	3.408	223	0.301	0.059	188
16	4.207	7.326	397	0.300	0.039	382

Table 2.1: Average Packet Delay and Packet Delay Variation summary for TCP Cubic

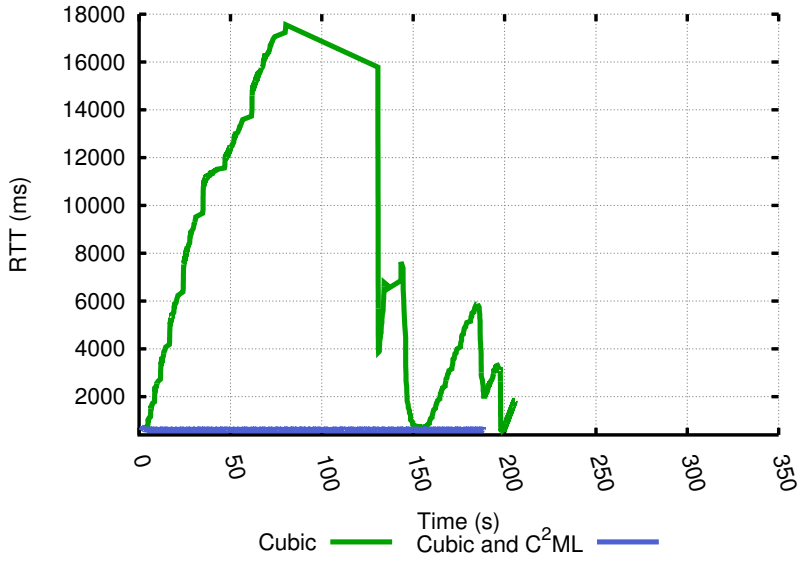
In Figure 2.25a and Figure 2.25b we show the RTT experienced by one node over eight (or sixteen in Figure 2.25b) nodes running TCP NewReno. As we can see, RTT of packets for TCP NewReno is very high, reaching for the sixteen node simulation the value of 11s. This is caused by the buffering in the gateway queue: the sender continues to grow its window until the queue is full. Then, it notices the first drops, and it starts to limits itself. After some iterations of this process, it reaches the optimal window size for the bottleneck link, unfortunately leaving many packets in the gateway queue. As one packet is transmitted over the bottleneck link, another is arriving in the queue, effectively filling it for the entire duration of the file transmission (turning it into a bloated buffer).

Coupling TCP NewReno with C^2ML, RTT effectively decreases (thanks to the lower buffer usage), and the overall transfer time decreases as well. The congestion control, having its bandwidth assigned, does not long see the queue as an empty space to fill, so it does not have to wait or notice the first drops to limit itself. Therefore, most of the queue in the gateway is left unused.

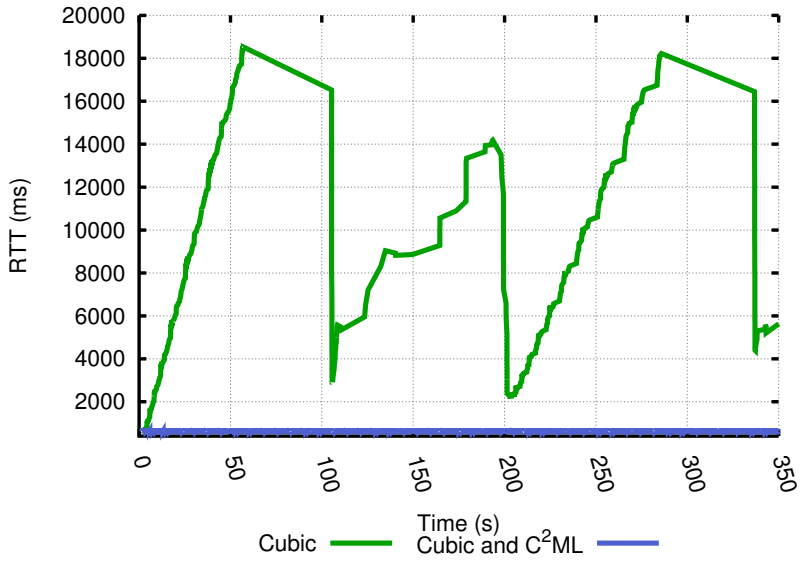
Similar results, with an overall decrease of both RTT experienced by packets and a decrease of the transmission time, are obtained by replacing TCP NewReno with TCP Cubic, as in Figure 2.26a and Figure 2.26b. We tested other TCPs, and in no cases a high buffer usage, when using C^2ML, has occurred.

For TCP Noordwijk, the improvement is remarkable (Figure 2.27). Reminding the high buffer usage of Noordwijk without C^2ML, the results are rather intuitive and the user experience in terms of latency is considerably improved. To reinforce this result, some statistics about the delay experienced in the network are reported next. Every 100 ms the gateway buffer occupancy is read, indicated as Q_USAGE_t at time t, and Average Packet Delay is computed as

$$D(t) = \frac{Q\_USAGE_t \cdot MSS}{BW} + PD \quad APD = \frac{\sum_{i=0}^{N} D(i \cdot 10^{-3})}{N}$$



(a) 8 nodes



(b) 16 nodes

Figure 2.26: TCP Cubic RTT analysis

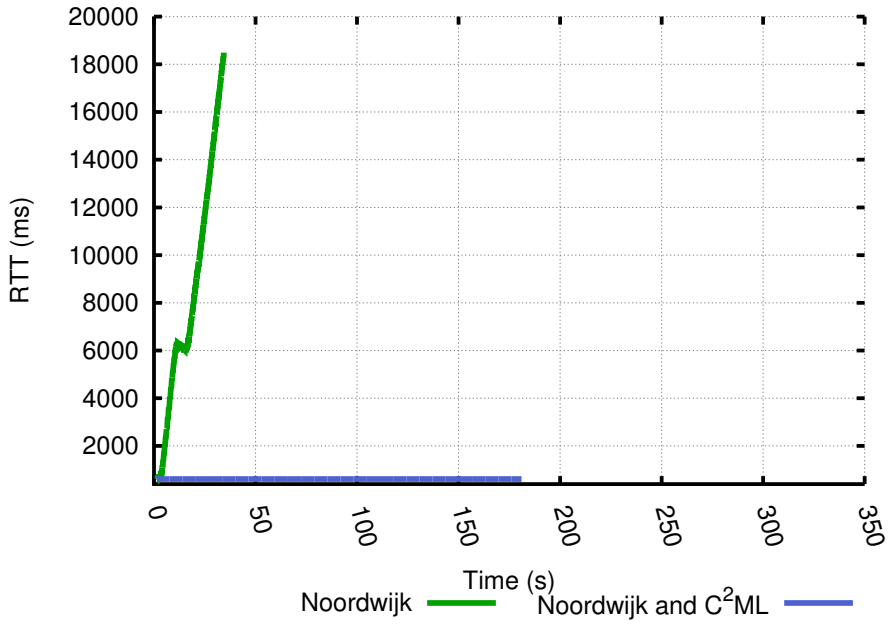


Figure 2.27: TCP Noordwijk RTT analysis, 8 nodes

	TCP NewReno			TCP NewReno+C^2ML		
n	APD (s)	PDV (s)	End (s)	APD (s)	PDV (s)	End (s)
2	0.300	0.002	116	0.300	0.037	47
4	0.339	0.209	123	0.301	0.039	93
8	0.762	0.965	211	0.301	0.059	188
16	1.796	2.482	384	0.300	0.037	382

Table 2.2: Average Packet Delay and Packet Delay Variation summary for TCP NewReno

	TCP Bic			TCP Bic+C^2ML		
n	APD (s)	PDV (s)	End (s)	APD (s)	PDV (s)	End (s)
2	0.340	0.375	222	0.300	0.037	47
4	0.659	1.291	864	0.301	0.039	93
8	1.592	3.100	900	0.301	0.059	188
16	3.587	6.764	900	0.300	0.039	382

Table 2.3: Average Packet Delay and Packet Delay Variation summary for TCP Bic

where T is the total simulation time, N is the number of samples gathered during the simulation (each 100 ms), BW is the channel bandwidth and PD is the propagation delay. The default MSS value for all our simulations is 500 byte. We reported the results for each TCP version, along with maximum

	TCP Noordwijk			TCP Noordwijk+C^2ML		
n	APD (s)	PDV (s)	End (s)	APD (s)	PDV (s)	End (s)
2	0.757	0.492	691	0.309	0.020	46
4	1.679	1.680	665	0.304	0.018	91
8	3.580	4.477	709	0.307	0.068	181
16	2.639	8.715	900	0.315	0.178	360

Table 2.4: Average Packet Delay and Packet Delay Variation summary for TCP Noordwijk

	TCP Hybla			TCP Hybla+C^2ML		
n	APD (s)	PDV (s)	End (s)	APD (s)	PDV (s)	End (s)
2	0.748	0.490	101	0.303	0.203	49
4	1.738	1.514	168	0.311	0.299	93
8	3.727	3.561	378	0.360	0.209	179
16	6.187	7.020	690	0.620	0.363	376

Table 2.5: Average Packet Delay and Packet Delay Variation summary for TCP Hybla

	TCP HighSpeed			TCP HighSpeed+C^2ML		
n	APD (s)	PDV (s)	End (s)	APD (s)	PDV (s)	End (s)
2	0.300	0.004	100	0.301	0.039	47
4	0.432	0.502	191	0.301	0.039	93
8	1.030	1.555	271	0.301	0.059	188
16	2.369	3.637	393	0.300	0.037	382

Table 2.6: Average Packet Delay and Packet Delay Variation summary for TCP HighSpeed

Packet Delay Variation (PDV) and the total time needed for all nodes in order to complete the 5 MB transfer; the former is calculated as

$$PDV = \max(D(0) - D_{min}, \cdots, D(T) - D_{min})$$

where D_{min} is the minimum delay value over the entire data set. For TCP Cubic, data are reported in Table 2.1, and it can be seen that by employing C^2ML the packet transfer is faster (as the End Time of the transfer is considerably lower) and both APD and PDV are shorter, except in the 2 nodes cases where the PDV is 30 ms higher with C^2ML. This is not a big deal, as the APD is near the transmission delay of the bottleneck link (300 ms). In general, this behaviour holds for other TCPs (Table 2.2 for TCP NewReno, Table 2.3 for TCP Bic, Table 2.4 for TCP Noordwijk, Table 2.5 for TCP Hybla and finally Table 2.6 for TCP HighSpeed). It should be noted that, while slightly worsening the base

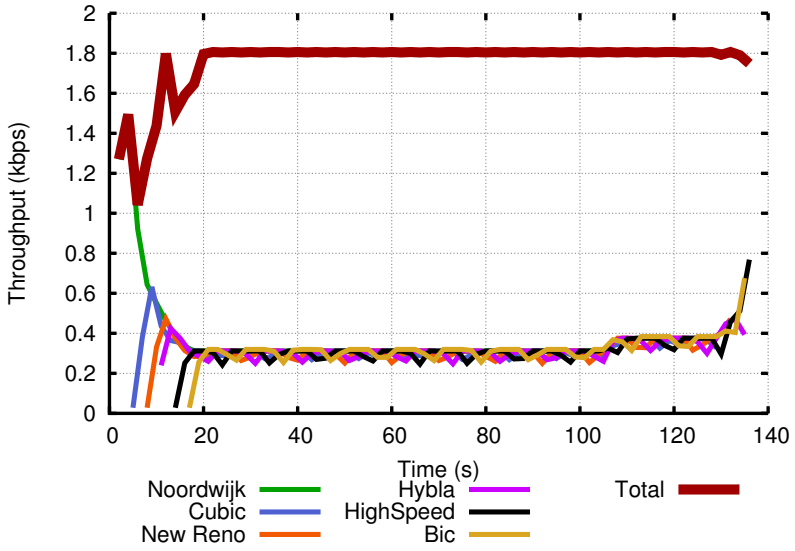


Figure 2.28: Friendliness among TCP flows, *standard* buffers

		Transmitted data					
		1.5MB		5MB		10MB	
		C^2ML+	No C^2ML+	C^2ML+	No C^2ML+	C^2ML+	No C^2ML+
Nodes	32	61.6	117.2	232	458	476.8	947.7
	64	111	219.9	402.5	803	820.6	1639.1
	128	228.4	455.5	810.7	1620.3	1662.7	3324.3

Table 2.7: Average short-transfer completion time

cases, employing C^2ML makes APD and PDV near constant among simulations, regardless of the number of nodes. It also makes the different congestion controls to behave the same over an identical bottleneck link: for instance, data of NewReno, Cubic or Bic coupled with C^2ML are the same. Therefore, an interesting follow-up is to see how different and competing congestion controls work together.

2.2.4.4 Short-transfer performance assessment

With this set of simulations we aim to analyse C^2ML behaviour in case of high dynamism of traffic, thanks to scenarios with several short connections and a larger number of nodes. We carried out tests with 32, 64 and 128 users. For

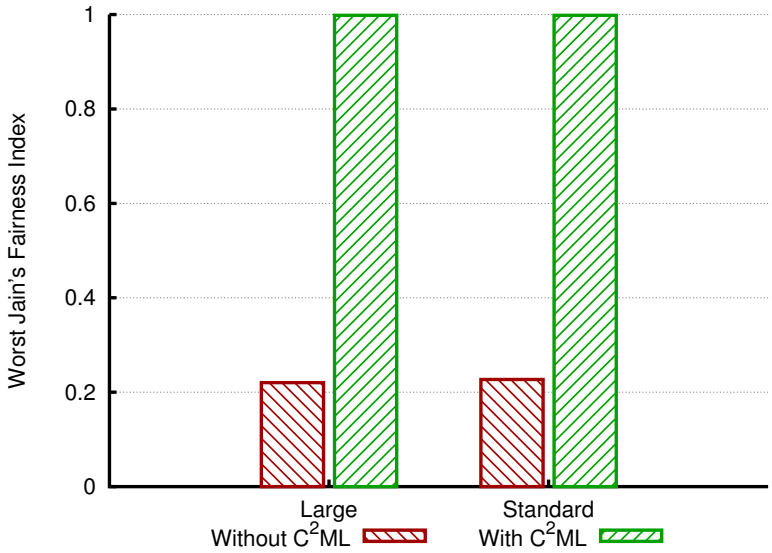


Figure 2.29: Worst Jain's Fairness Index between different TCP

		Transmitted data					
		1.5MB		5MB		10MB	
		C^2ML+	No C^2ML+	C^2ML+	No C^2ML+	C^2ML+	No C^2ML+
Nodes	32	64.9	365.9	247.9	490.2	505.5	1005.4
	64	113.9	226.2	400.1	798.4	820.6	1639.5
	128	230.6	460.3	818.7	1636.4	1658.9	3316.8

Table 2.8: Average short-transfer completion time with background traffic load

each test we considered a number of short-living connections equal to twice the number of active nodes, and each connection is opened randomly in the range of [0,30] seconds of simulated time by a random user (i.e. each user can open more than 1 connection). We repeated each test as a function of the short-transfer size considering transmissions of 1.5MB (the average web page size), 5MB (huge web page size) and 10MB (high resolution image). The evaluation parameter is the average completion time for each connection experienced with or without C^2ML. Results of these tests are reported in Table 2.7. The table highlights the ability of C^2ML to maintain very good performance even in presence of high dynamism on the network; in fact, it halves the average completion time of the connections in almost all the cases.

We repeated the same set of experiments adding to the short transfers 10 long-living TCP flows, continuously backlogged, as background load for each simulation in order to increase the network congestion while continuing to maintain the bandwidth subdivision dynamism. Therefore, we evaluated C^2ML performance in a more realistic and challenging scenario. We then reported the average completion time of all short-living connections in Table 2.8. By comparing Table 2.8 with Table 2.7, it is worth to notice that the average completion time of the connections slightly increases due to the presence of the background load traffic; at the same time, C^2ML leads again to an halving of the completion time.

2.2.4.5 Inter-protocol friendliness

Here the focus is on how C^2ML may help the various TCP protocols in terms of both throughput and latency when they compete for the same link. Using 2 Mbit/s and 300 ms delay for the bottleneck channel, we simulated six nodes, each one with a different TCP version: Cubic, Noordwijk, NewReno, Hybla, HighSpeed, and Bic. Each node starts 3 seconds after the preceding one, with the *standard* buffer setup, and their throughput is plotted in Figure 2.28. Comparing it with Figure 2.9, without C^2ML and with the *large* buffers (leaving other network parameters unchanged), it can be noted how employing C^2ML helps to maintain throughput stable, with a low buffer usage, after the initial slow-start phase. As a consequence, the overall flows friendliness is boosted, and also JFi results are higher. It is calculated for both configuration of buffer sizes (*large* and *standard*), with and without C^2ML, and the worst case JFi is plotted in Figure 2.29.

2.2.5 Conclusions

A new cooperative transmission control middleware, called C^2ML, has been described and evaluated. This solutions effectively optimizes performance in networks with high delay, such as satellite networks. Design goals and requirements were based on the analysis of current issues of state-of-the-art TCP algorithms in such links; the focus was on satellite channels because they represent an almost-ubiquitous medium to provide communication and data connectivity in crisis scenarios, but it can be adapted in other contexts were a gateway provides connectivity to a large number of clients.

C^2ML acts primarily on the transport layer by regulating flows between connected nodes, and it indirectly tunes network-level buffer usage, requiring only sender-side modifications (with minor adaptation to the employed congestion control protocol) along with a central gateway, resulting in a feasible and deployable system.

Through numerical results it has been shown that C^2ML leads to high throughput and it improves intra-protocol fairness and inter-protocol friendliness regardless of queue sizes, thus reducing the overall latency and avoiding the "Bufferbloat" problem. Moreover, same results have been achieved even when the number of competing nodes increases, therefore confirming C^2ML as a practical and scalable system.

Finally, the use of C^2ML exploits much more effectively the potential offered by satellite links, even for real-time applications. The next Section includes the possibility to renegotiate the bandwidth allocation for problematic connections (e.g. restricted signal due to propagation obstacles) which can not fully exploit the assigned bandwidth.

2.3 DyBRA

The massive diffusion of the Internet for personal and commercial use and the ever increasing consumer demand for availability and performance have pushed the development of many physical technologies, such as terrestrial wired and wireless links as well as non-terrestrial wireless mediums (e.g. satellites). The latter, in particular, exhibits poor performance when coupled with the TCP protocol, due to the high delay distinctive of the channel. This is a well-known issue that has been extensively studied, and methods to mitigate this problem are operative in real world networks under the form of various congestion control algorithms [26,27,31,40], TCP connection splitters and Performance-Enhancing Proxies (PEPs) [52], as well as various link-layer techniques (RFC 2488).

These channels are used to provide connectivity in areas not covered by commercial terrestrial networks (or where their coverage is narrowband only) and in specific utilization contexts, such as emergency networks or military environments.

In scenarios where a high end-to-end latency is acceptable, however, the major limit is not represented by a particular congestion control algorithm, nor by the links performance, but instead by the transport protocol fundamentals. The affected scenarios are those in which a number of client hosts are wirelessly connected, through a gateway, to a backhaul link (e.g. during a post-disaster emergency situation). In these cases, shifting to a *top-down*, centralized and collaborative management of resources from the gateway has the potential to guarantee overall better performance for hosts [25, 53]. To reach this result we previously proposed Congestion Control Middleware Layer (C^2ML), which improves users' QoE in such scenarios [54].

The TCP protocol infers channel conditions indirectly, by loss detection and/or ACK-based timing information, and then reacts accordingly. Since the protocol always interprets issues as signals of congestion, its reaction is not always suitable to the link conditions. For example, degradation on the wireless interface of radio access systems may lead to unnecessary performance drops [55]. In fact, even in absence of congestion, channel conditions of wireless systems can change rapidly (e.g. for physical obstacles), much faster than the TCP reaction time, in particular when the available capacity increases. Collaborative tools like C^2ML may further degrade the situation if channel conditions are not taken into

account, because hosts can be given more bandwidth than they are able to use, therefore stealing it from hosts willing to use more.

This Section proposes *DyBRA*, an improved version of C^2ML with a bandwidth management algorithm designed to exploit the full bottleneck link potential, even in the case of fluctuations in bandwidth utilization from client hosts. *DyBRA* does not intend to address channel changes that are due to radio oscillations, but instead aims to intervene in cases where the channel state variation is significant and abrupt, as when a client node moves behind an obstacle between its receiver and the gateway. The algorithm is deployed on the latter, and manages the bandwidth allocation among connected hosts. These hosts know their allocated bandwidth thanks to C^2ML, and through Channel State Information (CSI) [56] [57] they may also know if it can be fully exploited in the air interface. When there is a significant limitation, the unused bandwidth is reassigned to the trouble-free hosts by the *DyBRA* algorithm.

In Section 2.3.1 we present related work, Section 2.3.2 explains the DyBRA algorithm, Section 2.3.3 details the simulations performed and discusses main results, while in Section 2.3.4 we draw our conclusions.

2.3.1 Literature Overview

Mobile networks may exhibit relatively long lasting efficiency problems. In fact, due to the exponential TCP back off algorithm, repeated timeouts result in long intervals between two TCP retransmission attempts. Therefore, even if in the meanwhile the radio conditions improve, TCP does not timely recover. Authors in [55] propose a network-side TCP-freeze mechanism, tailored for radio access nodes such as LTE eNodeBs or HSPA+ base stations. With it, the retransmission timer does not expire and the TCP sender does not send any new data until better network conditions are restored. It has also been shown, in wireless mesh networks, that using cross-layer information may significantly improve goodput thanks to an accurate packet loss identification. Unfortunately, this approach alone [58] [59] does not prevent congestion and does not ensure a low queue usage on network elements.

Techniques that aim to make end points aware of the congestion degree of the network (therefore preventing congestion instead of reacting to it) are based

on congestion feedback from network elements. An example is the eXplicit Control Protocol (XCP) [34], that outperforms TCP in conventional environments and remains efficient, fair, and stable as the link bandwidth or the round-trip delay increases. It generalizes the Explicit Congestion Notification (ECN) [35], informing senders about the degree of congestion in the network. The protocol is stateless, as no router along the path should maintain a per-flow state, and can thus scale to any number of flows. Unfortunately, each router over the entire end to end path plus the receiver have to be XCP-aware, and it also presents oscillations in cases of large delays. A similar, yet correlated, approach has been proposed with Rate Control Protocol (RCP) [60]. This solution enables typical Internet-sized flows to complete one to two orders of magnitude faster than TCP Reno and XCP using the same technique of the latter: explicit feedback from routers. Consequently, RCP suffers the same drawbacks of XCP.

XPLIT [61] is a network architecture based on TCP cross-layering, aimed at optimizing the transport layer performance on a DVB-S2 satellite link. XPLIT-TCP is the cross-layer TCP protocol that manages the system load. Although it is not a completely PEP-based solution, at least one PEP is needed, and the evaluation discussion, based on simulations performed with the ns-2 network simulator, presents no analysis of latency, RTT and queues usage.

In order to prevent congestion on high-delay links, we proposed a cooperative scheme to be adopted by clients of the same Local Area Network connected through a central gateway, that we called C^2ML [54]. It introduces only sender-side modifications to existing TCP algorithms, plus a stand-alone application that runs over the gateway. The scheme is applicable to all cases in which the bottleneck is known, as showed by Figure 2.12, and it guarantees regular operation to legitimate users even in presence of malicious ones performing resource exhaustion attacks, thanks to an active queue management technique that we called QRM [62]. It lacks, however, the capability of dynamically adjusting bandwidth on the basis of clients feedback.

2.3.2 The *DyBRA* Algorithm

To properly describe how *DyBRA* works, it is necessary to move back to Figure 2.12 used to describe C^2ML in the previous Section. In fact, Figure 2.12 shows the C^2ML architecture. Conceptually, it inserts a Congestion Control

Layer (C^2L) between the Application and the Transport Layers, that uses Congestion Control Layer Protocol (C^2LP) as the default standard to communicate. Basically, to the gateway that provides clients with external connectivity, C^2ML gives the authority of instructing these client hosts on the amount of bandwidth they are allowed to use. This is trivially calculated dividing the bottleneck link capacity by the number of clients to serve, in order to guarantee a fair utilization of the former. For a detailed description of the protocol and its primitives we remind to [54].

In order to adapt to the case of bandwidth underutilization from some clients, C^2LP should be extended to support the exchange of bandwidth utilization feedbacks between them and the gateway if the channel state variation is significant, i.e. if a client host is unable to utilize a bandwidth amount that is equal to at least 20% of the previously assigned bandwidth.

Messages: To give clients the possibility to communicate the gateway (G) the actual bandwidth they are using, the following messages are added to C^2LP:

- `USED_BW`, sent by a client to inform G on the maximum bandwidth it uses.
- `ACK_HELLO` and `ACK_USED`, sent by G as an ACK for a `CLIENT_HELLO` message and the `USED_BW` message, respectively.

As parameter, `ACK_HELLO` and `ACK_USED` carry a blob of data (which should be signed by G to prevent forging) that represents the client i state, $State_i$, composed by two fields: bw with the bandwidth assigned to i, and s with i status, which can be either $GOOD$ if the client has declared the full utilization of bw (default) or BAD if it has declared only its partial utilization. i echoes back to G the received data blob with a `USED_BW` or a `CLIENT_BYE` message. As internal state, each client node i keeps its available bandwidth b_i (received through an `AVAIL_BW` message), its usable bandwidth c_i (gathered through CSI), its status s_i (which can be either $GOOD$ or BAD) and the data blob $blob_i$ received from G.

Client operating principles: Thanks to the `USED_BW` message, clients are capable to inform G on their maximum used bandwidth. By default, clients are self-marked as $GOOD$ with no need to send an `USED_BW` message unless they are unable to utilize all the bandwidth assigned to them, holding the threshold condition stated above; in this case, i.e. $c_i < b_i$, they should mark themselves as

```
procedure HandleMessage(Bandwidth bw)
    b_i ← bw
    if b_i < c_i then
        if s_i == BAD then
            s_i ← GOOD
            SendUsedMessage(c_i)
        end if
    else
        if s_i == GOOD then
            s_i ← BAD
            SendUsedMessage(c_i)
        end if
    end if
    C²ML::HandleMessage(bw)
end procedure
```

Figure 2.30: Client node i receives an `AVAIL_BW` message.

BAD and send an `USED_BW` message to G with their usable bandwidth c_i. On the other hand, if an i is self-marked as BAD and a reduction of b_i or a growth of c_i occurs, it may result that $c_i \geq b_i$; the node should mark itself as $GOOD$ and send an `USED_BW` message to G, with the $\max(c_i, b_i)$ bandwidth value. Except for these cases, client operations remain unchanged from the original C^2ML specification [54]. Pseudocode invoked when an `AVAIL_BW` message is received from a client is reported in Figure 2.30.

Bandwidth subdivision: G knows the total bandwidth S_G of the bottleneck link. It also maintains the number of active nodes n_G and the number of nodes in the BAD status bad_G. With the variable S^G_{eff} it keeps track of the overall amount of bandwidth that is managed by all clients in the $GOOD$ status; this variable is initialized with the value S_G when the first client connects. It is important to note that client states are distributed to client themselves and echoed back to the server in each client request. Therefore, the memory usage on the gateway is $\mathcal{O}(1)$. When a client i sends a `CLIENT_HELLO` packet, G increases n_G and creates $State_i$. Then it replies to i with an `ACK_HELLO` message, which contains:

$$State_i = \langle bw \leftarrow \frac{S^G_{eff}}{n_G - bad_G} \ , \ s \leftarrow GOOD \rangle$$

Initially, as already stated, each node is considered in the $GOOD$ status, and it is assigned a bandwidth $bw = \frac{S_G}{n_G}$. With an `USED_BW` message, a client i sends as parameters c_i and the last received $State_i$. When G receives the message, it

uses them to recalculate the bandwidth to assign to every node. Depending on s and c_i, there can be three events:

$State_i[s] ==$ GOOD $\wedge c_i < \frac{S_{eff}}{n-bad}$. The client is unable to use all the assigned bandwidth; its status should therefore move from *GOOD* to *BAD* and the unused bandwidth should be reassigned to other *GOOD* clients (Figure 2.31).

$bad \leftarrow bad + 1$
$S_{eff} \leftarrow S_{eff} - c_i$
$State_i[bw] \leftarrow c_i$
$State_i[s] \leftarrow BAD$
SEND_ACK_USED $(State_i)$
BROADCAST_AVAIL_BW $(\frac{S_{eff}}{n-bad})$

Figure 2.31: Client i moves from *GOOD* to *BAD* status.

It is worth to note that the client i (and other *BAD* clients) will ignore the received AVAIL_BW message with the updated available bandwidth (see Figure 2.30).

$State_i[s] ==$ BAD $\wedge c_i \geq \frac{S_{eff}}{n-bad}$. In this case i was *BAD*, but now it can use at least as much bandwidth as a *GOOD* one. G should reassign the bandwidth accordingly (Figure 2.32). $State_i[s] ==$ BAD $\wedge c_i < \frac{S_{eff}}{n-bad}$.

$bad \leftarrow bad - 1$
$S_{eff} \leftarrow S_{eff} + \frac{S_{eff}}{n-bad}$
$State_i[bw] \leftarrow \frac{S_{eff}}{n-bad}$
$State_i[s] \leftarrow GOOD$
SEND_ACK_USED $(State_i)$
BROADCAST_AVAIL_BW $(\frac{S_{eff}}{n-bad})$

Figure 2.32: Client i moves from *BAD* to *GOOD* status.

The client mark remains *BAD*, because c_i is less than the *GOOD* threshold. However, G should distinguish between two sub-cases. In the first, $bw > c_i$, which means that i is using even less bandwidth than before. In the other, $bw < c_i$, which means that i has improved its bandwidth usage, but not at the point such that it can shift its status from *BAD* to *GOOD*. Again, because *DyBRA* aims to address significant channel state variations only, in either case the appropriate action is triggered exclusively if a set threshold is exceeded (e.g.

c_i at time t_1 is less than 20% with respect to c_i at time t_0); this is for simplicity not reported in the pseudo-code. In the former case, G should reassign the unused bandwidth, while it should interpret the latter as an indication that client i may soon return in a *GOOD* status; therefore, G should increase the bandwidth assigned to i, reducing the extra bandwidth that has been assigned to already *GOOD* clients. Figure 2.33 reports the pseudo-code for these two events.

```
diff ← |c_i − State_i[bw]|
if bw > c_i then
    S_eff ← S_eff + diff
else
    S_eff ← S_eff − diff
end if
State_i[bw] ← c_i
State_i[s] ← BAD
SEND_ACK_USED (State_i)
BROADCAST_AVAIL_BW (S_eff / (n − bad))
```

Figure 2.33: Client i remains *BAD* but varies its bandwidth usage.

Formal analysis of *DyBRA*: Both the C^2ML and the *DyBRA* routines require $\mathcal{O}(1)$ of time computation. Nevertheless, the convergence time of *DyBRA* after a network event should be analysed. Changes propagation following an event is monotonic (not proven here for space constraints); e.g. after a bandwidth deallocation (due to CSI or network leaving) other bandwidth space may be released by other nodes (moving from *GOOD* to *BAD* because unable to use the new bandwidth). The worst scenario in such a distributed environment is a "linear" events propagation.

Theorem 1. *Let n be the amount of clients in the network, let s_i, b_i and c_i be the status (*GOOD *or* BAD*), the assigned bandwidth and the channel bandwidth bound for the i-th node, respectively. If one event e occur, the worst-case* DyBRA *convergence time is $\mathcal{O}(n)$.*

Proof. Consider the bottleneck bandwidth BW as constant and equal to 1 and an initial perfectly fair system composed by n *good* nodes with $b_i = \frac{1}{n}$. Consider c_0, the channel bandwidth bound of the node 0, to be equal to $\frac{1}{n}$, and for all the

other nodes i with i in $[1, n-1]$ the channel bound to be:

$$c_i = \sum_{j=1}^{i} \prod_{k=0}^{j-1} (n-k)^{-1}. \tag{2.3}$$

Consider now a network event called e_0 as the channel update $c_0 = 0$ for the node zero (e.g. node zero leaves the network). After this event e_0, an amount equal to $\frac{1}{n}$ of BW will be deallocated and reassigned to the remaining $n-1$ nodes. Each node will then receive $n^{-1} \cdot (n-1)^{-1}$ of additional bandwidth updating its b_i. Unfortunately, from Equation 2.3, for the node 1, $b_1 = n^{-1} + n^{-1} \cdot (n-1)^{-1} \geq c_1$ and this will generate the event e_1 in which node 1 moves from *GOOD* to *BAD* and deallocates an amount of bandwidth equal to $n^{-1} \cdot (n-1)^{-1}$, that in turn will be reassigned to the remaining $n-2$ *good* nodes. This new event will force the event e_2 where node 2 moves from *GOOD* to *BAD* and so on in a linear propagation of status updates. It is easy to claim that for a generic event e_h with h in $[1, n-1]$:

$$\forall i \in [h, n-1], b_i = \sum_{j=1}^{h+1} \prod_{k=0}^{j-1} (n-k)^{-1}. \tag{2.4}$$

Equations 2.3 and 2.4 generates, for each e_h, the following invariant, that holds for all nodes i in $[h, n-1]$:

$$\begin{cases} b_i > c_i & \text{if } i = h \\ b_i \leq c_i & \text{otherwise.} \end{cases}$$

In particular, when $i = h$, $b_i = c_i + \prod_{k=0}^{h} (n-k)^{-1}$, only triggering the update of the h-th node from *GOOD* to *BAD*. The number of these events is linear $\mathcal{O}(n)$ and, consequently, the computational time required by *DyBRA* to converge in a stable solution is $\mathcal{O}(n)$. □

As a matter of fact, by introducing a constraint in which only a finite number of c_i values are allowed (i.e. digitalizing the CSI sensibility), the worst-case convergence time falls down to a $\mathcal{O}(1)$ complexity.

2.3.3 Performance

All simulations have been performed with the ns-3 network simulator. The network topology consists in a total of 12 ns-3 nodes. Ten of these are the LAN clients that connect to the gateway in order to reach an external host; both the gateway and the external host are modelled with one ns-3 node each. Client nodes and gateway are connected through point-to-point ns-3 links with a bandwidth of 20 Mbit/s and a propagation delay of 5 ms each, representing an average-performing LTE access network [63]; gateway and remote node, instead, are connected through a point-to-point ns-3 link with a bandwidth of 20 Mbit/s and a propagation delay of 350 ms, representing a good performance Geosynchronous Orbit satellite channel. Because *each* client node has 20 Mbit/s available between itself and the gateway, while the total bandwidth of the high-delay link is 20 Mbit/s, the latter can be safely defined as the bottleneck. To validate *DyBRA*, all traffic has been modelled as an unbounded upload from client hosts to the remote node for all the duration of simulations, always equal to 60 seconds (i.e. clients always send data at their maximum speed). To compare *DyBRA* to the general case we performed two different simulations, one with the former enabled and the other with it disabled, in which the nodes fall-back to the standard TCP operations. For the same purpose, we also set for half the number of client nodes (i.e. 5) a fixed variation in CSI at the 15th and 45th second of simulation: for the specific node, at the former simulated time the bandwidth between it and the gateway drops to 0.5 Mbit/s, while at the latter simulated time this bandwidth shifts back to 20 Mbit/s. Please note that, when *DyBRA* is employed, the maximum bandwidth used by a client node does not exceeds 2 Mbit/s when to all of them are assigned equal shares of the total bottleneck bandwidth; therefore, the bandwidth drop of those 5 client nodes at the 15th second of the simulations represents a 75% drop in nominal performance. With reference to Section 2.3.2, the drop and the rise in performance trigger a status transition from *GOOD* to *BAD* and from *BAD* to *GOOD*, respectively.

To determine the validity of *DyBRA* under uncommon conditions, we also concurrently utilized different TCP protocols in the simulations; each couple of nodes shares the same TCP protocol, and one of them always remains in the *GOOD* status while the other shifts between *GOOD* and *BAD*. Furthermore, we want to address generic high-delay links (i.e. not only satellite channels), and therefore five different TCP protocols have been used: the standard TCP

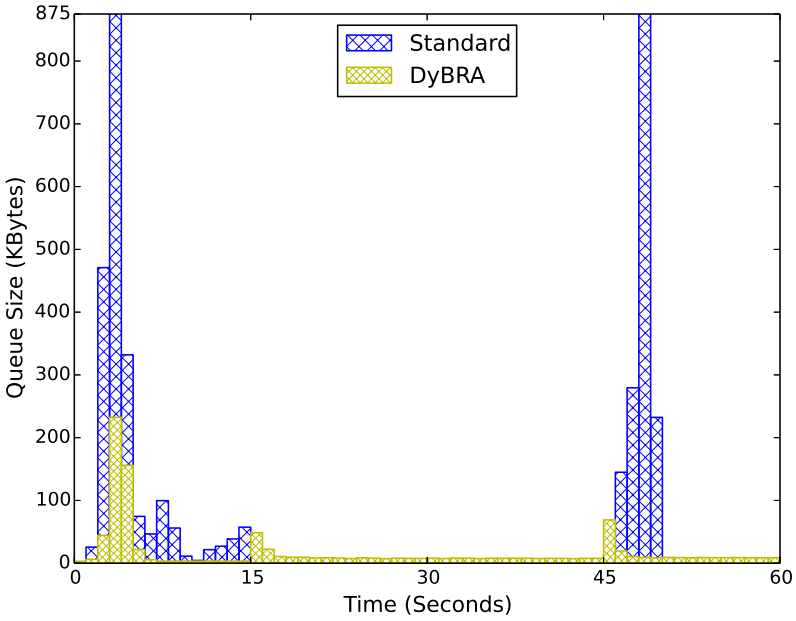


Figure 2.34: Average gateway queue occupancy.

NewReno, TCP Westwood [64], TCP HighSpeed [27], TCP Cubic [26] and TCP Noordwijk [40]; the last three ns-3 models have been developed by our team, together with the TCP Option for Window Scaling and the *DyBRA* model. The TCP Maximum Segment Size (MSS) has been set to 1000 bytes, and each client starts with an initial congestion window of 10 MSS [65]; the queues sizes of the client nodes have not been taken into account, as in this respect the issues are due to the gateway queue size where all the incoming traffic may accumulate, as a consequence of the bottleneck posed by the high-delay link. The gateway queue size has been set to 875 KB, equal to the average bandwidth-delay product of the bottleneck link; Drop-Tail has been used as the queue management algorithm because AQM schemes do not behave well with TCP Noordwijk. All throughput values have been measured at the data-link layer. Source code, simulation setup and scripts to reproduce our results are available [47].

Results: As expected, gateway queue utilization is much more efficient when *DyBRA* is employed. As Figure 2.34 shows, without it the queue usage is inefficient and may reach critical levels; in fact, at the 3rd and the 48th seconds of simulated time, the gateway queue is overloaded which in turn leads to subsequent throughput drops, as Figure 2.35 indicates. On the contrary, with *DyBRA*

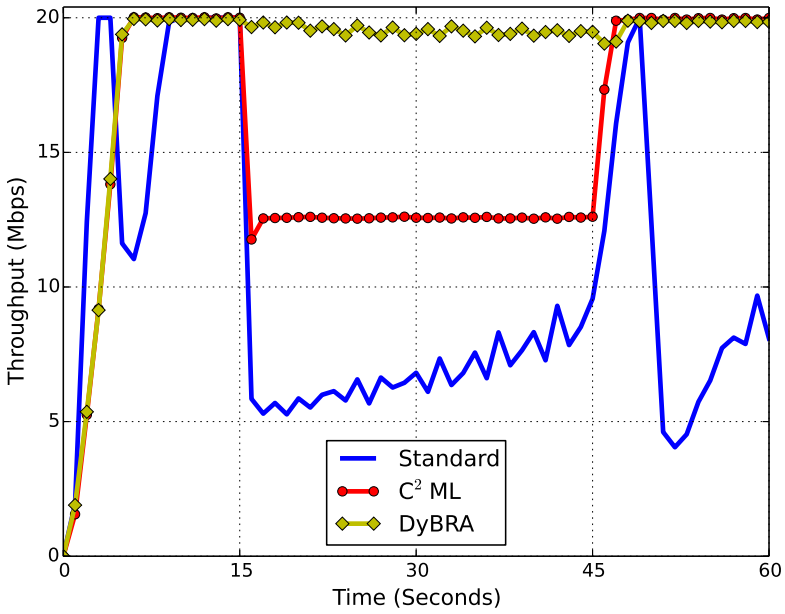


Figure 2.35: Total aggregate throughput.

the queue usage is most of the time negligible, and it never reaches a problematic level even at its peaks. Paradoxically, between the 15th and the 45th seconds of simulated time the queue is only used if *DyBRA* is employed, although it remains very low. Without *DyBRA*, in fact, the empty queue is due to the throughput drop of the nodes that shift to a *BAD* status, coupled with the slow growth of the TCP congestion windows of the hosts that remains *GOOD*; as it is clear from Figure 8, these two effects combined lead to poor aggregate throughput performance in the relative time frame. Subsequent inefficiency is due to the return in a *GOOD* status from the hosts that were *BAD*; because in the mean time the other nodes have increased their congestion windows, the sudden peak of channel utilization fills the gateway queue leading to packets and throughput drop.

Figure 2.35 also shows how without *DyBRA* the overall throughput strongly oscillates between approximately 4 and 20 Mbit/s. With *DyBRA*, instead, the bottleneck link utilization is always optimal in standard conditions and quasi-optimal in the dynamic bandwidth management phase, except immediately after the 45th second mark when the TCP variants need some seconds to regulate their congestion windows. This phenomenon, however, accounts only for an

almost unnoticeable overall throughput reduction. On the contrary, the original C^2ML is unable to adapt to the changes in channel conditions, and the overall throughput suffers a significant drop between the 15th and the 45th seconds of simulated time, as expected. We therefore conclude that *DyBRA* efficiently manages a dynamic allocation of the bandwidth, an efficiency not reachable by TCP alone nor by the only use of C^2ML. These results are confirmed by the recorded total transferred data, which respectively amount to 143.88 MB with *DyBRA* and to 75.93 MB with TCP alone.

DyBRA aims to address generic high-delay links; furthermore, it is not a PEP-like solution, and it should be friendly with respect to both satellite-tailored TCP variants as well as to generic ones. In order to consider the difference in friendliness among TCP variants with or without *DyBRA*, the throughput of nodes that remains in the *GOOD* status has been observed. As Figure 2.36 shows, without *DyBRA* the friendliness among TCP algorithms is impaired; TCP Noordwijk, for example, is very aggressive but does not behave well in case of losses. After the aforementioned queue filling in the 3rd second of simulated time and the subsequent throughput drop, in fact, TCP Noordwijk is unable to recover, partly as a consequence of the already consolidated presence of other TCP variants. On the contrary, with *DyBRA* the friendliness is greatly improved, as it can be seen in the right part of Figure 2.36.

To investigate *DyBRA* scalability, we also performed another set of simulations with only TCP Cubic and a variable amount of nodes (in each simulation, half of the nodes continue to experience a *bad* period). Figure 2.37 reports the results of this experiment in terms of queue space used by the gateway. It is important to remark how *DyBRA* reduces from 1 to 2 orders of magnitude the average queue occupancy, in an inversely proportional fashion with respect to the amount of running nodes. On the contrary, TCP Cubic manifests the opposite trend, where the queue occupancy grows along with the number of running nodes. As a matter of fact, *DyBRA* has a highly beneficial impact on queue size consumption and, consequently, on the latency introduced, resulting in an effective and scalable solution.

2.3.4 Conclusions

For scenarios where a number of client nodes share a gateway in order to access a bottleneck link, this Section has described *DyBRA*, a centralized yet collaborative network resource manager for high delay links. It is able to dynamically

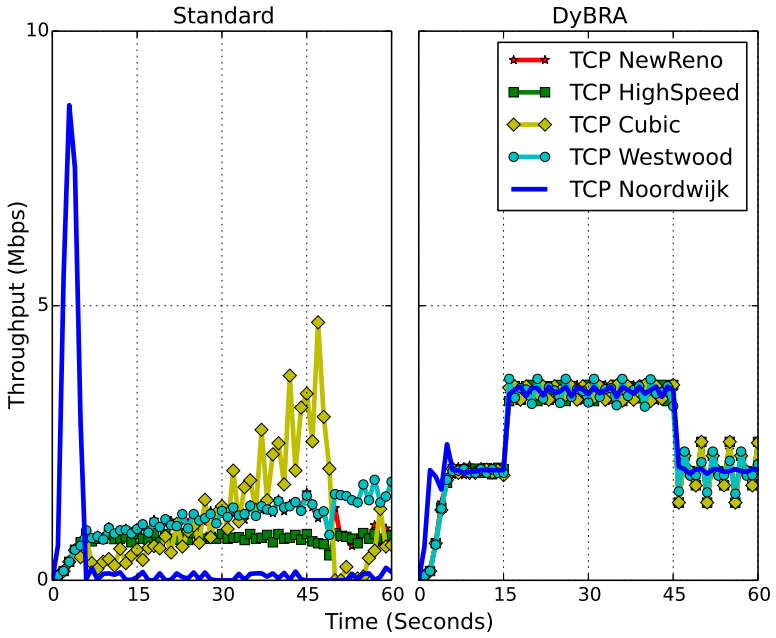


Figure 2.36: Friendliness among TCP variants.

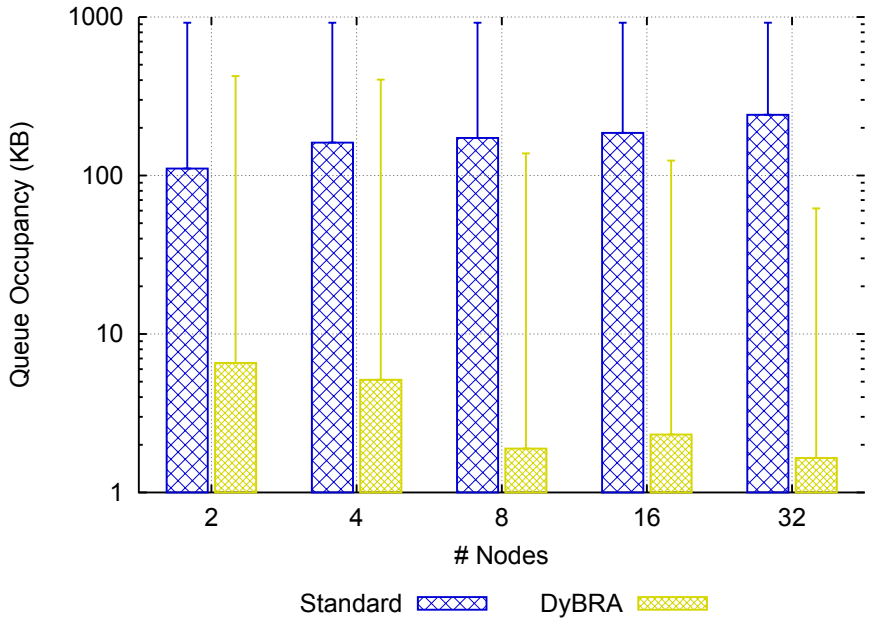


Figure 2.37: *DyBRA* queue scalability.

adapt to significant changes in the client nodes exploitable bandwidth, which can result for example from channel conditions variations that are due to the sudden onset of obstacles between a client receiver and a base station. *DyBRA* considerably improves the queue usage on the gateway, especially during time frames in which the client nodes TCPs congestion windows are in process to converge to a stable size. It also greatly improves the network efficiency, by allowing the set of client nodes to exploit the totality of the bottleneck link resources also in cases of bandwidth underutilization by some client hosts. It is important to note that *DyBRA* has been designed to be applicable to high delay links in general, and it therefore guarantees an almost-optimal friendliness among TCP variants, being these tailored for satellite links or not. As the final remark, we conclude by stating how *DyBRA* inverts the natural tendency to couple the increase of client nodes number with higher queues occupancy, effectively decreasing the gateway queue usage as more hosts are in place.

Chapter 3

Active Queue Management

In this Chapter, we describe AQM as a solution for the congestion control problem, moving from the centralized middleware approach to a general one suitable for internet nodes, regardless the topology. In fact, considering C^2ML, we propose a novel AQM technique called QRM (Queue Rate Manager). QRM is a very efficient algorithm that coupled with C^2ML provides security protection to resource exhaustion attacks, increasing the robustness of the cooperative middleware while preserving features like high fairness and high throughput. We then adapt QRM to create a general and smart no-drop AQM for emergency networks. The solution is uncoupled from the middleware and studied for general PPDR satellite networks. This novel AQM called PINK (Proactive INjection into acK) is deeply investigated and analysed though mathematical formulations. Finally, the algorithm is compared with general AQM solutions for general purpose networks and strong performance analyses are presented over several figures of merit (goodput, latency, RTT variation, fairness and energy consumption).

3.1 QRM: a Cooperative Solution

When a high-delay link is shared among a number of client hosts, both the original TCP and other TCP variants designed to improve performance on those links are unable to effectively make optimal use of the channel. In these scenarios, in fact, the major limit is not represented by a particular congestion control algorithm, nor by the link performance, but instead by the transport protocol fundamentals. TCP infers the channel conditions indirectly, by loss detection and/or ACK-based timing information, and then reacts accordingly. When client hosts are *wirelessly* connected to the gateway that in turn provides them with access to the high-delay link, their collective performance may further degrade as a consequence of the nature of wireless access; the end-to-end congestion control of each node, which is independent from adjacent hosts, may not be able to ensure a high Quality of Experience (QoE).

These kinds of network topologies are often employed for emergency management in post-disaster situations, where deployable networks must be backhauled by satellite channels when terrestrial infrastructures are destroyed. In these cases, shifting to a top-down, centralised and collaborative management of resources from the gateway has the potential to guarantee overall better performance for hosts [25, 53, 60, 66, 67]. To reach this result we previously presented C^2ML [54] in Section 2.2, which aimed at improving QoE for users in such scenarios. We remind to Figure 2.12 for its architecture and protocol details. Basically, C^2ML gives to the gateway the authority of instructing the client hosts on the amount of bandwidth they are allowed to use. This is trivially calculated dividing the bottleneck link capacity by the number of clients to serve, in order to guarantee a fair utilisation of the former.

Disasters may be natural or man-provoked. In the latter case, C^2ML does not provide adequate security guarantees; more specifically, it does not protect legitimate users from Resource Exhaustion Attacks nor flooding techniques aimed at ruining end-users QoE, because it assumes that all users would respect the bandwidth upper bound assigned to them. Making a client host to behave maliciously by purposely forcing it to disobey the C^2ML rules, thus sending data at a rate that exceeds the allowed bandwidth, is an operation that does not require extensive technical abilities and that can easily result in denial of service for all the other hosts. For an attacker, it would thus be possible to

overload the backhaul link by sending a lot of traffic to a remote host. This would surely exhaust the channel resources, effectively cutting out legitimate hosts from the service. Moreover, a client host that for some reason does not update its rate after a gateway order may lead to impaired performance for all the other clients. The Section presents a per-node Queue Rate Management (QRM), an Active Queue Management (AQM) scheme that aims to fill this gap; coupled with any cooperative solution like C^2ML, it detects an attacker flow on the basis of incoming packets, and reacts accordingly by enqueuing or dropping them. We show how, if compared to other AQM schemes such as CoDel, RED, and GREEN, QRM offers better performance and QoS guarantees when coupled with the aforementioned systems. The discussion is organised as follows: Section 3.1.1 presents related work; Section 3.1.2 describes the scenarios that may benefit from centralised and collaborative solutions; Section 3.1.3 presents QRM; Section 3.1.4 details a formal analysis of QRM, and Section 3.1.5 presents simulation results. The conclusions are drawn in Section 3.1.6.

3.1.1 Literature Overview

AQM techniques complement the end-to-end congestion control performed at the transport layer, either explicitly (e.g. through Explicit Congestion Notification) or implicitly (e.g. through packet drops) [68]. The simplest buffer management algorithm, Drop Tail, simply drops whichever packet tries to enter an already full queue, thus it only acts when congestion is already occurred. In time, the main goal of AQM schemes has evolved so as to perform congestion avoidance [69, 70]; a widespread example is RED [20, 21], which keeps the average queue size small by anticipating congestion with a linear dropping function or, like a RED variant called Non Linear RED [71], with a quadratic dropping function. However, it does not always provide fair queuing; protocols that do not adjust their transmission rate according to congestion, such as UDP, or protocols that are aggressive in adjusting it to congestion events, such as TCP variants designed for satellite links [27, 29–31, 40], end up using more bandwidth than other TCP flows. Furthermore, RED needs a manual configuration related to the characteristics of the link in order to behave at its best.

FRED [72] introduces per-flow state information to RED, and it effectively succeeds in improving fairness among flows; however, the drawback is the high quantity of information that has to be maintained on the gateway. CHOKe [73]

has been designed to improve the fairness of RED without the addition of heavy resource needs; because of its probabilistic algorithm, though, some flows may be unnecessarily punished by this scheme.

Newer algorithms, such as WARD [74], CoDel [22,75], FABA [76] and CHOKeR [77], are able to improve fairness among flows, mitigate the bufferbloat problem, and allocate bandwidth proportionally to a prioritisation of flows, respectively. None of them, however, is able to precisely differentiate flows between those operating "legally" and those operating erroneously or maliciously, where the former are flows that behave according to a centralised and collaborative model such as C^2ML, without exceeding the allowed bandwidth that has been assigned to them. Therefore, using any of the above methods to provide security from resource exhaustion attacks in an emergency scenario would inevitably lead to a performance decrease of legitimate users.

GREEN [78] is a rate-based AQM algorithm that is the most similar to our proposal; however, it has been designed for wired links with high capacity, not for high delay links and neither for wireless channels. GREEN operates giving each packet a drop probability that depends on the flow RTT, on the number of active flows, and on the Maximum Segment Size (MSS) encountered during a sampling time. Unfortunately, GREEN holds some drawbacks:

- It has not a reference implementation.
- It is not designed to provide protection against flooding attacks.
- GREEN infers the MSS by considering the highest value only, therefore resulting in a coarse-grained drop policy.
- Sampling time is RTT independent and not specified.
- The RTT measurement assumes the usage of TCP options (i.e. it leaves to end nodes the responsibility to provide RTTs at the gateway; this way, malicious nodes could lie about their actual RTT, therefore cheating the AQM algorithm).

Summarising, GREEN would need a structural redesign and further assumptions in order to be appropriately used with mission-critical high-delay links, in particular in the presence of malicious users.

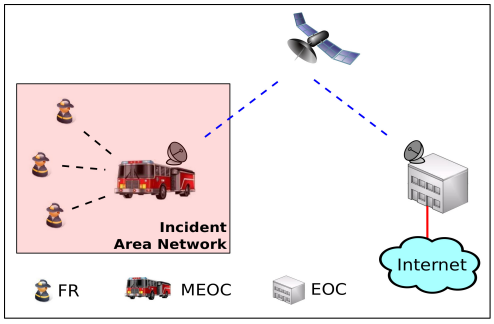


Figure 3.1: Emergency Network.

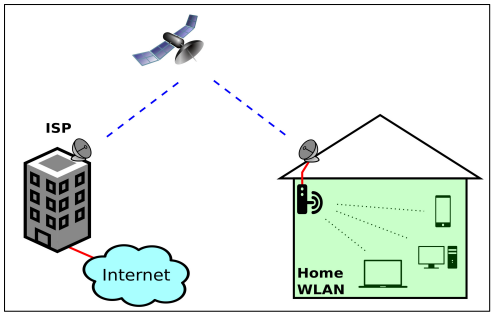


Figure 3.2: Rural Satellite Connection.

Other kinds of flooding attacks may exploit the peculiarities of a certain transmission technology, as it is the case for Signalling Attacks in wireless networks [79]; however, countermeasures [80] are forced to be centred on the specific technology, and are therefore inadequate in the general case.

3.1.2 Enabling a Cooperative Solution

In order to show that cooperative solutions may be effective in very different environments, we introduce two different, but topologically equivalent, scenarios. The first one is depicted in Figure 3.1. The context is emergency networks, and the scenario has been extracted from the EU FP7 *Public Protection and Disaster Relief* project. In such a case, there are a certain number of First Responders who connect to a MEOC (Mobile Emergency Operations-Control Centre) that brings to them LTE coverage, thus representing a deployable (and mobile) LTE repeater station for the operators. The MEOC connects in turn to the EOC (Emergency Operations-control Centre), which is non-mobile and represents the operations headquarters, via a backhaul satellite link in order to be able to communicate with it independently from its position. The second scenario is depicted in Figure 3.2, and it is an example of rural home internet connection through, again, a satellite channel. In such a case, there are several devices which connect to a router that brings to them WiFi coverage. The router connects in turn to the Internet Service Provider (ISP) via a satellite link. In both these scenarios, there are a finite number of clients competing for the high-delay satellite bottleneck, a particularly challenging situation for TCP connections.

In Figure 3.3 we show, on the left, the rate achieved by 5 TCP flows continuously backlogged which are sending data through the satellite channel, together

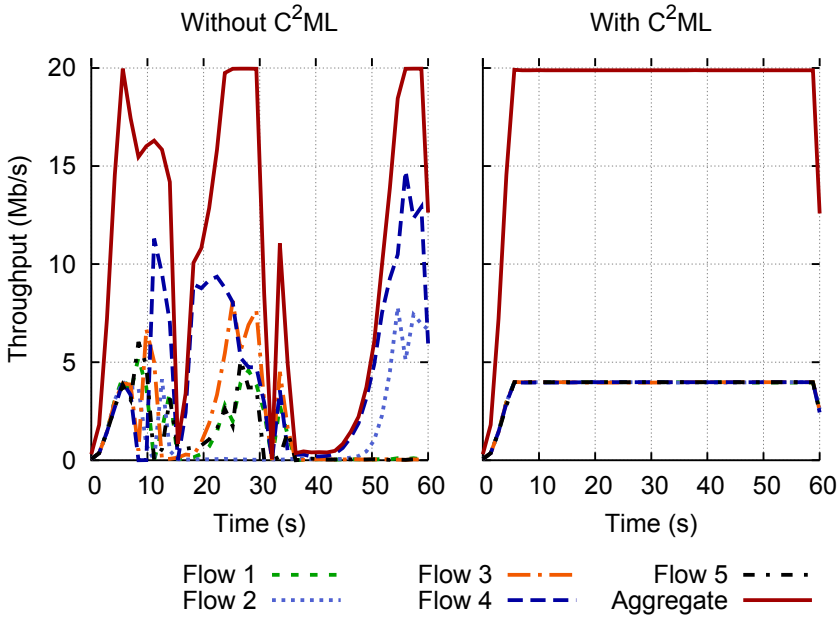


Figure 3.3: Benefits of C^2ML on channel exploitation.

with the cumulative rate, to give an idea about their channel exploitation. The link capacity is 20 Mbit/s. It is possible to notice how these 5 flows struggle to find a fairness balance in such a high delay link, resulting in continuous drops and channel underutilisation. On the contrary, the right part of the same picture highlights in a single plot how with C^2ML they are able to optimally exploit the high delay link capacity, resulting in a stable and fair network environment. It is remarkable how this solution is also TCP variant independent (i.e. it provides fairness regardless of the TCP variants employed by client nodes, as all window-based TCP variants are adapted in the same way and utilise the same functions when an ACK is received [81]) and effective in terms of delay, queue occupancy and latency, as demonstrated in Section 2.2 [54, 81].

3.1.3 The QRM algorithm

In the TCP/IP networking stack, QRM should be placed at the IP level, just as any other Queueing Discipline. As it needs information about bandwidth allocation, here QRM is coupled with C^2ML. It is worth to note that any other bandwidth allocation manager system could be used; therefore, QRM is not intrinsically bound to C^2ML, and it can be easily adapted to any analogous

```
procedure DoInEnqueue(pkt)
    if pkt.flags ≠ (SYN) ∨ (SYN ∧ ACK) then
        if ¬IsIPAllowed(pkt.dest) then
            return false
        end if
    end if
    TrackRcv(pkt.dest, pkt.ack)
    return true
end procedure
```

Figure 3.4: Pseudocode for an incoming packet

environment. Before introducing the design principles, we should state that C^2ML should be modified in order to exchange the allowed rate with the queue algorithm, an operation that could be done when a node joins or leaves the network. We will identify the allowed bandwidth for each client host with the BW_i variable, and we assume that a client sends firstly a SYN message to a remote destination, and then sends to the gateway a `CLIENT_HELLO` packet when the SYN-ACK is received. The gateway should also maintain a *white list* of clients inside the LAN, updating it each time a `CLIENT_HELLO` or a `CLIENT_BYE` message is received. This list is populated by users that have been authorised to use the service, and it represents a first line of defence because it allows to block any activity coming from clients that are not registered inside C^2ML.

We first analyse the operations involved on the gateway each time a packet travelling from a remote host to a client node is received. The relative pseudocode is shown in Figure 3.4. In this case, the source address is an outside-LAN IP, while the destination address is an inside-LAN IP. A packet with either the SYN or the SYN-ACK flag should be enqueued regardless of the destination, because it is needed to open the end-to-end TCP connection; in all the other cases, a check on the destination IP is required. If this is not in the *white list*, the packet should be rejected and thus immediately dropped. If the packet is accepted, before enqueueing it the destination IP and the ACK number should be tracked (with TrackRcv()) in order to calculate the average RTT of the connection. Trivial checks are enough to restrict SYN or SYN-ACK flooding attacks. For sake of brevity, we omit these checks in this description, but they can be found in the provided source code [82].

When a packet is travelling from a client node to a remote host, the operations involved on the gateway are showed in Figure 3.5. The checks over the SYN and

```
procedure DoOutEnqueue(pkt)
    if pkt.flags ≠ (SYN) ∨ (SYN ∧ ACK) then
        if ¬IsIPAllowed(pkt.src) then
            return false
        end if
    end if
    if pkt.flags = (SYN) then
        TrackSent(pkt.src, pkt.seq, pkt.size)
        return true
    end if
    if BYTE_OF(pkt.src) / MIN_RTT_OF(pkt.src) ≥ BW_i then
        return false
    end if
    TrackSent(pkt.src, pkt.seq, pkt.size)
    return true
end procedure
```

Figure 3.5: Pseudocode for an outgoing packet

SYN-ACK packets, as well as the IP checks (in this case the source is an inside-LAN host and the destination is an outside-LAN host) are unchanged. Therefore, *SYN* packets should be always accepted and tracked (TrackSent()), in order to calculate the RTT when the corresponding *SYN-ACK* is received. On the contrary, packets without the *SYN* flag should be enqueued only according to the following condition: if the total amount of bytes transferred in the minimum RTT that has been measured from the source node (inside-LAN) is less than the allocated bandwidth, then the packet should be enqueued. Updating this byte amount is a matter of updating a single variable for each registered client when a packet is received. When the first packet for a registered client is enqueued, a timer starting from a value equal to the last measured RTT is created; when it reaches zero, the byte amount is reset.

The algorithm presented so far is not optimal. Firstly, the delay inside the LAN is not taken into account when the gateway calculates the RTT, as the time difference between a packet and the relative *ACK* is measured in the queue of the gateway; as a consequence, the client may see a greater RTT value than the gateway. As the C^2ML specification demands, the client will react by opening slightly its congestion window. This could be seen as an exceeding rate by the queue, which would consequently drop the exceeding packets. This issue can be resolved by introducing a tolerance on the bandwidth threshold: this should be

a parameter, as LAN delay may vary.

We conclude this Section by stating that QRM needs only $\Theta(n)$ memory (in addition to the memory needed for the queue itself), where n is the number of active nodes (not flows). FRED [72], as a comparison, has in addition to the queue a memory cost of $\Theta(m)$, where m is the number of active flows, and it always holds that $m \geq n$. To complete the description, in the next section we will formally discuss the complexity of QRM.

3.1.4 QRM guarantees

In this section the effect of the QRM algorithm on the gateway queue will be formally analysed. In particular, by controlling the maximum flow rate achievable by each node it is possible to calculate the maximum gateway queue size with a worst-case upper bound value.

Theorem 2. *Let $Q_{gw}(t)$ be the output queue length of the gateway at the generic time t and let RTT_{net} be the average round trip time of the network. If the link bandwidth is constant and equal to BW, with the QRM algorithm the following inequality holds for any time t:*

$$Q_{gw}(t) \leq BW \cdot RTT_{net}. \tag{3.1}$$

Proof. Consider a generic i-th node equipped with C^2ML, if it is a node behaving legally it will be sent in the network a burst B_i which is at most equal to the product $BW_i \cdot RTT_i^{min}$ each RTT_i, as showed in Figure 3.5. The burst size cannot exceed that value due to the QRM policy that limits the flows bandwidth. Consider the case in which all nodes want to send their bursts simultaneously, at the same time t. The gateway queue, immediately before t, can be either empty or containing packets; this divides the proof in two cases.

Case 1: Empty queue. The gateway queue is filled with the aforementioned bursts and it follows that:

$$\begin{aligned} Q_{gw}(t) &= \sum_i B_i \\ &= \sum_i BW_i \cdot RTT_i^{min}. \end{aligned} \tag{3.2}$$

Considering RTT_i^{min} upper bounded and equal for each node to RTT_{net}, we can write:

$$\begin{aligned} Q_{gw}(t) &= \sum_i BW_i \cdot RTT_{net} \\ &= (\sum_i BW_i) \cdot RTT_{net}. \end{aligned} \tag{3.3}$$

Here we can use the C^2ML bandwidth rule $BW \geq \sum_i BW_i$, by substituting it to (3.3) we get the thesis.

Case 2: Non-empty queue. Let the gateway queue, immediately before t, contains at least one packet p. After receiving the aforementioned bursts it follows that:

$$\begin{aligned} Q_{gw}(t) &= p + \sum_i BW_i \cdot RTT_i^{min} \\ &> \sum_i BW_i \cdot RTT_i^{min} \\ &= BW \cdot RTT_{net}. \end{aligned} \tag{3.4}$$

It can be proved that this contradicts our assumptions. The packet p, for the QRM policy (see Section 3.1.3), belongs to one of the n C^2ML nodes. To continue, consider that the packet p belongs to the i-th node. If follows that, at time t, the gateway queue would contain an amount of packets belonging to the node i which is:

$$\begin{aligned} p + BW_i \cdot RTT_i^{min} &> BW_i \cdot RTT_i^{min} \\ &= B_i. \end{aligned} \tag{3.5}$$

The amount of packets in the queue, belonging to the node i, would be greater than the product $BW_i \cdot RTT_i^{min}$; this is absurd because it violates the QRM property, i.e. the algorithm must not enqueue one of these packets.

To conclude, by analysing both case 1 and case 2 we completed the proof. Theorem 2 is proved. □

This way it is possible to have an upper bound guarantee about the queue length and, consequently, an upper bound about the maximum queueing delay introduced by the gateway. A key point of Theorem 2 is that this deterministic bound is equal to the standard suggested buffer size [19], which in turn means

that it is not necessary to modify the gateway queue in order to enable C^2ML to work in an already deployed system. Moreover, Theorem 2 let us also prove the effectiveness of the QRM algorithm.

Corollary 3. *The space complexity of QRM is linear.*

Proof. In light of Theorem 2 we know that the queue occupancy of the gateway at a generic time t is upper bounded by the bandwidth-delay product. However, the used queue memory is not the only space memory allocated by the QRM algorithm. In fact, it also needs (i) the list of the allowed nodes and (ii) the list of packets waiting for RTT tracing (i.e. packets waiting to receive ACKs and, if it is the case, to update the RTT_i^{min} of a node i).

For the former, as a first hypothesis, we can assume that the amount of network nodes is a constant value, which is fairly true in a wireless (cooperative or not) environment where this quantity is bounded also by physical layer constraints[1]. Moreover, this memory is already allocated and used by the C^2ML layer, hence even without this weak assumption it can be considered to be not an extra memory required by QRM.

Let us now consider the latter, i.e. the list of packets waiting for RTT tracing. By Theorem 2 we know that, each RTT_{net}, the amount of data collected in the queue is deterministically bounded (Equation 3.1); this means that in the following RTT_{net} the amount of memory waiting to be traced is again bounded by the same value. In fact, packets do not need to wait to be traced if the RTT_{net} expires (e.g. if an ACK has been lost or a congestion has occurred), this safely means that RTT_i^{min}, for all nodes i, will not be updated in this sampling period. Consequently, the amount of memory used by QRM is $\mathcal{O}(b)$ for the queue and $\mathcal{O}(b)$ for the tracing, thus concluding the proof in a cumulative linear space complexity of $\mathcal{O}(b)$. □

Corollary 4. *QRM is computationally efficient, with $\mathcal{O}(b)$ space complexity and $\mathcal{O}(1)$ time complexity.*

Proof. By Corollary 3 we know that QRM is efficient in terms of used memory, which is $\mathcal{O}(b)$. Considering Figure 3.4 and Figure 3.5, the enqueue and dequeue functions of QRM, respectively, it is easy to infer that the number of instructions

[1]It is important to notice that QRM allocates such a memory as a function of the nodes number and not of the flows number, which can be far greater.

required for this executions is constant and no cycles are computed. This way, we can claim that QRM is also efficient in terms of time complexity, which is $\mathcal{O}(1)$, thus concluding the proof. □

In the next section the effectiveness of our model will be validated by simulation results, which shows an actual queue occupancy level that is well below the theoretical worst case value provided by Theorem 2 due to the reasons that follow.

- The channel bandwidth between client nodes and gateway is not infinite. This means that the gateway starts to transmit packets while it is still receiving packets from a node.
- Perfect synchronization among nodes is hard to achieve, especially due to the previous synchronization with the gateway that each node must perform before starting to transmit.
- The TCP-like algorithm adopted in C^2ML uses burst transmissions only in the slow-start phase. In the congestion avoidance phase each packet is transmitted when an ACK is received, leading to a time-spreaded packet dispatch, not a bursty one.

3.1.5 Performance

In this section, we present the simulation environment and the collected results by analysing QRM behaviour with respect to well known AQM techniques. We will evaluate performance like throughput, fairness, latency, RTT variations and queue occupancy by considering different numbers of clients node and different numbers of malicious nodes (attackers). The general topology used for tests is depicted in Figure 3.6 and is the topological representation of Figures 3.1 and 3.2 introduced in Section 3.1.2, while Table 3.1 summarises the main parameters used to configure our experiments. All simulations have been performed with the ns-3 network simulator.

The network topology consists in a number of clients varying from 2 up to 32 nodes that connect to the gateway in order to reach an external host; both the gateway and the external host are modelled with one ns-3 node each. Client nodes and gateway are connected through point-to-point ns-3 links with

parameter	value
simulation time	60 seconds
number of nodes	2-32 clients, 1 gateway, 1 remote host
number of attackers	0-16 between the clients
bottleneck bandwidth	20 Mbit/s
bottleneck delay	350 ms
number of TCP flows	1 per client
TCP used	TCP Cubic
MSS	1000 bytes
Gateway queue size	1805 Kbytes
AQM algos	DropTail, RED, Codel, GREEN, *QRM*
data to transmit	unlimited

Table 3.1: Simulations parameters.

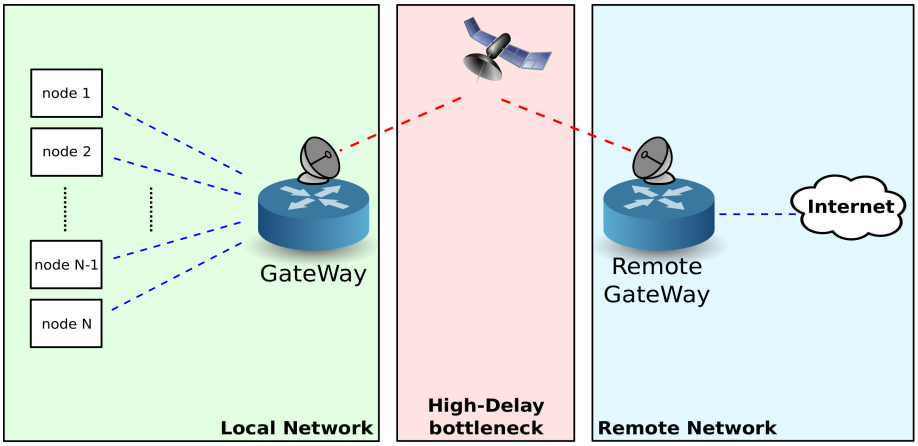


Figure 3.6: Topology of the experiments.

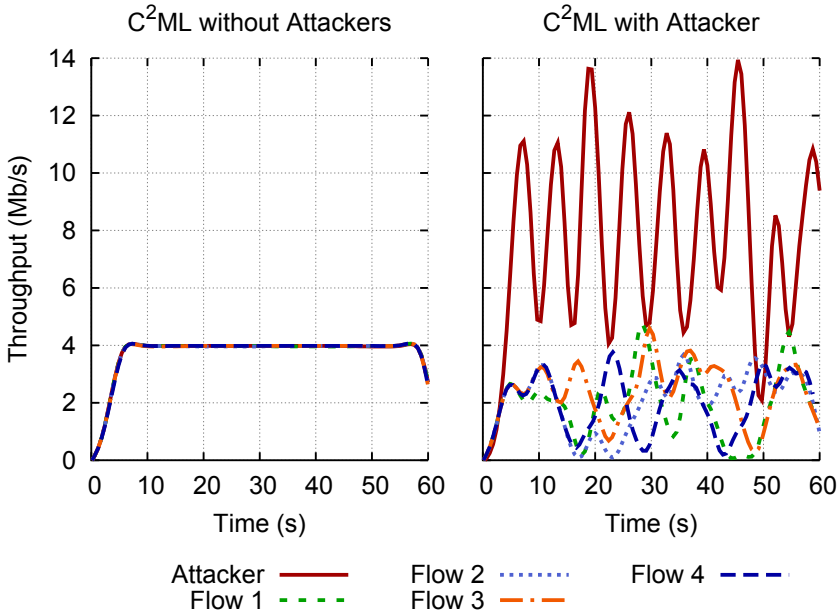


Figure 3.7: Drop Tail throughput.

a bandwidth of 50 Mbit/s and a delay of 1 ms each, while gateway and remote node are connected through a point-to-point ns-3 link with a bandwidth of 20 Mbit/s and a delay of 350 ms. TCP Cubic has always been used as the Transport protocol. Because *each* client node has 50 Mbit/s available between itself and the gateway, while the total bandwidth of the high-delay link is 20 Mbit/s, the latter can be safely defined as the bottleneck link.

3.1.5.1 Experiment one: needs of AQM

The simulations of this first set of experiments have been performed following the parameters summarised in Table 3.1 with 5 active client nodes.

We now focus on what happens when one node violates the cooperative protocol and tries to perform a resource exhaustion attack by flooding the network with a huge amount of packets i.e. exceeding the rate assigned by C^2ML to the node. The effects of such an attack are shown in Figure 3.7. The chart on the left shows the fair bandwidth allocation guaranteed by C^2ML when the five client nodes are behaving correctly, equivalently to the experiment reported in Section 3.1.2. The chart on the right, instead, shows the attacker in action; in this situation, 4 of the 5 total nodes are behaving correctly, while one node behaves

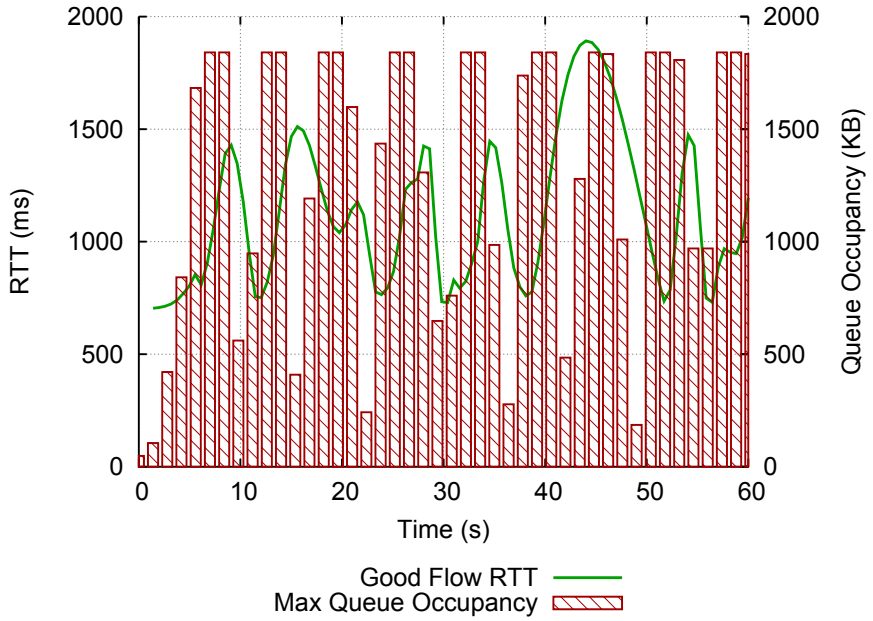


Figure 3.8: Drop Tail queue flooding: RTT and queue occupancy.

as the attacker; simply by trying to increase its data rate like a regular TCP, the attacker halves the actual bandwidth that is available for legitimate hosts. This is the behaviour of an attacker that would try to disguise itself as a regular TCP flow; a malicious user motivated to completely deny service to legitimate users could simply flood the gateway "turning off" its congestion control.

A disguised attacker damages not only the fairness that would be guaranteed with C^2ML, but also the low queue occupancy guarantees, resulting in higher latency and higher RTTs. Figure 3.8 highlights the effect of an attacker to the queue usage and the RTT experienced by a legitimate node during the same experiment. We do not activate any high-performance AQM for this first result, leaving the gateway queue behaving in a standard drop-tail fashion. From Figure 3.8 it is also possible to note that the maximum queue occupancy of 1805KB is reached several times during the experiment, causing packet drops and RTT peaks of 1.8 seconds, almost doubling the benchmark network RTT of 0.7 seconds.

In order to mitigate the adverse effects of such an attacker on the global network behaviour, we first tested the most widely deployed AQM techniques, namely RED and CoDel, together with GREEN. Figure 3.9, Figure 3.10 and

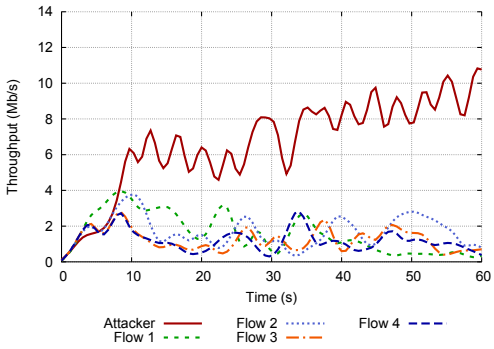


Figure 3.9: CoDel queue flooding: RTT and queue occupancy.

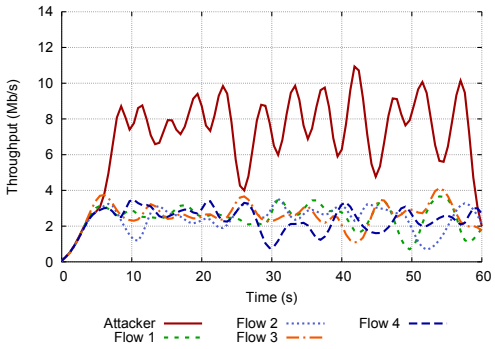


Figure 3.10: RED queue flooding: RTT and queue occupancy.

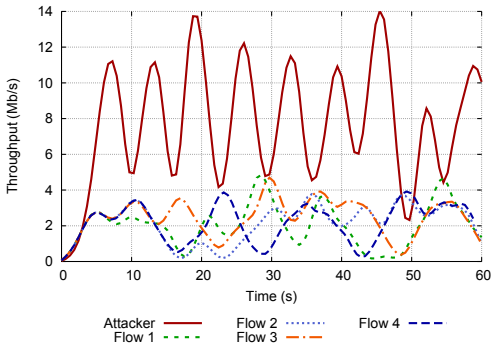


Figure 3.11: GREEN queue flooding: RTT and queue occupancy.

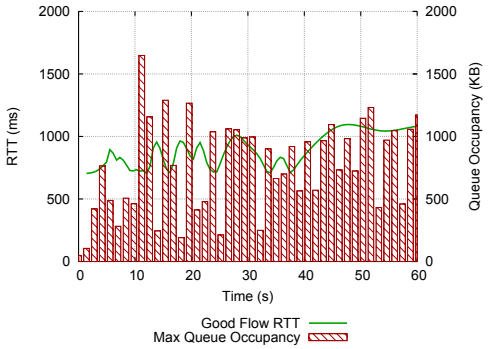


Figure 3.12: CoDel RTT and queue occupancy with attacker

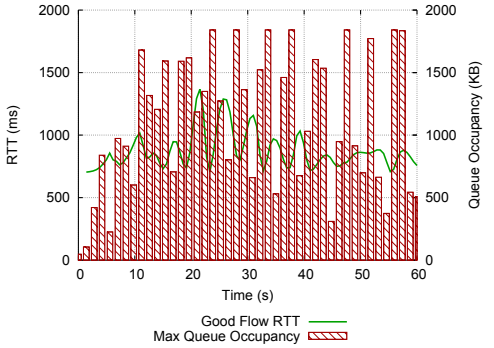


Figure 3.13: RED RTT and queue occupancy with attacker

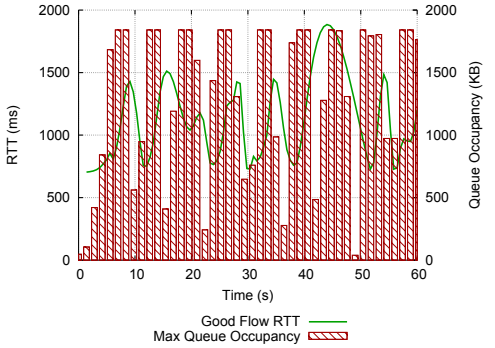


Figure 3.14: Green RTT and queue occupancy with attacker

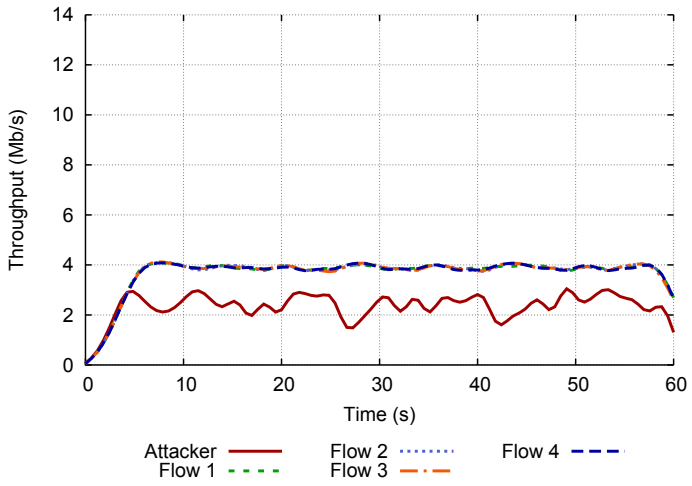


Figure 3.15: QRM throughput with attacker.

Figure 3.11 report the results with CoDel, RED and GREEN, respectively. The Figures show how the use of a standard AQM technique slightly mitigates the attacker link monopolization; at the same time, however, the dropping policies also involve packets belonging to legitimate nodes, leading to high throughput oscillations and low global fairness. These poor results are attributable to the fact that the AQM policy is based on the queue level and not on the rate achieved by different flows. Results about queue occupancy and RTT variation are reported in Figures 3.12, 3.13 and Figure 3.14 for CoDel, RED and GREEN, respectively; it is possible to notice that CoDel outperforms RED in terms of queue occupancy, never reaching the maximum queue level and containing the RTT in a more stable and lower range. We expected GREEN to perform better because of its rate-based behaviour, but instead its performance are very similar to the ones obtained with a simple DropTail technique; this is due to the probability drop function of GREEN that results in very low drop chances with high RTTs. In fact, GREEN is tailored for short RTT networks with a high number of flows.

To complete the picture, the same experiment has been also conducted with the proposed QRM solution. The results of this experiment are summarised in Figure 3.15 and Figure 3.16. Figure 3.15 reports the throughput achieved by both the legitimate nodes and the attacker by showing how, using QRM, the attacker behaviour is totally mitigated; in fact, its maximum achievable throughput is now equal to the bandwidth assigned by the gateway. Moreover, legitimate nodes maintain the C^2ML fairness property, achieving a nominal throughput of

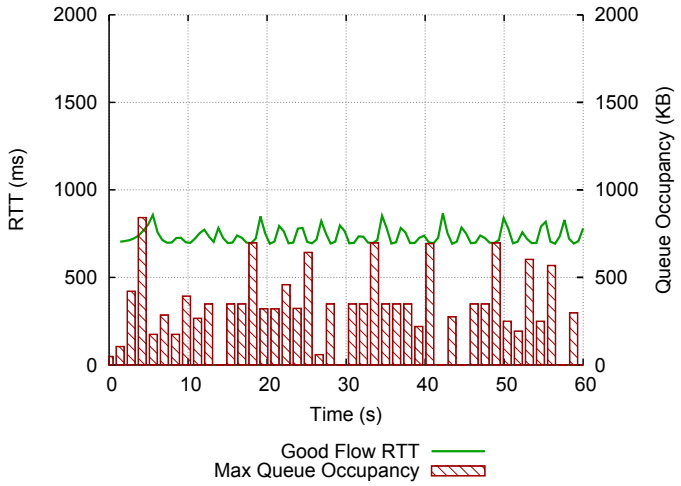


Figure 3.16: QRM queue flooding: RTT and queue occupancy.

4 Mbit/s that is equal to the gateway assignment. Figure 3.16 reports the RTT experienced by legitimate nodes and the maximum queue occupancy achieved by the gateway in each second of simulated time. The depicted results are totally aligned with the standard C^2ML property even in the presence of a malicious node, while queue occupancy levels are compliant to the theoretical upper bound imposed by Theorem 2. Furthermore, the RTT variation for each good client node is contained between the network benchmark of 700 ms and the maximum encountered value of circa 800ms, demonstrating QRM stability.

3.1.5.2 Experiment two: nodes goodput with a malicious node

In a second set of simulations, we focused on the amount of data correctly sent by each good node in the presence of a malicious attacker. We designed several simulations as a function of the number of active client nodes, that range from 2 to 32 in an exponential fashion. For each simulation, only one of these clients was an attacker. We will call $TxData_i$ the data correctly received from node i (i.e. the goodput of node i) at the end of the experiment. Figure 3.17 shows the transmitted data of all nodes (minimum, average and maximum value in a box-style plot) for all the queueing disciplines as a function of the active nodes. According to the results depicted in Figure 3.17, QRM outperforms the standard AQM techniques by always achieving higher values of $TxData_i$, also restraining the variance between the minimum $TxData_i$ and the maximum one, therefore

guaranteeing a high fairness independently from the number of active flows. At the same time, it is possible to notice how RED, which in the previous set of experiments performed poorly compared to CoDel in terms of latency and queue occupancy, achieve here better results in terms of average goodput with respect to CoDel. GREEN also obtains remarkable results if compared to the previous set of experiments showing good scaling properties, with results comparable to the RED algorithm.

To better understand the benefits provided by RED, CoDel, GREEN and, in particular, QRM, we introduce the following as the definition of worst-case goodput:

$$wcTxData = \frac{\min_i(TxData_i) \cdot n}{BW \cdot SimulationTime}$$

where $\min_i(TxData_i)$ is the minimum amount of correctly received data from any nodes (i.e. the amount of data correctly received from the unluckiest one), BW is the bottleneck link bandwidth, $SimulationTime$ is the duration of the simulation expressed in seconds, while n is the number of active clients (including the attacker). The meaning of the formula is that $\min_i(TxData_i)$ is divided by $\frac{BW \cdot SimulationTime}{n}$, which is the maximum amount of data transmitted by each node in a perfectly fair and backlogged environment during a simulation. By doing this, $wcTxData$ is a pure number, always in the range $[0, 1]$[2]; the higher $wcTxData$ is, the higher are the system fairness and the goodput in the worst case.

To give an example, Figure 3.18 shows the $wcTxData$ of the same experiment of Figure 3.17, highlighting that while QRM provides high and stable value of this worst case quantity, CoDel, RED and GREEN provide a growing rate of such an indicator with respect to the DropTail technique which suffers from the increase of client nodes.

3.1.5.3 Experiment three: nodes goodput with multiple attackers

In this third set of simulations, we focused on the amount of data correctly sent by each good node in the presence of multiple malicious attackers. This time we

[2]The $wcTxData$ value of 1 (i.e. perfect fairness and complete channel exploitation) is almost impossible to achieve during our simulations because of the initial slow start phase of TCP which do not completely fulfil the satellite channel for the first few seconds of each simulation.

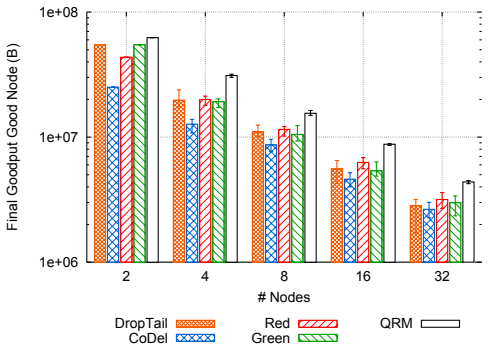


Figure 3.17: $TxData_i$ for different AQM, one attacker.

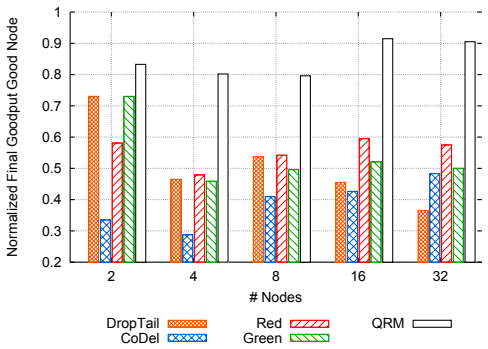


Figure 3.18: $wcTxData$ for different AQM, one attacker.

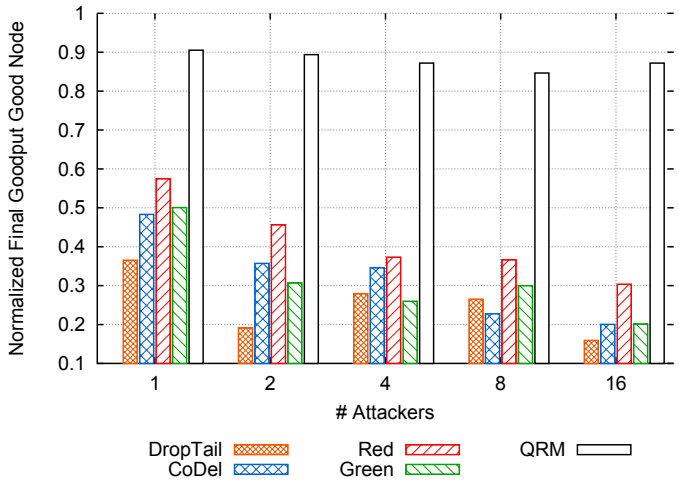


Figure 3.19: $wcTxData$ for different AQM with different attackers (32 nodes).

set a static value of 32 active client nodes and designed several simulations as a function of the number of attackers, which range from 1 to 16 in an exponential fashion. We summarise the result of this experiment in Figure 3.19, where $wcTxData$ represents the data correctly transmitted by good nodes in the worst case, and it is reported as a function of the malicious users number. In these simulations as well, the stability of QRM is remarkable; in fact, $wcTxData$ is almost constant and always close to 90%. However, the same is not true for CoDel, RED and GREEN, that decrease $wcTxData$ when the number of attackers flooding the gateway increases.

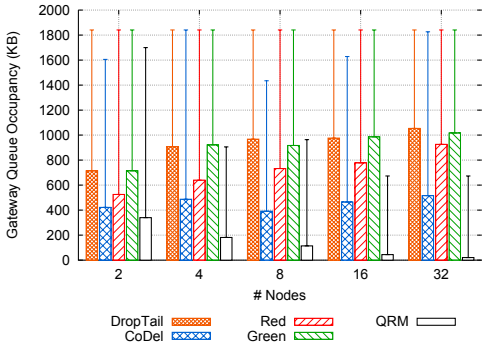


Figure 3.20: Queue occupancy with one attacker.

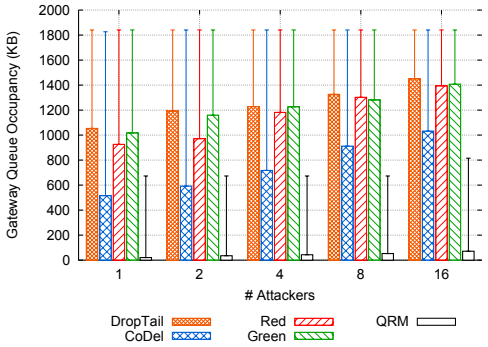


Figure 3.21: Queue occupancy with different attackers (32 nodes).

3.1.5.4 Experiment four: gateway queue occupancy

In this last set of simulations we focused on the queue level reached by the gateway. We checked this quantity during the simulations of Experiments 3 and 4, that are represented by Figures 3.20 and 3.21, respectively. Both figures report the average and the maximum level of gateway queue occupancy during each simulation[3]. Figure 3.20 highlights that, with a single attacker in the network, the average queue occupancy grows with DropTail, RED and GREEN as a function of the active nodes number; CoDel, instead, manifests an almost stable average queue occupancy, while QRM is the only AQM technique able to reduce both the average and the maximum queue level as a function of the number of nodes. Moreover, the growing queue occupancy rate with DropTail, RED and GREEN is even more manifest in Figure 3.21. The same Figure also shoes how CoDel, this time, is more affected by the increment of the attackers, and presents a growth in the average queue size. QRM instead, demonstrates again a stability in both average and maximum queue occupancy. It is important to notice that all of the QRM values are definitely compliant to the worst case guarantees bound provided by Theorem 2, therefore never reaching the maximum queue size placed at the Bandwidth-Delay Product value; in fact, it achieves a queue occupancy amount that is almost always lower than the half of the maximum value.

[3]The minimum amount of queue occupancy is always 0 KB

3.1.6 Conclusions

In this Section we described QRM, an Active Queue Management scheme that aims to provide protection against resource exhaustion attacks in scenarios where access to a shared, high-delay link is controlled by a centralized and collaborative management system, such as C^2ML. This security property may be especially useful in mission-critical systems, such as emergency networks, that could be vulnerable to malicious activities. In fact, disasters may be man-provoked, and in these cases there may be interest to disturb or hinder communications used for disaster recovery operations and coordination among public services. It is shown how QRM provides protection from these attacks while guaranteeing optimal throughput for legitimate users, thus avoiding to degrade system performance. We formally proved that the Bandwidth-Delay Product of the channel is the upper bound value of the gateway queue size when QRM is employed. Furthermore, through simulation results we showed how QRM performs better, in the aforementioned scenarios, when compared to RED, CoDel or GREEN; in fact, it effectively counteracts the threat, allowing legitimate users to continue their network activities undisturbed by guaranteeing high fairness and QoS. Moreover, we showed how the actual queue occupancy levels are far lower than the theoretical upper bound, leading to minimum RTTs, that in turn allow the efficient utilisation of a high-delay link. In the following Sections we will try to maintain these remarkable features even without a centralize environment nor a collaborative management system.

3.2 PINK: a General AQM for PPDR Systems

Original TCP and ad-hoc TCP variants are unable to effectively make optimal use of high delay channels. Indeed, in some scenarios their major limit is not represented by a particular congestion control algorithm, but instead by the transport protocol fundamentals; TCP, in fact, infers the channel conditions indirectly, by loss detection and/or ACK-based timing information, and then reacts accordingly.

The situation may even get worse in cases where client hosts are wirelessly connected to the gateway that in turn provides them with access to a high-delay link, because TCP has no mechanism to differentiate losses that are due to channels congestion to losses that are due to wireless links errors, resulting in a degradation of the aggregate performance.

This kind of network topologies are often employed in Public Protection and Disaster Relief (PPDR) communication systems for the emergency management in post-disaster situations; in these challenging scenarios, deployable networks must be backhauled by satellite channels when terrestrial infrastructures are disrupted or destroyed. There is also the further complication to aggregate different operators traffic (fire brigades, air forces, police etc.), when each of them may employ different TCP algorithms with different (usually high) Round Trip Times (RTTs) that may depend on the position of the related headquarters. These constraints together pose remarkable difficulties in controlling network congestion and providing fairness among TCP flows.

To deal with these issues we designed PINK, a Proactive INjection into TCP acKnowledgements Active Queue Management (AQM) algorithm, able to impose fair sharing of a bottleneck link resources among competing client TCP flows, regardless from the TCP version employed and independently to each flow RTT. This makes PINK suitable for several network conditions, and in particular for PPDR-systems.

The discussion is organized as follows: Section 3.2.1 summarizes related work. Section 3.2.2 describes our proposal. Section 3.2.3 introduces the PPDR validation scenario, while Section 3.2.4 shows the simulations results. In Section 3.2.5 the conclusions are drawn.

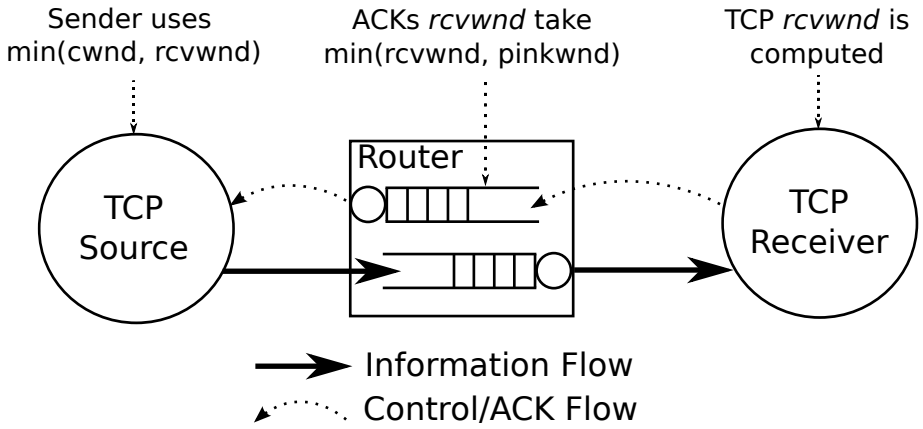


Figure 3.22: PINK algorithm on a simple network.

3.2.1 Suitable AQMs for PPDR Systems

In this section we carefully revise the literature involving feedback-based congestion control algorithms.

An approach to make TCP cooperate with routers is the Explicit Congestion Notification (ECN) proposed in [35]. The authors presents an algorithm able to extract the network status from subsequent binary congestion information and to estimate a fair window size. With this solution, the window size reduction is identical to the one induced by the fast retransmit mechanism of TCP Reno and it does not therefore require packet drops. From one side, the ECN technique is effective in reducing packet losses but, on the other, binary feedback alone is not enough to avoid window and network oscillations or to operate fine-grained adjustments.

One of the first attempts to explicitly control network congestion through a more substantial feedback has been proposed by Gerla et al in [83]. The authors proposed a Generalized Window Advertising (GWA) for TCP congestion control that aims to avoid window oscillations and the related fluctuations in the offered load and, consequently, in network performance. The key idea of GWA is to operate on the Receive Window (RWIN) set by the receiver in the header of TCP ACK segments in order to convey both the receiver available buffer space and the network congestion status. Unfortunately, GWA requires a modification of the existing protocol stack of end-nodes, because congestion information must be transported for the GWA-TCP sender at the IP level.

A similar approach is described in [84, 85]. This work also proposes the use of the RWIN field in TCP headers to convey network congestion notifications. Even if both the congestion control algorithm and the protocol implementation are different to the GWA technique, this approach holds the same drawbacks, requiring modification to the end hosts stack. Moreover, both GWA and this work rely on the router queue length, that could represent an outdated congestion information when high RTTs channels (like satellites) are in place.

A substantial improvement in this field has been provided by Barbera *et al.* in [86, 87]. In these works, the authors modify the Receive Window of the TCP ACK in a transparent way, with no need to modify the end host protocol stack. The solution is simple and focuses on congestion avoidance, latency reduction and jitter containment. Unfortunately, such an approach is designed controlling the gateway queue, differently from our approach which is rate-based. For this reason, the solution proposed by the authors is not suitable in case of RTTs variations like in PPDR systems.

3.2.2 The PINK algorithm

PINK is a *per-flow* yet *stateless* AQM algorithm, designed to be deployed on a gateway that provides network access to a set of client hosts, in scenarios where it is critical to guarantee operativeness and fairness, as it is in PPDR networks. It operates *per-flow* because it divides the bottleneck bandwidth over the number of active flows on the channel, and it enforces the resulting value by modifying the TCP Receive Window of ACKs belonging to each flow. It's *stateless* simply because it does not maintain any state associated with flows, but it always acts the same way.

Conceptually, PINK operates as illustrated in Figure 3.22. It *prevents* congestion by forcing the TCP flows of client hosts to regulate their windows on the basis of what PINK itself decides. It does so by "hacking" the TCP Receive Window of every ACK that passes through the gateway, so as to impose an upper bound to the flows windows. When a client host flow receives its ACK, it regulates the amount of segments to send on the basis of its Congestion Window, the outstanding segments and the freshly modified Receive Window.

The information that the algorithm necessitates are the number of active connections, the RTT of each flow and the shared link bandwidth. The latter is

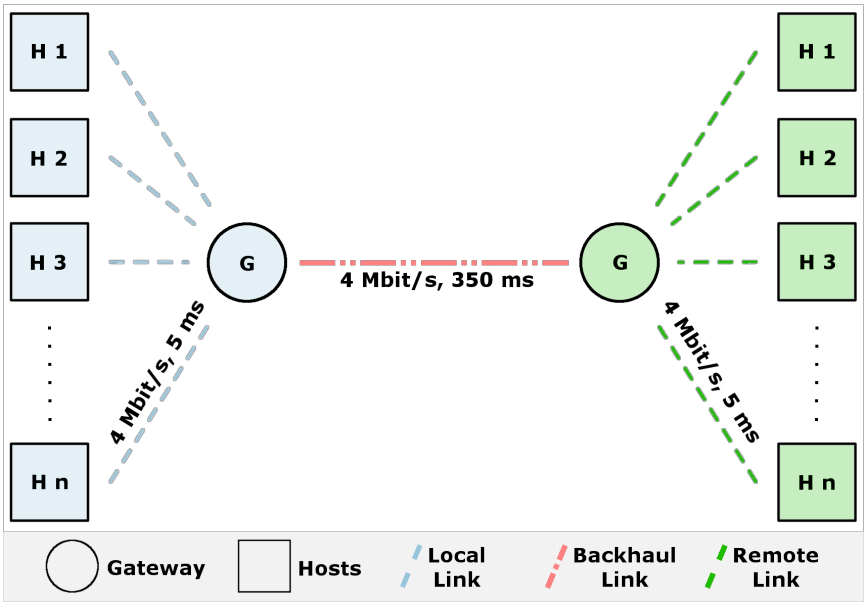


Figure 3.23: Simulated network

assumed to be an information always available in advance to the gateway; the others, instead, can safely be computed at run-time. Calculating the number of active connections is trivial, and this number needs only to be updated when new flows pass through the node. It is besides possible to efficiently calculate each flow RTT [88–90] on the basis of packets crossing the gateway. Remarkable is the fact that PINK can be coupled with any queue scheduler; in a single scheduler queue, in fact, all flows are weighted the same [91], an attribute that it is not violated.

A possible constraint to the algorithm deployment may be the performance overhead introduced by the need of recomputing the TCP Checksum after the alteration of the TCP Receive Window, an operation that must be done for each and only ACK *that needs to be changed*. However, this performance degradation can be avoided by employing equipment that compute hash functions through specialized hardware [92], instead of performing these operations in software.

3.2.3 Simulation Environment

This Section presents the simulation set up. Everything has been built using the ns-3 network simulator. The simulated network is composed by a PPDR Local

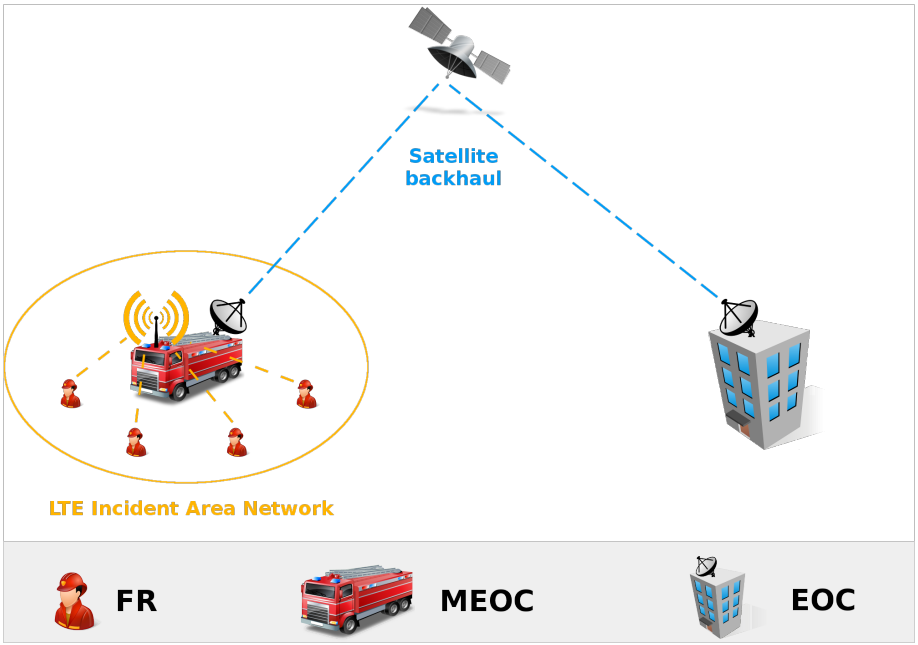


Figure 3.24: Emergency network scenario

Area Network (LAN), which is in turn composed by a gateway that provides a number of client hosts with access to a backhaul link. The latter is modeled as a point to point link with a capacity of 4 Mbit/s and a propagation delay of 350 ms, while each of the links connecting the aforementioned gateway with clients has a capacity of 4 Mbit/s and a propagation delay of 5 ms. As Figure 3.23 shows, client hosts access the backhaul link in order to communicate with some remote nodes; these are organized in a specular fashion with respect to the LAN, but are placed at the opposite edge of the bottleneck channel. TCP Cubic has always been the used protocol for reliable data transfer. The LAN gateway queue size has been set to 350000 bytes, equal to the Bandwidth-Delay Product of the bottleneck link.

As the reference scenario, we selected the one depicted by Figure 3.24. The context is emergency networks, and the scenario has been extracted from the EU FP7 Public Protection and Disaster Relief project. In this environment, there are a certain number of First Responders connected to a MEOC (Mobile Emergency Operations-Control Centre) that brings to them LTE coverage, thus representing a deployable (and mobile) LTE repeater station for the operators. The MEOC connects in turn to the EOC (Emergency Operations-control Centre), which is non-mobile and represents the operations headquarters, via a backhaul satellite

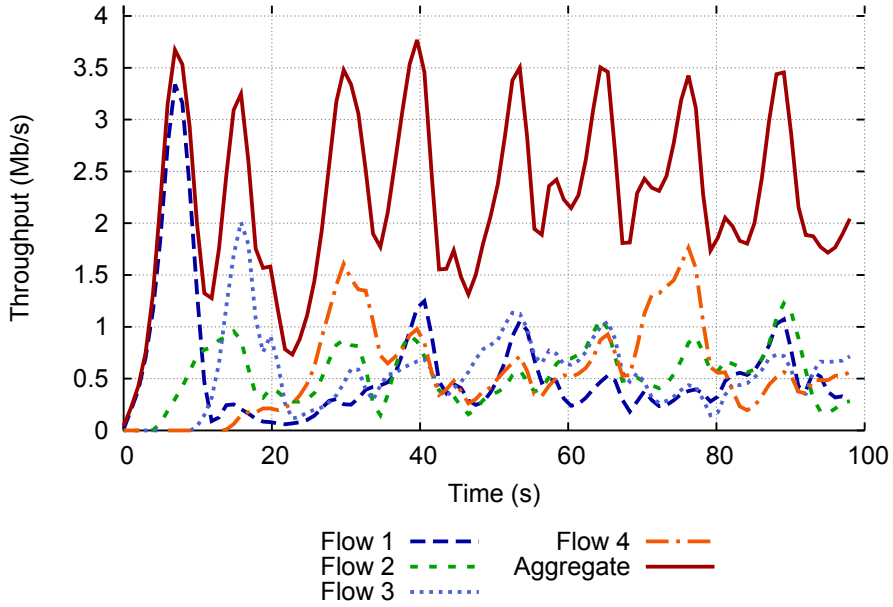


Figure 3.25: Throughput of 4 nodes with CoDel.

link in order to be able to communicate with it independently from its position. The EOC usually acts also as the satellite hub, as more than one MEOC may be present on the field.

3.2.4 Performance

In this section, we present the simulations conduced and the collected results by analysing PINK behaviour with respect to well known AQM techniques. We will evaluate performance like throughput and fairness considering also RTT variations as well as completion time needed to conclude several file transfer.

3.2.4.1 Throughput

With a first set of experiments we analysed the achieved throughput of 4 TCP Cubic flows competing for the same satellite link. In such a simulation, which lasts for 100 seconds, we considered four TCP connection continuously backlogged, in order to isolate the behaviour of the AQM to let the flows converge to a fairness point avoiding oscillation and helping in a fair bandwidth distribution. The four connections started delayed of 5 seconds one each other. In Figure 3.25

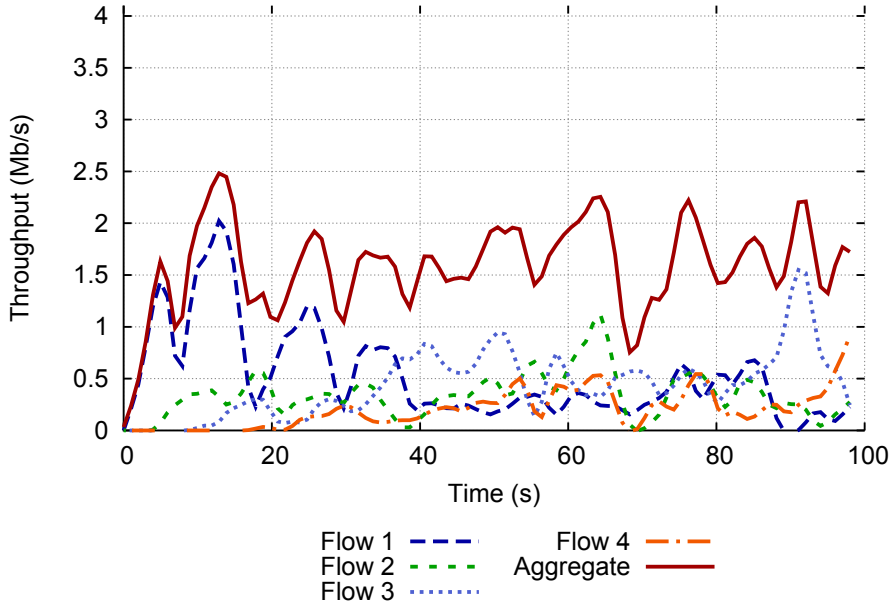


Figure 3.26: Throughput of 4 nodes with RED.

is depicted the CoDel [22] behaviour with respect to this experiment; is it possible to notice how the 4 flows struggling in achieving a balance configuration continuing to fluctuate during time, reaching period in which at least one flow experiences a few kbps of throughput. The RED algorithm looks to suffer more the high RTT environment achieving poor performance in Figure 3.26 where for long periods (e.g. from 40 to 60 seconds of simulated time) all the 4 flows are below an average throughput of 1Mbps, all below the fairness point and resulting in a bad channel exploitation. In both Figure 3.25 and Figure 3.26 is it can be notice how standard well known AQM like CoDel and RED respectively are not able to fully exploit the channel capacity, this because of the high RTT. In Figure 3.27 is it possible to notice how DropTail technique from one side is not able to achieve fairness among competing flows due to the absence of smart queueing policy, but on the other side exploit better the channel capacity with respect to CoDel and RED. Anyway the general poor performance of DropTail can be recognize with drastic transitions from a good channel exploitation to a very poor throughput period on the bottleneck link, this due to the high network congestion caused by the total filling of the queue followed by an high drop period. Situation in different in Figure 3.28 where PINK is in place; with this algorithm, each time a flow join the network an immediate negotiation of band-

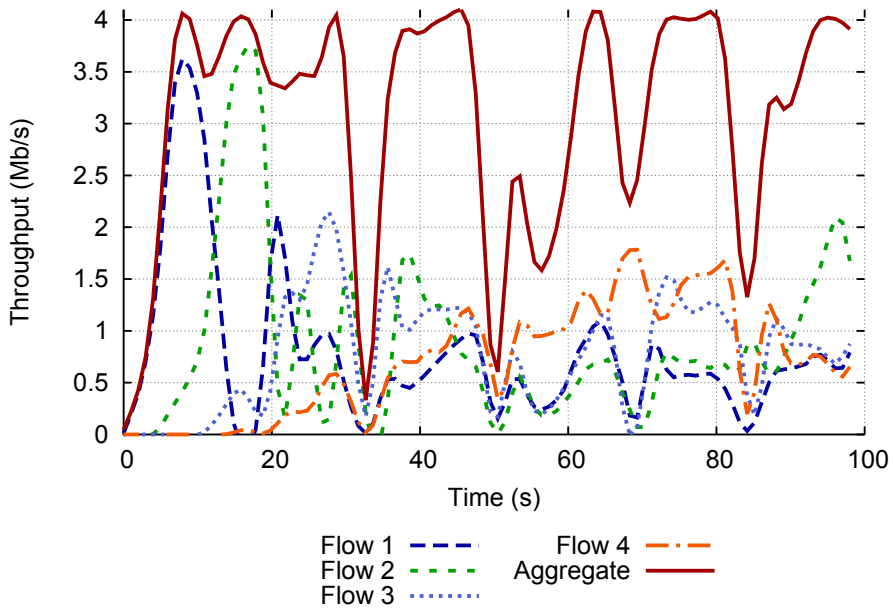


Figure 3.27: Throughput of 4 nodes with DropTail.

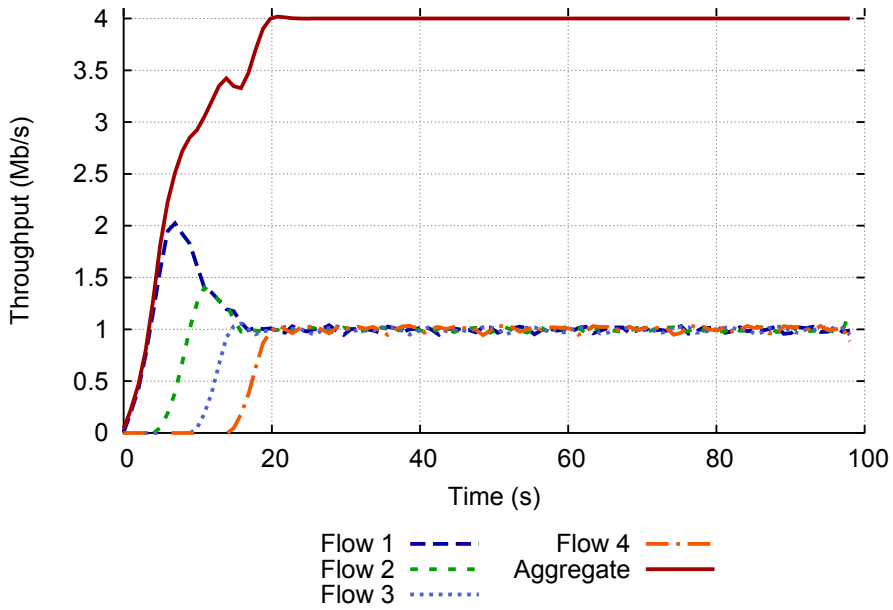


Figure 3.28: Throughput of 4 nodes with PINK.

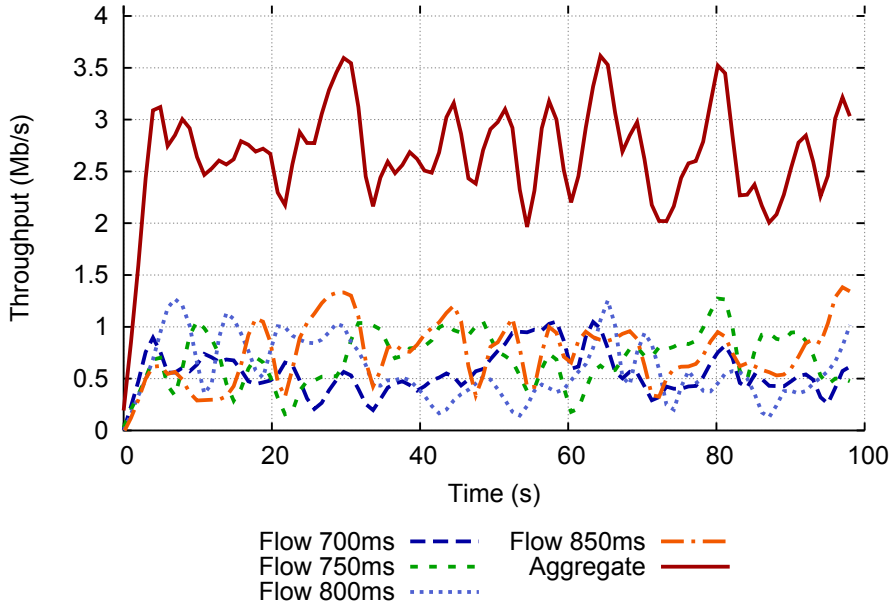


Figure 3.29: Throughput of 4 different RTT flows with CoDel.

width resources is given, in this way each flow achieve immediately the optimal fairness point in a stable way avoiding oscillations. The final result, when all the four nodes are competing for the bottleneck link, is a stable resources subdivision with 1Mbps of throughput per flow. We conduced the same experiment with also RED and DropTail as AQM algorithm over the gateway, anyway the performance are not better than the ones achieved by CoDel.

3.2.4.2 Round Trip Time variations

A common challenging for the TCP is the ability to be RTT independent, i.e. performing well regardless the network delay. With this second set of experiment we considered again 4 TCP Cubic flows, starting together, with different RTTs ranging linearly from 700ms to 850ms and we evaluate also in this case the performance achieved by different AQM. In Figure 3.29 is reported the result obtained by CoDel; it is easy to see that the situation is at least not worst than the one depicted in Figure 3.25, with 4 flows at the same RTT, in terms of throughput oscillation. Even in this case seems hard to achieve a fairness bandwidth negotiation reaching, for each flow, peaks of throughput underutilization of few kbps. The situation is again completely different in Figure 3.30 where

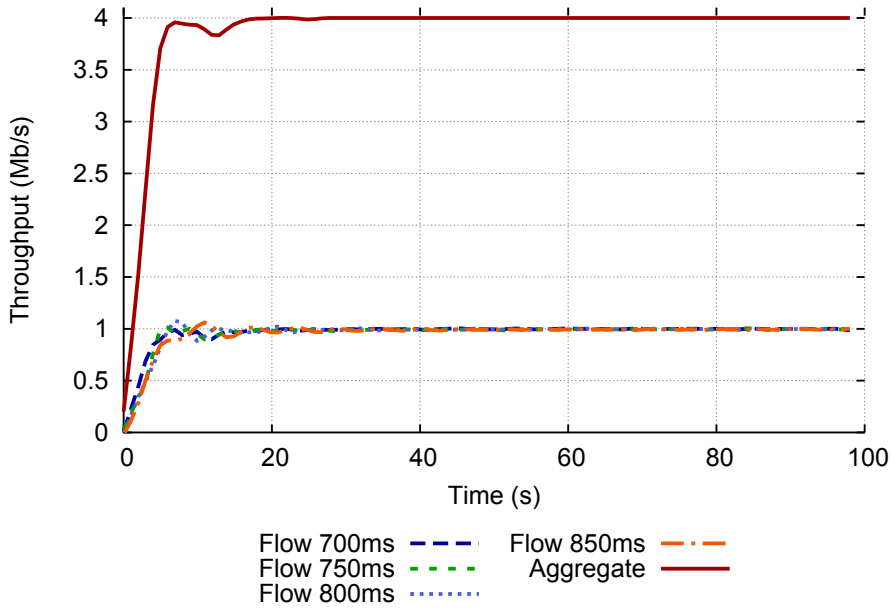


Figure 3.30: Throughput of 4 different RTT flows with PINK.

PINK is used. With PINK in place the fair resource allocation is possible even with different values of RTTs, achieving an optimal balance point in the very beginning of the simulation. This results is possible only because of the RTT tracing system implemented by PINK which helps long higher RTT flows to use wider TCP windows in order to obtain the desired amount of bandwidth.

3.2.4.3 File transfer completion time

In this last set of simulations we tested how different AQM algorithms help, in terms of fairness, to contain the worst case completion time of a variable number of competing flows over the same bottleneck link. We tested from 2 to 32 flows, simultaneously active, with a file transfer of 5MB. Results of this simulation are collected in Figure 3.31 where RED, CoDel, DropTail and PINK are analysed as a function of the competing flows number. In each simulation all the file transfer start together in the beginning of the experiment. From the figure is it possible to notice immediately that RED do not perform well with this experiment resulting for each number of flows considered the worst AQM in terms of transfer completion time. At the same time, there are no impressive difference between DropTail and CoDel, the latter perform a little better for

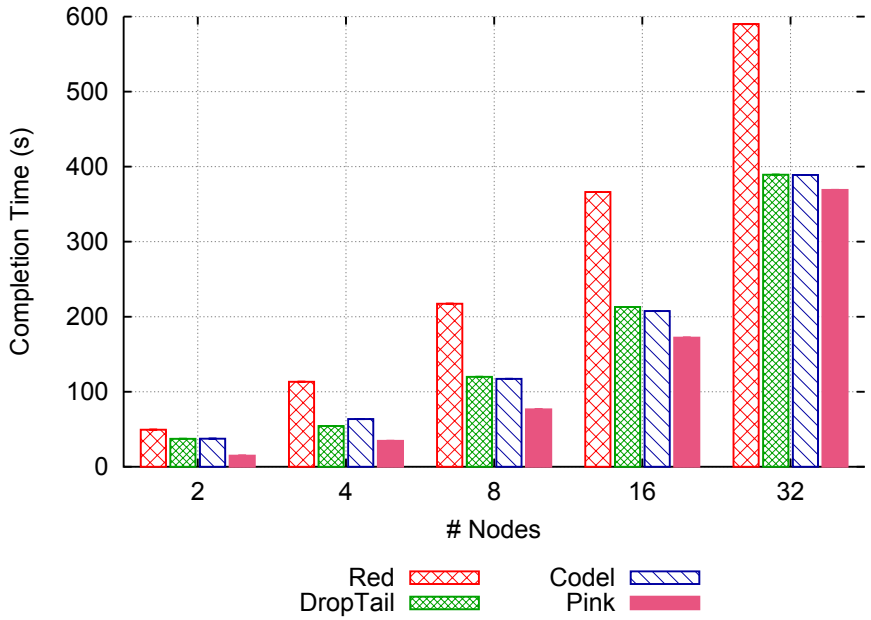


Figure 3.31: Worst case completion time for different AQMs.

all the simulations with more than 4 nodes where the fine-grained CoDel drop policy make the difference. As a matter of fact, the better performance for all the simulations set are reported by PINK which is stably the AQM with the lowest worst case completion time resulting also in a scalable solution regardless the number of flows competing in the network.

3.2.5 Conclusions

In this Section we presented a novel AQM algorithm called PINK (Proactive INjection into acK). This solution has the ability to enhance the TCP congestion control properties in completely transparent way with respect to the end nodes, i.e. PINK works regardless the TCP protocol employed and does not need any modification to the end hosts to work properly but only to be installed on the router nodes. PINK requires only the amount of active TCP flows, the flows RTTs and the channel bandwidth. Throughout simulations with ns-3, we validated the PINK behaviour with respect to the TCP Cubic algorithm enhancing the fairness properties of PINK compared to existing well known AQM like RED and CoDel. Moreover, PINK exhibits a strong fairness property regardless the RTT of each flow making this algorithm particular suitable with high RTT

(satellite links) and different RTT/TCP, a challenging situation which is a characteristic of several PPDR systems. In the following section we will move to a general purpose case study in order to assess PINK performance also for general access networks.

3.3 PINK: A view on General Purpose Network

In recent years, due to the explosion in the number of connected devices, the interest on congestion issues over computer networks has risen as well. In particular, researchers and developers have implemented and tested various proposals, often with the common property of not being limited to a particular layer of the network stack, but instead collaborating over different layers to resolve a particular problem. Moreover, logical layers in networks are not seen anymore as static and closed black-box, but they frequently interact across-the-border.

Active Queue Management (AQM) algorithms and packet schedulers are the most representative frameworks that enforce cross-layer collaboration. For example, if network congestion is detected among TCP flows by a router, its AQM algorithms could decide to selectively drop packets, in order to indirectly signal senders to slow down the transmission rate. Direct signalling (such as ECN, RFC 2481) has also been proposed, but this comes at the cost of updating all nodes across the networks, together with senders and receivers.

The most sensitive transport protocol to network congestion is, without any doubt, Transmission Control Protocol (TCP): it is a stream-based, ordered and reliable protocol, and it must react to congestion to preserve its properties. The downside of recovering algorithms is a performance degradation; its degree depends on the intensity of the congestion.

In this Section we extend PINK (Passive INverse feedbacK), an algorithm that has been designed to run on access gateways as an AQM technique that prevents congestion. It directly signals a TCP sender (without adding non-standard headers or options to packets, without requiring any modification on any other nodes along the path, sender included) the link status and the maximum sending rate. Besides the description of our proposal, we present results that demonstrates how PINK is able to exploit the channel bandwidth, maintaining a very low queuing delay, and imposing fairness among different flows. Furthermore, these results are achieved without forcing a single packet drop.

The discussion is organized as follows: Section 3.3.1 summarizes related work. Section 3.3.2 describes the extended PINK algorithm. Section 3.3.3 presents a mathematical analysis of the gateway queue occupancy upper bound when employing our proposal. Section 3.3.4 introduces our validation scenario, while Section 3.3.5 shows the simulations results. In Section 3.3.6 the conclusions are drawn.

3.3.1 AQMs of the Literature

In this section we carefully revise the literature involving feedback-based congestion control algorithms by extending the work list reported in Section 3.2.1.

In a very recent work presented at INFOCOM [93] the performance of a self-tuned variant of RED, called Adaptive RED (ARED) has been analysed and compared with CoDel on a general access network environment. ARED, differently from RED, does not work by forcing, as a configuration parameters, different thresholds at which change the drop probability; ARED instead works by referring to a target delay to be considered in order to decide if enqueue or not a certain packet. In this way the algorithm itself calculate the thresholds as a function of RTT and does not need user tuning which has been the main limitation of RED diffusion.

An AQM algorithm called GREEN [78] is not so far from our proposal due to its rate-based approach an worth to be analysed; GREEN operates giving each packet a drop probability that depends on the flow RTT, on the number of active flows, and on the Maximum Segment Size (MSS) encountered during a sampling time. GREEN also exhibits some drawbacks: (i) it does not have a reference implementation; (ii) it infers the MSS by considering the highest value only, which results in a coarse-grained drop policy; (iii) the RTT measurement assumes the usage of TCP options (i.e. it leaves to end nodes the responsibility of providing RTT values to the gateway). GREEN is therefore a non-transparent solution for end nodes too.

To address these issues on satellite-based emergency networks that employ high delay links we presented in [94] and described in Section 3.2 a preliminary work on PINK. We here extend the proposal, by also cross-comparing our contribution with the aforementioned algorithms, in order to validate it and provide an answer to the drawbacks that are being exposed in this Section. We based our comparison testbed on recent works such as [95] and [93], which are also based on AQM techniques. The interesting result of the latter work is that, in several instances, ARED obtains better results. In [96] it is shown instead that while AQM algorithms are able to significantly improve performance on the long run they also exacerbate TCP flows unfairness, especially among TCP flows with different RTTs, and may lead to large latency spikes caused by queuing delays when flows startup.

3.3.2 The Extended PINK algorithm

In this section, the PINK algorithm is described through both a theoretical explanation and implementation directives.

PINK is a per-flow yet stateless AQM algorithm, designed to be deployed on a gateway that provides network access to a set of client hosts. It exploits the flow control of TCP, and in particular the one of the receiver, through the well-known field "Window size" (RCV.WND) of the TCP header. This field represents the quantity of bytes that the destination is currently willing to receive, and an RFC-compliant sender will not exceed this value when transmitting segments, choosing as its window size the minimum value between congestion and receiver window.

Basically, PINK computes (for each ACK packet) the value for the advertised RCV.WND, which will be then written on the header. The key concept is that this value is calculated so as to prevent bottleneck congestion, tricking the sender.

Mathematically, it holds that:

$$RCV.WND_i = B_i = \left\lfloor \frac{BW \cdot RTT_i^{min} \cdot c}{n} \right\rfloor \tag{3.6}$$

Considering that $\frac{BW}{n}$ is the bandwidth allocation given to each active flow i, in Equation 3.6 BW is the bottleneck bandwidth, n is the current number of active flows, RTT_i^{min} is the minimum RTT calculated by PINK for the flow i while $RCV.WND_i$ is the new RCV.WND value to write in the TCP window field, equal to B_i that represents the burst value of flow i. The constant parameter c, called the "exploitation parameter", ranges from 0 to 1 and represents the channel exploitation factor. Is has been introduced to allow the network designer to tune the amount of bandwidth shared among nodes, in order to compensate packet overhead for headers introduced by network layers. Indeed, when $c = 1$ the entire bandwidth is considered to be used by TCP for the application data, leaving no space for TCP, IP, and link layer headers. This exceeding amount leads to the use of some queue space (that therefore generates a low degree of congestion) that can be avoided by using $c < 1$. The bottleneck bandwidth is assumed to be an offline and always available information; the number of active connections, instead, is actively tracked by counting the number of SYN, FIN

and RST packets that are received or sent. It is likewise possible to efficiently calculate the RTT of each flow [88–90].

PINK operates per-flow since it divides the bottleneck bandwidth over the number of active flows on the channel. We remind to Figure 3.22 (in the previous section) for an operational scheme of PINK, where *pinkwnd* represents the calculated RCV.WND value.

As a matter of fact, TCP connections are bi-directional, because data could be exchanged both ways (upload and download). In concrete terms, there are two information flows and two ACK flows, but for the sake of brevity we only investigate the uplink direction, applying PINK on the ACKs sent by remote receiver nodes. The setup could be flawlessly extended to prevent congestion on both directions, by applying PINK on the ACKs sent by the sender too.

As PINK is designed for access networks, a possible constraint to its deployment may be the performance overhead introduced by the need of recomputing the TCP Checksum after the alteration of the TCP advertised Receive Window, an operation that must be done for each and only ACK that needs to be changed. However, in Gigabit-level access networks, this performance degradation can be avoided by employing equipment that compute hash functions through specialized hardware [92], instead of performing these operations in software as it is feasible in lower-bandwidth access systems.

3.3.3 PINK guarantees

In this section we analyse the impact of PINK on the gateway queue level, considering in particular the worst-case upper bound value of queue occupancy.

Theorem 5. *Let $Q_{gw}(t)$ be the output queue length of the gateway at the generic time t and let RTT_{net} be the average round trip time of the network. If the link bandwidth is constant and equal to BW, with PINK as AQM algorithm the following inequality holds for any time t:*

$$Q_{gw}(t) \leq BW \cdot RTT_{net}. \tag{3.7}$$

Proof. Consider a time interval with a constant number of n active TCP flows. Consider now a generic i-th TCP flow; it will send in the network a burst B_i which is at most equal to the product $BW_i \cdot RTT_i^{min}$ each RTT_i, as showed in Equation 3.6 with $c = 1$ for a worst-case analysis. Consider the case in which all flows send their bursts simultaneously, at the same time t. The gateway queue, immediately before t, can be either empty or containing packets; this divides the proof in two cases.

Case 1: Empty queue. The gateway queue is filled with the aforementioned bursts and it follows that:

$$\begin{aligned} Q_{gw}(t) &= \sum_i B_i \\ &= \sum_i BW_i \cdot RTT_i^{min}. \end{aligned} \tag{3.8}$$

Considering RTT_i^{min} upper bounded and equal for each flow to RTT_{net}, we can write:

$$\begin{aligned} Q_{gw}(t) &= \sum_i BW_i \cdot RTT_{net} \\ &= (\sum_i BW_i) \cdot RTT_{net}. \end{aligned} \tag{3.9}$$

Considering Equation 3.6 we can easily write its fair bandwidth allocation rule in the form $BW \geq \sum_i BW_i$, and by substituting it to (3.9) we get the thesis.

Case 2: Non-empty queue. Let the gateway queue, immediately before t, contain at least one packet p. After receiving the aforementioned bursts it follows that:

$$\begin{aligned} Q_{gw}(t) &= p + \sum_i BW_i \cdot RTT_i^{min} \\ &> \sum_i BW_i \cdot RTT_i^{min} \\ &= BW \cdot RTT_{net}. \end{aligned} \tag{3.10}$$

This contradicts our assumptions. The packet p must belong to one of the active TCP flows (remember that PINK operates on an exclusive TCP queue). Consider also that the packet p belongs to the i-th flow. If follows that, at time t, the gateway queue would contain an amount of packets belonging to the flow

i which is:

$$\begin{aligned} p + BW_i \cdot RTT_i^{min} &> BW_i \cdot RTT_i^{min} \\ &= B_i. \end{aligned} \tag{3.11}$$

The amount of packets in the queue, belonging to the flow i, would be greater than the product $BW_i \cdot RTT_i^{min}$, which is $RCV.WND_i$; this is absurd because it violates the TCP property.

To conclude, by analysing both case 1 and case 2 we completed the proof. Theorem 5 is proved. □

This way it is possible to have an upper bound guarantee about the queue length and, consequently, an upper bound on the maximum queuing delay introduced by the gateway. A key point of Theorem 5 is that this deterministic bound is equal to the standard suggested buffer size [19], which makes PINK easy to deploy, while this upper bound in the queue occupancy level prove the effectiveness of the PINK algorithm and its drop-free behaviour.

From an experimental point of view, Theorem 5 manifests a stable behaviour even in highly dynamic environments. In fact, when a new flow starts to transmit, a SYN packet is sent through the network and captured by PINK; the algorithm increases the number of active flows and starts to update the RCV.WND of each ACK according to Equation 3.6, scaling down the bandwidth of each flow in order to accommodate the new one. Conversely, when a flow stops to transmit, a FIN packet is sent through the network and captured by PINK; this time, the algorithm decreases the number of active flows and increases the RCV.WND of each ACK according to Equation 3.6, increasing the bandwidth of each active flow in order to continue to efficiently exploit the bottleneck link capacity.

The only shortcoming of this description is represented by SYN packets, that are not contemplated by Theorem 5. These packets, in fact, belong to flows that are not currently part of the proof. This highlights the main current weakness of PINK: it is not resilient to Denial of Service attacks. If a group of malicious flows would start to flood the network with SYN packets, the current implementation of PINK is not able to face this attack, as it will drastically reduce the RCV.WND of each flow.

Model	ns3-2.24
Access/Remote Network	Ethernet 100 Mbit/s
Bottleneck Network	Ethernet 10 Mbit/s
Base RTT	100 ms
MTU	1500 Bytes
MSS	1000 Bytes
Active Clients	2, 4, 8, 16, 32, 64
AQM	DropTail, ARED, CoDel, GREEN, PINK

Table 3.2: Experimental network setup: Parameters underlined are taken by [93]

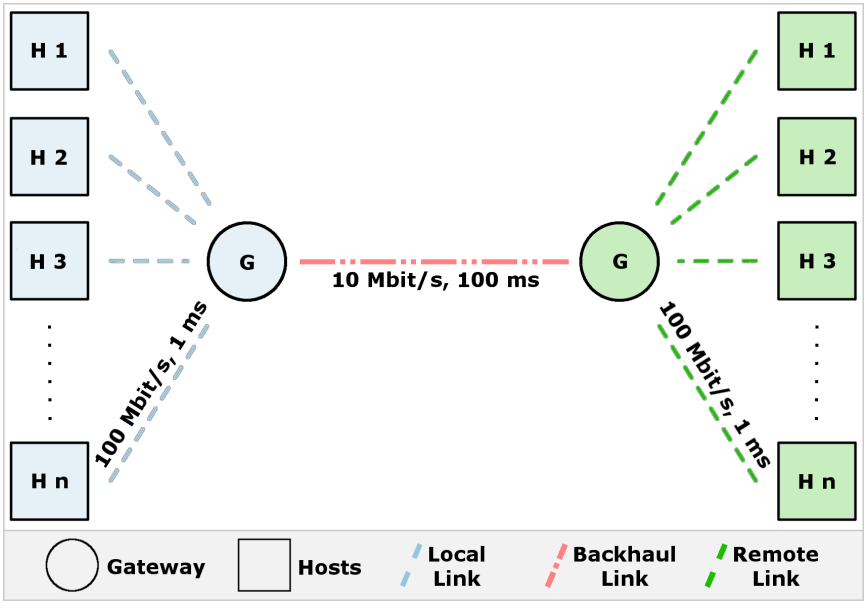


Figure 3.32: Network testbed topology.

3.3.4 Simulation Environment

In order to demonstrate the effectiveness of PINK, we developed a testbed compliant to the one used in [93]. As simulation platform, we used the ns-3 network simulator; PINK source code, with instructions to reproduce our results, is available at [97]. The parameters used in our simulations are summarized in Table 3.2; they basically represent two distinct networks, connected with a bottleneck link that is accessible to the end hosts of each network through a border gateway as depicted in Figure 3.32.

We decided to compare PINK with four Queue Manager algorithms, listed as follows together with a brief explanation of the rationale behind each choice.

- DropTail, as it can be used as a benchmark to represent the default solution when no AQM technique is in place.
- CoDel, as it can be considered the state of the art.
- ARED, due to the performance provided in [93].
- GREEN, due to its rate-based approach that is shared from PINK.

We ran various experiments consisting in a set of backlogged TCP flows, created between pairs of end hosts in which sender and receiver belong to different networks. Common parameters for the experiments are the bottleneck maximum queue size, set equal to the network Bandwidth Delay Product of the bottleneck link (125 KB), and their duration, set to 300 seconds. We describe comparisons and outcomes, along with specific experiment parameters, in the next Section.

3.3.5 Performance

In this section we analyse the performance of PINK, comparing its results with The next subsections present the results on goodput, an analysis on per-packet RTT distributions, a flow fairness evaluation and a simple drop analysis.

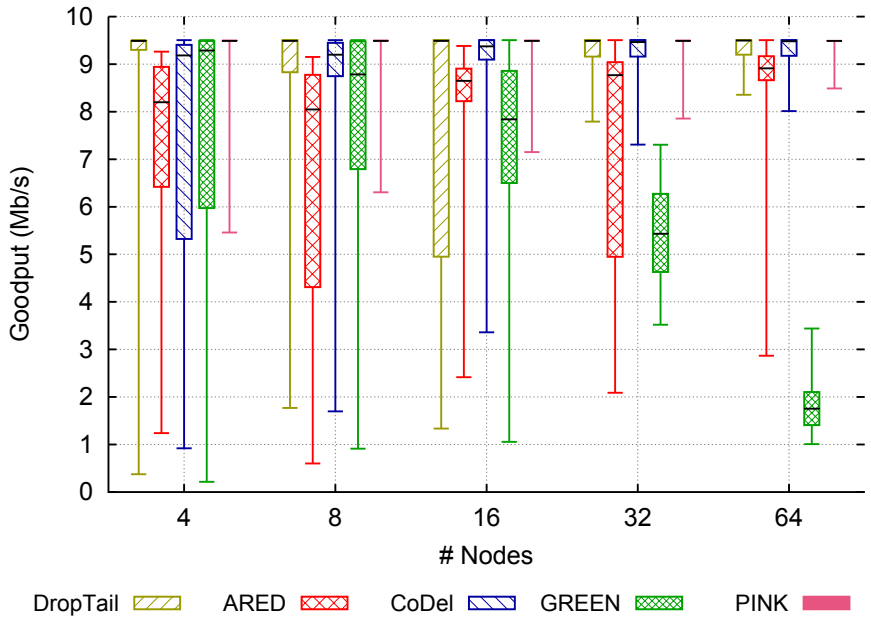


Figure 3.33: Application goodput at the bottleneck link (per 1-sec intervals). Different numbers of active nodes with RTT_{base} = 100ms; bottom and top of whisker-box plots show the 10th and the 90th percentiles, respectively.

3.3.5.1 Goodput

Figure 3.33 shows the achieved goodput of DropTail, ARED, CoDel, GREEN and PINK for different levels of congestion at the bottleneck link. Higher values denote higher performance. The congestion degree is proportional to the number of the active nodes, which varies from 4 up to 64. All the AQMs under test are configured with their default parameters (e.g. *target_delay* of 5ms for CoDel). By looking at the Figure, as first remark we claim that the results are compliant to [93], recording a minor difference on the goodput exploitation of CoDel that is slightly better with respect to ARED. The latter, in fact, suffers a bit, especially in congested scenarios. Considering instead "classical" algorithms, CoDel behaves very well, achieving a goodput level comparable to those of DropTail which is the benchmark for this figure of merit. Furthermore, the 10th and the 90th percentiles of CoDel goodput tend to approach each other as the number of nodes increases.

On the contrary, the performance of GREEN are very poor. With the values of bandwidth and RTT_{base} (see 3.2) adopted in the experiments, the drop

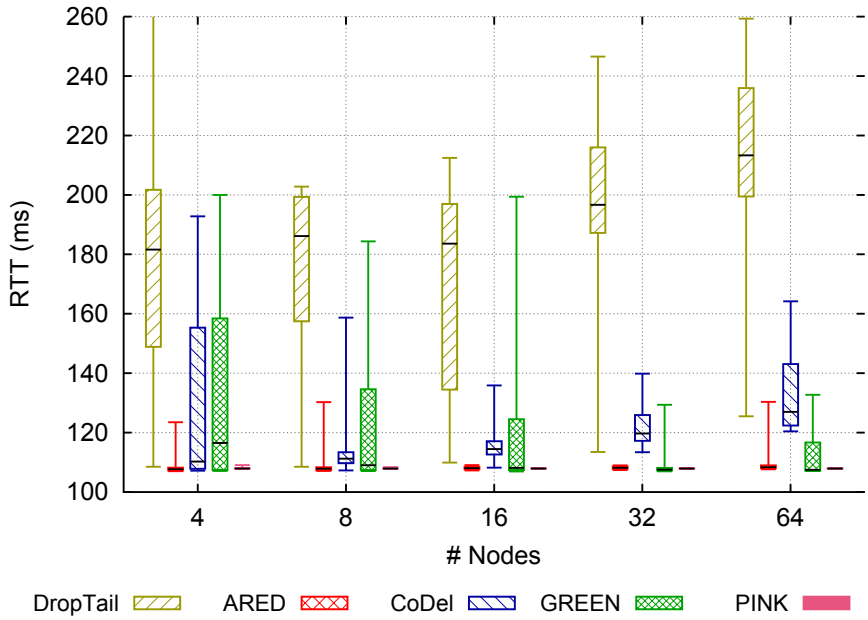


Figure 3.34: Per-packet RTT. Different numbers of active nodes with RTT_{base} = 100 ms; bottom and top of whisker-box plots shows the 10th and the 90th percentiles, respectively.

probability (which is also related to the amount of active nodes[4]) increases too much, resulting in a goodput degradation.

Results even higher than DropTail and CoDel, and considerably better than GREEN, are those achieved by PINK. With the latter, in fact, the goodput is extremely stable with the 10th and the 90th percentile almost identical and close to the optimal value[5], with the bottom of the candlestick that increases as a function of the congestion level.

It is clearly undesirable to have a higher goodput if it comes at a cost of an increased application delay: in the following, we investigate the delay perceived at the application level (through an RTT analysis) to see if PINK increases the RTT of the flows.

[4]See Equation 2 of [78] for further details.

[5]With a MSS of 1000 bytes, the frames in this simulations have the size of 1052 bytes. This upper-bounds the goodput at 9.5Mbit/s, which is lower than the upper bound of [93]. We have been unable to align to their goodput value since their MSS value is not disclosed.

3.3.5.2 RTT analysis

Figure 3.34 shows the per-packet RTT distributions of AQMs for different congestion levels, where lower values (and lower variances) indicate better performance. As a side note, in this test as well our CoDel and ARED results are compliant with [93]. With regards to queuing delay, DropTail represents a negative benchmark of what happens on a bottleneck link when no AQM techniques are employed. The RTT_{base} derived from the sole propagation delay of this channel is 100ms, but by considering in addition the transmission time and the delay inside LAN networks, the smallest RTT value that has been registered during the simulations is equal to 107ms. On the Figure, the difference between the reported value and the smallest value of the candlesticks (107ms) corresponds to the queuing delay at the bottleneck.

From this experiment, some observations can be made: CoDel's median, as well as the 10th and the 90th percentiles of queuing delay, increase proportionally to the network congestion level, especially when the number of nodes is above 4. If ARED was suffering the comparison with CoDel from a goodput point of view, here ARED outperforms CoDel, obtaining an almost stable queuing delay; this is especially true for the median, with the values of the 10th and the 90th percentile that are close to the lower bound.

Likewise, GREEN performs better with regards to the queuing delay performance metric. In fact, it provides a trend which is the CoDel opposite; the median, 10th and 90th percentiles decrease as a function of network congestion.

The main result of this Figure, however, is the performance of PINK. Firstly, it is extremely stable, with all the candlesticks collapsed and close to the lower bound value. Coupled with a higher goodput, this indicates that PINK is allowing the exploitation of the bottleneck channel by TCP flows without introducing any unwanted delays.

The next question to investigate is the following: is PINK able to serve equally all flows, or some of them unfairly hold more resources than others? And if that is the case, what is the proportion of the unfair bandwidth distribution?

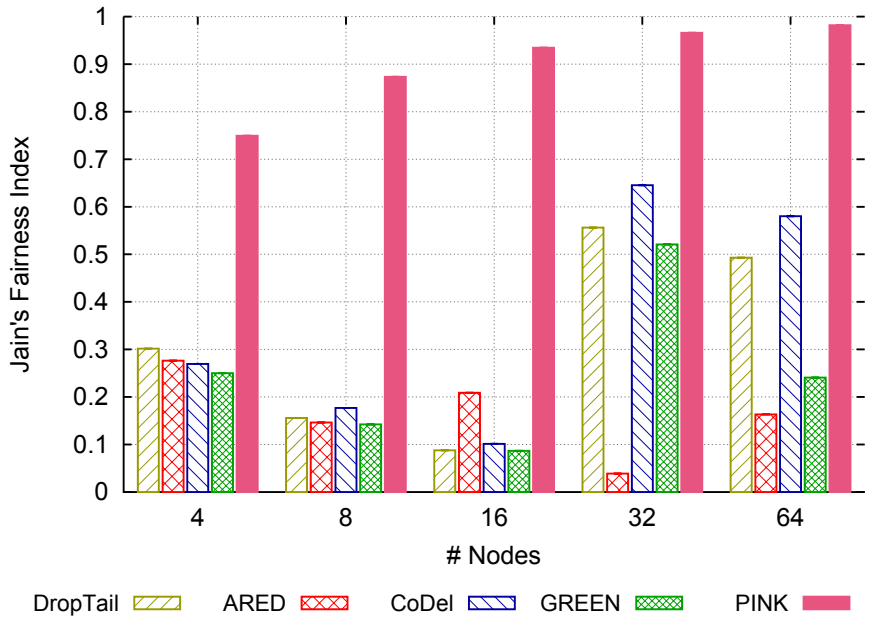


Figure 3.35: Worst-case Jain's Fairness Index calculated per 1-sec intervals, varying the active nodes number.

3.3.5.3 Fairness

In order to evaluate the fairness between different TCP flows when coupled with different AQM algorithms, we considered the Jain's Fairness Index. Assuming equal data rate requirements among flows, the instantaneous Jain's Fairness index JFi is defined in terms of the instantaneous data rate R_i as:

$$JFi(R_1, R_2, \ldots, R_n) = \frac{(\sum_{i=1}^{n} R_i)^2}{n \cdot \sum_{i=1}^{n} {R_i}^2}$$

where n is the number of active flows, R_i is the instantaneous data rate of flow i and JFi is a real number in the interval $[\frac{1}{n}, 1]$ with a maximum best-case value of 1, if the achieved rate is equal for all flows, and a minimum worst-case value of $\frac{1}{n}$ if only one aggressive flow is filling the entire data rate of the system.

We reported in Figure 3.35 (higher values indicates better performance) the worst-case JFi calculated during each simulation, with the different AQM techniques and varying the network congestion level. At a first look, it emerges that ARED is particularly unfair, without exhibiting a clear variation pattern as a function of the congestion level. On the contrary, DropTail, GREEN, and

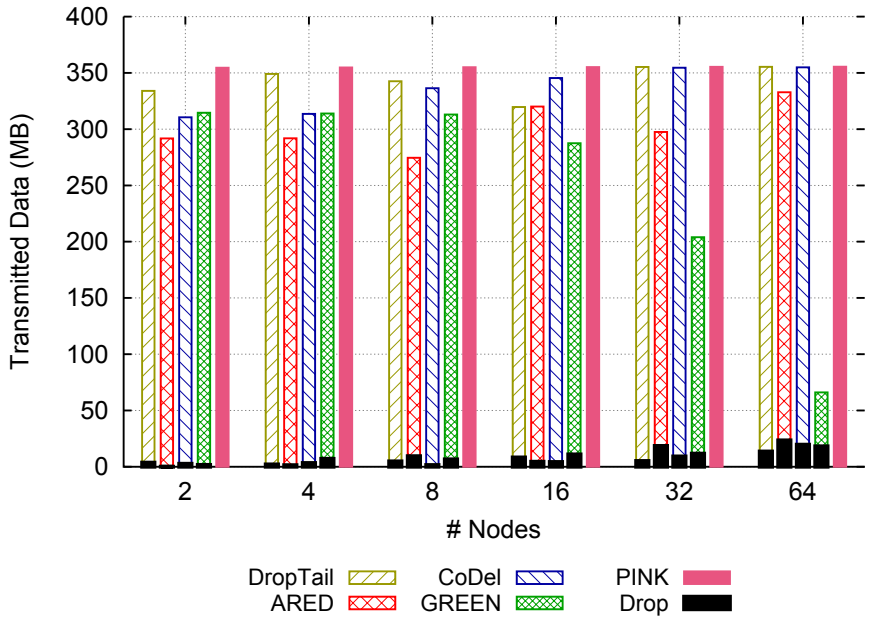


Figure 3.36: Transmitted data at the end of the experiment on the bottleneck link overlapped by the dropped data.

CoDel suffer to maintain a fairness balance with low congestion levels, while they increase the performance when moving towards a more congested environment (e.g. with 32 or 64 active flows). Remarkable results are obtained by PINK: in fact, it presents an excellent scalable behavior, approaching the best-case fairness value of 1 (which means that all flows have the same rate) as the network congestion level increases.

3.3.5.4 Drop analysis

Last but not least, we analysed the AQMs behaviour in terms of packet drops. This test has been performed in order to analyse the waste of energy and processing time for managing packets that, eventually, will be lost. Indeed, dropped packets at the gateway have been transmitted over the LAN (consuming resources) and processed by the gateway itself. Figure 3.36 shows the amount of successfully transmitted data at the end of experiment over the bottleneck link, paired with the amount of dropped data by the bottleneck AQM. The upper-bound of the successfully transmitted data at transport layer, considering a MSS

of 1000 Byte, a gateway frame of 1052 Byte, a bottleneck link of 10 Mbit/s and a simulation of 300 seconds is equal to 356 MB as result of $\frac{10 \cdot 10^6}{8} \cdot 300 \cdot \frac{1000}{1052}$.

The ratio between dropped and transmitted data is almost the same for all the AQMs, except for GREEN that has a lower transmitted data value (it is a consequence of a lower goodput). The amount of drops grows as a function of network congestion and approaches the 10% of goodput, but not for the flows that employ PINK, that indirectly avoids packet drops (as explained in Section 3.3.2). This trend poses PINK in a new position with respect to other AQM, with an emphasis on efficiency, thanks to the fact that it exploits the channel data rate without dropping packets and wasting energy.

3.3.6 Conclusions

In this Section we extended PINK, a queue management algorithm designed for access networks that prevents congestion, in a transparent way, by imposing a maximum upper bound on the data rates for each client. This is done through updating the RCV.WND value in the acknowledgement segments, allowing PINK to supervise bandwidth utilization, in order to efficiently exploit a bottleneck channel capacity, forcing fairness among different flows, and at the same time maintaining a low queue occupancy on the gateway. Furthermore, these results are achieved without discarding a single packet. Another important aspect to remark is the algorithm scalability, as all the figures of merit that have been analysed reveal that PINK manifests a positive, or at least a constant, trend as a function of the network congestion level.

Chapter 4

Packet Scheduling

In this Chapter the packet scheduling problem is deeply studied as an important solution for providing QoS that decides the order in which packets are sent over a link. The scheduling of packets is a challenging topic for emergency networks due to the massive use of wireless technologies (e.g. WiFi, LTE, 4G/5G) with which to provide high QoS guarantees. Unfortunately, it is difficult to compare different solutions and actually to test them in order to select the most proper packet scheduler for each particular environment. In this final Chapter we present a novel modular architecture to ease all of these tasks. Together with the architecture, we also define a novel packet scheduler and TEMPEST, a new Test EnvironMent for Performance Evaluation of the Scheduling of packeTs, which is a novel tool able to help the research in the packet scheduling field. TEMPEST is able to measure the actual performance of a packet scheduler in several environments, both wired and wireless, and some figures of merit, like the execution time, QoS metrics and throughput, giving prompt feedback about the quality of the solution studied. It can also measure the performance of the proposed modular architecture as well. The goal of this final Section is to present in detail the current design of the modular architecture, the benefits introduced by the adoption of the novel packet scheduler and the features of TEMPEST.

4.1 The Modular Architecture

V2V systems are playing a critical role in ICT world, because for their nature they are considered an important solution for next generation communication environments: they also are a protagonist solution for the Internet of Things (IoT) world. Among such communication systems, a particular challenging one, suitable for V2V communications, are the Public Protection and Disaster Relief (PPDR) systems. We are currently investigating V2V networks as emergency network solutions in the EU FP7 Project "Public Protection and Disaster Relief - Transformation Centre". In such environments, all the constraints that we use to consider as challenging are present:

- Vast applications range;
- Broad use of heterogeneous wireless technologies;
- High mobility;
- Though QoS guarantees provisioning.

These difficulties can be faced from a high level architectural point of view like in [63, 98] or from a software solution point of view like this section case.

In particular, here we focus on the QoS provisioning in V2V networks. From the theory, we know that the best way to provide QoS guarantees is to use a packet scheduler and in the last decades several high performance packet schedulers for wired links have been provided [99–101]. The problem is that moving from a wired environment to a wireless one is not trivial.

Defining a packet scheduler for a general V2V environment is not an easy task and typical solutions address the problems of *only* a specific technology, like [102]; state-of-the-art solutions for QoS provisioning over wireless links are based on cross-layering packet schedulers that deal both with the QoS guarantees and the wireless link issues. Unfortunately, such an approach is not flexible and requires a technology-dependent approach, with the definition of a different software version for each of the deployed technologies, such as [103] and [104] for WiMAX and LTE, respectively.

Summarizing, main open issues are:

- How to recycle and reinvest the decade of research done over wired packet schedulers?
- How to let coexist the current high performance solutions for QoS provisioning with the issues related to the wireless links?
- How to solve such problems with a flexible approach in order to be able to move transparently from a technology to another?

We address these issues through a modular architecture which permits the use of existing high-performance packet schedulers for wired links over generic wireless technologies, as they are, and at the same time allows for the flexibility to adapt to different channel conditions. The architecture proposed is not simply "theory" but it has been concretely used in the context of the PPDR-TC project to produce HFS [91], the first packet scheduler of this novel modular scheduling family suitable for PPDR, WiFi based, V2V systems. With this article we provide some initial results, and identify new perspectives, which pave the way for efficient QoS provisioning and throughput boosting in V2V systems.

4.1.1 A General PPDR Case Study

Together with the FP7 Project "Public Protection and Disaster Relief - Transformation Centre" several applications, technologies and software solutions for PPDR future systems have been studied. A key role in such a system is played by V2V communications, since in post disaster event the existing infrastructure technology which brings internet coverage on the field may be disrupted or damaged. This case is faced by the use of specific emergency vehicles called Mobile Emergency Operative Center (MEOC) [105], that provide connectivity for the First Responders on the Incident Area Network. A general view of such an event is depicted in Figure 4.1, in which several MEOCs connect to each other through the X2 interface of LTE, in order to compensate the absence of internet coverage provided by the core network infrastructure disrupted by the disaster. External nodes such as the Headquarters of the emergency team can then be reached through satellite links, deployed by at least some of the MEOCs. Therefore, each MEOC can extend the coverage to the personnel on the field by adopting a specific technology, as reported in Figure 4.2. This Figure is particularly meaningful because it basically represents the core of such a system; several MEOCs might

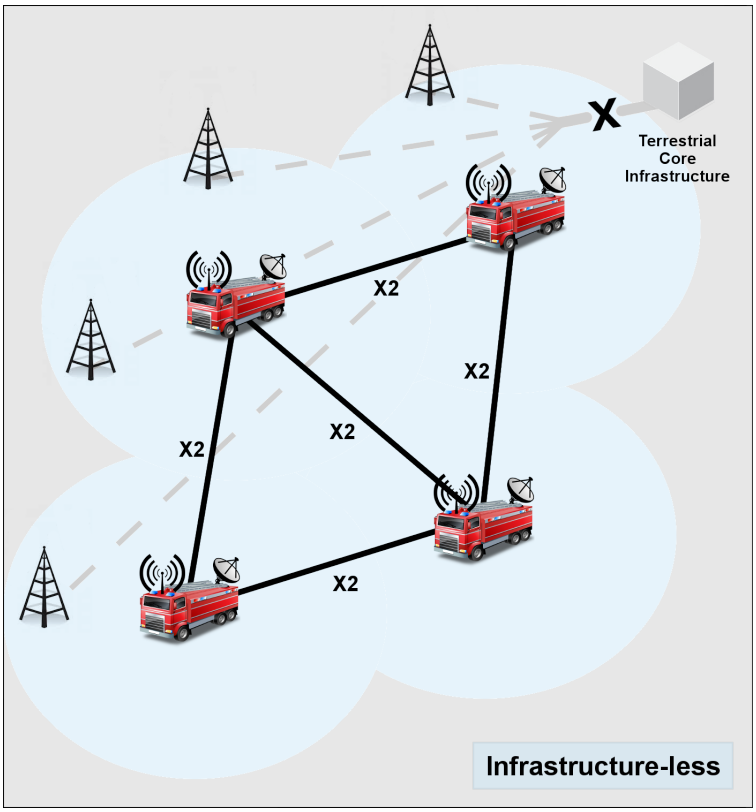


Figure 4.1: V2V mesh infrastructure-less topology.

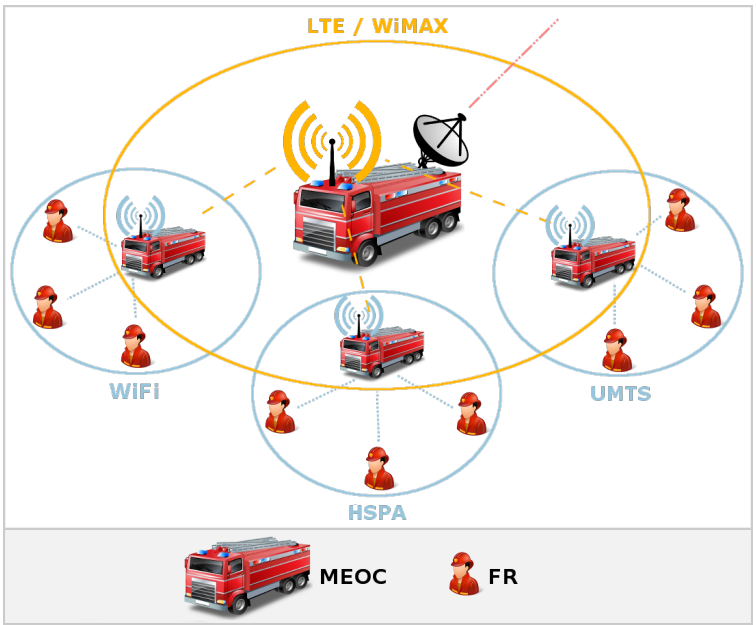


Figure 4.2: Emergency network hierarchical star topology.

belong to different PPDR subsystems (fire brigades, air-forces, border brigade, police and so on) and each vehicle could therefore employ specific yet proprietary technology, making hard the adoption of a general software solution that could be adopted by a general MEOC in order to optimize some network functions like throughput or QoS requirements.

4.1.2 The Modular Architecture

This section aims to describe the proposed modular solution for the definition of a new family of packet schedulers but also aims to help non familiar readers to understanding how a packet scheduler generally works. Figure 4.3 shows a logical sketch of the modular architecture. It is composed by a cascade of two classical packet scheduling schema divided on two layers, the *QoS Provisioning Layer*, hereafter called just QoS, and a *MAC Scheduling&Abstraction Layer*, hereafter abbreviated as MAC-SAL. There is also a software module called *packet prefetcher* which implements the logic that rules the movements of packets from the above layer to the below one. Both the layers and the packet prefetcher will be described and commented in their characteristics.

4.1.2.1 QoS layer

The QoS layer is represented in the top box of Figure 4.3. It presents the typical organization of a packet scheduler as we used to see on a wired environment (e.g. the scheduler used by our laptop when it is connected to a router through an ethernet cable). The name *QoS layer* is simply because of the purpose of a packet scheduler, i.e. to decide the order the packets are sent on a output link with the goal to provide QoS. The general components implemented by almost all the classical wired packet schedulers defined in the last decades of research are depicted: the packet classifier and the scheduler algorithm itself. The packet classifier is the software component that decides the *importance* of a packet. It decides the QoS guarantee level that needs to be associated to the packets itself, depending on the type of service, and then it places the packet in the corresponding queue. For example, UDP packets have different QoS requirements compared to TCP packets and they will then belong to different queues in almost all the existing packet schedulers. A typical classifier uses the Type of Service (ToS) field of the packet and/or the segment service port for

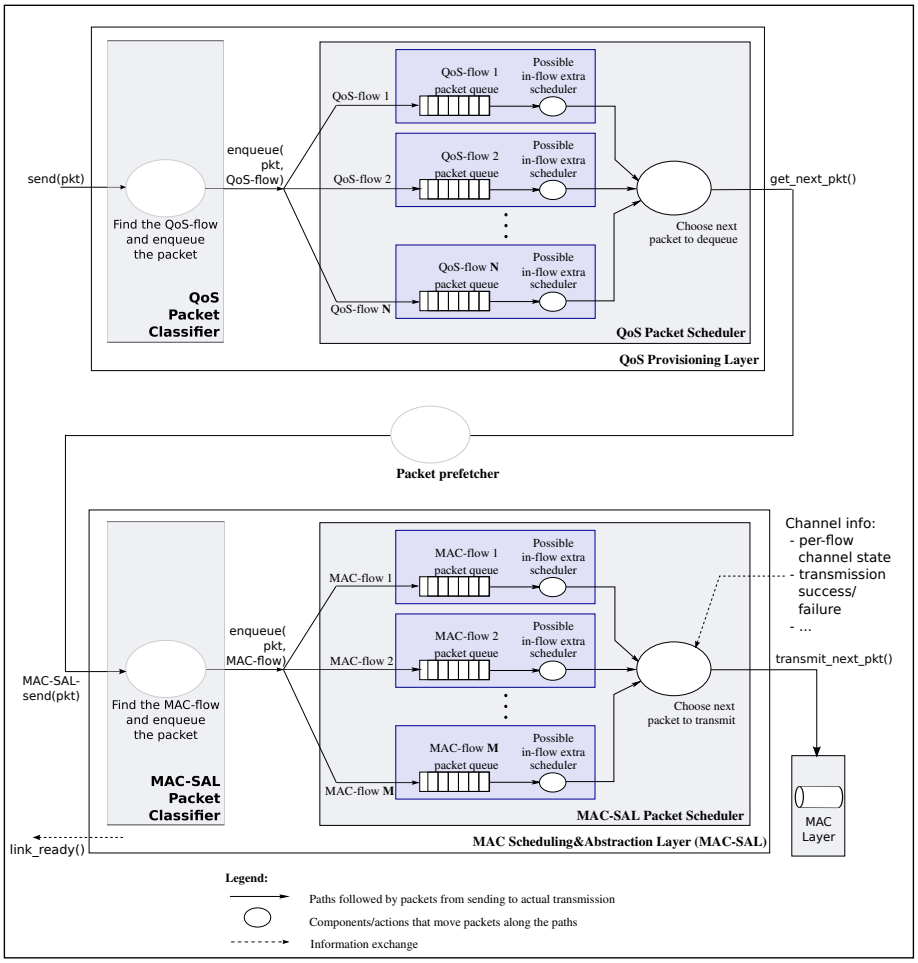


Figure 4.3: Modular architecture for providing QoS over a wireless link.

fine-grained classification of each service. After the classification, we have the scheduling algorithm which simply answers to the following question:"which is the next packet to transmit?". To do this, each scheduling algorithm follows its own strategy.

For what concern the existing packet scheduling architecture for wired links, the logic description stops here. In fact, what happens in a general wired environment is that the QoS box depicted in Figure 4.3 is directly connected to the outgoing link, nothing more.

4.1.2.2 And the cross-layering solutions?

Understanding the relationship between the cross-layering scheduling solution and the modular architecture depicted in Figure 4.3 is simple. A cross-layering solution operates with only the top box, the QoS one, and it is connected directly to the outgoing link like a standard wired scheduling solution. The difference is that a cross-layering scheduler tries to improve scheduling decisions by the use of Channel State Information at the QoS layer, i.e. using information coming from the MAC layer below to improve the scheduling benefits. This technique has been used, and it is currently under use, even if it poses some limitations:

- Taking an existing wired scheduler from the literature and "converting" it in a cross-layering scheduler is not something that happens for free. The formal model associated to an existing packet scheduler is altered by the presence of CSI used in the scheduling decisions, i.e. the QoS guarantees bounds of the scheduler must be recomputed, if feasible.
- Moving an existing cross-layering scheduler from a technology to another is almost impossible, because the information captured by CSI are different, like the idiosyncrasies of the new channel technology.
- Modifying the same cross-layering scheduler to operate in a different way (boost the throughput instead of providing strict QoS guarantees) leads again to a scheduling bounds computation.

In short, the cross-layering solution is not flexible. This problem of flexibility, with the modular solution, is tackled with the adoption of a second layer described in the following paragraph.

4.1.2.3 MAC Scheduling&Abstraction layer

The idea to attach the QoS scheduler to another scheduler instead of directly to the outgoing link is to decouple the QoS task from the link issue task. In this way, at least the QoS scheduler could always be the same and it can be selected through the ones available in the literature without the need to be changed. This approach enables the possibility to cherry pick from the literature an existing packet scheduler for wired links and place it in the modular architecture in order to operate on a wireless link.

The MAC-SAL has the same structure of the QoS layer. The main difference is the purpose of the components, and of the layer as a whole. It may implement algorithms for maximizing the link throughput and, to achieve this, the MAC-SAL must classify packets according to their chances of successful transmission, and it must be able to change the order by which packets are sent to the MAC layer. To do this the MAC-SAL scheduler makes (only) use of CSI to implement its scheduling policy. In the end, even with a different goal, the MAC-SAL scheduler accomplishes the same tasks of the QoS scheduler: it divides packets into distinct flows, it stores the packets of each flow in a distinct queue and it schedules the head-of-line packets of the flows according to the desired policy.

Finally, modern systems are often equipped with heterogeneous outgoing links, which provide alternative options for forwarding a packet toward the next hop. For such systems, our architecture can be generalized by turning the MAC-SAL layer into a container of the specific MAC-SAL.

4.1.2.4 Packet prefetcher

The packet prefetcher is a software module that decides when dequeue packets from the QoS scheduler and enqueue packets in the MAC-SAL scheduler. This because the higher is the number of MAC-SAL packets, the higher is the number of packets among which the MAC-SAL scheduler can choose the next packet to transmit. Hence, the higher is the probability that the MAC-SAL scheduler can pick good packets with respect to the goals it wants to achieve. For example, let us suppose that the goal of the MAC-SAL scheduler is to keep a high throughput and that some MAC-SAL queues contain packets for destinations with bad channel conditions. If many/few other queues are not empty, then the probability for the scheduler to have at its disposal better packets to transmit is high/low. In

the end, to maximize the effectiveness of the MAC-SAL, its queues should never be empty. This is exactly the purpose of the *packet prefetcher* depicted in the middle of Figure 4.3.

4.1.3 New Results and Perspectives

Here we show some results collected by implementing the modular architecture on a real Linux system in order to simulate some possible scenarios and to test its performance.

4.1.3.1 Wireless scenarios

To validate the proposal we faced two possible scenarios selected from the EU FP7 Project "Public Protection and Disaster Relief - Transformation Centre" depicted in Figure 4.2.

The first one consists of an IEEE 802.16 (WiMAX) based network, where a base-station (BS) is connected to several subscriber-stations (SS), adopting a star-based topology. This scenario is represented by the top layer of Figure 4.2 and indeed it is a V2V star network. Since any given SS communicates directly solely with the BS, only the latter needs CSI to adapt its transmissions to the actual channel conditions. The CSI that the BS detects can be represented by the *received signal strength indicator* (RSSI) and by the *carrier-to-interference-plus-noise ratio* (CINR). For both, mean sample value and sample standard deviation are estimated and they can be sent back to the BS, by using proper messages defined by the standard, such as the *channel measurements report response* (REP-RSP), or by using fast feedback *channel quality indicator channels* (CQICH) [106].

The second scenario is composed by an infrastructure-based IEEE 802.11 (or WiFi) network, which is very similar, conceptually, to that envisaged in the previous paragraph about WiMAX. This scenario is the equivalent of the bottom left network of Figure 4.2 in which a MEOC connects several First Responders. As the BS in the previous scenario, only the AP needs CSI to adapt the transmission to the channel conditions. In this case, metrics such as RSSI and MIMO settings could be effectively employed to assess the state of the channel.

In order to perform a fine-grained evaluation is important to consider different types of services, possibly reasonable, equipped by the PPDR personnel.

Moreover, different services require different QoS bounds and then the evaluation of such a scenario enhances the challenging aspects of packet scheduling in these environments. We considered the following parameters:

- Total bandwidth: 54Mb/s;
- 20 Subscriber Stations;
- 2.7Mb/s per SS;
- For each SS, 5 flows:
  - One with $\sim$ 1.6Mb/s reserved for video or VoIP (20 flows with weight 20);
  - Another with $\sim$ 0.8Mb/s reserved for WEB browsing or direct downloads (20 flows with weight 10);
  - The other three with $\sim$ 0.1Mb/s reserved for download/sharing ($3 \cdot 20$ flows with weight 1).

We make use of the TEMPEST network simulator [107] suitable for packet scheduling testing. In particular, we make use of the CSI module of TEMPEST to map different Signal to noise Ratio (SnR) to different Packet loss probability (P_{loss}) or, equivalently, different flow bandwidth, in order to consider different link configurations.

4.1.3.2 Performance

Here we report two main results obtained by testing the modular architecture and comparing its results with the ones obtained by a well known integrated scheduler available in the literature [108] and two well known wired schedulers [100, 101]. The modular scheduler has been realized by placing the packet scheduler QFQ+ [99] both at the QoS and at the MAC-SAL layer.

Considering the scenario of Section 4.1.3.1, we define the normalized throughput as the number of packets successfully transmitted over the total number of packets sent. This index provide values in the range [0,1] indicating with 0 a total loss, i.e. no packets scheduled by the packet scheduler have been able to be successfully delivered, while 1 indicates no loss at all. Figure 4.4 gives the normalized throughput index for the modular, integrated and wired packet scheduler.

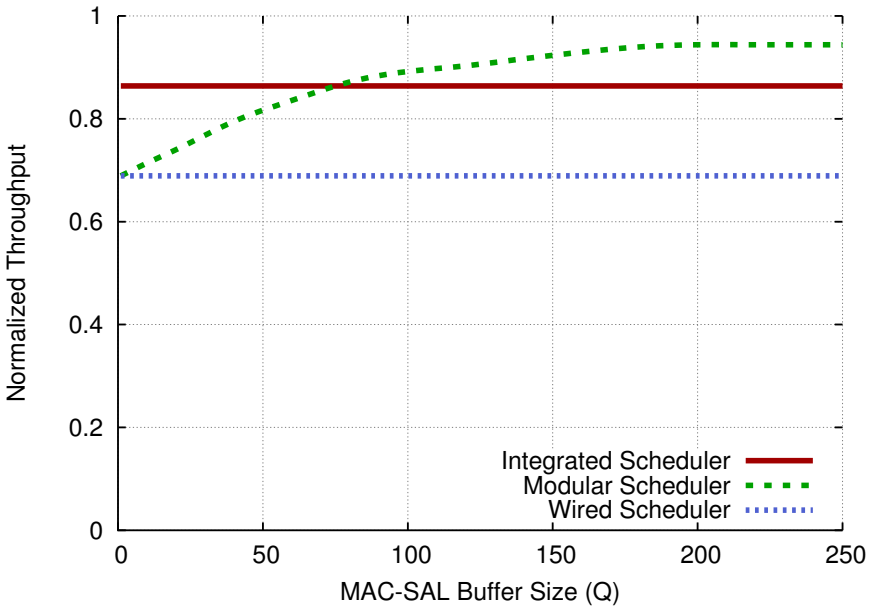


Figure 4.4: V2V normalized throughput for different scheduling architectures.

Since the behavior of the modular architecture clearly depends on the "balance" between the QoS and the MAC-SAL schedulers, we represent the results as a function of Q, which is the amount of packets that the packet prefetcher gives to the MAC-SAL scheduler. It is possible to notice how the integrated scheduler performs better than the wired scheduler, and this is a trivial results because the integrated scheduler, by using CSI, is able to choose the next packet to transmit also accordingly to the link status, thus transmitting more packets facing less loss periods. At the same time the modular scheduler shows even better performance when Q grows. As discussed before, the higher is the amount of packets that the packet prefetcher shares with the MAC-SAL scheduler, the higher is the probability of the scheduler to choose a "good" packet to be transmitted with no loss. This is clear from the picture, when Q is equal to 0, it means that the QoS scheduler is not altered from the MAC-SAL one; in fact, the performance of the modular scheduler is identical to the one achieved by the wired scheduler. At the same time, when Q grows, the MAC-SAL scheduler have more and more packets and its fine-grained choices provide an higher and higher throughput.

If, from one side, increasing Q helps the modular scheduler, on the other side it degrades the QoS guarantees provided by the QoS layer scheduler, as we can see from Figure 4.5. In this experiment the inverse of a well known metric

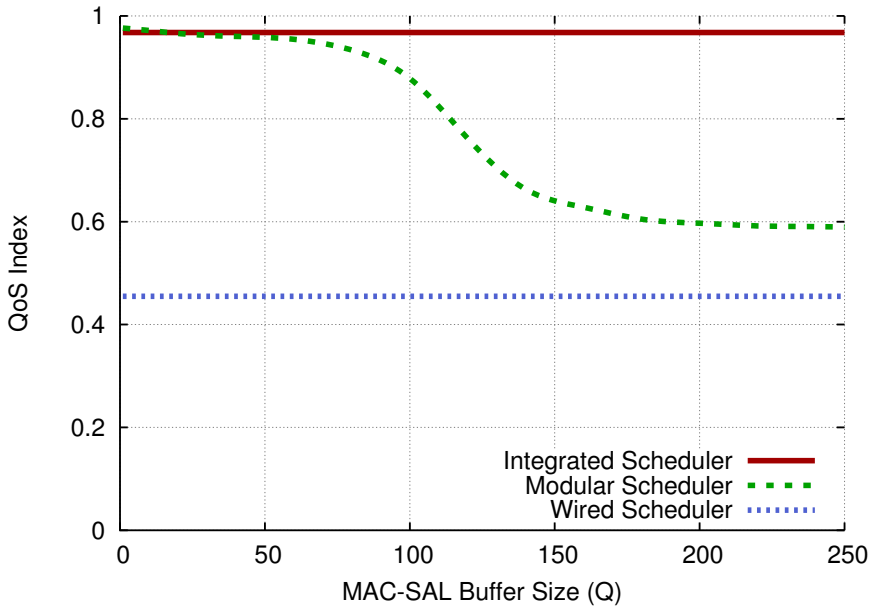


Figure 4.5: Scheduling QoS index for different architectures.

used to study packet schedulers QoS performance, the Time Worst Case Fair Index [109], is calculated. This index has been chosen because it can be evaluated in the same way of the previous one, in the range [0,1], where 0 means unfairness, while 1 means optimal fairness. Also in this case it is trivial to evaluate wired performance against integrated ones. The wired scheduler is not able to follow the channel variations given by the wireless link and it loses packets, failing to provide QoS too. At the same time the integrated scheduler was conceived exactly for this purpose and it provides a quasi optimal QoS index. Like in the previous example, the modular scheduler shows a variation on the performance by varying the Q value: at the beginning, with Q equal to 0, the QoS layer impose high QoS guarantees, by increasing Q, the MAC-SAL scheduler may reorder packets accordingly to its purpose, degrading the QoS guarantees.

The main feature of the modular scheduler is that it decouples these two main tasks, boosting the throughput and providing QoS guarantees. This is a key feature for V2V networks so that, by simply operating on the Q value, it is possible to switch from optimal throughput (high Q) to optimal QoS guarantees (low Q) or to get trade-off as for Q=100 for which the modular scheduler has a throughput higher than that of the integrated schedulers and the QoS guarantees close to the optimal ones.

4.1.4 Conclusions

This section describes and validates a modular architecture that decouples the task of providing QoS guarantees from the task of dealing with the idiosyncrasies of a wireless link. This task separations leads to a better configuration and flexibility properties scheduling in V2V networks. Therefore, moving from a network to another a node simply needs to adjust a value, named Q, to go from high throughput requirements to high QoS requirements. Moreover, the architecture is developed in a way that, if we want to deploy it on another node with a different technology, we need to change the MAC-SAL scheduler only. These features have been discussed as key points in a PPDR environment and, more generally, in a V2V network where these requirements are very challenging. Finally, we also highlighted a value for the Q parameter (equal to 100 packets) which permits network operators to expect a higher throughput than that with the best integrated scheduler available, while being able at the same time to provide QoS guarantees that are close to the optimal ones.

4.2 HFS: High througput twin Fair Scheduler

Modern wireless technologies (e.g. 3G, 4G, WiFi) steer nowadays the principles behind the design and implementation of wireless networks. State-of-the-art wireless technologies aim to support increasingly higher data rates for applications such as video streaming, web browsing and file sharing, in both stationary and nomadic/mobile scenarios. Moreover, given the growing spread of smartphones and tablets, an increasing number of users access to wireless networks everyday.

This trend puts several limits on how network and/or service providers can effectively supply adequate Quality of Service (QoS) guarantees to their users. The majority of current wireless systems directly provides some QoS capabilities (e.g. traffic differentiation and traffic prioritization). To this respect, one of the most important network sub-systems involved in the provision of QoS is the packet scheduler, which properly sets the order in which packets are sent over a given interface, both in the uplink and downlink directions. As showed in [110–112], providing high QoS guarantees in wireless systems through a smart packet scheduler represents a challenging topic.

State-of-the-art solutions to concurrently provide QoS guarantees and high throughput are based on *cross-layering* techniques; i.e. where the scheduling decisions are also made using channel state information coming from the MAC layer [113–116]. For example, per-destination channel conditions may be considered when choosing which flow to serve, in order to avoid transmission failures. In the aforementioned proposals, just *one integrated* scheduler takes *all* the scheduling decisions, based on the detected issues and on the desired QoS.

Unfortunately, integrated solution entails a few drawbacks. A high-quality scheduler for reliable links [100, 101, 109, 117–119] cannot be used and converted into a cross-layer scheduler, not without modifying it.

Even after the necessary modifications are done, the guarantees provided by the scheduler are likely to change and would therefore need to be recomputed. Finally, if the medium access protocol or the channel technology changes, or if we want to use new techniques for achieving an even higher throughput or saving energy, then the scheduler is likely to not fit the new technology or requirement. Hence, it may need to be modified again. Especially if we want to use the same scheduler on heterogeneous wireless technologies, we may need to define

a different version of it for each technology, which leads to a *set* of solutions each tailored for a specific technology. As an example, this is the case of [104, 120], [103, 121] and [122], packet scheduler solutions tailored for LTE, WiMAX and Satellite respectively.

In this section, we analyse a packet-scheduling architecture originally proposed in [123] and described in Section 4.1. The architecture focuses on *local* packet schedulers, i.e. those executed inside wireless nodes, and decides (only) the order in which packets are transmitted over the nodes outgoing links. Such an architecture preserves both effectiveness and flexibility, and permits to reuse existing schedulers without any modification; the scheduler picks the next packet to transmit according to its QoS policy, but delivers it to a *middle layer* instead of delivering it to the MAC layer. This *middle layer* then deals with the issues of the wireless medium and reorders packet transmissions if needed. With this architecture it is easy to cherry pick from the literature and combine the best solutions in terms of service guarantees, computational cost, power consumption and throughput boosting. As for the last two goals, we note that this architecture allows a system to *profit from cross-layering* while still *preserving flexibility*.

As a first contribution, we compute the overall guarantees of the entire architecture as a function of *QoS layer* and *middle layer*, and we formally define the guarantees perturbations caused by the introduction of the latter. In addition, we leverage the key property of the modular architecture (i.e. to allow an easy investigation of several schedulers combinations) by defining a new scheduler with accurate performance and low energy consumption, called High-throughput twin Fair Scheduler (HFS). HFS provides two contributions to energy reduction. Firstly, by increasing the throughput, it increases the number of packets that are successfully transmitted per fixed energy consumed (the number of retransmissions lowers); secondly, according to our experiments, the time and the energy needed to execute HFS for each packet to enqueue/dequeue are close to those of a Deficit Round Robin (DRR).

In Section 4.2.1 we provide some more details about the packet prefetcher mechanism. In Section 4.2.2 we compute the overall guarantees of the modular architecture. Then in Section 4.2.3 we show the testbed that has been used to deploy the architecture and define HFS. In Section 4.2.4, we present the new packet scheduler for wireless links. In Section 4.2.5, we validate HFS by comparing it with the best high-performance schedulers for wired links and with the best integrated scheduler. Finally, in Section 4.2.6 we highlight our conclusions.

4.2.1 Packet prefetcher

The higher is the number of non-empty MAC-SAL queues, i.e. flows, the higher is the number of head packets among which the MAC-SAL scheduler can choose the next packet to transmit. Hence, the higher is the probability that the MAC-SAL scheduler can pick good packets with respect to the goals it wants to achieve. For example, suppose that the goal of the MAC-SAL scheduler is to keep a high throughput and that some MAC-SAL queues contain packets for destinations with bad channel conditions. If many/few other queues are not empty, then the probability for the scheduler to have at its disposal better packets to transmit is high/low. In the end, to maximize the effectiveness of the MAC-SAL, its queues should be always kept as full as possible. This is exactly the purpose of the *packet prefetcher* depicted in the middle of Figure 4.3.

The behaviour of the prefetcher depends on a parameter Q, which is the MAC-SAL input buffer size, measured in number of bytes. We say that the prefetcher prefetches a packet if it invokes the function *get_next_pkt()* to get the next packet, say *pkt*, from the QoS layer and then invokes *MAC-SAL-send(pkt)* to insert the packet in the MAC-SAL. We assume that a prefetched packet is never dropped by the MAC-SAL. Finally, we denote as $S(t)$ the sum of the sizes of the packets queued in the MAC-SAL at time t.

The prefetcher operates on each of the following two events: when a new packet arrives in the QoS layer and when a new packet is dequeued from the MAC-SAL. On the former, let *pkt* be the new arriving packet. If $S(t) < Q$ when *pkt* arrives, then the packet prefetcher immediately prefetches *pkt*. On the latter, if $S(t)$ becomes lower than Q when the packet is dequeued from the MAC-SAL, then the packet prefetcher starts prefetching new packets until $S(t) \geq Q$. In other words, the combination of the set of queues in the MAC-SAL and of the packet prefetcher implements a *shared buffer* with *virtual queueing*, a device in which the memory used to store the packets is shared and the queues are only virtually separated.

Finally, it is worthwhile noting that this scheme is much more effective than an approach with distinct queues and with the same total memory. As for the latter, suppose for example that all the queues have the same length, equal to Q/M plus one maximum packet size L. If all the packets prefetched in a given time interval are destined to the same MAC-SAL queue, then the queue

becomes full only after $Q/M + L$ bytes have been prefetched. If the next packet to prefetch is again destined to the same queue, then the prefetcher must block it. In the shared-buffer scheme, instead, the prefetcher must block a packet only after $Q + L$ bytes have been prefetched, and this increases the probability that more queues are not empty.

4.2.2 Service guarantees

In this section we define first a simple *byte-delay* parameter to model the MAC-SAL. From this parameter we immediately derive the per-packet delay that the MAC-SAL introduces. Using this special delay parameter we also compute the amount of service guaranteed to each QoS flow over any time interval. Finally, from the latter service property, we compute the bandwidth guaranteed to each flow over any time interval. This last guarantee is not particularly meaningful for short time intervals, during which the bandwidth received by a flow is likely to be null or extremely low. In fact, packet delays can cause a flow to receive little or no service during a short time interval. This guarantee is instead useful for computing the long-term bandwidth received by a flow as the size of the time interval grows. For space limitations, in this contribution we do not compute other service metrics, as, e.g., worst-case fair index [100] or relative fairness [124] (they can however be easily derived from the service properties reported in this section).

We compute all the following quantities assuming that no packet is dropped by the MAC-SAL because some MAC queue is full. In practice, we assume that the offered load for each MAC queue is, on average, not higher than the rate at which the MAC scheduler guarantees that the queue is emptied. There may be temporary overloads, but the maximum size of the queue is never exceeded. As for packet dropping, consider the overall scheduling policy implemented by the combination of the QoS layer and the MAC-SAL. If this policy leads to packet dropping for memory constraints, then the same problem would occur with an integrated scheduler enforcing the same policy.

We can now introduce the special quantity from which we derive all the service guarantees reported in this section. To this purpose we consider that the MAC-SAL influences service guarantees mainly because it may reorder packets, and hence may not respect the service order suggested by the QoS scheduler.

We take this fact into account by modelling the abstract link implemented by the MAC-SAL as a special link characterized by a *byte delay* due to packet reordering. Given any packet p just queued in the MAC-SAL, we define the byte delay of the abstract link as the maximum possible value of the sum of the sizes of the packets that are queued *after* p but are transmitted over the outgoing link *before* p. We assume that the byte delay experienced by a packet is a function of the flow it belongs to, and we denote as $\Delta_i(t_1, t_2)$ the byte delay experienced by the packets of the i-th flow during a generic time interval $[t_1, t_2]$. Of course, for all flows $\Delta_i(t_1, t_2) = 0$ if packets are transmitted in the same order as established by the QoS scheduler.

Consider a packet p entering the MAC-SAL at time t_1 and exiting from the MAC-SAL at time t_2. From the byte delay we can immediately derive the component $D_i(t_1, t_2)$ of the worst-case delay of p due to the fact that the MAC-SAL reorders packets. If we denote as $B(t_1, t_2)$ the minimum bandwidth of the outgoing link during $[t_1, t_2]$, then

$$D_i(t_1, t_2) \leq \frac{\Delta_i(t_1, t_2)}{B(t_1, t_2)}. \tag{4.1}$$

We compute now the amount of service guaranteed to the i-th QoS flow in a generic time interval $[t_1, t_2]$ during which the i-th QoS flow is continuously backlogged, i.e., has pending packets either in the QoS-scheduler or in the MAC-scheduler. We express this guarantee as a function of two further variables. The first is the maximum possible value Q for the sum of the sizes of the packets that can be present in the MAC-SAL at the same time. The second is the minimum fraction of the link bandwidth guaranteed by the QoS scheduler to the i-th flow during $[t_1, t_2]$ if all the packets queued in the MAC-SAL are served in the same order as they are dequeued from the QoS scheduler. We denote as $\alpha_i^{QoS}(t_1, t_2, Q)$ this quantity. We assume that $\alpha_i^{QoS}(t_1, t_2, Q)$ is a function of also Q for two important reasons. The first is that queueing packets in the MAC-SAL before serving them introduces a delay between when a packet is dequeued from the scheduler and when the packet starts to be transmitted. This delay of course influences the guarantees provided to a flow. The second reason is that the sum of the sizes of the packets dequeued, but not yet transmitted, negatively affects the time-stamping rules and hence the service guarantees of many schedulers. The following theorem shows that the service lost by the i-th flow because of the byte delay is, in the worst-case, equal exactly to $\Delta_i(t_1, t_2)$.

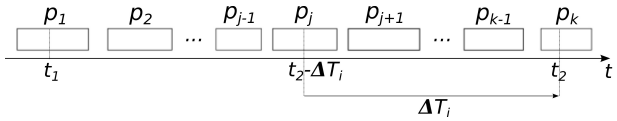


Figure 4.6: Illustration of how the byte delay reduces the service received by a flow (proof of Theorem 6).

Theorem 6. *Let $W_i^{QoS}(t_1, t_2, Q)$ be the minimum number of bytes of the i-th flow that the QoS scheduler would guarantee to be transmitted during $[t_1, t_2]$ if the link bandwidth was constant and equal to $B(t_1, t_2)$ and if $\Delta_i(t_1, t_2) = 0$, i.e., if all the packets queued in the MAC-SAL were served in the same order as they are dequeued from the QoS scheduler. Let $W_i(t_1, t_2, Q)$ be the minimum number of bytes of the i-th flow that are actually guaranteed to be transmitted if only the first hypothesis holds, i.e., if the link bandwidth is constant and equal to $B(t_1, t_2)$. The following inequality holds:*

$$W_i(t_1, t_2, Q) \geq W_i^{QoS}(t_1, t_2, Q) - \Delta_i(t_1, t_2). \tag{4.2}$$

Proof. Consider the sequence of packets of the i-th flow that the QoS scheduler would guarantee to be transmitted during $[t_1, t_2]$ if the link bandwidth was constant and equal to $B(t_1, t_2)$ and if $\Delta_i(t_1, t_2) = 0$ held. A possible sequence of such packets is shown in Figure 4.6. Each packet is depicted as a rectangle with the following property: the projection onto the x axis of its left/right side is equal to the start/finish time of the packet it represents. Especially, in the figure the packet p_1/p_k starts/ends to be transmitted before/after time t_1/t_2. According to the figure, $W_i^{QoS}(t_1, t_2)$ is equal to the sum of the sizes of the portions of p_1 and p_k served after t_1 and before t_2, plus the sizes of the packets p_2 through p_{k-1}.

To prove the thesis, we consider, with the help of the figure, what packets and portions of packets may not be served any more during $[t_1, t_2]$ if $\Delta_i(t_1, t_2) > 0$. Let ΔT_i be the time needed to transmit $\Delta_i(t_1, t_2)$ bytes at speed $B(t_1, t_2)$, i.e., $\Delta T_i = \frac{\Delta_i(t_1,t_2)}{B(t_1,t_2)}$ (recall that we assume that the link bandwidth is constant and equal to $B(t_1, t_2)$ during $[t_1, t_2]$). In the worst-case, the effect of the byte delay $\Delta_i(t_1, t_2)$ introduced by the MAC-SAL is letting the transmission of all the packets in Figure 4.6 start ΔT_i time units later. It follows that the packets and the portion of packets that, if $\Delta_i(t_1, t_2) = 0$, would start to be transmitted *too close* to t_2, more precisely not before time $t_2 - \Delta T_i$, risk to not be transmitted

at all during $[t_1, t_2]$. Figure 4.6 helps us visualize this fact. The start time of the portions of the packets p_j and p_k transmitted during $[t_2 - \Delta T_i, t_2]$, as well as of the whole packets p_{j+1} through p_{k-1} may be delayed enough to let none of these portions and full packets be served during $[t_1, t_2]$. In the end, $W_i(t_1, t_2, Q)$ is, in the worst case, equal to the sum of the sizes of only the portions and the whole packets served during $[t_1, t_2 - \Delta T_i]$ in case $\Delta_i(t_1, t_2) = 0$ (i.e., in Figure 4.6, of the portions of p_1 and p_j served during $[t_1, t_2 - \Delta T_i]$ and of the packets p_2 through p_{j-1}).

To express what we have stated so far with a formula, we denote as $W_i^{QoS}(t_1, t_2 - \Delta T_i, Q)$ and $W_i^{QoS}(t_2 - \Delta T_i, t_2, Q)$ the sums of the sizes of the full packets and the portions of packets transmitted, if $\Delta_i(t_1, t_2) = 0$, during $[t_1, t_2 - \Delta T_i]$ and $[t_2 - \Delta T_i, t_2]$, respectively. It follows that:

$$\begin{aligned} W_i(t_1, t_2, Q) \geq W_i^{QoS}(t_1, t_2 - \Delta T_i, Q) = \\ W_i^{QoS}(t_1, t_2, Q) - W_i^{QoS}(t_2 - \Delta T, t_2, Q). \end{aligned} \tag{4.3}$$

The get the thesis from the above inequality we consider that $W_i^{QoS}(t_2 - \Delta T, t_2, Q)$ is at most equal to the number of bytes that can be transmitted by the link during $[t_2 - \Delta T_i, t_2]$. Since we assume that the link works at constant speed $B(t_1, t_2)$, this number of bytes is equal to $B(t_1, t_2) \cdot \Delta T_i = \Delta_i(t_1, t_2)$. □

Finally, in the following theorem we report the bandwidth guaranteed to each QoS flow. To write the theorem, we define a last symbol, $B_i(t_1, t_2, Q)$, by which we denote the minimum bandwidth guaranteed by the system to the i-th QoS flow during $[t_1, t_2]$ in the worst-case, i.e., if the link bandwidth is constant and equal to the minimum bandwidth $B(t_1, t_2)$.

Theorem 7. *Given any time interval $[t_1, t_2]$ during which the i-th QoS flow is continuously backlogged, we have*

$$\begin{aligned} B_i(t_1, t_2, Q) \geq \\ \alpha_i^{QoS}(t_1, t_2, Q) \cdot B(t_1, t_2) - \frac{\Delta_i(t_1, t_2)}{t_2 - t_1} \end{aligned} \tag{4.4}$$

Proof. By the definitions of $\alpha_i^{QoS}(t_1, t_2, Q)$ and $B_i(t_1, t_2, Q)$, the following equalities hold for the quantities defined in Theorem 6: $W_i^{QoS}(t_1, t_2, Q) = \alpha_i^{QoS}(t_1, t_2, Q) \cdot$

$B(t_1, t_2)$ and $W_i(t_1, t_2) = B_i(t_1, t_2, Q) \cdot (t_2 - t_1)$. This allows us to rewrite (4.2) as follows:

$$\begin{aligned} B_i(t_1, t_2, Q) \cdot (t_2 - t_1) \geq \\ \alpha_i^{QoS}(t_1, t_2, Q) \cdot B(t_1, t_2) \cdot (t_2 - t_1) - \Delta_i(t_1, t_2) \end{aligned} \tag{4.5}$$

We get the thesis by dividing both sides by $t_2 - t_1$. □

In the next subsection we make some observations on the above service guarantees and on their relationship with the characteristics of an actual system.

4.2.2.1 Discussion

First, to use (4.1), (4.2) and (4.4), it must be possible to model the components of the architecture in terms of the variables that appear in those inequalities. In this respect, we note that these variables express the minimal guarantees that must be provided by these components for the whole system to provide any guarantee. In fact, if the link cannot guarantee a minimum bandwidth, or the scheduler cannot guarantee a minimum fraction of that bandwidth, or, finally, if the MAC-SAL cannot guarantee that a queued packet gets eventually transmitted, then, of course, no overall guarantee can be provided at all.

Besides, the bound (4.4) shows that the MAC-SAL affects per-flow bandwidths in both an explicit and an implicit way. The explicit way is through $\Delta_i(t_1, t_2)$. If this quantity grows less than linearly with $t_2 - t_1$, then, in the long term, the fraction of the bandwidth guaranteed to the i-th flow tends to the ideal fraction that would have been guaranteed by the QoS scheduler alone. In contrast, if $\Delta_i(t_1, t_2)$ grows at least linearly with $t_2 - t_1$, then the i-th flows may get, in both the short and the long term, a lower fraction of the bandwidth than desired. The implicit influence on the bandwidth guaranteed to each flow stems instead from the fact that the MAC-SAL influences the value of $B(t_1, t_2)$ itself in (4.4).

To lay down some guidelines for guaranteeing a low Δ_i, consider a sequence of packets to send to a next hop characterized by a bad signal or by other transmission problems. These packets will unavoidably experience high delays. If some of them carry more time-sensitive information than others, it may be important to grant to the former at least a differentiated service. The simplest

way to achieve this goal is using separate MAC queues for them and a careful enough scheduling algorithm. Another possibility is dequeuing them before other lower-priority packets inserted in the same queues. This is the main reason why we added in-flow schedulers in the MAC-SAL in Figure 4.3.

4.2.2.2 Guarantees instantiation

What's happen if we attach the QoS scheduler to the MAC-SAL layer instead of directly to the transmission link? Here we answer, in a sense, to this question by deriving the total perturbation of the service guarantees caused by the MAC-SAL scheduler.

In [125] it has been proved that the actual guarantees provided by any scheduler depend on the presence and the size of a *FIFO transmit queue*, such as a buffer ring in a modern NIC. These queues are typically used to drive communication devices and to absorb link-feeding latencies. In more detail, both the packet completion times and the service lag guaranteed by any scheduler happen to always contain an additional variation component equal to, respectively, $\frac{Q}{R}$ (on a link with a constant rate R) and Q, where Q is equal to the size of the transmit queue in bytes [125].

For a timestamps-based scheduler, these components are smaller than expected at a first glance. In fact, they are equal to the minimum possible deviation from the original service that the presence of the queue may cause: the first component is equal to just the time to empty the queue and the second component is equal to just the size of the queue. In contrast, the presence of the queue also affects timestamps computation. In this respect, the simplest scheme for computing timestamps in presence of an intermediate queue before the transmission link is updating flow timestamps on each packet dequeue as if the just-dequeued packet was immediately transmitted [125]. As a consequence, if a packet arrives, in the worst case, when the queue is full, then the corresponding flow is timestamped as if the system had already transmitted all the packets still in the queue. Despite this large timestamps perturbation, the scheduler does not however suffer from further delay and lag components, apart from the above $\frac{Q}{R}$ and Q.

We can use this result to easily derive the modification of the service guarantees of the QoS scheduler caused by the presence of the MAC-SAL. To this

purpose, we denote as Q' the maximum number of bytes that can be dequeued from the MAC-SAL after a packet p is inserted into it, and before p is dequeued from it. It follows that the queueing delay in the MAC-SAL is at most $\frac{Q'}{R}$, i.e., it is equal to the queueing delay in a FIFO queue of size Q'. In contrast, the actual buffer size is just Q, which is usually much smaller than Q'. Hence, according to the above arguments, the timestamps perturbation is equal to that caused by a FIFO queue of size Q, and therefore much smaller than that caused by a FIFO queue of size Q'. In the end, we can safely conclude that the additional packet delay variation caused by the MAC-SAL, with respect to the original time guarantees of the QoS scheduler, is not higher than $\frac{Q'}{R}$, whereas the additional service-lag variation is at most Q.

4.2.3 Test environment

In this section we present the experimental setup used to deploy, test and validate HFS and its modular structure. We configure our experiments by using a test environment called TEMPEST [107], extended to test schedulers over the modular architecture solution showed in Section 4.1; the modular structure coded in the test environment is depicted in Figure 4.7. TEMPEST is a novel tool created for evaluating the actual packet schedulers performance in several and realistic operational scenarios. With the help of this flexible tool it is possible to measure and compare each scheduler, against any other scheduler, over execution time, simulated throughput and QoS performance. An existing packet scheduler can be easily plugged into this environment, after at most some little interface changes. Inside TEMPEST, the scheduler can then be exercised with the desired sequence of enqueue/dequeue requests, through a *controller* (see the top of Figure 4.7) that iteratively switches between two phases: an enqueue phase in which it generates fake packets by picking them from a free list, and a dequeue phase in which it dequeues packets from the scheduler and reinserts them into the free list. The switch occurs according to two configurable max-total-backlog and min-total-backlog thresholds.

4.2.3.1 Configuration

Each test consisted of 10M[1] events with a typical balancing of 5M packet enqueues and 5M packet dequeues, with the controller configured so as to let flows

[1]Source code is available in [126] and contains the script *run_test.sh* that we used to run the tests.

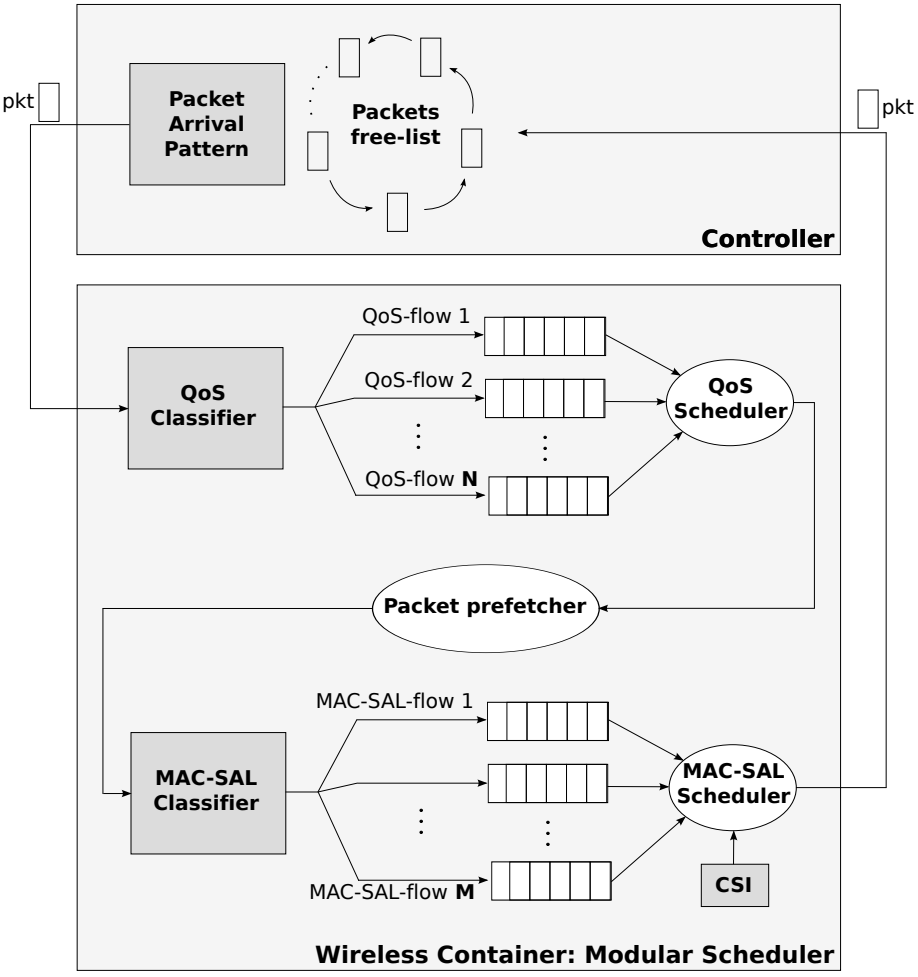


Figure 4.7: Modular architecture deployed in TEMPEST.

oscillate between a null backlog and a backlog of 10 packets each. Such an enqueue/dequeue pattern happened to be the most demanding one for the schedulers. Packets had a fixed size of about 1700 bytes, with no cache-line alignment. The payload of the packets was never either read or written. No migration and no packet drop occurred in any run.

4.2.3.2 Arrival Pattern

The controller switches between enqueue and dequeue mode to control the number of pending packets of each run. The packet arrival pattern block (see Figure 4.7) runs by default two basic arrival patterns:

- *Smooth* pattern: the controller, in the enqueue phase, iteratively generates one packet for each flow with a lower backlog than the other flows (by filling flows in a round-robin fashion);
- *Bursty* pattern: the controller, in the enqueue phase, iteratively generates a burst of packets for each flow with a lower backlog than the other flows, where the burst has a random size proportional to the flow weight.

In addition to the above rules for switching between the dequeue and enqueue phase, if the backlog of a flow drops below a configurable threshold while the controller in dequeue mode (e.g. one packet), then the controller may switch back, with a configurable probability, to the enqueue phase, filling again the queue of this flow and the one of the others. This additional mechanism is used to increase the randomness of the arrival pattern, and hence make harder for a packet scheduler to guarantee a tight packet delay.

The tests described here have been executed with a *bursty* packet arrival pattern.

4.2.3.3 Wireless scenarios

In this subsection we present two general purpose scenarios, their equivalent specific purpose scenarios have been studied for the EU FP7 Project "Public Protection and Disaster Relief - Transformation Centre" and are depicted in Figure 4.2.

The first consists of an IEEE 802.16 (WiMAX) based network, where a base-station (BS) is connected to several subscriber-stations (SS), adopting a star-based topology. Since any given SS communicates directly solely with the BS, only the latter needs CSI to adapt its transmissions to the channel conditions. The CSI that the BS can sample can be represented by the *received signal strength indicator* (RSSI) and by the *carrier-to-interference-plus-noise ratio* (CINR). For both, mean sample value and sample standard deviation are estimated and can be sent back to the BS, by using proper messages defined by the standard, such as the *channel measurements report response* (REP-RSP), or by using fast feedback *channel quality indicator channels* (CQICH) [106]. The equivalent of this general purpose scenario in the FP7 Project is the star topology connection between the Mobile Emergency Operations-control Center (MEOC) and the First Responder Chiefs (FRc).

The second scenario is composed by an infrastructure-based IEEE 802.11 (or WiFi) network, which is very similar, conceptually, to that envisaged in the previous paragraph about WiMAX. Consider a specific case, where a set of users, each equipped with a WiFi-enabled station (STA), are wireless-connected together using a single deployed AP. As the BS in the previous scenario, only the AP needs CSI to adapt the transmission to the channel conditions. This could be the case for several laboratories and offices located on the same floor of a building. In this case, metrics such as RSSI and MIMO settings could be effectively employed to assess the state of the channel. The equivalent of this general purpose scenario in the FP7 Project is the star topology connection between the FRc and the First Responders (FR).

In both the previously described scenarios (general and/or specific purpose), several applications and services can be effectively deployed. To name a few, video and voice services, together with web-browsing or database query, represent some of the most-used network applications.

TEMPEST is able by default to configure both the described scenarios thanks to the CSI module depicted in Figure 4.7 which is the core of the wireless simulation part. For the sake of brevity, we present the simulation results for the second scenario. As a side note, we obtained similar results in the first scenario, so our claiming are relevant also for the second one. The simulation parameters are the following:

- Total bandwidth: 54Mb/s;

- 20 Subscriber Stations (SS, or FR in Figure 4.2);
- 2.7Mb/s per SS;
- For each SS, 5 flows:
  - One with $\sim$ 1.6Mb/s reserved for video or VoIP (20 flows with weight 20);
  - Another with $\sim$ 0.8Mb/s reserved for WEB browsing or direct downloads (20 flows with weight 10);
  - The other three with $\sim$ 0.1Mb/s reserved for download/sharing ($3 \cdot 20$ flows with weight 1).

To simulate the wireless link we used the CSI module of TEMPEST in which different Signal to noise Ratio (SnR) are mapped in different Packet loss probability (P_{loss}) or, equivalently, different flow bandwidth accordingly to the ns-3 WiFi module. For base stations channel condition, we configured the emergency network scenario in order to experience a packet loss probability ranging linearly from 10^0 to 10^{-4}. These different values are used at the MAC-SAL level to configure the packet scheduler behaviour (it will be properly analysed in Section 4.2.4.2) by defining different flow weight distributions through a conversion function. The same conversion can be made also by starting from the SnR input values or, equivalently, by using the different instantaneous flow bandwidth achieving the same weight distribution through a complementary conversion function. Due to its simplicity, in the following we will always talk about P_{loss} and its related conversion function to calculate the MAC-SAL flow weights.

From an user classification point of view, we define also a threshold to distinguish user in *good* and *bad* channel condition. This threshold is placed at 20% of packet loss. Users/flows with less or equal to 20% of packets lost are considered as users/flows in good conditions, while users/flows with more than 20% of packets lost are considered as user/flows in bad conditions. This because above 20% most applications do not work properly; for instance, both the TCP window and the VoIP controller mechanism do not perform well under the aforementioned condition.

4.2.3.4 Statistics

To measure and validate schedulers' performance we implemented different well-known metrics. We used all of these indicators to easily design and test different packet scheduler solutions. Here is a list of the main performance metrics and a relative detailed description:

- **Throughput:** to measure the throughput we simulated the *normalized throughput* achieved by the schedulers. This parameter is the amount of successfully transmitted packets for each flow divided by the amount of total sent packets. We computed the normalized throughput as

$$\text{thr} \equiv \frac{\sum_i pkt_{sent_i}(1 - \text{P}_{loss_i})}{\sum_i pkt_{sent_i}}$$

  where pkt_{sent_i} is the number of packets sent by the flow i, P_{loss_i} is the packet loss probability of the flow i and $pkt_{sent_i}(1 - \text{P}_{loss_i})$ is the number of successfully transmitted packets by the flow i;

- **Execution time:** the actual execution time measurement is a key point of TEMPEST. As a matter of fact, it is possible to measure, due to TEMPEST emulation nature, the real execution time of a packet scheduler by running actual Kernel code in user space. At the end of each run, we measured the total execution time of the run and divided it by the total number of enqueues, or equivalently of dequeues, executed. We obtained therefore the average total packet-processing time, i.e. the execution time of an enqueue plus a dequeue, inclusive also of the cost of generating and discarding an empty, fixed-size packet;

- **Energy consumption:** according to the models in [127,128], lower/higher relative execution times imply also lower/higher relative energy consumptions. Therefore, we consider the energy consumption parameter as a direct consequence of the execution time of a scheduler;

- **Queueing delay:** the queueing delay for each flow is measured as the maximum amount of time experienced by any packet inside the flow expressed in number of dequeue events. The delay experienced by a packet pkt has been calculated as $t_{deq} - t_{enq}$ where t_{enq} is the number of packets

dequeued by the system when pkt is enqueued, while t_{deq} is the number of packets dequeued by the system when pkt is dequeued;

- **Time Worst-case Fair Index (T-WFI):** another useful QoS guarantees metric is the T-WFI (described in [99]), which allows to evaluate, in a single value, both fairness and delay: on one hand it shows the fairness, in terms of delay from the worst-case completion time of a packet in a perfectly fair system, while on the other hand it allows to instantly calculate the actual delays incurred by packets depending on the occupation of queues. In a perfectly fair system, the worst-case completion time of a packet is equal to its queue length (including the packet itself) divided by the packet's flow guaranteed rate. Assuming that the link rate R is constant, we measured T-WFI^k for flow k as:

$$\text{T-WFI}^k \equiv \max\left(t_{deq} - t_{enq} - \frac{Q^k(t_{enq})}{\phi^k R}\right)$$

where t_{enq} and t_{deq} are the same of the queueing delay parameter, $Q^k(t_{enq})$ is the backlog of flow k just after the arrival of the packet and ϕ^k is the relative weight of the flow.

4.2.3.5 Test equipment

We ran our tests on two systems with the following software and hardware characteristics:

- Ubuntu 12.04.2, 64-bit kernel 3.2.0, Intel Core Dual-E2200 @ 2.20GHz, gcc 4.6.3 -O3;
- OS X 10.7.5, Darwin 11.4.0, Intel Core i5-2557M @ 1.8 GHz, gcc 4.2.1 -O3.

Since the relative performance of the schedulers were about the same on the two systems, we report our results only for the first system. In the next two sections we first show how we used the test environment described here to define HFS packet scheduler, and then we show how we validated its performance.

4.2.4 HFS definiton

Once described the architecture in Section 4.1 and the test environment in Section 4.2.3, it is possible to show an easy procedure to define new flexible and efficient schedulers for wireless links. Inside this environment it is easy to cherry pick from the literature and combine the best solutions in terms of service guarantees, computational cost, power consumption and throughput boosting just by combining existing high-performance schedulers for wired links. To run our tests and design new high-performance schedulers for wireless links we have considered the best schedulers for wired links available in the literature, that is:

- WF^2Q+: an optimal service guarantees with $O(logn)$ complexity [100];
- DRR: a scheduler with extremely low time complexity $O(1)$, but with $O(n)$ deviation form optimal service [101];
- QFQ+: a quasi-optimal service guarantees scheduler with execution time close to the DRR one [99].

In the following, we will describe the procedure of combining these schedulers placing them at QoS layer and/or at MAC-SAL layer and evaluating the final performance results.

4.2.4.1 QoS scheduler

From a QoS point of view, the characteristics of the schedulers are the same for both the modular architecture over wireless links and for the typical one over wired links. Due to this fact, we do not present any test results for the QoS algorithm choice and we select QFQ+ because it achieves quasi-optimal service guarantees while still preserving execution time close to the optimal, i.e. DRR [99]. Moreover, we have verified also that the choice of the QoS scheduler does not affect any of the MAC-SAL figures of merit. Accordingly to these facts, in the next Subsection 4.2.4.2 we consider the QoS scheduler settled with QFQ+ in place.

4.2.4.2 MAC-SAL scheduler

A different evaluation must be done for the MAC-SAL layer. Here we need to consider all the metrics in field and the wireless-scenario parameters presented in Section 4.2.3.3. Inside the MAC-SAL layer, the scheduler can be configured with different weight distribution policy among flows based on their P_{loss}[2]. The distributions considered to run our experiments have been:

- **Linear**: considered 10 flows $\mathrm{f}_0, \ldots, \mathrm{f}_i, \ldots, \mathrm{f}_9$ with 10 different P_{loss} ordered from the higher losses flows to the lower losses flows. With a linear weight distribution each MAC-SAL flow is initialized with a weight equal to i, where i is the position of the flow in the ordered list;
- **Exponential**: considering the same example of the previous distribution, the weight of a generic flow in this case is initialized as 2^i where (again) i is the position of the flow in the ordered list;
- **Analogical**: here a flow k is initialized with a weight equal to $(1 - \mathrm{P}_{loss_k}) \cdot 1000$.

Here we show only results computed with the last distribution because it achieves the best trade off in terms of throughput boosting and QoS guarantees.

To remind that the goal of MAC-SAL scheduler is to boost the throughput, in figure 4.8 it is reported the normalized throughput of different MAC-SAL schedulers: it is shown that the one achieving the best performance is QFQ+, followed very closely by WF^{2}Q+, while DRR algorithm is the worst one. Besides, because a second scheduler placed under the QoS layer can introduce a delay, it is important to evaluate this metric also at MAC-SAL layer. In fact, in figure 4.9 it is showed the queueing delay introduced by the three different schedulers to the flows/users in best channel condition: also in this case QFQ+ shows the best performance, introducing a lower delay with respect to the other schedulers. Moreover, the effective maximum delay measured is lower than the worst case delay calculated in Section 4.2.2. Finally, by looking at figure 4.10, only QFQ+ reaches execution time values close to the optimal one of DRR.

[2]The same weight distributions can be achieved by using different channel state indicators and their equivalent transformation function accordingly.

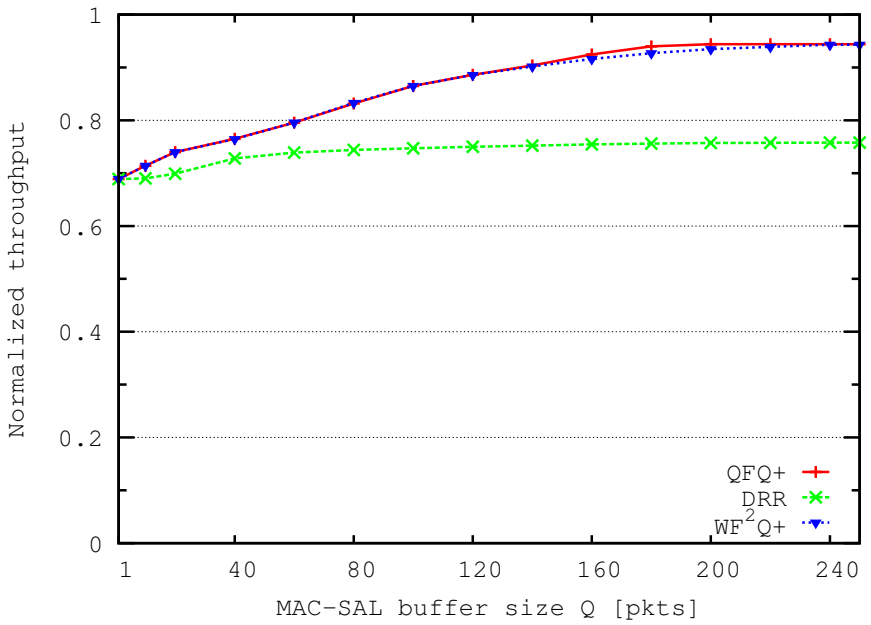


Figure 4.8: Normalized throughput for different MAC-SAL scheduler algorithms.

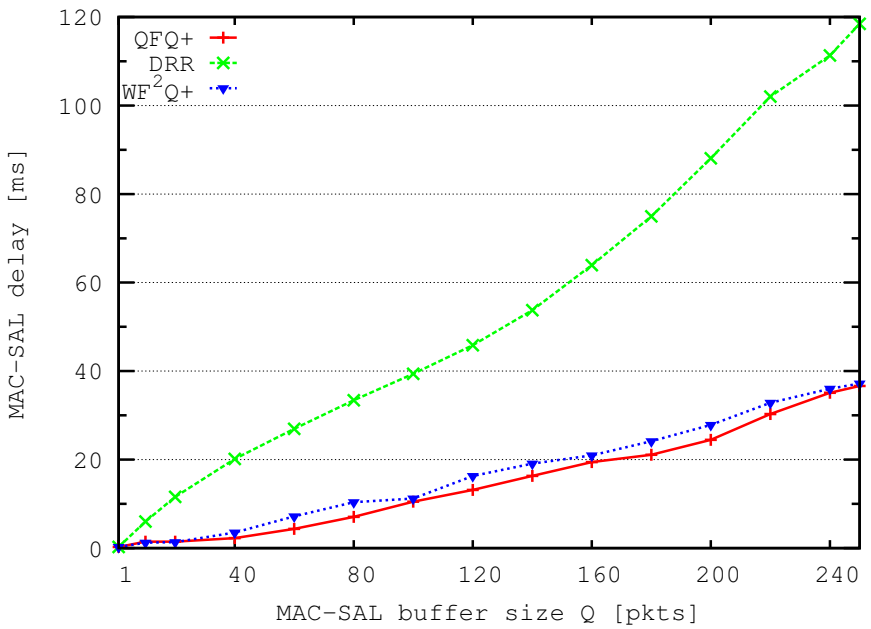


Figure 4.9: Queueing delay introduced by different MAC-SAL scheduler algorithms.

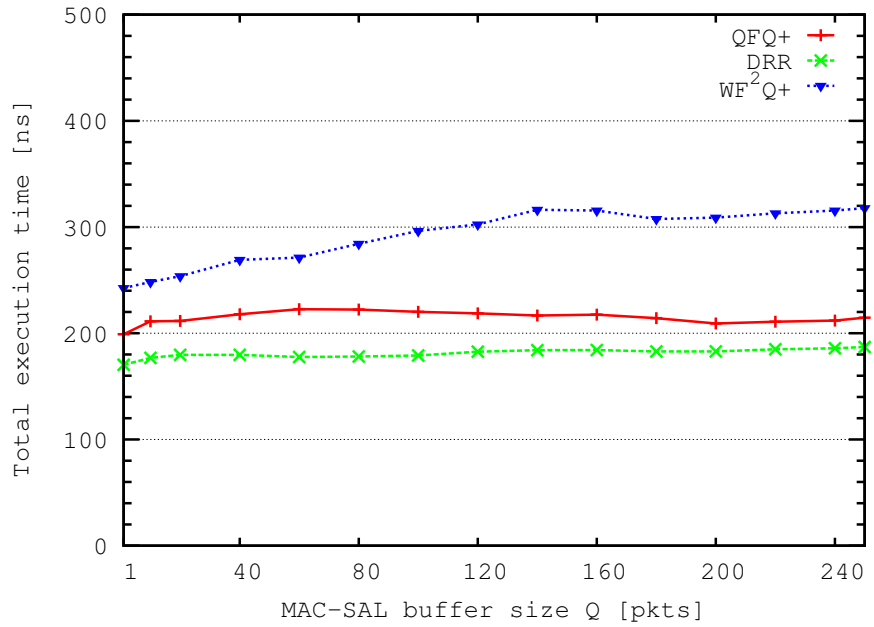


Figure 4.10: Execution time for different MAC-SAL scheduler algorithms.

4.2.4.3 HFS

After the evaluation of different schedulers placed at the QoS layer and at the MAC-SAL layer we defined High-throughput twin Fair Scheduler (HFS) as the best solution, by placing QFQ+ scheduler both at the QoS layer and at the MAC-SAL layer [91]. The cascade of two QFQ+ packet schedulers follows the flexible approach of the modular architecture: at the QoS layer QFQ+ is always the same, independently of the hardware and of the network, while at the MAC-SAL layer QFQ+ uses Channel State Information to improve the scheduling decision and only this last module should change moving from an hardware platform to another.

According to the Section 4.2.2, HFS scheduler gives the same service guarantees of the QoS scheduler, which is QFQ+, with an additional packet delay variation not higher than $\frac{Q'}{R}$, whereas the additional service-lag variation is at most Q.

While in this section we have considered intra-model tests to deploy and to configure the best high-performance scheduler solution compliant with the modular architecture, in the next section we will consider extra-model tests among HFS, the best schedulers for wired links and the best integrated one.

4.2.5 Results

Here we validate the efficiency of HFS packet scheduler by comparing its performance with the best packet schedulers for wired links and with the best integrated scheduler for wireless links. To validate HFS results we considered different schedulers as a benchmark:

- **W^{2}F^{2}Q**, to validate HFS throughput results against the best integrated scheduler available in the literature [115].
- **WF^{2}Q+**, to validate HFS guarantees against the best high-performance scheduler for wired links in terms of service guarantees.
- **DRR**, to validate HFS execution time against the best high-performance scheduler for wired links in terms of time complexity.

Let us call *double-SCHED* a double instance of the scheduler *SCHED* compliant with the modular architecture with *SCHED* placed both at QoS and at MAC-SAL layer (e.g. double-DRR is obtained placing DRR scheduler both at QoS level and at MAC-SAL level)[3].

4.2.5.1 Throughput

The integrated scheduler W^{2}F^{2}Q models temporary bursty channel errors of a wireless link only with a two-state Markov chain in order to to simulate flows in good or bad channel conditions. Unfortunately, such a distinction does not hold in our model, since we considered a scenario in which a flow can experience long-term bad conditions as well, based for instance on the position of the user. Anyway, comparing W^{2}F^{2}Q with HFS can benefit the first in terms of throughput, since users in good state manage the total bandwidth available leaving flows in bad state with null weight. We simulate this kind of event by using a threshold in our environment, where flows with more than 20% of packet loss are classified as flows in bad state with weight equal to zero, while the other flows are classified as flows in good state with weight equal to their own weight according to the weight distribution policy. Figure 4.11 shows how in this favourable case W^{2}F^{2}Q scheduler obtains high normalized throughput. However, HFS achieves better throughput performance with respect to W^{2}F^{2}Q for Q $\geq$ 100 due to his fine-grained choice among good flows, which increases with the MAC-SAL shared buffer size Q.

[3]Following this nomenclature HFS is exactly the same of double-QFQ+.

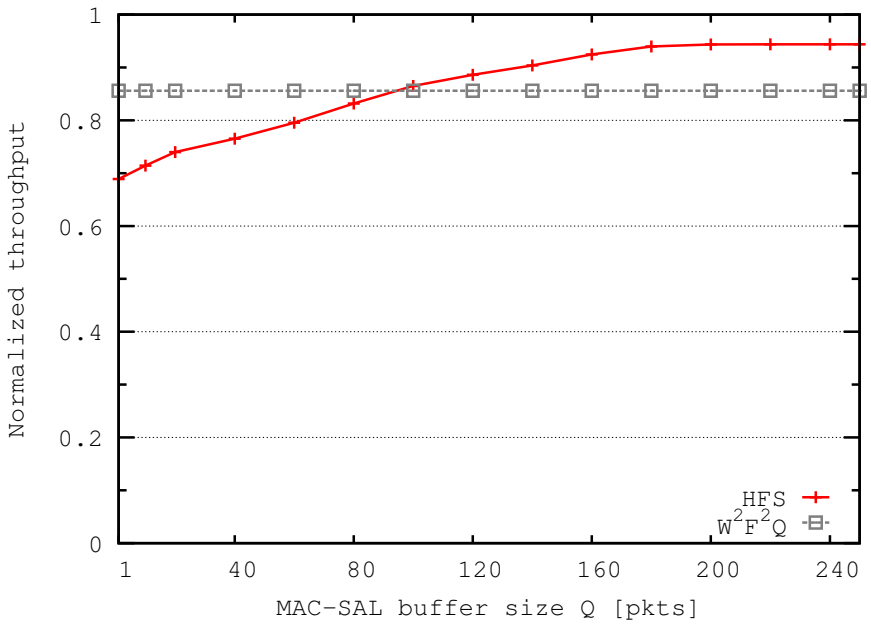


Figure 4.11: Normalized throughput for HFS and W^2F^2Q.

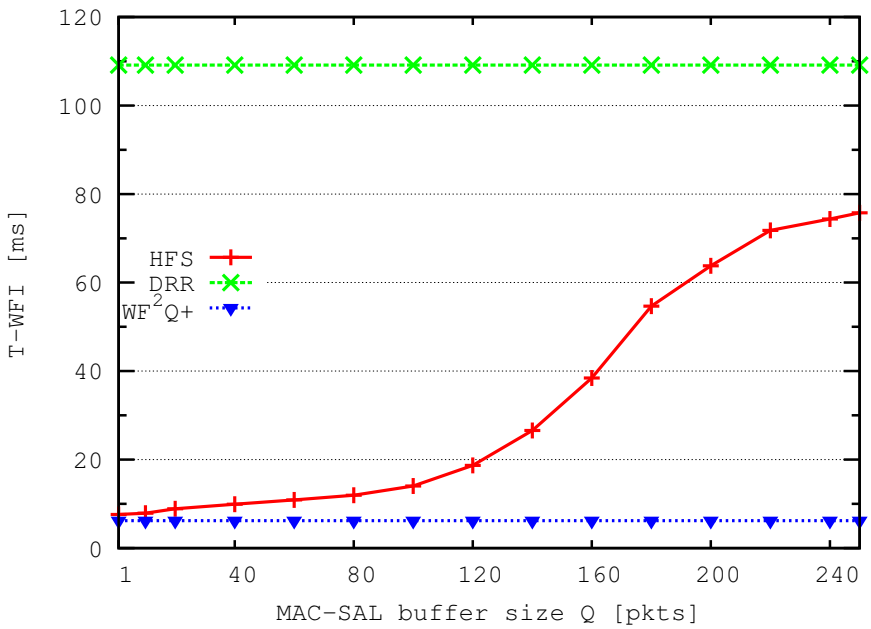


Figure 4.12: T-WFI for HFS, WF^2Q+ and DRR, for flows in *intermediate conditions*.

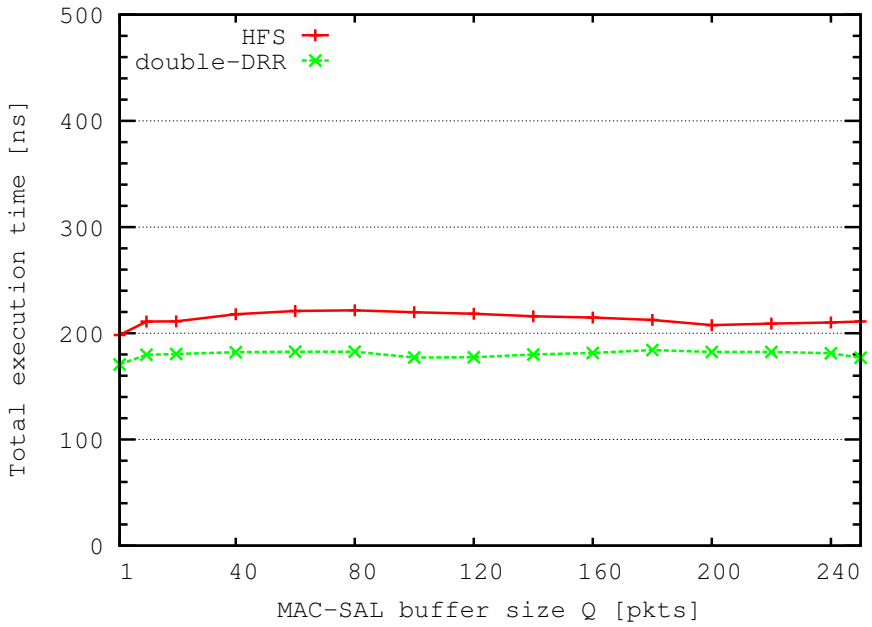


Figure 4.13: Execution time of HFS against DRR and WF^{2}Q+.

4.2.5.2 QoS guarantees

With respect to QoS metric, we show only Time Worst-case Fair Index because it allows to evaluate, in a single graph, both fairness and latency: on one hand it shows the fairness in terms of delay from the worst-case completion time of a packet in a perfectly fair system, while on the other hand it allows to instantly calculate the actual delays incurred by packets depending on the occupation of queues. In a perfectly fair system, the worst-case completion time of a packet is equal to its queue length (including the packet itself) divided by the packet's flow guaranteed rate. Figure 4.12 validates the performance of HFS in terms of QoS for flows in *intermediate conditions*, i.e. for flows/users with at least 0.8 of normalized throughput per-flow (HFS guarantees this kind of worst case performance if the station lose an amount of packets that is less or equal to 20%). With a packet loss ratio above 20% most applications do not work properly; for instance, both the TCP window and the VoIP controller mechanism do not perform well under the aforementioned condition. Why we haven't considered W^{2}F^{2}Q in the QoS-guarantees graph? As we have said, the algorithm W^{2}F^{2}Q model temporary bursty channel errors of a wireless link only. This way the scheduler tries to implement the *compensation* mechanism to guarantee long

term fairness between different flow-condition services under the hypothesis that an error-prone flow has sufficient time to make up for their lag after recovery of channel (see [115]). Unfortunately, such assumption does not hold in our model because we considered a scenario in which a flow can experience also long-term bad channel condition, based on the position of the user. In this sense, comparing W^2F^2Q with our model from a QoS guarantees point of view can be unfair because users/flows classified in bad state can lag forever with quasi-zero service guarantees.

4.2.5.3 Execution time

To evaluate HFS in terms of execution time (and energy consumption) we use as benchmark the scheduler double-DRR, which is the simplest possible solution to achieve high throughput preserving QoS guarantees with a low computational cost of $O(1)$ with our modular architecture. Figure 4.13 shows that HFS has a really close execution time to the double instance of DRR which is the best high-performance scheduler for wired links in terms of execution time. Furthermore, the picture shows that the packet-processing time of HFS is compliant with the link rate (i.e. much lower than the packet transmission time). Why we haven't considered W^2F^2Q in the execution time graph? W^2F^2Q scheduling complexity is $O(N)$, too high for schedulers in backbone network where N is very large and more than the HFS scheduling complexity which is close to the DRR optimal cost of $O(1)$.

4.2.5.4 Energy consumption

According to the models in [127, 128], lower/higher relative execution times imply also lower/higher relative energy consumptions. It is the case of HFS, as showed in Figure 4.13, with a really close execution time to DRR which is the best high-performance packet scheduler for wired links in terms of execution time. Moreover, in Figure 4.11 we show that HFS achieves also higher throughput with respect to the best integrated packet scheduler for wireless links, which is W^2F^2Q. In this way, by increasing the throughput, HFS increases the number of packets successfully transmitted per energy consumed (the number of retransmissions is also lower). Therefore, the reduction of the execution time and the throughput's boosting permits to HFS to reduce the energy consumption.

By comparing Figure 4.11 and 4.12 we can say that it is possible to choose the desired trade-off between throughput-boosting level and granularity of the service guarantees, by only setting the parameter Q. For instance, by setting a value of Q equal to 100 packets, HFS reaches a normalized throughput close to 90%, which is greater than the W^2F^2Q one, while still preserving service guarantees close to the optimal ones of WF^2Q+.

4.2.6 Conclusions

In this section we validated a modular architecture that decouples the task of providing QoS guarantees from the task of dealing with the idiosyncrasies of a wireless link. We extended a test-environment which allows to easily run, test and analyse existing schedulers over this architecture. To complete the picture, we also defined HFS, a new flexible, efficient and accurate scheduler for providing QoS guarantees and high throughput over wireless links. To prove the effectiveness of HFS we showed, through experimental results, its high performance. HFS provides higher throughput with respect to Wireless Worst-case Fair Weighted Fair Queueing, the best integrated scheduler available so far. Furthermore, HFS provides low latency and accurate fairness, close to the optimal ones of Worst-case Weighted Fair Queueing. Last but not least, HFS guarantees execution times and energy consumptions that are close to those of a Deficit Round Robin. We also demonstrated the possibility to dynamically set, by changing the MAC-SAL buffer size Q, the desired trade-off between throughput-boosting level and granularity of service guarantees, a feature that allows HFS to be an adaptive packet scheduler able to tune its performance by following users and network requirements. Finally, we also highlighted a value for the Q parameter (equal to 100 packets) which permits network operators to expect a higher throughput than with the best integrated scheduler available so far, while being able in the same time to provide QoS guarantees that are close to the optimal ones.

4.3 TEMPEST

The packet scheduler algorithm is an important component which decides the order in which packets are sent on a link, it is important in order to provide QoS and insulate different type of traffic on a network; in fact, each node should implement it in order to guarantee high level of QoS (as RFC 3289 recommends). In the last two decades, the packet scheduling problem has been well investigated and lot of research papers and studies have provided a wide amount of solutions [99–101, 119, 129–131], in particular for wired networks. At the same time, wireless technologies have become vital for public networks, several technologies like WiFI, WiMAX, LTE, 4G and future 5G are still growing and defining new standards. Moreover, technologies like LTE and 4G/5G are QoS-based/QoS-driven and providing QoS guarantees in wireless environment is a hot and challenging topic in terms of scientific research. However, this is not the only challenge; the number of connected devices is still growing due to the capillarity of wireless access and the Internet of Things (IoT) and how to provide low energy consumption as well as QoS is a topic under investigation [132].

A software module involved in this scenario is the packet scheduler; to design a packet scheduler for a wireless link is not trivial, high quality packet schedulers for wired links cannot be used due to the different idiosyncrasies of the wireless medium. This scenario has forced the introduction of several "technology-dependent" packet schedulers like [108, 133–136]. Such schedulers are based on the *cross-layering* technique: scheduling decisions are also made using channel state information coming from the MAC layer, in this way just one *integrated* scheduler takes all the scheduling decisions, on the basis of the channel issues and of the desired QoS. Following this approach, if the medium access protocol or the channel technology changes, or if we want to use new techniques for saving energy, then the scheduler is likely not to fit the new technology or requirements. In this scenario it is clear that is importat to have a prompt feedback regarding the packet scheduler solution in order to evaluate it and steering the research in this field.

Is also important to evaluate existing schedulers, because there exist many different algorithms with many different features. How to choose among them? The decision depends a lot on the operating conditions. For large number of flows, asymptotic complexity is important. For small number of flows, or certain

weight distributions, QoS guarantees or actual run times should be more important. Theory can tell us about worst-case service guarantees and asymptotic complexity, but only measurements and/or testbeds can tell us about actual QoS guarantees and run-time constants for the time complexity.

Unfortunately, a simple yet effective way to test a packet scheduler is hard to achieve. The possible ways are the following: testing the performance of a packet scheduler algorithm in operational network (real testbed) or in simulated environment. With the former approach it is not easy to set and control important scheduling parameters such as bandwidth, delay and queue size and, at the same time, it requires to properly run a full environment setup with an almost complete ad-hoc implementation of the scheduling algorithm. The latter approach is easier to control, but simulators are often only an approximate model of the desired setting, especially for what regards the packets arrival patterns and their interaction with the protocol itself.

To solve the aforementioned problems, here we present TEMPEST, a new Test EnvironMent for Performance Evaluation of the Scheduling of packeTs. TEMPEST is an UNIX-based open tool able to measure the actual performance of a packet scheduler under the desired operating conditions. With its fine-grained level of configuration parameters, TEMPEST is able to help in evaluating existing scheduler in order to properly choose the best one depending on the operation conditions (e.g environment condition, amount of flows, channel bandwidth, queue size, the packet arrival pattern and so on). Scheduling performance can be evaluated in terms of execution time, QoS guarantees with standard fairness indexes (Queueing Delay, bit worst-case fair index B-WFI , time worst-case fair index T-WFI and relative fairness index RFI [99]) and throughput. A key point of TEMPEST is that it permits to measure the kernel code in user space, reducing the surrounding operations involved in the packet scheduling task, allowing TEMPEST to be a flexible, portable and effective tool for packet scheduling testing able to generate traffic at 40Mpps and more on a common workstation. Furthermore, adding a new scheduler to TEMPEST is trivial[4], a fundamental aspect in order to help the research in emerging areas like 4G/5G in which QoS is a key parameter and the design and test of a new packet scheduler is challenging and time-consuming.

[4]With at least some interface changing is trivial also to move from the TEMPEST scheduling algorithm to the kernel one.

The rest of the description is organized as follows. Section 4.3.1 discusses the related work. In Section 4.3.2 we present TEMPEST model and describe the performance metrics available to evaluate packet schedulers. In Section 4.3.3, we validate the accuracy of TEMPEST through simulation results. Finally, in Section 4.3.4 we draw our conclusions.

4.3.1 Related tools

One of the first and major attempt to provide an accurate yet simple network simulator has been performed by Luigi Rizzo with Dummynet [137]. Dummynet works by intercepting communications of the protocol layer under test and simulating the effect of finite queues, bandwidth limitations and communication delays. It requires minimum modifications to the protocol stack to being built, allowing experiments to be run on a standalone system. The Dummynet approach gives basically most of the advantages of both simulations and real-world testing: great control over operating parameters, simplicity, easy use of traffic generator and, consequently, running an experiment is as easy as running the desired application on a workstation. An interesting strong point of Dummynet is the introduction of almost no overhead in the communications, in this way experiments can be done up to the maximum operating speed supported by the system in use, and this is also a TEMPEST feature. Recently, Dummynet has been extended in [138] after becoming a widely used link emulator. The authors added different packet schedulers, available also in linux Kernel and used on wired links, such as FIFO, DRR, QFQ and WF^2Q+. Moreover, they also implemented Linux and Windows version of the simulator, in addition to the native FreeBSD and Mac OS X ones.

The same author of Dummynet also implements a lightweight tool to measure the actual execution time of a packet scheduler in [131]. In this work, the authors designed a featherweight model to reduce the surrounding operations involved in the packet scheduling task, reducing the entire overhead of the system so as to measure the actual execution time of the packet scheduler in exam. The tool derives from Dummynet in some parts and the available schedulers are the same available in the Dummynet extension [138]. Unfortunately, this tool does not permit the use of realistic and different packet arrival patterns to validate the computed measurements. Moreover, no service guarantees can be calculated and no wireless scenarios can be simulated as well.

The investigation about packet scheduling performance has been carried out also by Sami Ben-Guedria *et al.* in [139]. The authors implemented an advanced module for ns-2, named PolyMAX, in order to evaluate the performance of Mobile WiMAX networks. The aim of the work was to help the research in the design and the implementation of the Quality of Service (QoS) classes and Mobile WiMAX scheduling for the WiMAX protocol. To do it, authors added a complete QoS support and AMC ability to the NIST module of ns-2 [140], implementing new features such as the management of service flows and QoS parameters. The tool comes with three scheduling disciplines like Round Robin (RR), Weighted Round Robin (WRR) and maximum Signal to Interference Ratio (mSIR). A weak point of this solution is the flexibility. It is tailored for WiMAX technologies with the ns-2 module and it is also not clear how to add existing schedulers or a new one.

In another work [141], Costas Courcoubetis and Vasilios A. Siris presented procedures and tools for the analysis of network traffic measurements. The tools consist of stand-alone modules that implement advanced statistical analysis procedures without assuming a specific source traffic model. The goal of the paper was to demonstrate the application of the tools for answering questions related to network management and dimensioning, such as the maximum link utilization when some quality of service is guaranteed, and how this utilization is affected by the link buffer and the scheduling discipline. The tools consist of two statistical analysis modules (written in C), and a flexible user interface (written in Java), that allows users with a Java enabled browser to create, modify, save, and execute experiments. As an interesting aspect, these tools can help network engineers to understand how the network control mechanisms, such as packet scheduling, can affect the performance of a network, hence assist them in operating and dimensioning networks more efficiently. The tools proposed operate with two possible scheduling disciplines, FIFO and priority scheduling. A strong point of this solution is the user interface which provides a flexible environment; following this approach, TEMPEST comes out with multiple scripts in order to run easily extensive simulations and to plot all the results for an user-friendly feedback. A weak point of this work is the passive measurement of network traffic; in this way, stressing the packet scheduling behaviour in different conditions is hard as deploying different real testbeds.

All of these related researches are able to evaluate, in a sense, different packet schedulers behaviour but none is able to measure effectively all the performance

metrics used to characterize a packet scheduler. The TEMPEST solution comes out with the motivation to help the research in the packet scheduling field, giving clear information about actual execution time, real QoS guarantees and throughput, validating these parameters under realistic traffic models and over different and easy-to-configure scenarios. It doesn't require specific OS as well as powerful hardware and provides several scripts to ease the learning curve and the output readiness.

4.3.2 TEMPEST Module

We implemented TEMPEST [107], starting from the test environment [142] used to prove the effectiveness of QFQ in terms of execution time [131]. The main goals has been to add realistic and configurable packet arrival patterns (traffic generator) to validate measures upon the packet scheduler in exam, to add the principal QoS indicators to evaluate the actual QoS guaranteed by a packet scheduler and, finally, to add the wireless module simulator in order to test *integrated* packet scheduler as well as generally evaluate (or define) wireless scheduling solutions. To improve the learning curve of TEMPEST, several scripts have been also deployed in order to help the researchers in configuring multiple environments for extensive simulations and visualize user-friendly results.

A key point of TEMPEST is that it permits to run the kernel code in user space. Measuring packet scheduling performance in kernel space is difficult for two reasons:

- Setting up a suitable test environment is as tough as setting up a real testbed;
- Packet generation, reception and device drivers dominate the measurements.

Basically, TEMPEST makes measurements by running the kernel code in user space, this permit to achieve several benefits:

- It is easy to generate traffic at 40 Mpps and more (compared to 200-500 Kpps on the wire for the same hardware);

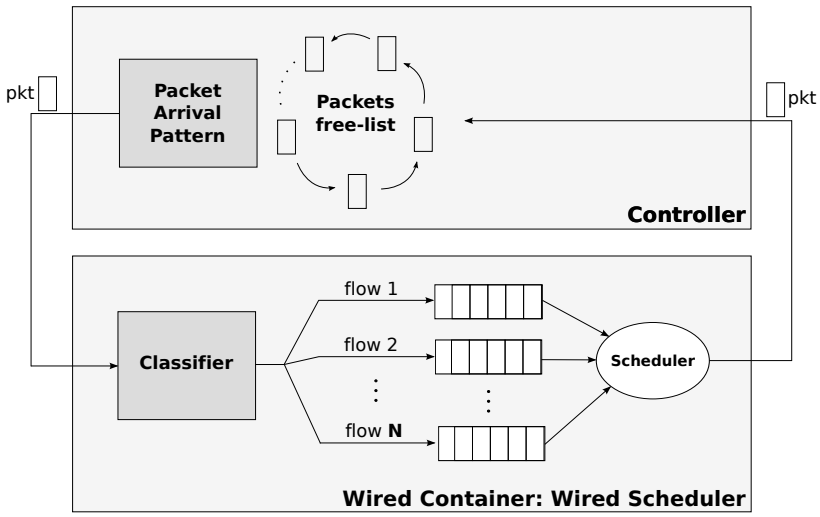


Figure 4.14: High-level representation of TEMPEST with wired scheduler.

- The possibility to generate realistic traffic for a programmable number of flows, packet size and weight distribution, leading to a flexible environmental setup;
- The ability to control the operating point of the scheduler during tests;
- The data structures are the same, moving from kernel to TEMPEST and vice-versa is trivial.

In the rest of the section we present how TEMPEST works, starting from his building blocks.

4.3.2.1 TEMPEST blocks

In order to understand how TEMPEST works, we present the three main blocks that manage the simulation:

1. **Controller.** The controller is the block which simulates the desired realistic packet arrival pattern, deciding in what order generate and serve packets to the packet scheduler in exam. It also manages a free list of packets. The goal of this list to reduce the overhead of memory-consumption operations like creation and reception of packets, in order to increase the accuracy of the execution time measurement;

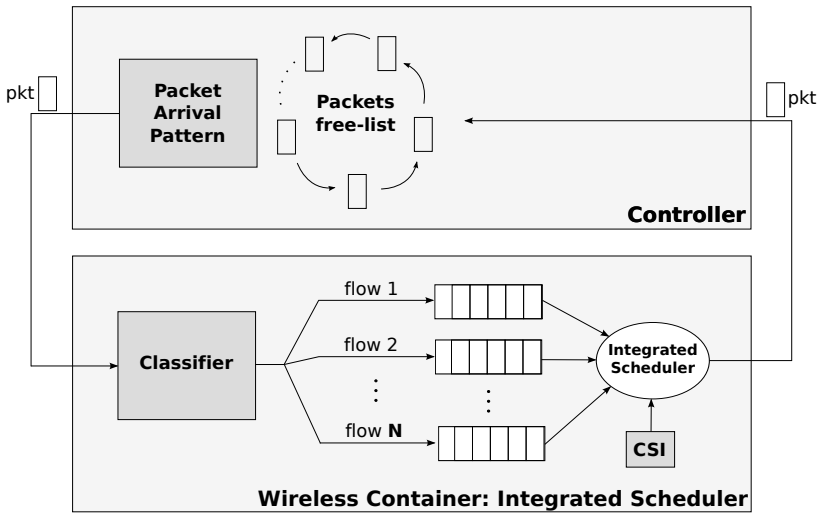


Figure 4.15: High-level representation of TEMPEST with wireless *integrated* scheduler.

2. **Scheduler Container.** The scheduler container is a featherweight model created to encapsulate a packet scheduler and reduce the overhead of all the surrounding operation in order to be able to measure only the execution time of the packet scheduler operations. It tracks the necessary information to compute the QoS parameters and the throughput. The container is also organized to encapsulate the kernel data structure in user space; in this way, plugging a packet scheduler inside the container is extremely flexible and at most some little interface changes are needed;

3. **Channel State Information.** It is the simulation engine behind the wireless scenario representation. It is an optional block used only when a wireless scenario is needed instead of a wired one. It gives per flow status, like Packet Loss Probability (P_{loss}) or Signal to Noise Ratio (SNR or S/N), to the packet scheduler in order to implement smarter scheduling disciplines.

In Figure 4.14 is depicted the TEMPEST environment for a wired scheduler testing. In the figure, only the Controller and the Scheduler Container blocks are depicted. At the same time, Figure 4.15 shows the environment for a standard wireless test, in this scenario the packet scheduler uses also Channel State Information to determine which packet to send next.

4.3.2.2 Controller Behavior

The controller exercises the packet scheduler with the desired sequence of enqueue/dequeue requests, iteratively switching between two phases: an enqueue phase and a dequeue phase. The switch occurs according to two configurable max-total-backlog and min-total-backlog thresholds. The max-total-backlog is the maximum amount of pending packets stored in scheduler flows, while the min-total-backlog is the minimum amount of pending packets stored in the scheduler flows. When the controller is in enqueue phase, it generates fake packets by picking them from a free list while, when the controller is in dequeue phase, it dequeues packets from the scheduler and reinserts them into the free list.

The controller can also implement realistic behaviour of packet arrival patterns to exercise the scheduler. Packet arrival patterns can be configured in different ways:

- *Smooth* pattern: the controller, in the enqueue phase, iteratively generates one packet for each flow with a lower backlog than the other flows (filling flows in a round-robin fashion);
- *Bursty* pattern: the controller, in the enqueue phase, iteratively generates a burst of packets for each flow with a lower backlog than the other flows, where the burst has a random size proportional to the flow weight.

In addition to the above rules for switching between the dequeue and enqueue phase, if the backlog of a flow drops below a configurable threshold while the controller is in dequeue mode (e.g. one packet), then the controller may switch back, with a configurable probability, to the enqueue phase, filling again the queues of this and of the other flows. This additional mechanism is used to increase the randomness of the arrival pattern, and hence to make harder for a packet scheduler to guarantee a tight packet delay.

4.3.2.3 Scheduler Container

The scheduler container not only contains the kernel code of packet scheduler to test, but also the general characteristics of the environment. Three possible containers can be used:

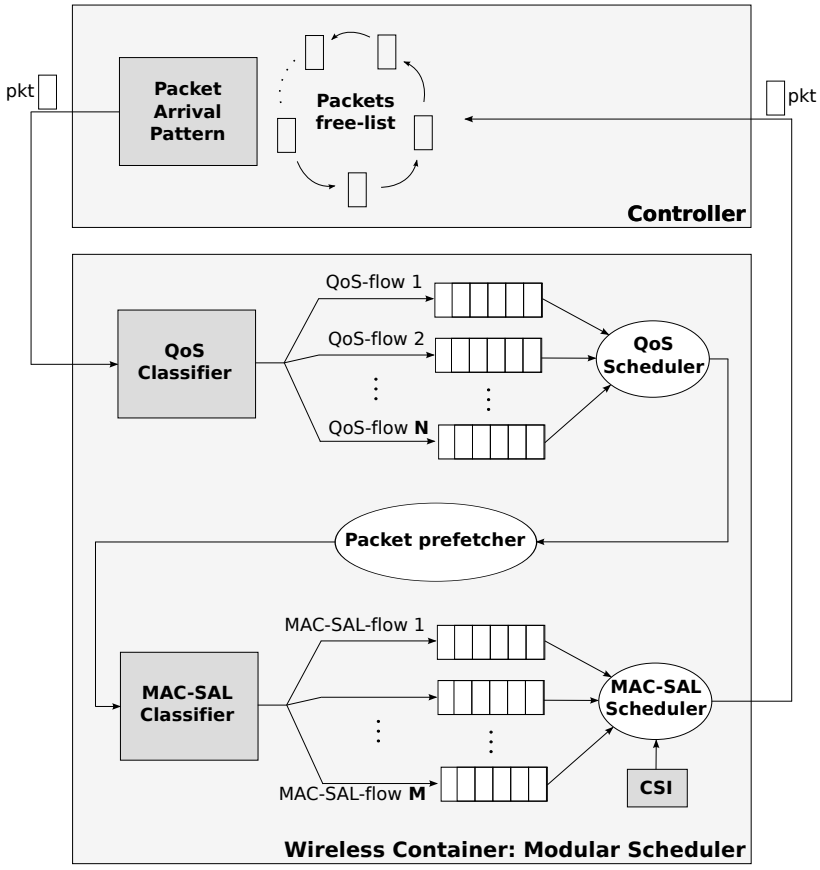


Figure 4.16: High-level representation of TEMPEST with wireless modular scheduler.

- **Wired Container**: this represents the wired architecture of a packet scheduler and is depicted in Figure 4.14. In this container one packet scheduler will decide the order in which packets are sent over the link, without considering channel conditions;

- **Wireless Container with Integrated Scheduler**: this represents the typical and state-of-the-art wireless architecture of a packet scheduler and is depicted in Figure 4.15. In this container, just one *integrated* scheduler takes all the scheduling decisions, based on the channel conditions (provided by the CSI module) and on the desired QoS;

- **Wireless Container with Modular Scheduler**: this represents a novel and modular wireless architecture of a packet scheduler. The container is depicted in Figure 4.16. As a brief description, a cascade of two packet

schedulers decides the order in which packets are sent. In this solution one scheduler takes care of QoS guarantees and another one takes care of throughput-boosting or energy-saving issues. For example, the first scheduler may deal with QoS guarantees at IP level, and the second scheduler may be placed just on top of the MAC layer and use Channel State Information (CSI) to boost the throughput. For brevity, hereafter we will call QoS scheduler the top-level scheduler, and MAC-SAL scheduler (MAC Scheduling and Abstraction Layer) the bottom-level scheduler.

Both wireless containers use CSI to simulate the idiosyncrasies of a wireless link and stress the packet scheduler in order to optimize the scheduling decision, taking into account also the channel condition. When a wireless container is used, TEMPEST calculates also the throughput achieved by the scheduler (execution time and QoS guarantees are measured as well).

4.3.2.4 Packet Schedulers

TEMPEST contains many accurate and efficient state-of-the-art packet schedulers, in the following are described some of them with their main characteristics:

- DRR [101]: a packet scheduler with extremely low time complexity $O(1)$, but with $O(n)$ deviation from optimal service;
- WF^2Q+ [100]: a *timestamp-based* algorithm with optimal service guarantees in $O(logn)$ time;
- KPS [130]: an approximated variant of timestamp-based schedulers with near-optimal guarantees and $O(1)$ time complexity (but several times slower than DRR);
- QFQ [131]: an approximated variant of timestamp-based schedulers with near-optimal guarantees and $O(1)$ time complexity almost fast as DRR;
- QFQ+ [99]: an improved version of QFQ with quasi-optimal service guarantees and execution time still closer to the DRR one;
- W^2F^2Q [108]: the best integrated packet scheduler for wireless link based on WF^2Q+ algorithm;

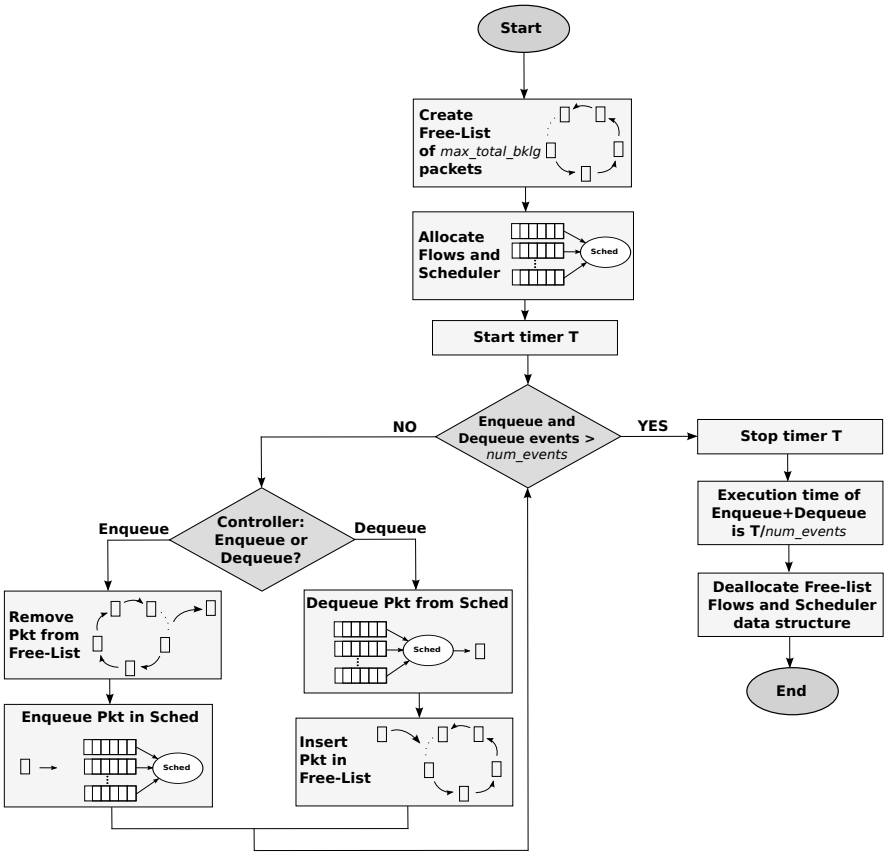


Figure 4.17: Flowchart of TEMPEST: execution time measurement.

- HFS [91]: a new accurate and modular packet scheduler for wireless link, based on QFQ+, with $O(1)$ time complexity, quasi-optimal service guarantees, high throughput and low energy consumption.

4.3.2.5 Performance Metrics

TEMPEST measures different well-known metrics in order to validate packet schedulers performance. We used all of these indicators to easily design and validate different packet scheduler solutions. Here is a list of the main performance metrics and a detailed description:

- **Throughput:** to measure the throughput, TEMPEST calculates the *normalized throughput* achieved by the schedulers. This parameter is the

amount of successfully transmitted packets for each flow divided by the amount of total sent packets. It is computed as

$$\text{thr} \equiv \frac{\sum_i pkt_{sent_i} \cdot (1 - \text{P}_{loss_i})}{\sum_i pkt_{sent_i}}$$

where pkt_{sent_i} is the number of packets sent by the flow i, P_{loss_i} is the packet loss probability of the flow i and $pkt_{sent_i} \cdot (1 - \text{P}_{loss_i})$ is the number of successfully transmitted packets by the flow i;

- **Execution time:** how to measure the execution time? The actual execution time measurement is a key point of TEMPEST and Figure 4.17 describes the flowchart of the tool for this purpose[5]. At the end of each run, TEMPEST measures the total execution time of the run (without considering the initial/final memory allocation/deallocation of the environment, which are not significant measurements for this purpose) and divide it by the total number of enqueues, or equivalently of dequeues, executed. It obtains therefore the average total packet-processing time, i.e., the execution time of an enqueue plus a dequeue, inclusive also of the cost of generating and discarding (from the free list) an empty and fixed-size packet;

- **Energy consumption:** according to the models in [127,128], lower/higher relative execution times imply also lower/higher relative energy consumptions. Therefore, the actual execution time calculated by TEMPEST should be considered as a key factor in order to evaluate a packet scheduling solution from a power consumption point of view. This is an important information for certain environments like Wireless Sensors Networks or Data Centers;

- **Queueing delay:** the queueing delay for each flow is measured as the maximum amount of time experienced by any packet inside the flow, expressed in number of dequeue events. The delay experienced by a packet pkt has been calculated as $t_{deq} - t_{enq}$ where t_{enq} is the number of packets dequeued by the system when pkt is enqueued, while t_{deq} is the number of packets dequeued by the system when pkt is dequeued;

[5] In Figure 4.17 compare *num_events* and *max_total_bklg* described in details in Section 4.3.2.6.

- **Time Worst-case Fair Index (T-WFI):** another useful QoS metric is the T-WFI (described in [99]), which allows to evaluate, in a single value, both fairness and delay: on one hand it shows the fairness, in terms of delay from the worst-case completion time of a packet in a perfectly fair system, while on the other hand it allows to instantly calculate the actual delays incurred by packets depending on the occupation of queues. In a perfectly fair system, the worst-case completion time of a packet is equal to its queue length (including the packet itself) divided by the packet flow guaranteed ratio. Assuming that the link rate R is constant, we measured T-WFI^k for flow k as:

$$\text{T-WFI}^k \equiv \max\left(t_{deq} - t_{enq} - \frac{Q^k(t_{enq})}{\phi^k R}\right)$$

  where t_{enq} and t_{deq} are the same of the queueing delay parameter, $Q^k(t_{enq})$ is the backlog of flow k just after the arrival of the packet and ϕ^k is the relative weight of the flow;

- **Bit Worst-case Fair Index (B-WFI):** this metric indicates how much, in terms of service received, a flow remains behind what it would receive on an ideal scheduler (and it is also described in [99]). We measured B-WFI^k for flow k as:

$$\text{B-WFI}^k \equiv \max_{[t_1,t_2]} \{\phi^k W(t_1,t_2) - W^k(t_1,t_2)\}$$

  where $[t_1, t_2]$ is any time interval during which the flow is continuously backlogged, $\phi^k W(t_1, t_2)$ is the minimum amount of service the flow should have received according to its share of the link bandwidth, and $W^k(t_1, t_2)$ is the actual amount of service provided by the scheduler to the flow. In particular, $W(t_1, t_2)$ is measured as the number of dequeue events computed by the scheduler between t_1 and t_2, while $W^k(t_1, t_2)$ is measured as the number of dequeue events computed by the scheduler only for the flow k;

- **Relative Fairness Index (RFI):** the Relative Fairness Index is defined as the maximum difference, over any time interval $[t_1, t_2]$ and pair of flows k and h, between the normalized service given to two continuously backlogged flows:

$$\mathrm{RFI}^k \equiv \max_{\forall k,h,[t_1,t_2]} \left| \frac{W^h(t_1,t_2)}{\phi^h} - \frac{W^k(t_1,t_2)}{\phi^k} \right|$$

where $W^k(t_1,t_2)$ and $W^h(t_1,t_2)$ are the actual amount of service provided by the scheduler to the flows k and h, while ϕ^k and ϕ^h are the relative weights of flows k and h. In particular, $W^k(t_1,t_2)$ and $W^h(t_1,t_2)$ are computed as in the B-WFI for flows k and h.

4.3.2.6 Setup and Input Parameters

This subsection has the goal to describe how to setup the entire tool and which are the parameters involved in the configuration. Such a fine-grained description is needed in order to understand and reproduce the experiments performed in Section 4.3.3. A high-level overview of these parameters is:

./`test` `-n` *num_events* `-qmin` *min_total_bklg* `-qmax` *max_total_bklg*
[`-len` *length_of_pkt*]
[`-burst` *number_of_burst_pkts*]
[`-qsing` *number_pkts*]
`-mode` { *single* | *integrated* | *double* }
{ `-alg` *sched_alg* | `-qos` *sched_alg*1 `-mac` *sched_alg*2 }
[`-flows` *num_flows* |
`-flowsets` [*weight*]::*num_flows*[, [*weight*]::*num_flows*] . . .]
[`-flowsmac` *num_mac_flows* |
`-flowsetsmac` [*weight*]::*num_flows*[, [*weight*]::*num_flows*] . . .]
[`-qmac` *number_of_pkts*]
[`-wdistr` *id_number*]
[`-ploss` $ploss_1$:$ploss_2$: . . . :$ploss_{num\_mac\_flows}$]
[`-intgr_th` *weight_th*]

where { `A` | `B` } is a choice between `A` and `B`, and [`-param` *value*] is an optional parameter. Here we present and describe in details the meaning of all of these parameters dividing them in high-level groups to easily understand how to combine them:

Controller: the controller is configured by the first parameters:

- `-n` *num_events*, the program executes an amount of *num_events* packet enqueues/dequeues with a typical balancing of $\frac{num\_events}{2}$ enqueue events and $\frac{num\_events}{2}$ dequeue events;

- `-qmin` *min_total_bklg*, the number of pending packets inside the controller cannot drops below *min_total_bklg* packets;

- `-qmax` *max_total_bklg*, the number of pending packets inside the controller cannot rises above *max_total_bklg* packets. The same parameter is used to define the size of the free-list of packets shown in Figure 4.17. Both *min_total_bklg* and *max_total_bklg* could be followed by "n", in this case the value (*min_total_bklg* or *max_total_bklg*) will be multiplied by the number of flows of that run;

- `-len` *length_of_pkt*, the length of every packet will be *length_of_pkt* bytes. It is an optional parameter, the default value is 1700 bytes.

Packet arrival patterns: the packet arrival patterns can be configured with these parameters:

- `-burst` *number_of_burst_pkts*, if *number_of_burst_pkts* is equal to zero, then the packet arrival pattern will be smooth; otherwise, if *number_of_burst_pkts* is greater than zero, then the packet arrival pattern will be bursty and the higher the value of *number_of_burst_pkts*, the higher the burst size in proportion to the flow weight. The default value is zero, which means that the default pattern is the smooth one;

- `-qsing` *number_pkts*, if this parameter is enabled, a single threshold per flow is forced; if a flow drops below *number_pkts*, then the controller refill that flow with a given probability. The default value is zero and does not affect the packet arrival patterns behavior.

Scheduler Container: TEMPEST can be configured to simulate wired or wireless scenarios with these parameters:

- `-mode` { *single* | *integrated* | *double* }, the program can be run in *single*, *integrated* or *double* mode; in *single* and *integrated* mode only one packet scheduler will decide the order in which packets are sent, like respectively in Figure 4.14 and Figure 4.15, while in *double* mode a cascade of two packet

schedulers decides the order in which packets are sent, one scheduler takes care of QoS guarantees and the other one takes care of throughput-boosting or energy-saving issues, as depicted in Figure 4.16. The *integrated* mode and the *double* mode are used to simulate wireless scenarios;

- `-alg` *sched_alg*, this parameter is used to set the *sched_alg* packet scheduler to test in *single* and *integrated* mode;

- `-qos` *sched_alg*1 `-mac` *sched_alg*2, these parameters are used to set the two packet schedulers in *double* mode, the scheduler *sched_alg*1 will be placed at the IP layer and takes care of QoS guarantees, while the scheduler *sched_alg*2 will be placed at the MAC-SAL layer and will use channel state informations to boost the throughput and save energy;

- `-flowsets` [*weight*]::*num_flows*[, [*weight*]::*num_flows*], a flowset constituted by *num_flows* number of flows of the same weight *weight* is created, different flowsets can be created at the same time with a comma separated pattern. The value *weight* can be omitted, in that case the default weight is 1. This parameter is used to configure the flowsets in *single* and *integrated* mode but also to configure the QoS flowsets in the *double* mode;

- `-flows` *num_flows*, can be used instead of the previous parameter to set only the number *num_flows* of flows with the default weight of 1;

- `-flowsetsmac` [*weight*]::*num_flows*[, [*weight*]::*num_flows*], this parameter is equivalent to the `-flowsets` one and is used, in *double* mode, to configure the flowsets at the MAC-SAL layer;

- `-flowsmac` *num_mac_flows*, this is equivalent to the `-flows` parameter, it is used to configure the number of flows at the MAC-SAL layer if *double* mode is chosen, or simply the number of subscriber stations in *integrated* mode.

Wireless Environment: the wireless environment can be configured with these parameters:

- `-qmac` *number_of_pkts*, this parameter is used to set the MAC-SAL Q shared-buffer size in *double* mode;

- `-wdistr` *id_number*, this implements different weight distribution policies among MAC-SAL flows based on the channel state information. Different values define different policies:
  - *id_number* = 0: it leads to a constant weight distribution, every MAC-SAL flow will be initialized with a weight equal to 1, thus, without considering CSI values;
  - *id_number* = 1: it leads to a linear weight distribution. The *num_mac_flows* flows $f_0, \ldots, f_i, \ldots, f_{num\_mac\_flows}$ are ordered from the higher losses flow to the lower losses flow. To each MAC-SAL flow is assigned a weight equal to i, where i is the position of the flow in the ordered list;
  - *id_number* = 2: it leads to an exponential weight distribution, considering the same example of the previous distribution, the weight of a generic flow is initialized with 2^i where (again) i is the position of the flow in the ordered list;
  - *id_number* = 3: it leads to an analogical weight distribution where a flow k is initialized with a weight equal to $(1 - P_{loss_k}) \cdot 1000$.
- `-ploss` $ploss_1$:$ploss_2$:...:$ploss_{num\_mac\_flows}$, this parameter is used in *integrated* mode to assign different packet loss probabilities for each MAC flow (or subscriber station);
- `-intgr_th` *weight_th* is used to simulate the typical *integrated* model where a two-state Markov chain organizes flows in *good* or *bad* state. Such a distinction can be made by using a threshold. Accordingly to the *weight_th* value a MAC-SAL flow (or a subscriber station) with weight lower or equal to *weight_th* will be placed in the *bad* flows group, while a MAC-SAL flow (or a subscriber station) with weight greater than *weight_th* will be placed in the *good* flows group.

4.3.3 Simulation Results

In this section we present examples of results obtained using TEMPEST. To do this, we will show some extensive simulations used to enhance the tool performance.

parameter	**command**	**value**
number of events	-num_event	10M
min pending packets	-q_min	0
max pending packets	-q_max	30n
four possible flow sets	-flowsets	1::1k 1::500,2::250,8::125 1::32k 1::16k,2::8k,8::4k
smooth packet arrival pattern	-burst	0
scheduler container	-mode	single
packet scheduler	-alg	FIFO DRR WF^{2}Q+ QFQ QFQ+
default packet length in Bytes	-len	1700

Table 4.1: Configuration parameters for the simulation in Figure 4.18.

TEMPEST has been recently used to demonstrate the high efficiency in terms of execution time of QFQ+ in [99] and to show how HFS [91] is able to guarantee high QoS while boosting throughput and save energy over wireless links. In both cases the flexibility of the tool has played an important role, in fact it permits to have quick feedback in terms of performance with a fine-grained tuning level of the configuration parameters. We present different simulations to validate the measurements of TEMPEST in terms of execution time, QoS performance and throughput. All the results presented in this section can be reproduced using the scripts available in the TEMPEST package [107]. The test equipment for these simulations has been a workstation with the following characteristics: OS X 10.7.4, Darwin 11.4.0, Intel Core i5-2557M @ 1.7 GHz, gcc 4.2.1 -O3.

4.3.3.1 Execution time

One of the key features of TEMPEST is the ability to measure the actual execution time of a packet scheduler under different packet arrival patterns. This is the case of Figure 4.18 and Figure 4.19 in which QFQ, QFQ+, DRR, WF^{2}Q+ and FIFO are compared under different flow sets with two different packet arrival patterns. Figure 4.18 has smooth arrival pattern configuration, while Figure 4.19 has bursty arrival pattern configuration, with burst weight equal to two packets, as reported in the parameters list of Table 4.1 for Figure 4.18, and of Table 4.2 for Figure 4.19. Both figures represent on the ordinate the time of an enqueue

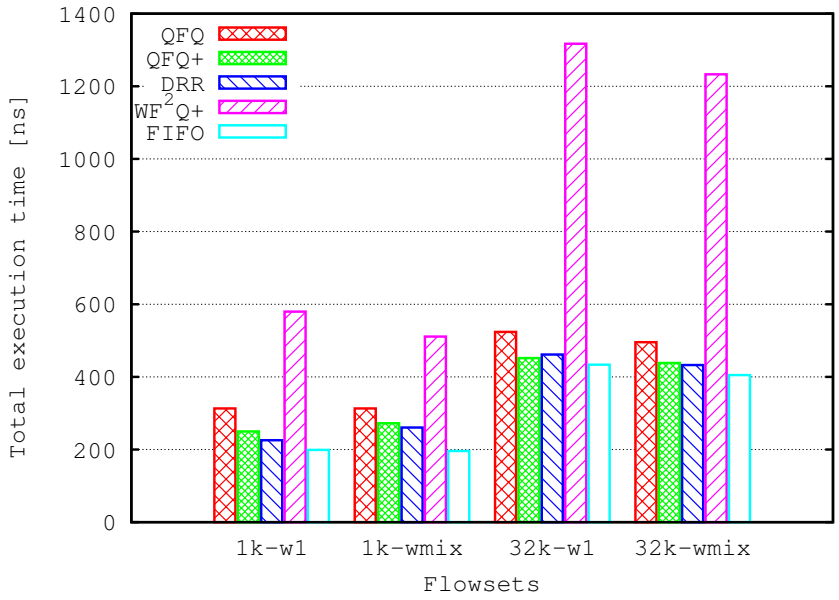


Figure 4.18: Execution time for different schedulers with *smooth* packets arrival pattern (lower is better).

parameter	**command**	**value**
number of events	-num_event	10M
min pending packets	-q_min	0
max pending packets	-q_max	30n
four possible flow sets	-flowsets	1::1k 1::500,2::250,8::125 1::32k 1::16k,2::8k,8::4k
bursty packet arrival pattern	-burst	2
scheduler container	-mode	single
packet scheduler	-alg	FIFO DRR WF^2Q+ QFQ QFQ+
default packet length in Bytes	-len	1700

Table 4.2: Configuration parameters for the simulation in Figure 4.19.

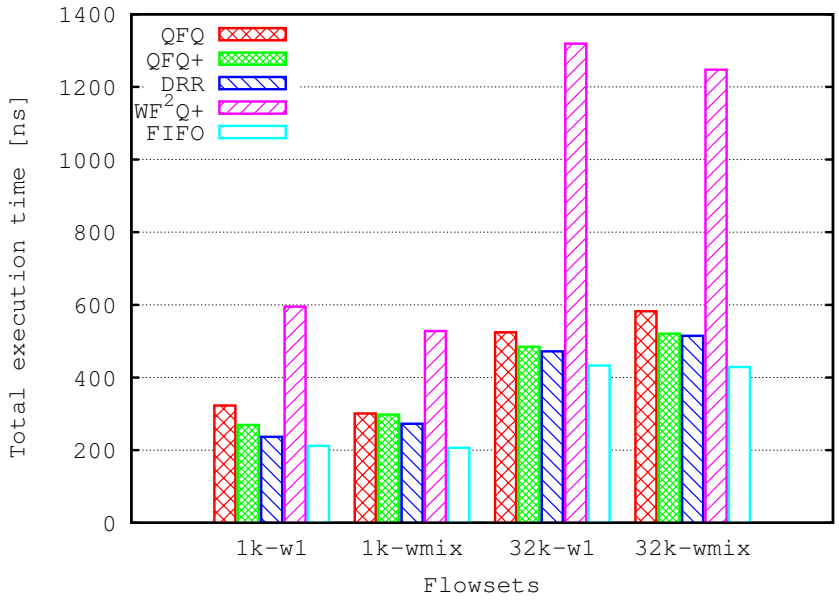


Figure 4.19: Execution time for different schedulers with *bursty* packets arrival pattern (lower is better).

parameter	**command**	**value**
number of events	-num_event	10M
min pending packets	-q_min	4n
max pending packets	-q_max	30n
ten possible flow sets	-flowsets	from 1::1 to 1::30000
bursty packet arrival pattern	-burst	1
scheduler container	-mode	single
packet scheduler	-alg	FIFO DRR WF^{2}Q+ KPS QFQ+
default packet length in Bytes	-len	1700

Table 4.3: Configuration parameters for the simulation in Figure 4.20.

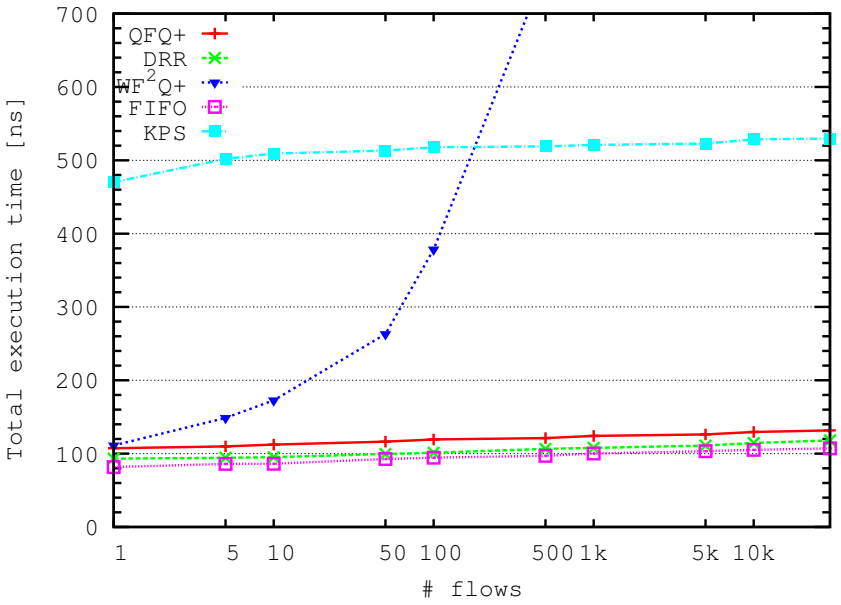


Figure 4.20: Execution time behavior for different schedulers (lower is better).

event plus a dequeue event expressed in nanoseconds, while on the abscissa there are the following four flow sets:

- 1k-w1: 1k flows all with weight 1;
- 1k-wmix : 500 + 250 + 125 flows with weights 1, 2, 8;
- 32k-w1: 32k flows all with weight 1;
- 32k-wmix : 16k + 8k + 4k flows with weights 1, 2, 8.

For this first example, TEMPEST has been used to prove the efficiency in terms of execution time of QFQ+ showing time complexity close to DRR and even better in particular scenarios, like flow set configuration 32k-w1 of Figure 4.18, due to the smaller number of cache missing of QFQ+ with respect to DRR[6].

Furthermore, TEMPEST can be used to easily evaluate the execution-time trend with respect to the number of flows, as a fundamental figure of merit for a packet scheduler: its time complexity. Theory can tell us about asymptotic

[6]A deep analysis of cache missing is also reported in [99].

parameter	command	value
number of events	-num_event	5M
min pending packets	-q_min	15n
max pending packets	-q_max	25n
seven possible flow sets	-flowsets	from 1::100,1::1 to 1::100,1000::1
bursty packet arrival pattern	-burst	2
threshold per flow	-qsing	1
scheduler container	-mode	single
packet scheduler	-alg	QFQ DRR WF^{2}Q+ QFQ+
default packet length in Bytes	-len	1700

Table 4.4: Configuration parameters for the simulations in Figure 4.21 and Figure 4.22.

complexity, but only effective measurements can determine the run-time constants. This is the case of Figure 4.20, obtained by the simulation synthesized in Table 4.3. From the plot, it immediately results that FIFO, DRR, KPS and QFQ+ have a constant time complexity while WF^{2}Q+ grow has a function of the number of flows, in fact it actually has a $O(logn)$ time complexity. TEMPEST can uncover the high run-time constant of a scheduler, in this case of KPS. This characteristic is important because, even if KPS has an $O(1)$ complexity instead of the $O(logn)$ of WF^{2}Q+, due to the multiplicative constants WF^{2}Q+ results faster in environments with less than few hundreds of flows, which is the case of lots of real scenarios like domestic networks and small laboratory networks.

4.3.3.2 QoS guarantees

Another fundamental figure of merit for a packet scheduler is its QoS level. From the theory, we know about scheduler service bounds like B-WFI and T-WFI. Also in this case, TEMPEST provides real measurements of this parameters, evaluating a packet scheduler by considering his run-time guarantees. As an example, Figure 4.21 reports the measured B-WFI, expressed in packets, for the schedulers WF^{2}Q+, DRR, QFQ and QFQ+. The experiments have been carried out using 101 flows, 100 flows with weight 1 and the remaining flow with variable weights in the set {1, 5, 10, 50, 100, 500, 1000}. The weight value of this variable flow is reported on the abscissa. This has been done in order to have a flow with different bandwidth requirements, stressing the packet

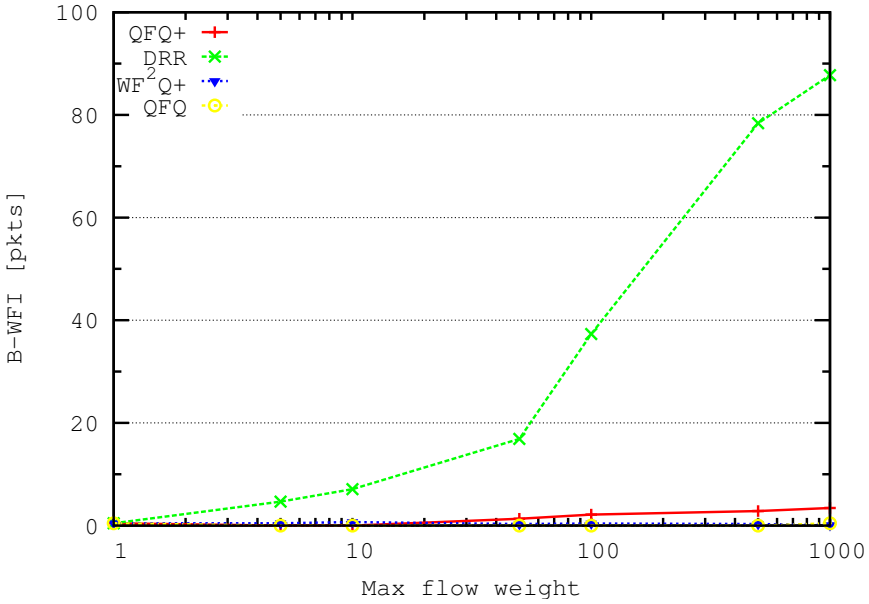


Figure 4.21: B-WFI value for different schedulers (lower is better).

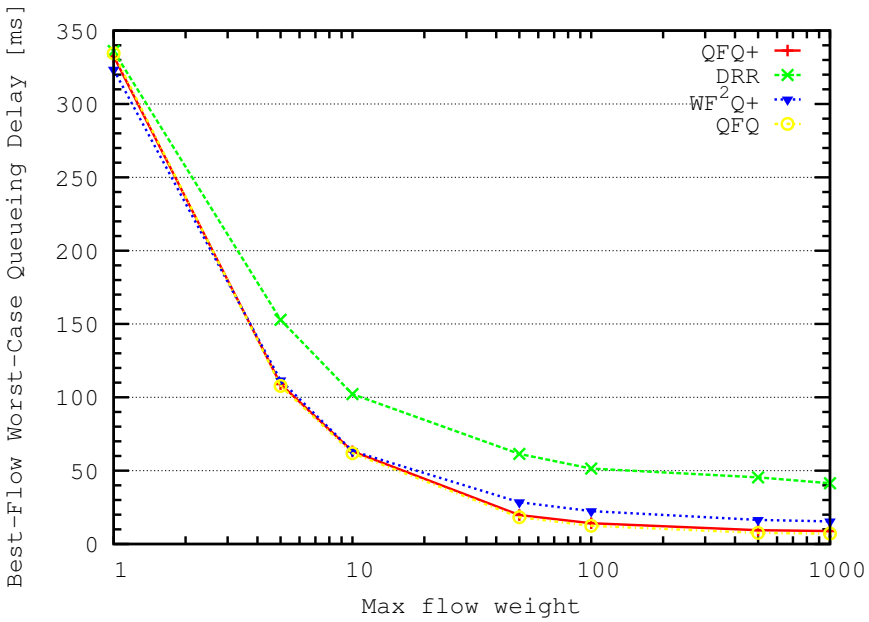


Figure 4.22: Worst-case queueing delay of maximum priority flow for different schedulers (lower is better).

parameter	command	value
number of events	-num_event	1M
min pending packets	-q_min	0
max pending packets	-q_max	300
flow set	-flowsets	20::20,10::20,1::60
bursty packet arrival pattern	-burst	1
two scheduler container	-mode	single double
packet scheduler	-alg	DRR HFS WF^{2}Q+
buffer size Q in pkts	-qmac	from 1 to 250
number of MAC flows (SS)	-flowsmac	20
channel bandwidth (Mb/s)	default	54
packet loss per MAC flow	default	from 10^0 to 10^{-4}
default packet length in Bytes	-len	1700

Table 4.5: Configuration parameters for the simulation in Figure 4.23.

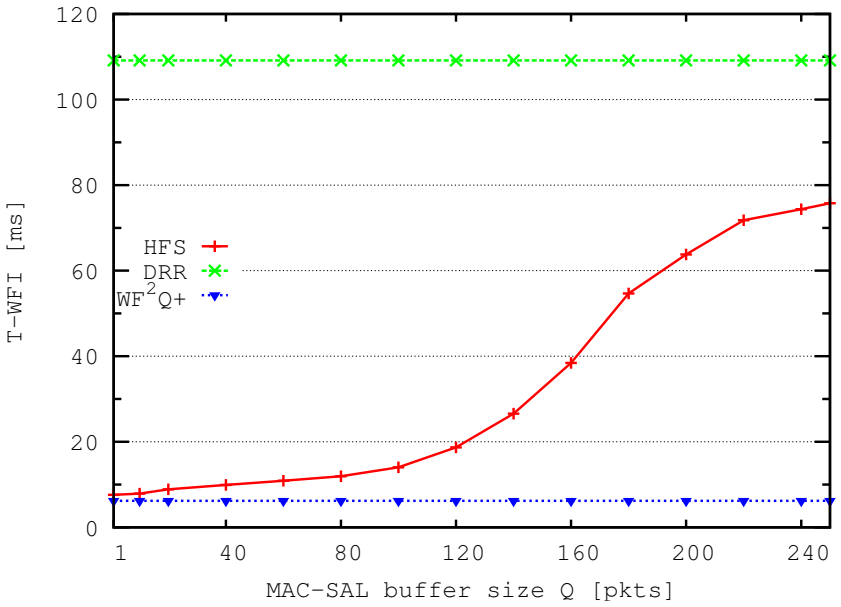


Figure 4.23: T-WFI for HFS, WF^{2}Q+ and DRR (lower is better).

scheduler in order to provide more and more skewed bandwidth distributions. TEMPEST provides measurements that are compliant with the literature. From the theory [99,143] we know that WF^{2}Q+, QFQ and QFQ+ have constant worst-case bound for the B-WFI (i.e. do not vary with the flow weights distribution). In particular, WF^{2}Q+ has an optimal theoretic B-WFI bound of 1 maximum segment size, which is 1 packet for Figure 4.21 (all the experiments have the same and constant packet size); QFQ has a theoretic B-WFI bound of circa 5 packets and QFQ+ has a theoretic B-WFI bound equivalent to QFQ plus a constant value depending on the number of flows in the aggregates[7]. From Figure 4.21 we can see how WF^{2}Q+, QFQ and QFQ+ have actual B-WFI, calculated through the experiments, which are lower than the theoretical ones; in particular, QFQ is really close to WF^{2}Q+ (both recorded B-WFI values between 0 and 0.5 packets), and QFQ+ is also close to the optimal value and under the theoretical bound of QFQ. Moreover, QoS guarantees of QFQ and QFQ+ are definitely lower than DRR. The latter has linear service deviation with respect to the number of flows and their weights distribution, as showed in [143], and the experiments highlight his high B-WFI (again, the measures are compliant with the theoretical bounds).

By performing the same experiment Figure 4.22 has been also generated. This time we report the worst-case queueing delay experienced by packets of the flow with highest weight. Such a measure is important to show how much a packet scheduler is able to guarantee a thin delay bound for high priority flows. QFQ and QFQ+ appear to be the most scalable solutions guaranteeing the lowest queueing delay while growing the flow weight. In particular, they also outperform WF^{2}Q+. DRR packet scheduler manifests always a higher queueing delay resulting also, for high flow weights (above 50) in a more than double delay bound.

Then, we performed several experiments for wireless environments, one of them is described in Table 4.5 and it has been conduced in order to validate HFS packet scheduler in terms of QoS guarantees. The experiment results are summarized in Figure 4.23 in terms of measured T-WFI. DRR and WF^{2}Q+ are wired packet schedulers and they do not consider the CSI at all as well as throughput or energy consumption optimizations, they do only care about the QoS flows which are constant during the experiment and described here:

- 20 flows of VoIP traffic with weight 20;

[7] More details on this parameter can be found in [99].

- 20 flows of web browsing with weight 10;
- 60 flows of background file download/sharing with weight 1.

These QoS flows are mapped into 20 MAC flows which represent 20 user sharing the same access point. Each user has one VoIP flow, one WEB browsing flow and three background file downloading/sharing flows. While DRR and WF^{2}Q+ consider only the QoS flows in order to guarantee the proper QoS level, HFS consider also Channel State Information coming from the 20 MAC flows (the 20 users) in order to maximize the throughput on the link and save energy by avoiding retransmissions. In this experiment we varied the size of the Q buffer, that is a core point of the modular architecture on which HFS is based. Figure 4.23 shows how HFS, with small buffer size Q, is able to guarantee QoS guarantees close to the optimal ones of WF^{2}Q+. If Q grows, the MAC-SAL module of HFS has more packets pending and it is able to reorder them with the goal of boosting throughput (as we will show in the next subsection) while still preserving lower QoS guarantees than DRR. The focus here is not on HFS performance but on how easily it is to configure TEMPEST in order to run extensive simulations to compare wired and wireless packet schedulers with fine-grained configurable parameters. Moreover, it is important to notice that also the T-WFI values computed by TEMPEST for WF^{2}Q+ and DRR are perfectly compliant with the worst case T-WFI bound of service guarantees collected in [143]. In fact, the T-WFI theoretical bound of WF^{2}Q+ is from the theory $2\frac{L^k}{\phi^k R}$ (plus a constant value that we will not consider for simplicity) where L^k is the packet size, ϕ^k is relative weight of the flow and R is the link bandwidth. Considering our simulations, the T-WFI bound for WF^{2}Q+ for flows with best weight will be

$$2 * \frac{1700 \cdot 8}{\frac{20}{660} \cdot 54 \cdot 10^6} = 17ms$$

which is higher than the circa 8 ms calculated by TEMPEST (660 in the formula is the weight sum, $20 \cdot 20 + 20 \cdot 10 + 60 \cdot 1$). The same applies for DRR: from the theory the worst-case T-WFI bound is approximately $(\frac{1}{\phi_{min}} + \frac{1}{\phi^k} + N - 1)\frac{L^k}{R}$, where ϕ_{min} is the minimum weight flow while N is the number of flows. In our experiment such a bound will be

$$(\frac{1}{\frac{1}{660}} + \frac{1}{\frac{20}{660}} + 100 - 1)\frac{1700 \cdot 8}{54 \cdot 10^6} = 199ms$$

parameter	command	value
number of events	-num_event	10M
min pending packets	-q_min	0
max pending packets	-q_max	300
flow set	-flowsets	20::20,10::20,1::60
bursty packet arrival pattern	-burst	1
two scheduler container	-mode	integrated double
packet scheduler	-alg	HFS W^2F^2Q+
buffer size Q in pkts	-qmac	from 1 to 250
number of MAC flows (SS)	-flowsmac	20
integrated threshold	-intgr_th	80
channel bandwidth (Mb/s)	default	54
packet loss per MAC flow	default	from 10^0 to 10^{-4}
default packet length in Bytes	-len	1700

Table 4.6: Configuration parameters for the simulation in Figure 4.24.

which is higher than the circa 110ms calculated by TEMPEST.

4.3.3.3 Throughput

The last figure of merit that we describe is throughput, another parameter that TEMPEST is able to compute. To show it, we considered the simulation summarized in Table 4.6 in which two schedulers, HFS based on the modular architecture and the integrated scheduler W^2F^2Q, are tested to measure their achieved throughput, assessing their behaviour in maximizing the amount of packets correctly sent over the wireless link by also considering CSI per flow. W^2F^2Q do not consider the MAC-SAL buffer size Q as a parameter, and his throughput level depends only on the CSI and the threshold between good-state flows and bad-state flows. In the experiment such a threshold has been placed at 80%, this means that only flows with 80% of chance to effectively transmit (without loss) the packets are considered in the scheduling decision, while other flows will not receive any service. In fact, in Figure 4.24 W^2F^2Q reaches a value close to 0.9, which means that 90% of packets have been successfully transmitted on the link. A different evaluation should be performed for HFS, which decides the next packet to transmit accordingly to the MAC-SAL scheduler, considering all the flows in the scheduling decision. When the MAC-SAL buffer size Q grows,

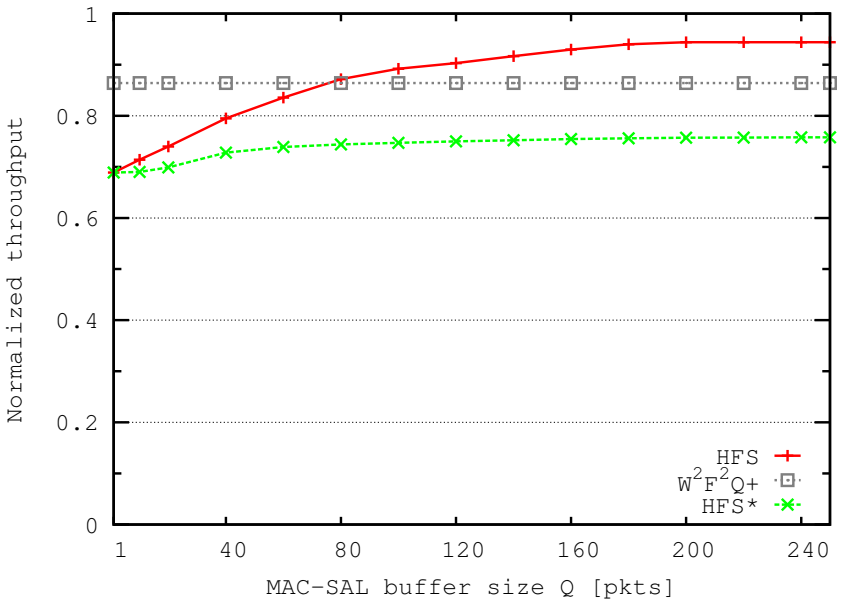


Figure 4.24: Normalized throughput for HFS and W^2F^2Q.

HFS has more packets among which to choose the next to transmit and his fine-grained decision among all flows let it reach throughput values over 90% when Q buffer is higher than 100 packets. Of course, to complete the picture, increasing the Q buffer size improves on one side the global throughput, while on the other causes a QoS degradation. To enhance TEMPEST flexibility, a dummy scheduler HFS* has been also deployed, based on the modular architecture but with the DRR scheduler placed at the MAC-SAL layer instead of QFQ+. In this way it is easy to evaluate a different solution. Unfortunately, the DRR packet scheduler is not able to boost the throughput due to his Round Robin serving technique; it basically also schedules flows with low probability of packet transmission causing a throughput degradation and increasing the number of retransmissions.

4.3.4 Conclusions

In this section, we presented TEMPEST, a new Test EnvironMent for Performance Evaluation of the Scheduling of packeTs, which is a novel UNIX-based open tool able to measure the actual performance of a packet scheduler under the desired operating conditions. With its fine-grained level of configuration parameters TEMPEST is able to help research in evaluating existing schedulers

to properly choose the best one depending on the operational conditions (e.g environment conditions, amount of flows, channel bandwidth, queue size, the packet arrival pattern and so on); in fact, TEMPEST has been actually used in packet-scheduling research field with QFQ+ and HFS. Therefore, the tool measures the main figures of merit in order to evaluate a packet scheduler such as: the execution time, QoS guarantees with standard fairness indexes (Queueing Delay, bit worst-case fair index B-WFI , time worst-case fair index T-WFI and relative fairness index RFI) and throughput.

A key point of TEMPEST is that it permits to measure the kernel code in user space, reducing the surrounding operations involved in the packet scheduling tasks, and thus allowing TEMPEST to be a flexible and portable tool for packet scheduling testing, able to generate traffic at 40Mpps and more on a common workstation. Furthermore, adding a new scheduler to TEMPEST is simple, an important aspect in order to help the research in emerging areas like 4G/5G in which QoS is a key parameter and where designing and testing a new packet scheduler is challenging and time-consuming. Moreover, to validate the efficiency of the tool, we presented also extensive simulations comparing results with the asymptotic values and worst-case bounds coming from the literature. All these simulations can be reproduced on different machines by using the scripts available in the tool package.

Chapter 5

Conclusions

This thesis addressed the study of PPDR emergency networks and proposed several networking algorithms in order to improve service performance. PPDR environments are particularly challenging due to the harsh conditions and the high variety of technologies deployed, and are characterized by very restrict requirements. In fact, by addressing this topic, all the solutions provided could be easily extended to a general purpose network as well. All the algorithms proposed in this thesis are enriched with mathematical formulations and extensive packet level simulations in order to validate their proposal.

In the first Chapter we completely focused on the Emergency Network topic by defining and presenting the reference architecture. The integration between LTE as access network and Satellite as backhaul network has been defined as the reference architecture. On top of this architecture we started to improve networking performance by presenting Congestion Control Middleware Layer (C^2ML), a cooperative middleware that effectively optimizes performance in networks with high delays, such as satellite networks. Through numerical results it has been shown that C^2ML leads to higher throughput while improving intra-protocol fairness and inter-protocol friendliness regardless of queue sizes, thus reducing the overall latency and mitigating the Bufferbloat problem. Moreover, same results have been achieved even when the number of competing nodes increases, therefore confirming C^2ML as a practical and scalable system. Finally, the use of C^2ML exploits much more effectively the potential offered by satellite links, even for real-time applications. Coupled with C^2ML we also presented $DyBRA$, a centralized yet collaborative network resource manager for high delay links.

DyBRA is able to dynamically adapt to significant changes in the client nodes exploitable bandwidth, which can result for example from channel conditions variations that are due to the sudden onset of obstacles between a client receiver and a base station. By performing extensive packet level simulations we proved that *DyBRA* considerably improves the queue usage in the gateway, especially during time frames in which the client nodes TCPs congestion windows are in process to converge to a stable size. It also greatly improves the network efficiency, by allowing the set of client nodes to exploit the totality of the bottleneck link resources also in cases of bandwidth underutilization by some client hosts. As the final remark, we conclude by stating how *DyBRA* inverts the natural tendency to couple the increase of client nodes number with higher queues occupancy, effectively decreasing the gateway queue usage as more hosts are in place.

In the second Chapter we focused on the buffer management problem by defining AQM algorithms for PPDR systems and general networks as well. Starting from C^2ML, we proposed a novel AQM technique called QRM (Queue Rate Manager). QRM is a very efficient algorithm that coupled with C^2ML provide security protection to resource exhaustion attacks increasing the robustness of the cooperative middleware while preserving features like fairness and throughput exploitation. We then adapted QRM to create a general and smart no-drop AQM for emergency networks, as the solution was uncoupled from the middleware and studied for general PPDR satellite networks. Finally, this novel AQM called PINK (Proactive INjection into acK) has been investigated and analysed though mathematical formulations and numerical simulations. PINK works by imposing, in a transparent way, a maximum upper bound on the data rates for each client. This is done through updating the RCV.WND value in the acknowledgement segments, allowing PINK to supervise bandwidth utilization in order to efficiently exploit a bottleneck channel capacity, forcing fairness among different flows and at the same time maintaining a low queue occupancy on the gateway. Furthermore, these results are achieved without discarding a single packet. Another important aspect to remark is the algorithm scalability, as all the figures of merit that have been analysed reveal that PINK manifests a positive, or at least a constant, trend as a function of the network congestion level.

In the third and final Chapter the packet scheduling problem has been deeply studied. We initially provided both a description and a validation of a modu-

lar architecture that decouples the task of providing QoS guarantees from the task of dealing with the idiosyncrasies of a wireless link. This tasks separation leads to better configuration and flexibility properties of the scheduling in PPDR networks. Therefore, moving from a network to another a node simply needs to adjust a value, named Q, to go from high throughput requirements to high QoS requirements. Continuing, we defined HFS (High-throughput twin Fair Scheduler) a new flexible, efficient and accurate scheduler deployed on top of the modular architecture. To prove the effectiveness of HFS we showed, through experimental results, its high performance. HFS provides higher throughput with respect to Wireless Worst-case Fair Weighted Fair Queueing, the best integrated scheduler available so far. Furthermore, HFS provides low latency and accurate fairness, close to the optimal ones of Worst-case Weighted Fair Queueing. Last but not least, HFS guarantees execution times and energy consumptions that are close to those of a Deficit Round Robin. We also demonstrated the possibility to dynamically set, by changing the MAC-SAL buffer size Q, the desired trade-off between throughput-boosting level and granularity of service guarantees, a feature that allows HFS to be an adaptive packet scheduler able to tune its performance by following users and network requirements. Finally, we also highlighted a value for the Q parameter (equal to 100 packets) which permits network operators to expect a higher throughput than with the best integrated scheduler available so far, while being able in the same time to provide QoS guarantees that are close to the optimal ones. As final contribution we presented TEMPEST, a new Test EnvironMent for Performance Evaluation of the Scheduling of packeTs, which is a novel UNIX-based open tool able to measure the actual performance of a packet scheduler under the desired operating conditions. With its fine-grained level of configuration parameters TEMPEST is able to help research in evaluating existing schedulers to properly choose the best one depending on the operational conditions (e.g environment conditions, amount of flows, channel bandwidth, queue size, the packet arrival pattern and so on); in fact, TEMPEST has been actually used for validating HFS. Therefore, the tool measures the main figures of merit in order to evaluate a packet scheduler such as: the execution time, QoS guarantees with standard fairness indexes (Queueing Delay, bit worst-case fair index B-WFI , time worst-case fair index T-WFI and relative fairness index RFI) and throughput.

Bibliography

[1] "PPDR's Needs and Requirements," Project Deliverable D2.2, PPDR-TC, January 2014.

[2] D. Tewfik, F. D. Mike, T. Said, C. Alessio, T. George, A. Kiran, and F. Dino, "LTE for Public Safety Networks," *IEEE Communications Magazine*, February 2013.

[3] F. Ramon and O. Sallent, "LTE: The Technology Driver for Future Public Safety Communications," *IEEE Communications Magazine*, October 2013.

[4] "PPDR's Current and Future Scenarios," Project Deliverable D2.1, PPDR-TC, October 2013.

[5] R. T. F. and C. Sacchi, "Opportunistic radio access techniques for emergency communications: Preliminary analysis and results," in *IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, Rome, Oct, 2-5, 2012.

[6] "PPDR's Technological Gaps," Project Deliverable D2.3, PPDR-TC, January 2014.

[7] "Overview of 3GPP Release 12 V0.1.3," 3GPP, June 2014. [Online]. Available: http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-12_description_20140924.zip

[8] "The LTE Network Architecture," Alcatel-Lucent Whitepaper, Alcatel-Lucent, 2009. [Online]. Available: http://www.cse.unt.edu/~rdantu/FALL_2013_WIRELESS_NETWORKS/LTE_Alcatel_White_Paper.pdf

[9] "The European table of frequency allocations and applications in the frequency range 8.3 kHz to 3000 GHz (ECA table)," European Conference of Postal and Telecommunications Administrations (CEPT), Electronic Communications Committee (ECC), May 2014. [Online]. Available: http://www.grss-ieee.org/wp-content/uploads/2014/07/ECA_European_Table_of_Frequency_Allocations_May_2014.pdf

[10] "LTE for Utilities, Supporting Smart Grids," Ericsson Whitepaper, Ericsson AB, September 2013. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-lte-for-utilities.pdf

[11] "Realistic LTE Performance, From Peak Rate to Subscriber Experience," Motorola Whitepaper, Motorola, August 2009. [Online]. Available: http://www.motorolasolutions.com/web/Business/_Documents/static%20files/Realistic_LTE_Experience_White_Paper_FINAL.pdf

[12] N. Sihavong, "O3b Networks," O3b presentation at ITU meeting, O3b, 2009. [Online]. Available: http://www.itu.int/ITU-D/asp/CMS/Events/2009/PacMinForum/doc/PPT_Theme-2_O3bNetworks.pdf

[13] "SCRAM: Space Communication Rates at Multi-Gbps," Workshop on X-Ray Mission Architectural Concepts, L-3 Communications, December 2011. [Online]. Available: http://pcos.gsfc.nasa.gov/studies/xray/workshop/33_T.McIntyre_SCRAM_12-15-2011.pdf

[14] "Why Latency Matters to Mobile Backhaul," O3b Whitepaper, O3b Networks, Sofrecom, 2014. [Online]. Available: http://www.o3bnetworks.com/media/45606/o3b_latency_mobile%20backhaul_130417.pdf

[15] C. Caini and R. Firrincieli, "End-to-End TCP Enhancements Performance on Satellite Links," in *Proceedings of the 11th IEEE Symposium on Computers and Communications*, June 2006, pp. 1031–1036.

[16] A. Donner, J. Saleemi, and J. Mulero Chaves, "Tetra backhauling via satellite: Improving call setup times and saving bandwidth," *Journal of Computer Networks and Communications*, vol. 2014, 2014.

[17] V. Jacobson, "Congestion avoidance and control," *SIGCOMM '88 Symposium proceedings on Communications architectures and protocols*, pp. 314–329, 1988.

[18] F. Kronewitter, B. Ryu, Z. Zhang, and L. Ma, "Tcp accelerator for dvb-rcs satcom dynamic bandwidth environment with haipe," *Communications and Networks, Journal of*, vol. 13, no. 5, pp. 518–524, Oct 2011.

[19] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[20] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 4, pp. 397–413, 1993.

[21] S. Floyd, R. Gummadi, S. Shenker *et al.*, "Adaptive RED: An algorithm for increasing the robustness of REDs active queue management," *Preprint, available at http://www. icir. org/floyd/papers. html*, 2001.

[22] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[23] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "Pie: A lightweight control scheme to address the bufferbloat problem," in *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*. IEEE, 2013, pp. 148–155.

[24] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2002, pp. 239–248.

[25] H. Hirai, M. Oguchi, and S. Yamaguchi, "A proposal on cooperative transmission control middleware on a smartphone in a WLAN environment," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, Oct 2013, pp. 693–700.

[26] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[27] S. Floyd, "Highspeed TCP for Large Congestion Windows," RFC 3649 (Experimental), Internet Engineering Task Force, December 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3649.txt

[28] A. Kuzmanovic, E. W. Knightly, and R. Les Cottrell, "HSTCP-LP: A protocol for low-priority bulk data transfer in high-speed high-rtt networks," in *The Second Int. Workshop on Protocols for Fast Long-Distance Networks (February 2004)*, 2004.

[29] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," in *ACM SIGCOMM Computer Communication Review, vol. 33, no. 2, pp. 8391*, 2003.

[30] P. S. Akyldiz IF, Morabito G, "TCP-Peach: a new congestion control scheme for satellite IP networks," in *IEEE/ACM Transactions on Networking, 9(3):307321*, 2001.

[31] C. Caini and R. Firrincieli, "TCP Hybla: a tcp enhancement for heterogeneous networks," in *International Journal of Satellite Communications and Networking, vol. 22, no. 5, pp. 547566*, 2004.

[32] A. Kuzmanovic and E. W. Knightly, "TCP-LP: A distributed algorithm for low priority data transfer," in *in Proceedings of IEEE INFOCOM03, San Francisco, CA*, 2003.

[33] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for tcp," in *Communications Surveys and Tutorials, IEEE, vol. 12, no. 3, pp. 304-342*, 2010.

[34] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-delay Product Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 89–102, Aug. 2002. [Online]. Available: http://doi.acm.org/10.1145/964725.633035

[35] K. Ramakrishnan and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481 (Experimental), Internet Engineering Task Force, Jan. 1999, obsoleted by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc2481.txt

[36] B. P. W. S. H. Low, L. L. H. Andrew, "Understanding XCP: Equilibrium and fairness," in *In Proc. IEEE INFOCOM, volume 2, pages 10251036*, 2005.

[37] H. Jung, S.-g. Kim, H. Y. Yeom, S. Kang, and L. Libman, "Adaptive delay-based congestion control for high bandwidth-delay product networks," in *INFOCOM, 2011 Proceedings IEEE.* IEEE, 2011, pp. 2885–2893.

[38] L. T. Saswati Sarkar, "Fair Distributed Congestion Control in Multirate Multicast Networks," in *IEEE/ACM Transactions on Networking, Vol. 13, No. 1, February*, 2005.

[39] Y. N. Tarik Taleb, Nei Kato, "REFWA: An Efficient and Fair Congestion Control Scheme for LEO Satellite Networks," in *IEEE/ACM Transactions on Networking, Vol. 14, No. 5, October*, 2006.

[40] C. Roseti, M. Luglio, and F. Zampognaro, "Analysis and Performance Evaluation of a Burst-Based TCP for Satellite DVB RCS Links," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 3, pp. 911–921, June 2010.

[41] N. Celandroni and R. Secchi, "Suitability of dama and contention-based satellite access schemes for tcp traffic in mobile dvb-rcs," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 4, pp. 1836–1845, May 2009.

[42] J. Rajamaki, "The mobi project: Designing the future emergency service vehicle," *Vehicular Technology Magazine, IEEE*, vol. 8, no. 2, pp. 92–99, June 2013.

[43] "Digital Video Broadcasting (DVB); Interaction Channel for Satellite Distribution Systems," ETSI EN 301 790, April 2005.

[44] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," RFC 6928 (Experimental), Internet Engineering Task Force, Apr. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6928.txt

[45] W. Stevens, "RFC 2001: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," Jan. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2001.txt

[46] "Interoperable PEP (I-PEP) transport extensions and session frame-work for satellite communications," Air Interface Specification, October 2005.

[47] http://netlab.ing.unimo.it/sw/C2ML.zip, 2014.

[48] M. Casoni, C. A. Grazia, M. Klapez, and N. Patriciello, "A congestion control middleware layer with dynamic bandwidth management for satellite communications," *International Journal of Satellite Communications and Networking*, pp. n/a–n/a, 2015. [Online]. Available: http://dx.doi.org/10.1002/sat.1129

[49] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2000, pp. 1157–1165.

[50] C. Caini and R. Firrincieli, "Packet spreading techniques to avoid bursty traffic in long RTT TCP connections [satellite link applications]," in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 5. IEEE, 2004, pp. 2906–2910.

[51] M. Casoni, C. A. Grazia, M. Klapez, and N. Patriciello, "Implementation and Validation of TCP Options and Congestion Control Algorithms for ns-3," in *Proceedings of the 2015 Workshop on Ns-3*, ser. WNS3 '15. New York, NY, USA: ACM, 2015, pp. 112–119. [Online]. Available: http://doi.acm.org/10.1145/2756509.2756518

[52] M. Muhammad, M. Berioli, and T. de Cola, "A simulation study of network-coding-enhanced PEP for TCP flows in GEO satellite networks," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014.

[53] N. Mohammadizadeh and W. Zhuang, "Cooperation of heterogeneous wireless networks in end-to-end congestion control for QoS provisioning," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013.

[54] M. Casoni, C. Grazia, M. Klapez, and N. Patriciello, "Reducing latency in satellite emergency networks through a cooperative transmission control,"

in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec 2014, pp. 2850–2855.

[55] C. Deak, A. Drozdy, P. Szilagyi, Z. Vincze, and C. Vulkan, "TCP Performance Improvement in Mobile Networks with Coverage Problems," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013.

[56] G. Liu, H. Ji, Y. Li, X. Li, and Y. Wang, "Transmission control protocol performance enhancement for mobile broadband interactive satellite communication system: a cross-layer approach," *International Journal of Satellite Communications and Networking*, 2014.

[57] S. Kota, G. Giambene, and N. L. Candio, "Cross-layer approach for an air interface of geo satellite communication networks," *International Journal of Satellite Communications and Networking*, vol. 25, no. 5, pp. 481–499, 2007.

[58] B. Fu, Y. Xiao, H. Deng, and H. Zeng, "A Survey of Cross-Layer Designs in Wireless Networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 110–126, First 2014.

[59] S. Paris, C. Nita-Rotaru, F. Martignon, and A. Capone, "Cross-Layer Metrics for Reliable Routing in Wireless Mesh Networks," *Networking, IEEE/ACM Transactions on*, vol. 21, pp. 1003–1016, June 2013.

[60] N. Dukkipati, N. McKeown, and A. Fraser, "Rcp-ac: Congestion control to make flows complete quickly in any environment," in *INFOCOM'06. IEEE International Conference on Computer Communications*, April 2006, pp. 1–5.

[61] J. Alins, J. Mata-Diaz, J. L. Muoz, E. Rendn-Morales, and O. Esparza, "XPLIT: A cross-layer architecture for TCP services over dvb-s2/etsi qos BSM," *Computer Networks*, vol. 56, no. 1, pp. 412 – 434, 2012.

[62] M. Casoni, C. Grazia, M. Klapez, and N. Patriciello, "Towards emergency networks security with per-flow queue rate management," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2015 IEEE International Conference on*, March 2015, pp. 497–502.

[63] M. Casoni, C. Grazia, M. Klapez, N. Patriciello, A. Amditis, and E. Sdongos, "Integration of satellite and lte for disaster recovery," *Communications Magazine, IEEE*, vol. 53, no. 3, pp. 47–53, March 2015.

[64] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP westwood: end-to-end congestion control for wired/wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 467–479, 2002.

[65] I. Biswas, "An investigation on TCP large initial window," *International Journal of Satellite Communications and Networking*, vol. 31, no. 3, pp. 111–121, 2013. [Online]. Available: http://dx.doi.org/10.1002/sat.1023

[66] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 118–130, Oct. 1998. [Online]. Available: http://doi.acm.org/10.1145/285243.285273

[67] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 50–61, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2043164.2018443

[68] R. Adams, "Active queue management: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 3, pp. 1425–1476, Third 2013.

[69] S. Oruganti and M. Devetsikiotis, "Analyzing robust active queue management schemes: a comparative study of predictors and controllers," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 3, May 2003, pp. 1531–1536 vol.3.

[70] R. Zhu, H. Teng, and J. Fu, "A predictive pid controller for aqm router supporting tcp with ecn," in *Communications, 2004 and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceedings. The 2004 Joint Conference of the 10th Asia-Pacific Conference on*, vol. 1, Aug 2004, pp. 356–360 vol.1.

[71] K. Zhou, K. L. Yeung, and V. O. Li, "Nonlinear red: A simple yet efficient active queue management scheme," *Computer*

Networks, vol. 50, no. 18, pp. 3784 – 3794, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606000879

[72] D. Lin and R. Morris, "Dynamics of random early detection," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 127–137, Oct. 1997. [Online]. Available: http://doi.acm.org/10.1145/263109.263154

[73] R. Pan, B. Prabhakar, and K. Psounis, "Choke - a stateless active queue management scheme for approximating fair bandwidth allocation," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2000, pp. 942–951 vol.2.

[74] C.-Y. Ho, Y.-C. Chan, and Y.-C. Chen, "Ward: a transmission control protocol-friendly stateless active queue management scheme," *Communications, IET*, vol. 1, no. 6, pp. 1179–1186, Dec 2007.

[75] D. Raghuvanshi, B. Annappa, and M. Tahiliani, "On the effectiveness of codel for active queue management," in *Advanced Computing and Communication Technologies (ACCT), 2013 Third International Conference on*, April 2013, pp. 107–114.

[76] A. Kamra, H. Saran, S. Sen, and R. Shorey, "Fair adaptive bandwidth allocation: a rate control based active queue management discipline," *Computer Networks*, vol. 44, no. 2, pp. 135 – 152, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128603003475

[77] L. Lu, H. Du, and R. P. Liu, "Choker: A novel aqm algorithm with proportional bandwidth allocation and tcp protection," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 637–644, Feb 2014.

[78] W. chun Feng, A. Kapadia, and S. Thulasidasan, "Green: proactive queue management over a best-effort network," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 2, Nov 2002, pp. 1774–1778 vol.2.

[79] G. Kambourakis, C. Kolias, S. Gritzalis, and J. H. Park, "Dos attacks exploiting signaling in {UMTS} and {IMS}," *Computer Communications*, vol. 34, no. 3, pp. 226 – 235, 2011, special Issue of Computer Communications on Information and Future Communication Security.

[Online]. Available: http://www.sciencedirect.com/science/article/pii/S014036641000085X

[80] M. Pavloski and E. Gelenbe, "Signaling attacks in mobile telephony," in *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, 2014, pp. 206–212. [Online]. Available: http://dx.doi.org/10.5220/0005019802060212

[81] M. Casoni, C. Grazia, M. Klapez, and N. Patriciello, "A cooperative middleware for enhancing tcp performance over high delay networks," in *Global Communications Conference (GLOBECOM), to appear in 2015 IEEE*, Dec 2015.

[82] http://www.dii.unimo.it/wiki/images/b/b7/QRM_2014_11_27.zip, 2014.

[83] M. Gerla, R. L. Cigno, S. Mascolo, and W. Weng, "Generalized window advertising for tcp congestion control," *European Transactions on Telecommunications*, vol. 13, no. 6, pp. 549–562, 2002. [Online]. Available: http://dx.doi.org/10.1002/ett.4460130602

[84] L. Kalampoukas, A. Varma, and K. Ramakrishnan, "Explicit window adaptation: a method to enhance tcp performance," *Networking, IEEE/ACM Transactions on*, vol. 10, no. 3, pp. 338–350, Jun 2002.

[85] H. Byun, "An explicit window adaptation algorithm over tcp networks using supervisory control," *J. High Speed Netw.*, vol. 17, no. 4, pp. 207–218, Dec. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1971866.1971870

[86] M. Barbera, A. Lombardo, C. Panarello, and G. Schembra, "Active window management: An efficient gateway mechanism for tcp traffic control," in *Communications, 2007. ICC '07. IEEE International Conference on*, June 2007, pp. 6141–6148.

[87] ——, "Queue stability analysis and performance evaluation of a tcp-compliant window management mechanism," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 4, pp. 1275–1288, Aug 2010.

[88] S. D. Strowes, "Passively measuring tcp round-trip times," *Queue*, vol. 11, no. 8, pp. 50:50–50:61, Aug. 2013.

[89] P. Marchetta, A. Botta, E. Katz-Bassett, and A. Pescapé, "Dissecting round trip time on the slow path with a single packet," in *Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362*, ser. PAM 2014. New York, NY, USA: Springer-Verlag New York, Inc., 2014, pp. 88–97.

[90] B. Veal, K. Li, and D. Lowenthal, "New methods for passive estimation of tcp round-trip times," in *Proceedings of the 6th International Conference on Passive and Active Network Measurement*, ser. PAM'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 121–134.

[91] M. Casoni, C. Grazia, and P. Valente, "A low-latency and high-throughput scheduler for emergency and wireless networks," in *Communications Workshops (ICC), 2014 IEEE International Conference on*, June 2014, pp. 231–236.

[92] F. Yamaguchi and H. Nishi, "Hardware-based hash functions for network applications," in *Networks (ICON), 2013 19th IEEE International Conference on*, Dec 2013, pp. 1–6.

[93] N. Khademi, D. Ros, and M. Welzl, "The new aqm kids on the block: An experimental evaluation of codel and pie," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, April 2014, pp. 85–90.

[94] C. Grazia, M. Casoni, M. Klapez, and N. Patriciello, "Pink: Proactive injection into ack, a queue manager to impose fair resource allocation among tcp flows," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, Oct 2015, pp. 132–137.

[95] Y. Gong, D. Rossi, C. Testa, S. Valenti, and M. Taht, "Fighting the bufferbloat: On the coexistence of aqm and low priority congestion control," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, April 2013, pp. 411–416.

[96] T. Hiland-Jrgensen, P. Hurtig, and A. Brunstrom, "The good, the bad and the wifi: Modern AQMs in a residential setting," *Computer Networks*, vol. 89, pp. 90 – 106, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615002479

[97] "Pink ns-3 source code," http://netlab.ing.unimo.it/sw/PINK.zip, December 2015.

[98] A. Polydoros, N. Dimitriou, G. Baldini, I. Fovino, M. Taddeo, and A. Cipriano, "Public protection and disaster relief communication system integrity: A radio-flexibility and identity-based cryptography approach," *Vehicular Technology Magazine, IEEE*, vol. 9, no. 4, pp. 51–60, Dec 2014.

[99] P. Valente, "Reducing the execution time of fair-queueing packet schedulers," *Computer Communications*, vol. 47, pp. 16–33, 2014.

[100] H. Bennet, Jon C. R.and Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 675–689, 1997.

[101] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *IEEE/ACM Transactions on Networking*, 1995, pp. 375–385.

[102] A. Gotsis, A. Lioumpas, and A. Alexiou, "M2m scheduling over lte: Challenges and new perspectives," *Vehicular Technology Magazine, IEEE*, vol. 7, no. 3, pp. 34–39, Sept 2012.

[103] C. Cicconetti, L. Lenzini, A. Lodi, S. Martello, E. Mingozzi, and M. Monaci, "Efficient two-dimensional data allocation in ieee 802.16 ofdma," *Networking, IEEE/ACM Transactions on*, vol. 22, no. 5, pp. 1645–1658, Oct 2014.

[104] R. Cohen and G. Grebla, "Multidimensional ofdma scheduling in a wireless network with relay nodes," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[105] J. Rajamaki, "The mobi project: Designing the future emergency service vehicle," *Vehicular Technology Magazine, IEEE*, vol. 8, no. 2, pp. 92–99, June 2013.

[106] Y. Zhang, *Mobile WiMAX: Toward broadband wireless metropolitan area networks*. Auerbach Pub, 2007.

[107] M. Casoni, C. A. Grazia, and P. Valente, "Tempest: a new test environment for performance evaluation of the scheduling of packets," *Simulation Modelling Practice and Theory*, vol. 49, pp. 258–276, December 2014.

[108] Y. Yi, Y. Seok, T. Kwon, Y. Choi, and J. Park, "W2f2q: packet fair queuing in wireless packet networks," in *WOWMOM*, 2000, pp. 2–10.

[109] P. Valente, "Exact gps simulation and optimal fair scheduling with logarithmic complexity," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1454–1466, 2007.

[110] F. Sabrina, "A novel resource scheduling algorithm for qos-aware services on the internet," *Computers and Electrical Engineering*, vol. 36, no. 4, pp. 718–734, 2010.

[111] N.-H. Lee, J.-G. Choi, and S. Bahk, "Extended proportional fair scheduling for statistical qos guarantee in wireless networks," *Communications and Networks, Journal of*, vol. 12, no. 4, pp. 346–357, 2010.

[112] W. Formyduval and D. Thuente, "Priority inversion and queue management for 802.11 priority wlans," in *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, 2013, pp. 565–573.

[113] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," *Proceedings of IEEE INFOCOMM 98*, vol. 3, no. c, pp. 1103–1111, 1998. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=662920

[114] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 473–489, Aug. 1999. [Online]. Available: http://dx.doi.org/10.1109/90.793003

[115] Y. Yi, Y. Seok, T. Kwon, Y. Choi, and J. Park, "W2f2q: packet fair queuing in wireless packet networks," in *WOWMOM*, 2000, pp. 2–10.

[116] A. Iera, A. Molinaro, and S. Pizzi, "Channel-aware scheduling for qos and fairness provisioning in ieee 802.16/wimax broadband wireless access systems," *Ieee Network*, vol. 21, no. 5, pp. 34–41, 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4305171

[117] D. Vasiliadis, G. Rizos, and C. Vassilakis, "Class-based weighted fair queuing scheduling on dual-priority delta networks," *Journal of Computer Networks and Communications*, 2012.

[118] S. Ramabhadran and J. Pasquale, "The stratified round robin scheduler: design, analysis and implementation," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1362–1373, 2006.

[119] X. Yuan and Z. Duan, "Fair round-robin: A low complexity packet scheduler with proportional and worst-case fairness," *IEEE Transactions on Computers*, vol. 58, no. 3, 2009.

[120] S. Louvros and M. Paraskevas, "Analytical average throughput and delay estimations for lte uplink cell edge users," *Computers and Electrical Engineering*, vol. 40, no. 5, pp. 1552–1563, 2014.

[121] S.-C. Ke, Y.-W. Chen, and H.-A. Fang, "An energy-saving-centric downlink scheduling scheme for wimax networks," *International Journal of Communication Systems*, vol. 27, no. 11, pp. 2518–2535, 2014.

[122] E. Rendon-Morales, J. Mata-Daz, J. Alins, J. Muoz, and O. Esparza, "Cross-layer packet scheduler for qos support over digital video broadcasting-second generation broadband satellite systems," *International Journal of Communication Systems*, vol. 27, no. 10, pp. 2063–2082, 2014, cited By 0.

[123] M. Casoni, A. Paganelli, and P. Valente, "A modular architecture for qos provisioning over wireless links," in *Proc. of the 8th IEEE International Workshop PAEWN*, Barcelona, (Spain), 2013.

[124] S.J.Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proceedings of IEEE INFOCOM '94*, pp. 636–646, 1994.

[125] L. Rizzo and P. Valente, "Analysis of fair queueing schedulers in real systems," 2012.

[126] http://algogroup.unimore.it/people/paolo/agg-sched/test-env.tgz, 2013.

[127] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, "A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, 2011, pp. 1–6.

[128] M. Sadri, A. Bartolini, and L. Benini, "Single-chip cloud computer thermal model," in *Thermal Investigations of ICs and Systems (THERMINIC), 2011 17th International Workshop on*, 2011, pp. 1–6.

[129] A. Ghaffar Pour Rahbar and O. Yang, "Lgrr: A new packet scheduling algorithm for differentiated services packet-switched networks," *Computer Communications*, vol. 32, no. 2, pp. 357–367, 2009.

[130] M. Karsten, "Approximation of generalized processor sharing with stratified interleaved timer wheels," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 708–721, June 2010.

[131] F. Checconi, L. Rizzo, and P. Valente, "Qfq: Efficient packet scheduling with tight guarantees," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 802–816, 2013.

[132] C. Peoples, G. Parr, S. McClean, B. Scotney, and P. Morrow, "Performance evaluation of green data centre management supporting sustainable growth of the internet of things," *Simulation Modelling Practice and Theory*, 2013.

[133] B.-H. Liu and J.-Y. Jhang, "Efficient distributed data scheduling algorithm for data aggregation in wireless sensor networks," *Computer Networks*, vol. 65, pp. 73–83, 2014, cited By 0.

[134] K. Huq, S. Mumtaz, J. Rodriguez, and R. Aguiar, "A novel energy efficient packet-scheduling algorithm for comp," *Computer Communications*, vol. 50, pp. 53–63, 2014.

[135] W. Lai and C.-L. Tang, "Qos-aware downlink packet scheduling for lte networks," *Computer Networks*, vol. 57, no. 7, pp. 1689–1698, 2013.

[136] B. Bellalta, A. Faridi, J. Barcelo, V. Daza, and M. Oliver, "Performance analysis of a multiuser multi-packet transmission system for wlans in non-saturation conditions," *Computer Networks*, vol. 60, pp. 88–100, 2014.

[137] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," vol. 27, no. 1, 1997, pp. 31–41.

[138] M. Carbone and L. Rizzo, "Dummynet revisited," vol. 40, no. 2, 2010, pp. 12–20.

[139] S. Ben-Guedria, B. Sans, and J.-F. Frigon, "Polymax, a mobile wimax module for the ns-2 simulator with qos and amc support," *Simulation Modelling Practice and Theory*, vol. 19, no. 9, pp. 2076–2101, 2011.

[140] "The network simulator ns-2: Nist add-on - ieee 802.16 model (mac+phy)," *National Institute of Standards and Technology*, 2009.

[141] C. Courcoubetis and V. Siris, "Procedures and tools for analysis of network traffic measurements," *Performance Evaluation*, vol. 48, no. 1-4, pp. 5–23, 2002.

[142] http://info.iet.unipi.it/∼luigi/papers/20100210-qfq-test.tgz, 2010.

[143] P. Valente and L. Rizzo, "Analysis of fair queueing schedulers in real systems," 2012.

Publications List

Journal Papers

1. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "QRM: a Queue Rate Management for fairness and TCP flooding protection in mission-critical networks", Computer Networks, Elsevier, pp. 54-65, Vol.93, December 2015, ISSN: 1389-1286, DOI: 10.1016/j.comnet.2015.10.010.

2. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "A congestion control middleware layer with dynamic bandwidth management for satellite communications", International Journal of Satellite Communications and Networking, John Wiley & Sons, 2015, ISSN: 1542-0981, DOI: 10.1002/sat.-1129.

3. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, A. Amditis and E. Sdongos, "Integration of Satellite and LTE for Disaster Recovery", Communications Magazine, IEEE, March 2015, Pages 47-53, ISSN: 0163-6804, DOI: 10.1109/MCOM.2015.7060481.

4. M. Casoni, Carlo A. Grazia, P. Valente, "TEMPEST: a new Test EnvironMent for Performance Evaluation of the Scheduling of packeTs", Simulation Modelling Practice and Theory (SIMPAT), Elsevier, Volume 49, December 2014, Pages 258276, ISSN: 1569-190X, DOI: 10.1016/j.simpat.-2014.10.005.

5. M. Casoni, Carlo A. Grazia, P. Valente, A Green, "Modular and Fair Packet Scheduler for boosting Throughput over Wireless Links", Journal of Networking Technology (JNT), Vol. 4, Issue 4, pp. 192-202, December 2013, ISSN: 0976-898X.

Conference Papers

1. M. Casoni, Carlo A. Grazia, M. Klapez, "Enabling Resource Pooling in Wireless Networks through Software-Defined Orchestration", IEEE International Conference on Communications (ICC'16), May 23-27, 2016, Kuala Lumpur, Malaysia.

2. M. Casoni, Carlo A. Grazia, M. Klapez, "An SDN and CPS based opportunistic Upload Splitting for mobile users", EAI International Conference on CYber physiCaL systems, iOt and sensors Networks (CYCLONE), October 26, 2015, Rome, Italy.

3. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "DyBRA: a Dynamic Bandwidth Reservation Algorithm to enhance Satellite Communications", IEEE Global Communications Conference (GLOBECOM), December 6-10, 2015, San Diego, California (USA), ISBN: 978-1-4799-5952-5.

4. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "A Cooperative Middleware for Enhancing TCP Performance over High Delay Networks", IEEE Global Communications Conference (GLOBECOM), December 6-10, 2015, San Diego, California (USA), ISBN: 978-1-4799-5952-5.

5. Carlo A. Grazia, M. Klapez, N. Patriciello, M. Casoni, "PINK: Proactive INjection into acK, a queue manager to impose fair resource allocation among TCP flows", to be presented at IEEE WiMob Conference, Workshop EN4PPDR, October 19-21, 2015, Abu Dhabi, UAE, pp. 132-137, ISBN: 978-1-4673-7700-3.

6. Carlo A. Grazia, M. Klapez, N. Patriciello, M. Casoni, H. Gierszal, P. Tyczka, K. Pawlina, A. Amditis, E. Sdongos, "Performance Evaluation and Economic Modelling of PPDR Communication Systems", to be presented at IEEE WiMob Conference, Workshop EN4PPDR, October 19-21, 2015, Abu Dhabi, UAE, pp. 75-82, ISBN: 978-1-4673-7700-3.

7. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "Implementation and Validation of TCP Options and Congestion Control Algorithms for ns-3", Proc. of the 2015 ACM Workshop on ns-3 (WNS3), 13-14 May 2015, Barcelona (Spain), pp. 112-119, ISBN: 978-1-4503-3375-7, DOI: 10.1145/2756509.2756518.

8. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "Towards Emergency Networks Security with Per-Flow Queue Rate Management", International Workshop on Pervasive Networks for Emergency Management (PerNEM), March 27, 2015, St. Louis, Missouri.

9. M. Casoni, Carlo A. Grazia, M. Klapez, N. Patriciello, "Reducing Latency in Satellite Emergency Networks through a Cooperative Transmission Control", Proc. of IEEE Global Communications Conference (GLOBECOM), December 8-12, 2014, Austin, Texas.

10. Carlo A. Grazia, M. Casoni, P. Valente, "TEMPEST: Test EnvironMent for Performance Evaluation of the Scheduling of packeTs", Proc. of IEEE Emergency Networks for Public Protection and Disaster Relief (EN4PPDR) Workshop within IEEE WiMob 2014, 8 October 2014, Larnaca (Cyprus), pp.300-307, ISBN: 978-1-4799-5040-9.

11. Carlo A. Grazia, M. Klapez, N. Patriciello, M. Casoni, A. Amditis, E. Sdongos, H. Gierszal, D. Kanakidis, C. Katsigiannis, K. Romanowski, P. Simplicio, A. Oliveira, S. Sonander, J. Jackson, "Integration between Terrestrial and Satellite Networks: the PPDR-TC vision", Proc. of IEEE Emergency Networks for Public Protection and Disaster Relief (EN4PPDR) Workshop within IEEE WiMob 2014, 8 October 2014, Larnaca (Cyprus), pp.202-209, ISBN: 978-1-4799-5040-9.

12. M. Casoni, C. A. Grazia, P. Valente, "A Low-Latency and High-Throughput Scheduler for Emergency and Wireless Networks", IEEE International Conference on Communications (ICC), June 10-14, 2014, Sydney, Australia.

13. M. Casoni, Carlo A. Grazia, N. Patriciello, "On the Performance of Linux Container with Netmap/VALE for Networks Virtualization", IEEE International Conference On Networking (ICON), December 11-13, 2013, Singapore.

14. M. Casoni, Carlo A. Grazia, P. Valente, "A Flexible and Green Scheduler for providing QoS and High Throughput over Wireless Links", IEEE International Conference on Communications Systems and Technologies (ICCST), October 04-06, 2013, Hammamet, Tunisy, pp.218-225, ISBN: 978-9938-9511-6-5.

15. E. Giachino, Carlo A. Grazia, C. Laneve, M. Lienhardt, P. Wong, "Dead-Lock Analysis of Concurrent Objects - Theory and Practice", International Conference on Integrated Formal Methods (iFM), Springer, June 10-14, 2013, Turku, Finland.

Acknowledgments

It is not easy to write about all the people that have contributed to your personal and professional growth over a period of more than 3 years. For this reason I am pretty sure that this final part of my thesis will be more an "apologize" section instead of an "acknowledgement" section.

First, I would like to highlight the effort given by the other-side-of-the-PhD people, starting from my girlfriend Aurora. She deserves the first words. If I approached this PhD journey and realized that this has been the right choice for me is simply because of her. She probably believes in me more than myself, and this has been necessary, if not fundamental, in messy and stressful situations (from an academic point of view) like workshops and conferences in which you could feel belittled in a couple of minutes. Together with my family and her family I always found the right amount of support for going on and do my best, in a very natural way.

Starting with the research-affected people, I would like to express my deepest gratitude to my supervisor, Prof. Maurizio Casoni, for accepting me as his PhD student and always being very positive from the start. Without his constant support this work would not have been possible. I am very grateful to Prof. Paolo Valente, for introducing me to Maurizio and participate to the initial and most slippery part of my PhD, steering me in the right direction from the very beginning. I would like to massively thank the "MarNatArlo" research group, the best research team that a PhD candidate could dream on, Natale and Martin have been solid colleagues, valuable co-authors and important friends that helped me in making this journey a success.

I would then like to thank all my friends "cumpa" in Spilamberto. I would like to write about each of them but this would require another entire PhD thesis long at least twice the present one. I will thus limit myself to say thank to all

of them for our long-time and everlasting friendships. I will just like to mention a few names apologizing for those who are not here: Olo, Ide, Panca, Tonno, Omo, Boldro, Dea, Zanna, Jack e Cesco. I would also like to thank the friends and colleagues in the Elecom Lab and OptoLab in Modena, especially Matteo, Laura and Mario.