
# Università degli Studi di Napoli Federico II

Facoltà di Ingegneria

Corso di Dottorato di Ricerca in Ingegneria Informatica ed Automatica
XXII Ciclo
Dipartimento di Informatica e Sistemistica


# Towards informed diversity in IP networks: techniques, issues, and solutions


Alessio Botta


Ph.D. Thesis


Tutor
Prof. Giorgio Ventre

Cotutor
Prof. Antonio Pescapé
Prof. Ernst Biersack

Coordinator
Prof. Luigi Pietro Cordella



November 2009

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter we introduce the problem under study, illustrating the limitations of current Internet scenario. We describe the approaches proposed in literature to overcome these issues, which are either revolutionary or evolutionary. Afterwards, the contributions provided in this thesis are sketched, and the relation between our approach and the others in literature are evidenced. The ending part of the chapter outlines the organization of the thesis.

## 1.1 The current Internet scenario

The Internet has been designed for heterogeneity. In particular, Prof. David Clark from MIT formulates in the design goals of the DARPA Internet that the network must support (i) multiple types of services and (ii) accommodate a variety of physical networks. These design goals are the most important ones besides the interconnection of existing networks and survivability [1]. Moreover, they have led to two design principles: the end-to-end argument and layering. These principles have coined the Internet architecture and were among the key enablers of the stunning success of the Internet. In particular, they have shaped the architecture of the Internet into the well-known hourglass (see Figure 1.1).

However, is the heterogeneity envisioned four decades ago still the same heterogeneity we experience today? We argue that the notion and the challenges of heterogeneity have significantly changed over time. In particular, the heterogeneity targeted in the early days focused on *co-existence*, i.e. the ability to seamlessly connect different network technologies and shield the upper-layer protocols and end systems from the details of the underlying technologies and protocols. In contrast, today, we are challenged to make the

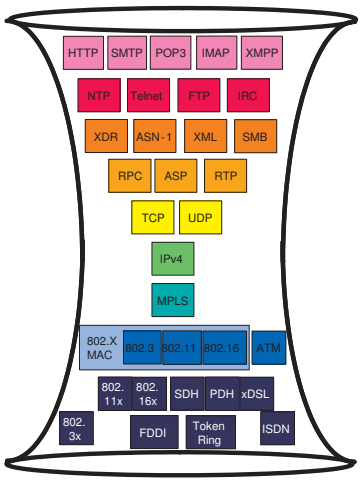



Figure 1.1: Internet "hourglass"

heterogeneous technology *concurrently collaborate*. In particular, in the wake of the fixed-mobile convergence, networks suddenly face the challenge to either dynamically choose one of the available technologies or even to concurrently use multiple technologies. For example, modern cities typically provide multiple wireless access technologies, such as GSM, 3G, WLAN or even WiMax. All these technologies are concurrently available and modern devices are even equipped with multiple radios to take advantage of the concurrent availability of the heterogeneous technology. At the same time, the pletora of applications that have spread in the last years depicted a very heterogeneous network scenario in terms of QoS requirements, traffic patterns, user profiles, etc. This situation pushes towards breaking the layering principle as the applications would like to know the underlying technologies to behave properly. The availability of heterogeneous resources and applications puts forward a set of unprecedented challenges.

When the Internet started to become universally used and the its applications started to differentiate, many researchers claimed that a global Quality of Service support was needed, to allow the use of applications such as video or voice communication. We saw, instead, that the over-provisioning policy of the operators has somehow solved the problem, even if in a very inefficient way. We are aware, however, that these kinds of solutions are not only inefficient, but also ineffective in providing a long-term answer to the problem. More carefully planned strategies are needed to sustain the new kinds of traffic that are transported today over a very heterogeneous infrastructure. We analyze the proposals in this area in Section 1.2. Before that, we explain what are the main challenges that

have to be faced.

### 1.1.1 New access technologies

The access network technology has seen improvements both in the LAN and in the WAN areas. The most challenging scenarios are related to wireless Internet access.

As for the LAN environment, in the early '90 the IEEE approved the first 802.11 standard [2], dictating the specifications for physical and data-link layer of wireless LAN (WLAN) using both radio waves and infrared rays. The limited speed of this standard ($1 \rightarrow 2$ Mbps) was the cause of its little success. Its evolution yielded to the standardization of the IEEE 802.11a and 802.11b in 1997, allowing data rates up to 11 Mbps, and more recently, of the IEEE 802.11g and 802.11n, which is still an ongoing work and should allow data rates up to few hundreds of Mbps. The 802.11 MAC uses a multiple access technique called CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), which requires the stations to *sense* the channel before transmissions in order to detect if other stations are currently transmitting. This technique is similar to the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) used by the IEEE 802.3 at the base of the Ethernet standard. However, while with the 802.3 the stations are surely able to detect the presence of other transmissions on the shared channel, in wireless environments such detection is not always possible. Therefore, the CSMA/CA differs from the CSMA/CD in that the stations have to wait for a random time before transmitting also when the channel seems to be clear. Another important characteristic of the 802.11 MAC is the presence of a packet acknowledgement mechanism to ensure transmission reliability. Basically, after having successfully received a packet, a station has to send a special packet (called acknowledgement or simply ack) to the transmitting station. In case the ack is not received, the packet is scheduled for retransmission. These mechanisms have a clear impact on the Quality of Service (QoS) parameters observed in WLAN. For instance, the acknowledgement mechanism can cause large and variable transmission times in networks with several stations, which introduces more unpredictability in the end-to-end path [3].

As for the WAN environment, the cellular network is more and more used for Internet access since the introduction of the General Packet Radio Service (GPRS). The GPRS is actually a set of evolved services based on the GSM infrastructure, which allows packet-switching based communications on the circuit-switching network of the GSM. The spread of the GPRS was mainly due to the fact that few modifications were required for the

telecom operators to deploy this technology. In particular, it was necessary to add a node called Serving GPRS Support Node (SGSN) and a gateway called Gateway GPRS Support Node (GGSN). Moreover, these two components were also useful to progressively upgrade the network to the UMTS standard, which has now become very common in Europe for broadband Internet access. Over the last decade, the data rates achievable through the cellular network have increased from a few tens of Kbps using GSM to about one hundred Kbps for GPRS, then several hundred Kbps with basic UMTS, and finally today, several Mbps with the High Speed Downlink Packet Access (HDSPA) that is part of the latest UMTS releases [4]. However, the complexity of the infrastructure of the cellular networks still results in unpredictable behaviours of the QoS parameters, as better explained in the following.

### 1.1.2 New applications and protocols

In recent years people have started to use the Internet for activities previously happening only in certain contexts and through different technologies. This is the case, for example, of peer-to-peer IP Television (P2P IPTV), and network gaming in virtual worlds. The time and place of such activities change, and services become ubiquitous. People move from the *sofa at home* to the workplace or a café to enjoy such services. Moreover, they interact with communities that range on a global scale rather than having a strong local geographical bound. This phenomenon seems to have just started. For instance, the popularity of Internet-based television is expected to grow during the next years for several reasons [5]: i) the Internet has become one of the major source of information for people at their workplace; ii) users appreciate the generalist TV always less, whereas they are more interested in specialized contents, in being able to interact somehow with other users, and in adding contents [6]; iii) in some countries the quality and the range of the offer of TV contents is scarce; iv) the "*Broadcast yourself*" phenomenon is constantly increasing, both with "**Tube*" sites and the creation of more elaborate TV programs with realtime broadcasting created by single users [7, 8, 9]. Other notable examples of this small revolution are the explosion of Internet Blogs, Video publishing and distribution systems, social networks built through the Web, Virtual Worlds, network games, etc. [10, 11].

From a networking point of view, such new applications sometimes use traditional protocols (e.g. HTTP) and communication paradigms (e.g. client-server). Other times, new protocols and communication paradigms are conceived to support them. As an ex-

ample, in 2006 Floyd et al. have standardized the Datagram Congestion Control Protocol (DCCP) [12] to add only a congestion control to a datagram transport protocol, typically used by real time and streaming applications. DCCP is proposed to avoid congestions on the Internet caused by the increase of the traffic of multimedia applications, which make extensive use of UDP.

### 1.1.3 Performance problems in heterogeneous environments

This new scenario poses new challenges to the Internet infrastructure. In a situation in which all the traffic is carried by the IP, which is unreliable, datagram, and totally unaware of both the underlying technology and the application requirements, it becomes difficult to provide the necessary guarantees in terms of throughput, delay, jitter, and losses to the new applications. On the other hand, Internet users are willing to use all the available applications despite the kind of connection they use. It is easy to imagine, however, that the new access technologies (e.g. the cellular network) do not allow to effectively utilize new applications such as those for P2P IPTV or real-time communication. The result is that users remain largely unsatisfied of the performance they get, and the new access technologies are used by a small fraction of the potential users.

Typical problems affecting the communications in this scenario are poor throughput, high and largely varying latency, and persistent loss. For example, in Fig. 1.2 we can observe the difference between the round trip time (RTT) measured in the wireless and wired sides of the cellular network of one of the main telecom operators in Europe. In particular, in this figure we show the cumulative distribution function (CDF) of the average RTT per connection we measured on the radio access network (named Cellular 1 in the figure) and the connection to the Internet (named Internet 1 in the figure)[1].

Fig. 1.2 shows that the RTT samples on the radio access network have much larger values than on the Internet. Considering that the wireless side constitutes only the access network, while the wired side represents the rest of the path towards the final destination, we can easily understand the impact of the complexity of the cellular network infrastructure on QoS parameters. Performance problems caused by the new access technologies are reported by several studies in literature [3, 13].

To understand the consequences of this situation on the quality of experience of the users, we can compare the values of the QoS parameters typically measured on these

[1]More information on these research activities are provided in Section 3.3

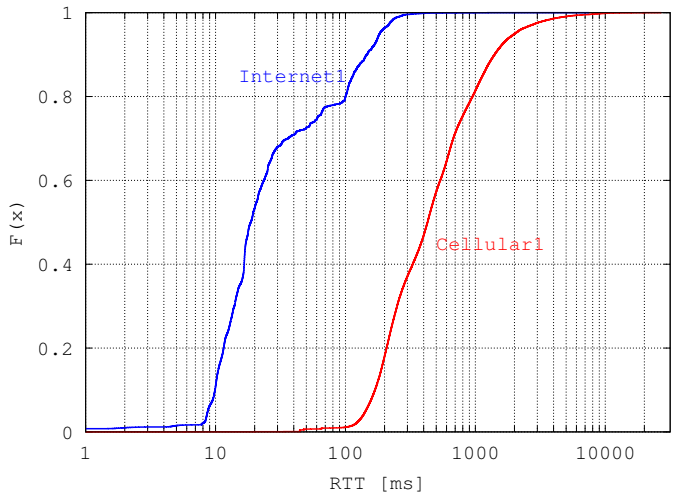



Figure 1.2: RTT measured on the wired and wireless sides of a cellular network.

networks, which are reported in literature (see e.g. [14]), with the values necessary for multimedia communications. For example, VoIP calls tolerate a maximum jitter of 50 ms, and a maximum delay of 150 ms to offer a good voice quality [15]. However, as also shown in Fig. 1.2, it easily happens that higher values are obtained of this access network. It is therefore clear that today's cellular networks cannot be used in substitution of traditional access networks, such as ADSL, for VoIP calls.

The performance problems of new generations applications on new generation access networks have been recognized by the research community and different approaches are currently pursued in order to move towards the *Future Internet*.

## 1.2 Towards the Future Internet

In this section, we overview the main research directions in the area of the Future Internet. We first describe the revolutionary approach, aimed at designing a new network infrastructure, rethinking the basic design choices of the current Internet such as packet format, layering, routing, etc. Afterwards, we review the evolutionary approach, which aim at adding innovative features, both in the core and in the edges, in order to overcome current limitations.

As for the revolutionary approach, a few years ago many prominent researches recognized the need for a new Internet. For example, Prof. David Clark has been advocating for many years the necessity to adopt a "clean slate" approach to the new infrastructure

design [16]. The original protocols of the Internet were designed for a specific purpose: allow communications between a few hundred academic and government users. The design assumed that all the network users could be trusted and that the hosts connected to the Internet were mostly fixed. As explained before, this is not the case anymore. For this reason, according to Prof. Clark, "the Net's basic flaws cost firms billions, impede innovation, and threaten national security". Several initiatives are arising all over the world to promote research in the field of revolutionary approaches to the Future Internet. In the United States, for example, the National Science Foundation (NSF) has invested about $20 million in two projects called Global Environment for Network Innovations (GENI) and Future Internet Design (FIND). In Japan, the National Institute of Information and Communications Technology has launched a programme called Akari, whose aim is the development of a new generation network architecture using a clean slate approach. In Europe, one of the major initiatives under the European Union's Framework Programmes for technological development is the Future Internet Research and Experimentation (FIRE).

On the other hand, three main reasons are at the base of the evolutionary approach. The first one is that the Internet is almost fully commercial, and the investments by operators and individuals make the revolutionary approach unfeasible: industries have invested billions of dollars, and they want to be sure that the current Internet survives and prospers. As a second reason, it has already been shown that the current Internet architecture can be adapted to new services and applications that were not originally imagined. The third reason is that some of the current problems, such as security and spam, are not necessarily due to the Internet architecture. Stepping from these assumptions, the evolutionary approach tries to meet the new demands by adding targeted patches to the Internet to fix current problems, as it has been done over the past decade. In this view, there is the need for end-user and access-network devices equipped with autonomic capabilities, i.e. with information sensing, decision making and enforcement.

## 1.3 Thesis contribution

Many past tentatives to improve the Internet in a revolutionary way have failed. The main failure cause has been the reluctance of vendors and telecom operators to replace a technology that has proved to be effective in a wide range of operating scenarios, with a technology with unknown capabilities. We therefore believe that the evolutionary ap-

proach has more chances to succeed, and we pursue this approach in this thesis.

In the following sections we introduce the contributions provided in this thesis. We start describing the contributions in the area of network monitoring and measurements. Thanks to these activities we have to both acquired important knowledges and devised new methodologies and techniques to infer the status of a heterogeneous network path. This allowed us to advance the state of the art in the field of packet-level diversity, devising new techniques for *informed diversity* at IP layer: packet-level diversity, mainly in time and space, aided by measurements to sense the status of the network, and therefore obtain better performance [2].

### 1.3.1 Inferring the status of the network

In this thesis, we firstly studied the problem of how to acquire information about the status of a heterogeneous network. We were interested in understanding how to effectively and efficiently measure QoS parameters in current Internet scenarios, and then, how to extract the most important indicators of the network status, from the measured information. This task is important for our thesis as we believe that diversity techniques can achieve better performance if supplemented by information about the status of the network.

We performed four experimental campaigns over different networking scenarios. We first explored the possibilities offered by active measurements, which consist in injecting probing packets into the network and collecting measures from the received packets. For this aim, we collaborated to the development of a tool, called D-ITG [17], that allows to perform traffic generation and QoS parameter measurements. D-ITG has been developed by the researchers of the COMICS [18] research group of the Dipartimento di Informatica e Sistemistica of the University of Napoli Federico II. In our activities, we added the possibility to run on different architectures (ARM, Intel network processors, etc.) and operating systems (Snapgear [19], Montavista [20], OpenWrt [21], etc.), and to generate traffic using new transport protocols (DCCP [12] and SCTP [22]). This allowed to perform measurement experiments in heterogeneous scenarios.

Using D-ITG, we analyzed the performance of three different networks: PlanetLab, Magnets, and HetNet. PlanetLab is a planetary-scale testbed constituted by about 1000 hosts made available by universities and research institutes from several countries all over the world. On this infrastructure, we performed a first activity aimed at providing the

[2] A formal definition of informed diversity is provided in Section 2

possibility for the nodes to use a UMTS connection. This allowed to overcome one of the limitations of PlanetLab infrastructure, that is the scarce heterogeneity of such testbed, as all the nodes are connected to the Internet through Ethernet connections. After that, we performed a measurement campaign aimed at comparing the performance achievable using two different paths connecting the same two nodes of the testbed: the first path was made of all wired links, and the second one comprised a UMTS link.

Magnets is a wireless wide area network deployed by Deutsche Telekom Laboratories in Berlin. One of its key distinguishing characteristics is heterogeneity along several dimensions: nodes in the network feature multiple wireless interfaces with different technologies, diverse link characteristics, nodes with varying degrees of processing and storage capabilities, and interconnection of multiple mesh networks with disparate routing protocols. The activities focused on the analysis of the performance of the high-speed WiFi backbone. We performed a deep measurement campaign in order to characterize the QoS parameter behaviour on the links of the backbone, exploiting all the special features of this infrastructure: i) there are two links deployed in parallel between the same two buildings, which allowed a characterization of the influence of environmental factors; ii) there are links that span a distance of 500+ meters, but differ in technology, which allowed to understand the behaviour of 802.11 links over large distances as a function of the frequency band used; iii) we could compare one-hop and multi-hop flow performance; iv) performance improvements achieved by special operating modes (called *Turbo Mode* and *Burst Mode*) were analyzed.

HetNet is a heterogeneous laboratory testbed, comprising different networking technologies (e.g. Ethernet, ADSL, and UMTS), different operating systems (e.g. Linux and Windows), and different devices (e.g. Laptops, Workstations). On this infrastructure, we performed a measurement campaign taking into consideration all the previously cited variables. Moreover, we analyzed the collected measures evaluating different *concise statistics* (mean, standard deviation, inter quantile range, minimum, maximum, and median) as well as *detailed statistics* (probability distribution functions, tail analysis, entropy, and long-range dependence). This allowed to highlight some behaviors that were hidden when applying a traditionally statistical approach, and to give insights to better understand the differences between traditional networks and heterogeneous wired/wireless networks.

Beside the active measurements, we also worked on an operational network using a passive approach. We used packet traces collected in the cellular network of one of the

major operators of central Europe, with the aim to understand the behavior of the network by looking at the traffic generated by the users. We started the investigation by using a methodology previously proposed in literature [23]. However, we verified that the results were affected by errors. Therefore, we designed a new methodology that overcomes the limitations of the previous one. Using this methodology, we firstly studied the performance of all the users of the cellular network as a whole, revealing a high degree of heterogeneity. Afterwards, we looked at the performance of the bulk data transfers of some specific users. We were able to show the results in terms of throughput, loss rate, round trip time, and number of parallel connections of representative users. We also highlighted the impact of the number of parallel connections on the achieved performance.

### 1.3.2 Improved path diversity

Thanks to the knowledge acquired in the previous activities, we could start to work on new schemes for path diversity. We approached this topic performing simulations in Matlab. For this aim, we developed a simulator to explore the potential benefits of classical path diversity techniques such as the round robin. We used a model for packet loss based on 2-state Markov chains, which captures the correlation of the loss process on the Internet. The simulator allows to understand the loss decorrelation power (i.e. the capacity of transforming bursty losses into isolated losses) of this packet distribution scheme.

Afterwards, we developed a first technique for informed path diversity, which works on a packet-by-packet basis. The technique is based on Markov Decision Processes to make informed decisions on which path to choose, in order to minimize or maximize a certain function of the QoS parameters (e.g. minimize the file transfer time). Information about the status of the available paths are obtained by using a passive measurement technique. The network paths are modeled using queueing theory. The reward function of the Markov Decision Process is then able to decide, for each packet, which is the best path to be used in order to reach the required objective. The approach was evaluated with *ns2* simulations in different application scenarios (wireless networks, overlay networks, etc.). The performance achievable were compared with that of two classical packet distribution schemes, that are Weighted Round Robin (WRR) and join the Shortest Queue (JSQ), Obtained results showed how our approach is able to achieve better performance than both WRR and JSQ in all the considered scenarios.

In order to verify if such benefits may be achieved also in real networks, we devel-

oped a tool to experiment with informed path diversity in real scenarios. The first issue we had to solve in this activities is related to the time scale of the path selection. The previous approach required information about the status of the path before each packet transmission. In our research we verified that: i) this granularity results in a high overhead for the measurement process, and ii) the benefits remain almost unchanged if we increase the path switching period. These considerations allowed to relax the hypothesis of taking decisions packet-by-packet, and to develop a technique that switches the path with less frequency (i.e. every $1 \rightarrow 5$ minutes). This technique was implemented in a tool called PathD, which performs active measurements on the available paths to obtain accurate information about the path status. The improvements achievable with PathD were evaluated on a controlled network scenario, and compared with classical path diversity schemes. Obtained results show that our approach achieves better performance.

### 1.3.3 Time diversity at IP layer

In the field of space diversity, we advanced the state of the art proposing new techniques and tools that achieve better performance than the ones currently used. For time diversity, instead, we tackled the problem of how to apply such a transmission schema at IP layer. In literature, several approaches have been proposed to use time diversity in computer communications. However, only few works have been proposed for the IP layer, and none of them is really independent of the application and the underlying technology.

For this aim, we firstly studied the potential benefits of packet interleaving in simulation, in order to understand its loss decorrelation power and to determine the interleaving configuration most suited to a given network condition. To this end, we developed a simulation framework in Matlab using a 2-state Markov chain to model loss behaviour, and a *block interleaver* to reorder the packets as required by the time diversity at IP layer. The simulations allowed to verify that the block interleaver is actually able to provide the expected benefits in terms of loss decorrelation, if properly configured. Moreover, they allowed to obtain a clear understanding of the relation between the interleaver parameters and the loss-burst length (i.e. the number of packets consecutively lost).

We then designed and implemented an application to experiment with packet-level time diversity over real networks. The application, called TimeD, works on Linux-based hosts by intercepting packets passing through the network stack of the kernel, copying such packets in userspace, and manipulating their order according to the interleaving pol-

icy. After the implementation, we firstly evaluated the impact of the packet manipulation process, verifying that this activity does not constitute a bottleneck for the performance. Afterwards, we performed an experimental measurement campaign in order to understand the pros and cons of packet interleaving in real networks. Confirming the simulation results, the experiments showed that time diversity can actually be helpful in real networking scenarios. However, they also evidenced some important issues, such as, for example, the impact of the buffering operation performed by the interleaver on transport protocols with congestion control algorithms, and the necessity to manually tune the interleaver parameters. To cope with these issues, we added measurement capabilities to TimeD, and we devised a dynamic buffering mechanism. Using active measurement tools, TimeD is able to estimate the status of the network in terms of loss characteristics, and therefore, to use the interleaving configuration most suited for this situation. Using passive techniques, TimeD estimates the protocol used by the application that is generating traffic, and the packet rate of such traffic. Thanks to these sensing capabilities, TimeD is actually able to provide the required benefits in heterogeneous environments. Such benefits were evaluated performing a measurement campaign in controlled networks.

## 1.4 Thesis organization

This thesis is organized as follows. In Chapter 2, we introduce the problem of diversity in general, and of packet-layer diversity in particular. We then provide specific information about space and time diversity at IP layer in Section 2.1.1 and 2.1.2 respectively, and we discuss the importance of measurements for these techniques in Section 2.1.3. The motivations for which we decided to work on these topics are illustrated in Section 2.2. After this introduction, we review the works in literature about network measurements, and about path and time diversity in Section 2.3. We describe the open issues in Section 2.4, in order to set the context for the activities described in the following.

In Chapter 3, we report the research activities we performed in order to understand how to accurately estimate the status of a heterogeneous network. In particular, our approaches based on active measurement techniques are described in Section 3.2, providing specific information about PlanetLab in Section 3.2.3, Magnets in Section 3.2.3, and HetNet in Section 3.2.3. In Section 3.3 we detail our measurement approaches based on passive techniques, describing the new methodology we devised in Section 3.3.1, the

analyzed network and the obtained results in Section 3.3.2.

Our research activities on informed path diversity are reported in Chapter 4. In Section 4.3, basic path diversity is analyzed through Matlab simulations. Then, in Section 4.4, we describe our first approach to informed path diversity. The system model is sketched in Section 4.4.2, while the algorithm at the base of our technique is illustrated in Section 4.4.3. Section 4.4.4 is devoted to the discussion of the *ns2* simulations and the results obtained. After the simulation activities, we developed a tool to experiment with informed diversity on real networks, which is described in Section 4.5.1. The experiments conducted and results obtained are reported in Section 4.5.3.

Chapter 5 is related to time diversity. After providing basic definitions and assumptions in Section 5.2, we describe the Matlab simulations we performed in Section 5.3. Then, the tool we developed to experiment with time diversity in real networks is described in Section 5.4, with specific information regarding the first experiments we performed. In Section 5.5, we discuss the issues related to the deployment of time diversity in real networks and the solutions we devised to cope with these issues. In Section 5.6, we report the experiments performed with the improved version of TimeD, commenting on how the information about the status of the network allow to use packet interleaving in heterogeneous scenarios.

Finally, the thesis conclusions are drawn in Chapter 6.

# Chapter 2

# Packet-level diversity in IP networks

In this chapter, we introduce the problem of informed packet-level diversity applied to IP networks, paying particular attention to path and time diversity, two techniques on which we conducted our research activities. After introducing the concept of diversity in general in Section 2.1, we provide insights related to space diversity in Section 2.1.1 and time diversity in Section 2.1.2. Afterwards, in Section 2.1.3, we explain how the knowledge of the status of the network can provide useful information, in order to perform an informed diversity, which achieves higher performance with respect to a non-informed one. Once the basics have been introduced, we discuss the related work in this area in Section 2.3, and we review the main issues that are still to be solved in Section 2.4, also outlining the ones we tackled in this thesis. We finally draw conclusions related to this issues in Section 2.5.

## 2.1 Diversity in computer communications

In the context of computer communications, the term *diversity* refers to techniques to improve the performance and reliability of transmissions in noisy environments by using two or more physical or virtual communication channels, generally with different characteristics. Different kinds of diversity have been used so far. Among the most relevant we can cite the following.

- Space diversity: transmissions are performed over channels that are separated in space. This is, for example, obtained by geographically separating the different transmitters and receivers in wireless scenarios, so that they cannot interfere with

each other, or by using different communication circuits (e.g. cables) in wired scenarios.

- Time diversity: information is transmitted in opportunely separated time periods. If we pursue the objective of increasing the performance and reliability of communications, we will achieve benefits with respect to transmitting data as soon as it is available, only if the channel presents memory (e.g. if the losses are correlated). Spreading the information over different time periods may not provide additional performance or reliability if the channel is memoryless, therefore behaving always in the same way.

- Frequency diversity: transmissions use separate frequencies, even better if orthogonal, so that the channels are physically separated.

- Code diversity: transmissions happen on the same frequency, time, and space, but they use different coding schemes so that the resulting encoded messages are independent. This technique is particularly useful for concurrent user access, and it is used, for example, for sharing the medium among mobile stations residing in the same cell of a cellular network.

The availability of different channels can be exploited in several ways, to pursue different objectives, such as, for example: for transmitting different copies of the same information in order to increase the reliability of the communication; for transmitting the base information on one channel and additional data on others (e.g. with scalable or layered coding schemes), to increase performance or reliability; for transmitting different independent versions of the information (e.g. with multiple description coding schemes); for transmitting different pieces of information in order to decorrelate the losses and reconstruct the original message more easily; for increasing the overall throughput. It is clear that the availability of different channels for the transmission of the information provides several benefits. However, it also bears some costs, which are analyzed in details in the following sections.

Diversity has been traditionally used at physical or data-link layers (see Fig. 2.1). This is not only for historical reasons [1], but mainly because of the characteristics of physical

[1]The technique has been originally devised by researchers working in the Information Theory field and concerned with physical or data-link layer issues.

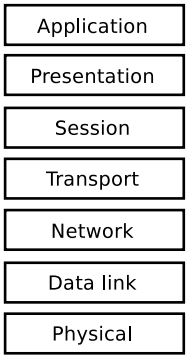



Figure 2.1: The 7 layers of the ISO/OSI model.

channels and the availability of models for such channels. In the last decade, different studies have analyzed packet dynamics at network layer [2], and models have been proposed for the characteristics of the channel at such layer [24, 25, 26, 27, 28, 29, 30]. This has triggered the spread of works aimed at exploiting the benefits of diversity at network layer, which we call packet-level diversity in the following. A survey of these works, with an emphasis on path and time diversity, is reported in Section 2.3. Further motivations for the adoption of space and time diversity at network layer are provided in Section 2.2.

While packet-level diversity seems to be very promising, its application to real networks is not straightforward. As we will see in details in the following sections, this operation has a cost in terms of performance, even more because the characteristics of the channels exploited by these techniques may be different from each other, and the upper-layer protocols (e.g. TCP) have not been designed to work in those conditions. To this end, in this thesis we tackle the problem of how to apply diversity at network layer in IP networks. We focus our attention on space and time diversity, claiming that the use of measurements can help to perform an informed diversity, which mitigates some of the problems of unaware diversity, thus obtaining higher performance.

Before explaining in details the related work in this area and the main contribution of this thesis, it is necessary to introduce the basic concepts of these two techniques. In the following sections, we therefore provide general information on space and time diversity at network layer.

[2]Also know as layer 3 or IP layer in IP networks.

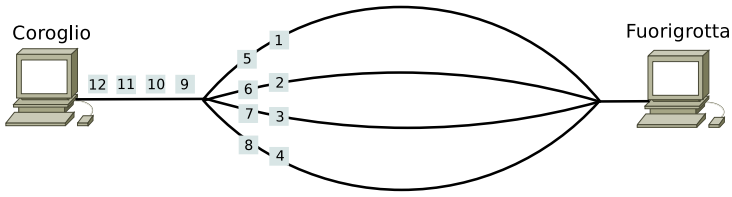



Figure 2.2: Path diversity with round robing scheduling.

### 2.1.1 Space diversity at network layer: path diversity

Space diversity at network layer means transmitting IP packets on different network paths available between a source and a destination. For this reason, space diversity is also called path diversity in the context of packet-switched networks. Clearly, this technique requires the availability of different paths between the two hosts. This is actually more and more common in the current Internet both at user- and network-side, as reported by several works in literature (see for example [31, 32, 33] and references therein).

To explain how path diversity actually works, let us introduce a simple example. In Fig. 2.2, we observe two end hosts connected with each other through four different paths. We can imagine that the host named *Coroglio* has to send packets to the other host named *Fuorigrotta*. A very simple technique can be used to distribute the packets on the available paths: each packet is sent on a different path in a round robin fashion (1st packet on 1st path, 2nd packet on 2nd path, and so on). Though simple, such scheduling discipline can already be effective in improving both the performance and reliability of the communication. The total capacity available to Coroglio can be up to the sum of the capacities of the four path, and if a path is unexpectedly affected by problems, only one fourth of the packets will experience such problems. It is easy to understand that more improvements can be achieved with more sophisticated policies. One, for example, could think of distributing the packets according to the bandwidth (or some other metric) of the available path, thus achieving a higher aggregated bandwidth. A simple policy realizing this is called weighted round robin, which sends more packets on some paths and less on the others, according to the weight previously assigned to each path. This is a first example of informed path diversity and requires some knowledge about the network. A number of issues then arise, ranging from how this knowledge is obtained, to the most effective granularity of the measures. More details on the problems and available solutions for informed path diversity are provided in Section 2.4.

On the other hand, it is worth noting that usually such improvements are not for free. The paths between a source and a destination at network layer can be composed of several layer-2 links, and they may connect two hosts that are very far away from each other. While this complexity is hidden at the IP layer, it plays an important role in augmenting the complexity of models for packet dynamics at this layer. More importantly, such complex communication scenario translates into the fact that the available paths can actually have very different characteristics. In spite of IP, upper layer protocols such as TCP sometimes implicitly assume that consecutive packets between a source and a destination follow almost the same route, or routes with similar characteristics [3]. This has unpredictable consequences on end-to-end performance. Also in this case, having some knowledge about the transport protocol in use, and about the characteristics of the communication in general, may help in mitigating this problem. This is another example of informed path diversity.

Path diversity can be used in scenarios in which more paths are available to reach a destination. As said before, this situation is common both at user- and network-side. In the first case we speak about multi-homed hosts: hosts being connected to the Internet with more than one connection (e.g. WiFi and Ethernet or UMTS). In this case, a mechanism can be used inside the multi-homed host to exploit the benefits of path diversity, improving the performance and reliability of its Internet connection. In case multiple paths are available to an entire network, we speak about multi-homed networks (e.g. a corporate network connected to the Internet through multiple providers). In this case, the path diversity mechanism should reside at the border gateway of the network, in order to exploit the available paths and, again, improve the performance and reliability of all the connections from the multi-homed network. Finally, path diversity can also be used inside a single Autonomous System or, in general, a network under the same administrative domain. If the network is sufficiently meshed, multiple paths may be available to connect two hosts of such network. In this can the path diversity can be realized, for example, by properly configuring the routers of the network.

[3]This assumption is not actually embedded in the protocol. However, the closed loop nature of TCP requires the network conditions to be slowly varying, in order for the throughput to converge to the available bandwidth of the path. Distributing every single packet on a different link with different characteristics is clearly in contrast with such assumption.

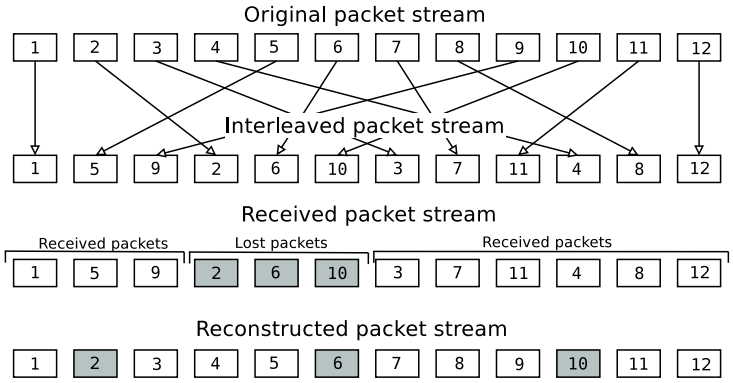



Figure 2.3: Time diversity scheme.

### 2.1.2 Time diversity at network layer: packet interleaving

Time diversity at network layer means altering the time of packet sending in order to benefit from different channel conditions. When applied to a flow of packets between a source and a destination, it translates into altering the reciprocal order of packets (interleaving consecutive with other packets) For this reason it is also called packet interleaving in the following. Using packet interleaving, originally adjacent packets will be spaced by a time interval opportunely chosen, and will experiment different network conditions.

The basic idea to realize packet interleaving is then to resequence the packets before transmission. Thanks to this technique it is possible to decorrelate the dynamics experimented by packets traversing the network: the resequencing allows to space out originally close packets such that their dynamics will not be correlated (e.g. in a environment with bursty losses, a loss of *consecutive packets* will be translated in a loss of *distant ones*). To better understand how this is realized, in Fig. 2.3 we report an example of such technique, aimed at decorrelating the loss process (i.e. we want that losses do not involve consecutive packets). A sequence of 12 packets has to be transmitted from a source to a destination, and a sequence number is assigned to each packet. The loss process on the channel has a burtsy behaviour, and it will result in the loss of three adjacent packets: the fourth, fifth, and sixth one. We can observe that, using the interleaving on the packet sequence before transmission, the loss of such three consecutive packets results, at receiver side, after deinterleaving (i.e. restoring the original sequence), as the loss of packets with sequence number 2, 6, and 10. If interleaving was not used, we would have lost the packets with sequence number 4, 5, and 6, which are consecutive. With this example it is clear that an

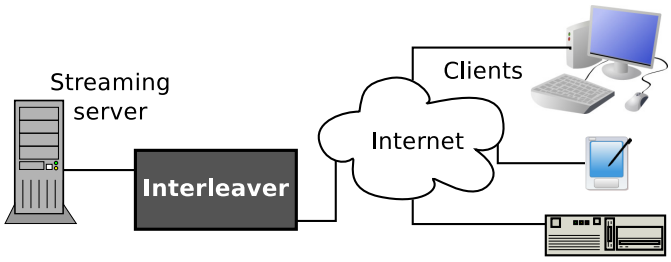



Figure 2.4: Interleaver used beside a video streaming server.

interleaver (coupled with a deinterleaver) may help in decorrelating the packet dynamics, making a channel with bursty loss behaviour perceived as a memoryless channel. But, why is this useful ?

The reasons why consecutive losses are generally worse than the same quantity of isolated losses are different, and they depend on the particular application. While a deeper discussion is reported in Section 2.2.2, we can easily understand the benefits of dispersing the losses if we think to packets transmitted using Forward Error Correction (FEC). In this case, the error recovery information related to a packet may travel in adjacent packets (i.e. packets of the same FEC block). Therefore, if a number of packets within the same block are lost, it is impossible to reconstruct the original information.

The main disadvantage of packet interleaving is the delay introduced. In order to resequence a stream of $k$ packets, it is necessary to wait for them. In the example of Fig. 2.3, packet number 4 has to wait until packet number 11 arrives, in order to be transmitted. This delay is acceptable for certain applications, such as audio and video streaming. Moreover, it is controllable through the length of the interleaving sequence.

Packet interleaving can be used both at host-side and before a particularly congested link. Fig. 2.4 shows one possible application scenario: a packet interleaver is used besides a video server in order to operate on the video flow generated by such server. This allows to protect the packets transporting the video flow from possible bursty losses, in order to increase the quality perceived by the user. While this can also be obtained by introducing an interleaver inside the video server (before or after the encoder), having the interleaver as a separate entity allows to achieve higher flexibility and to be independent of the particular codec or protocol used by the server.

### 2.1.3 The role of measurements in packet-level diversity

We have introduced the basic concepts of path and time diversity in IP networks in the previous sections, providing first insights on their usefulness for current networking scenarios. In this thesis, however, we also advocate that such transmission strategies could provide higher benefits if supplemented with information about the status of the network. We believe that if we want to deploy tools that implement path and time diversity in an operational network, we have to integrate network measurement and monitoring techniques into such tools. *In the long term we envisage the use of autonomic network devices (hardware or software) having sensing capabilities to acquire information about their context, and having at their disposal a set of techniques, such as path and time diversity, to mitigate the impairments detected.*

In general, measurements are necessary for packet-level diversity to configure, and possibly adapt at run time, their parameters. For example, in the case of path diversity, it is first of all necessary to know the number and kind of the available paths in order to set up the proper transmission strategy. Clearly, this configuration can be manually done by an operator, who has some knowledge about the network on which the path diversity will be applied. However, two main problems arise from the manual configuration. The first is that only a limited visibility is obtainable from a single vantage point in the Internet. That is to say, if an operator configures a path diversity scheme looking at his own domain only, he will not generally achieve optimal performance. An example of this situation is a user willing to deploy a path diversity strategy, using weighted round robin, in his own laptop that is connected to the Internet with both WiFi and UMTS connections. Due to the higher capacity of WiFi, he would probably assign a higher weight to such connection so that more packets are sent through it. It may happen, however, that packets directed towards a subset of destination hosts may experience better network conditions if using primarily the UMTS connection. This happen, for instance, if the path towards those hosts, using the WiFi access network, has some chronically congested links. In this situation, an informed path-diversity technique would correctly assign the weights to the paths for these destinations, thanks to a wider knowledge of the paths acquired performing ad-hoc end-to-end measurements. The second problem is related to the possibility that these conditions may vary over time. Again, the availability of a continuous monitoring system would result in better performance as the paths will be selected dynamically, according to the results of the measurements.

Similar considerations can be made for time diversity at packet level. The dynamics of packet loss can be time varying indeed. Meaning that, the decisions on how to interleave the packets, in order to protect them from bursty losses, have to be repeatedly updated. Moreover, interleaving can have different effects on traffic from different applications and transport protocols. For example, in case the application or the transport protocol employs a congestion control algorithm, the buffering performed by the interleaver may cause serious performance problems, being mistaken for a congestion on the network. This aspect, which will be investigated in details in Section 5.5.4, requires the interleaver to be aware of the characteristics of the traffic it has to work on. Also in this case, an operator may provide some hints knowing the application characteristics. However, modern applications employ rate- and protocol-adaptation techniques as well as use different kinds of traffic for different services provided to the user, which limits the viewpoint of the operator [34, 35]. On the other hand, tailored measurements can be more effective in estimating the real needs of the application traffic.

## 2.2 Motivation

In the previous sections, we have already provided a number of good reasons for the use of path and time diversity in IP networks. In this section, we summarize such reasons, providing the motivations at the base of our work.

Let us first justify our choices in terms of the level to work at. Path and time diversity have proved to be effective in improving network performance at physical or data-link layer (see the ISO/OSI protocol stack in Fig. 2.1). In this thesis, however, we advocate the use of such techniques at network layer. One of the main motivations for the choice of this layer comes from the recent availability of models for packet dynamics at IP layer [24, 25, 26, 27, 28, 29, 30]. Those models characterize the effect of the network on IP packets, thus abstracting an end-to-end path between two network hosts. Additionally, such models are similar to those previously proposed for lower layers. This means that many conclusions and achievements obtained with physical and data-link layer models may possibly be generalized to the network layer. One of such conclusions is indeed that diversity helps in achieving better performance. Translating such benefits in reality is not straightforward, and it involves several issues that range from inaccuracy of channel models, to implementation difficulties in precisely realizing the required scheduling policy.

Besides, the potential of this techniques at network layer has been quickly recognized by the research community, resulting in a number of works dealing with this research problem (see Section 2.3 for a survey). While some have been solved, many issues are still open in this field. This thesis aims at moving further in this direction, towards the use of packet-level diversity in the Internet. Beside the previous motivation, we decided to work at packet level for several additional reasons: i) the losses (and other characterizing phenomena) on the Internet happen at such level; ii) it allows to exploit the flexibility of IP, which is not tight to a particular technology or to a particular application; iii) in the TCP/IP protocol stack, IP represents the lowest layer that is exposed to the users; iv) innovative value-added network services, such as routing overlays and peer-to-peer networks, are generally realized at IP layer.

### 2.2.1 The utility of (informed) path diversity

The availability of multiple paths between a source and a destination is more and more common in the current Internet both at user- and network-side, as reported by several works in literature (see for example [31, 32, 33] and references therein). At user-side, this is due to the use of multiple wired and wireless connections available to modern computers and networks. For example, in office environments it is possible to be connected to the Internet using both a Ethernet and a WiFi network interface. At network-side instead, the use of multiple connections to the Internet, possibly with multiple providers, is a common measure to avoid disconnections caused by problems on one connection.

As introduced in Section 2.1, path diversity is able to improve the performance and reliability of communications even with simple scheduling policies, such as round robin. As for the performance, it is easy to understand how using multiple paths in an integrated fashion results in more bandwidth available to the applications, up to the sum of the bandwidth on the paths. More bandwidth available can be translated into lower end-to-end data-transfer time, i.e. lower time taken to perform a file transfer. Furthermore, using multiple paths in a smart way, which means recurring to measurements to infer the status of the paths, can also be useful to reduce the end-to-end losses, delay and jitter. Our approach to reach these objectives is presented in Section 4.4.

On the other hand, path diversity is useful also because the default path chosen by the routing algorithms between two nodes is often not the best, as reported by several works in literature. As a representative example, the work [36] shows the results of traceroute

measurements collected from a number of hosts around the world. Such results show that in several cases (up to 80%) a path with better quality (smaller round trip time and losses and higher bandwidth) than the default one can actually be found.

Moreover, path diversity can been successfully applied for improving the performance of specific applications. A good example is provided by multimedia streaming. The use of multiple paths has been proposed to overcome the loss [37] and delay [38] problems that affect streaming applications. For instance, when the bandwidth bottleneck is not at the access network, multiple paths (possibly using multiple disjoint routes) may increase the throughput of the streaming flow. Path diversity can also be used as a diversification scheme as it alleviates the unpredictability of Internet path congestions: if a path is affected by a congestion, the server can recover lost packets on one path from the received packets on the other paths using, for example, a FEC scheme.

### 2.2.2 The utility of (informed) time diversity

Packet loss degrades the performance of Internet applications, affecting the quality perceived by the users (the so called Quality of Experience). This is even more true with multimedia streaming applications. In facts, while such applications are less sensitive to the network delay, their performance is dominated by the loss of packets. On the other hand, Internet has being more and more used for audio and video streaming. Even more because current technology allows users to upload their multimedia contents both on- and off-line. YouTube as well as p2p streaming applications such as Soapcast or TvAnts are clear examples of such a trend. Several studies in literature reported the characteristics of the loss process in different environments (backbone [27] and stub networks [29]), with different kinds of traffic (unicast [25] and multicast [26]), different protocols (UDP [25] and TCP [39]), and at different time scales (sub-round trip time [40] or larger [39]). All these studies claimed that losses on the Internet are not isolate but rather happen in bursts. The problem is that the performance of streaming applications, which often use predictive codecs such as MPEG, is more impacted by bursty losses than the by same quantity of isolated losses [41] for different reasons:

- Video sequences are characterized by a high degree of spatial and temporal correlation. Spatial correlation means that different parts of the same frame can be similar. Temporal correlation means that consecutive frames are similar. These properties

are exploited by video codecs to both reduce the bitrate and recover from losses. However, this recovery becomes less and less effective as a larger amount of consecutive frames, or frame parts, is lost.

- Streaming applications often use FEC, which is highly impacted by bursty losses, because it distributes loss-recovery information over a set of packets belonging to the same FEC block. Therefore, it is necessary to receive packets belonging to the same block in order to recover from a loss.

- The loss pattern, with or without using FEC, affects audio and video quality as well as the effectiveness of loss concealment [42].

Interleaving represents a good candidate to cope with this problem, allowing to spread the losses over the multimedia stream without transmitting any additional information.

Streaming applications are not the only ones that suffer from loss correlation and can therefore benefit from the interleaving. Loss correlation has an impact also on congestion control. The more packets are lost consecutively, indeed, the more quickly the congestion control mechanisms reduces the transmission rate. If the losses are random, instead, the control loop of these mechanisms is able to easily detect the situation and achieve a stable rate. For this reason, new TCP version, such as *new reno* [43], use congestion control mechanisms that are able to cope with the bursty nature of packet loss. As explained in details in Section 5.5.5, packet interleaving can be used for the protocols and applications that use congestion control over an unreliable transport protocol [44, 12]. Moreover, it can be used for several applications using UDP, which are more and more used over the Internet [45].

## 2.3 State of the art

In this section, we survey related work in the area of path and time diversity in order to understand what has been achieved, and which problems are still open. Before that, we review some interesting works in the field of network measurements and modeling. The analysis of the literature in this field is useful for two main reasons: i) we also work on methodologies and techniques to infer the status of the network in order to *inform* our diversity techniques; ii) we are interested in understanding the models that have been proposed in literature for packet dynamics.

### 2.3.1 Measuring Internet performance

Research in the area of Internet performance measurement and modeling is very broad and fertile, mainly because its results have a wide range of possible applications (e.g. network planning, problem diagnosis, understanding user behaviour, etc.). In the following we discuss only the works that are more useful for the aim of this study.

In [46], the authors measure the end-to-end delay of paths connecting the 40 sites of the RIPE Test Traffic Measurement Project. The authors study the behaviour of the histograms computed using delay samples collected over the same path. The results show that about 84% of such histograms present a Gamma-like shape with heavy tail.

The authors of [47] describe a passive monitoring system designed to capture GPS-synchronized packet traces on OC-3, OC-12, and OC-48 links of the SPRINT IP backbone. They present results to demonstrate the strength of the system, identify changes in Internet traffic characteristics, and analyze the delay and jitter distributions. They find that the main part of the delay is the speed of light and that the jitter is extremely low.

In [48] and [49], the authors measure and analyze the single-hop packet delay through operational routers, comparing it with well known analytical models. They present results related to about three million packets extracted from 90 gigabytes of data collected from the Sprint IP backbone. They show that the single-hop delay is long tailed and fits a Weibull distribution with scale parameter $a = 0.5$, and shape parameter $b = 0.58 \rightarrow 0.6$ or $b = 0.6 \rightarrow 0.82$.

The authors of [50] present results related to delay and loss of IP packets traversing the University of Auckland Internet access path. The analysis is performed exploiting the peculiarities of the traffic measurement equipment, which comprises a host with multiple DAG cards capturing packet traces at all the devices (routers, switches, and firewall) operating on a certain path. The authors find that: i) the shortage of bandwidth is the main cause for the queueing delays observed at the network devices; ii) the maximum delay observed at the access router is equivalent to typical delays observed between New Zealand and the US West Coast; iii) about 10% of all packets experience delays unacceptable for any real-time communication, such as interactive audio and video.

One of the first works to present an analysis of the self-similarity of Internet packet delay is [51]. This self-similarity implies that the delay experienced by application layer protocols is extremely bursty, and that traditional Poisson models cannot accurately predict the degree of burstiness. Over some selected paths, the authors find evidences that

the degree of self-similarity for a round-trip delay may be correlated with the packet loss observed on the same path. This result has been achieved using tools such as the variance-time plot, R/S analysis, periodogram analysis, and Whittle's estimator.

The work [52] presents evidences that the packet round trip delays exhibit Long Range Dependence (LRD), showing that the complementary probability distribution decays more slowly than the exponential one. The authors also describe the possible implications of this phenomenon, saying that it probably justifies the studies on the prediction of the queue length distribution with LRD network traffic. They also use a multi-queueing system to investigate on the origins of LRD in the packet round-trip delay process, which they believe to be caused by LRD of Internet traffic.

In [53], the authors study the delay characteristics of ADSL service in Korea. They measure traffic delays path-by-path across the whole network to locate the bottleneck, and they study the relationship between delay and packet size and between delay and network utilization.

The work [26] presents a study of the loss experienced by audio packets sent by an audio conferencing tool in both unicast and multicast mode. In this second case, they use the MBone facility. The authors show that the number of consecutive lost packets is small unless the network is highly congested. They then derive a model to represent the loss phenomenon, which is partly based on queueing theory. The paper ends with a discussion on the use of the FEQ or ARQ for this kind of traffic.

In [27], the authors report the results of a measurement campaign conducted on the MBone network. By monitoring the packets received from an Internet radio station by different hosts around the world, the authors study properties of packet loss such as correlation in space (different sites at the same time) and correlation in time (same sites at different times). The results show that the majority of the bursts have lengths smaller than 7 packets.

Paxon presents the results of a large scale TCP measurement in [39]. The obtained results show how some popular assumptions such as in-order packet delivery, FIFO bottleneck queueing, independent loss events, single congestion time scales, and path symmetry are violated in practice. As for the losses in particular, Paxon reports the average values as well as the conditional probability of having a loss after another, which shows that the loss events are not independent.

In [25], the authors present an analysis of the correlation properties of packet loss.

They perform long-term measurements using both unicast and multicast from a host located at the University of Massachusetts at Amherst towards different hosts in USA and Sweden. The results show that: i) there is a high degree of non-stationarity in loss samples; ii) the minimum time after which the samples are uncorrelated is about 1 s; iii) the losses can be modeled by using a Bernoulli model (in 7 cases), a 2-state Markov chain (in 10 cases), or a higher order Markov chain (in 21 cases). They also discuss whether it is better to use an exponential smoothing rather than a sliding window to obtain a good estimate of the average loss rate.

In [29], the authors report passive measures of packet rate, bit rate, and loss from a campus network. With regard to this last parameter, the results show that, while a high loss is observed on highly congested links, losses are also observed on very underutilized links (i.e. utilization of about 5%). The authors say that such losses are probably caused by spikes in the traffic rate that are not observable at the time resolution they used (larger than or equal to 1 second).

Wei et al. in [40] report the results of an analysis of the behaviour of packet loss in sub-RTT scale performed with simulations, emulations, and on the Internet. They show that the loss process is bursty on such time-scale, and they show the implications for congestion control algorithms. They also report on the possible origins of such burtsiness. Moreover, they perform investigations on the fairness between congestion control algorithms based on packet windows and on bit rate.

The work [54] presents an analysis of packet loss and delay as seen by VoIP-like UDP traffic artificially generated between 5 different sites in the USA. The authors, after discussing the main characteristics of the processes of delay and loss, present a model for the latter parameter, which complements the extended Gilbert model from [55] with a parameter called inter loss distance. They show that thanks to this additional parameter it is possible to fully reproduce the distribution of losses. The authors evaluate the quantity and distribution of residual losses after applying different FEC and playout control strategies, showing how such losses still present a bursty behaviour.

### 2.3.2 Path diversity

In 1997, a survey of the existing literature on the use of multiple paths was performed [56]. Such work analyzes the main attempts to exploit the availability of multiple paths mainly to mitigate the performance impairments due to traffic burtiness. The considered works

are mainly concerned with ATM networks because, in that period, such technology was the main focus of a large part of the networking literature. The authors summarize the work done up to then, evidencing the main topics on which the problems were still open.

The work presented in [33] is concerned with the experimental analysis of the possible *path diversity* (with this term the authors refer to the number of routes available to transmit a packet between two hosts in a network) in ISP networks. The authors use an active measurement tool, called *rocketfuel*, to infer the IP level topology of some ISP, and then they compare the results of the tool with real information for one of such providers (i.e. SPRINT). Besides the discussion on the performance of the tool, which is out of the scope of this thesis, the authors show that in large ISP there is significant path diversity.

### Quality assessment and monitoring of video streams

Path diversity is often used to improve the quality of video streamed over the Internet. For this aim, it is important to be able to estimate the expected quality of the received video based on network parameters, in order to decide the path over which to transmit the packets. While most of the already cited works provide some insights on that, several works have been presented in literature to explicitly cope with this issue. In the following we review a few important ones.

The work [57] presents an approach to accomplish this task aimed at scalability, being independent of several application characteristics such as the packetization strategy and the video bit-rate. The authors first introduce a model to derive video quality, which takes into account these characteristics as well as network performance parameters such as the packet loss rate and loss burstiness. Afterwards, they simplify the model in order to get rid of application-dependent information and rely only on network statistics. The accuracy of such model is verified both in simulation and with Internet experiments, which show how increasing the measurement time provides increased accuracy.

On the same topic, the work [58] presents a model using the received bitstream to estimate the mean square error of an MPEG-2 video sequence subject to losses. The technique, called FullParse, partially looks into the payload of the received video packets without performing a full decoding. Performing simulations with real video sequences, the authors verify the accuracy of the model, which seems reasonable, and they claim that the use of the Gauss-Markov model is the main source of error.

### Multipath routing

In [59], the authors overview the main motivations, challenges, and techniques for multi-path routing at different levels (packet, flowlet, flow), and in different operating scenarios (intradomain, interdomain). They discuss the main problems dividing them in forwarding and routing. As for the forwarding problem, beside illustrating currently available solutions, the authors also introduce an innovative technique called *flowlet cache*. With regards to the routing problem, they discuss the possible solutions for intradomain and interdomain routing, analyzing both cooperative and non-cooperative scenarios.

The work [60] presents an algorithm, borrowed from wireless scheduling problems, to distribute packets on different available paths while minimizing delay, maximizing throughput, and respecting the split proportions assigned by the routing algorithm. The paper first presents a theoretical analysis of the performance of the algorithm, which shows that the gain over a weighted round robin increases with the Hurst parameter of the traffic. Through simulations, the authors verify the improvements achievable with respect to weighted round robin, which are higher than the cost of reordering caused by the packet level granularity of the scheduler.

### Media streaming and real time communications

The authors of [61] present different techniques to perform video streaming in presence of performance limitations. The first technique is to be applied when the streaming client is suffering from bandwidth limitations in its access network. Two other techniques are proposed for the cases in which the limitations are not related to the client access network but rather to intermediate links. In such scenarios, the authors present solutions for both the cases of single and multiple senders. In the latter situation, the authors suggest the use of an overlay network in which multiple paths are established through relay nodes. The experiments show how the benefits of using multiple paths diminish as the number of shared links among the paths increases.

A good overview of current approaches for media streaming using multiple paths is reported in [62]. The authors provide motivations to use multi-path for this application underlining the benefits achievable. Moreover, they discuss different coding schemes that are used for media streaming and how they can be exploited in multi-path transmission scenarios. Finally, they analyze the different architectures available for multi-path

streaming such as single- and multiple-senders as well as wireless networks, evidencing the characteristics of such architectures that have major impacts on the achievable performance, such as, for example, the presence of a bottleneck shared among the paths.

The authors of [63, 64] present an approach to identify and follow the best available path for a video streaming application. The work uses a model from literature to predict the quality of a video stream using loss information acquired by probing the available paths. Such loss model is used to evaluate the quality achievable on all the paths. If a noticeable difference is found between the paths, then the path is switched. The authors show the potential of their approach performing Internet experiments.

The work [65] presents a theoretical and simulation analysis of multi-path streaming. The authors assume to have an application streaming a continuous media from different servers to a single receiver. The authors, demonstrate that, in these conditions, the multi-path is able to reduce the average loss rate as well as the loss-burst length. Regarding this parameter, the authors also describe an experiment showing how bursty losses decrease the quality of a video stream more than the same quantity of isolated losses. Furthermore, they analyze the lag-1 correlation of the loss process, which is also reduced with respect to the single path case, when streaming over multiple paths.

The work [38] presents a technique to improve the performance of real-time voice communications using multiple paths. The technique uses a two-stream multiple description coding scheme and, to chose which samples to reproduce, a combination of adaptive play-out and a Lagrangian cost function, which can be used to trade off delay, late loss and speech reconstruction quality. The authors show the results of both Internet experiments and simulations using *ns2*. In the first case they illustrate how their scheme is able to actually reduce the erasure rate (i.e. the rate of packets that arrive too late to be reproduced). The simulations are instead mainly performed to analyze the effect of shared bottlenecks between the paths. With this respect, the authors show that although the common link introduces a high correlation in the delay of the two paths, the packet erasure rate can still be reduced.

### Wireless networks

In [66], an analysis of path diversity at packet level in IEEE 802.11 scenarios with multiple channels is conducted. Such work aims at understanding the benefits obtainable with path diversity in such environments, in spite of the differences between theoretical models

and reality (e.g. model inaccuracy, packet-sending time issues, etc.). The authors verify the effectiveness of a simple round robin policy to distribute packets on the available paths. Furthermore, they analyze the impact of channel switching and of the presence of interfering signals. The results show that, even with a simple packet distribution policy, path diversity is able to provide higher and more stable throughput, and that the impact of channel switching and interfering signals is negligible.

The problem of multi-path routing in ad-hoc networks is analyzed in [67]. In this work the authors propose a routing strategy that uses multiple paths and operates with the packet level granularity. The routes are built on-demand and cached for a fixed period. The algorithm is compared with dynamic source routing in simulation, and improvements achievable in terms of losses, delay and hop distance are presented.

Another theoretical approach for multi-path routing in ad-hoc networks is presented in [68]. The approach is aimed at splitting the available information in several blocks, adding redundancy so as to allow the reconstruction of the original message even if some pieces do not arrive. The authors work under the assumption of independent paths and present, in this work, only results related to a policy that sends the same amount of data on every path.

### Transport layer

Several works aim at distributing packets on multiple paths modifying the transport layer protocol. For example, the work [69] proposes an enhanced TCP scheme which works just above the transport layer, opening different TCP connections towards the same host. The main contribution of such work is the use of a scheduling policy called arrival-time load balancing, which takes into account the buffering and path delays in order to choose the path to send the packets on. The proposal is evaluated in simulation and on a wireless laboratory testbed. The results show higher throughput when the paths exhibit different bandwidths and loss rates. Similar performance is instead achieved when only the delays are different. The testbed experiments show how the proposed technique is able to follow the dynamics artificially introduced on the wireless paths.

Modifications to STCP aimed at effectively using multiple independent paths are proposed in [70]. The authors, leveraging the multi-homing features of SCTP, identify three main issues affecting the performance of concurrent transmissions over multiple paths and propose algorithms to overcome them. The achievable improvements are evaluated in sim-

ulation, comparing the transfer time and retransmissions of the proposed protocol with those of an ideal application that has perfect knowledge of the status of the paths. Finally, the authors illustrate and compare five possible strategies to distribute packets on the available paths, showing that those taking the losses of the paths into account achieve better performance.

### 2.3.3 Time diversity

A survey of many packet loss recovery techniques for streaming audio is reported in [71]. The authors consider techniques operating at sender side and discuss the achievable improvements in the context of audio streaming over multicast networks. Moreover, they discuss on the potential of error concealment techniques to recover from lost audio samples. The authors conclude the paper with a discussion on which technique is better for different application scenarios. Interleaving is found to be the most effective for non-interactive applications.

#### Bit or symbol level

In [72], the authors develop a model that allow to understand the achievable performance with given channel conditions and coding scheme. The model is also able to capture the effect of interleaving on the performance. Thanks to a theoretical analysis, the authors show how the interleaving is effective in decorrelating the losses, asymptotically approaching the memoryless case. They then conclude that interleaving is a very effective tool if the additional delay is acceptable.

A similar analysis is conducted in [73]. In this paper, the authors study the second- and third-order statistics of the residual error process when block transmissions are performed over a bursty channel. Using the well known Gilbert-Elliott model [74, 75], they analytically evaluate the bit error rates for which it is useful to employ an interleaver at the block level.

An extension of the Gilbert-Elliott model is presented [55], where a general model taking into account different loss statistics, and considering the G-E model as a special case, is proposed. In such work, the authors discuss the various parameters of Internet losses that are worth to be considered, most of which are included in their model. The proposal is evaluated with Internet experiments, using several traffic traces collected by an application emulating a VoIP phone.

With regards to wireless networks, an interleaving scheme to alter the order of symbols before applying a FEC scheme is proposed in [76]. Taking into account Bluetooth networks, the authors present both analytical and simulation results that show how changing the order of the bits inside every single packets, and then applying the standard Bluetooth FEC, allows to obtain better performance. Simulation results related to TCP throughput are also presented.

**Frame or packet level**

The work [77] presents a research exploiting interleaving on MPEG encoded video frames. The proposed system acts just before the MPEG encoder applying the interleaving to the sequence of frames, i.e. frame position is altered before encoding. The resulting sequence is more robust to possible frame loss, but the compression is less efficient because some temporal redundancy is lost due to the reordering. The system is evaluated using real videos ranked by real users, showing that the average score is higher for interleaved frames.

In [78, 79], the authors present an interleaving scheme that operates before a predictive video encoder. Basically, the video stream is divided in two sub streams, which are then encoded separately. After encoding, the frames are rearranged in their original position. The fact that they have been coded by separate encoders, however, allows to be more resilient to burst losses as the reference and predicted frames are never consecutive. The decision on which frames to put into each sub stream is made using a Markov Decision Process, whose reward function takes into account the event that frames can not be decoded because a reference frame has been lost.

The authors of [80] perform a simulation study of interleaving applied to layered coding schemes. The proposed solution interleaves base information with enhancement one. The authors also test the usefulness of randomizing the obtained sequences. Simulations are conducted by using loss patterns from real traffic. Results are presented in terms of how many base packets are lost after applying both the interleaving only and the interleaving plus randomization. In most of the cases, the interleaving and randomization are able to actually protect the base information from burst losses.

The authors of [80] refer to [41] for what concerns the study of the channel and the derivation of the loss model. In [41], in particular, they provide information about the effect of bursty rather than isolated losses on videos. They present a model to estimate the distortion caused by losses on videos. Simulation results show how a burst of losses

actually causes more distortion than an equal number of isolated losses.

The work [81] is concerned with the design and verification of a technique to transmit voice over the Internet. The packet receiver continuously informs the sender about the status of the channel, i.e. the loss pattern. The sender then applies a matrix-based transformation to the voice samples in order to compensate the effect of lost samples. Moreover, the voice samples are interleaved by putting the odds ones in one packet and the even ones in another packet. Before explaining the details of the technique, the authors present the results of a measurement campaign aimed at estimating the loss pattern between 6 Internet hosts located in China, Japan, Italy, and USA.

The work [82] is based on the assumption, demonstrated in [83], that the MPEG encoder is more sensitive to isolated losses rather than burst losses. Therefore they propose to apply interleaving at source level (to the bit stream produced by the source encoder), then to apply the FEC, and then to re-apply the interleaving on the obtained symbols. This will protect the stream from channel errors but will result in bursty losses at the MPEG decoder. The reported results show that this scheme is able to increase the peak signal-to-noise ratio of about 5 dB.

### Packet level

In [84], the authors present a simulation framework to study packet interleaving. Results are presented related to two channel models, one with uncorrelated random losses and the other one with correlated losses following a Markov chain. They assume the delays to be Gamma distributed, citing the work [24], and they use a quality function to estimate the benefit of the packet interleaving.

In [85], the authors present a system that performs packet interleaving on MPEG audio. The work contains an interesting introduction to the problem of packet interleaving and a description of the system to both interleave the packets and recover from possible errors. The authors briefly discuss the problem of determining the optimal interleaving configuration, and they present the results of an evaluation performed both in simulation and on real networks, which show the loss decorrelating power of their scheme.

In [86], a study of the effect of packet interleaving on real video sequences is performed. A loss model is used to determine the interleaving configuration most suited to the channel conditions. A delay-distortion optimization problem is set in order to choose the interleaving block sizes taking into account both the maximum acceptable delay con-

straint and the minimal distortion objective. Basically, knowing the maximum delay, they test all the possible configurations and choose the one providing the minimum average distortion according to the loss model.

### 2.3.4 Competing techniques

In [87], the authors propose and evaluate a technique altering the order of packets in a streaming flow such that I and P frames are sent before the related B frames. In this way, even when the delay of the network is such that the I and P frames cannot be reproduced because they are too late, it can still be possible to reproduce the related B flows. This technique is particularly useful in networks with largely varying delays.

Kalman et al. [88] propose a technique that combines adaptive media playout with rate-distortion optimized streaming. Basically, they suggest to employ a control strategy that works coordinating the media streaming server with the receiver. In such proposal the server uses an already proposed system to decide the rate of the streaming flow in order to optimize the total distortion. The client instead employs a technique to decide the playout speed (how long a frame has to be displayed) as a function of the buffer status.

## 2.4 Open problems

The analysis of the literature reported in the previous section has shown that a number of works have been proposed in the area of path and time diversity. However, several issues are still open. In Section 1.3, we have already explained the main contributions of this thesis. In the remaining of this section, we provide a general view of the open issues, referring the reader to the next sections for the solutions devised.

### 2.4.1 Path diversity

As shown in Section 2.3.2, several works have been proposed in literature to exploit the availability of multiple paths in the Internet. Such works differ in terms of application domain (inter- and intra-domain routing, multi-homed hosts, etc.), path switching granularity (packet, time period, etc.), path switching technique (round robin, weighted round robin, etc.), aim (streaming or real time application performance, network reliability, etc.), and operating level (mac, network, transport, etc.). In this thesis we propose to use path diversity at IP layer, independently from the particular application or networking

scenario, and relying on measurements to acquire the status of the available paths in order to make informed decisions. In the remaining of this section we underline the main issues that have to be solved for this aim.

A first problem is related to the switching mechanism. In principle, this problem can be stated in the following way: having perfect knowledge of the status of all the paths with highest possible resolution, how do we chose the best among these paths in order to improve the performance and reliability of the communication ? To answer this question, it is important to take into account that, even if we have perfect knowledge of the status of a network path at a certain time, we can never be sure that such status will remain the same after a certain time. In facts, beside the variations due to the external factors (e.g. background traffic), when we choose a path we are already altering its status by sending new traffic on it. We therefore need an approach that is able to take into account the effects of the decisions made. The situation is even more complicated by the fact that it is practically impossible to obtain perfect information about the status of a path. We can rather try to infer it.

It is clear, then, that two main issues have to be solved in order to choose the best available path: path switching mechanism possibly with feedback information, and imperfect accuracy of path state estimation. As for the first one, the use of a feedback has not been considered for a long time due to the high packet-processing speed required for a path diversity scheme operating at router level (i.e. on the traffic of an entire network). On the other hand, the feedback-based approach has been considered feasible in other operating scenarios such as overlay networks, and multi-homed hosts. Highly related to this problem is the switching time granularity. While frequent switchings allow to better exploit the different performance of the available paths, they also require frequent updates on the path status. Moreover, frequent switchings at IP layer have an impact on upper layer protocols, as they may cause packet reordering, difficult estimation of the path state, etc. All these issues are considered in this thesis proposing two approaches to exploit path diversity called multi-path and path switching, which are formally defined in Section 4.2 and described in details in Section 4.4 and Section 4.5 respectively. The issues related to the correct estimation of the status of a heterogeneous network path are firstly tackled in general in Section 3, and then applied to the specific context of multi-path in Section 4.4.2 and of path switching in Section 4.5.1.

Another important issue is related to the real implementation of a path diversity tech-

nique. In literature, some approaches have been proposed, but there are no clear winners. Therefore, in this thesis we propose techniques that are aimed at being easily adopted in real scenarios because they both are simple and allow to obtain better performance.

### 2.4.2 Packet interleaving

The use of packet interleaving has been advocated by few research works to mitigate the problems caused by bursty losses on some specific application. In this thesis, however, we aim at devising a technique that works at IP layer and is independent on the particular application or network scenario. In this section, we review the main issues that have to be solved in order to reach this target.

A first question that has to be answered is related to how to interleave the packets: given a flow of packets that has to traverse a link with bursty losses, what is the best packet order to cope with such losses ? This problem will be defined as the search for a proper *interleaving policy* in Section 5.2. If we look at what has been done in literature, interleaving techniques can be classified in periodic and pseudo-random. In the first case data is divided in sequences of equal length, using the same interleaving schema for all the sequences. In the second case, instead, pseudo-random sequences are used, in which the position of data in the interleaved sequence is not the same for all the sequences. In practical realizations, periodic interleaving is almost always used due to its simplicity. As we aim at realizing a technique to be used in real scenarios, in this thesis we focus on periodic interleaving. In Section 5.3.2 we describe the interleaving policy used in this thesis which is called *block interleaving*.

The *loss decorrelation power* of block interleaving (i.e. its ability to translate loss bursts into isolated losses) can be easily tuned through the size of the block Increasing this size, the loss decorrelation power increases, i.e. we can deal with bursts that are longer and longer. On the other hand, there are two counter effects: 1) the number of bursts of small length increases; 2) the buffering delay increases. This trade-off represents one of the main aspects to consider when we want to implement the interleaving on a real network. Moreover, such parameter cannot be manually set by an operator because its best value depends on different dynamic factors, such as the channel conditions. Therefore, another important aspects is how to automatically configure a block interleaver performing ad-hoc measurements on the network. The main issue here is the trade-off between accuracy and intrusiveness of the measurements. Therefore, it is important to understand how

to accurately measure the losses, and in general, the status of a network path, without causing disruptive interference to the background traffic. In Section 3, we describe the measurement activities we conducted on real heterogeneous networks, using both passive and active techniques. In Section 5.5.1 we show how to properly set-up a block interleaver in an autonomous fashion, thanks to the acquired knowledge.

Another important issue of time diversity at IP layer is the interaction between the operations performed by the interleaver and the upper layer protocols and applications. As explained previously, the interleaver has to buffer the packets in order to fill the block, and then it has to reorder these packets. Both buffering and reordering operations have an impact, especially on protocols and applications interested in-order delivery and concerned with congestion control. The buffering can indeed be mis-detected as a congestion on the network, with negative consequences on the throughput. The reordering can cause unnecessary retransmissions, which, again, can lower the throughput. We carefully examine these issues, analyzing the interactions between the block interleaving and protocols with congestion control in Section 5.5.4 and with reliability mechanisms in Section 5.5.5 respectively.

## 2.5 Final remarks

In this chapter, we have provided basic information about diversity in general as well as details about time and path diversity. The utility of these techniques has been discussed in order to provide the motivations at the base of this work. We described how these techniques work, using simple application scenarios, and we clarified how these techniques can benefit from accurate and continuous information about the status of the network. We finally reviewed the main results obtained in literature, evidencing the main issues that are still to be solved. In the next chapters, we will firstly see how to estimate the status of a heterogeneous network, and then how to exploit this information in order to achieve better performance with informed time and space diversity.

# Chapter 3

# Inferring the status of heterogeneous IP networks

In this chapter we present the research activities we performed in order to acquire the knowledge, and to devise the methodologies and techniques, for inferring the status of a heterogeneous network, using both active and passive measurement techniques.

## 3.1 Introduction

Inferring the status of a heterogeneous network is not always easy. Difficulties come from several factors: new networking environments employing unknown and proprietary protocols, variability of the measured parameters with external factors (e.g. interference from other networks in the case of wireless links), influence of the measurement tool and methodology, etc.

In our activities we pursued the aim of developing methodologies and techniques able to work in real and heterogeneous scenarios. To this end, we worked on different networks, using both active and passive measurement techniques. As we will see in the following sections, while active measurements allow to obtain detailed and tailored information, passive techniques allow to infer what real users experiments. We can say that each of them as its pros and cons, and it is better suited for a specific purpose. In facts, the informed diversity techniques we present in Section 4 and 5 require both active and passive measurements to obtain all the necessary information.

## 3.2 Active measurements

Measuring network performance with an active technique means injecting probing traffic into the network with a custom tool, and extracting measures from the received packets. Different techniques and tools exist for this aim. For our activities we mainly used D-ITG, which is described in the following, because it provides high accuracy and a large set of configurable parameters. We refer the reader to [89] for a comparison of D-ITG with other active measurement tools.

### 3.2.1 Methodology

In this thesis, we consider a more general framework with respect to existing literature, in terms of both statistical approach and heterogeneous network scenarios (i.e. composed of a large mix of variables regarding the considered end-user device, operating systems, access networks technologies, etc.). We consider a number of configuration parameters as the components of our novel and wide definition of end-to-end path. In more details, in this work we define an end-to-end path (e2eP) as:

$$e2eP = (S_{UD}, R_{UD}, S_{OS}, R_{OS}, S_{AN}, R_{AN}, Protocol, Bitrate) \tag{3.1}$$

where $UD$ identifies the User Devices ($S_{UD}$ at sender side and $R_{UD}$ at receiver side); OS identifies the Operating Systems of each of the two users, $S_{OS}$ at sender side and $R_{OS}$ at receiver side; $AN$ is the Access Network, $S_{AN}$ at sender side and $R_{AN}$ at receiver side; $Protocol$ is the transport protocol the users employ; and, finally, $Bitrate$ is that imposed by the application.

In this heterogeneous scenario, the complexity of assessing the impact of all these parameters arises for two reasons. First, the measurement of a single parameter is already challenging due to the only partial controllability of our testbeds. Second, the parameters have mutual impacts on each other. Therefore, a systematic measurement approach is necessary. Our strategy is to vary only one parameter at a time, perform experiments with a duration that is larger than the expected variation time of that parameter, and repeat the experiments multiple times. Therefore, a large number of measurements have to be performed, to gather a relevant statistical sample space for each aspect to investigate. The measures presented in the following sections were taken over multiple months, several measurements a day, resulting in a large quantity of collected and analyzed data traces.

Such traces were previously inspected and sanitized in order to detect and remove samples affected by errors. At [90] we made freely available several archives containing outcomes of these measurements over real networks.

By integrating and customizing established and well-known tools, we set up a methodology to provide a complete statistical analysis of the collected samples. Along with the evaluation of mean, standard deviation, inter quantile range, maximum, minimum values, we adopted tools for the distribution estimation, tail analysis, auto-correlation function, and entropy measurement. In Section 3.2.2, the motivations to select each of these tools, in the field of heterogeneous network analysis, are provided. To permit the experiment repeatability, and to improve the knowledge in this field, statistical software tools (and, as said before, the data traces) we used are freely available at [90] together with the outcomes of the measurements.

### 3.2.2 Metrics and tools

We mainly analyzed four QoS parameters: throughput, delay (both one-way and round-trip), jitter, and loss. We evaluated several statistics of the samples of these parameters: minimum, maximum, average, median, standard deviation (StDev), and the inter quantile range (IQR). The IQR we used is defined as the difference between the $75th$ and $25th$ percentiles. It is worth noting that average and standard deviation are more useful when analyzed along with minimum and maximum values. And, the IQR and median are better estimators for skewed distributions than respectively the standard deviation and the average value, because they are less influenced by extreme samples. Moreover, we also evaluated the probability density functions (PDFs) of the QoS parameters as well as the auto correlation function (ACF), the entropy, and the distribution-tail behaviour.

As for the temporal correlation, we analyzed the ACF as a function of sample distance (the *lag*), using the correlation coefficient of Pearson. It is defined as:

$$r = \frac{\sum_{i\in[1,n]}(X_i - \overline{X}) \cdot (Y_i - \overline{Y})}{\sqrt{\sum_{i\in[1,n]}(X_i - \overline{X})^2 \cdot \sum_{i\in[1,n]}(Y_i - \overline{Y})^2}} \tag{3.2}$$

where $\overline{X}$ and $\overline{Y}$ represent the mean values of the random variable $X$ and $Y$, $r$ ranges from -1 to +1. In our case, the variable $Y$ is the same of $X$ but shifted by a number of samples equal to the *lag*. When the traces are related to synthetic CBR traffic (i.e. packet size and inter packet time series are constant and therefore perfectly correlated), the more

the ACF values approach 0 the more uncorrelation among packet arrival times has been introduced by the e2e path (of course the path does not influence the packet size).

In order to better understand the variability of the collected samples, we also evaluated the entropy of each trace. Generally speaking, the entropy is a measure of the uncertainty of a random variable $X$. It is defined as:

$$H(X) = -\sum_{x \in X} P(x) \cdot \log_2 P(x) \tag{3.3}$$

where $P(x)$ is the probability of each value $x$. For the sake of comparing entropy values of different configurations, when estimating such parameter, we used the same bin size for all the configurations (i.e. we quantized the samples with a fixed period).

In order to analyze the tail behavior, we plotted the complementary CDF (CCDF) in logarithmic scales, reporting also a line representing the exponential decay (a straight line in logarithmic axes). Such line allows to evaluate if the distribution has a heavy tail, that is, if it decays more slowly that the exponential one. With such a plot, the presence of a heavy tail in the distribution is easily observable. It is important to underline that a close relationship exists between the tail behavior of some QoS parameters (e.g. the delay) and the long range dependence of the traffic. When injecting self-similar traffics into routers, the queue length distribution exhibits a heavy-tail behavior [91].

In order to collect the samples of the cited QoS parameters, we started using different tools such as D-ITG [17], iperf [92], MGEN [93], and RUDE [93]. We compared such tools to verify the possibilities offered by each of them, and to assess the impact of the traffic generator on the measurement results [89]. Such analysis evidenced that D-ITG is better suited to our requisites. It has been therefore chosen to perform most of the analysis reported in the following sections.

D-ITG is a platform capable to produce IPv4 and IPv6 traffic that accurately adheres to patterns defined by the Inter Departure Time (IDT) between packets and the Packet Size (PS) stochastic processes. Such processes are implemented as an i.i.d. sequence of random variables. A rich variety of probability distributions is available: constant, uniform, exponential, Pareto, Cauchy, normal, Poisson and gamma. Also, D-ITG embeds some statistical models proposed to replicate traffic related to various applications: DNS, Telnet, VoIP (G.711, G.723, G.729, Voice Activity Detection, Compressed RTP) and some network games (e.g. Counter Strike). Measurements of One Way Delay (OWD),

Round Trip Time (RTT), packet loss, jitter, and throughput can be performed. For each generation experiment, it is possible to set a seed for the random variables involved. This option gives the possibility to repeat many times exactly the same traffic pattern by using the same seed. Also, D-ITG permits the setting of the TOS (DS) and TTL fields of the IP header.

D-ITG allows to perform measurements at both sender and receiver sides. Additionally, D-ITG enables the sender and receiver to delegate the logging operation to a remote server. This option is useful when the sender or the receiver have limited storage capacity (e.g. PDAs, Palmtops, etc.), or when a central point of log-collection is required. Also, it can be used to analyze log information "on-the-fly". Another feature is that the sender can be remotely controlled by using ITGApi. This means that D-ITG sender can be launched in daemon mode, waiting for instructions on how to generate traffic flows.

Before the activities described in this thesis, D-ITG was available for Linux, Windows, and Linux Familiar [1] platforms. For our aims, we ported D-ITG to SnapGear [2] [19], Montavista [20], and OpenWRT [21] operating systems, which are typically used by networked embedded systems such as Access Points, SOHO Routers, etc. Moreover, we implemented SCTP [22] and DCCP [12] support. Finally, we also added several new features for traffic generation (e.g. new traffic profiles) and analysis (e.g. new statistics of the QoS parameters extracted from the log files) [95].

Several tools have also been developed for post-processing the collected data. Most of them are publicly available at [90].

### 3.2.3 Analyzed networks

#### PlanetLab Europe

As of today, the most relevant large scale distributed testbed for networking research is PlanetLab [96]. PlanetLab is a geographically distributed testbed for deploying and evaluating planetary-scale network applications in a highly realistic context. Nowadays, the testbed is composed of more than 900 computers, hosted by about 400 academic institutions and industrial research laboratories. Since its initial deployment in 2003, PlanetLab has become of indisputable importance for the academic research community and industry, which use such infrastructure for the evaluation of a wide range of distributed

[1] A porting of Linux for ARM-based palmtop devices.
[2] An operating system for Intel XScale-based network processors [94]

systems. Examples of scenarios that have been tested on PlanetLab are peer-to-peer systems [97], overlay [98] and content-distribution networks [99], and network measurement frameworks [100]. Fig. 3.1 shows a conceptual view of the current architecture of the PlanetLab testbed, whose node set is composed by the union of disjoint subsets, each of which is managed by a separate authority. As of today, two such authorities exist: one located at Pricenton University (PLC) and another located at Université Pierre et Marie Curie UPMC in Paris (PLE).

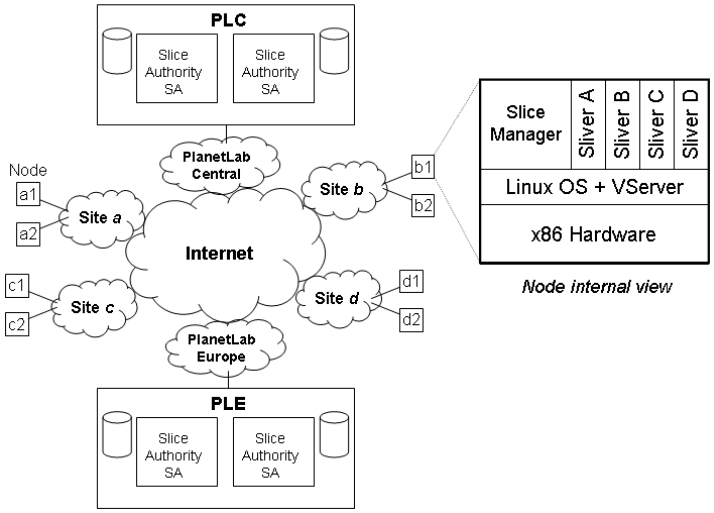



Figure 3.1: Conceptual PlanetLab architecture.

One of the main limitations of PlanetLab is the lack of heterogeneity, that jeopardizes the effectiveness of such infrastructure in providing realistic results from the experimentations [101], [102], [103]. Nearly all PlanetLab nodes are server-class computers, connected to the Internet through high-speed research or corporate networks. This contrasts with the real Internet, in which connected devices also comprise, for example, laptops, handheld computers, cellular phones, network attached storage devices, video-surveillance appliances, as well as household devices. Besides, all such systems are connected to the Internet through a variety of access technologies, both wired (ADSL, CATV) and wireless (WiFi, GPRS, UMTS, CDMA , etc.). As a consequence, it has also been noted that the behavior of some applications on PlanetLab can be considerably different from that on the Internet [104]. In order to introduce more heterogeneity into the PlanetLab architecture, we participated to the activities of ONELAB project [105]. Working in this project,

we added UMTS connectivity to PlanetLab Europe nodes [14, 106]. More precisely, our aim was to provide every node of the testbed with the possibility of using a UMTS interface, and of handling the related Point-to-Point Protocol (PPP) connection. While this task can be easy in a common Linux box, it is not as easy in a PlanetLab node. This is because such nodes have a modified kernel and also because very limited privileges are granted to normal users.

Our UMTS extension has been initially implemented and validated in a private PlanetLab deployment. After testing and validating our extension in this testbed, we have made it publicly available in PlanetLab Europe. It is also worth noting that our main goal was not to integrate a specific UMTS network into PlanetLab, but rather to allow PlanetLab institutions to equip their nodes with such kind of connectivity using a Telecom Operator of choice. In principle, this allows to perform experiments by using the UMTS connection provided by different networks and to compare the results.

### Magnets

The Magnets testbed aims at deploying a next-generation high-speed wireless access infrastructure in the city of Berlin. The network is designed as a wireless network supported by an operator to perform research, but access is given for free to the students of the Technical University of Berlin to create the semi-productive environment. The Magnets architecture consists of three basic network parts: (i) a high-speed 802.11-based *wireless backbone* consisting of point to point links; (ii) an 802.11-based wireless access *mesh network*; and (iii) a *heterogeneous access network* that combines multiple wireless technologies, such as GPRS, UMTS, WiFi and WiMax [107, 108, 109].

The details of the Magnets WiFi backbone deployment in Berlin are depicted in Fig. 3.2. All nodes reside on top of high-rise buildings and have unobstructed line of sight. The distances between the buildings varies between 330 m and 930 m, resulting in a total span of approximately 2.3 km between T-Labs and T-Systems. All transmissions are in the unlicensed ISM spectrum (2.4 and 5 GHz) and all backbone components (antennas, access points) consist of off-the-shelf hardware supporting both 802.11a and 802.11g modes at 54 Mbps. A proprietary protocol enhancement called *Turbo Mode* can be optionally switched on to double the gross throughput to 108 Mbps. Each Magnets backbone node consists of an Intel P4-PC based router with a 3 GHz processor, 1 GB of RAM and 80 GB HDD, running Linux with kernel version 2.6.15. 12 LanCom WiFi

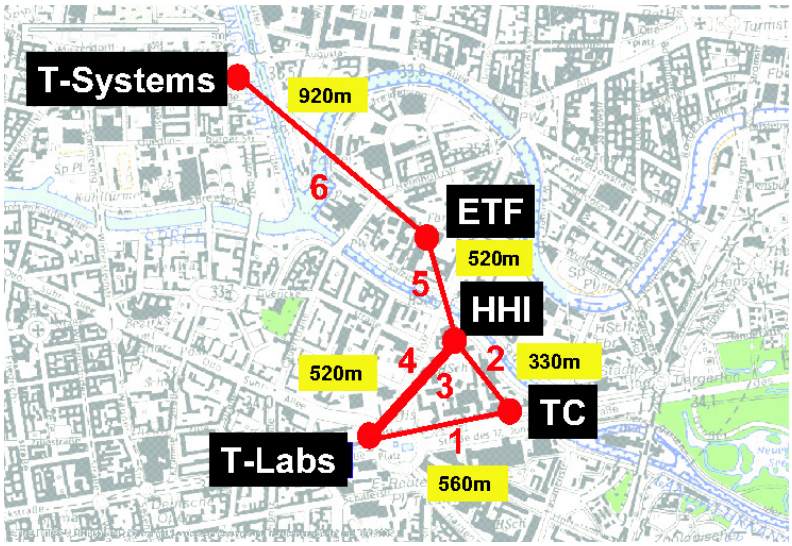



Figure 3.2: Magnets WiFi backbone.

access points (APs)[3] are attached to the routers. Of particular interest is that 2 parallel links are deployed between T-Labs and HHI. These parallel links (referred to as "twin links") are independent as they have their own APs and antennas and the distance between their respective antenna centers on the masts is roughly 50 cm. Routing can be configured to consider the twin links as two distinct paths or to load-balance traffic flowing through them. The access points are based on Intel IXP420@266 MHz (IAP) and IXP425@533 MHz (OAP) programmable network processors (NP) respectively. APs based on the IXP4xx series architecture have shown to outperform APs based on Broadcom or IBM processors from the same class [110]. The access points are connected to directional antennas. 8 antennas operate at 2.4 GHz with a gain of 18 dBi, 4 operate at 5 GHz with 9°/23 dBi.

### Hetnet

Hetnet is a heterogeneous laboratory test-bed [14, 111]. As shown in Fig. 3.3, it comprises different operating systems (i.e. Windows, Linux, Linux Familiar, Montavista, Snapgear, OpenWRT), end user devices (i.e. Laptop, Palmtop, Workstation, Desktop PC, Embedded device) , and access networks (LAN, IEEE 802.11 in infrastructure and ad-hoc modes, ADSL, GPRS, UMTS). Therefore it allows to perform experiments aimed at characterizing

[3]http://www.lancom-systems.de

several networking scenarios, and at understanding the impact of all these variables on the performance.

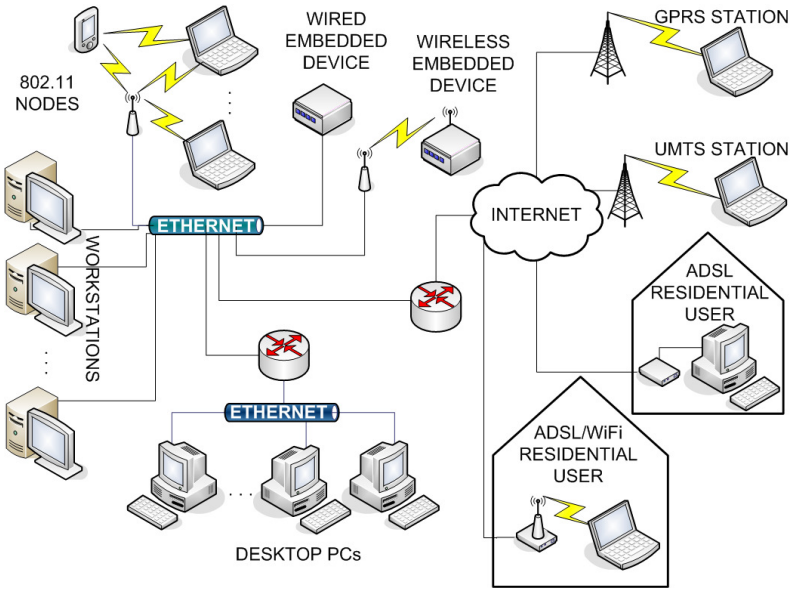



Figure 3.3: Hetnet: a heterogeneous laboratory testbed.

The testbed has been partly deployed in the Dipartimento di Informatica e Sistemistica (Department of Computer Science and Systems) of the University of Napoli Federico II, and partly in the Item Lab of the Consorzio Inter-universitario Nazionale per l'Informatica (CINI) in Napoli. Most of the parts of the testbed (end hosts, switches, and routers) are under the control of the operator. Some others (ADSL, GPRS, and UMTS connections) are part of the network of Italian telecom operators. This allows, on one hand, to tune the parameters of the testbed as required by the operator, and, on the other hand, to asses what real users experiment.

### 3.2.4 Analysis and results

In this section, we report the results of our active measurement analysis on the three network scenarios described before.

#### PlanetLab Europe

**Analysis** In this section, we show how it is possible to experimentally study the performance of a UMTS link using both PlanetLab nodes and the tools we developed (described

in Section 3.2.3). For this analysis, we equipped one of our PlanetLab nodes (located in Italy) with a commercial UMTS connection provided by one of the major 3G operator in Italy. Even if this analysis is far from being a through performance study, it constitutes a use case for the introduced features, providing some interesting insights into the UMTS connections commonly used. In more details, we performed two kinds of experiments: (i) we assessed the performance of the UMTS up-link in non-saturated conditions, i.e. using low-bitrate traffic, with both UDP and TCP; (ii) we measured the the maximum throughput achievable in the uplink direction (that with the lowest capacity) of a UMTS connection using both UDP and TCP, also evaluating other performance indicators in such saturated conditions.

To have a performance reference, all the experiments have been performed also using the Ethernet connections of the same machines. This means that, for each experiment, we measured the characteristics of two network paths connecting the same couple of hosts: the first path connects the UMTS interface of the PlanetLab host located in Italy to the Ethernet interface of the PlanetLab host located in France (we refer to this path as "*UMTS-to-Ethernet*"); the second path connects the Ethernet interface of the host located in Napoli to the Ethernet interface of the host located in France (we refer to this path as "*Ethernet-to-Ethernet*"). This allowed to pin-point which of the observed behaviors were depending on the UMTS connections, and therefore discover the real causes of the obtained results.

We compared the characteristics of the two end-to-end paths by using two traffic classes carried by both UDP and TCP. The first one was obtained by generating a low- and constant-bitrate traffic resembling the characteristics of a real VoIP call using codec G.711 (i.e. 72 Kbps obtained with packet size equal to 80 Bytes of voice samples + 12 Bytes of RTP header = 92 Bytes of payload, and packet rate equal to 100 pps). The other one was obtained by using 1-Mbps CBR traffic (packet size equal to 1024 Bytes and packet rate equal to 122 pps). The former kind of traffic allows to assess the performance of the UMTS in non-saturated conditions, and the feasibility of a voice call over the UMTS, also comparing the achievable quality with that on a high capacity link ("*Ethernet-to-Ethernet*" path). The latter kind of traffic, instead, allows to analyze the behavior of the UMTS up-link in fully saturated conditions (as we will see in the following 1-Mbps is more than double the capacity of the UMTS uplink). All the experimented lasted for 120 seconds and were repeated 20 times, obtaining very similar results.

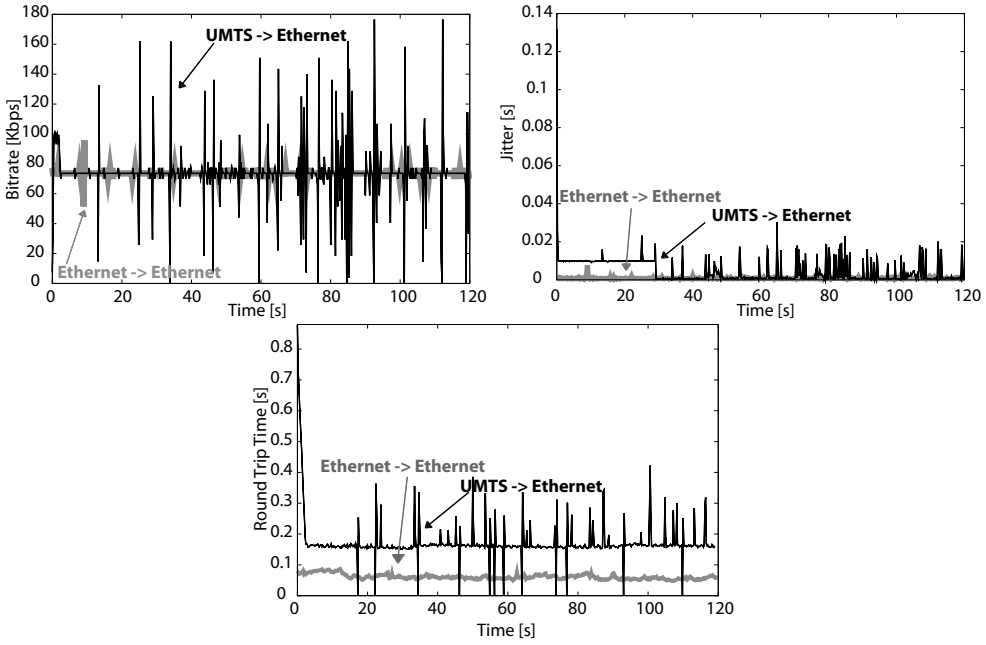



Figure 3.4: Bitrate, jitter, and RTT of UDP in non-saturated conditions.

After the traffic generations ended, we retrieved the log files from the two nodes and we analyzed them by means of ITGDec, another component of D-ITG suite. In this way, we obtained the samples of four QoS parameters that are bitrate, jitter, loss, and round-trip time (RTT). Such samples represent the average values calculated over non-overlapping windows of 200 milliseconds. In the following subsections, we show the time plot of the samples of such parameters obtained with the two traffic classes, using the two network paths, and for both UDP and TCP. For all the tests with TCP, the packet loss results are not reported because we are interested in observing the performance perceived by the applications and, with TCP, the losses are not visible at application layer.

**Results of UDP in non-saturated conditions** Fig. 3.4 shows the time plots of the bitrate, jitter, and round trip time. The packet loss is not reported for this experiment because it was always equal to 0. Each figure contains the results obtained on both "*UMTS-to-Ethernet*" and "*Ethernet-to-Ethernet*" paths. As we can see, at first look, all the parameters present different behaviors in the two cases.

In more details, we observe that the bitrate of the UMTS connection is more fluctuating than in the Ethernet case, even though, in both cases, the required imposed value is achieved in average. The jitter plot, instead, shows that the UMTS connection introduces

a higher jitter, which is also more fluctuating. It reaches values up to 30 milliseconds, which, however, can be easily compensated and still allow a VoIP communication to be satisfactory for the users [112]. Moreover, we can observe that such high values are obtained only in the first seconds of the communication, after which the jitter is much lower. This behavior will be explained in the following section. Finally, looking at the round-trip delay, we can observe that the average value is higher for the UMTS connection with respect to the Ethernet one. Moreover, as seen for the jitter, the RTT is more fluctuating on the wireless connection and it reaches values up to 700 milliseconds.

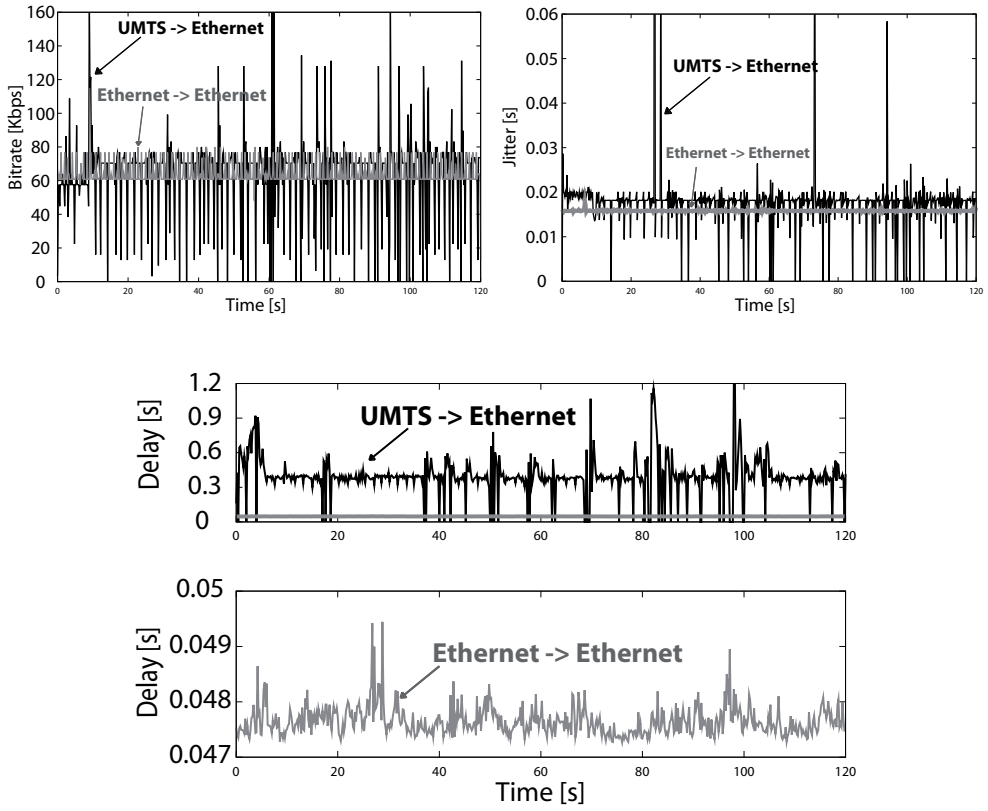



Figure 3.5: Bitrate, jitter, and RTT of TCP in non-saturated conditions.

**Results of TCP in non-saturated conditions** To study the case of non saturated conditions with TCP in a real scenario, we analyze the results we obtained with the VoIP-like flow and TCP. The use of TCP for both voice and video of traffic has highly increased in the last years [113]. This is mostly because, with respect to UDP, TCP can more easily traverse firewalls and NAT boxes, and the bandwidth available on many Internet paths

allows to overcome the delays introduced by such protocol.

In the left upper plot of Fig. 3.5, we report the throughput obtained with TCP in non saturated conditions. As we can see, the required value (i.e. 72 Kbps) is achieved in average even if the UMTS presents a more spiky trend with respect to the Ethernet. The right top plot of Fig. 3.5 shows the jitter obtained with TCP. It can be noticed that the maximum jitter can be as high as 60 ms. This value, however, is reached only in very few periods while in most cases the values are about 20 ms. This means that, as far as the jitter constrains are concerned, the UMTS connection allows to perform a VoIP call using TCP too. The RTT of the UMTS in non-saturated conditions observed with TCP is reported in the bottom plot of Fig. 3.5. If we compare these values with those obtained with UDP (Fig. 3.4), we can observe the impact of TCP. Such protocol, indeed, causes RTT values nearly double with respect to UDP, and reaching an average value of about 350 ms and maximum values of more than 1 s. This means that it is very difficult to perform a VoIP call using TCP, at least with a satisfactory quality. On the contrary, looking at the bottom graph of Figure 3.5, we can observe that on the *Ethernet-to-Ethernet* path the RTT never reaches 50 ms, which allows to perform multimedia communications with a good quality [112].

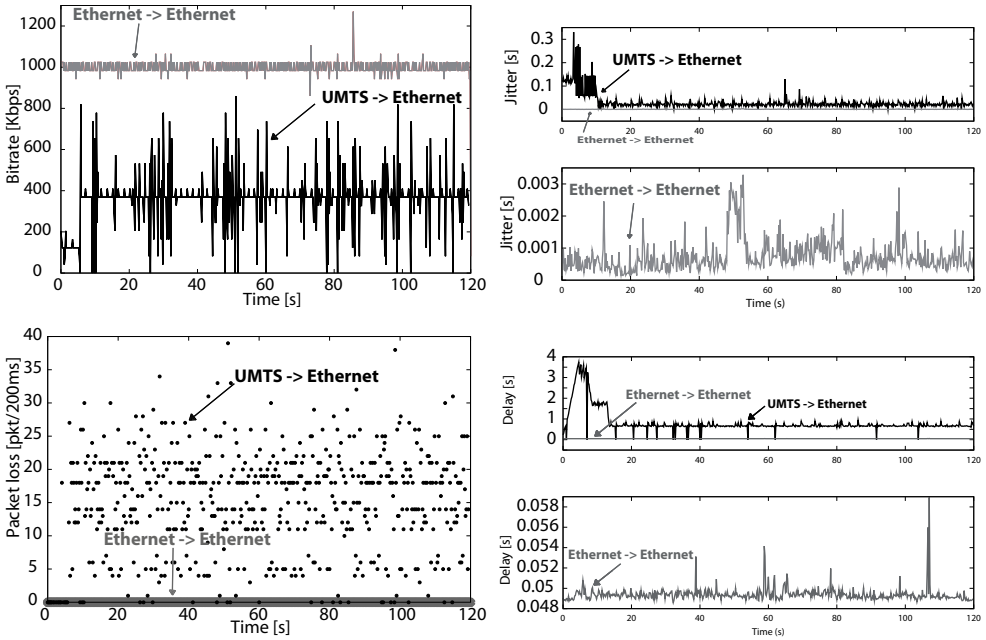



Figure 3.6: Bitrate, jitter, loss and RTT of UDP in saturated conditions.

**Results of UDP in saturated conditions** Fig. 3.6 shows the bitrate, jitter, loss, and round trip delay we obtained on both end-to-end paths. This is to show the differences between the characteristics of these two network connections in the case of a 1-Mbps flow, which clearly saturates the up-link of the UMTS connection but not the Ethernet. As a first consideration, we observe that the bitrate of the UMTS reaches a maximum value of around 400 Kbps. Such value is lower than the one requested, and it is therefore representative of the maximum capacity of the up-link of the UMTS connection. The jitter, packet loss, and round-trip delay plots show the very low performance achieved by the UMTS connection in saturation. In particular, we can see that the jitter reaches values larger than 200 milliseconds, which makes a real time communication nearly impossible. This is also confirmed by the values of the RTT which can be as large as 3 seconds.

Besides, looking more closely at the figures we observe that for the UMTS connection there is a first time period in which the situation is much worse than in the rest of the time. In particular, if we look at the top left plot of Fig. 3.6, we can see that, in the first 5 seconds, the achieved bitrate is about 150 Kbps. After that time, instead, the bitrate is more than doubled. This is due to an adaptation algorithm implemented by the UMTS network, which allocates the network resources to the users in a *on-demand* fashion[4][114]. The change of link characteristics has also an effect on the other parameters, which have an increment of the performance after the first 5 seconds.

**Results of TCP in saturated conditions** In the top left plot of Fig. 3.7, we report the bitrate obtained with TCP in saturated conditions. In such case, we are generating a bitrate that is higher than the capacity of the UMTS link. Therefore, as shown in this figure, the imposed bitrate is never achieved, while the average obtained value is around 400 Kbps and the plot is very spiky. In the *Ethernet-to-Ethernet* case, instead, the requested rate is achieved and the trend is smooth. The obtained jitter is reported in the top right plot of Fig. 3.7. As we can see, for the *UMTS-to-Ethernet* case, the maximum value is about 100 ms and spikes are present for the entire duration. Instead, for the *Ethernet-to-Ethernet*, the trend is very smooth, and the values are lower than 5 ms. Finally, the bottom plot of Fig. 3.7 the RTT values are reported. For the *UMTS-to-Ethernet* path, the average of value is high (about $1.5s$). This makes it impossible to

[4]Basically, a dedicated channel is assigned to the mobile station if necessary. Otherwise, a shared channel is used.

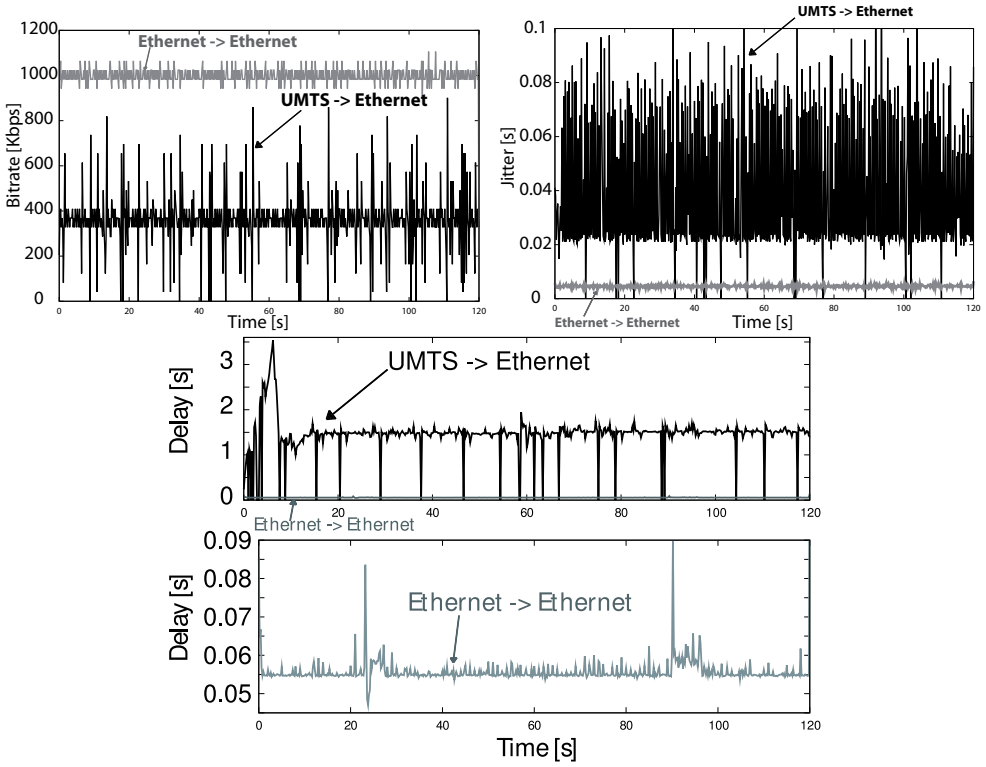



Figure 3.7: Bitrate, jitter, and RTT of TCP in saturated conditions.

have satisfactory communications.

### Magnets

**Link measurements** As a baseline for the subsequent high-speed measurements, we first assess the performance of each link individually in its basic configuration, i.e. with 802.11a/g as defined in the standard. We generate UDP traffic for 600 seconds at the source node using Iperf at 70 Mbps, which is well above the available data rate and therefore saturates the link. At the receiving node, packets are captured using tcpdump. To calculate the bandwidth, the raw trace is sampled at 50 ms interval and the bytes received in this interval are summed up. Tables 3.1(a) and 3.1(b) show the mean and standard deviation of throughput respectively. Link 1 outperforms the others with an average throughput of 31.3 Mbps. Moreover, the low standard deviation of 0.9 Mbps indicates that the link is very stable. Next, links $2-4$ have an average throughput between 6.2 and 12.2 Mbps. These links operate in the 2.4 GHz range; the throughput degradation is attributed to interference. Finally, links 5 and 6 are the weakest links, with

an average bandwidth of 4.3 and 5.4 Mbps respectively. Link 5 has strong interference because the ETF building is lower than the others, and link 6 spans a much larger distance with 930m.

Table 3.1: Influence of Turbo- and Burst Mode on UDP throughput.

(a) Mean throughput (Mbps).

| Link | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Basic | 31.3 | 12.2 | 8.4 | 6.2 | 4.3 | 5.4 |
| Burst | 34.2 | 14.1 | 14.2 | 12.8 | 4.5 | 5.7 |
| Turbo | 53.8 | 22.3 | 39.1 | 38.4 | 6.2 | 12.7 |
| Both | 62.4 | 24.2 | 50.3 | 51.2 | 8.4 | 13.8 |

(b) Stdev (Mbps).

| Link | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Basic | 0.9 | 8.2 | 1.0 | 2.1 | 3.2 | 2.1 |
| Burst | 1.2 | 8.6 | 2.1 | 3.0 | 4.1 | 3.3 |
| Turbo | 3.6 | 14.5 | 5.8 | 5.3 | 5.3 | 8.2 |
| Both | 1.7 | 18.2 | 4.9 | 5.7 | 6.0 | 9.5 |

Now we can assess the impact of *Turbo-* and *Burst Mode* on the link performance using Iperf. Even though the reference manual indicates a doubling of the throughput via *Turbo Mode*, and an increase of 10 Mbps with *Burst Mode*, it is not obvious how these modes impact the link characteristic of Magnets. In Tables 3.1(a) and 3.1(b) we report the throughput obtained with these modes enabled, for all the links The *Turbo-* and *Burst Mode* increase the throughput such links significantly. For link 1, for instance, the throughput increases from 31.3 Mbps with basic mode, to 34.2 Mbps with *Burst Mode*. *Turbo Mode* boosts the throughput to an average of 53.8 Mbps. Finally, with both modes enabled, the average throughput reaches 62.4 Mbps! Link 3 also presents throughput gains with *Turbo-* and *Burst Mode*. The corresponding rates are 8.4, 14.2, 39.1, and 50.3 Mbps. Note here that the improvement with *Turbo Mode* is more than twice the base rate. With all other links, the results are comparable to link 3. Therefore, we argue that the Magnets backbone is able to support a substantial amount of traffic.

An important issue in wireless multi-hop networks is fairness. It is well known that the throughput of flows in multi-hop wireless networks is biased towards flows which traverse few hops [115]. Figure 3.8 shows the fairness problem in Magnets. We inject TCP traffic into the backbone at TLabs targeted to all other nodes. The y-axis shows the measured throughput as a function of the time. Two observations are important. First, the use of directional antennas and the ability to send and receive at the same time mitigates unfairness at the MAC and PHY layer. Therefore, unlike in the scenarios reported in [115], all flows have a throughput $> 0$. However, second, the low throughput of links 5 and 6 and the large RTT lead to a dismal performance of the 3- and 4-hop flow.

A particular feature of the backbone are 2 parallel links between T-Labs and HHI (links

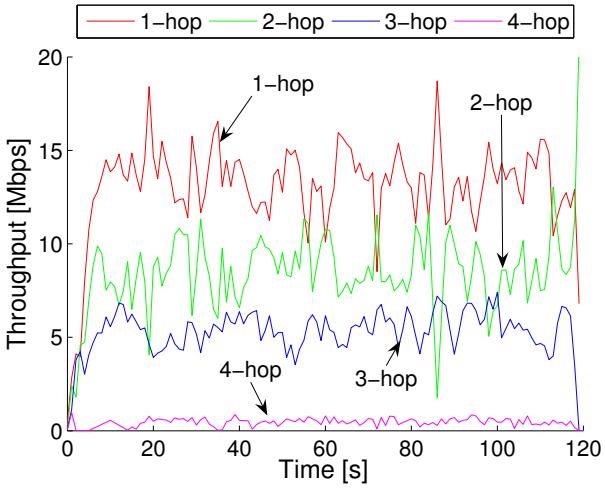


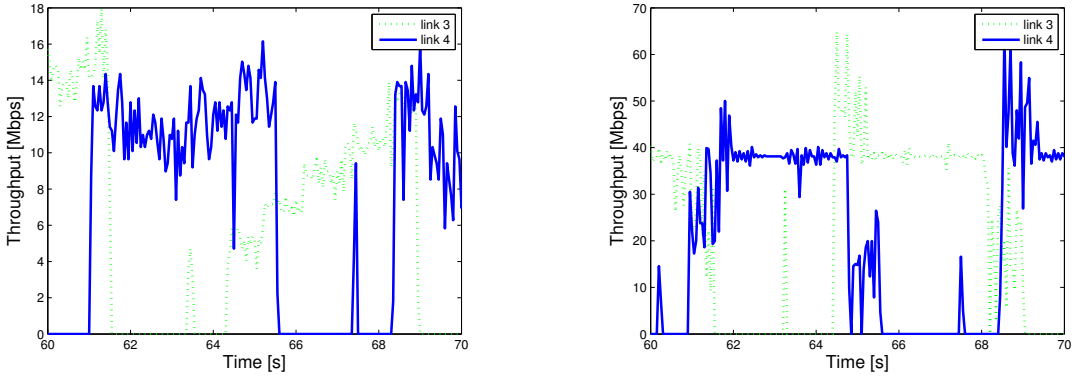

Figure 3.8: Multi-hop TCP measurements.

3 and 4). Two parallel links can be useful in several scenarios(load balancing, redundancy, etc.) Figure 3.9 shows the per-link performance when both links are simultaneously activated. Fig. 3.9(a) shows the per-link throughput in basic mode when the links are set to orthogonal frequencies (link 3 to channel 6, link 4 to channel 13), and Fig. 3.9(b) shows the throughput with Turbo- and Burst Mode enabled (with Turbo Mode, both links are automatically set to channel 6). We see that in basic mode, the links can be enabled simultaneously, e.g. between 64 and 66 sec. However, we can see the effect of the interference: when one link reaches 6 Mbps, the other link goes down. As shown in first two lines of Tables 3.2, the average throughput reaches 23.1 Mbps and 22.8 Mbps, compared to 8.4 and 6.2 Mbps in the basic mode. If we can compare these results with the single link activation results in Table 3.1(a), we can see that the total UDP throughput when both links are enabled ($14.6 = 8.4 + 6.2$ *Mbps*) is exactly the sum of the throughput obtained when each link is independently activated ($14.6 = 8.4 + 6.2$ *Mbps*). When we enable the special modes, the sum of the throughput we obtain ($45.9 = 22.8 + 23.1$ *Mbps*) is lower than the throughput of each of the two links obtained in the previous tests (50.3 and 51.2 *Mpbs*). Concluding, in order to achieve higher throughput, it is better to use just one link with *Turbo mode* enabled. The channel of a single link is sufficiently stable to support a high data rate (Table 3.1). The antennas are not sufficiently separated, therefore the link transmissions cause high mutual interference (mainly at receiving side). When *Turbo mode* is not available, two parallel links (operating at orthogonal channels)

Table 3.2: Twin links under simultaneous activation.

| Link | Basic | Burst | Turbo | Turbo+Burst |
|---|---|---|---|---|
| 3 UDP | 8.4 | 13.0 | 19.0 | 23.1 |
| 4 UDP | 6.2 | 6.9 | 20.1 | 22.8 |
| 3 TCP | 7.3 | 14.4 | 16.9 | 23.0 |
| 4 TCP | 5.8 | 9.0 | 14.2 | 27.0 |

can almost double the throughput and can therefore be used to increase the network capacity.

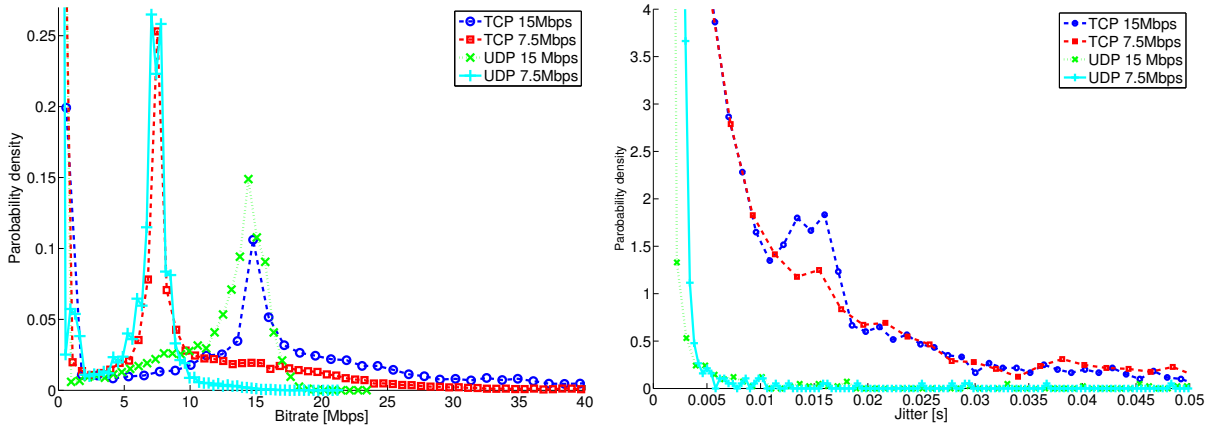



(a) Basic Mode. (b) *Turbo+Burst Mode*.

Figure 3.9: Timeplot of UDP throughput under simultaneous activation.

**Application-level traffic measurements** The previous analysis was characterized by that fact that, in every measurement interval, a single traffic flow was present in the network (every cause of interference with other flows was intentionally avoided). The results we discuss here, are related to measurements performed with a varying number of concurrent traffic flows. In particular, we first present results related to traffic generated with 4 CBR flows, and than with 12 VBR flows. We firstly generated 4 concurrent CBR traffic flows: 2 TCP and 2 UDP flows. For both protocols, two different throughput were adopted, that are, 15 Mbps and 7.5 Mbps. The 15 Mbps flows were obtained sending 512 Byte packets at a rate of 3667 pkt/s, while the 7.5 Mbps flows were produced sending packets of the same size (512 Byte) at half rate (1833 pkt/s). The total throughput we injected into the network is about 60 Mbps, which causes Magnets to be in saturation status. Figure 3.10(a) and 3.10(b) depict the PDFs of throughput and jitter samples

respectively. Figure 3.10(a) shows that TCP flows present a heavy upper tail caused by the packet retransmissions. Such mechanism allows TCP flows to sustain, in average, the imposed throughput causing a maximum throughput of about 56 Mbps (greater than the imposed average values, 7.5 Mbps and 15 Mbps). This could have a severe impact on policy mechanisms applied to the network (e.g. shaping). Also, TCP retransmissions cause throughput samples to have higher entropy. All these considerations are not true for UDP samples, whose PDFs decay more rapidly to 0. UDP flows react to the congestion loosing packets and, therefore, their throughput is lower than the imposed value. Figure 3.10(b) shows the PDFs of jitter samples. TCP distribution decays slower than UDP. This causes higher mean, median, standard deviation, and IQR values compared to UDP values.

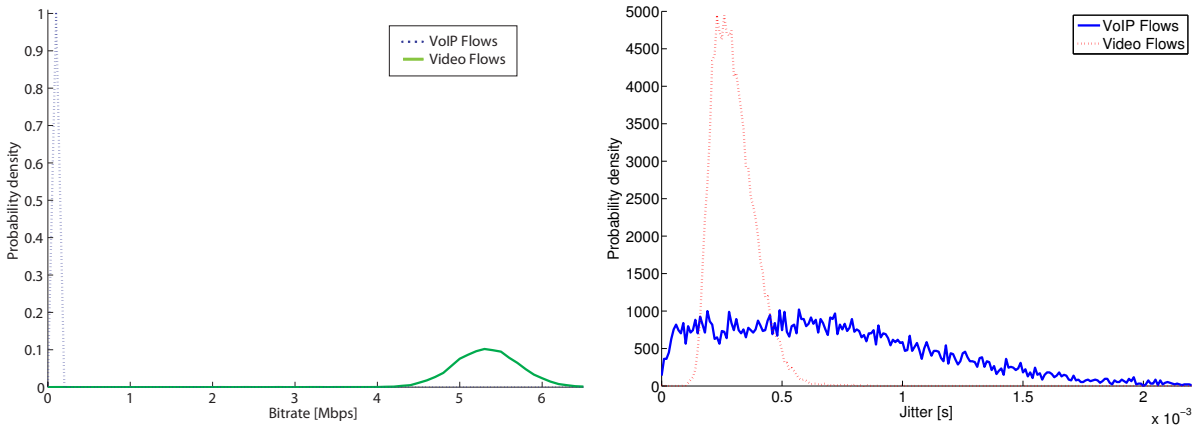



(a) Throughput (pdf). (b) Jitter (pdf).

Figure 3.10: Impact of CBR multi-sources.

We then assessed the behavior of application-like traffic. In particular, we performed 20 measurements of 2 minutes. During each measurement, 12 concurrent flows were injected in the network. The first 8 flows are representative of UDP video streaming traffic. According to model proposed in [116], they were generated with a constant IDT ($24 frames/s * 30 pkt/frame = 720$ pkt/s) and a Normal PS ($\mu = 926.4 Bytes$ and $\sigma = 289.5 Bytes$). The remaining 4 flows are representative of CBR VoIP traffic flows codified using ITU G.711.1 codec. This kind of traffic was characterized by a PS equal to $92 Bytes$ ($80 Bytes$ of RTP payload plus $12 Bytes$ of RTP header) and an IDT equal to $100 Pkt/s$. The average total throughput we injected is equal to about 45 Mbps. For this reason

the link was close to saturation status. Figure 3.11(a) and Figure 3.11(b) show the PDF of the throughput and jitter of the two types of traffic. Considering that the total generated traffic is about 45 Mbps, we can state that Magnets provides very satisfying results. Interestingly, Magnets is able to accurately transport all the sent packets. Also, the average jitter of the VoIP flows is 7.21 ms. This means that, according to the values reported in [15], the link is able to carry real-time traffic at very high bit rates. The compliance with [15] is also confirmed by the statistics of the packet loss samples. The average packet loss is indeed 0.58 and 0.08 pkt/s for Video and VoIP flows respectively. Which means, in percentage terms, about 0.08% for the both kinds of traffic.




(a) Throughput (pdf). (b) Jitter (pdf).

Figure 3.11: Impact of VBR multi-sources.

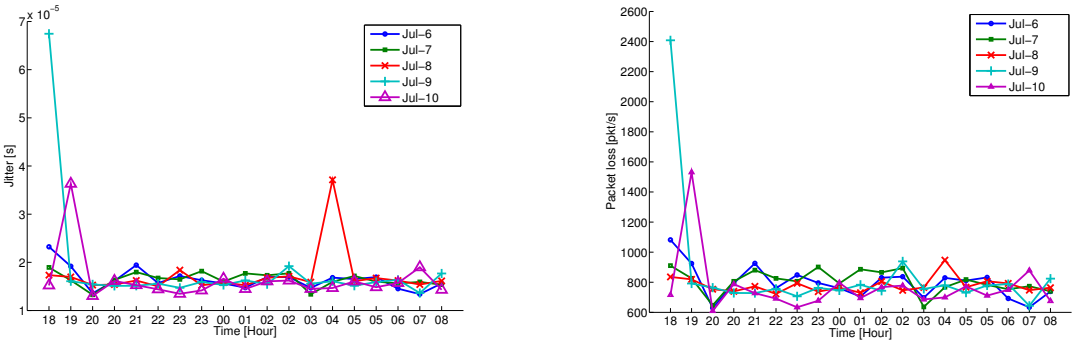
**Impact of environmental factors** The measurements we showed before are just single snapshots of the backbone conditions. However, we ignore how the performance of Magnets varies over larger time scales, e.g. between day and night or due to special environmental influences. To assess this impact, we performed a *24h* measurement. On link 3 we measured the application-level throughput, delay, jitter and packet loss in 33 measurements performed every 45 minutes and using D-ITG. In each measurement, UDP traffic was injected for 120 sec at a rate of 10000 pkt/s with a payload of 512 Bytes, i.e. at rate of 41 Mbps. The resulting traces are sampled at 50 ms. To assess the day time impact, we define the period between 7:45 a.m. to 9:15 p.m as *day*, and the rest as *night*. In Table 3.3 several statistics of the samples of throughput, jitter, packet loss, and delay are

Table 3.3: UDP traffic: concise statistics of the *24h* trace.

| | Mean | Min | Max | Median | StDev | IQR | Entropy |
|---|---|---|---|---|---|---|---|
| Bitrate [Mbps] | 37.23 | 0.00 | 45.47 | 36.86 | 3.36 | 4.18 | 5.10 bit |
| Bitrate day [Mbps] | 36.85 | 0.00 | 44.24 | 36.78 | 3.61 | 4.26 | 4.95 bit |
| Bitrate night [Mbps] | 37.90 | 0.25 | 45.47 | 37.19 | 2.73 | 4.10 | 4.55 bit |
| Jitter [s] | 1.96e-005 | 0.00e+000 | 5.08e-003 | 1.50e-005 | 3.67e-005 | 1.00e-005 | 3.64 bit |
| Jitter day [s] | 2.17e-005 | 0.00e+000 | 5.08e-003 | 1.60e-005 | 3.86e-005 | 1.30e-005 | 3.56 bit |
| Jitter night [s] | 1.59e-005 | 4.00e-006 | 2.91e-003 | 1.40e-005 | 3.29e-005 | 8.00e-006 | 2.85 bit |
| Packet loss [pps] | 910 | 0 | 10000 | 980 | 902 | 1060 | 4.67 bit |
| Packet loss day [pps] | 1004 | 0 | 10000 | 1020 | 992 | 1060 | 4.54 bit |
| Packet loss night [pps] | 745 | 0 | 10000 | 920s | 687 | 1020 | 4.15 bit |
| Delay [s] | 1.78e-002 | 0.00e+000 | 2.88e-001 | 1.57e-002 | 1.28e-002 | 8.03e-003 | 4.60 bit |
| Delay day [s] | 1.75e-002 | 0.00e+000 | 2.88e-001 | 1.54e-002 | 1.26e-002 | 7.57e-003 | 4.37 bit |
| Delay night [s] | 1.84e-002 | 0.00e+000 | 2.16e-001 | 1.63e-002 | 1.32e-002 | 8.72e-003 | 4.23 bit |

reported. Samples collected in the *day* hours are more spread around their median value (36.78 Mbps), as they have a higher standard deviation value (3.61 Mbps). In contrast, the median of the night samples is 37.19 Mbps with a stdev of 2.73 Mbps only. The entropy of the *day* samples (4.95 bit) is higher than that of the *night* samples (4.55 bit). Similar considerations apply for the jitter, packet loss and delay. Samples collected during *night* have a smaller mean, stdev and entropy. All these differences are likely due to the lower degree of interference. However, the differences are far below 1%. Thus , we conclude that the Magnets links are not influenced by day and night.

Finally, we study the impact of special social events, that happened in Berlin during the 2006 FIFA World Cup. During the games, up to a million people gathered in the streets near the backbone location, and a large part of the 3.3 million inhabitants of Berlin were watching the game on TV. Do these non-technical variables change the interference patterns or have other effects that may impact the backbone performance? To assess the impact, we performed a set of 14h long measurements during 5 days on link 3. 18 measurements lasting 2 minutes each were performed with the same parameters described above. On July 9, the championship final was played in Berlin's Olympiastadion. On July 8, the game was played in Munich, but since the German team played, similar conditions can be expected. As baselines, we measured the parameters on July 6, 7 and 10. Figures 3.12(a) and 3.12(b) show the jitter and packet loss as a function of the day time for the 5 different days. The results show that the links are more stable on July 8, 9 from 21:00 to 23:00, i.e. during the matches. But also on July 10, the link seemed stable. Moreover, the differences are not significantly large. We conclude that the environmental conditions have a negligible effect on the backbone performance.




(a) Jitter (timeplot).

(b) Packet loss (timeplot).

Figure 3.12: Impact of 2006 FIFA World Cup on UDP traffic.

### Hetnet

**Analysis** Experiments have been carried out by using three traffic conditions namely *Low*, *Medium*, and *High Traffic*. For each of them a number of PS have been used. Due to the nominal bandwidth of some of the used wireless connections (i.e. GPRS and UMTS), we consider here, only *Low Traffic* condition using IDT equal to $1/100s$ and PS equal to $256Bytes$ (thanks to this choice we have a maximum theoretical bit rate equal to $204.8Kbps$). To point out the end-to-end communication differences, we show the behavior of throughput, jitter, and round trip time measured with UDP connections. We do not present the packet loss because, in the scenarios including WLAN and ADSL, it was always equal to 0. The results presented in the following have been averaged on several tests in order to minimize the effect of random error on measures. In the following, the mean values across 20 test repetitions are reported. Each sample is calculated using non-overlapping windows of 10 ms length.

In the following, we present the results of the statistical analysis performed over six e2e paths. The characteristics of the analyzed scenarios are presented in Tab. 3.4.

Table 3.4: Characteristics of the considered paths.

| ANs | Protocol | OSs | UDs |
|---|---|---|---|
| GPRS-to-Ethernet | UDP | Windows XP-to-Linux | Laptop-to-Workstation |
| UMTS-to-Ethernet | UDP | Windows XP-to-Linux | Laptop-to-Workstation |
| ADSL-to-Ethernet | UDP | Linux-to-Linux | PC-to-Workstation |
| Ethernet-to-GPRS | UDP | Linux-to-Windows XP | Workstation-to-Laptop |
| Ethernet-to-ADSL | UDP | Linux-to-Linux | Workstation-to-Desktop PC |
| Ethernet-to-WLAN | UDP | Linux-to-Windows XP | Workstation-to-Laptop |

Table 3.5: *Concise* statistics of UDP throughput [Kbps].

| e2e Path | Min | Max | Avg | StDev | IQR | Med |
|---|---|---|---|---|---|---|
| GPRS-to-Ethernet | 0 | 40.96 | 18.44 | 6.913 | 0 | 20.48 |
| UMTS-to-Ethernet | 0 | 327.6 | 52.31 | 47.92 | 20.48 | 61.44 |
| ADSL-to-Ethernet | 122.8 | 245.8 | 204.6 | 6.924 | 0 | 204.8 |
| Ethernet-to-GPRS | 0 | 61.44 | 39.56 | 16.60 | 0 | 40.96 |
| Ethernet-to-ADSL | 20.48 | 348.2 | 204.5 | 15.01 | 0 | 204.8 |
| Ethernet-to-WLAN | 184.3 | 225.3 | 204.7 | 3.934 | 0 | 204.8 |

**Results of a concise statistical analysis** Tables 3.5, 3.6, and 3.7 present the results of the concise statistical analysis for throughput, jitter, and delay respectively. Each table is related to one of these parameters and, for each considered path, it contains the *minimum*, the *maximum*, the *average*, and the *median* values of throughput, jitter and round trip time. Also, it contains the *standard deviation* (StDev) and the *inter quantile range* (IQR) of the same parameters. As for the throughput, Table 3.5 shows that, as expected, in the configurations including GPRS and UMTS connections, the minimum, average, and median values of throughput are lower than the corresponding values of the other configurations. Also, the standard deviation looks very similar for very different configurations. Despite this, such result can be related to very different causes and it can be misleading if not observed together with the average values (see GPRS/ADSL-to-Ethernet). It is also interesting to note that, in the UMTS-to-Ethernet configuration, we achieved a standard deviation very close to the mean value. This implies that the average is not much representative of the sample values. That is, the samples achieved values very different from each other. In this case, we also observe a mean value quite different from the median. As for the jitter, Table 3.6 shows that the GPRS/UMTS based configurations achieved the worst performance (higher jitter values) also for this parameter. Indeed, they present higher maximum, average, median, and standard deviation values. The RTT values presented in Table 3.7 confirm such trend. Indeed, all the values are higher for the samples collected by using GPRS and UMTS connections. It is worth noting that, in the case of Ethernet-to-GPRS and GPRS-to-Ethernet, the average and median values are quite different. This is not true in the case of other paths. This behavior is amplified in the case of throughput, and, it means that the role (sender or receiver) of different access networks impacts the performance.

Table 3.6: *Concise* statistics of UDP jitter [s].

| e2e Path | Min | Max | Avg | StDev | IQR | Med |
|---|---|---|---|---|---|---|
| GPRS-to-Ethernet | 0.048 | 5.048 | 0.179 | 0.531 | 0.003 | 0.090 |
| UMTS-to-Ethernet | 0 | 1.768 | 0.054 | 0.171 | 0.02 | 0.03 |
| ADSL-to-Ethernet | 0 | 0.089 | $7 \cdot 10^{-4}$ | 0.002 | $3 \cdot 10^{-4}$ | $7 \cdot 10^{-4}$ |
| Ethernet-to-GPRS | 0 | 0.518 | 0.055 | 0.076 | 0.066 | 0.047 |
| Ethernet-to-ADSL | 0 | 0.034 | $6 \cdot 10^{-3}$ | 0.001 | $4 \cdot 10^{-4}$ | $3 \cdot 10^{-4}$ |
| Ethernet-to-WLAN | 0 | 0.023 | $7 \cdot 10^{-4}$ | 0.001 | $7 \cdot 10^{-4}$ | $5 \cdot 10^{-4}$ |

Table 3.7: *Concise* statistics of UDP RTT [s].

| Path | Min | Max | Avg | StDev | IQR | Med |
|---|---|---|---|---|---|---|
| GPRS-to-Ethernet | 6.309 | 17.11 | 10.95 | 3.92 | 6.311 | 12.69 |
| UMTS-to-Ethernet | 1.535 | 4.182 | 2.551 | 0.573 | 0.543 | 2.494 |
| ADSL-to-Ethernet | 0.042 | 0.638 | 0.277 | 0.130 | 0.070 | 0.260 |
| Ethernet-to-GPRS | 0.801 | 14.31 | 10.15 | 3.017 | 3.040 | 11.26 |
| Ethernet-to-ADSL | 0.044 | 0.200 | 0.095 | 0.014 | 0.019 | 0.100 |
| Ethernet-to-WLAN | $3 \cdot 10^{-4}$ | 0.135 | 0.084 | 0.02 | 0.002 | 0.090 |

**Results of a detailed statistical analysis** In this section, we show our results in terms of *Probability Density Functions* (PDFs) as well as some results regarding the *Auto Correlation Function* (ACF), the *Entropy* measure, the *tail* analysis, and the *Bivariate Probability Density Function*. It is important to underline that the throughput samples were collected evaluating the average value on fixed size time intervals ($100ms$), while for RTT and jitter, each packet represents one sample. Also, to plot the PDFs of all the considered parameters, we used the bin width suggested by the Scott Rule [117]. In Figure 3.13 the PDFs of the throughput samples are depicted. Figure 3.13 shows that (i) in the GPRS-to-Ethernet case the main part (87%) of the samples achieved the median value ($20.48Kbps$), while more than 10% was $0Kbps$; (ii) in the UMTS-to-Ethernet scenario the samples are spread over the interval $[0, 350]Kbps$; (iii) in the ADSL-to-Ethernet case the median value ($204.8Kbps$) is obtained by more than 90% of the samples; (iv) in the Ethernet-to-GPRS case the samples are multi-modally distributed over 4 values (0, 20.48, 40.96, and $61.44Kbps$); (v) in the Ethernet-to-ADSL scenario even if more than 90% of the samples attained the median value ($204.8Kbps$), the remaining ones range from 20.48 to $348.2Kbps$; (vi) in the Ethernet-to-WLAN case the samples are very highly concentrated around their median value ($204.8Kbps$).

In the left plot of Fig. 3.14, the PDFs of the jitter samples are depicted. As shown, the distributions look similar in the shape, indeed, they present the majority of the samples close to 0 even if a not negligible upper tail is noticed. For better seeing the main part of

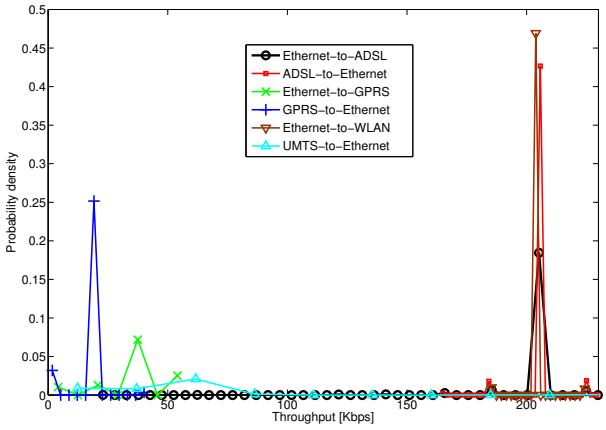



Figure 3.13: PDF of throughput of UDP.

the distribution, in the plot we included also a zoom on sample lower values. However, the sample values of the configurations including GPRS/UMTS differ of about 1 order of magnitude from the other configurations. Indeed, in the GPRS-to-Ethernet, Ethernet-to-GPRS, and UMTS-to-Ethernet cases they are mainly distributed (95% of samples) over the interval $[0, 0.15]s$. Instead, in the other cases, the 95% of the samples present values less than $0.015s$.

In Figure 3.14 the PDFs of the RTT samples are sketched. To ease the analysis, such figure includes also a zoom on the first part of the distributions. In contrast with the jitter, here the distributions are very different from each other. Indeed, the GPRS-to-Ethernet samples are multi-modally distributed around 4 values (6.5, 10, 12.5, and $17s$). In the UMTS-to-Ethernet case the distribution is bimodal with the modes not strictly separated. In the ADSL-to-Ethernet configuration the samples are spread all over the $[0.05, 0.7]s$ interval with a concentration (the 50% of the samples) around their median value ($0.26s$). Ethernet-to-GPRS samples are close to their median value ($11.26s$) and a heavy lower tail is present. The Ethernet-to-ADSL configuration presents samples that are mainly distributed over the interval $[0.04, 0.2]s$, and, in the mainly populated interval ($[0.06, 0.12]s$), spikes are present at multiple of $0.01s$. Finally, the Ethernet-to-WLAN samples are bimodally distributed around $0.02s$ and $0.09s$.

To understand the samples statistical dependence, in Figure 3.15 the ACF of RTT of UDP samples as a function of sample distance (called *lag*) is sketched. As we can see the $ACF(1)$ values is higher than 0.9 for all the considered configurations. Also,

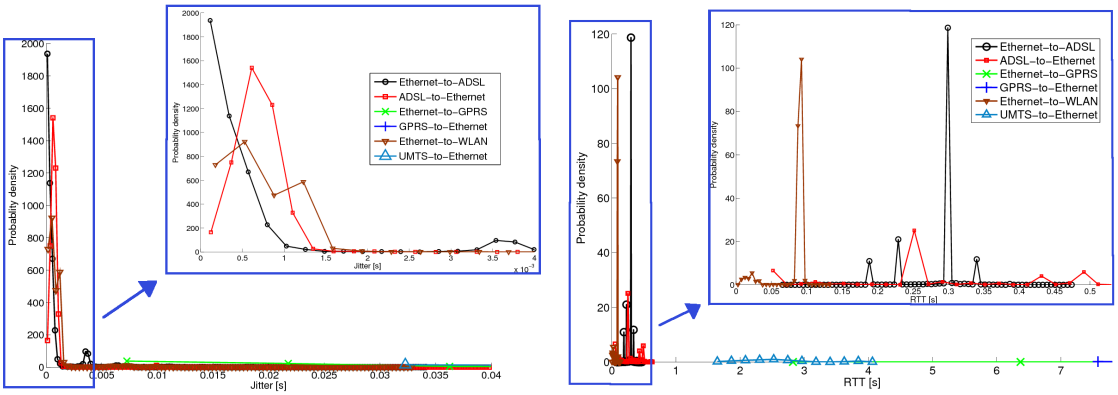



Figure 3.14: PDF of jitter and RTT of UDP (normal and zoomed view).

the configurations that include GPRS and UMTS connections present more uncorrelation among the samples. Indeed, for such configurations, the autocorrelation plot decays more rapidly than in the other cases. Such behavior proves that the GPRS and UMTS connections introduce uncorrelated randomness in packet arrival process. In the case of GPRS and UMTS at sender side, the ACF shows an oscillating behavior. This is due to periodicities in the RTT sequences. Our preliminary analysis shows that such behavior is related to the packet loss trend [118].

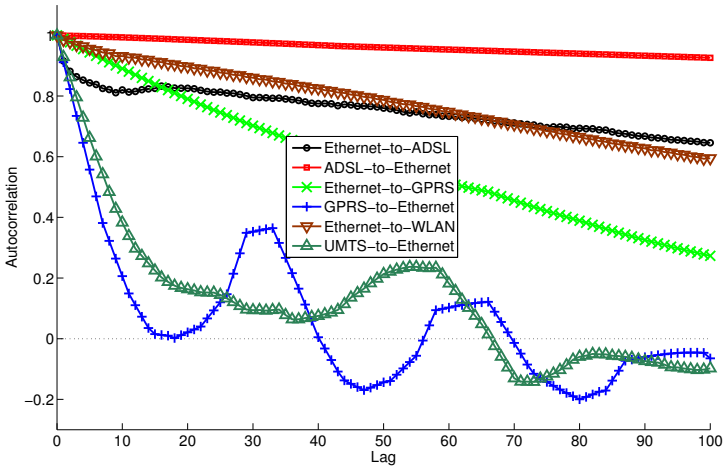



Figure 3.15: ACF of RTT of UDP samples.

In order to better understand the variability of the collected samples we have also evaluated the entropy of each trace. For the sake of comparing entropy values of different

configurations, when estimating such parameter, we used the same bin size for all the configurations instead of that suggested by the Scott rule. In the RTT case, we used a fixed bin width equal to $0.01s$, while for the jitter, the bin width is equal to $0.001s$. Table 3.8 presents entropy values calculated for the jitter and RTT samples. Such table shows that the entropy values obtained on the GPRS-to-Ethernet, Ethernet-to-GPRS, and UMTS-to-Ethernet paths are always higher than the ones achieved in all the other configurations. Furthermore, the reported values prove that the randomness introduced by GPRS and UMTS connections influences both the delay and its variations (jitter). Finally, it is interesting to note that when the GPRS is used at sender side, both RTT and jitter entropy values are much higher with respect to the other direction of the communication.

Table 3.8: Entropy of jitter and RTT [bit].

| E2E path | GPRS-to- Eth | UMTS-to-Eth | ADSL-to-Eth | Eth-to-GPRS | Eth-to-ADSL | Eth-to-WLAN |
|---|---|---|---|---|---|---|
| Jitter | 3.978 | 3.258 | 0.465 | 6.079 | 0.628 | 0.910 |
| RTT | 4.399 | 6.406 | 2.976 | 8.504 | 2.562 | 1.571 |

In the left plot of Fig. 3.16 the CCDF of the jitter samples is depicted, which shows that the jitter presents a heavy tail behavior for all the analyzed configuration. In the right plot of Fig. 3.16 the CCDF of RTT samples is sketched using logarithmic scales. In contrast to the previous parameter, there is no evidence of a heavy tail behavior. Indeed, for all the considered configurations the sample distributions decay to zero with an over-exponential rate.

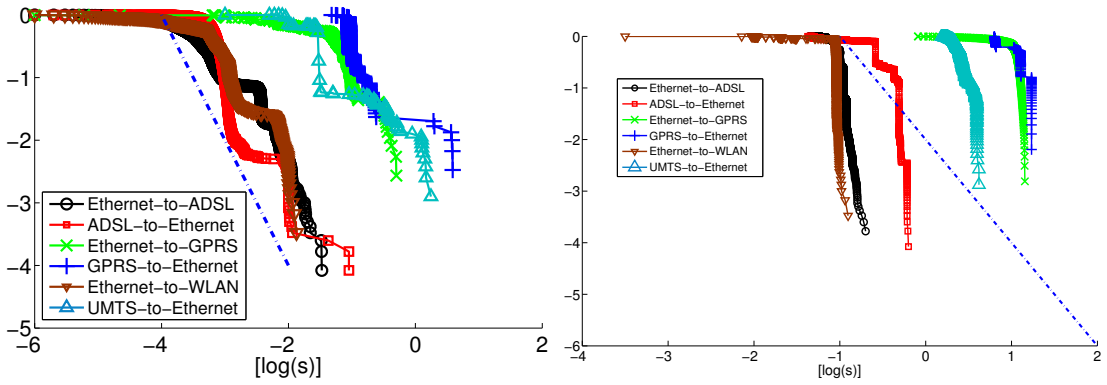



Figure 3.16: Log-Log CCDF of UDP jitter (left) and RTT (right).

### 3.2.5 Summary and conclusion

This analysis evidenced the importance of performing a statistical characterization that: 1) is based on a combined approach composed of both "*concise*" and "*detailed*" statistics; considers all the parameters that can have an impact on the performance such as the environmental ones; analyzes the different paths available between the considered hosts. This allows to have a clear picture of the status of the network, which is useful if we want to inform the path and time diversity schemes. We proposed an active measurement approach using a novel platform called D-ITG, illustrating how this tool can be used to acquire a deep knowledge of the network under analysis. To improve the knowledge in the field of the performance evaluation of heterogeneous wireline/wireless networks, data and tools used in this work are freely available.

In the next section, we illustrate another methodology based on passive measurements, and we show how this methodology complements the one shown before allowing to obtain performance information without injecting additional traffic in the network.

## 3.3 Passive measurements

In this section, we present our activities aimed at inferring the status of an operational network using passive measurement techniques, which means that we used packet traces of traffic from real users. From such traces we extracted parameters useful to understand the performance experimented by the users, and then, the status of the network.

### 3.3.1 Methodology

In this research, we analyze performance parameters extracted from packet traces passively collected on an operational network. We initially use the methodology presented in [23], which is called RCAT. As we evidenced some weaknesses of the methodology, we had to extend it. In the following we therefore report information regarding the additional metrics we considered in order to complement RCAT results.

The **retransmission** rate is calculated by using the Sequence Number field (SeqNum) from the TCP header and the Identification field (ID) from the IP header: a packet is considered to be a retransmission if its ending sequence number (i.e. SeqNum plus the payload size) is smaller than the current maximum, while its ID is larger than the

current maximum. The retransmission rate represents the ratio between the amount of retransmitted bytes and the total amount of transmitted bytes.

The **RTT** samples, instead, are calculated by taking the difference between the timestamp of a data packet and the timestamp of the corresponding ACK. In this way we can easily have several samples for each connection. But, for the samples to be meaningful, the packets should be captured with accurate hardware and, even more importantly, at a location that is as close as possible to the packet sender. This is the case for our traces, as better explained in the following.

To evaluate the **number of parallel connections**, we first computed the starting and ending time of each connection, and then counted the number of connections that overlap in each time interval.

Finally, the **throughput** is simply calculated by dividing the total amount of bytes transmitted in a given interval by the duration of the interval. Such parameter will be calculated both for all the packets of a user and for only those related to its BTPs. This is to highlight the effect of the application.

All the parameters are calculated using non-overlapping time windows of 30 seconds, and are related to all the traffic generated by and directed to a single IP address (i.e. a mobile station). Aggregating traffic per IP allows to discover details that are normally hidden when looking at the single connections, as shown in the following.

In many cases we also needed information regarding the application in use. For this aim, we performed DNS inverse lookup, reverse WHOIS queries, and manual verifications.

### 3.3.2 Network and traces

In this work we analyze four packet traces collected on an operational cellular network, whose schema is reported in Fig. 3.17. Such network comprises different kinds of radio access technologies: GSM, GPRS, EDGE, UMTS, and UMTS+HSDPA. This is common in currently deployed cellular networks, because they are still in the transition between 2nd and 3rd generation[5]. Moreover, it allows to have a complete picture of the performance of different cellular network technologies.

This network employs a performance enhancing proxy (PEP), which operates as a

[5]Even if the operators have almost completely upgraded their networks, they still have to wait for all customers to buy new equipments before switching to the new technology. Moreover, for some rural areas, they prefer to use the GSM/GPRS network due to its higher coverage.

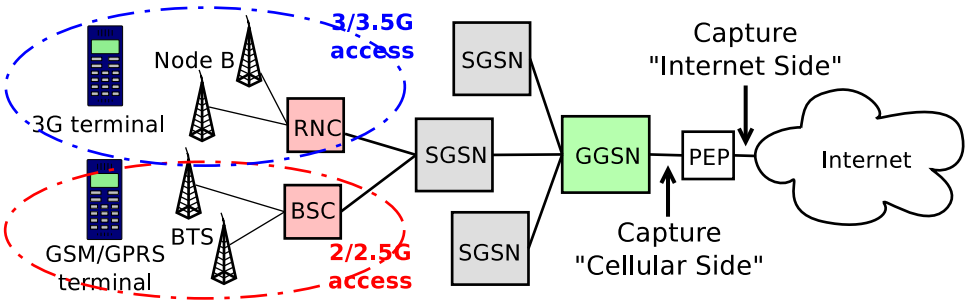



Figure 3.17: Schema of our UMTS network.

transparent proxy: it terminates all the TCP connections of the mobile stations and opens new connections towards the Internet, while preserving the source and destination IP addresses and ports. We analyzed traces collected at both sides of the proxy, as shown in Fig. 3.17 (we will refer to them as to Cellular and Internet sides in the following). From a network measurement point of view, we have some advantages with respect to other works in literature (e.g. [119]): i) we can have information such as losses and retransmissions related only to the cellular network, i.e. not dependent on the Internet side; ii) by analyzing the information related to both the Cellular and the Internet sides, we can precisely pinpoint the causes of the performance limitations we observe; iii) the PEP represents the traffic sender for most connections (i.e. the downlink ones) of the Cellular side, which allows to have an accurate estimation of the RTT on the cellular network. The PEP is acting only on the TCP connections directed towards port 80, therefore our analysis has been be focused on this kind of traffic.

Details about the analyzed traces are reported in Table 3.9. As we can see, we have two traces for each side of the PEP, each of which is related to a different group of mobile stations. Table 3.9 also shows that on the Internet side we have more packets and connections than on the other side. The PEP was originally deployed to improve the download performance of 2G users in case of HTTP traffic. To this purpose, the PEP opens multiple parallel TCP connections towards a mobile user to transmit the data coming from an HTTP server in the Internet. The PEP tries to open many parallel connections when processing an HTML page requested in order to overcome the restrictions imposed by HTTP 1.1 to open no more than two parallel connections [120]. It is worth noting that the traces are very recent as they have been collected at the end of 2008. Therefore, they allow to observe the performance of cellular networks currently deployed.

Table 3.9: Characteristics of the analyzed traces.

| Trace | Date and time | Duration | Packets | Bytes | Connections |
|---|---|---|---|---|---|
| Cellular1 | 2008-12-11 20:31 | 00:59:56 | 5Mega | 4Giga | 187,034 |
| Internet1 | 2008-12-11 20:31 | 01:00:07 | 6Mega | 4Giga | 226,814 |
| Cellular2 | 2008-12-11 20:31 | 01:00:22 | 5Mega | 4Giga | 178,292 |
| Internet2 | 2008-12-11 20:31 | 01:00:21 | 6Mega | 4Giga | 223,175 |

### 3.3.3 Results

In the following we present the results we obtained. Firstly, we provide an overview of the performance throughout the network; then, we concentrate on some specific users.

#### Overview

Table 3.10 reports important global statistics related to the trace Cellular1. Similar considerations apply for the other cellular trace. Firstly, we observe that the ratio between the number of packets sent and received by the mobile stations is about 10%, while the same ratio in Bytes is about 5%. Moreover, we notice that, in the downlink direction (i.e. from the Internet to the mobile stations) the largest connection is responsible for about 249 MBytes of traffic, the 7.5% of the total. While, in the uplink direction, the largest connection is responsible for about 7.42 MBytes, the 4.4% of the total.

Table 3.10: Cellular1: overall statistics.

| | |
|---|---|
| Bytes in payload from servers to clients | $3.30 * 10^9$ |
| Bytes in payload from clients to servers | $1.68 * 10^8$ |
| Data packets from servers to clients | $2.62 * 10^6$ |
| Data packets from clients to servers | $2.55 * 10^5$ |
| Max bytes/connection from servers to clients | $2.49 * 10^8$ |
| Max bytes/connection from clients to servers | $7.42 * 10^6$ |
| Max pkts/connection from servers to clients | $1.83 * 10^5$ |
| Max pkts/connection from clients to servers | $5.17 * 10^3$ |
| Average RTT | 683 ms |
| Retransmissions from servers to clients due to timeout expiration | $5.76 * 10^4$ |
| Retransmissions from servers to clients due to fast retransmit | $1.45 * 10^4$ |

Fig. 3.18 shows the cumulative distribution function (CDF) of the number of bytes per connection, in both the uplink and downlink directions. We can see that, in both cases, a small number of users is responsible for the most part of the traffic, i.e. the distributions are skewed. Moreover, we observe that most of the connections in the uplink direction generate between 0.5 and 10 KBytes. While much more variation is observed in the other direction.

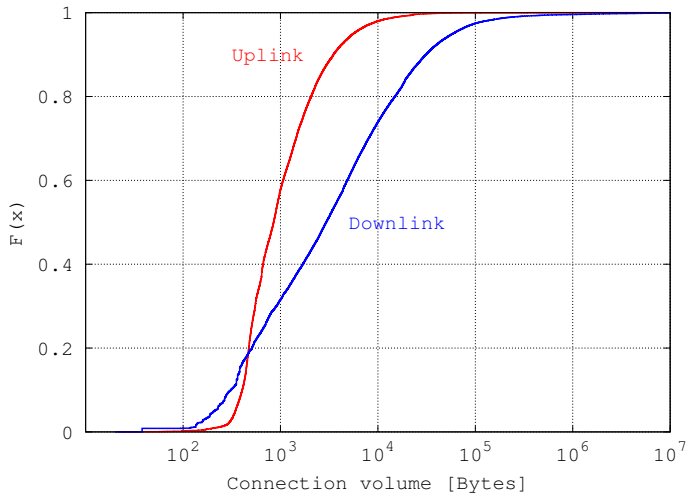



Figure 3.18: CDF of connection volume.

In Table 3.10 we also report the average RTT and the total number of retransmissions. The value of the former parameter is 683 ms, which is very high compared to the Internet [121]. Indeed, in Fig. 3.19 we report the CDF of the average RTT per connection for both the traces Cellular1 and Internet1. As we can see, on the cellular side, the values range from 50ms to 5s, with 50% of the connections having an average RTT value larger than 400ms. While on the Internet side, 80% of the connections have an average RTT lower than 100ms. This indicates that i) there is a high level of heterogeneity in the user performance, and ii) on average RTT is quite high. Comparing the values of RTT with results reported in 2005 by Vacirca et al. [119] we discover that in average the RTT has remained almost unchanged. Moreover, in our trace about 15% of the RTT samples fall into the range 100 to 200 ms. While, according to the results from [119], such percentage was about 10% in 2005 (see Fig. 2 of such paper).

As for the retransmissions, summing the values last two rows of Tab. 3.10 (i.e. those due to timeout expiration and fast retransmit mechanism) we obtain that about the 2.7% of packets from servers to clients are retransmitted. Again, this value is much higher than that on a typical Internet link [122, 123]. Moreover, we observe that there are much more retransmissions due to timeout expiration than due to fast retransmit, which can be due to the high variability of the RTT (shown in the following).

From this analysis we learned that current cellular networks are characterized by a high degree of heterogeneity in terms of technology and therefore performance. Taking into

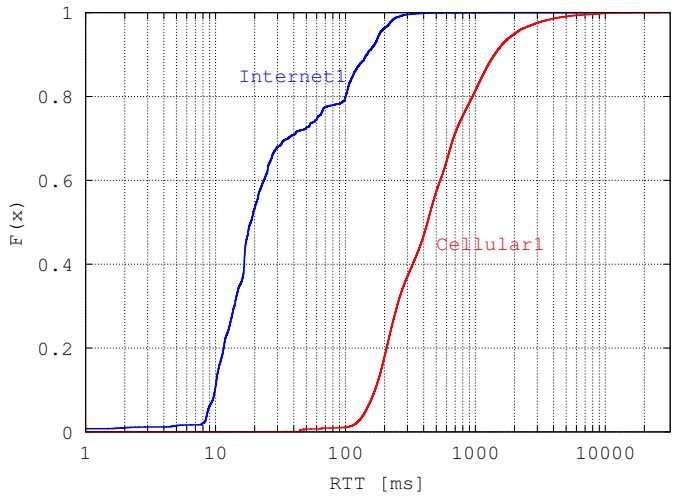



Figure 3.19: CDF of the average RTT.

Table 3.11: Statistics of the bulk transfer periods for the users with most BTPs.

| Name | # of BTPs | Duration [s] Min | Duration [s] Max | Tput [Kbps] Min | Tput [Kbps] Max | Capacity [Kbps] Min | Capacity [Kbps] Max | *retr score* Min | *retr score* Max | *rwnd score* Min | *rwnd score* Max | MBytes in BTP | BTP Bytes/ all Bytes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | 141 | 1 | 73 | 271 | 2921 | 652 | 3199 | 0.000 | 0.173 | 0.00 | 0.86 | 111.523 | 0.58 |
| Green | 86 | 6 | 833 | 11 | 297 | 8 | 405 | 0.000 | 0.205 | 0.00 | 0.86 | 58.997 | 0.75 |
| Blue | 85 | 1 | 111 | 662 | 2308 | 8806 | 76090 | 0.000 | 0.048 | 0.00 | 1.00 | 120.778 | 0.95 |
| Yellow | 69 | 1 | 130 | 707 | 2386 | 874 | 1690 | 0.000 | 0.025 | 0.00 | 0.83 | 99.653 | 0.87 |
| Purple | 43 | 3 | 514 | 4 | 576 | 27 | 301 | 0.016 | 0.169 | 0.00 | 0.03 | 26.201 | 0.61 |
| Cian | 40 | 2 | 93 | 101 | 816 | 84 | 915 | 0.000 | 0.085 | 0.00 | 0.80 | 25.068 | 0.50 |
| Black | 38 | 1 | 965 | 130 | 1621 | 598 | 8254 | 0.000 | 0.068 | 0.00 | 0.94 | 205.104 | 0.95 |
| Grey | 36 | 2 | 12 | 374 | 882 | 611 | 817 | 0.000 | 0.000 | 0.56 | 0.99 | 10.584 | 0.61 |

account also the variability of physical channel conditions, we obtain that very different performance can be achieved. As an important consequence, in these network scenarios, the average values of the performance parameters are not very representative, which is to say that aggregating the measures on the entire network or performing measurements from a single vantage point may not provide representative performance samples.

### Users with the highest number of BTP

**Analyzing RCAT results** Tab. 3.11 contains some parameters related to the BTPs of 8 users that are responsible for the largest amount of BTPs (about 38% of the total number of BTPs in the trace).

To understand the weight of the BTPs on the traffic of this users, Tab. 3.11 reports the ratio between the amount of bytes in BTPs with respect to the total bytes generated. As shown, for all the users at least 50% of the bytes are related to the BTPs (ratio $\geq 0.50$), and for half of the users the ratio is equal to or higher than 0.75. This means that for

most of the Bytes of these users the throughput is not limited by the applications. Note that all the BTPs in the trace account for about 67% of the total amount of Bytes, but for about the 1% of the total time of the connections.

On the other hand, we observe a high degree of variation in both the minimum and maximum values of the throughput. Looking at the durations of the BTPs we can see that they can be very small, which indicates that such high throughput can be related to very short BTPs. Recall that the throughput is calculated by dividing the total amount of bytes seen at the measurement point (i.e. the sender for this trace) by the difference between the timestamp of the last packet and the timestamp of the first packet (i.e. the total duration). So, if the measurement point is close to or at the sender side (as in our case), it is possible that some packets in the trace are actually never received by the destination. While this can have a small or negligible impact on long lasting BTPs (TCP would back-off and reduce the sending rate when no acknowledgements are received), it can lead to very inaccurate values in the case of short transfers. Possible solutions to this issue include calculating the throughput by only looking at the packets that have an acknowledgement, filtering the BTPs shorter than a certain threshold, or, as we will show in the following, aggregating the values on some (large enough) time window.

Another important observation is related to the receiver window score: with the exception of user *Purple*, the maximum value is always very high. This is due to a phenomenon we often observed in the cellular network, which, if not correctly understood, could wrongly lead to the conclusion that these hosts have too small receiving buffers. The real explanation is instead different, as reported in the following. In circumstances such as temporary link quality worsening, the packets are buffered by the cellular network infrastructure because different operations are performed at the data link layer (error control, retransmissions, ...). As a result the RTT increases, and if there is enough space in the buffers, no losses are observed. This leads to periods of up to a few minutes in which the sender is not sending packets at the maximum possible speed because it is limited by the receiver window. As a consequence, we observe a receiver window score increasing even if there is space in the receiver buffer.

It is also interesting to note that the minimum retransmission score is often equal to 0. Which means that some BTPs do no experiment any packet loss. On the other hand, some BTPs achieve values up to 0.20, which means that 20% of the Bytes are retransmitted. This aspect surely deserves a deeper analysis, which will be performed in

the next section.

**A closer look** The RCAT provides us some concise information regarding the performance of the users. From the analysis of previous section, we discovered that there is a certain variation in the obtained results. However, we could not make any general conclusion. In this section, we therefore go a bit deeper, analyzing in more details the results provided by RCAT. Fig. 3.20 shows the throughput achieved by the single BTPs of each of the 8 users. Note that, the RCAT provides an average value of the throughput for the BTP. In Fig. 3.20 we therefore report such average value with a segment located at the BTP time (i.e. the endpoints coincide with BTP starting time and ending time). This is to understand the similarities and difference between the BTPs of the same users, also in relation to the time in which the BTPs take place. This picture evidences that, for some hosts - e.g. the *Blue*, *Yellow*, and the *Red* - the BTPs of the same user achieve different throughput values (i.e. high variation). Other mobile stations instead - e.g. the *Grey*, and *Green* - see quite uniform throughput.

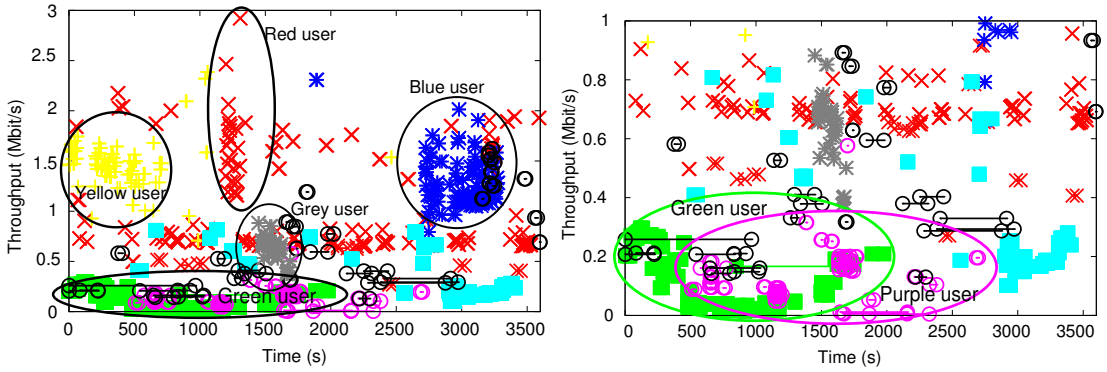



Figure 3.20: Throughput vs Time of the BTPs (entire and zoom).

In Fig. 3.21 we report the average retransmission score achieved by the BTPs of the same 8 users. In this case we can easily see how some of these users usually experiment a high number of retransmissions, i.e. the *Green* and the *Purple* (highlighted with a circle). If we look at Table 3.11 and Fig. 3.20, we observe that such users achieve the lowest maximum throughput. Which means that a high retransmission score achieved by the BTP is an indication of a possible problem affecting the mobile station. Moreover, from this picture we can observe that some users have many parallel transfer at the same time. This behavior was partly hidden in the Fig. 3.20 because such parallel connections

generally obtain a low throughput. Therefore, the segments related to them are overlapped in the bottom part of Fig. 3.20.

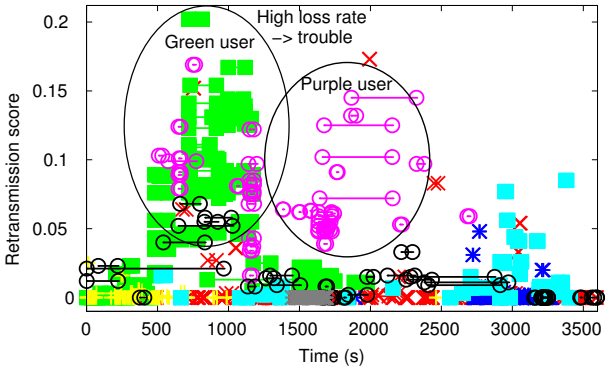
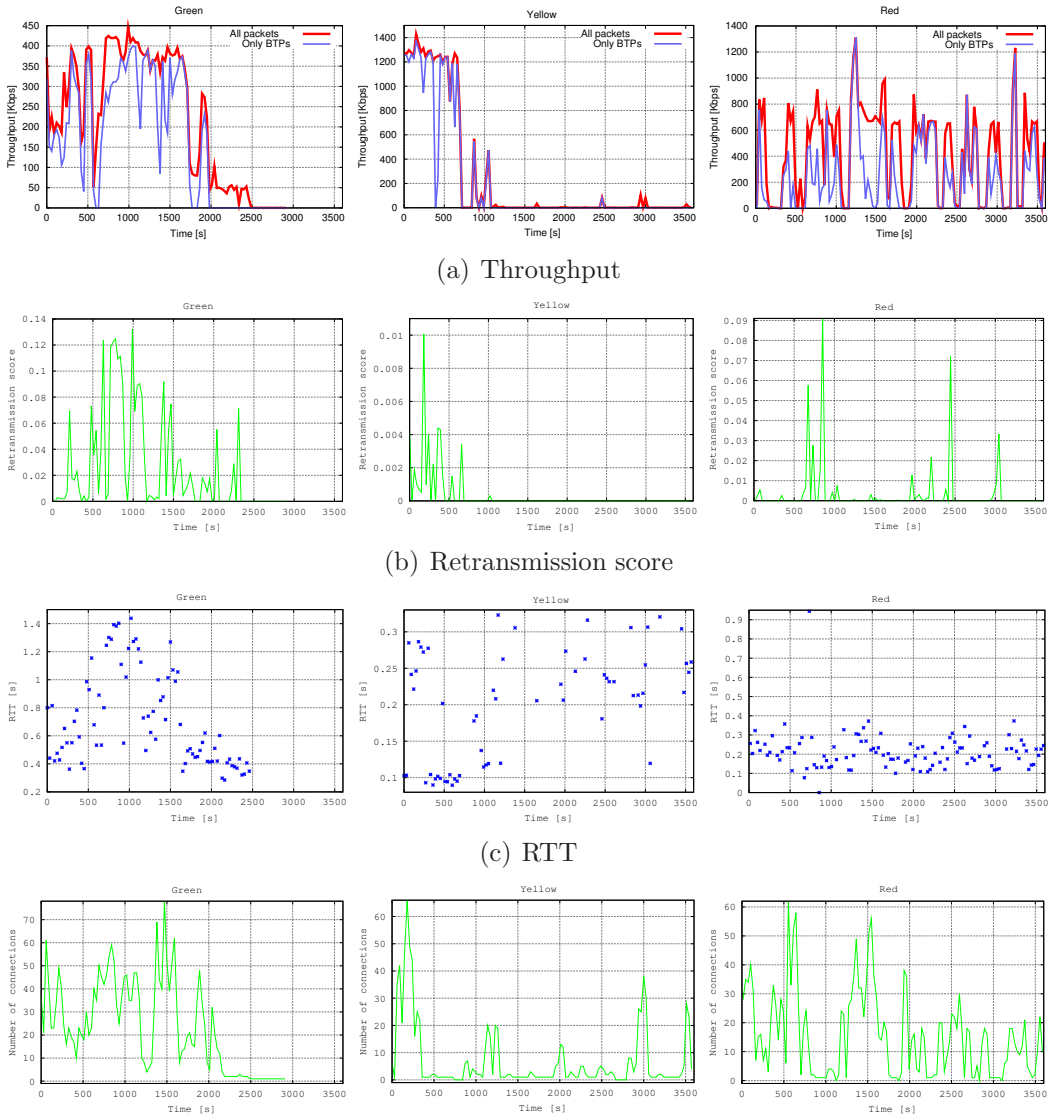



Figure 3.21: Retransmission scores vs Time of the BTPs.

From this analysis we learned something more about the mobile stations and the performance of the cellular network. For instance, we observed that the retransmissions are a good indicator of a performance problem, and that the users tend to open several parallel connections. This second aspect is particularly important. In 2005, Vacirca et al. [119] reported that they excluded p2p traffic from their analysis because of the use of several parallel connections, which are likely to induce self-congestions. We observed that in our trace, also HTTP traffic is characterized by the use of many parallel connections. We cannot assess if this is related to new kinds of applications and new user behaviors, or it existed also in 2005. However, we believe it is important to take these aspects into consideration when performing an analysis of the performance of cellular networks.

However, also in this case we could not make any general conclusion about such performance. This is mainly due to the fact that the current users and their applications tend to open many parallel connections, which can compete with each other for the available resources. As a consequence, looking at a single TCP connection is not sufficient to understand the performance of the network, and in some cases, also of the applications. Such consideration led us to perform another analysis on the collected traces, considering two additional aspects: the variation of the performance parameters over time and the total traffic sent and received by the mobile stations within all their connections. For these analyses we chose the four important performance parameters explained in Section 3.3.1.




(a) Throughput

(b) Retransmission score

(c) RTT

(d) Number of connections

Figure 3.22: Performance of *Green*, *Yellow*, and *Red* users.

**Time behaviour analysis** Fig. 3.22 shows the time series of the throughput, retransmission score, RTT, and number of parallel connections of three mobile stations from the group previously analyzed. We report these users because they allow to make some remarks on the behavior of the network and on the accuracy of the results presented before. Similar considerations apply for the others.

The user *Green*, whose performance are reported the in left plots of Fig. 3.22, is one of those experimenting poor performance. The reason for the low performance can be found

in the values of retransmission rate and RTT obtained. The former parameter has a bursty behavior and reaches values up to 0.12 (i.e. 12% of bytes are retransmitted). The RTT, instead, increases in the first period, reaches its maximum (about 1.5 s), and then starts decreasing, which is about the same behavior of the throughput. The synchronization between throughput and RTT is related to the buffering mechanisms implemented at both the TCP and the data link layer (i.e. within the cellular network), which try to compensate the poor channel conditions with error correction and retransmissions. Another interesting consideration is that at about second 1250 the number of connections is very low (down to 6). At the same time, we observe a decrease in the retransmission score and RTT, while the throughput remains to the same values as before. This suggests that, when a user achieved the maximum throughput, increasing the number of connections does not provide a benefit, but rather worsen the performance.

The plots related to the user *Yellow* are reported in the middle of Fig. 3.22. This user achieves high throughput, which reaches values up to 1.4 Mbps, and remains at about 1.2 Mbps for almost 10 minutes. We also observe that the RTT remains always under 350 ms, indicating that less buffering is happening. The plot of the retransmission score shows that very few bytes are lost. It is also interesting to note that, while in the previous period there are many concurrent connections, from second 400 to 700 only 1 or 2 connections are active. This has a positive impact on the RTT, which remains at about 100 ms during this period, while the throughput is still achieving a high value. Such behavior confirms what we observed for the user *Green*, i.e. parallel connections can have a counterproductive effect when the user tries to go beyond the limits of the network.

The user *Red* is experimenting "reasonably good" performance, which is in-between those of the *Green* and the *Yellow* users. The throughput plot shows a largely fluctuating trend, with spikes that reach 1.3 Mbps for a few measurement periods (i.e. a few times 30 seconds). However, the RTT stays almost always below 400 ms (only one sample exceeds this value). The retransmission score presents low values, but some spikes are also present. It is also interesting to note how the spike in the number of connections between 500 s and 700 s results in a decrease of the throughput. We can conclude that the performance of this user are impacted by different factors that are the network conditions, the use of multiple connections, and the application. All these factors interact with each other creating a highly variable throughput behaviour.

Table 3.12: Statistics of the bulk transfer periods for the users generating most Bytes.

| Name | BTPs | Duration [s] Min | Duration [s] Max | Tput [Kbps] Min | Tput [Kbps] Max | Capacity [Kbps] Min | Capacity [Kbps] Max | *retr score* Min | *retr score* Max | *rwnd score* Min | *rwnd score* Max | MBytes in BTP | BTP Bytes/ all Bytes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pink | 8 | 1 | 2086 | 327 | 1463 | 129 | 2380 | 0.000 | 0.008 | 0.00 | 0.53 | 252.548 | 0.98 |
| Emerald | 7 | 19 | 3419 | 81 | 152 | 56 | 105 | 0.000 | 0.005 | 0.00 | 0.97 | 146.037 | 0.96 |
| Azure | 6 | 202 | 694 | 240 | 459 | 90 | 615 | 0.008 | 0.017 | 0.00 | 0.01 | 113.345 | 0.98 |
| Amber | 17 | 20 | 445 | 265 | 616 | 130 | 854 | 0.000 | 0.018 | 0.00 | 0.84 | 105.147 | 0.98 |
| Amethyst | 6 | 2 | 454 | 445 | 973 | 256 | 2184 | 0.000 | 0.009 | 0.00 | 0.42 | 58.972 | 0.83 |
| Aqua | 1 | 1721 | 1721 | 270 | 270 | 391 | 391 | 0.001 | 0.001 | 0.00 | 0.00 | 58.195 | 1.00 |
| Dark | 12 | 9 | 539 | 115 | 331 | 63 | 408 | 0.012 | 0.153 | 0.00 | 0.20 | 48.258 | 0.91 |
| Silver | 2 | 17 | 3596 | 100 | 391 | 174 | 211 | 0.045 | 0.055 | 0.00 | 0.02 | 45.862 | 0.97 |

### Users generating the highest amount of traffic

In previous sections we have assessed the methodology and gained the first insights into the performance of the cellular network. In this section we look at the performance achieved by heavy users, i.e. those generating the largest amount of data. This allows to have a better understanding of what cellular-network users can achieve. To select these users we firstly ordered all those in the trace by the amount of traffic generated, and then we picked up the first eight, excluding the ones previously selected. This is to avoid analyzing the same users twice.

Tab. 3.12 shows the statistics related to the BTPs of such users. We observe smaller values of throughput and capacity with respect to the users analyzed in Section 3.3.3. This is actually due to the fact that these BTPs are longer, and are therefore less sensitive to the problem related to the very high throughput values, discussed in Section 3.3.3. Moreover, we observe that the ratio between the amount of Bytes in the BTPs and the total amount of generated Bytes is very high. This means that, for such heavy users, most of the Bytes are related to BTPS. As a general consideration, we can state that the performance of the users that generate the highest amount of Bytes are typically dominated by the network and not by the application.

The fact that the BTPs are longer than those from the previous group of users is also an indication of the fact that the connections of these users are longer. Some BTPs are about 1 hour long (see users *Emerald* and *Silver*), while others are about 1/2 an hour long (see users *Pink* and *Aqua*). As an interesting case, the user *Aqua* has only one and very long connection, which carries about 58 MBytes of data. More information about these users and their BTPs are reported in the following sections. Also, we observe more variation in the values of the receiver window score obtained by these users. This is due to the fact that they generate fewer BTPs and therefore less *rwnd score* samples.

Another interesting consideration is related to the retransmission scores. Comparing

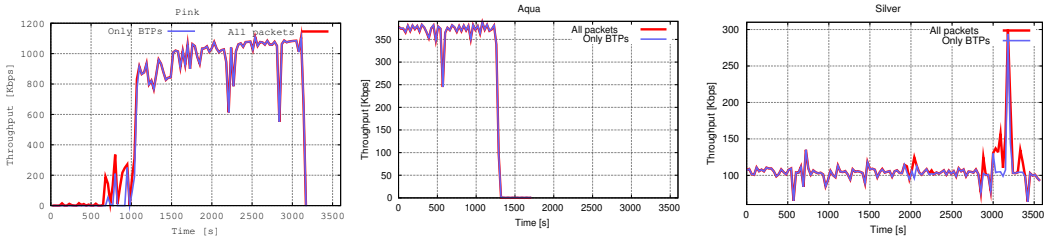
Table 3.11 and 3.12, we observe that the users from the latter group achieve smaller values of such score. This is again related to the duration of the measurement period: the values of the former group can be related to very short BTPs. As another general consideration, we can state that when performing network measurements using packet traces (i.e. passive measurements), particular attention should be paid to short connections. Their dynamics can be indeed shorter then those of the network, which can therefore yield measures difficult to understand or even non consistent with the real conditions of the network.

**Time behavior analysis** In this section we analyze the time behavior of the throughput, retransmission score, RTT, and number of parallel connections of three users from the previous group.

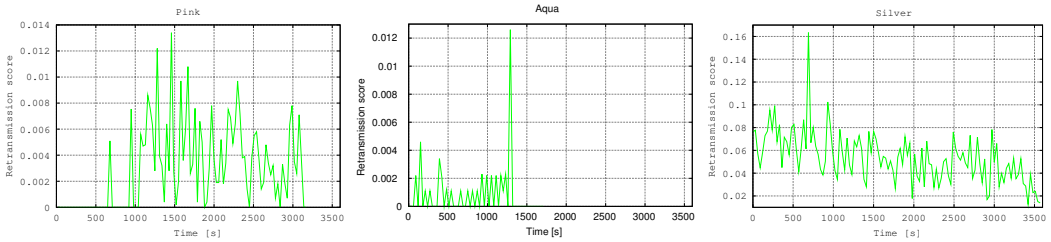
The results of the *Pink* user are reported in the plots on the left of Fig. 3.23. As shown, the throughput reaches high values (up to 1.1 Mbps), and maintains this values for more than 30 minutes. We also observe that both the retransmission score and the RTT present small and stable values during this period. It is also interesting to note how after second 1100, the user has only one connection running. This connection is then able to progressively increase the network usage, until achieving the highest possible throughput.

The middle plots of Fig. 3.23 report the results obtained by the user *Aqua*. This user has one BTP lasting for almost 28 minutes and responsible for about 58MB of data. Such BTP is related to a connection with a Limelight[6] host in Germany. A first observation is related to the throughput. Looking at Table 3.12, we can see that, according to RCAT, this BTP has a throughput of 270 Kbps. However, in Fig. 3.23 we observe that the actual throughput is about 370 Kbps, and it is quite stable for the entire period. The difference between these two values is due to the fact that RCAT considers the BTP lasting for 1721 s (see Table 3.12), even if during the last 300 seconds the throughput is almost equal to 0. This happens because after second 1400 the sending host does not receive more acknowledgements from the mobile stations. It then performs retransmissions for the successive 7 minutes using TCP exponential backoff. Clearly those packets should not be considered if we are interested in observing the performance of the network. We can also observe that the RTT of this user presents a high variation, and reaches values up to 1.2 s. The retransmission score is always very low.

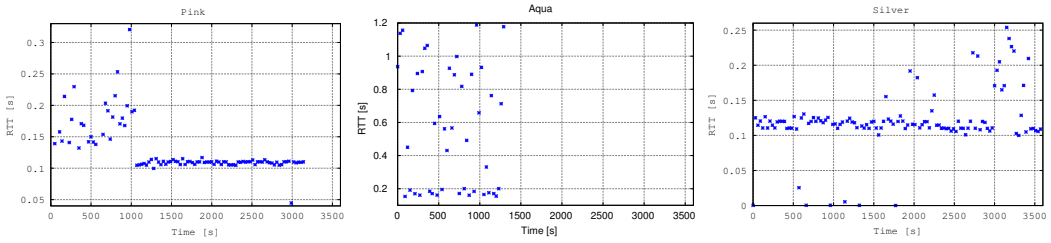
[6]http://www.limelight.com



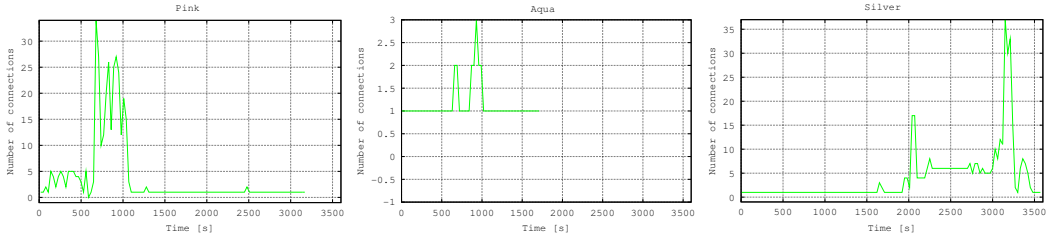

(a) Throughput




(b) Retransmission score




(c) RTT




(d) Number of connections

Figure 3.23: Performance of *Pink*, *Aqua*, and *Silver* users.

The right plots of Fig. 3.23 show the behavior of the four parameters related to the *Silver* user. This user has a few connections but most of the traffic comes from only one of them. Such connection is established with an IP address assigned to AOL[7], and belonging to the domain stream.aol.com. Therefore it is very likely to be a video streaming flow. Moreover, most of the traffic is related to only one BTP with a large size (about

[7]http://www.aol.com

45MBytes) and duration (about 1h). The throughput achieved by this BTP is about 100 Kbps. While this may seem to be related to the video characteristics (for some codecs the bitrate can actually be around this value), the real cause of such low throughput is the high number of retransmissions, which are shown in the left plot of Fig. 3.23(b). In fact, we observed that, for most users, the traffic from this kind of video streaming applications (typically based on Adobe flash or similar products[8]) resembles more a web download (big packets with small inter-packet time) than a typical video streaming flow (i.e. VBR traffic). On the other hand, the RTT presents low values, which increase when multiple parallel connections are established.

### 3.3.4 Summary and conclusion

In this section we a presented methodology and the results we obtained from a passive measurement campaign on a cellular network, which comprises different kinds of technologies (GSM, GPRS, EDGE, UMTS, UMTS+HSDPA). The methodology relies only on the information contained in the IP and TCP headers, and can be therefore applied to other network scenarios and by other researchers. After illustrating such methodology, we presented a general overview of the performance experimented by all the users of the network, and a deeper investigation of the behavior of some selected users. The results we obtained allow to confirm some findings of simulative and theoretical studies, as well as to highlight some new issues. In particular, we showed that: i) current cellular networks present a very heterogeneous scenario in terms of both user performance and behaviors; ii) opening more concurrent TCP connections can provide either an increase or a decrease of the performance depending on the network condition.

We believe that cellular networks are still evolving, and they have the potential to become one of the most popular Internet access technology. However, in spite of their popularity, the performance of such networks still needs to be throughly analyzed. In this section we presented an approach useful for this aim.

## 3.4 Final remarks

In this chapter we showed how we can measure the performance parameters of heterogeneous networks. We introduced the network scenarios characteristics of today's Internet,

[8]http://www.adobe.com/flashplatform

that have been used to perform our activities. Such activities allowed to provide innovative contributions in the field of network monitoring and measurement, in terms of both improved knowledge of the characteristics of current Internet scenarios, and new methodologies and techniques to infer the status of a network.

In particular, we discussed one of the main limitations of PlanetLab, and how we overcame it. Using this solution and an active measurement technique, we analyzed the similarities and differences between the performance of two end-to-end paths connecting the same hosts of this infrastructure. This analysis allowed to understand the characteristics of UMTS connections when compared to Ethernet ones.

Afterwards, we presented our activities related to a wireless wide area network called Magnets. We discussed the main characteristics of such infrastructure, and we presented a deep active measurement campaign aimed at understanding all the possible aspects of the performance of Magnets. One of the main contributions of this analysis is the understanding of the importance of considering also the environmental impact on the performance of wireless WAN.

Using active measurement techniques, we also investigated the performance of a laboratory testbed called Hetnet. The peculiarity of such testbed lays in its heterogeneity in terms of access networks, host types and operating systems, etc. This characteristic allowed us to understand the impact of all these variables on the obtained performance.

We finally presented an analysis of an operational cellular network carried out by using passive measurement techniques. We showed the limitation of a state-of-the-art methodology for performance analysis, and described a new methodology we devised to overcome such problems. Insights provided by the application of this methodology to the traffic of real users of the cellular network were also reported.

Thanks to knowledges acquired with all these activities we are now ready to tackle the problem of realizing informed time and space diversity techniques.

# Chapter 4

# Solutions for informed path diversity

In this chapter we describe the contributions we provided in the field of space diversity at packet level, also known as path diversity. The literature on path diversity is wide, and several interesting works have been published. However, different problems are still open, and there is space to provide innovative contributions.

In the next sections, we firstly introduce the general problem of packet-level space diversity. Then, we provide the definitions we used in this work. Afterwards, we study the problem of path diversity, developing a simulation environment in Matlab. In this activity we use a basic scheme for path diversity, that is the round robin. After that, we describe a new technique we proposed for informed path diversity. We analyze the performance of our technique, comparing the achievable benefits with those of two well-known techniques called weighted round robin and join the shortest queue. Obtained results allow to understand that our proposal is actually able to achieve better performance, thanks to the knowledge of the status of the network. Finally, we describe the contributions related to the development of a tool to experiment with informed path diversity in real networks. We discuss the design choices at the base of the tool, and the implementation issues we had to face. In the final part of the chapter, we report the results obtained by using this tool in a controlled environment. Such results show that informed diversity schemes allow to obtain better performance with respect to state-of-the-art techniques.

## 4.1 Introduction

The analysis of the literature reported in Section 2.3.2 evidenced that several approaches have been proposed in literature to exploit the benefits of space diversity at IP layer.

However, as explained in Section 2.4.1, several problems are still open.

Path diversity can be applied on different time scales, that range from an entire packet-flow duration (i.e. few minutes) to an inter-packet time (e.g. few microseconds on a high-speed link) [1]. Moreover, path diversity can be used to pursue different objectives: find the best path for a specific flow, use multiple path for a single communication in order to achieve higher throughput and lower delay, etc. Clearly, the more we use a small time scale, the more we can exploit the differences between the available paths. At the same time, using a small time scale implies several issues: 1) difficult implementations, especially in high speed networks; 2)the measurement technique has to provide information with a very high frequency, which can cause an unacceptable overhead; 3) sending packets of the same flow on paths with different delay characteristics can cause performance problems to transport protocols such as TCP because of packet reordering. Therefore, to obtain a net benefit, the path diversity scheme has to compensate the performance problems caused at transport layer.

In this thesis we propose to use path diversity at IP layer independently from the particular application or networking scenario, and relying on network measurements to acquire the status of the available paths in order to make informed decisions. To achieve this goal several issues have to be solved, and a careful methodology is necessary.

For this reason, we firstly develop a Matlab simulation environment, which reproduces a simple path diversity scheme and emulates the behaviour of the network using models from literature. The simple simulation environment allows to concentrate on specific aspects of the problem under analysis, excluding interference from other variables that may possibly influence the obtained results. Our simulator, described in details in Section 4.3, does not implement any transport protocol or application, and it uses a 2-state Markov chain to model the loss process on the network. Performing a large set of experiments, we show that benefits in terms of loss decorrelation can already be obtained with a simple round robin packet distribution policy.

Then, we progressively introduce more variables in our analysis until a complete understanding of how to deploy a path diversity scheme in a real network. In particular, we firstly devise a new technique for packet scheduling called OPI and based on Markov Decision Processes. The available paths are seen as queues, and a Markov Decision Process is setup to take the right decision on how to send the next available packet. Moreover, a

[1] A good overview of these aspects is reported in [59]

path state monitoring mechanisms is introduced, in order to provide information to the decision maker about the consequences of its choices. Using a reward function, that takes into account the requirements of the application, it is possible to minimize or maximize a function of the QoS parameters (e.g. minimize the flow transfer time). The approach is evaluated performing *ns2* simulations. *ns2* allows to consider more complicated scenarios with respect to our first simulation environment, because it implements the entire protocol stack and uses state-of-the-art models for physical channels. We apply our path diversity technique in different heterogeneous environments, including overlay and wireless networks. A comparison is performed with two techniques called weighted round robin and join the shortest queue. The results show that OPI allows to obtain better performance.

In the successive step we tackle the problem of how to implement the technique previously described in a real scenarios. In a preliminary analysis we verified that the time resolution used in that technique is not suited for its application in general networking scenarios, because of the overhead of the path monitoring mechanism when operating at high-speed. In order to pursue our aim of deploying path diversity at IP layer, independently from the particular application or networking scenario, we therefore decided to increase the switching period. This is motivated also by the fact that in our simulations we verified that almost the same benefits can be achieved with an higher switching period. We therefore developed an application that implements this new approach. We tested the application in different environments, that comprise virtual machines and real testbeds, and we compared the achievable benefits with those of round robin. The obtained results show that our tool allow to achieve better performance.

## 4.2 Definitions

- **Informed path diversity**. It is a path diversity scheme that acquires the status of the network by means of passive or active measurement techniques.

- **Scheduling policy**. It is the way the packets are sent over the available paths. Basically, in any path diversity scheme it has to be decided which packet(s) to send over which path. This is the scheduling policy.

- **Multi-path**. It is a path diversity scheme in which the scheduling policy operates

with the granularity of a single packet. For example, a round-robin multi-path will send consecutive packets on different paths.

- **Path switching**. It is a path diversity scheme in which the scheduling policy operates with the granularity of a time period larger than the packet inter-arrival time. This means that using the path switching more than one consecutive packet will be sent over the same path.

- **Loss pattern**. It is a loss process characterized by means of a 2-state Markov chain with parameters $\pi_b$ (loss rate) and $\rho$ (loss correlation). More information about the 2-state Markov chains to model packet loss are reported in Section 4.3.1.

For the analysis of path diversity, we perform different simulations using different environments, as explained in the following. Each of these simulations is based on a specific set of assumptions as they are aimed at investigating different aspects of path diversity. The assumptions are therefore reported in Section 4.3.1 and in Section 4.4.2, which describe the different simulation environments.

## 4.3 Basic path diversity

One of the most troublesome aspects of the study of path diversity is the tight link that exists between the attainable performance and the loss pattern. Starting from this consideration, we setup a simulation environment using models able to reproduce the specific characteristics of the different data transmission scenarios that could benefit from path diversity.

Such a setup required the joint modeling of the aspects regarding the network with those more strictly pertaining to working details of the technique. In the following paragraphs all the choices will be shown, together with the solutions adopted for the development of the simulation environment used in the analysis.

### 4.3.1 Simulation environment

In the following we describe the models we used for the end-to-end path and the loss process on it. Then, we illustrate how such models have been used in the simulator, to understand the potential benefits of basic diversity techniques.

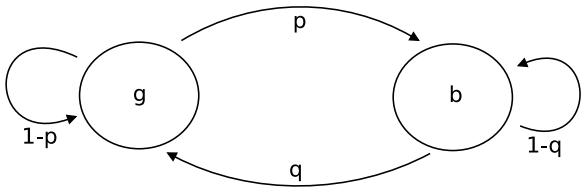



Figure 4.1: 2-state Markov chain.

### Loss and network model

The loss process over the Internet has been typically characterized by means of the continuous-time Gilbert-Elliott model [74, 75], a 2-state continuous-time Markov chain (2-MC) [124, 40]. Such model captures the correlation of the loss process over the Internet [25]. A detailed discussion of the 2-MC is reported in Section 5.3.1. Here we report the most important parameters of the model, which have been used in these simulations.

Let $X = \{X(t) : t \geq 0\}$ be the aleatory process of losses following the Gilbert model. The state $X$ at time $t$ can assume one of the following values: $b$ or $g$ ($b$ = "bad" and $g$ = "good"). Process $X(t)$ at a fixed time is characterized by the parameters $\mu_g$ and $\mu_b$, which can be though as the rates at which the chain passes from state $g$ to state $b$ and vice versa. A diagram of the 2-state Markov chain is reported in Fig. 4.1.

In general, when $X(t)$ is in state $b$ the probability to loose a packet is much larger than that in state $g$. To simplify the problem we assume that this probability is equal to 1 in state $b$ and to 0 in state $g$. For this reason we will refer to the two state as "loss" and "no-loss" beside "bad" and "good" respectively. It is worth noting that we consider a packet to be lost either when it is not delivered to the destination, or when it is delivered too late, and it is therefore not useful for the application (e.g. audio samples arriving after playback time).

The steady-state probabilities for the two states are $\pi_b$, which is the probability of staying in state $b$, and $\pi_g$, which is the probability to stay in state $g$. They can be evaluated using the following

$$\pi_b = \frac{p}{p+q}, \quad \pi_g = \frac{q}{p+q} \tag{4.1}$$

In particular, $\pi_b$ represents the average loss probability. $\rho$ is another important parameter of this model, and it is typically considered as the channel memory because it represents the correlation between losses. $\rho$ is defined as

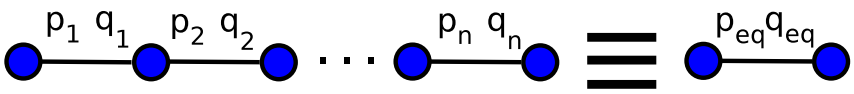



Figure 4.2: End-to-end equivalent channel.

$$\rho = \frac{1-q}{p} \tag{4.2}$$

To model the end-to-end path, we consider it as a series of a number of links characterized by burst losses. If every link is modeled using a 2-MC, and the links are independent, it has been demonstrated [28] that the end-to-end path can still be modeled using a Gilbert model with equivalent parameters. In facts, given a series of $N$ links of a path, modeled with 2-MC with parameters $(p_1, q_1), (p_2, q_2), \ldots, (p_N, q_N)$, it is possible to obtain an equivalent model with parameters $(p_{eq}, q_{eq})$, which represents the entire end-to-end path (or equivalent channel) as shown in Fig. 4.2.

### The simulator

Starting from the results of [28], it seems reasonable to adopt the "equivalent channels" model for the analysis of path diversity. The study of this data transmission technique has in fact been made thanks to the setup of a simulator, developed in the *Matlab* environment, based on a *equivalent channels model* for the representation of the network behavior. The model has been completed, according to the case, with the blocks needed for the simulation of the path diversity.

The simulation scheme is based on the following hypotheses:

- A single source has to send a number of packets to a single destination.
- The total amount of packets is available from the beginning of the data transmission.
- Each channel has enough bandwidth to transport the full bit-rate needed by the transmission.
- The source knows a-priori the number of available paths.
- Each path is selected by a common network-level routing algorithm or by relay, and it has no shared links with other possible paths.

- Each path is represented by an equivalent channel modeled by a discrete 2-MC with parameters ($p_eq$,$q_eq$); such parameters depend directly on the packets sending rate $S$, and embed the effects of the background traffic.
- The transmission will be analyzed at application level.

As regards the basic communication scheme, in absence of path diversity, we have the following steps: the transmitting application produces a number of packets, of a fixed number of bytes, characterized by a sequential number; the channel introduces possible losses, but does not change in any way the sequential order; the receiving application is able to reconstruct the data flow. The hypothesis of having routers with single queue with FIFO queueing policy would of course verify the assumption that the original order of packets is not altered on the path. On the other hand, as we are interested in the evaluation of a technique that aims at reducing the correlation of the loss pattern, we do not consider reordering in the network. Neglecting the network reordering of the packets means working in the worst case scenario, as we eliminated an additional possible source of uncorrelation for the loss pattern. This allows us to evaluate the potential of path diversity in an objective way, focusing only on the benefits that are intrinsically attributed to it.

The first step for the setup of the simulation model regards the choice of the representation of the data flow. A set of packets produced by the source will be represented by an array of $n$ cells. Each cell corresponds to a time slot available for the transmission of a single packet, and it will contain, in absence of path diversity, the sequence number of the packet as originally ordered (equal to the cell index).

A loss introduced by the channel in a given time slot will be represented by the substitution of the value in the cell corresponding to that time slot with a 0. In the following, the cell arrays will be referred to as packet sequences or traces.

A Matlab function we developed, `path.m`, reproduces the behavior of the single virtual channel, and it presents the following interface:

$$U = path(I, loss, rho)$$

This function receives in input the vector $I$ constituted by the sequence of indexes of packets produced by the source application. Moreover, it receives the values of *loss* and `rho` with the following meaning:

- $loss \equiv \pi_b = \frac{p}{p+q}$.
- $rho \equiv \rho = \frac{1-q}{p}$.

Using the values of `loss` and `rho`, it calculates the related values of $p$ and $q$ through the following equations:

$$p = \frac{loss}{1 + (rho - 1)loss}, \tag{4.3}$$

$$q = \frac{1 - loss}{1 + (rho - 1)loss}. \tag{4.4}$$

Those values, as previously said, embed the information related to both the characteristics of the considered data flow and the background traffic. Thanks to the values of $p$ and $q$, the function generates a sequence of losses, denoted with a 0, and no-losses, denoted with a 1, according to a Gilbert model; this loss sequence is "applied" to the vector $I$ to obtain $U$.

The $U$ vector represents the sequence of packets as they arrive to the destination, being denoted by their own sequential number, in case of correct reception, or by 0 otherwise.

If, as an example, we want to transmit 16 packets on our channel, we will have the following values for $I$, the random "mask" of losses, and resulting $U$:

| $I$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mask | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** |
| $U$ | **1** | **0** | **3** | **4** | **5** | **0** | **0** | **8** | **9** | **0** | **11** | **0** | **13** | **14** | **15** | **16** |

Another important function is `path-diversity.m`, that reproduces the behavior of the transmission strategy analyzed by the model. The function presents the following interface:

$$U = path - diversity(I, N, loss, rho)$$

This Matlab module simulates the transmission of a data sequence $I$ from a single source to a single destination, using $N$ equivalent channels. The paths are supposed disjoint and homogeneous, in the sense that each of them is modeled as a discrete 2-MC with the same values of `loss` and `rho`.

### 4.3.2 The adopted metrics

Besides the total amount of lost packets, for many applications such as audio and video streaming, it assumes relevant importance the way losses are distributed. Different loss

patterns can indeed generate different perceptions of QoS at the application level [55]. Recent studies [65][41][125] about video transmission using codecs such as H.264, point that, given the same amount of lost information, the signal-noise ratio (SNR) can be significantly lower in case of strongly-correlated losses. As a consequence, the perceived quality of video can be heavily degraded.

It becomes therefore clear that the fraction of lost packets is not adequate to understand the benefits of path diversity. One of the main advantages of this technique is indeed the decorrelation of the loss pattern. For this reason, we need parameters able to catch this particular aspect. The choice of using the loss fraction alone appears even more inadequate in our case if we consider that the path diversity using $N$ homogeneous paths [65] does not reduce of the total amount of losses, but performs just a redistribution of them aiming to reduce bursts, i.e. consecutive losses.

As we adopt a Gilbert model for the description of the equivalent channels, we can express the probability of loosing a packet in the multiple paths scenario $P_mp$ as:

$$P_{pm}=\sum_{i=1}^{N}\alpha_i\pi_b(i)=\sum_{i=1}^{N}\alpha_i\pi_b=\pi_b \tag{4.5}$$

where $\alpha_i$ the fraction of the total information traveling on the $i$-th path, and $\sum_{i=1}^{N}\alpha_i = 1$. This means that the probability of loosing a packet in the case of multiple paths is the same of that in the case of a single path. Starting from these considerations, we decided to extend the number of parameters considered in our study, in order to actually have a wide set of analysis tools.

It is worth stating that our simulator provides as output traces as they arrive to the receiving applications. Such traces allow to evaluate a large number of metrics for the study of path diversity, comprising:

- Loss fraction;
- Distribution of loss runlengths (i.e. of the groups of consecutive lost packets);
- Average loss burst length;
- Variance of loss-burst length;
- Distribution of no-loss run lengths (i.e. of the groups of consecutive successfully delivered packets);

- Average no-loss burst length;
- Variance of no-loss burst length;
- Video distortion

These parameters will be analyzed in detail in the following, together with the Matlab functions to calculate them. Let us explain their meaning and the actual capacity to synthesize the characteristics of the loss pattern. Despite the distortion, a parameter extremely representative but valid for only one specific application, we can divide the other parameters in two sub sets, corresponding to the two most representative characteristics of a generic loss pattern. In fact the first three parameters give a very clear idea of how the losses are structured, while the remaining three ones tell us how much they are separated.

We preliminarily note that, even though the distributions comprise also the informations related to the average lengths and the variances, they are not easily readable. For this reason, we will use in the result presentation almost always the other two statistics that, even if less representative, are able to provide in a concise way a summary of the loss characteristics of the received data flow. Specifically, looking at the average values, we can immediately have an idea of the loss scheme we are working with, as we know how long in average the losses are, and how far in average they are apart. This kind of information is fundamental in the case of, e.g., FEC coding, for the choice of the amount of redundancy to be used.

Even if the mean values represent valid parameters for a preliminary analysis, they cannot throughly represent the characteristics of the loss patterns, and above all, they can not give any information about their regularity. For this purpose, the variances have been added, to give an idea of the trustworthiness of the mean values, or how regular the pattern actually is.

Another consideration worth doing, is about the choice of the metrics related to the loss-less (or no-loss) sequences. In fact, while the reason for the choice of the metrics regarding the loss-burst length are clear, it seems less obvious to take into account also the distance between losses. This choice may seem even more superfluous if we consider that path diversity with homogeneous channels should not reduce the probability of losses. This means that, given the amount of lost packets, the loss-burst length and the distance between losses constitute two sides of the same coin: the more we reduce the loss-burst length, the more the losses get close to each other. It seems therefore useless to take

into account also statistics related to no-loss runlength. The problem is that, in some cases, losses occurring too frequently close can generate effects analogous to loss-bursts, lowering the QoS. This happens, for example, when specific video codecs are used, and the distance between losses undergoes a given threshold (see [41]).

Let us analyze the Matlab functions we developed to evaluate these parameters. The function that calculates the loss fraction *losses_ratio.m* has the following interface:

```
n = losses_ratio(I)
```

This function receives in input a sequence affected by losses (`I`), and returns the fraction of losses associated to it (`n`).

The process of the burst length distribution $X$, related to the transmission of $N$, packets is a discrete-time, discrete-values process defined as:

$X(k) = m$ : "there have been $m$ occurrences of $k$ consecutive packet losses (i.e. $m$ loss-bursts of $k$ packets each)"; for $k = 1, \ldots, N$ and $m = 0, \ldots, N$

We intend the isolated packet loss as a degenerate case of burst with length equal to 1. If, for example, we consider a generic transmission of 50 packets on an equivalent channel, we may have the following realization of $X$:

$$X : \begin{cases} X(1) = 3 \\ X(2) = 1 \\ X(6) = 1 \\ X(i) = 0 \quad \text{for } i = 4, 5 \text{ and } 6 < i < 50 \end{cases}$$

The function developed to generate the outcome of the process $X$ is *countburst.m* and presents the following interface:

```
Distr_burst = countburst(I,k)
```

This function takes in input the sequence (`I`), and the number ($k$) that identifies the loss ($k = 0$ in our case) and returns an array (`Distr_burst`) having as its $i$-th element the number of occurrences of bursts of length $i$.

From the burst length distribution $X$, it is straightforward to calculate the other two considered metrics (the average and the variance of the burst length). The first one is calculated as:

$$E[X] = \frac{1}{M} \sum_{i=1}^{N} iX(i),$$

Where $i$ scans the lengths of the bursts, that are weighted by $X(i)$ and averaged on $M$ (the number of total burst occurrences). The variance of the burst length is equal, by definition, to:

$$Var[X] = E[X^2] - E[X]^2$$

where:

$$E[X^2] = \frac{1}{N} \sum_{i=1}^{N} i^2 X(i).$$

Both the average and the variance are calculated by the function *stat_burstl.m*, presenting the interface:

```
[M,Var] = stat_burstl(I,k)
```

This function accepts as input parameters the trace of the sequence `I` affected by losses (indicated by the value 0 for the parameter `k`). In the function body, a call is made to function *countburst.m*, and starting from the knowledge of the realization of $X$, associated with `I`, the average burst length `M` and its variance `Var` are calculated and returned as output.

The process $Y$ of the lossless sequence length is basically the dual of the process $X$ previously defined. It also is a discrete-time, discrete-value process. Considering the transmission of $N$ packets, the process $Y$ is defined as:

$Y(k) = m$ : "there have been $m$ occurrences of $k$ consecutive packets correctly delivered (i.e. $m$ times the inter-loss distance (ILD) has been found equal to $k$)";
for $k = 1, \ldots, N$ and $m = 0, \ldots, N$

The function developed to generate the outcome of the process $Y$ is *nolossesl.m* and presents the following interface:

```
Distr_nolossesl = nolossesl(I,k)
```

This function is the dual of the one used for loss bursts. It receives as input the trace of the sequence (`I`), and the number ($k$) that identifies a packet loss (in our case $k = 0$, and returns an array (`Distr_nolossesl`) having as its $i$-th element the number of occurrences of inter-loss distance (ILD) equal to $i$.

For the process $Y$, starting from the process distribution, we define the average and the variance in analogy with the ones defined for $X$. In this case, the metrics are calculated by the function *stat_nolossesl.m*, that presents the following interface:

```
[M,Var] = stat_nolossesl(I,k)
```

This function accepts as input parameters the trace of the sequence `I` affected by losses (indicated by the value 0 for the parameter `k`). In the function body, a call is made to function *nolossesl.m*, and starting from the knowledge of the realization of $Y$ associated with `I`, the average no-loss length `M` and its variance `Var` are calculated and returned as output.

In the following, we present the last parameter we considered, that is the video distortion. The theoretical model chosen for the computing the distortion has been proposed in [41]. This model has proved able to estimate the real evolution of the distortion associated to video, in the presence of generic loss patterns, being more accurate than the so called "additive models". These models do not account for the real losses pattern, as they assume that the total distortion is equal to the sum of the distortions related to isolated losses. In the following we report some of the final results from [41] for the estimation of the distortion of the coded video. For the sake of brevity we considered the most simple and representative cases, i.e. single loss, loss burst of length two, and two losses separated by a "lag".

Said $\sigma_s^2[k]$ the mean square error (MSE), i.e. the distortion initially related to the isolated loss of frame $k$, we can express the total distortion $D_s[k]$ for the case of single loss as:

$$D_s[k] = \alpha \cdot \sigma_s^2[k], \tag{4.6}$$

where $\alpha$ takes into account the effect of the propagation of the initial error due to intra coding and spatial filtering.

The total distortion in the case of a loss burst of length two is instead equal to:

$$D_s[k-1,k] = \sigma_s^2[k-1] + D_s[k-1] + D_s[k] + 2\rho\sqrt{D_s[k-1] \cdot D_s[k]}, \tag{4.7}$$

which is a sum of two single distortions, a cross-correlation term, and the initial distortion due to the loss of the frame $k-1$. The these last two terms characterize this model as opposed to additive models, and cause its greater effectiveness in representing the distortion.

The last case here reported regards the loss of two frames separated from a number $l$ of frames ("lag") not greater than $N$ ("intra update period", that is the distance between two subsequent frames of type $I$). We note that if $l > N$, the two losses can be considered

as independent, and the total distortion as additive. The distortion due to two separated losses localized in $k-l$ and $k$, with $l \leq N$, is given by:

$$D[k-l,k] = \vartheta(l) \cdot D_s[k-l] + \frac{\sigma^2[k]}{\sigma_s^2[k]} D_s[k], \tag{4.8}$$

where $\vartheta(l)$ takes into account the error attenuation capabilities of the decoding scheme depending on $l$, and $\sigma^2[k]$ corresponds to the MSE for frame $k$ due to the contributes from both the loss of frame $k$ and the error propagation related to the loss of frame $k-l$. It is worth noting that the distortion in equation 4.8 is a function of the distortions caused by the isolated losses of frames $k-l$ and $k$. The coefficients of these two distortions, that depend on $l$ and on the correlation between errors, represent another characteristic aspect of the presented model.

A very strong hypothesis is needed for the application of the described model: the initial distortion related to the loss of a specific frame has to be known, being previously measured and stored by simulating the loss event. So there is the need of "*pre-measured distortions*". In order to obtain such preliminary measurement, two widely adopted test video sequences have been used, known as *Foreman* and *Claire*. Each sequence is coded according to the video compression standard JVT/H.2L[2], in format QCIF, and it is 280 frames long, coded at 80 fps with fixed quantization level and 36 dB PNSR[3]. The first frame of each sequence is intra-coded and is followed by P-frames. Every 4 frames a slice is intra-coded in order to reduce the propagation error in case of losses. The intra update period is equal to $N = 4 \cdot 9 = 36$ frames. Starting from this data, a function for calculation the distortion has been created, in order to simulate the transmission with losses of the encoded video. The hypothesis is that each packet contains one encoded video frame, belonging to one of the two reference sequences, whose total distortions related to bursts are known.

Moreover, another simplifying hypothesis is adopted, by assuming that the losses are enough separated to let the distortions related to a generic loss pattern be approximated by distortions obtained for isolated bursts of losses. This hypothesis is not a limitation because, as shown in [41], losses separated by more of 10 packets can be considered as isolated, for what concerns the total distortion. Moreover, the distance between bursts

[2]also known as ITU-T H.264/AVC, corresponding to part 10 of the standard MPEG-4 ISO/IEC 14496-10.

[3]$PSNR = 10 \log_{10}(255^2/D_e)$ where $D_e$ is equal to the quantization error

is a measured parameter, that can be used to evaluate the validity of such hypothesis. From this assumption, we introduce another metric for performance evaluation of path diversity, the *total normalized distortion* $D_{tot}^{norm}$ related to the transmission of $N$ packets:

$$D_{tot}^{norm} = m_1 + m_2 \cdot D_2^{norm} + \ldots + m_N \cdot D_N^{norm}, \tag{4.9}$$

where $D_i$ is the distortion due to a loss burst of length $i$, $m_i$ is the number of occurrences of loss bursts of length $i$, $D_i^{norm} = D_i/D_1$, $D_1^{norm} = 1$.

A source of uncertainty comes from the method used to obtain the value of coefficients $D_i^{norm}$ with $i = 1, \ldots, N$. In fact in order to have such values for all $i$, the missing ones are obtained by *interpolation* or *extrapolation* on measured ones. For this purpose, the Matlab function *calculate_distortion.m* has been created, with the following interface:

```
distortion = calculate_distortion(B,D,n)
```

This function accepts as input the vectors `B` and `D`, containing respectively the burst lengths and the related normalized distortions (known), and the number $n$ that sets the maximum length at which the calculation must end (in practice corresponding to $N$). The function approximates the missing values with a first grade polynomial and returns the full vector of normalized distortions $D_i^{norm}$ with $i$ up to length $n$.

Having a full reconstructed distortion profile, the Matlab function *distortion_norm.m* has been defined in order to calculate the total normalized dispersion. This function presents the following interface:

```
Dtot = distortion_norm(I,B,D)
```

It accepts as input the trace `I` of the sequence affected by losses and the vectors `B` ed `D` containing respectively the burst lengths and the related normalized distortions (known). The function internally calls *calculate_distortion.m* to obtain all needed values of $D_i^{norm}$ and *countburst.m* to get the burst occurrences $m_i$. Then applying the (4.9) the value of the total distortion `Dtot` is returned as output.

As in the tests the vectors `B` and `D` contained always known data from the *Foreman* and *Claire* sequences, a simplified version of the function has been used, with the following interface:

```
[Dtot_F,Dtot_C] = distortion_new(I)
```

The function accepts as input only the sequence $I$ of packets with losses, while embeds the values related to the pre-measured sequences, and returns as output a vector of two values, corresponding to the total distortions in the cases $I$ is of type *Foreman* or *Claire*, respectively.

**Scheduling policy**

As scheduling strategy we choose **round robin**. If a packet in the sequence is sent through the path $r$, the next one will be sent through the $(r+1)|_N$ path where the symbol "$|_N$" denotes the *modulo-N* defined as: $(r)|_N = \varsigma + 1$ with $\varsigma$ equal to the reminder of $(r)/N$.

This solution, widely adopted for the study of path diversity ([124][65]), represents in the case of homogeneous paths the best load partition [126], besides being the most simple one from a complexity point of view. In fact, under the hypothesis of having more channels with the same characteristics in terms of losses and the same available bandwidth, there is no reason for not using them in the same way equally dividing the load.

In Section 4.4 we describe a new scheduling policy we devised to implement the informed path diversity.

### 4.3.3 Tests performed and obtained results

The data transmission model that has been simulated in the case of path diversity, considers the utilization of multiple equivalent channels for the transmission of the packets. Therefore, the original data flow produced by the application will be transferred through different paths, routing the packets according to the policy presented before. The various channels will be homogeneous for the loss characteristics in the sense that, if the same bitrate ($S = 1/\tau$) is transferred on each of them, they can be represented with discrete 2-MC all with the same values of $p_{gb}(\tau)$ and $p_{gb}(\tau)$.

In this first scheme of path diversity, we will hypothesize to use the channels all at the same bitrate that would be used by default by the transmission in the case of a single channel ($S = S_d$). This kind of path diversity provides a reduction, proportional to the number $N$ of channels, of the total delay, but on the other side it requires a total *throughput* that increases with $N$. According to the stochastic model we adopted for the channel, the hypothesis of using homogeneous paths at a same bitrate regardless of the

total number of used channels translates into the simple assignment of the same values of $\pi_b$ and $\rho$ to all the channels.

The received data flow, possibly affected by losses, is reordered according to the original sequence. Starting from these traces the values of the considered metrics are grouped for different scenarios. The simulation parameters and their values are shown in Table 4.3.3.

| Parameters | Used Values |
|---|---|
| packet number | 4608 |
| used paths | [1, 2, 4, 6] |
| $\pi_b$ $(loss)$ | [0.01, 0.03, 0.1, 0.25] |
| $\rho$ $(rho)$ | [1, 3, 8, 15, 30] |
| number of repetitions for simulation | 1000 |

Table 4.1: Values of the simulation parameters for path diversity.

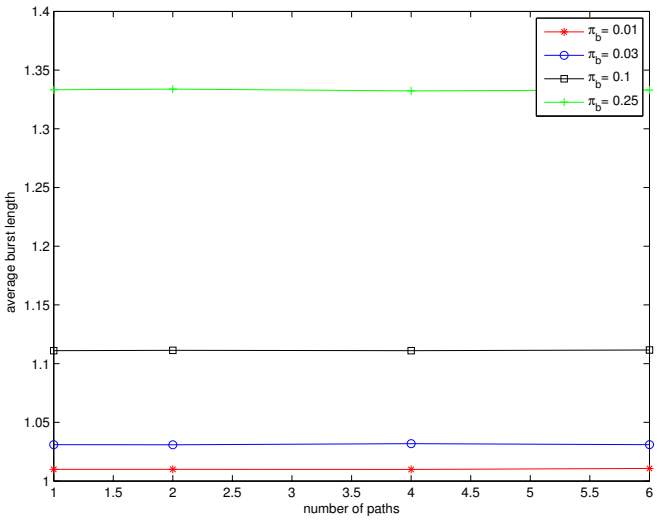



Figure 4.3: Average burst length, $\rho = 1$.

We observe that, in the case of memory-less channels (Figure 4.3), path diversity has no effect. The loss-burst length is constant regardless of the number of paths used. A different behavior is detected for channels with memory. In this case, in fact, we find a strong influence on the loss pattern related to the channel utilization.

Fig.4.4 - 4.6 report the values of the used metrics for four kinds of channels, characterized by four specific couples $(\pi_b, \rho)$:

- "good" channel : $\pi_b = 0.01$ , $\rho = 3$
- "medium-good" channel : $\pi_b = 0.03$ , $\rho = 8$
- "medium-bad" channel : $\pi_b = 0.1$ , $\rho = 15$
- "bad" channel : $\pi_b = 0.25$ , $\rho = 30$

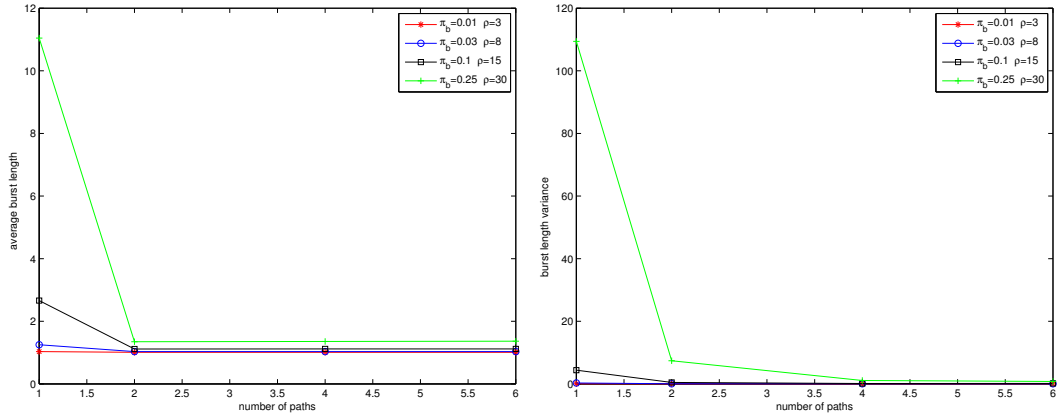



Figure 4.4: Average and variance of burst length with fixed $\pi_b$ , $\rho$.

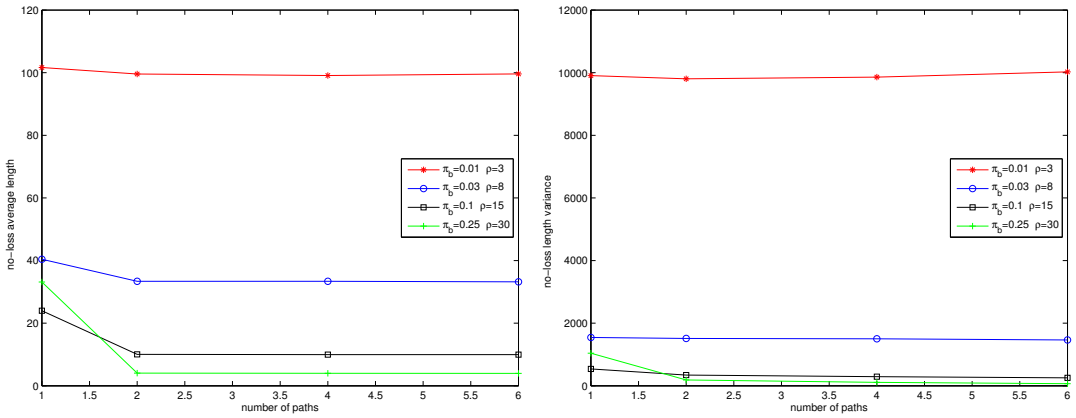



Figure 4.5: Average and variance of no-loss length with fixed $\pi_b$ , $\rho$.

By varying the number of used paths and the average loss probability for each fixed value of memory of the channel, we detect high effectiveness in the reduction of the burst length when two channels are used. But there is no improvement if we add further channels. Moreover, we notice that the burst length reduction is higher for increasing

values of loss rate and correlation. As an example, for $\rho = 30$ and $\pi_b = 0.25$, by using 2 channels we have a reduction in the average burst length of about 9.6 packets per flow (from 11 to 1.4), while for $\pi_b = 0.1$ and $\rho = 15$ the reduction is about 1 packet per flow. We can then state that, as regards the burst length reduction, the more memory the channel has, the more the path diversity is effective.

Analyzing the variance of burst length, we find that the minimum of variance is not achieved in correspondence of the number of channels that obtain the minimum of burst length. In fact, especially for high values of $\pi_b$, the minimum of variance is obtained for more than 2 channels (right plot of Figure 4.4).

As regards the proximity of the losses (average no-loss length), we observe a similar behavior: a reduction, when 2 channels are used, of the average distance between subsequent loss bursts due to dispersion (Figures 4.5).

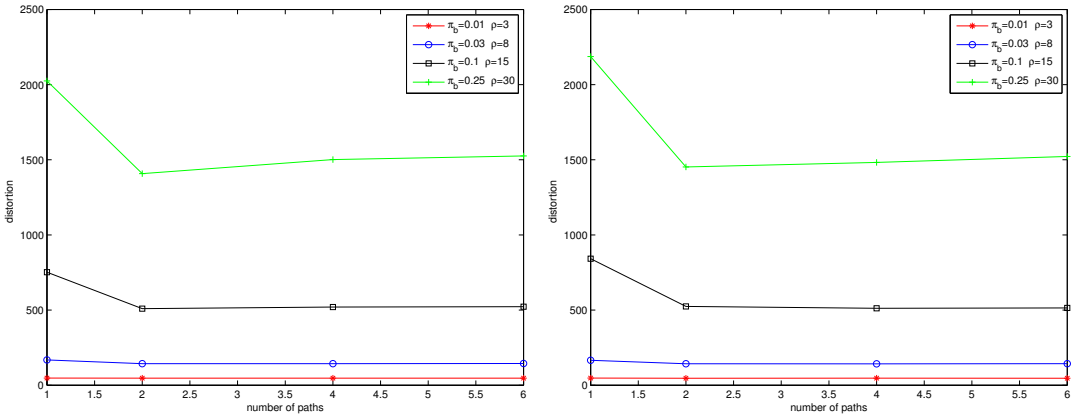



Figure 4.6: Distortion for the *Foreman* (left) and *Claire* (right) cases.

In Fig. 4.6 the results of Foreman and Claire cases are reported. The graphs show the effectiveness of path diversity in improving the QoS at application level, which is highest when the loss characteristics of the channel are the worst.

We can conclude that, whichever the considered metrics are, the effect of path diversity can be appreciated when the channels are not memoryless, and when two channels are used. Increasing the number of channels, there is no substantial improvement and, in the case of distortion, a slight worsening can also occur. So we find, in coherence with what is stated in [126], that choosing a round robin scheduling policy does not guarantee that increasing the number of used channels increases benefits.

### 4.3.4 Conclusion

In this section we have studied a simple scheduling policy, i.e. the round robin, in a simulator developed in Matlab. The analysis of this classic (and uninformed) technique has shown how it is possible to achieve a performance improvement, exploiting the existence of multiple paths between a source and a destination. This motivates our next analysis, in which we propose an informed path diversity technique. As we will see, thanks to the knowledge of the status of the network, it is possible to achieve even better performance.

## 4.4 Informed multi-path

In this section we present an approach to distribute packets over multiple paths, which is based on Markov Decision Process (MDP), and allows to achieve better performance than Weighted Round Robin (WRR) and Join the Shortest Queue (JSQ) [127, 128].

Thanks to the existence of Markov property in many network characteristics, e.g. arrival processes, loss, and delay [129, 130], MDPs have been used in the networking field to address various decision problems (see e.g. [131]). However, this does not mean that applying an MDP framework is straightforward. Major issues lie in the formulation of decision problems as an MDP, and in the solution of the MDP.

In this work, we show its application to a set of data distribution problems, providing a solution to these issues. In particular, first, we design a simple network-path monitoring mechanism to track the state of the system under study. Second, by using the Imbedded Markov Chain technique, we demonstrate that the system, if observed at specific instants, possesses the Markov property. Third, using this result, we cast the data distribution problem into an MPD. Fourth, we propose a lightweight algorithm named Online Policy Iteration (OPI) to efficiently solve the MDP.

Then, we show the application of this novel distribution scheme in three network scenarios, where different objectives are pursued. We present the results of a complete simulation study, using ns-2[4], in both wired and wireless environments and with both UDP and TCP. This confirms the superior performance of our approach against classical schemes, including WRR and JSQ. In addition, we also investigate issues related to the impact of the: (i) number of nodes; (ii) number of paths; (iii) path diversity; (iv) inaccurate estimation of the traffic parameters.

[4]http://nsnam.isi.edu/nsnam/index.php

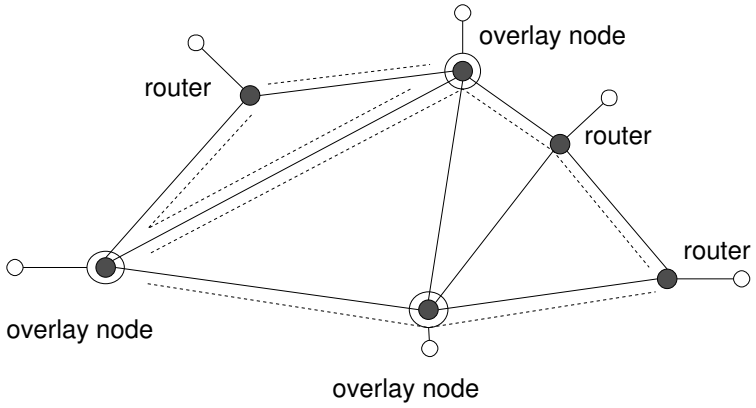



Figure 4.7: MPRON: An overlay network with multiple paths.

### 4.4.1 Considered scenario

To contextualize the problem under study, we introduce the notion of Multi-path Routing Overlay Network (MPRON). An MPRON consists of Overlay Agents (also referred to as relays or proxies) deployed by one or several ISPs on top of the underlying IP routing network. The aim of an MPRON is to provide multiple independent[5] overlay paths, connecting source and destination pairs of the underlying network. Hence, traffic can be routed between the network nodes through multiple paths, and the network resources can be more efficiently utilized.

Fig. 4.7 illustrates the conceptual MPRON, in which three overlay agents are used to provide multiple paths that connect each pair of the network nodes. From any node in this network, it is always possible to construct at least two independent paths connecting this node to the other nodes in the network, by going through appropriate overlay agents.

MPRONs constitute our first case study for informed multi-path. In this context, we want to solve the problem of optimal overlay data transfer in two scenarios. In the first scenario, an application (or user) wants to transfer an amount of data from a source to a destination. The application is fully aware of the existence of the multiple overlay paths and actively routes traffic over $M(M > 1)$ chosen paths. The data is segmented into constant-length[6] bins before being dumped with a constant rate $\lambda$ into an application

[5]In the context of this work, the term independent does not mean the paths are fully disjoint. Instead, they can share some non-congested links. Under what traffic condition the paths can be considered independent is discussed in Sec. 4.4.5.

[6]The fixed size here is assumed to simplify the formulation of the problem, which can be easily generalized for variable size.

module called *traffic distributor*. This module distributes the bins over $M$ connections, i.e. sockets, representing the $M$ overlay paths.

In the second scenario, the application is not aware of (or has no capability to distribute traffic over) multiple paths. Subsequently, it simply routes traffic to an appropriate Overlay Agent, which then forwards it to the destination using multiple paths. In this scenario we assume that the data arrives at the Overlay Agent with an average rate of $\lambda$. Since the arrival process is discrete, it is approximated using a Poisson process [132], with inter-arrival times between consecutive bins following an exponentially independent and identically distributed (i.i.d.) random variable. It is worth underlining that the arrival process here is not purely Poisson, but similar to Markov Modulated Poisson: the arrival process is considered Poisson($\lambda$) only in a period of time $T$, after which, $\lambda$ may change to another value. The i.i.d. assumption is reasonable since Internet paths are stationary over a scale of minutes [133].

Without loss of generality, we assume that the amount of data to be transferred in both scenarios, in terms of number of bins, is a random variable $\nu$ governed by a Geometric distribution[7] with parameter $\gamma$:

$$P(\nu = n) = (1-\gamma)\gamma^{n-1}, (0 \leq \gamma \leq 1); \;\; n = 1, 2, \ldots \tag{4.10}$$

Our objective is to construct a control strategy that specifies on which path, and at what time, each bin of data should be transferred to the destination to achieve the minimum transfer time for the whole data set.

### 4.4.2 System model and problem formulation

Previous studies on similar problems ([60, 135, 136]) when applying decision frameworks, often assume perfect knowledge of the network path characteristics, e.g. loss, delay and throughput. To avoid this assumption, we design a path state monitoring mechanism, which reveals loss and delay conditions of the path by keeping track of data bins being transmitted. The proposed mechanism works as follows. Before the traffic distributor sends a bin over a selected path, the bin ID and a timestamp are recorded on a list of size $K$. This information is kept until the bin is correctly received. In this manner, given the data stream, the evolution of the number of bins on the list will implicitly reflect the

[7] Vu *et al.* [134] supports this assumption in the case of overlay multimedia streaming.

path state in terms of Round Trip Time (RTT) and loss.

In a real implementation, the information regarding the transmission success can be retrieved directly from the transport layer protocols, in case of connection oriented protocols, or be explicitly acknowledged, in case of connectionless protocols[8]. At a first glance, this can be seen as a considerable overhead. However we have verified that, even if this information is obtained every 10 to 15 bins, the results presented in Sec. 4.4.4 remain almost unchanged. Fig. 4.8 illustrates the path state monitoring mechanism.

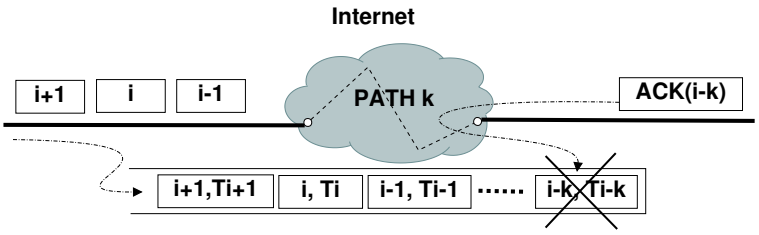



Figure 4.8: Path state monitoring mechanism.

From the traffic distributor viewpoint it is easy to see that, by using the path state monitoring mechanism, each path is modeled as a single server queue. The inter-departure intervals between "customers" in the queue are characterized by the distribution of RTT jitter[9]. Since each path is a single server queue, the system under study is a set of $M$ parallel queues (thanks to the monitoring mechanism the buffer can be dynamically allocated in order to avoid overflows). The system states are created by vectors of "customers" queuing in the system. Fig. 4.9 depicts the system model.

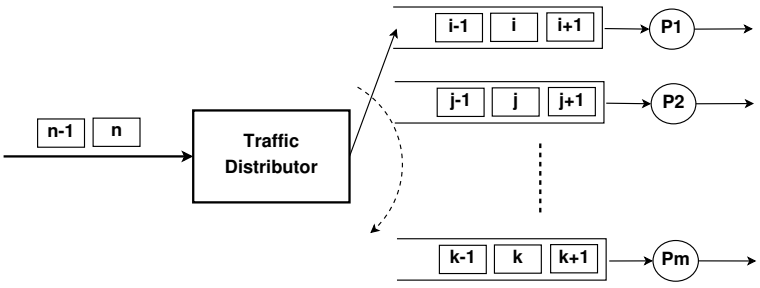



Figure 4.9: The system model of $M$ independent paths.

[8]To avoid the influence of congested reverse paths, acks should also carry a timestamp indicating the exact time in which the packet has been received.

[9]RTT jitter is defined as the absolute difference between two consecutive RTT values i.e. $Jitter_i = |RTT_{i+1} - RTT_i|$.

For each of the two scenarios presented in Sec. 4.4.1, we have a different arrival process: constant in the first, Poisson in the second. Having defined the system model and its states, in principle decisions can be made on how to optimally distribute data over the paths. Unfortunately, this system in general is not Markovian, i.e. the state of the system at an arbitrary time step may not depend exclusively on its previous state. Therefore, to apply the MDP framework, it is important that the Markov property of the system is identified. In the following section the Imbedded Markov Chain is applied to attain the Markov property of the system.

### Imbedded Markov Chain

At the time instant when a data bin arrives at the traffic distributor, a decision has to be made on where to send the bin. Depending on the decision, the system may change its state. Therefore, the system behaviors observed at the arrival instant, are important for the decision maker. To study these behaviors, the method of Imbedded Markov Chain [137] in combination with renewal theory [138] is applied.

Let $t_i$ be the arrival time of bin $i$ at the traffic distributor. In accordance, let $Q_m(t_i)$ be the number of bins found in the $m$-th queue at $t_i$. Consequently, the system state at $t_i$ is a vector $\{Q_m(t_i)\}, (m = 1 \dots M)$. We are interested in the evolution of $\{Q_m(t_i)\}, (i = 1, 2, 3 \dots \infty)$, under the impact of the data distribution actions taken by the decision maker at the traffic distributor. Assume that at arrival instant $t_i$, bin $i$ is sent to the $m-$th queue by the decision maker. To eliminate the possible transitive period and simplify further discussions, it is assumed that a decision and its action can be completed instantly. As a consequence, the system will move immediately from state $\{Q_1(t_i), Q_2(t_i), \dots, Q_m(t_i), \dots, Q_M(t_i)\}$ to state $\{Q_1(t_i), Q_2(t_i), \dots, [Q_m(t_i) + 1], \dots, Q_M(t_i)\}$. From this moment until the arrival of the next bin, the evolution of $\{Q_m(t_i)\}$ depends only on the number of bins departing from each queue during the $[t_{i+1} - t_i]$ interval. In other words, if $P_{QQ'}$ denotes the probability that the system is in state $\{Q'_m(t_{i+1})\}$ at the arrival instant of bin $(i + 1)$, then $P_{QQ'}$ equals to the probability that there are $Q_1(t_i) - Q'_1(t_{i+1})$ bins departing from the first queue, and $Q_2(t_i) - Q'_2(t_{i+1})$ bins departing from the second queue, and so on during the $[t_{i+1} - t_i]$ interval. Since these departure processes are independent[10], we can obtain $P_{QQ'}$ by taking the product of the probabilities.

[10]The departure processes are independent as long as the paths are independent.

To compute the probability that there are $Q_m(t_i) - Q'_m(t_{i+1})$ bins departing from the $m$-th queue, some results from renewal theory are applied. Let $\tau_k$ and $\tau_{k+1}$ subsequently denote the departure times of bins $k$ and $k+1$ from the $m$-th queue. Assume that the inter-departure intervals $[\tau_{k+1} - \tau_k]$ are i.i.d. random variables governed by a general distribution $g_m(x) = P(\tau_{k+1} - \tau_k \leq x)$. It is clear that the sequence of departure points $\{\tau_k\}$ forms an *ordinary* renewal process. Suppose the queuing process has attained a steady state. Subsequently, the sequence of departure points $\{\tau_k\}$ started from $t_i$ forms the *equilibrium* renewal process of the above ordinary renewal process [138]. Fig. 4.10 illustrates the queuing process.

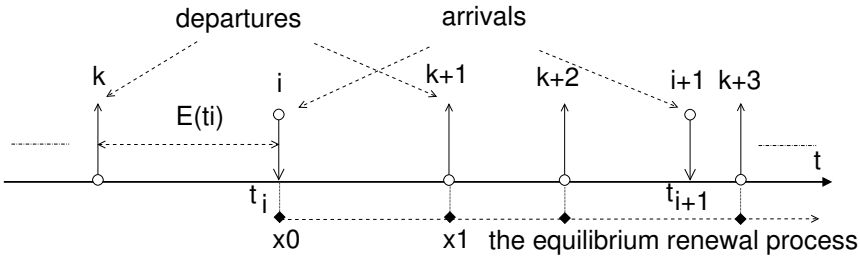



Figure 4.10: The queuing process of the $m$-th queue.

According to renewal theory [138] as applied for the *equilibrium* renewal process, the number of bins departing from the queue in the $[t_{i+1} - t_i]$ interval is proportional to the length of the interval and does not depend on $t_i$. Thus, $Q_m(t_{i+1})$ is fully determined by $Q_m(t_i)$ and does not depend on $E(t_i)$, which is the time elapsed since the last departure observed at $t_i$. Therefore, the random variable $Q_m(t_i)$ forms a discrete-state Markov chain embedded in the queuing process, whose state space is comprised of vectors of all of the possible numbers of bins in the queue. Hence, the Markov property of the system has been identified.

Let $\beta_j$ denote the probability of $j$ bins, $j = 0 \ldots K$, departing from the $m$-th queue in the $[t_{i+1} - t_i]$ interval. Since the sequence of departure times $\{\tau_k\}$ started from $t_i$ forms an equilibrium renewal process, $\beta_j$ is the probability of having $j$ renewals during the $(0, t]$ interval, where $t = t_{i+1} - t_i$. In the first scenario, the bins arrive at a constant rate. Hence, $t = t_{i+1} - t_i$ is fixed and determined by the data rate $\lambda$. Subsequently, let $P[N(u) = r]$ denote the probability of having $r$ renewals in an arbitrary interval $(0, u]$ of the equilibrium renewal process. We obtain:

$$\beta_j = P[N(t) = j] \tag{4.11}$$

In the second scenario, the arrival process is Poissonian with the average rate of $\lambda$. Hence, $t = t_{i+1} - t_i$ is a random variable governed by the exponential distribution with p.d.f. $\lambda e^{-\lambda t}$. Consequently, $\beta_j$ is computed as follows:

$$\beta_j = \int_0^\infty P[N(t) = j]\lambda e^{-\lambda t} dt \tag{4.12}$$

Recall that the inter-departure intervals $[\tau_{k+1} - \tau_k]$ are i.i.d. random variables with a general p.d.f. $g_m(x)$. Let $G_m(x)$ be the corresponding cumulative distribution function of the inter-departure intervals and $G_m^{(n)}(x)$ be the $n$-fold convolution of $G_m(x)$. Let $\mu_m \equiv E[\tau_{k+1} - \tau_k]$ be the mean of the inter-departure time of the *m-th* queue. Subsequently, as pointed out in [138], the probability of having $j$ renewals during interval $(0, t]$ of the equilibrium renewal process is obtained as follows:

$$\begin{aligned} P[N(t) = j] &= \frac{1}{\mu} \int_0^t (P[N^o(u) = j - 1] - P[N^o(u) = j]) du \\ &= \frac{1}{\mu} \int_0^t [(G_m^{(j-1)}(u) - G_m^{(j)}(u)) - (G_m^{(j)}(u) - G_m^{(j+1)}(u))] du \end{aligned}$$

where $P[N^o(u) = r] = G_m^{(r)}(u) - G_m^{(r+1)}(u)$ is the probability of having $r$ renewals during interval $(0, u)$ of the corresponding *ordinary* renewal process. Substituting received $P[N(t) = j]$ in (4.11) and (4.12), we obtain $\beta_j$.

As mentioned, the random variable $Q_m(t_i)$ forms a discrete-state Markov chain. Since the state transitions take place only at the arrival epochs, the system state $\{Q_m(t_i)\}, m = 1 \dots M$, also forms a discrete-state Markov chain. Having defined the formula for $\beta_j$, the state transition probability matrix of the chain is determined. Fig. 4.11 illustrates the system chain. The states with dotted lines are unobservable states.

### Transition Probability Matrix Computation

Although we are able to obtain the transition probability matrix of the Markov chain in the general form, it is challenging to compute the matrix with inter-departure intervals following a general distribution $g(x)$. Several approaches are available to cope with this computation, including simulation such as Monte-Carlo, reinforcement learning such as

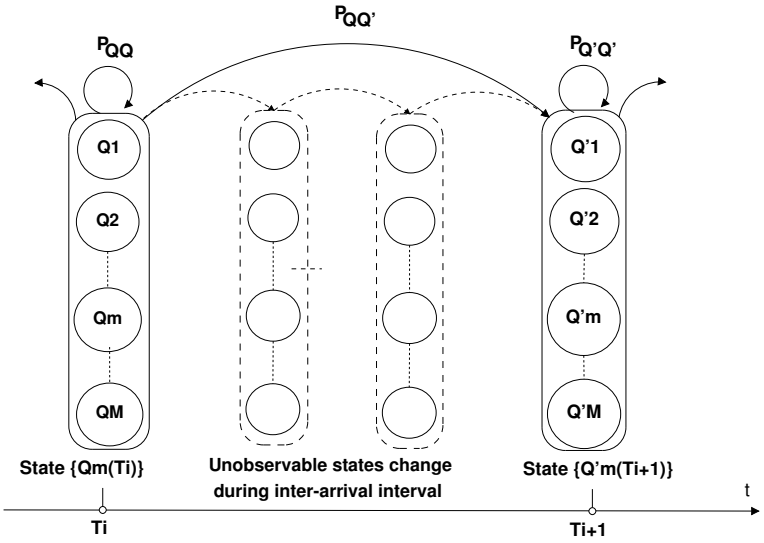



Figure 4.11: The Imbedded Markov Chain.

Q-learning, and function approximation. Due to space constraints, we present here only the function approximation approach. In particular, $g(x)$ is approximated by the $a$-stage Erlang distribution with rate $\rho$:

$$g(x) \approx \frac{\rho^a x^{a-1} e^{-\rho x}}{(a-1)!}$$

In Sec. 4.4.2, we show that the approximation is acceptable by studying RTT jitter of real heterogeneous networks. Since

$$\int_0^t \frac{\rho^{m+1} u^m e^{-\rho u}}{m!} du = 1 - \sum_{n=0}^{m} \frac{(\rho t)^n e^{-\rho t}}{n!}$$

$P[N(t) = j]$ can be computed directly:

$$
\begin{aligned}
P[N(t)=j] &= \frac{\rho}{a}\int_0^t \left\{ \sum_{m=ja-a}^{ja-1} - \sum_{m=ja}^{ja+a-1} \right\} \frac{(\rho u)^m e^{-\rho u}}{m!} du \\
&= \frac{1}{a}\left\{ \sum_{m=ja}^{ja+a-1} \sum_{n=0}^{m} - \sum_{m=ja-a}^{ja-1} \sum_{n=0}^{m} \right\} \frac{(\rho t)^n e^{-\rho t}}{n!} \\
&= \frac{1}{a} \sum_{n=ja}^{ja+a-1} (ja+a-n) \frac{(\rho t)^n e^{-\rho t}}{n!} \\
&+ \frac{1}{a} \sum_{n=ja-a}^{ja-1} (n-ja+a) \frac{(\rho t)^n e^{-\rho t}}{n!}
\end{aligned}
$$

Having obtained $P[N(t) = j]$, we can compute $\beta_j$ in each scenario. In the first scenario, by assumption, bins arrive evenly at the traffic distributor with a constant rate $\lambda$. Subsequently, $t = t_{i+1} - t_i = 1/\lambda$. By substituting into the above $P[N(t) = j]$ formula, $\beta_j$ is computed as follows:

$$
\begin{aligned}
\beta_j = P[N(1/\lambda)=j] &= \frac{1}{a} \sum_{n=ja}^{ja+a-1} (ja+a-n) \frac{(\rho/\lambda)^n e^{-\rho/\lambda}}{n!} \\
&+ \frac{1}{a} \sum_{n=ja-a}^{ja-1} (n-ja+a) \frac{(\rho/\lambda)^n e^{-\rho/\lambda}}{n!}
\end{aligned} \tag{4.13}
$$

In the second scenario, by assumption, bins arrive at the traffic distributor with inter-arrival time following the exponential i.i.d random variable governed by p.d.f. $\lambda e^{-\lambda t}$. As a consequence, $\beta_j$ can be computed as follows:

$$\beta_j = \int_0^{\infty} P[N(t) = j]\lambda e^{-\lambda t} dt \tag{4.14}$$

$$= \frac{\lambda}{a} \int_0^{\infty} \sum_{n=ja}^{ja+a-1} (ja + a - n) \frac{(\rho t)^n e^{-(\rho+\lambda)t}}{n!}$$

$$+ \frac{\lambda}{a} \int_0^{\infty} \sum_{n=ja-a}^{ja-1} (n - ja + a) \frac{(\rho t)^n e^{-(\rho+\lambda)t}}{n!} \tag{4.15}$$

$$= \frac{1}{a} \left[ \sum_{n=ja}^{ja+a-1} \frac{(ja + a - n)(\frac{\rho}{\lambda})^n}{(1 + \frac{\rho}{\lambda})^{n+1}} + \sum_{n=ja-a}^{ja-1} \frac{(n - ja + a)(\frac{\rho}{\lambda})^n}{(1 + \frac{\rho}{\lambda})^{n+1}} \right]$$

Given the $\rho/\lambda$ ratio, $\beta_j$ can be directly computed from the above formula. Obtaining $\beta_j$ for each queue, the state transition probability matrix of the Markov chain is fully determined. In real implementations, $\rho/\lambda$ can be periodically estimated from the evolution of the system queues. In Sec. 4.4.4, we show that the control mechanism is robust to a certain level of errors in the $\rho/\lambda$ estimation.

### MDP Formulation

A time homogeneous Markov Decision Process (MDP) [139] consists of a four-component tuple $\{S, A, P, R\}$ in which $S = \{s\}$ is a countable state space; $A = \{A(s)\}$ is a finite action space where $A(s) = \{a\}$ is a set of admissible actions in state $s$; $R : S \times A(S) \rightarrow \Re_+$ is a non-negative immediate reward function that maps action $a$ in state $s$ to a non-negative value $r$; and $P = \{p(s'|s, a)\}$ is a set of conditional probabilities, in which $p(s'|s, a)$ is the probability that the system moves from state $s$ to state $s'$ if action $a, a \in A(s)$ is taken. In the unconstrained MDP formulation, the MDP has a single objective, which is to maximize the cumulative (average) reward achieved over a period of time. In the context of the problem under study, the objective is to minimize the time required to transfer an amount of data. Subsequently, the four-component tuple of the MDP is made of:

- $S$ is the state space of the described system, which consists of $M, (M \geq 1)$ overlay paths. It is clear that $S$ is the state space of the embedded Markov chain.
- $A = \{a_1, a_2 \ldots a_M\}$ is the set of $M$ possible actions, each of which corresponds to the action of forwarding a data bin to one of $M$ overlay paths.

- $P$ is the state transition probability matrix of the Markov chain, which can be calculated as shown in Sec. 4.4.2.

- $R$ is the immediate reward function, which should reflect the minimum transfer time objective. According to Little's theorem [140], the average delay experienced by a bin is related to the average number of bins in the queue $\overline{D} = \frac{1}{\lambda}E[C_i]$. Therefore, we define the immediate reward function: $R(s,a) = \mu_a s + d_a$ where $\mu_a$ is the average inter-departure interval of the path chosen by action $a$; $s \in \{1, 2 \dots K\}$ is the state of the path; and $d_a$ is the path propagation delay. In a practical implementation, $d_a$ can be estimated as $0.5 \times \min(RTT)$. This estimation, however, does not undermine the generalization of the approach we propose.

A Markov policy is a description of behaviors, which specifies the action to be taken in correspondence to each system state and time step. If the policy is *stationary*, it specifies only the action to be taken in each state independently from the time steps. Given policy $\pi$ and initial state $s$ of the system, one can quantitatively evaluate $\pi$ based on the expected cumulative reward, which is defined as follows:

$$v^{\pi}(s,T) \equiv E_s^{\pi}\left\{\sum_{t=1}^{T} R(S(t), A_{\pi}(S(t)))\right\} \tag{4.16}$$

where $v^{\pi}(s,T)$ denotes the expected cumulative reward achieved by the decision maker from time step 1 to $T$ with initial state $s, s \in S$; $R(S(t), A_{\pi}(S(t))$ denotes the immediate expected reward received by the decision maker at time $t$ by taking action $A_{\pi}(S(t))$ while the system is in state $S(t)$ in correspondence with policy $\pi$. Recall that the amount of data (in the number of bins) to be transferred is a random variable $\nu$ governed by a Geometric distribution with parameter $\gamma, (0 \leq \gamma \leq 1)$. Subsequently, the expected cumulative reward obtained by the decision maker when using policy $\pi$ to transfer the data set is as follows:

$$v^{\pi}(s) \equiv E_s^{\pi}\left\{\sum_{n=1}^{\infty}\sum_{t=1}^{n} R(S(t), A_{\pi}(S(t)))(1-\gamma)\gamma^{n-1}\right\} \tag{4.17}$$

Since the delay has a finite value, (4.17) can be further simplified [139], which gives:

$$v^{\pi}(s) = R(s, a_{\pi}) + \sum_{s' \in S} \gamma p_a(s'|s) v^{\pi}(s') \tag{4.18}$$

where $R(s, a_\pi)$ is the immediate expected reward received by the decision maker when taking action $a$ in state $s$ in accordance with policy $\pi$ for the case of discrete state space; and $p_a(s'|s)$ is the probability that the system moves from state $s$ to state $s'$ when action $a$ is taken. In the problem under study, $p_a(s'|s)$ is computed as the product of corresponding $\beta_j$ as shown in Sec. 4.4.2. An optimal policy is the one that minimizes the cumulative reward $v^\pi$. Note that, it is sufficient to find an optimal policy in the Markov policy space, since for any history-dependent policy, there is a Markov policy that yields the same cumulative reward. In this MDP formulation, it is possible to find a stationary optimal policy since $(0 < \gamma < 1)$ [139].

**RTT Jitter Distribution**

Before explaining how our packet distribution algorithm works, it is important to justify the function approximation approach adopted in Sec. 4.4.2. To this end, we studied the distribution of the RTT jitter we measured over real heterogeneous network paths. The results provide an experimental proof of the hypothesis for which such jitter is well approximated by an Erlang distribution. The network scenarios on which the data was collected are part of a real heterogeneous wired/wireless testbed (see Section 3.2.3 for more details). To show our claim, we examined network paths with different characteristics. In particular, the following four types of paths were considered: (i) Ethernet-to-ADSL; (ii) Ethernet-to-802.11b; (iii) 802.11b-to-802.11b (ad-hoc mode); and (iv) 802.11b-to-802.11b (infrastructure mode). By taking into account RTT jitters related to various types of paths with different characteristics in both wired and wireless environments, we justify our assumption in more general and real network scenarios. To provide a graphical evidence of this assumption, in Fig. 4.12 we report the RTT jitter distribution for one of the above mentioned network scenarios. To provide a numerical evidence, we evaluated the discrepancy (by using the $\lambda^2$ [141]) between the empirical RTT jitter and three analytical distributions (Erlang, Normal, and Weibull). The obtained values, reported in Tab. 4.2, show that the Erlang distribution provides a good fit in all the considered scenarios.

### 4.4.3 Optimal data distribution algorithm

An optimal control strategy can be found by solving the formulated MDP. In principle, the standard dynamic programming techniques to solve an MDP (e.g. Value Iteration and Policy Iteration) can be used [139]. However, these techniques are computationally

Table 4.2: Discrepancy measures of Erlang, Normal, and Weibull.

| Type of path | Erlang | Normal | Weibull |
|---|---|---|---|
| Ethernet-Adsl | 0.22 | 1.60 | 0.46 |
| Ethernet-802.11 | 0.44 | 1.24 | 0.54 |
| 802.11-802.11(Ad-hoc) | 0.20 | 5.15 | 0.74 |
| 802.11-802.11(Infrastructure) | 0.25 | 3.89 | 0.48 |

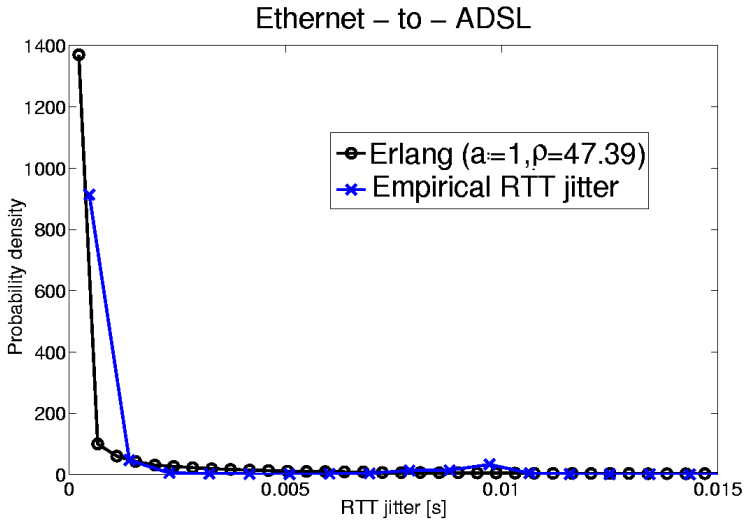



Figure 4.12: PDF of RTT jitter and Erlang in a real network.

expensive, and in general, they are not suitable for solving the problem under study. Fortunately, by exploiting the system specificity, a significant fraction of the computation time can be saved. Therefore, we propose a new computationally efficient algorithm namely OPI, i.e. On-line Policy Iteration, to optimally distribute data over multiple paths. The OPI algorithm is designed based on the idea of the Policy Iteration algorithm and asymmetric dynamic programming. The following interesting observations are taken into account to increase the computational efficiency of the algorithm.

First, dynamic programming and linear programming are computationally expensive, since they always sweep throughout the state and action spaces at every decision step to find an optimal action. The reason for such sweeping is that, in general, a system can move from one state to any other state, as a result of the action taken. Consequently, to find an optimal action in a particular state, it is necessary to evaluate every possibility in the state transition. Fortunately, in the system under study, the transition from one state to another is limited by physical constraints. For instance, it is impossible for a queue of

**Algorithm 1** Online Policy Iteration (OPI).

1: *// Initialize variables*
2: $\pi \leftarrow JSQ$ ▷ the starting policy is JSQ
3: $S = \{s_1, s_2 \dots s_M\}$, $s_i = \{1, 2 \dots K\}$ ▷ set of $M$ queue
4: $A = \{1, 2 \dots M\}$ ▷ set of $M$ actions
5: $R(s, a) = \mu_a s + d_a$ ▷ reward function
6: $\Lambda = \{\rho_1/\lambda \dots \rho_M/\lambda\}$ ▷ vector of path $\rho/\lambda$ ratio
7: $\Theta = \{\theta_1 \dots \theta_M\}$, $\theta_i = \{\beta_1 \dots \beta_K\}$ ▷ vector of $\beta_j$
8: $v^\pi \leftarrow \emptyset$ ▷ vector of state/action values
9: $T_{sta} \leftarrow T$ ▷ period the system is stationary
10: $T_{jsq} \leftarrow T$ ▷ period the system uses JSQ
11: *// Following JSQ for a first few steps*
12: **while** $(T_{jsq} > 0)$ **do**
13: **if** $(bin == true)$ **then** ▷ a new bin arrives
14: $s \leftarrow S$ ▷ obtain the current state
15: $a \leftarrow \pi(s)$ ▷ obtain the corresponding action
16: $DitributeData(a)$ ▷ send bin to path $a$
17: $T_{jsq} \leftarrow T_{jsq} - 1$
18: **end if**
19: **end while**
20: *// Estimate the system model parameters*
21: **for** $i = 1, 2 \dots K$ **do**
22: $\rho_i/\lambda \leftarrow N_i^{depa}/N^{arvl}$ ▷ estimation in JSQ period
23: **end for**
24: $\Theta = ComputeBetaJ(\Lambda)$ ▷ get $\beta_j$ for each path
25: *// On-line policy improvement*
26: **while** $done == false$ **do**
27: **if** $(bin == true)$ **then** ▷ a new bin arrives
28: $s \leftarrow S$ ▷ obtain the current state
29: $a \leftarrow \pi(s)$ ▷ obtain the corresponding action
30: $DitributeData(a)$ ▷ send bin to path $a$
31: $\Omega \leftarrow GetReachableStates(s, \Theta)$ ▷ from $s$
32: $v^\pi(s) \leftarrow R(s, a) + \sum_{s' \in \Omega} \gamma p_a(s'|s) v^\pi(s')$
33: *// Improve policy $\pi$ for state $s$*
34: $\pi(s) \leftarrow \min_{a \in A} \{R(s, a) + \sum_{s' \in \Omega} \gamma p_a(s'|s) v^\pi(s')\}$
35: **else if** ($\pi$ is NOT $\epsilon$-optimal) **then**
36: *// Improve policy $\pi$ for neighbor states*
37: **for** each $s \in \Omega$ **do**
38: $\omega \leftarrow GetReachableStates(s, \Theta)$
39: $\pi(s) \leftarrow \min_{a \in A} \{R(s, a) + \sum_{s' \in \omega} \gamma p_a(s'|s) v^\pi(s')\}$
40: **end for**
41: **end if**
42: **if** $(T_{sta} \leq 0)$ **then** ▷ re-estimate system model
43: **for** $i = 1, 2 \dots K$ **do**
44: $\rho_i/\lambda \leftarrow N_i^{depa}/N^{arvl}$ ▷ estimation in $T_{sta}$
45: **end for**
46: $\Theta = ComputeBetaJ(\Lambda)$ ▷ get $\beta_j$ for each path
47: $T_{sta} \leftarrow T$
48: **else**
49: $T_{sta} \leftarrow T_{sta} - 1$
50: **end if**
51: **end while**;

hundreds of packets to be empty in a millisecond. As a result, we can greatly increase the computational efficiency by taking those constraints into account, i.e. only practically possible states and actions will be evaluated when searching for an optimal policy.

Second, it is also common in practice that some states of a system are more frequently visited than others. These states are usually more important to the decision maker as they contribute more to the final outcome. Hence, instead of equally evaluating and improving the control policy for every state, we could spend more time on evaluating and improving the policy for those important states. This policy improvement strategy is known as the prioritized sweeping techniques, which guarantees an $\epsilon$-optimal policy will be found if all states of the system are visited infinitely often [139].

Third, dynamic programming, in particular Policy Iteration, is an incremental policy improvement process: the decision policy of the current step is statistically better than the policies of the previous steps. However, this improvement process is non-linear, i.e. a significant improvement is usually observed at some initial steps, followed by an incremental improvement in further steps. Thus, it is not necessary to wait until an optimal policy is found. A policy obtained after the first few steps is good enough in many cases. Furthermore, if a reasonable policy is used as the starting policy, the policy improvement process could quickly converge to an optimal policy.

The OPI pseudo-code is detailed in Algorithm 1. In brief, it works as follows:

**Step 1**: select JSQ as the starting policy and start to distribute data immediately using this policy for a few dozens of data bins. Since JSQ is an optimal policy for the case of two symmetric queues, and an acceptable policy for several other cases [142], a lot of computation can be saved by starting from JSQ. **Step 2**: estimate $\rho/\lambda$ ratio and compute $\beta_j$ for each path. **Step 3**: when a new data bin arrives, observe the current system state, take an appropriate action following the current policy to distribute the data bin, and obtain the immediate reward for the action taken. **Step 4**: do the policy improvement for the currently observed state, and if the time is sufficient, for the neighboring states. During the improvement process, only reachable states will be evaluated. In the system under study, only states with sufficiently large $\beta_j$ (e.g $\beta_j > 0.0001$) are reachable. Also note that this policy improvement will be carried out until an $\epsilon$-optimal policy is obtained. Afterward, no further improvement is required. Therefore, OPI is more beneficial for a large data transfer (e.g. live content streaming, file sharing ...). **Step 5**: if the stationary period $T$ has expired, go to Step 2 to update the system model. Otherwise, go to Step

3. Since the OPI algorithm is a special case of the Policy Iteration algorithm where the policy improvement step is done on the fly, the convergence of the algorithm is proved when $0 < \gamma < 1$. Details of the proof can be found in [139].

**Complexity Analysis of OPI**

In general, the OPI algorithm is lightweight and can be used for making decisions on the fly. More specifically, at each decision epoch, the decision is made instantly by mapping an observed system state to the current control policy. In case an $\epsilon$-optimal policy has not been obtained, only (4.18) has to be computed, the complexity of which is $O(N)$, $(N = sizeof(\Omega))$ for each $a \in A$ ($\Omega$ is the set of reachable states from a state $s \in S$). In the problem under study, $\Omega$ is significantly reduced by the path capacity and the data arrival process. Therefore, $N$ is usually in the range of hundreds to thousands, depending on the number of transmission paths $M$. For instance, in our study, $N = 25 \ldots 100$ for 2 paths, $125 \ldots 1000$ for 3 paths, $725 \ldots 10000$ for 4 paths.

### 4.4.4 Simulations and results

To evaluate the performance of the proposed approach, a simulation study using *ns-2* was conducted. In this section we show the comparison of OPI with WRR and JSQ, in network settings that allow to obtain the highest performance gain [143], while simulations in different network configurations are presented in Section 4.4.5.

**Minimal-Time File Transfer**

The network we used for the simulations was constructed in the following way: i) we took the topology of the Telstra's backbone; ii) we used the values of the link delays from Rocketfuel[11]; iii) we chose different capacity values for the various links to simulate different operating conditions. The resulting network is illustrated in Fig. 4.13 with the capacity and propagation delay of each link. Background traffic injected into each link was generated using a random number of FTP, CBR and ON/OFF traffic sources, the parameters of which, e.g. file size, sending rate, and on/off time were chosen randomly to create a certain level of variation in the overlay path performance (see Table 4.3 for more details). Fig. 4.14 shows the dynamics of the background traffic through the evolution of the bot-

[11]http://www.cs.washington.edu/research/networking/rocketfuel

tleneck link queue, both packet-by-packet and averaged every 250ms. The performance of WRR[12], JSQ, and OPI algorithms was compared in the following simulation scenarios:

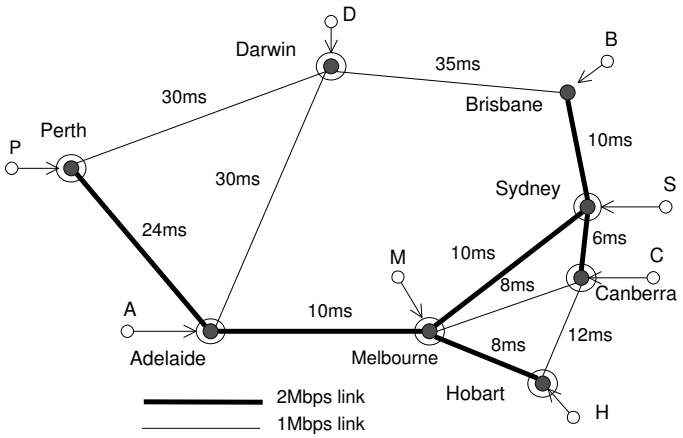



Figure 4.13: Network topology used for the simulations.

Table 4.3: Parameters of simulated background traffic sources.

| | **FTP** | **CBR** | **ON/OFF** |
|---|---|---|---|
| NumSrc per 100s | Rand[10-100] | Rand[10-50] | Rand[10-50] |
| Start time (s) | Rand[0-100] | Rand[0-100] | Rand[0-100] |
| Duration (s) | Rand[0-100] | Rand[0-100] | Rand[0-100] |
| Packet size (Bytes) | 1500 | 500 | 1000 |
| Rate(Mbps) | N/A | Rand[0.1-0.5] | Rand[0.1-0.5] |
| Data size | Rand[10-1500]KB | Rand[5-50]MB | N/A |
| Burst time | N/A | N/A | 500ms |
| Idle time | N/A | N/A | 500ms |
| Random/Shape | N/A | True (see *ns-2*) | Pareto(1.6) |

**Scenario 1**: A multi-path capable application located at Sydney wants to transfer multimedia contents to its partner located at Perth using an MPRON. The application establishes 2 overlay paths, which are Sydney-Perth via Brisbane-Darwin, and Sydney-Perth via Melbourne-Adelaide. The contents are segmented into a stream of 1500-Byte bins, and they are dumped into the application traffic distributor at a constant rate of 50 bins per second.

**Scenario 2**: A multi-path capable application located at Canberra wants to transfer multimedia contents to its partner located at Melbourne using an MPRON. The application establishes 3 overlay paths, which are Canberra-Melbourne direct route, Canberra-Melbourne via Sydney, and Canberra-Melbourne via Hobart to transfer the contents.

[12]Data bins are distributed proportionally to the path throughputs.

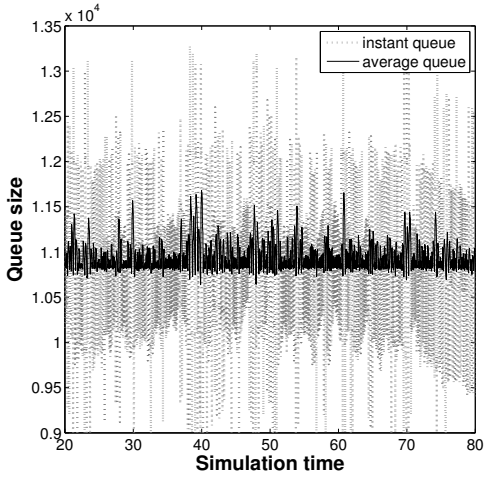



Figure 4.14: Simulated background traffic.

The contents are segmented into a stream of 1500-Byte bins and dumped into the traffic distributor at a constant rate of 100 bins per second.

**Scenario 3**: An application located at Adelaide wants to transfer multimedia contents to its partner located at Canberra using an MPRON. Since the application is not multi-path capable, it simply routes the contents to the MPRON relay located in Melbourne. The relay then forwards the received contents to Canberra using 3 overlay paths, which are Melbourne-Canberra direct route, Melbourne-Canberra via Sydney, and Melbourne-Canberra via Hobart. The contents arrive at the relay in a stream of 1500-Byte bins following the Poisson distribution with the average rate of 100 bins per second.

We use results of [143], claiming that most of performance gains are realized by using 2 to 4 overlay paths with an extra small amount of bandwidth. In addition, in real implementations, overhead introduced by a large number of path must be considered. We investigate the impact of the number of nodes in Sec. 4.4.5. In all the scenarios, the amount of data to be transferred was chosen between 10KB-100MB. Since the day-time traffic is usually characterized by a higher intensity and shorter transfer sessions, in comparison with the night-time, we simulated two scenarios using the following background traffic settings: (i) *Daytime setting*, the background traffic occupies $50\% - 70\%$ of the path capacity, and the data size subsequently is 10KB, 50KB, 100KB, 500KB, 1MB, 5MB, and 10MB; (ii) *Night-time setting*, the background traffic occupies $30\% - 50\%$ of the path capacity, and the simulated data size is 50MB, and 100MB.

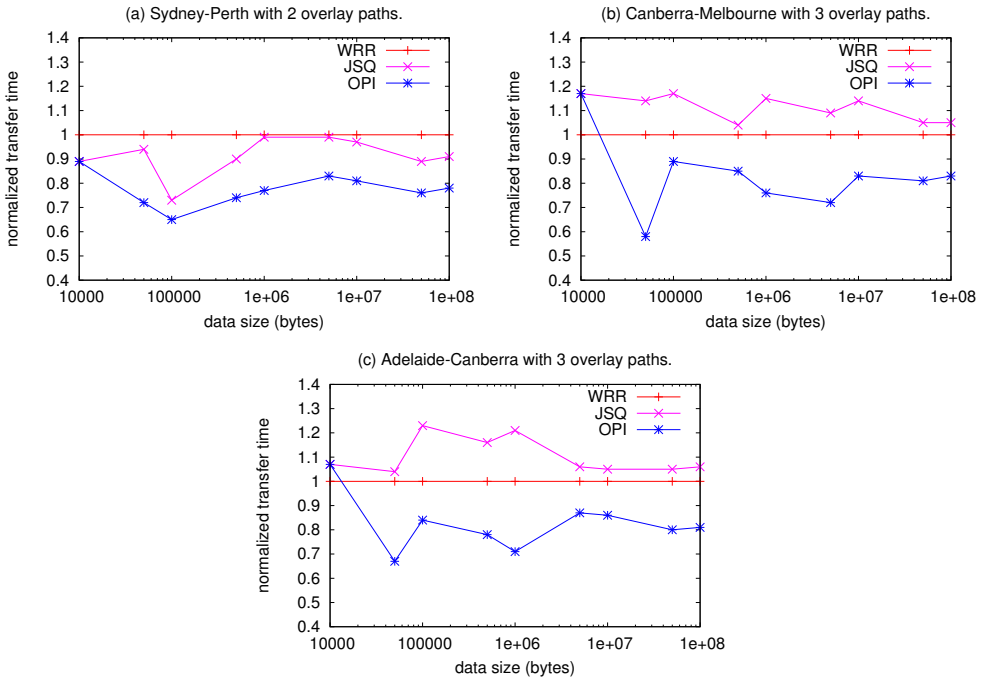



Figure 4.15: Performance comparison of WRR, JSQ and OPI.

The transport layer protocol in use was TCP. Unless otherwise specified, the simulation results are 10-run average, and presented in the form of the mean ± standard deviation. Tables 4.4, 4.5, and 4.6 show the transfer time obtained by each algorithm in the three simulated scenarios. In Fig. 4.15 we also report the results normalized over those obtained by WRR for visual comparison. As shown, OPI outperforms both WRR and JSQ in all the scenarios considered. In particular, OPI performs better than WRR and JSQ by 15% ca. in Scenario 1, and 20% in Scenarios 2 and 3. The OPI performance stabilizes after the warm-up period, which was 1MB ca. of transferred data. This indicates that OPI is capable of approaching an optimal policy within the first 600 iterations. We also see that the performance of the algorithms in Scenario 1 is less stable in comparison with that in Scenario 2 and 3, i.e. the variance of the transfer time is larger. We believe the reason is that the variation of the background traffic in Scenario 1 is higher since the paths are comprised of more physical links. Between Scenario 2 and Scenario 3, there is no significant difference in performance gain. This indicates that OPI works well with both constant-bit-rate and Poisson arrival processes.

Table 4.4: Sydney-Perth transfer time with 2 overlay paths.

| Data size | WRR (s) | JSQ (s) | **OPI** (s) |
|---|---|---|---|
| 10KB | $1.73 \pm 0.66$ | $1.54 \pm 0.28$ | $1.54 \pm 0.28$ |
| 50KB | $6.81 \pm 3.25$ | $6.38 \pm 4.61$ | $4.90 \pm 1.09$ |
| 100KB | $10.23 \pm 5.4$ | $7.43 \pm 2.54$ | $6.62 \pm 1.20$ |
| 500KB | $42.97 \pm 13.56$ | $38.71 \pm 7.43$ | $31.76 \pm 7.40$ |
| 1MB | $83.04 \pm 19.55$ | $82.55 \pm 17.71$ | $64.18 \pm 11.58$ |
| 5MB | $153.45 \pm 19.40$ | $152.39 \pm 13.22$ | $127.89 \pm 7.47$ |
| 10MB | $259.24 \pm 17.85$ | $252.56 \pm 14.53$ | $209.10 \pm 6.01$ |
| 50MB | $911.09 \pm 21.17$ | $811.91 \pm 28.56$ | $695.34 \pm 19.02$ |
| 100MB | $1823.58 \pm 48.39$ | $1666.57 \pm 41.69$ | $1419.04 \pm 29.25$ |

Table 4.5: Canberra-Melbourne transfer time with 3 overlay paths.

| Data size | WRR (s) | JSQ (s) | **OPI** (s) |
|---|---|---|---|
| 10KB | $1.05 \pm 0.15$ | $1.23 \pm 0.12$ | $1.23 \pm 0.12$ |
| 50KB | $2.39 \pm 0.41$ | $2.73 \pm 0.49$ | $1.39 \pm 0.13$ |
| 100KB | $3.15 \pm 0.88$ | $3.68 \pm 0.74$ | $2.79 \pm 0.41$ |
| 500KB | $8.07 \pm 1.30$ | $8.36 \pm 1.09$ | $6.88 \pm 0.77$ |
| 1MB | $17.14 \pm 1.82$ | $19.67 \pm 2.75$ | $13.09 \pm 1.76$ |
| 5MB | $79.42 \pm 5.96$ | $85.72 \pm 6.02$ | $64.40 \pm 5.03$ |
| 10MB | $141.74 \pm 5.51$ | $153.99 \pm 8.27$ | $127.39 \pm 4.64$ |
| 50MB | $625.75 \pm 17.07$ | $681.93 \pm 19.42$ | $523.48 \pm 18.77$ |
| 100MB | $1282.28 \pm 38.28$ | $1349.81 \pm 40.87$ | $1044.21 \pm 27.17$ |

Table 4.6: Adelaide-Canberra transfer time with 3 overlay paths.

| Data size | WRR (s) | JSQ (s) | **OPI** (s) |
|---|---|---|---|
| 10KB | $1.07 \pm 0.19$ | $1.15 \pm 0.09$ | $1.15 \pm 0.09$ |
| 50KB | $2.91 \pm 0.19$ | $3.03 \pm 0.24$ | $1.94 \pm 0.17$ |
| 100KB | $4.26 \pm 0.22$ | $5.26 \pm 0.27$ | $3.57 \pm 0.24$ |
| 500KB | $13.05 \pm 0.18$ | $15.14 \pm 0.20$ | $10.12 \pm 0.13$ |
| 1MB | $28.56 \pm 2.53$ | $34.47 \pm 4.69$ | $20.32 \pm 2.05$ |
| 5MB | $87.80 \pm 4.79$ | $93.11 \pm 5.03$ | $76.01 \pm 4.87$ |
| 10MB | $185.22 \pm 8.69$ | $194.41 \pm 8.74$ | $159.47 \pm 5.65$ |
| 50MB | $665.16 \pm 13.86$ | $697.57 \pm 13.03$ | $531.80 \pm 10.90$ |
| 100MB | $1321.63 \pm 20.56$ | $1397.38 \pm 26.18$ | $1069.87 \pm 23.75$ |

### Minimal-Distortion Streaming

In the previous settings, performance of the multi-path algorithms has been evaluated in the context of data transfer, where the transport protocol in use is TCP, and the control objective is to minimize the transfer time. In the context of live content streaming, e.g. IPTV, the content is streamed using datagram protocols, e.g. UDP, and the objective is often to minimize some audio-visual QoS parameters such as the distortion. In this section, we show that OPI, changing the reward function, can also be used to minimize the distortion when streaming content over multiple paths. In the streaming process, two main factors lead to the distortion: packet loss and delay. To quantitatively define the distortion, we introduce a notion of distortion rate ($r_d$) computed as follows:

$$r_d = \frac{\sum_{i=1}^{C_u} w_i}{\sum_{i=1}^{C_t} w_i}$$

where $C_t$ is the number of bins transferred; $C_u$ is the number of bins that could not be decoded due to losses or late arrivals (i.e. bins arriving after the decoding deadline); $w_i$ is the weight of bin $i$, which indicates the importance of the bin to the success of the content decoding process. For many coding schemes such as MPEG, the bins are correlated i.e. the successful decoding of one bin is necessary for the successful decoding of some other bins [135]. Thus, bins are not equal in terms of importance. Assigning weights to the bins is a simple approach to reflect this phenomenon.

As the distortion depends on losses and late arrivals, to minimize the distortion rate we have to minimize both the number of bins lost and the number of bins arriving late. To reflect the two objectives, the reward function of the MDP is, therefore, changed to the following:

$$R(s,a) = (\mu_a s + d_a)(1 - p_{s,a}^{loss})$$

where $p_{s,a}^{loss}$ is the probability that a bin is lost in state $s$.

We compared the algorithms in terms of distortion rate when streaming data from location Canberra to location Adelaide using 2 and 3 overlay paths. The overlay paths are Canberra-Adelaide direct route, Canberra-Adelaide via Sydney-Melbourne, and Canberra-Adelaide via Hobart-Melbourne. The content is streamed at a rate of 100 bins per second, with bin size of 500 Bytes, and using UDP. The simulation lasts for 1000 seconds. Fig. 4.16 depicts the average distortion rate obtained by using of the three distribution algorithms over 30 runs.

As shown, OPI outperforms WRR and JSQ in both the scenarios. In the case of 2 paths, OPI reduces the distortion rate of 15% ca. compared to JSQ, and of 23% ca. compared to WRR. In the case of 3 paths, OPI has even a higher performance gain, which is 28% and 26% in comparison with JSQ and WRR respectively.

### Minimal-delay and loss

Apart from the minimization of the file transfer time and of the distortion when streaming multimedia contents, we are also interested in the impact of the proposed multi-path control mechanism on QoS parameters, such as end-to-end loss pattern, delay, and jitter.

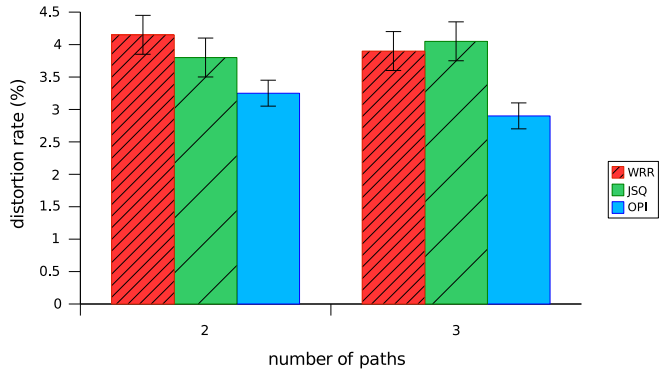



Figure 4.16: Average distortion rate using 2 and 3 paths.

To provide a clear evidence of the benefits of the proposed approach on QoS parameters, we use the same reward function of Sec. 4.4.4 but, instead to work on the network reported in Fig. 4.13, we investigate such impact in wireless environments, where the QoS parameters are more fluctuating.

For this analysis, we consider a scenario in which two wireless end points, i.e. cars with wireless capabilities, exchange data over respectively 2, 3, and 4 wireless paths while moving in parallel in a [1500m x 1500m] flat grid area, with speed of $5 - 10m/s$. With respect to Fig. 4.9, here the *traffic distributor* module is integrated into the end points. Each wireless path is comprised of 3 wireless nodes (2 hosts and an Access Point), which implement Ricean radio propagation model[13], 802.11 MAC with the basic rate of 1Mbps, $600m$ of transmission range, and Dynamic Source Routing Protocol [144]. To create disjoint wireless paths, different physical channels are used. Since in ns-2 the channels are orthogonal, the interference effect by default is not considered. To appropriately simulate transmission errors on 802.11 links, we implement a modification of the 802.11 error model[14] in conjunction with the Ricean fading propagation model. The two end nodes exchange data in a 500-Byte packet stream at the average rates of 2.0Mbps to 4.0Mbps depending on the number of paths in use. The stream of packets arrives at the traffic distributor following a Poisson distribution. The packets are then routed to the paths for transmission using 3 algorithms: OPI, WRR, and JSQ. The average loss rate (experienced by WRR) is approximately 6.0%. Table 4.7 shows the average, and variance in parenthesis, of delays and loss rates obtained by the algorithms with 2, 3 and 4 paths.

[13]http://www.ece.cmu.edu/wireless/
[14]http://www.comp.nus.edu.sg/~wuxiucha/research/reactive/

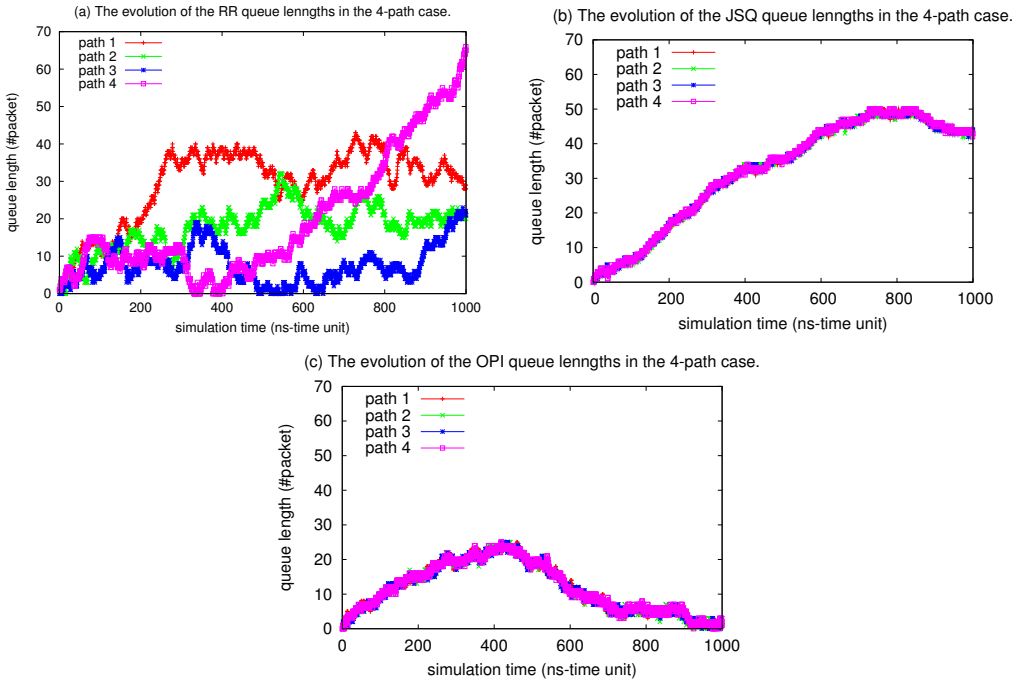



Figure 4.17: The average path queue lengths (4 paths).

Table 4.7: Average (and variance) of delay and loss rate.

| | 2 paths | | 3 paths | | 4 paths | |
|---|---|---|---|---|---|---|
| x | D(ms) | L(%) | D(ms) | L(%) | D(ms) | L(%) |
| WRR | 7.7(2.4) | 6.0(0.1) | 7.4(2.3) | 5.8(0.1) | 7.6(2.1) | 5.9(0.1) |
| JSQ | 8.3(0.9) | 6.2(0.1) | 8.1(0.7) | 6.5(0.1) | 8.2(0.7) | 6.1(0.1) |
| OPI | 7.2(0.6) | 5.9(0.1) | 6.7(0.5) | 4.7(0.1) | 5.9(0.5) | 4.2(0.1) |

As shown in Table 4.7, OPI obtains the best performance. Compared to JSQ and WRR, OPI achieves smaller average end-to-end delay and loss rate in all considered scenarios. A good explanation for these results is in Fig. 4.17, which illustrates the evolution of the path queue lengths of the three algorithms obtained with 4 paths. As shown, the fluctuations of the WRR queues indicate the dynamics of the wireless channels, which reflect both congestion and transmission errors. In comparison with WRR, JSQ manages to maintain a stable and equal queue length for all paths. However, with this algorithm the queue lengths are large, which subsequently yields the high average end-to-end delay and loss rate. As for the evolution of the OPI queues, Fig. 4.17 clearly shows the online policy improvement process. In the warm-up period (approximately first 500KB

of data), the evolution of the OPI queues is almost the same as the evolution of the JSQ queues since the two policies are, in fact, the same. However, after OPI obtained an optimal policy, its queue lengths start to decrease. As a result, OPI gives the lowest average end-to-end delay and the lowest delay variance.

Besides the end-to-end delay characteristics, the simulation results also indicate that the OPI algorithm improves end-to-end loss characteristics. Fig. 4.18 shows the distribution of consecutive losses obtained with OPI, WRR and JSQ. As shown, the probability of suffering from 2 consecutive losses is 0.07 for OPI, while this probability is subsequently 0.12 and 0.20 for JSQ and WRR. The probability of having more than 2 consecutive losses is 0.01 for OPI, 0.07 for JSQ and 0.10 for WRR. Although our approach is built to pursue the improvement of delay and loss characteristics, it is also interesting to look at the behavior of another relevant QoS parameter, that is the throughput. Fig. 4.19 shows the average throughput obtained by the algorithms using 2 paths with the average data rate of 2Mbps. We can see the impact of the fading effect on the JSQ algorithm, which gives a low and fluctuating throughput. Due to the fading effect, throughput of the wireless channels changes rapidly. As a result, greedy strategies like JSQ suffer heavily from these rapid variations. On the other hand, the OPI algorithm, which adapts to the network conditions, obtains a significantly higher throughput. The WRR algorithm, which distributes packets equally on all paths, also obtains a good throughput. However, it suffers more losses and high end-to-end delay variations since it does not take the channels dynamics into account. Finally, we can conclude that OPI reaches the best performance for all QoS parameters.

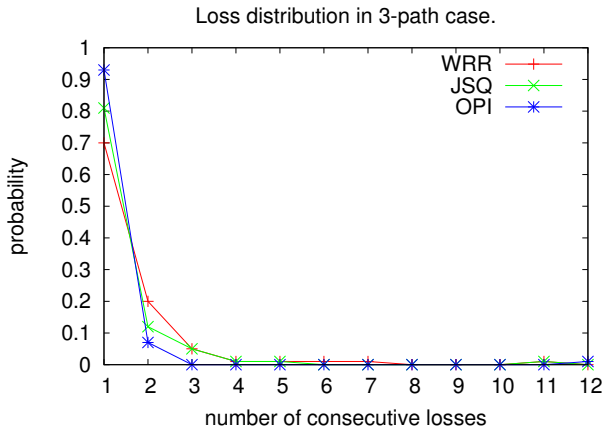



Figure 4.18: Probability of consecutive losses.

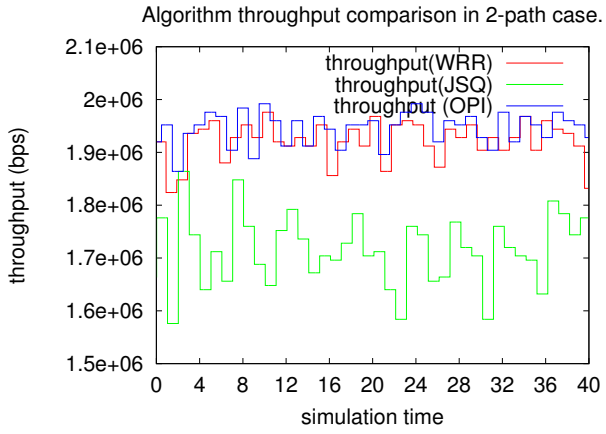



Figure 4.19: Throughput computed in 1s interval (2 paths).

### Load share analysis

In the previous simulations, we observed that in the long-term, OPI achieved a load share that was inversely proportional to the mean delay of the path. In particular, if the mean delay of each overlay path is respectively denoted as $\widehat{d}_1 \ldots \widehat{d}_M$ for $M (M > 1)$ overlay paths, then the portion of load share $l_i$ of path $i$ obtained in the long term by OPI is computed as follows:

$$l_i = \frac{\prod_{j=1, j\neq i}^{M} \widehat{d}_j}{\sum_{i=1}^{M} \prod_{j=1, j\neq i}^{M} \widehat{d}_j}$$

This observation indicates that OPI does maintain some level of fairness in the long term. However, it does not mean that one can "simulate" OPI by distributing data proportionally to the mean delay since (i) OPI does not maintain this fairness in the short-term; (ii) the mean delay is a function of the data distribution policy.

## 4.4.5 Further investigations

In order to compare OPI with WRR and JSQ, in previous sections we have shown the performance of such algorithms in network settings that allow to obtain the highest performance gain [143]. In the following sections we investigate what happens in different settings, when an higher number of nodes and paths are used, when the paths are not independent, and when the path states are not accurately estimated.

### Impact of the number of nodes

We investigate the impact of the number of nodes composing the paths in both wireless and wired scenarios using the topology shown in Fig. 4.20. The dotted lines in such topology represent a varying number of nodes on the three paths. This is to test the effectiveness of OPI on paths comprising up to 10 nodes. Cross traffic is generated between all the hosts in the top and bottom of the topology with parameters reported in Tab. 4.3. In Fig. 4.21 we report the results obtained with 10 nodes per path. As we can see, OPI outperforms the other distribution algorithms in all the cases, allowing to obtain lower transfer times also in dense overlay networks with a high number of nodes and comprising wireless links.

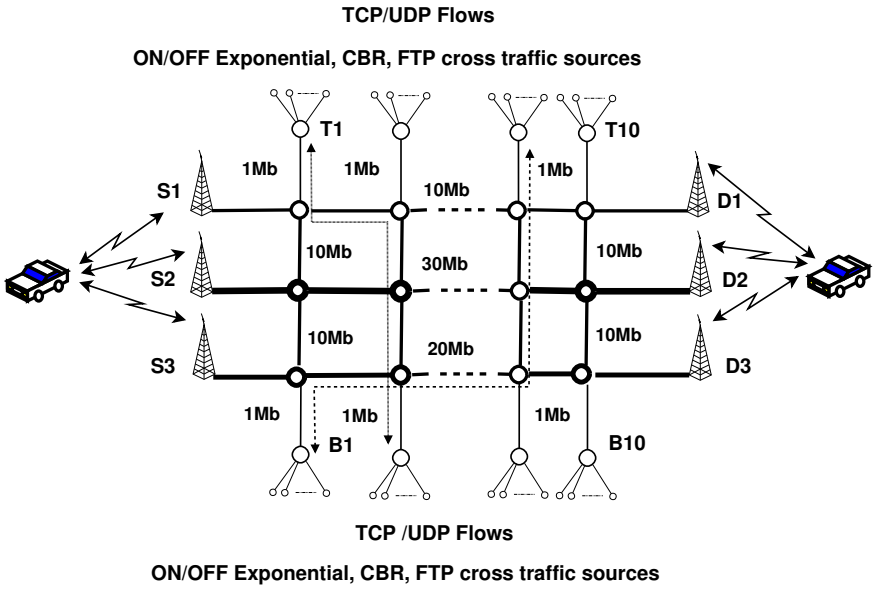



Figure 4.20: Heterogeneous network topology.

### Impact of the number of paths

We considered an amount of 10Mb of data to be transferred at a fixed rate of 384Kbps over 1 to 10 homogeneous paths using the OPI algorithm. The capacity of each path is 0.1Mbps, and the propagation delay is 50ms. For comparison purposes, the transfer time obtained in each case (depicted in Fig. 4.22) was normalized over the transfer time of the 1-path case.

As shown, from 50% to 85% of the performance gain is realized with 2 to 5 overlay paths. A further increase of the number of paths slightly increases the performance gain.

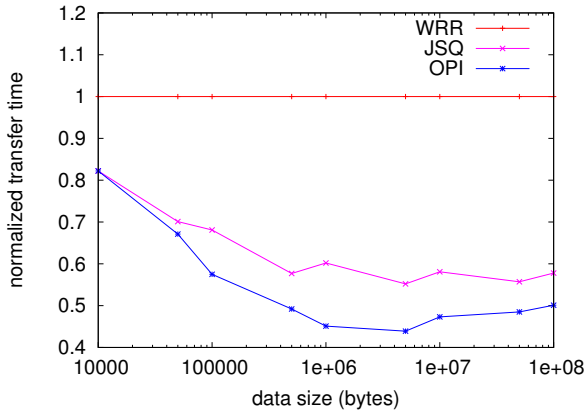



Figure 4.21: Transfer time over paths composed of 10 nodes.

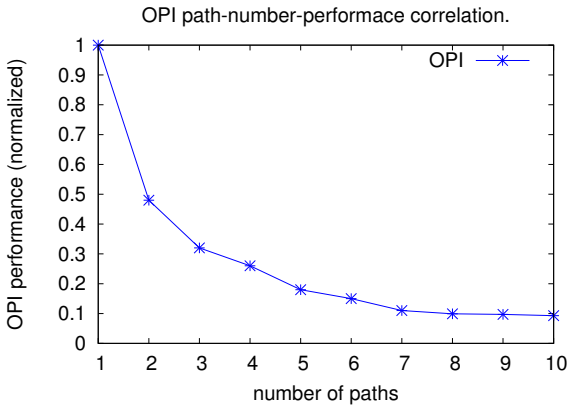



Figure 4.22: Transfer time vs. number of paths.

This result supports the statement of [143]. In addition, in real implementations a large number of paths implies a high overhead.

### Impact of the path diversity

Considering the topology reported in Fig. 4.13, an amount of 10Mb of data is transferred from Canberra to Adelaide using 3 overlay paths, which subsequently were Canberra-Adelaide direct route, Canberra-Adelaide via Sydney, and Canberra-Adelaide via Hobart. These 3 paths share the common Melbourne-Adelaide link, the capacity of which is 2Mbps. We repeated the transfer with different source rates, increasing from $0.5 - 4$Mbps to gradually saturate the shared link. The obtained performance of each algorithm was then compared with its performance when transferring data at the same rate, over 3

independent paths with an equivalent capacity and propagation delay. The result is illustrated in Fig. 4.23, which shows the percentage of performance loss due to the impact of the path dependence.

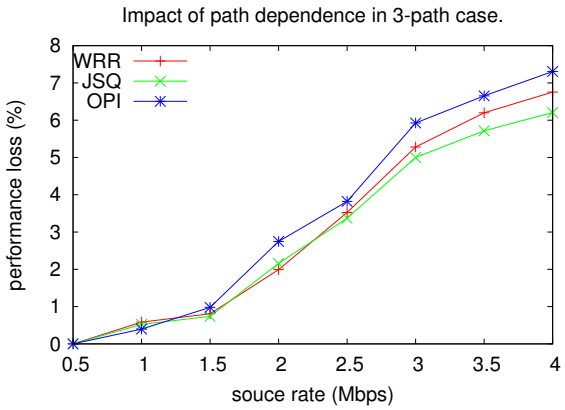



Figure 4.23: Impact of the path diversity.

As shown in Fig. 4.23 the dependence of the overlay paths has a negative impact on performance of all the three algorithms. However, when the source rate was over 2Mbps, OPI suffered more since the assumption on the independence of the overlay paths was heavily violated. In average, the performance loss due to this phenomenon is approximately 5%. Nonetheless, the loss only becomes significant when the shared link is saturated, i.e. the source rate is equal to the shared link throughput (approximately 1.5Mbps in this case). This indicates that overlay paths can be considered independent if they share non-saturated links. In this situation, the performance loss is less than 1%.

### OPI Sensitivity and Robustness

Looking at the formulas used to compute $\beta_j$ ((4.13) and (4.14) given in Sec. 4.4.2), we can see that the performance of OPI is only affected by the $\rho/\lambda$ ratio, but not the absolute values of $\rho$ and $\lambda$. Thus, as long as the ratio is maintained, the OPI performance is guaranteed. Moreover, OPI can resist to a certain level of error in the $\rho/\lambda$ estimation. Fig. 4.24 shows the values of $\beta_j$ computed using different $\rho/\lambda$ ratios. As shown, $\beta_j$ values computed with $\rho/\lambda$ ranging from $0.5 - 1.0$ are relatively close, which consequently would give a similar performance.

To increase the performance of the algorithm, the $\rho/\lambda$ ratio must be estimated as close to the actual value as possible. Since the $\rho/\lambda$ ratio is an average value, a long enough

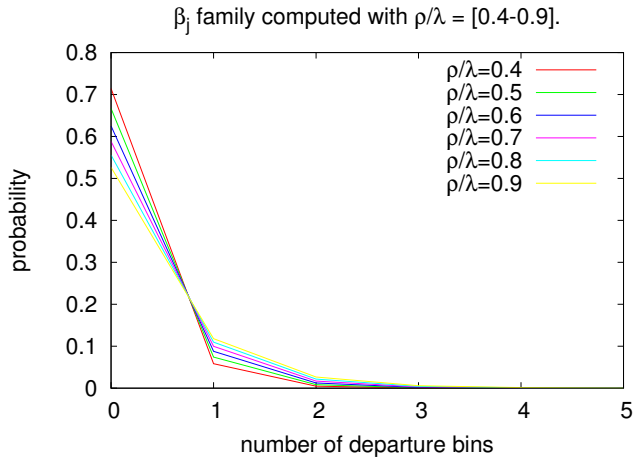



Figure 4.24: $\beta_j$ family.

history would give a better estimation. However, a trade-off between a long history and the dynamics of the overlay path should be found. In this study, the $\rho/\lambda$ ratio is re-estimated after every 10MB of data transferred. However, if the stationary period of an overlay path is known, a better $\rho/\lambda$ estimation can be obtained.

### 4.4.6 Discussion and conclusion

In this section we have presented a technique for informed path diversity. The design of the technique is based on a model of the network paths, using queueing theory. Each path is represented by a queue, whose state is monitored by means of a path-state monitoring mechanism. Thanks to this information is possible to chose, for each packet, the most appropriate path for the transmission, in order to minimize or maximize an objective function.

The knowledge of the status of the paths, however, is not sufficient to achieve the benefits, because the queues change state after that the packet has been transmitted. To cope with this situation, we used the framework of Markov Decision Processes, which take into account the effect of the decisions made, by means of a reward function. This allows to progressively achieve a scheduling policy that guarantees the required benefits. To solve the Markov Decision Process on the fly, we proposed a technique called Online Policy Iteration (OPI). OPI starts sending packets as the JSQ does. In the meanwhile, it *learns* the consequences of these actions, and improves the scheduling policy progressively, until the optimum has been reached. Thanks to this behaviour, it is possible to obtain better

performance than the weighted round robin and join the shortest queue. We applied this technique to different networking scenarios, including wireless and overlay networks. In each of them, a different objective has been pursues (e.g. minimal file transfer time and minimal video distortion). This allows to understand the possible applications of our proposal.

The results of the *ns2* simulations showed that OPI is actually able to improve the performance of the applications. It is worth noting, however, that this technique requires information about the status of the network with a very high frequency: the path state monitoring mechanism has to provide a result every time a packet has been sent on a path. This poses issues related to the overhead caused by the measurements. The simulations have also shown that the benefits remain almost unchanged if we lower the path switching time. For this reason, with the aim of deploying path diversity in every IP network, we designed a technique that schedules the packets on the available paths with less frequency. This *path switching* technique is explained in the following section.

## 4.5 Informed path switching

In this section we describe our activities aimed at developing a technique and a tool for informed path diversity. Thanks to the experimentations of the previous section, we understood that informed diversity techniques can actually provide better performance than uninformed ones. We therefore move another step in this direction with the informed path switching. The tool operates periodically switching the path over which to send the packets. An active measurement technique is used to infer the status of the available paths, and a ranking function allows to choose the best path for every period.

The rest of the section is structured as follows. In Section 4.5.1 we describe the design and implementation of the tool, called PathD. Section 4.5.2 presents the testbed, constituted of both virtual and real hosts, on which we performed our testing and experimentation activities. We discuss obtained results in Section 4.5.2, and provide concluding remarks in Section 4.5.4.

### 4.5.1 PathD: design and implementation

PathD, as shown in Fig. 4.25, is designed to periodically execute some measurements - directed to a specific destination - to obtain QoS parameters associated to the paths

available towards this destination. These parameters are then used to select the best outgoing interface for that time interval.

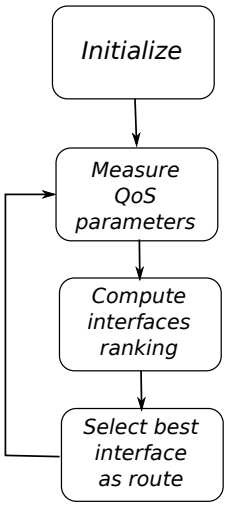



Figure 4.25: PathD algorithm overview.

To estimate the QoS parameters on the available paths, the application performs measurements using the D-ITG [17]. *ITGSend* and *ITGRecv* components are respectively executed on source and destination hosts, in order to generate packet flows on each path. *ITGLog* is executed on the source host and stores all the parameters related to the generation into a binary log file. Finally, *ITGDec* is used to elaborate the log file to extract the QoS parameters.

On the source host, *pathD* forks into two processes. The child process, using the system() function, runs *ITGLog*, which operates as a daemon waiting to receive and write on disk the log-file information. The parent process, instead, runs in sequence *ITGSend* and *ITGDec*, to generate the traffic on the available paths, and to analyze the results. In order to perform this task it executes the following operations:

- using *ioctl()* calls on */proc/net/dev* and */proc/net/if_inet6* files, it retrieves the ip address of active interfaces;
- it obtains the gateways associated to that interfaces using *ioctl()* calls on the */proc/net/route* file[15];

[15]Default routes are previously configured for each interface

- the previous data is stored into an array of structures, defined as shown in Fig. 4.26, containing the interface name, its ip address, the configured gateway and the rank compute in next steps;
- for each element of the array, *ITGSend* is executed using the options shown in Fig. 4.27;
- *ITGDec* is executed to extract QoS parameters (delay, jitter, bitrate and packet loss) from log files;
- the previous parameters are used to compute the rank of each interface;
- the interface having the better ranking value is then configured as the default route to reach the destination.

```
struct info {
    char if_name[15];
    char ip_addr[20];
    char gateway[20];
    float rank;
}
```

Figure 4.26: Structure used to store interfaces information.

```
ITGSend -a <dst_ip> -sp 9400 -rp 9500 -i <if_name> -C 1 -t 60000 -l send_log_file
        -L <src_ip> UDP -x recv_<timestamp>_<if_name> -X <src_ip> UDP
```

Figure 4.27: Options used to launch *ITGSend*.

The options passed to *ITGSend* are described in the following:

- **-a <dst_ip>**: specifies the destination ip address as passed to PathD on the command-line;
- **-sp 9400**: sets the transport layer source port number to 9400;
- **-rp 9500**: sets the transport layer destination port number to 9500;
- **-i <if_name>**: specifies the name of the interface to be used for generating traffic, as taken from the info.if_name variable of array elements;
- **-C 1**: sets the inter departure time to one packet per second;

- **-t 60000**: sets the duration of the generation to 1 minute;
- **-l send_log_file**: enables remote logging on the sender side, setting the log-file name;
- **-L <src_ip> UDP**: specifies the IP address of the log server and the transport protocol to be used for sender-side remote logging
- **-x recv_<timestamp>_<if_name>**: enables remote logging on the receiver side, setting the log-file names by using the timestamp associated to the measurement and the outgoing interface name.
- **-X <src_ip> UDP**: specifies the IP address of the log server and the transport protocol to be used for receiver-side remote logging;

For each interface the ranking value is calculated using the following formula:

$$R = \frac{\alpha\,(b - b_m) + 1}{\beta\,(d - d_m) + \gamma\,(j - j_m) + \delta\,(p - p_m) + 1} \tag{4.19}$$

where:

- $b$, $d$, $j$ and $p$ respectively represent the resulting bitrate, delay, jitter and packet loss;
- $b_m$, $d_m$, $j_m$ and $p_m$ respectively represent the minimum bitrate, delay, jitter and packet loss measured;
- $\alpha$, $\beta$, $\gamma$ and $\delta$ are weights $\in [0, 1]$ used to modify the importance of each QoS parameter in the rank.

The minimum values were introduced to allow ranking formula to work correctly also if source and destination clocks are not synchronized.

Once the best interface has been detected, to force all the packets directed to the destination to go through it, the application adds a new routing table entry[16], using the *ioctl()* call. PathD executes all the previous steps in a loop, waiting by default 120 seconds between each repetition. The routing table entry is maintained unmodified when the result given by the ranking procedure is unchanged. Otherwise, it is modified pointing

[16] A 255.255.255.255 netmask is associated to the entry

to the new best interface. Finally, when *PathD* is terminated, it automatically removes the entry.

### 4.5.2 Testbed and tools

In order to perform some experiments on informed path switching using PathD, we configured two similar small-scale testbeds respectively using a virtual and a real environment, as described in the following sections.

#### Virtual testbed

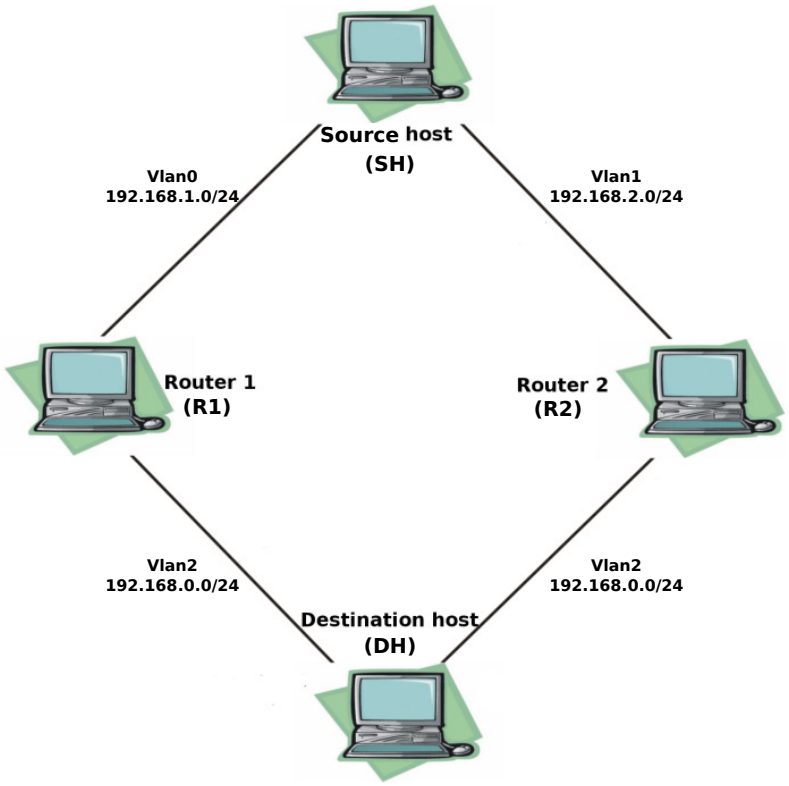



Figure 4.28: Virtual testbed setup.

To test the proper operation of PathD, we performed some preliminary experiments using a virtual environment. As depicted in Fig. 4.28, four virtual machines were configured using QEMU[17]: a generic and open source machine emulator and virtualizer. QEMU can be used as a machine emulator, running OSes and programs made for one machine (e.g.

[17]http://www.qemu.org/

an ARM board) on a different machine (e.g. your own PC), or as a virtualizer, achieving near native performances by executing the guest code directly on the host CPU. In the first case, it uses a dynamic translation technique to achieve better performances, while in the second one, it make use of a host driver called the QEMU accelerator (also known as KQEMU), and requires that both the host and guest machine use x86 compatible processors.

Table 4.8: Virtual machines network interfaces configuration.

| | **eth0** | **eth1** |
|---|---|---|
| **SH** | 192.168.1.1 | 192.168.2.1 |
| **R1** | 192.168.1.2 | 192.168.0.1 |
| **R2** | 192.168.2.2 | 192.168.0.4 |
| **DH** | 192.168.0.2 | 192.168.0.3 |

Providing operating systems and network resources virtualization, QEMU allowed us to deploy the testbed using a single physical x86 Linux workstation. All the four guest machines (SH, R1, R2 and DH) were configured using the same hardware characteristics, and using the latest Linux Debian distribution. Particularly, each virtual machine owns two 100Mbit Ethernet network interfaces, connected as shown in Fig. 4.28, and statically configured as reported in Table 4.8. Regarding routing configuration, R1 and R2 were configured to forward packets; SH default gateways for eth0 and eth1 interfaces were respectively set to 192.168.1.2 and 192.168.2.2; DH outgoing packets, using two customized routing tables, are forced to exit through eth0, if source address is 192.168.0.2, and through eth1, if source address is 192.168.0.3, and respectively routed towards 192.168.0.1 and 192.168.0.4.

Table 4.9: Real testbed hardware characteristics.

| | |
|---|---|
| CPU | Pentium 4 3.4 GHz |
| RAM | 2 GB DDR PC3200 |
| Disk | 300 MB Serial ATA |
| Network devices | Realtek 8139 Fast-Ethernet Embedded<br>Davicom Fast-Ethernet 10/100 PCI<br>Davicom Fast-Ethernet 10/100 PCI |

In order to emulate real network behaviors, such as variable packet delay, loss, duplication and re-ordering, we exploited the Netem component, available on Linux kernel releases since version 2.4.28. In the user-space, Netem is controlled by the command line tool *tc* (Traffic Control), which is part of the *iproute2* package. Netem is specifically designed for testing protocols by emulating the properties of wide area networks.

### Real testbed

The previous testbed was mainly setup to verify PathD proper operation. To investigate on the achievable performance, we setup a similar testbed using four real workstations, whose characteristics are reported in Table 4.9. Referring again to Fig. 4.28, the workstations are equipped with the following Linux distributions:

- **SH**: Debian Linux release 5.0
- **R1**: CentOs Enterprise-class Linux Distribution release 4.4
- **R2**: Fedora Linux release 10
- **DH**: Debian Linux release 5.0

All the hosts were connected to the Internet using the embedded network device, thus the other devices were used to build the private networks needed for the experiments. Finally, regarding routing configurations and software deployment, the real testbed was configured in the same way of the virtual one.

## 4.5.3 Experiments and results

### Virtual testbed

In this experiment, we used Netem to force a variable delay on packets going from the source to the destination. In practice, we applied such mechanism on the outgoing interfaces of the SH machine, changing delay values on both interfaces every 5 minutes, following the schema reported in table 4.10.

Table 4.10: Delay variations on SH interfaces.

| **Interval [min]** | $0-5$ | $5-10$ | $10-15$ | $15-20$ | $20-25$ | $25-30$ |
|---|---|---|---|---|---|---|
| **eth0 delay [ms]** | 50 | 100 | 150 | 200 | 150 | 100 |
| **eth1 delay [ms]** | 200 | 150 | 100 | 50 | 100 | 150 |

Once configured the testbed, the experiments were conducted as follows:

- run *ITGRecv* on the DH machine;
- capture generated traffic on both R1 and R2 nodes to monitor the traffic effectively flowing on the different paths;

- run PathD on the SH machine using 192.168.0.2 as destination.

Inside PathD, *ITGSend* is configured to generate UDP packets towards the destination at a rate of 2*pps*, carrying a 512 bytes payload. Since in these experiments we only introduced variable delays, in the ranking formula $\beta$ was set to 1 and all the other weights were set to 0. The measurement cycle was composed of the following steps:

- 30 seconds of measurements using eth0 as outgoing interface;
- 30 seconds of measurements using eth1 as outgoing interface;
- interfaces ranking computation and best interface selection;
- 120 seconds of inactivity.

Thus, the overall monitoring cycle lasts about 180 seconds, and it is continuously repeated for the duration of the experiment.

Working in the virtual scenario, we executed several times the experiment described before, in which QoS parameters were measured 10 times (each 180 minutes) during a period of 30 minutes. Obtained results are shown in Fig. 4.29a and 4.29b, in which the delay and jitter values are respectively reported.

Observed values mostly depend on three factors:

- the actual channel state while measurement is conducted;
- the additional delay imposed by Netem;
- the unpredictable delay due to virtual machine scheduling procedures.

Since the virtual environment was completely controlled and there was no cross-traffic, the first factor is mostly negligible. Moreover, depending of the ranking function definition and on the weights choice, on each measurement interval the best interface always obtains a rank of 1.

Looking at Fig. 4.29a we can notice that the delay values mostly follow the ones imposed by Netem on both paths, in every 5-minutes interval. At the beginning, the eth0 interface is selected as default route to reach the destination, because it obtains a higher rank with respect to eth1 interface. After 10 minutes, eth1 obtains a higher rank, and it is then preferred to eth0. 5 minutes before the end of the experiment, eth0 becomes again

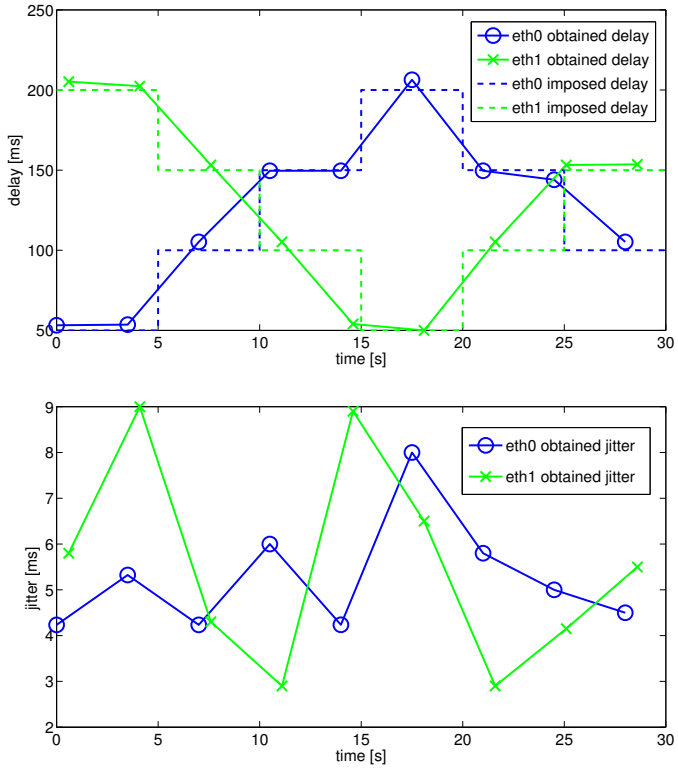



Figure 4.29: Delay (a) and jitter (b) measured on available paths.

the best choice. It can also be noticed that some measurements (e.g. the 5th on eth1 and the 9th on eth0) were conducted across a sudden delay variation, thus obtaining a value in-between the ones in the adjacent intervals.

Looking at figure 4.29b, it can be noticed that the most relevant jitter variations (e.g. the 2nd, the 3rd and the 5th values measured) occur when Netem imposes the highest delay difference between the two paths.

### Real testbed

In the following, we present the outcome of two sets of experiments performed configuring the testbed in Fig. 4.28 to use the two paths (composed of two links) between SH and DH. For such tests, we introduced variable conditions on the two links connecting *SH* with *DH* by using *Netem*. Two different kinds of experiments were then performed. In the first one, we instructed TimeD to seek for lower delay, while we introduced variable delays on the links. In the second one, TimeD was instructed to seek for lower losses, while we introduced variable losses on the links. Variable conditions on the two links

were created by means of a *bash* script, that operated in a cyclic way. The following operations were performed during each cycle: selection of a random number as reported in Table 4.11; setup of the random delay or loss rate, according to the random number previously selected; waiting of a time period of 120 s. This allowed to emulate time-varying network conditions.

Table 4.11: Network conditions imposed on the testbed.

| | |
|---|---|
| Delay | 20 → 200ms |
| Loss rate | 0.01 → 0.1ms |

In these conditions, we then used D-ITG to emulate the communication between two users located at *SH* and *DH*. Such two users were communicating by using UDP at different rates, ranging from 10 to 1000 pps. For each packet rate, we performed two different tests. In the first one, PathD was used to dynamically select the best path for the communication. In the second test, instead, a simple weighted round robin (WRR) was used for the path selection, i.e. the packets were transmitted on the two available paths in a cyclic way.

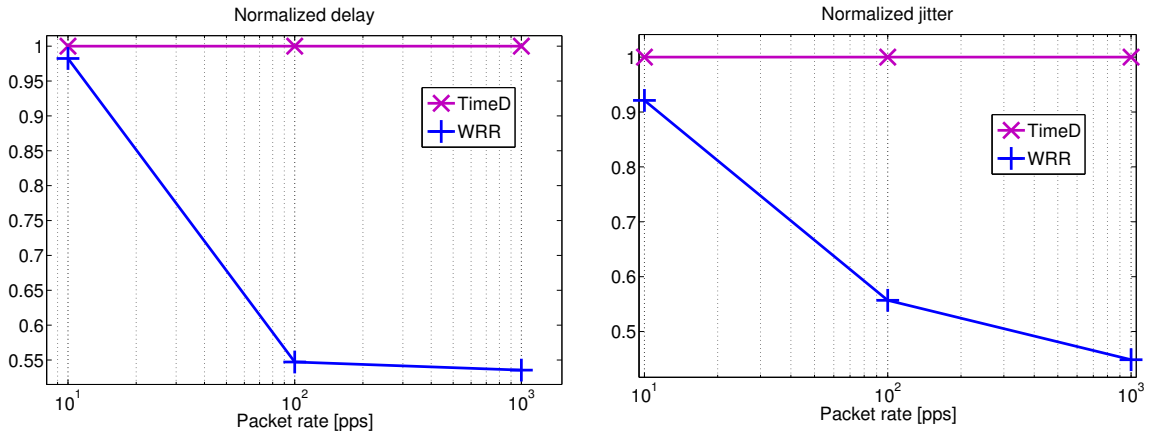



Figure 4.30: Performance obtained with TimeD and WRR with variable delay.

In Fig. 4.30 we show the results in terms of average delay and jitter obtained in the first set of experiments, i.e. when introducing variable delays on the links. We decided to show these two parameters because they are most impacted in this scenario, which means that we expect that the throughput and the loss do not change significantly. The results are reported in terms of relative values with respect to those achieved with WRR. For

this reason the values related to WRR are always equal to 1. As shown in this figure, PathD is able to provide better performance with respect to WRR. The performance improvements are higher at higher packet rate, because the more the path is used by the communication, the more it can benefit from better performance. We also observe that PathD is able to provide average delay and jitter values that are about the half of those achieved with WRR, when the packet rate is equal to 1000 pps.

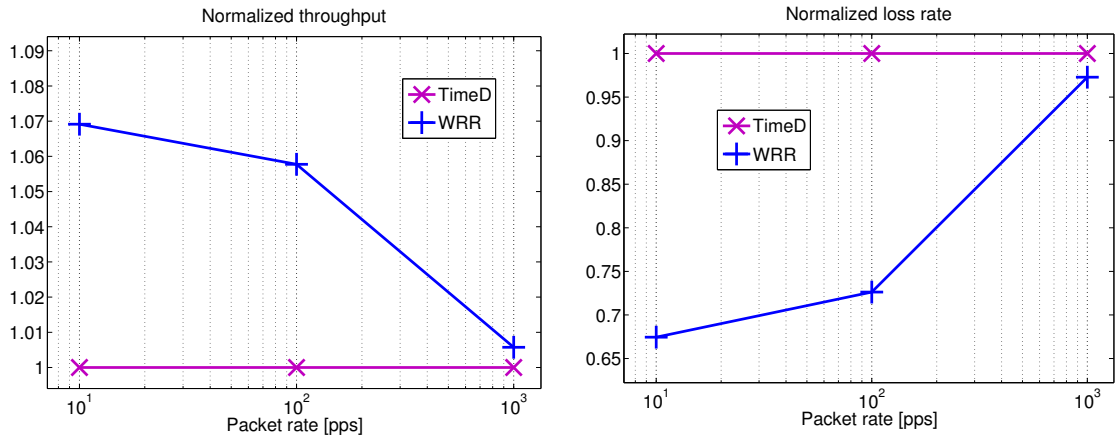



Figure 4.31: Performance obtained with TimeD and WRR with variable loss rate.

In Fig. 4.31 we show the results in terms of average throughput and loss rate obtained in the second set of experiments, i.e. when introducing random losses on the links. Also in this case, the results are normalized with respect to those achieved by WRR. PathD is able to provide better performance also in these conditions, both in terms of loss and throughput. However, here we observe that the improvements decrease with the increase of the packet rate.

### 4.5.4 Discussion and conclusion

In this section we showed how it is possible to achieve the benefits of informed path diversity in IP networks. We presented the design and implementation of a tool that performs active measurements on the available paths and chooses the best one according to a ranking function. We discussed our design choices, and the main issues we had to face during the implementation phase. We then described the testing we performed in an emulated environment setup, using a state-of-the-art computer virtualization software called QEMU. The testing results allowed to verify that the our tool behaves as expected. Af-

terwards, we performed an experimental measurement campaign using real machines. We configured a network scenario comprising different paths connecting two hosts. We setup different network conditions on the two paths, and compared the achievable benefits with a non-informed path diversity technique, using weighted round robin packet scheduling. The results we obtained allowed to verify that our approach achieves higher performance with respect to WRR.

## 4.6 Final remarks

In this chapter, we aimed at showing that informed path diversity techniques allow to effectively exploit the availability of multiple paths between a source and a destination. To achieve this task, we adopted a methodology comprising different steps.

In details, we firstly approached the problem developing a simulation environment in Matlab. The simplicity of this simulator allowed us to concentrate on specific aspects of this problem, and to investigate on the potential benefits of path diversity. In this simulator, we implemented models reported in literature for the loss behaviour on the network. We evaluated several statistics of the received packet flows, considering also the distortion caused on a video stream by the loss of packets. Afterwards, we proposed a technique for informed path diversity based on Markov Decision Processes. With this activity we moved a first step towards using informed path diversity in real scenarios. We performed a deep analysis of the performance of our technique, called OPI, with *ns2*. A comparison with weighted round robin and join the shortest queue showed that OPI is able to provide higher benefits than state-of-the-art techniques. Finally, we designed and implemented a tool to deploy informed path diversity in real networks. We described the issues we had to face for this activity, and how we managed to develop the tool called TimeD. We performed the testing of the tool in a virtualized environment. Then, we conducted a set of measurements using real hosts. Again, we compared the performance achievable with TimeD with that achievable using weighted round robin. The results obtained in these measurements show that our technique is actually able to provide improved performance with respect to WRR.

We believe that this research paves the way for the use of path diversity in IP networks.

# Chapter 5

# Introducing time diversity at IP layer

In this chapter we present a simulation and experimental analysis of packet-level interleaving aimed at understanding if and how such technique can be deployed in real networks, exploiting information about the status of the network.

## 5.1 Introduction

Packet-layer time diversity is also called packet interleaving because, as explained in Section 2.1.2, it is realized by altering the reciprocal order of packets. Thanks to this simple technique, it is possible for the packets to benefit from different network conditions and therefore experiment uncorrelated dynamics. However, how to obtain this benefit is still to be understood. While interleaving has been successfully applied at bit or symbol level, indeed, not much has been said about interleaving at packet level. In Section 2.3.3 we have analyzed the works presented in literature, evidencing how the few ones that do not apply interleaving at bit/symbol level are typically tight to a specific protocol or application. In this thesis, instead, we pursue the aim of using this technique at IP layer, independently on the particular application and protocol used on top of it.

In the next sections, we firstly discuss the definitions and main assumptions at the base of our work. Then, we present a simulation framework we developed in order to both understand the potential benefit of packet-level time diversity, and discover the configuration most suited for any channel condition. Afterwards, we present a prototype application we designed, developed, and publicly released, that allows to experiment with time diversity on real networks. Possible application scenarios are discussed together with

design and implementation issues. Thanks to such application, we perform an experimental analysis of the performance of time diversity with real traffic. Obtained results show how it is possible to deploy such technique on real networks. However, they also evidence that some important issues have to be addresses, such as operative parameters that have to be configured, and performance problems when using transport protocols with congestion control algorithms. We then show how, thanks to the addition of measurement capabilities to TimeD, it is possible to overcome those issues, deploying informed time diversity in real scenarios.

## 5.2 Definitions and basic assumptions

In this thesis we use the following definitions.

- **Interleaving policy**. It governs the way the packet order is altered. For the motivations reported in Section 2.4.2, the policy used in this thesis is the *block interleaving*.
- **Block interleaving**. It is an interleaving policy performed by inserting the packets in a matrix (the block) by row and outputting them by columns. This policy is explained in details in Section 2.4.2.
- **Bursty losses**. They are losses involving a number of consecutive packets.
- **Independent or uncorrelated losses**. They are losses generally not involving consecutive packets. If a network is affected by uncorrelated losses, then the loss process, observed as a time series, does not present correlation. Note here that, in general, an independent series of random variables is uncorrelated, but an uncorrelated series of random variables is not necessarily independent. Correlation refers only to linear dependence. However, in the following we will use the terms uncorrelated and independent interchangeably, as we do not consider non-linear dependences.
- **Loss pattern**. It is a loss process characterized by means of a 2-state Markov chain with parameters $\pi_b$ and $\rho$. More information about the 2-state Markov chains to model packet loss are reported in Section 5.3.1.

- **Loss decorrelation power**. It is the ability to transform bursty losses into independent losses.
- **Interleaver**. It is a module, implemented as a software or hardware component, that performs the operation required by the interleaving policy.
- **Deinterleaver**. It is a module, implemented as a software or hardware component, that restores the original order of packets. It basically performs a dual operation with respect to the interleaver.
- **Packet level**. It is the network level according to the ISO/OSI stack reported in Fig. 2.1.

As for the assumptions, in this analysis we mainly consider that the dynamics on the network are well modeled by a 2-state Markov chain. In particular, this model will be used for the loss process, which is the main cause of impairment we want to counter. The model is described in details in Section 5.3.1. In the simulations we also assume that the network traffic we want to interleave on has a constant bitrate. We are aware of the fact that this is not true for many real Internet applications. However, two main reasons are at the base of this choice: i) for some multimedia communication applications, the generated traffic has indeed a constant bitrate (e.g. VoIP flows without silence suppression); ii) the simulations are aimed at understanding the potential benefits of packet interleaving, and the relation between interleaving depth and loss decorrelation. In that respect, the assumption of constant bitrate traffic does not bias our results.

## 5.3 Understanding the benefits of packet interleaving

In this section we report our simulation analysis of packet interleaving. We describe the simulation environment we setup in Matlab, the models we used for the loss process and the network behaviour, and the results we obtained. This activity provides a first contribution towards informed time diversity at IP layer, because it allows to understand the interleaver parameters to be used for a given network condition. As we will see in Section 5.4, this information has been exploited for our tool implementing informed time diversity at IP layer.

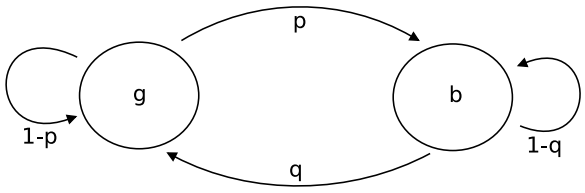



Figure 5.1: 2-state Markov chain.

### 5.3.1 Simulation environment

#### Loss model

As introduced in Section4.3.1, a typical characterization of the loss process over the Internet is provided by the continuous-time Gilbert-Elliott model [74, 75], which is also referred as 2-state continuous-time Markov chain (2-MC) [124, 40]. Such model is able to capture the potential correlation between consecutive losses, which is present on the Internet [25]. Due to FIFO queuing discipline used by routers, it is very common to discard a number of consecutive packets from the queue when a congestion event is detected. Moreover, routing decisions are taken at time intervals that are usually longer than flow durations. This implies that when a congestion happens, different packets from the same flow are discarded. On the other hand, such model has also proven to be able to capture the loss dynamics in wireless networks [76]. In these scenarios, once the wireless channel turns bad, because of attenuation, fading, scattering, or interference from other active sources, the errors happen in bursts and becomes dependent on each other. It is also worth noting that 2-MC is not the only neither the fittest model for Internet traffic dynamics. It has actually been proved that Markov chains with more states are able to obtain higher modeling accuracy in some cases [25]. However, 2-MC represents the best compromise between complexity and accuracy.

Let $X = \{X(t) : t \geq 0\}$ be the aleatory process of losses following the Gilbert model. The state $X$ at time $t$ can assume one of the following values: $b$ or $g$ ($b$ = "bad" and $g$ = "good"). Process $X(t)$ at a fixed time is characterized by the parameters $\mu_g$ and $\mu_b$, which can be though as the rates at which the chain passes from state $g$ to state $b$ and vice versa. A diagram of the 2-state Markov chain is reported in Fig. 5.1.

In general, when $X(t)$ is in state $b$, the probability to loose a packet is much larger than that in state $g$. To simplify the problem, we assume that when a packet is transmitted and channel is in state $b$, the packet is surely lost. The packet is surely received in the

opposite case. For this reason we will refer to the two state as "loss" and "no-loss", beside "bad" and "good" respectively. The steady state probabilities to receive or loose a packet are respectively equal to: $\omega_b = \frac{\mu_b}{\mu_b+\mu_g}$ and $\omega_g = \frac{\mu_g}{\mu_b+\mu_g}$. It is worth noting that we consider a packet to be lost either when it is not delivered to the destination, or when it is delivered too late, and it is therefore not useful for the application (e.g. audio samples arriving after playback time). To further simplify the problem, let us substitute the continuous-time model with a discrete-time one. Such transformation is justified by the fact that we are only interested in the behaviour of the network when packets are sent. Therefore, once we fixed the sending rate to $S = 1/\tau$, with $\tau$ being the inter-packet time[1], we can express the transition probability in the discrete case as:

$$p_{gg}(\tau) \equiv P(X_{n+1} = g|X_n = g) = \omega_g + \omega_b e^{-(\mu_g+\mu_b)\tau} \tag{5.1}$$

$$p_{gb}(\tau) \equiv P(X_{n+1} = g|X_n = b) = 1 - p_{gg}(\tau) \tag{5.2}$$

$$p_{bb}(\tau) \equiv P(X_{n+1} = b|X_n = b) = \omega_b + \omega_g e^{-(\mu_g+\mu_b)\tau} \tag{5.3}$$

$$p_{bg}(\tau) \equiv P(X_{n+1} = b|X_n = g) = 1 - p_{bb}(\tau) \tag{5.4}$$

In this way we can easily model the process of sending a flow of $n$ packets on a link using a time-discrete Markov process that evolves in $n$ steps. In particular, assuming the homogeneity of the Markov chain and choosing $\tau = 1$, the probabilistic description of such process can be obtained by using a 2-MC, see Fig. 5.1, having $p = p_{bg}(1)$, which is the probability that the next packet is lost, given that the previous is correctly received, and $q = p_{gb}(1)$, which is the probability that the next packet is correctly received, given that the previous was lost. In general, the following condition holds: $p+q \leq 1$. However, if $p + q = 1$, the model becomes a Bernoulli one, for which losses are independent and happen with an average probability $\hat{p}$.

In the general case, the probability to loose $n$ consecutive packets is equal to $(1-q)^{n-1}q$, according to the geometrical distribution. While the probability to receive $n$ consecutive packets is equal to $(1-p)^{n-1}p$. Moreover, the steady-state probabilities for the two states are $\pi_b$, which is the probability of staying in state $b$, and $\pi_g$, which is the probability to stay in state $g$. They can be evaluated using the following:

[1]Note that here we are implicitly assuming constant bit rate traffic. This is because the simulations presented in the following section are actually performed with this kind of traffic. Beside this, the model derivation can be extended for other kinds of traffic.

$$\pi_b = \frac{p}{p+q}, \quad \pi_g = \frac{q}{p+q} \tag{5.5}$$

In particular, $\pi_b$ represents the average loss probability. Another important parameter, which is typically considered as the channel memory, because it represents the correlation between losses, is $\rho$. It is defined as:

$$\rho = \frac{1-q}{p} \tag{5.6}$$

Again, to obtain a Bernoulli model it is sufficient to impose $\rho = 1$, and the losses will be independent and identically distributed. This implies that a loss has the same probability to happen given that the previous state was $b$ or $g$. On the other hand, $\rho > 1$ implies that a loss is more probable if the previous state was $b$ than if it was $g$, i.e. if the previous packet was lost. For this reason, $\rho$ is considered as an indicator of the channel memory.

A last important result is related to the probability of transition from state $i$ to state $j$ in $l$ steps (with $l = 1, \ldots, n; \quad j, i \in \{b, g\}$, defined as:

$$p_{ji}(l) \equiv P(X_{n+l} = j | X_n = i) \tag{5.7}$$

Starting from the 1-step transition matrix:

$$\mathbf{Q} = \begin{pmatrix} (1-p) & p \\ q & (1-q) \end{pmatrix}$$

We have to calculate the l-step transition matrix $Q_l = Q^l$, and recall that [145]:

$$p_{bg}(l) = Q_l(1,2), \quad p_{gb}(l) = Q_l(2,1). \tag{5.8}$$

Otherwise we can use directly the following [126]:

$$p_{bg}(l) = \frac{p}{q+p}[1 - (1-q-p)^l], \tag{5.9}$$

$$p_{gb}(l) = \frac{q}{q+p}[1 - (1-q-p)^l]. \tag{5.10}$$

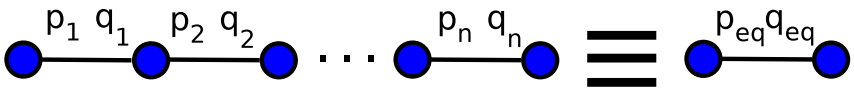



Figure 5.2: End-to-end equivalent channel.

### End-to-end path model

Let us consider a path between a source and a destination as a series of a number of links characterized by bursty losses. If every link is modeled using a 2-MC, it can be easily demonstrated [28] that the end-to-end path can still be modeled using a Gilbert model with equivalent parameters, if the various links are independent.

In fact, given a series of $N$ links of a path, modeled with 2-MC with parameters $(p_1, q_1), (p_2, q_2), \ldots, (p_N, q_N)$, it is possible to obtain an equivalent model with parameters $(p_{eq}, q_{eq})$, which represents the entire end-to-end path (or equivalent channel) as shown in Fig. 5.2.

It is worth underlining that the possibility to obtain such equivalent model is subject to the hypothesis that the parameters associated with the various links are related to the characteristics of the packet flow of interest. In other words, such parameters should be the ones experimented by our flow of interest, or by other flows with the same characteristics (packet sizes, inter-packet times, bit-rate, ...). Such hypothesis is reasonable because the parameters $(p_1, q_1), (p_2, q_2), \ldots, (p_N, q_N)$, even if specifically related to the flow of interest, take into account also the effect of other flows constituting the background traffic [25].

Under such hypothesis, the Gilbert model for our equivalent channel will be characterized by the following probabilities:

$$p_{eq} = 1 - \prod_{i=1}^{N}(1 - p_i) \tag{5.11}$$

$$q_{eq} = \frac{\prod_{i=1}^{N} \pi_{gi}}{1 - \prod_{i=1}^{N} \pi_{gi}} p_{eq} \tag{5.12}$$

where $p_{eq}$ is equal to the probability that the end-to-end path switches from state "no-loss" to state"loss", and $q_{eq}$ equal to the probability that it passes from state "loss" to state "no-loss". This can be used for modeling a generic network topology based on the description of the single links of all the possible paths between a source and a destination using equivalent channels [125]. In facts, if we have a topology in which the links are

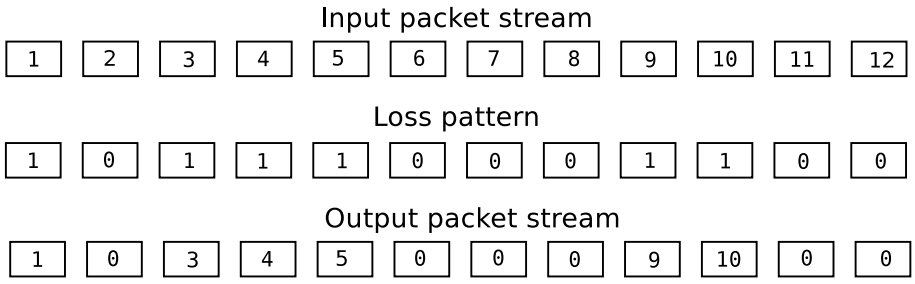



Figure 5.3: Simulation of packet loss.

described using 2-MC, it is possible to obtain an equivalent representation using (5.11) and (5.12). On the other hand, if we cannot obtain an equivalent channel model by means of such equations, it is also possible to use the parameters of the worst link for the end-to-end path [65]. In this case, however, a lower accuracy will be obtained. Finally, thanks to an on-line estimation of the behaviour of the end-to-end paths between a source and a destination, it is possible to directly obtain the couples $(p_{eq}, q_{eq})$ associated to the various paths, and to continuously update them [30].

### Running the simulations

Similarly to what we did for the analysis presented in Section 4.3, we model the packets generated by a source as an array of integers, each of which represents the sequence number of a packet. Such array is given as input to a function, called *path(I, $\rho$, $\pi_b$)*, which reproduces the behavior of the end-to-end path. The function therefore receives as input also the path characteristics in terms of $\pi_b$ and $\rho$. It then calculates a pattern of losses, using the model and the input parameters discussed above. After that, it applies the loss pattern to the input sequence in order to simulate the effect of the path traversal, substituting the sequence numbers of the packets that are lost with a 0. The output sequence will be therefore constituted by a series of positive numbers corresponding to the correctly received packets, and 0s corresponding to the lost packets (see Fig. 5.3).

When the interleaver is active, before being passed to the *path()* function, the input array is preprocessed by another function called *block_interleaver(I, l, n, m)*, which changes the order of the packets inside the input array in order to perform a block interleaving with parameters $l$, $n$, and $m$.

After the output sequence is obtained, we calculate some metrics to evaluate the effect of the interleaving. In particular, we evaluate the distribution of the length of the loss

bursts, and the distribution of the distance between two loss event (i.e. the length of the no-loss sequences). While the importance of the length of the loss bursts is immediately clear, some considerations should be made about the no-loss sequences. Once the average loss rate is fixed, the loss bursts and the no-loss sequences represent two tightly linked aspects of the same phenomenon: the more we reduce the loss-burst length, the more we bring losses closer to each other. For this reason, the no-loss sequence length may seem a redundant parameter. However, in some cases, too close loss events can cause similar effects to loss bursts, decreasing the application performance. This happens, for instance, when certain types of codecs are used for videos and the distance between losses becomes lower than a threshold [41]. In these cases no-loss sequence length plays an important role in order to understand if and how to use packet interleaving. For an increased readability of the results, in the following we report the average value of these two variables.

### 5.3.2 Interleaving policy

As introduced in Section 2.4.2, in this thesis we use a periodic interleaving policy called *block interleaving*. Periodic interleaving works on packet sequences of constant length. However, determining the optimal permutation of the sequence is not an easy task. Up to now, the only solution has been the exhaustive search, which is clearly feasible only for very small sequences. Being very time consuming, exhaustive search can be considered as a viable solution only when the time is not an issue. If we want the interleaver to operate *on-line* (i.e. on live network traffic), then we should relax the optimum permutation requisite and find a good compromise between interleaving performance and computation time. A solution to this trade-off problem is obtained using *block interleaving*.

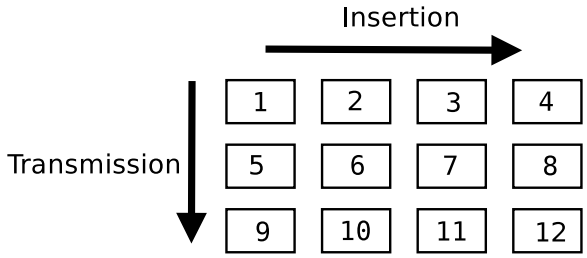



Figure 5.4: Block interleaving.

Block interleaving is made by inserting packets into a matrix by row, and picking them by column, as shown in Fig. 5.4. Basically, the flow of packets is divided in sequences of

Table 5.1: Parameters of the simulations performed.

| Parameter | Values |
|---|---|
| Interleaving block size | 48 |
| Interleaving block depth | [1, 2, 6, 12, 24] |
| $\pi_b$ $(loss)$ | [0.01, 0.03, 0.1, 0.25] |
| $\rho$ $(rho)$ | [1, 3, 8, 15, 30] |
| repetitions | 1000 |

$k$ packets. Each sequence is then placed into a *block*, a matrix of size $n \times m$, where $n$, is the number of rows, and it is called *interleaving depth*, and $m$ is the number of columns. Considering a flow of packets F $= \{P_a, P_b, P_c, ...\}$, and the example block of size $3 \times 4$ reported in Fig. 5.4, the packets will be inserted such that $P_a$ will be in position $(1, 1)$, $P_b$ will be in position $(1, 2)$, $P_c$ will be in position $(1, 3)$, $P_d$ will be in position $(1, 4)$, $P_e$ will be in position $(2, 1)$, and so on. The packets will be therefore transmitted in the following order: $P_a, P_e, P_i, P_b, P_f, P_m, ....$

### 5.3.3 Tests performed and obtained results

We performed a number of simulations with different values for all the previously cited parameters. Such values are reported in Tab. 5.1. The simulations were conducted using all the possible combinations of the parameters reported in such table. As shown, we tested different values of average loss as well as different values of loss correlation. This allows to understand the impact of each single parameter.

Let us restrict our analysis to the most important results we obtained, those most useful to assess the impact of the interleaving. The plots in Fig. 5.5 show the average value of both the loss-burst length and the no-loss sequence length for four representative combinations of $\rho$ and $\pi_b$, which are related to progressively degraded channel conditions. The plots in Fig. 5.6, on the other hand, present the same parameters but obtained using a channel with fixed $\pi_b$ (i.e. fixed loss rate) and progressively increasing loss correlation (i.e. increasing burst-loss length).

As a first consideration, the results confirm the effectiveness of the interleaving in reducing the average loss-burst length. As shown in the two left plots of Fig. 5.5 and 5.6, such parameter presents a decreasing trend when increasing the interleaving depth. We

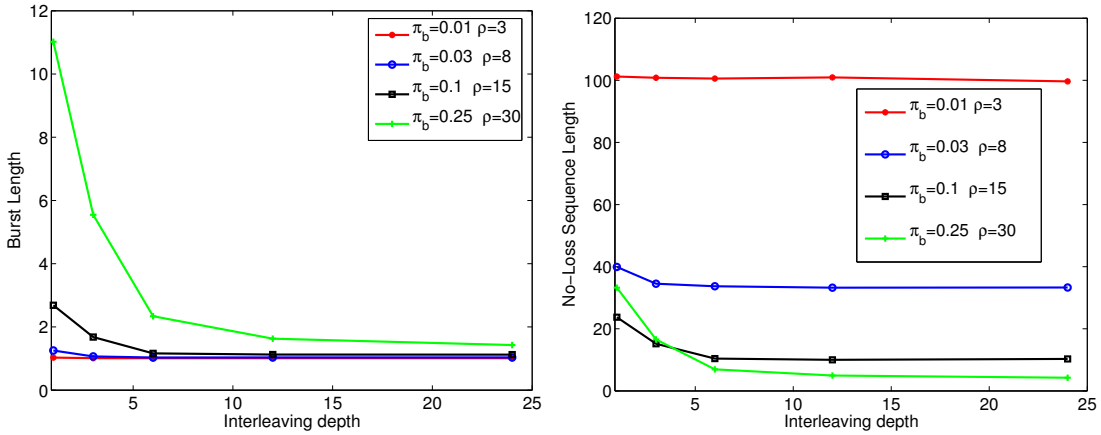



Figure 5.5: Simulation results obtained with different values of $\pi_b$ and $\rho$.

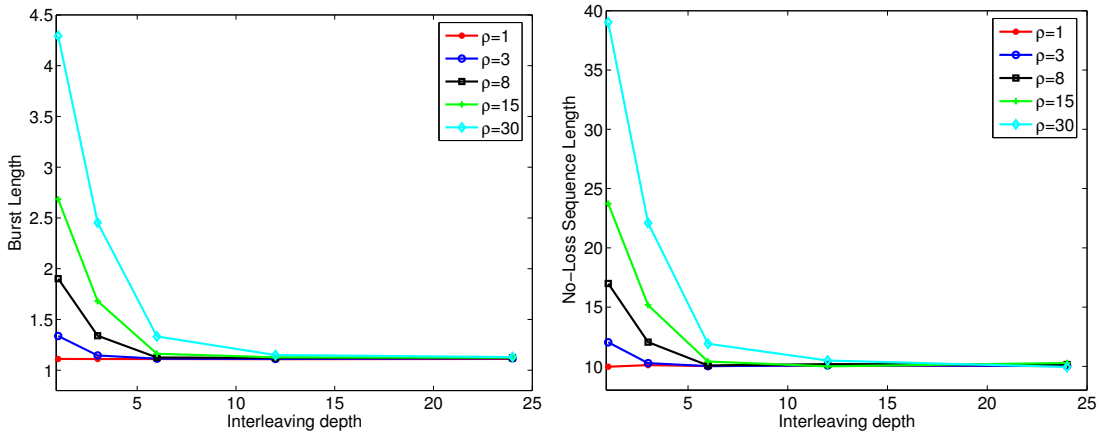



Figure 5.6: Simulation results obtained with different values of $\rho$ and $\pi_b = 0.1$.

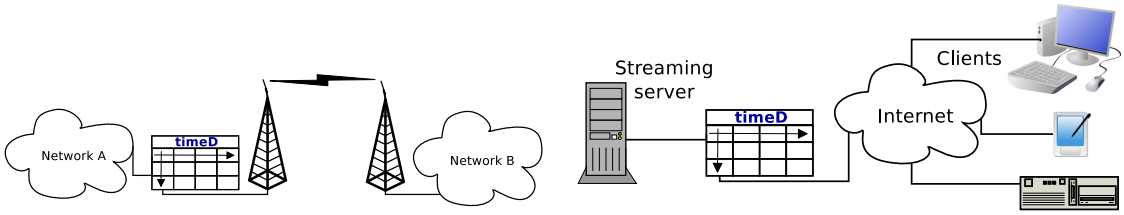
also observe that all the curves are steeper when the length of the loss-burst is high, i.e. the slope decreases when the interleaving depth increases. Which means that the more burstiness we have, the more convenient is to use the interleaving. The same behaviour can be observed comparing the curves related to the different values of $\rho$: the higher is the correlation (i.e. higher burstiness) the more effective is the interleaving.

This translates in an asymptotic behaviour of the curves. Increasing the interleaving depth, the curves tend to the one achieved on a channel with no memory, which is a line parallel to the x-axis and determined only by the value of $\pi_b$. This happens because, at a certain point, the interleaver manages to make the losses uncorrelated, i.e. the channel behaves as a Bernoulli one. After such point, further increasing the interleaving depth provides no additional benefits. This behaviour is also confirmed by the plots related to the no-loss sequence length (right plots of Fig. 5.5 and 5.6). For this second parameter, we observe how the interleaving actually decreases the no-loss sequence length (i.e. losses get closer), as anticipated in Section 5.3.1. Also, we observe that the trend of the plots is very close to that of the loss-burst length. This is due to the fact that the interleaving does not alter the average loss rate, which can be roughly seen as the ratio between the average loss-burst length, and the sum of the average no-loss sequence length and the average loss-burst length.

From this analysis we can devise a simple heuristic to determine the optimal parameters for the interleaving: the interleaving depth has to be chosen looking at the value for which the loss-burst length reaches the asymptotic value. Then, if we are interested to avoid that the no-loss sequence length goes below a certain value, we have to take into account also the right plots of Fig. 5.5 and 5.6.

### 5.3.4 Discussion and conclusion

In this section, we have provided the following main contributions: i) we have verified that packet interleaving can provide a benefit in terms of loss decorrelation; ii) we have understood the relation between the interleaver parameters and achieved results. The first contribution motivates our next activities, aimed at deploying packet interleaving in real scenarios. As we will see in the following sections, achieving those benefits in real scenarios is far from easy. However, the fact that an interleaver properly configured allows to decorrelate the losses, will be our driving criteria. The second contribution is very important for the real application of time diversity at IP layer. The simulations




(a) Interleaving box before a wireless link. (b) Interleaving box beside a streaming server.

Figure 5.7: Two possible application scenarios.

allowed to understand how to configure the block interleaver parameters according to the status of the network. In the next sections, we will show how this information is exploited to realize a tool for informed time diversity in IP networks.

## 5.4 TimeD: can we obtain such benefits in real networks ?

We stepped from the idea of developing a solution working on every IP-based network, with every transport protocol and every application. For this reason our platform had to work at the IP level.

As a starting point, we thought about two possible application scenarios, which are depicted in Fig. 5.7. In the first scenario, the interleaving (realized by the box called TimeD in such figures) operates on aggregated traffic just before it traverses a wireless link (or a particularly congested path). The second one sees the time diversity employed beside a streaming server, to protect the video/audio traffic from the losses it will encounter on the Internet.

While these are the two scenarios traditionally used to show the effectiveness of the interleaving, our application is actually able to work also in novel networking environments, as we illustrate in the next sections.

### 5.4.1 Design and implementation

After deciding the initial operating scenarios, we passed to the identification of the implementation methodology and tools most appropriate for our aims. We decided to look at Linux operating system (OS) because it allows easier access to and modification of the networking stack with respect to other OS. Moreover, this OS provides more flexibility,

and it allows the interleaving application to be easily deployed as an embedded system (we will call it interleaving box) in any operational network.

We then considered the possibility to work both at kernel and user level. We opted for a user-space application for two reasons: i) it allows quicker and easier prototype development; ii) more importantly, it allows to interact with user-space monitoring applications, whose output is useful to tune the interleaving parameters. The second reason was particularly attractive for us. As explained in Section 5.5, this allowed to extend the application in order to automatically and continuously tune the interleaving parameters in dynamic scenarios, which is possible thanks to the cooperation with network monitoring applications.

After looking at different available possibilities for modifying the order of packets traversing a Linux host in user-space, we decided to use the *libipq*[2]: a mechanism provided by *netfilter*[3] for passing packets out of the kernel stack, queueing them to user-space, and receiving them back into the kernel with a verdict specifying what to do (e.g. ACCEPT or DROP). The queued packets may also be modified in user-space prior to re-injection back into the kernel. With such mechanism, we were able to modify the order of the packets that are enqueued into such *netfilter* queue, before re-injecting them into the kernel. For instance, setting up the following rule on a Linux host:

```
#iptables -A OUTPUT -j QUEUE
```

All the packets that are exiting from such host will be redirected into a queue, from which they can be accessed through the libipq functions. As our application is based on *iptables*, the packets to be interleaved can be selected using several criteria, such as the destination host, the transport protocol and ports, etc.

Passing packets from kernel- to user-space has surely an impact on the performance, which poses a limit on the maximum rate supported by TimeD. However, we were interested in developing a prototype to experiment with time diversity on real networks, having in mind the two initial application scenarios in Fig. 5.7. Therefore, we verified that the delay introduced by such a process is negligible with respect to the one of our experimentation scenarios, and that TimeD was able to sustain the rate used for the experiments. We believe that performance issues have to be more carefully taken into account before the deployment of TimeD in operational environments. We left this as a future work.

[2]https://svn.netfilter.org/netfilter/trunk/iptables/libipq
[3]http://www.netfilter.org

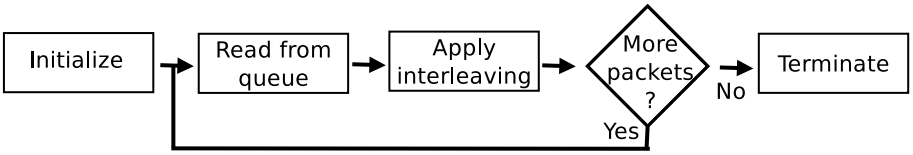



Figure 5.8: TimeD high-level view.

Table 5.2: Parameters of the experimentations.

| Parameter | Values |
|---|---|
| Interleaving block size | 12, 48 |
| Interleaving block depth | [1, 2, 3, 6, 12, 24] |
| $\pi_b$ ($loss$) | [0, 0.01, 0.03, 0.1, 0.25] |
| $\rho$ ($rho$) | [0, 1, 3, 8, 15, 30] |
| Repetitions | 20 |
| Protocols | UDP, TCP, SCTP, DCCP 2, DCCP 3 |
| Packet rate | 10, 100, 1000 pps |
| Packet size | 512 Bytes |

The implementation of TimeD was performed in C language. Fig. 5.8 shows a high-level view of the operations of TimeD. The first step is the initialization of the queue, creating a handle for it and defining the operation mode. After that, the application starts a cycle in which it reads the packets from the queue, inserts them into the interleaving block, reorders the block according to the specified policy, and re-injects the packets into the kernel. At the end of the cycle, i.e. before exiting, TimeD destroys the handle of the queue releasing the resources. This version of TimeD allows to define the size of the interleaving block (number of rows and columns) and an initial value of the timeout to wait for packets (its use is explained in Section 5.5.2), and it uses a fixed policy for packet interleaving. However, we have extended it in different directions, as explained in Section 5.5. An alpha version of TimeD is publicly available at [90] under the terms of the GNU General Public Licence (GPL).

### 5.4.2 Testing and first experimentations

We performed experimentations combining all the parameters reported in Table 5.2. As shown, we considered several variables. For the channel model and the block interleaving size, we tested the same loss patterns we used in simulation, and some more. This allows to verify that our results are as expected, and to uncover issues related to real world

application of the devised techniques. Besides, we used different transport protocols, both traditional (TCP and UDP) and novel (SCTP and DCCP). The choice to consider SCTP and DCCP is motivated by the peculiar characteristics of these novel protocols. SCTP is similar to TCP for what concerns the congestion control strategy and the reliability, even if many parameters are configurable (the reliability for example can be disabled for a specific flow). Moreover, its implementation is rather new with respect to TCP, and the application scenario imagined for this protocol includes real time communication applications (i.e. VoIP). For this reason, SCTP allows to uncover performance aspects related to the interleaved traffic which cannot be discovered with TCP. On the other hand, DCCP is similar to UDP as it is datagram, does not provide reliability and in-order delivery. However, it implements a customizable congestion control, which allows to understand the interaction between such mechanism and the packet interleaving.

**Testing environment**

In Fig. 5.9 we report the testbed we used for the experimentations. It is composed of two Linux hosts connected using a Fast Ethernet network interface to a hardware WAN emulator called Shunra Virtual Enterprise[4]. Such emulator is able to reproduce the behaviour of a WAN in terms of different parameters. For our experiments we used a feature called *WAN Cloud*, which allows to introduce arbitrary delay, jitter, and losses to the packets in transit. We set a loss pattern equal to a 2-MC (the parameters are reported in Table 5.2), and left the delay and jitter unset (i.e. no delay or jitter is intentionally added). On the host named *Server* we run the traffic sender application, TimeD, and the NTP [5] client (i.e. *ntpdate*). On the host named *Client* we run the traffic receiver application and the NTP daemon. NTP was used to synchronize the clocks of the two hosts. While providing a good accuracy in LAN environments, the NTP daemon performs continuous adjustments to the clock in order to cope with clock skew (i.e. the difference between the frequencies of the two clocks). This can impact the results of the measurements as the clocks can change during an experiment. To avoid that, we instruct the NTP daemon on *Client* to only provide the clock on the network, and we manually launch an NTP client (i.e. ntpdate) on *Server*, before each measurement cycle. To cope with the clock skew we performed a clock-skew detection and removal procedure on the acquired data [146].

[4]http://www.shunra.com/shunra-ve-overview.php
[5]http://www.cis.udel.edu/~mills/ntp.html

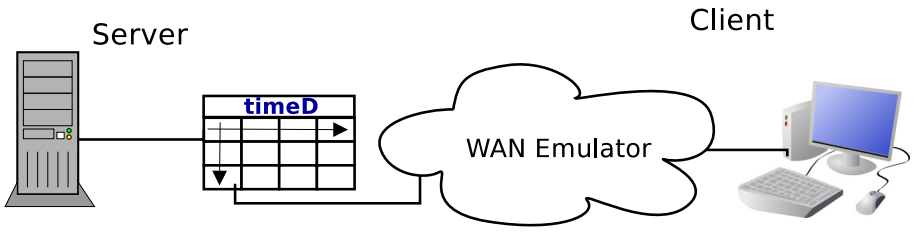



Figure 5.9: Testbed used for the experimentations.

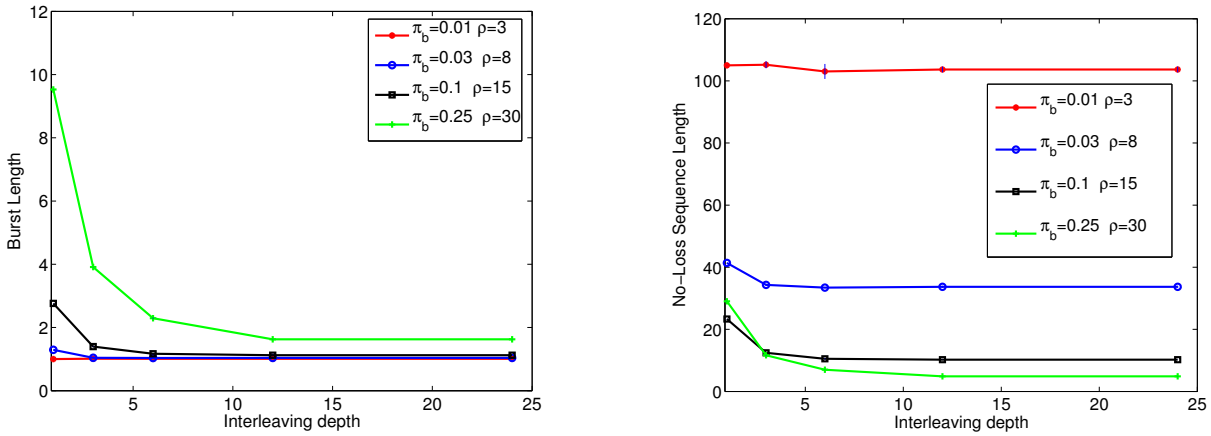



Figure 5.10: Experimental results obtained in four different channel conditions.

For generating probe traffic we use D-ITG [17], a highly customizable, packet-level traffic generator, which supports different transport-layer protocols (UDP, TCP, SCTP, and DCCP), and provides a large number of features for network measurement. Details about the configurations we considered in our experimentations are reported in Table 5.2.

### Results

We first present a comparison in terms of loss decorrelation power between the results obtained in the experimentations and those obtained in simulation. After that, we present results related to the additional delay introduced by TimeD. This is to understand what are the problems we are going to face when deploying such an interleaving strategy on a real networks. In Section 5.5, we analyze in details all the problems we discovered in these experiments, and how we dealt with them. For the first experimentations, we used UDP because we wanted to avoid interferences due to the activities performed by the other transport protocols.

Fig. 5.10 reports the average lengths of loss bursts and no-loss sequences in the four

channel conditions analyzed in Section 5.3.3 (Fig. 5.5): $\rho = 3$ and $\pi_b = 0.01$, $\rho = 8$ and $\pi_b = 0.03$, $\rho = 15$ and $\pi_b = 0.1$, $\rho = 30$ and $\pi_b = 0.25$.

For these tests, we instructed the WAN emulator to reproduce the behaviour of such four channel conditions, and we let the probing traffic traverse the emulated WAN. We also instructed D-ITG to produce a log of the experiments. In such log, D-ITG provides a sequence number and a timestamp for sent and received packets. Using this information we obtained the samples of the performance indicators discussed before (length of loss-bursts and no-loss sequences) as well as the throughput and delay of the packets.. After these experimentations, we also modified D-ITG in order to directly report information about the loss bursts.

The results in Fig. 5.10 have been obtained by sending UDP packets at a constant rate of 100 pps with a payload of 512 Bytes. In such figure, we observe that the average values of the length of the loss bursts and no-loss sequences is decreasing with the increase of interleaving depth. This shows the actual decorrelation power of the interleaving. Moreover, we observe that with a depth equal to 6, we achieve the asymptotic value for almost all the channel conditions.

The results obtained in such experimentations are very close to those from the simulations. This means that the benefit of interleaving can actually be exploited in a real environment thanks to TimeD. It is worth stating that similar consideration can be done for the other traffic and channel conditions. The main differences have been obtained with the other transport-layer protocols. We discuss these issues in details in Section 5.5.

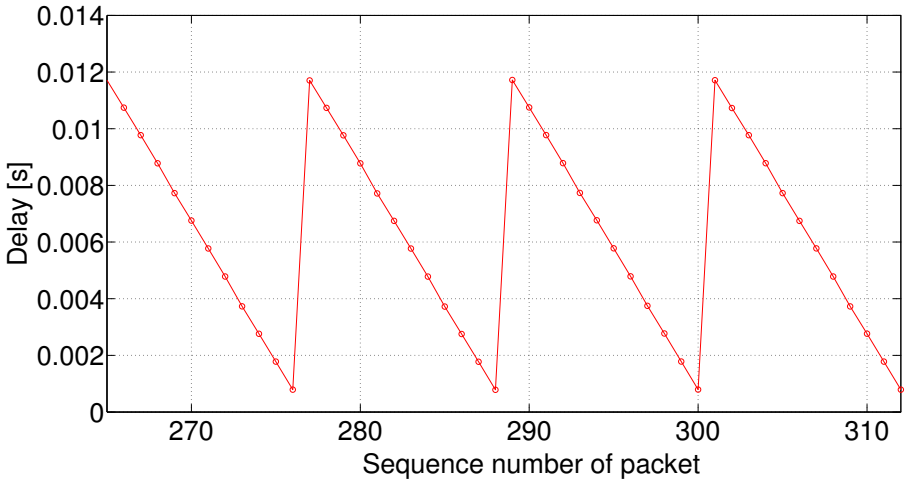



Figure 5.11: Packet delay of a UDP flow subject to a 3x4 block interleaving.

Beside the capability to decorrelate the losses we were also interested in understanding the additional effects that TimeD has on packet transmission. In simulation, we neglected the delay issues because we wanted to understand the potential benefit and the optimal configuration. However, if we want to deploy the interleaving on a real network, we have to consider all the effects that we might expect. For these experiments, we connected the two hosts in the testbed back-to-back, disabling the WAN emulator. We then performed two different kinds of measurements to estimate the two main delay components: the *forwarding delay* and the *buffering delay*.

To estimate the forwarding delay we performed experimentations with TimeD configured to use a block depth = 1, and we injected UDP packets at all the rates reported in Table 5.2. With such a block depth, TimeD was forwarding the packets as soon as they entered the queue, and no buffering was performed. We then performed the same experiments without TimeD, and compared the delays obtained.

From this analysis we discovered that the overhead due to forwarding operations is in the order of $10\mu$s at all the considered packet rates. We believe this delay can be considered negligible, at least for the two application scenarios considered.

For the buffering delay, we measured the transfer time experimented by UDP packets using all the rates and interleaving configurations reported in Table 5.2. Fig. 5.11 shows a zoom of the packet-by-packet delay of a UDP flow, with rate of 1000 pps, passing through TimeD instructed to perform a 3x4 block interleaving. The saw-tooth trend is due to the buffering performed by the interleaver, which waits for a block to be completed before re-injecting the packets into the network. The result is that the first packets are kept in the buffer for longer with respect to the last ones.

In particular, the buffering delay experimented by the $i$-th packet of a block of size $N$ is equal to

$$\delta_i = \sum_{j=i+1}^{N} IPT_j \quad \forall i = 1, \ldots, N-1 \tag{5.13}$$

where $IPT_j$ is the time elapsed between the arrival of packet $j-1$ and $j$ (i.e. the inter-packet time). The last packet will not experiment any buffering delay. In the CBR traffic case, $IPT$ is constant and the (5.13) becomes

$$\delta_i = (N-i) \times IPT \quad \forall i = 1, \ldots, N \tag{5.14}$$

This explains the regular trend of Fig. 5.11.

As already remarked before, such delay has to be carefully taken into account from the application point of view. For example, if we want to use the interleaving for the streaming server in Fig. 5.7, we may assume that the stream has a constant packet rate of 720 pps (i.e. frame rate of 24 frames per second, 30 slices per frame, and a single slice per packet) and packet sizes following a normal distribution [147]. In this case, using an interleaving block of 48 packets, implies a maximum buffering delay of about 65 ms, which is acceptable in most cases. However, such buffering delay will be a cause of problems for transport protocols implementing a congestion control algorithm. We discuss this issue in details in the following section.

Finally, we performed experiments aimed at understanding the maximum throughput that TimeD is able to sustain. From these experiments we learned that TimeD is able to work at full speed (i.e. 100Mbps) with all the interleaving configurations in Tab. 5.2, even when running on the same host as the traffic generator.

## 5.5 Problems identified and solutions devised

The results presented in the previous section allowed to understand that the interleaving in a real network behaves very closely to the simulation environment. This suggests that we can actually exploit such technique in real scenarios. It is important to recall, however, that such results have been obtained in highly controlled environments, not only from the point of view of the network-dynamics, but also from that of the interleaver configuration. This means that a user would have to manually configure TimeD according to both the network dynamics and the protocol used by the application, which is clearly in contrast with our aim of deploying packet-level interleaving in IP networks despite the application and protocol in use. To cope with this issue, we added measurement capability to TimeD, allowing the tool to auto-configure the operating parameters and provide the required benefits in all the scenarios. These activities are described in the next sections.

### 5.5.1 Block size

The first and most important problem evidenced from the analysis of Section 5.4.2 is related to the fact that the block size has to be tuned according to the loss pattern on the network. Actually, one may think that it is sufficient to adopt a block size (mainly in terms of interleaving depth) that guarantees the loss decorrelation with all the possible loss

patterns. The problem is that a large interleaving depth translates into a large buffering delay, as shown in Section 5.4.2. Therefore, we have always to choose the lowest possible interleaving depth that provides the required loss decorrelation.

To achieve this, TimeD has to infer the loss pattern on the network and to choose the right interleaving depth, according to such losses. The parameters that have to be estimated are the loss rate $\pi_b$ and the loss correlation $\rho$. If we want to work with every possible transport protocol, we cannot assume that such protocol provides a feedback to the transmitting host (as TCP does). Therefore, we can only rely on active measurement techniques to estimate these parameters. To this end, we developed an improved version of TimeD that exploits the features provided by D-ITG, as done for informed path diversity. Such improved version periodically performs active measurements with UDP, estimates $\pi_b$ and $\rho$ from the received traffic, and adjusts the block size accordingly.

Accomplishing the aforementioned tasks requires the investigation of different aspects: i) the measurement period, i.e. how frequently to estimate (and re-estimate) the status of the network; ii) the parameters to use for the probing traffic, i.e. the duration of the measurement flow, and the size and rate of its packets; iii) the block size to be used as a function of the loss pattern. These aspects are discussed in the Section 5.6.1.

### 5.5.2 Identifying the transport protocol

The results presented in Section 5.4.2 allowed to understand that the interleaving in a real network behaves very closely to the simulation environment. However, it worth saying that part of the merit is surely of UDP, which does nothing more than adding a small (yet important) header to IP packets. In current Internet scenarios, multimedia traffic is transported more and more frequently by protocols implementing at least a congestion control algorithm. This is mainly due to the fact that such protocols are better suited to adapt to varying network conditions For this reason, it is important to understand the interactions between packet interleaving and congestion control. To this end, we performed a set of experiments with different transport protocols implementing such a control: TCP, SCTP, and DCCP with both CCID 2 and CCID 3 (we simply call them DCCP 2 and DCCP 3 in the following).

In these experiments, we completely disabled the WAN emulator and we connected the hosts back-to-back. This was to isolate the effect of the losses from that of the interleaving. The first condition we experimented was that the connection between the

two hosts was not established with almost all the protocols. Only TCP was able, in some cases, to complete the 3-way handshake and to send some packets. This is due to the fact that the congestion control algorithms start sending packets at a very low rate, which is only increased when acknowledgments are received. The buffering operation performed by TimeD is clearly in contrast with such a behavior, because the block has to be filled in order to release the packets. The reason why TCP was able to actually overcome such big obstacle is that TCP initially filled the buffer with several retransmissions of the first packets. DCCP instead, while implementing a congestion control algorithm, does not retransmit packets. So it was completely stuck in this situation. For SCTP, finally, the number of maximum allowed retransmissions for the first packets was set to a lower value with respect to TCP. Therefore, also for such protocol TimeD locked the connection. It is also worth noting that the performance of TCP in this situation was really low, and the end-to-end delay reached values up to 30 s.

These results claim for ad-hoc solutions for protocols with congestion control. We therefore modified TimeD behaviour to be able, first of all, to understand the kind of protocol used by the application, and then act differently on the traffic of different protocols.

To identify the transport protocol, TimeD reads the payload of the first packets that are entering the netfilter queue. In particular, it reads the content of the field *protocol* from the IP header of such packets. From this information it determines if the flow rate is governed by a congestion control or not [6]

To avoid blocking traffic flows of protocols with congestion control, we modified TimeD introducing a timeout for the buffering operation. Basically, thanks to this modification, TimeD releases the packets when either the interleaving block is full or a certain amount of time has passed. A different approach is then adopted for setting the timeout of the buffering operation depending on the transport protocol, as described in the following.

### 5.5.3 Timeout for UDP

In the case of UDP, the flow rate is not determined by the conditions of the network but rather by the application behaviour. Besides the case of applications implementing a congestion control on top of UDP, we can generally assume that UDP flows have an

[6]Some applications are starting to implement a congestion control on top of UDP. We are aware that our current approach does not recognize correctly the flows of such applications. However, for this issue, it would be necessary to use more sophisticated traffic identification approaches such as those used by [148], which are out of the scope of this thesis.

average rate that is fixed during the entire transmissions. We are thinking, in particular, to applications for audio/video communications. While the flow rate of these applications is not necessary constant (MPEG traffic, for example, produce VBR flows), we can consider this rate to be fixed in average. This means that, if we observe the rate on sufficiently large intervals, the average value will be the same on all the observation periods.

The previous considerations drove us to devise a timeout setup mechanism that estimates the average rate of the flow, and then sets the buffering timeout as a function of this rate. The mechanism works as follows. Let us suppose that we want to operate with an block interleaver of size $n \times m = k$, on a flow of packets $p_1, p_2, ..., p_n, ...$, whose arrival times are $t_1, t_2, ..., t_n, ....$ Every time we release the current block because, either the block is filled or the timeout ($TO$) is expired after receiving $p$ packets (with $p < k$), we calculate a new rate sample $r(j)$ using the following:

$$r(j) = \frac{k}{t_n - t_{n-k}} \quad \text{if block is filled}$$

$$r(j) = \frac{p}{TO} \qquad \text{if timeout is expired}$$

And, we calculate the average and standard deviation of the flow rate using the following:

$$r_{avg} = \frac{1}{j}\sum_{i=1}^{j} r(i); \quad r_{std} = \sqrt{\frac{1}{j}\sum_{i=1}^{j}(r(i) - r_{avg})^2} \tag{5.15}$$

At the same time, we setup the timeout for the buffering operation, calculated as follows:

$$TO = \frac{r_{avg} + 4 * r_{std}}{k} \tag{5.16}$$

In this way, we update the timeout using the average and standard deviation of the packet rate, which are updated every time the block is filled or the previous timeout is expired. In Section 5.6.2, we show that in this way we are actually able to set the proper timeout value for UDP flows.

### 5.5.4 Timeout for protocols with congestion control

In case the protocol uses a congestion control, we expect the rate to variate continuously over time. For this reason, the previous mechanism has to be changed. In details, we cannot assume anymore that the rate will remain constant in average. Instead, most of

the congestion control algorithms start the transmissions sending a very small quantity of packets (i.e. 1 or 2), and they progressively increase the packet rate until the available bandwidth on the path has been reached. From this moment, the rate is maintained as constant as possible, unless the status of the network changes. Therefore, the initial timeout has to be set to a very small value, in order to avoid blocking the transmission in the initial phase, and it has to be varied more quickly with respect to the case of UDP. Moreover, it is not possible to estimate the packet rate anew for every block because, when the transport protocols use a window-based congestion control, the size of the window is not synchronized with the block size (i.e. the window can either be smaller than the block or span over multiple blocks).

Stepping from these considerations, we decided to use a timeout set-up mechanism that works as follows. Let us suppose that we want to operate with an block interleaver of size $n \times m = k$ on a flow of packets $p_1, p_2, ..., p_n, ...$, whose arrival times are $t_1, t_2, ..., t_n, ....$ Starting from the second packet, every time a new packet $p_n$ enters the buffer, we calculate a new rate sample $r(n)$ using the following

$$r(n) = \frac{k}{t_n - t_1} \qquad \text{if } n <= k$$

$$r(n) = \frac{k}{t_n - t_{n-k}} \quad \text{if } n > k$$

Then, every time a block is released, we setup the timeout as follows

$$TO = \frac{r_n}{k} \tag{5.17}$$

The way we setup the timeout in the case of protocols with congestion control differs from that we use for UDP in two main aspects. Indeed, in this case, we: i) update the rate for every new packet and not for every new block, and ii) we use the last value of the estimated packet rate instead of the average one. Thanks to these differences we are able to react more quickly to the rate variations, that we expect to happen with these kinds of protocols.

### 5.5.5 A note on reliable protocols

After speaking about congestion control, an important aspect has to be clarified concerning the interactions between TimeD and TCP, or, in general, any transport protocol

implementing a reliability mechanism. For those protocols, indeed, the loss decorrelation is not visible for the applications. The reliability mechanisms takes care of packet acknowledgement and of retransmissions in case of losses. In case the network is not fully congested, this mechanism is able to recover from losses and deliver all the transmitted packets. Therefore, applications using these protocols are not aware of losses happening at lower layers. For this reason, loss decorrelation is useful only if the transport protocol is impacted by bursty losses.

Works in literature reported that TCP is actually impacted by bursty losses [149], verifying that a TCP flow recovers more slowly from bursty losses than from uncorrelated losses. To cope with this problem, new versions of TCP have been standardized [43]. However, the versions of TCP currently in use are not fully compliant with these standards, and it is difficult to predict if and when this will happen [150]. This means that current applications using TCP should experiment a higher throughput if the losses on the network are opportunely decorrelated.

However, when modeling TCP throughput, studies such as [149] refer to bursty losses as to losses involving more than one packet in the same congestion window. This means that, to provide the required benefits to current applications, it would be necessary to distribute losses over more congestion windows. Unfortunately, with current approaches for packet interleaving this is not possible. Indeed, the congestion window refers to a set of packets which have been transmitted but not yet acknowledged (i.e. packets in flight). The packet interleaver can only redistribute the losses over packets in the same block, i.e. packets currently buffered by the interleaved. Such packet will not be acknowledged until released from the buffer and received by the end host. Therefore they belong to the same congestion window. For this reason, loss decorrelation can only be realized on packets in the same congestion window, i.e. translating bursty losses happening into one congestion window into uncorrelated losses in the same window. This kind of loss decorrelation should not provide any benefit to TCP.

## 5.6 Towards informed time diversity

In this section we discuss the lessons learned in order to deploy TimeD in real scenarios, and our experiments with the improved version of TimeD. This is to understand the benefits of informed time diversity.

### 5.6.1 Estimating loss pattern

As described in 5.5.1, the estimation of the loss pattern (i.e. loss rate and correlation) is necessary to setup the proper block size. TimeD has to perform ad-hoc measurements on the network, and three main parameters have to be configured: i) the measurement period, i.e. how frequently to estimate (and re-estimate) the status of the network; ii) the parameters to use for the probing traffic, i.e. the duration of the measurement flow, and the size and rate of its packets; iii) the block size to be used as a function of the loss pattern.

#### Measurement period

The measurement period is in strict relation with how frequently the loss pattern on the network changes. To set this parameter we performed an analysis of the literature about network dynamics. Particularly interesting has been the work [64], in which the authors perform a long-term measurement campaign aimed at understanding the potential benefits of path switching. Results reported in this work have also been used for our activities described in Section 4.5.

With regards to setting the proper measurement period, this work is interesting because it reports the long-term delay and loss rate measured over different paths connecting the same hosts. The results show that, in terms of delay, there is no big variation, and the best path remains the same for almost the entire measurement period (one week). In terms of loss rate, the results show that there is high variability. The authors show that, if observed with a resolution of 1 minute (i.e. computing the average loss rate over 1 minute intervals), the paths look quite different. They also show that the loss rate remains the same for a period that ranges from 1 to 22 minutes. The average value of this loss-variation period is $1 \rightarrow 4$ minutes. For this reason we decided to reiterate our measurements every 1 minute. In this way we are able to strictly follow the dynamics of the network.

#### Probing flow characteristics

Once we know the right period to estimate the network loss, we have to decide how to perform the measurements in this period. Here, different contrasting interests have to be considered, that are the measurement intrusiveness and the accuracy. In facts, the

more accurate we want our measurements to be, the more samples we have to collect. And, once the measurement period is fixed, to collect more samples we have to probe the network with a higher packet rate, which implies an higher intrusiveness.

To choose the parameters of the probing flow, we have performed a set of experiments in a controlled scenario. Using the testbed reported in Fig. 5.9, we have performed a large set of 1-minute long measurements with D-ITG, varying the packet rate and size and varying the loss pattern on the emulated WAN. We then looked at the values of loss rate and correlation estimated by D-ITG. The results showed that the loss rate is easily estimated with a small probing rate. In particular, we observed that a probing rate of 10 pps was already enough to obtain a relative error in the order of 0.1, which is enough for our aims. As for the rate correlation, instead, we observed a different situation. Fig. 5.12 shows the relative error obtained in two experiments performed using $\pi_b = 0.01$, one with $\rho = 3$ and the other with $\rho = 30$. As shown, the probing rate as an impact, and at low probing rates (e.g. 10 pps) the relative error reaches values up to 2.

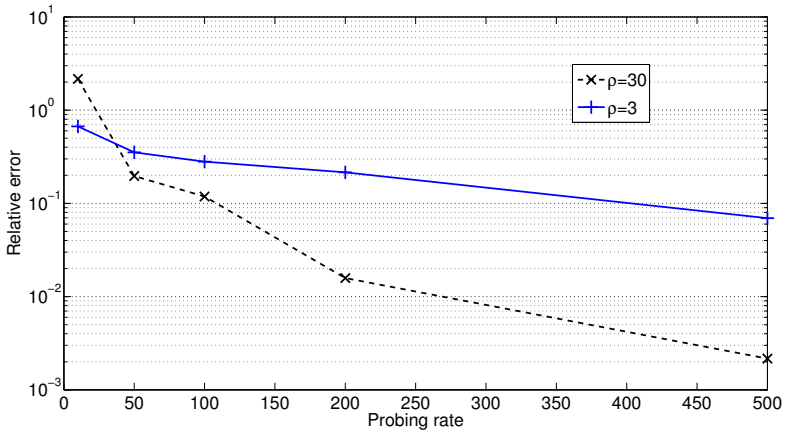



Figure 5.12: Relative error of loss correlation estimation.

However, as these results are necessary to setup the proper block size, the measurement accuracy required depends only on that necessary for this aim. In fact, the results of the simulations performed in Section 5.3, which are discussed in the next section, indicate that it is necessary to estimate the loss correlation with an accuracy in the order of few units, i.e. we can accept a relative error in the order of $10^0$. For this reason, we chose to probe the network at a packet rate equal to 50 pps. This guarantees both a sufficient accuracy and a low intrusiveness of the probing traffic (50 pps $\times$50 Bytes/pkt = 20 Kbps).

**Block size to be used for a given loss pattern**

The simulations performed in Section 5.3 allowed to understand the block size necessary for a given loss pattern. In particular, in that section we have shown how the interleaver depth is the most important parameter to be considered, in order to obtain the required loss decorrelation. Table 5.3 shows the interleaving depth sufficient to obtain uncorrelated losses for different loss pattern. This table has been obtained looking at ll the results of the analysis presented in Section 5.3. A depth value equal to 1 in this table means that the interleaving is not necessary.

Table 5.3: Interleaving depth as a function of the loss pattern.

| | $\pi_b < 0.01$ | $0.01 <= \pi_b < 0.02$ | $0.02 <= \pi_b < 0.05$ | $0.05 <= \pi_b < 0.15$ | $0.15 <= \pi_b < 0.25$ | $\pi_b > 0.25$ |
|---|---|---|---|---|---|---|
| $\rho <= 1$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $1 < \rho <= 5$ | 1 | 6 | 6 | 12 | 24 | 24 |
| $11 < \rho <= 22$ | 1 | 6 | 12 | 12 | 24 | 24 |
| $22 < \rho <= 30$ | 1 | 12 | 12 | 24 | 24 | 24 |
| $\rho > 30$ | 1 | 24 | 24 | 24 | 24 | 24 |

This table has been provided as input to TimeD. Basically, looking at the values reported in this table, TimeD automatically configures the block depth most suited to the loss pattern measured on the network, by using the procedure illustrated in Section 5.6.1. Note, however, that these results are related to a block size equal to 48, which guarantees the loss decorrelation in all the considered conditions. In case the application has strict time requirements, the block size can also be set to 24, using the block depth reported in Table 5.3. Using this information, TimeD is actually able to work autonomously in real application scenarios.

### 5.6.2 Estimating protocol and source rate

We performed a set of experiments in a controlled testbed to evaluate the capability of TimeD to accurately estimate the packet rate of the traffic flow it has to work on. This is an important aspect as TimeD uses this information to setup the proper timeout value. This means that, if the rate estimation does not provide accurate results, the packets will not be correctly buffered. In particular, if the rate is overestimated the timeout will be set to a too-small value, and the interleaving block may not be completely filled. On the other hand, if the packet rate is underestimated, the timeout will be set to a too-large

value, and the packets will remain for longer into the buffer.

To understand the performance of our rate estimation mechanism, we performed a set of experiments in a controlled scenario. Using D-ITG we simulated the behaviour of two users communicating by using TCP and UDP flows at different rates. In Fig. 5.13 we illustrate the results we obtained when the emulated users were communicating by using UDP. On the $x$ axis of this figure we report the packet rate of the communication. We performed tests with rates equal to $10, 50, 100, 500, 1000$ packets per second (pps). For each packet rate value, we performed 20 tests. On the $y$ axis we both the average and the standard deviation of the rate estimated by TimeD. The standard deviation is represented by the length of the vertical segment centered in the average rate value. In the plot, we report also the line corresponding to the reference value (i.e. perfect estimation).

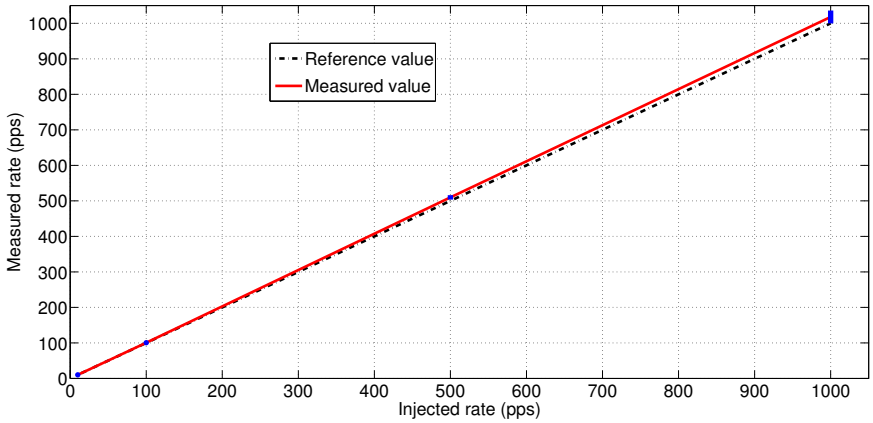



Figure 5.13: Results of packet rate estimation (UDP).

As we can see, with packet rates up to 100 pps, the two lines are undistinguishable, which means that the estimation provides very accurate results. A small difference is observable with rates values of 500 and 1000 pps. In particular, we observe a little overestimation performed by TimeD. However, no significant inaccuracies are noticed. We also note that the variance is always very small, as its segment is visible only at 1000 pps. Overall, we can conclude that the rate estimation mechanism provides accurate results.

### 5.6.3 Using the automatic timeout setup

To understand the improvements provided by the mechanism for automated timeout setup, we performed a set of experiments with different transport protocols and rates using D-ITG to emulate the user application. We connected the two hosts in Fig. 5.9 back-to-back. Therefore no losses were induced on the link. This allowed to understand the impact of the timeout setup mechanism in isolation, without the interference of other variables. Recall here that, without such mechanism, UDP was experimenting the delay reported in Fig, 5.11, while the protocols with congestion control algorithms were blocked by the interleaver buffering.

Table 5.4 shows the average delay experimented by the packets with different transport protocols and interleaving configurations. Remember that we are not inducing any loss on the path. Therefore we are only observing the effect of the interleaver. Recall also that the average buffering delay for a CBR flow interleaved with a block of size $N$ can be obtained using Eq. (5.14). In particular, the average buffering delay $\delta_{avg}$ is

$$\delta_{avg} = \frac{1}{N}\sum_{i=1}^{N}\delta_i = \frac{(N-1)\times IPT}{2} \tag{5.18}$$

If we look at the values reported in Table 5.4, for UDP we can observe that they are very close to the theoretical values. On the contrary, all the other protocols still considered the buffering performed by TimeD as a congestion. As a result they reduced the sending rate, and their packets experiment a higher delay in average. Such average delay is actually due to the fact that, when the congestion control reduces the transmission rate, out timeout-setup mechanism requires some time to react. Therefore, a few packets experiment a high delay.

It is interesting to note that the average delay experimented by DCCP packets is higher than that of TCP and SCTP. This is due to the fact that the last protocols performed a lot of retransmissions. As a consequence the actual rate of packets in the network is higher, and the packets waited in the buffer for a shorter time.

Table 5.5 presents the number of packets received using the 4 different protocols when generating CBR traffic at 100 pps for 120 s, and using a 3x4 interleaving block. We observe that UDP, TCP, and SCTP were able to transfer all the packets injected by D-ITG, while DCCP was able to transfer less than the half of such packets. The behaviour of UDP is clearly due to the fact that it completely ignores the status of the network.

Table 5.4: Average delay of the packets at 100 pps.

| Block size | Delay [s] | | | | |
|---|---|---|---|---|---|
| | *UDP* | *TCP* | *SCTP* | *DCCP* 2 | *DCCP* 3 |
| 3x4 | 0.059 | 0.123 | 0.197 | 0.280 | 0.277 |
| 6x2 | 0.059 | 0.123 | 0.146 | 0.280 | 0.276 |

TCP and SCTP were slowed down by TimeD in the first period. However, they were able to quickly recover thanks to the retransmissions. After a while, they reached a sending rate even higher than that requested by the application, and they managed to send all the requested packets.

Table 5.5: Percentage of received packets sending 100 pps for 120 s.

| Block size | Received packets [%] | | | | |
|---|---|---|---|---|---|
| | *UDP* | *TCP* | *SCTP* | *DCCP* 2 | *DCCP* 3 |
| 3x4 | 100.0 | 99.9 | 99.8 | 40.8 | 41.4 |
| 6x2 | 100.0 | 99.9 | 99.9 | 41.5 | 41.6 |

## 5.7 Final remarks

In this chapter we provided contributions towards the deployment of informed interleaving in IP networks. We performed an analysis of this problem using simulations and experiments on a laboratory testbed. The combined approach allowed to progressively unveil all the issues related to the application of this technique. To cope with them, ad-hoc measurement strategies were devised. We aimed at deploying diversity schemes at IP layer, independently on the particular application and protocol in use. In this chapter we saw that packet interleaving requires information about the network, but it also requires information about he application (i.e. the transport protocol and packet rate). This is because the interleaving must have a different behavior in different scenarios in order to provide the promised benefits. We believe that this analysis allows to understand that to achieve our goal we cannot use simple diversity schemes. Informed diversity techniques are necessary to cope with the complexity of current network scenarios.

# Chapter 6

# Conclusion of the thesis

In this thesis we pursued the objective of understanding how it is possible to use informed techniques for time and path diversity at IP layer. We considered both theoretical and implementation issues, showing how a careful investigation methodology allows to pass from theory to practice, progressively taking into account, and possibly solving, all the related problems. In the next section we review all the innovative contributions we provided during this research work, which constitute the single building blocks of our thesis.

## 6.1 Summary of findings and conclusion

One of the main claims of this thesis is that diversity techniques can achieve better performance if supplemented by information about the status of the network on which they operate. For this reason, the first problem we had to face is how to acquire information about the status of a heterogeneous network. In particular, we had to understand how to effectively and efficiently measure QoS parameters in current Internet scenarios, and then, how to extract from the measured information, the most important indicators of the network status.

To this end, we performed four experimental campaigns over different networking scenarios, exploring the possibilities offered by active and passive measurement techniques. As for the active, we provided a first contribution, collaborating to the development of a tool, called D-ITG [17], that allows to perform traffic generation and QoS parameter measurements. We added several innovative features to this tool, including the possibility to run on different architectures and operating systems, and to generate traffic using new transport protocols. This allowed to perform measurement experiments in real het-

erogeneous scenarios.

Using the new features of D-ITG, we analyzed the performance of three different networks: PlanetLab, Magnets, and HetNet. PlanetLab is the state-of-the-art research infrastructure for testing innovative services and applications, being constituted by a thousand hosts, spanning several countries all over the world. We firstly complemented the architecture to overcome one of its main limitations, providing the possibility to use UMTS connections. Exploiting this feature, we performed a measurement campaign aimed at comparing the performance achievable using two different paths connecting the same two nodes of the testbed: the first path was made of all wired links, and the second one comprised a UMTS link.

Magnets is a wireless wide area network deployed by Deutsche Telekom Laboratories in Berlin, and characterized by several innovative features (multiple different technologies, diverse link characteristics, nodes with varying degrees of processing and storage capabilities, etc.). In this activity, we focused on the analysis of the performance of the high-speed WiFi backbone of this infrastructure. We characterized the QoS parameter behaviour, exploiting all the special features of this infrastructure. This allowed to understand, among other things, the influence of environmental factors on a such wireless WAN.

HetNet is a laboratory testbed, characterized by a high degree of heterogeneity: different networking technologies, different operating systems, different devices, etc.. We performed a measurement campaign taking into consideration all these variables, evaluating several statistics of the collected measures, using an innovative approach. Such activity allowed to highlight some behaviors that were hidden when applying a traditional approach, and to better understand the differences between traditional networks and heterogeneous wired/wireless ones.

We also worked to the analysis of an operational network using a passive measurement technique. Using packet traces collected in the cellular network of one of the major operators of central Europe, we studied the behavior of the network by looking at the traffic generated by the users. We started the investigation by using a state-of-the-art methodology, and then, we designed a new methodology, that overcomes the limitations of the previous one. We studied the performance of all the users of the cellular network, and we zoomed into the properties of bulk data transfers of some specific users, which reveal interesting insights into the performance of the network.

The knowledges acquired in the previous activities, allowed to devise new techniques for informed diversity. As for the path diversity, we approached the problem performing simulations in Matlab. We developed a simulator that allowed to explore the potential benefits of classical path diversity techniques, such as the round robin. In the simulator we used a model for packet loss based on 2-state Markov chains, which captures the correlation of the loss process in current Internet scenarios. The results allowed to understand the capability of this packet distribution scheme to transform bursty losses into isolated ones.

A technique for informed path diversity was then developed. The technique exploits Markov Decision Processes to make informed path-selection decisions, minimizing or maximizing a function of the QoS parameters (e.g. minimize the file transfer time, minimize the losses, etc.). The status of the available paths is estimated by using a passive measurement technique. The approach was evaluated with *ns2* simulations in different application scenarios, comparing the performance achievable with that of two classical packet distribution schemes. Obtained results showed how our technique is able to achieve better performance than the others, in all the considered scenarios.

To verify if such benefits may also be achieved in real networks, we developed a tool to experiment with informed path diversity in real scenarios. For this aim, we had to solve issues related to the time granularity of the path selection. We then developed a technique that switches the path with less frequency with respect to the previous one. Such technique was implemented in a tool called PathD, which performs active measurements on the available paths, to obtain accurate information about the path status. The performance of PathD were evaluated on a controlled network scenario, and compared with classical path diversity schemes, showing that our approach achieves better performance.

We provided innovative contributions in the research field of path diversity proposing new techniques and tools that achieve better performance than the ones currently used. As for time diversity, instead, the objective of our research was how the establishment of how to apply such a transmission schema at IP layer. Several approaches have been proposed in literature to use time diversity in computer communications. However, only few works have been proposed for the IP layer, and none of them is really independent of the application and the underlying technology.

As done for path diversity, we firstly studied the potential benefits of time diversity at IP layer (i.e. packet interleaving) in simulation, using a simulation framework we devel-

oped in Matlab. This activity allowed to verify that a block interleaver is able to provide the expected benefits in terms of loss decorrelation, if properly configured. We also clearly depicted the relation between the interleaver parameters and the loss-burst length (i.e. the number of packets consecutively lost).

Thanks to this result we could move towards the realization of informed time diversity. We designed and implemented an application for packet-level time diversity in real scenarios. After the implementation, we evaluated the impact of the packet manipulation process, and other side activities performed by our tool. An experimental measurement campaign allowed to understand the pros and cons of packet interleaving in real networks: on the one hand, they confirmed the simulation results, showing that time diversity can actually be helpful in real networking scenarios; on the other hand, they evidenced some important issues related to real world application (the impact of the buffering operation performed by the interleaver on transport protocols with congestion control algorithms, the necessity to manually tune the interleaver parameters, etc.). We coped with these issues by adding measurement capabilities to TimeD, and devising a dynamic buffering mechanism. Using active measurement tools, TimeD is able to estimate the status of the network in terms of loss characteristics, and therefore, to use the interleaving configuration most suited for this situation. Using passive techniques, TimeD estimates the protocol used by the application that is generating traffic, and the packet rate of such traffic. Thanks to these sensing capabilities, TimeD is actually able to provide the required benefits in heterogeneous environments. Such benefits were evaluated performing a measurement campaign in controlled networks.

We believe that this thesis paved the way to the deployment of space and time diversity in IP networks. We also believe that these techniques can actually mitigate the problems of current networking scenarios, moving a step ahead towards the Future Internet.

# Bibliography


[1] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[2] ANSI/IEEE Stdandard 802.11. http://standards.ieee.org/getieee802/download/802.11-2007.pdf as of November 2009.

[3] G. Xylomenos, G.C. Polyzos, P. Mahonen, and M. Saaranen. Tcp performance issues over wireless links. *Communications Magazine, IEEE*, 39(4):52–58, Apr 2001.

[4] Emir Halepovic, Carey L. Williamson, and Majid Ghaderi. Wireless data traffic: a decade of change. *IEEE Network*, 23(2):20–26, 2009.

[5] Ramesh Jain. I Want My IPTV. *IEEE MultiMedia*, 12(3):96, 95, 2005.

[6] Dong Hee Shin. Potential user factors driving adoption of iptv. what are customers expecting from iptv? *Technological Forecasting and Social Change*, 74(8):1446 – 1464, 2007.

[7] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14, New York, NY, USA, 2007. ACM.

[8] B. Garfield. The YouTube effect. *WIRED*, 14(12):222, 2006.

[9] Susanne Boll. Multitube–where web 2.0 and multimedia could meet. *IEEE MultiMedia*, 14(1):9–13, 2007.

[10] M. Knights. Web 2.0. *Communications Engineer*, 5(1):30–35, February-March 2007.

[11] Ilya V. Yakovlev. Web 2.0: Is it evolutionary or revolutionary? *IT Professional*, 9(6):43–45, Nov.-Dec. 2007.

[12] E Kohler, M Handley, and S. Floyd. RFC 4340: Datagram Congestion Control Protocol (DCCP), IETF, 2006.

[13] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving tcp performance over wireless links. *IEEE/ACM Trans. Netw.*, 5(6):756–769, 1997.

[14] Alessio Botta, Antonio Pescapé, and Giorgio Ventre. Quality of service statistics over heterogeneous networks: Analysis and applications. *European Journal of Operational Research*, 191(3):1075 – 1088, 2008.

[15] N. Seitz. Itu-t qos standards for ip-based networks. *Communications Magazine, IEEE*, 41(6):82–89, June 2003.

[16] D. Clark, R. Braden, and K. Sollins. Tussle in cyberspace: defining tomorrow's Internet. In *ACM SIGCOMM: conference on applications, technologies, architectures, and protocols for computer communications*, 2002.

[17] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. Multi-protocol and multi-platform traffic generation and measurement. In *IEEE Conference on Computer Communications (INFOCOM '07), Demo session*, 2007.

[18] Comics research group. http://www.comics.unina.it as of November 2009.

[19] Snapgear. http://www.snapgear.org as of November 2009.

[20] Montavista embedded linux. http://www.mvista.com as of November 2009.

[21] Openwrt. http://openwrt.org as of November 2009.

[22] Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Rfc2960: Stream control transmission protocol, 2000.

[23] Matti Siekkinen, Guillaume Urvoy-Keller, Ernst W. Biersack, and Denis Collange. A root cause analysis toolkit for TCP. *Elsevier Comput. Netw.*, 52(9), 2008.

[24] Vern Paxson. End-to-end Internet packet dynamics. *SIGCOMM Comput. Commun. Rev.*, 27(4):139–152, 1997.

[25] M. Yajnik, Sue Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 345–352 vol.1, Mar 1999.

[26] Jean-Chrysostome Bolot, Hugues Crépin, and Andrés Vega-García. Analysis of Audio Packet Loss in the Internet. In *NOSSDAV '95: Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 154–165, London, UK, 1995. Springer-Verlag.

[27] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the MBone multicast network. In *Global Telecommunications Conference, 1996. GLOBECOM '96. Communications: The Key to Global Prosperity*, pages 94–99, Nov 1996.

[28] Petrovic Sanja. Analysis of packet loss process in multipath networks. *EPFL Technical Report*, 2004.

[29] S.H. Chung, D. Agrawal, M.S. Kim, J.W. Hong, and K. Park. Analysis of Bursty Packet Loss Characteristics on Underutilized Links Using SNMP. *IEEE/IFIP End-to-End Monitoring Techniques and Services (E2EMON)*, 2004.

[30] S. Tao and R. Guerin. On-line estimation of internet path performance: an application perspective. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1774–1785 vol.3, March 2004.

[31] K. Nayak and D. McKernan. Measuring provider path diversity from traceroute data: Work in progress. In *CAIDA-ISMA Winter Workshop*, December 2001.

[32] Aditya Akella, Bruce Maggs, Srinivasan Seshan, Anees Shaikh, and Ramesh Sitaraman. A measurement-based analysis of multihoming. In *ACM SIGCOMM: conference on applications, technologies, architectures, and protocols for computer communications*, pages 353–364, New York, NY, USA, 2003. ACM.

[33] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in isp networks. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 313–318, New York, NY, USA, 2003. ACM.

[34] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Skype video responsiveness to bandwidth variations. In *NOSSDAV '08: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 81–86, New York, NY, USA, 2008. ACM.

[35] S.-F. Chang and A. Vetro. Video adaptation: Concepts, technologies, and open issues. *Proceedings of the IEEE*, 93(1):148–158, Jan. 2005.

[36] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The end-to-end effects of internet path selection. *SIGCOMM Comput. Commun. Rev.*, 29(4):289–299, 1999.

[37] J.G. Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *Visual Communications and Image Processing (VCIP)*, 2001.

[38] Yi J. Liang, Eckehard G. Steinbach, and Bernd Girod. Real-time voice communication over the internet using packet path diversity. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 431–440, New York, NY, USA, 2001. ACM.

[39] Vern Paxson. End-to-end internet packet dynamics. *IEEE/ACM Trans. Netw.*, 7(3):277–292, 1999.

[40] David X. Wei, Pei Cao, and Steven H. Low. Packet Loss Burstiness: Measurements and Implications for Distributed Applications. *Parallel and Distributed Processing Symposium, International*, 0:222, 2007.

[41] Y.J. Liang, J.G. Apostolopoulos, and B. Girod. Analysis of packet loss for compressed video: does burst-length matter? In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 5, pages V–684–7 vol.5, April 2003.

[42] V. Hardman, M.A. Sasse, and A. Watson. Reliable audio for use over the Internet. In *INET'95 - the 5th Annual Conference of the Internet Society*, 1995.

[43] S. Floyd, T. Henderson, et al. RFC 3782: The NewReno Modification to TCP's Fast Recovery Algorithm, IETF.

[44] http://bittorrent.org/beps/bep_0029.html.

[45] Min Zhang, Maurizio Dusi, Wolfgang John, and Changjia Chen. Analysis of udp traffic usage on internet backbone links. In *SAINT '09: Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet*, pages 280–281, Washington, DC, USA, 2009. IEEE Computer Society.

[46] Bovy, CJ and Mertodimedjo, HT and Hooghiemstra, G. and Uijterwaal, H. and Van Mieghem, P. Analysis of end-to-end delay measurements in Internet. In *Fourth Passive and Active Measurements Workshop (PAM '02)*, 2002.

[47] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot. Packet-level traffic measurements from the Sprint IP backbone. *Network, IEEE*, 17(6):6–16, Nov.-Dec. 2003.

[48] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational backbone network. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 535–544, June 2002.

[49] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot. Measurement and analysis of single-hop delay on an ip backbone network. *Selected Areas in Communications, IEEE Journal on*, 21(6):908–921, Aug. 2003.

[50] K. Mochalski, J. Micheel, S. Donnelly, and N.Z. Hamilton. Packet delay and loss at the Auckland Internet access path. In *Fourth Passive and Active Measurements Workshop (PAM '02)*, 2002.

[51] M.S. Borella, S. Uludag, G.B. Brewster, and I. Sidhu. Self-similarity of Internet packet delay. In *Communications, 1997. ICC 97 Montreal, 'Towards the Knowledge Millennium'. 1997 IEEE International Conference on*, volume 1, pages 513–517 vol.1, Jun 1997.

[52] Li Qiong and D.L. Mills. On the long-range dependence of packet round-trip delays in Internet. In *Communications, 1998. ICC 98. Conference Record.1998 IEEE International Conference on*, volume 2, pages 1185–1191 vol.2, Jun 1998.

[53] KC Kang, K. Nam, DJ Jeon, and SY Kim. Delay characteristics of high-speed Internet access network: One case study. In *Asia-Pacific Network Operations and Management Symposium (APNOMS)*, volume 3, 2003.

[54] Wenyu Jiang and Henning Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *PROCEEDINGS OF NOSSDAV '2000*, 2000.

[55] H. Sanneck and G. Carle. A framework model for packet loss metrics based on loss runlengths. In *SPIE/ACM SIGMM Multimedia Computing and Networking Conference*, pages 177–187, 2000.

[56] E. Gustafsson and G. Karlsson. A literature survey on traffic dispersion. *Network, IEEE*, 11(2):28–36, Mar/Apr 1997.

[57] Shu Tao, John Apostolopoulos, and Roch Guérin. Real-time monitoring of video quality in ip networks. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 129–134, New York, NY, USA, 2005. ACM.

[58] A. R. Reibman and V. Vaishampayan. Quality monitoring for compressed video subjected to packet loss. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 17–20, Washington, DC, USA, 2003. IEEE Computer Society.

[59] Jiayue He and J. Rexford. Toward internet-wide multipath routing. *Network, IEEE*, 22(2):16–21, March-April 2008.

[60] C. Cetinkaya and E.W. Knightly. Opportunistic traffic scheduling over multiple network paths. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1928–1937 vol.3, March 2004.

[61] T. Nguyen, P. Mehra, and A. Zakhor. Path diversity and bandwidth allocation for multimedia streaming. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 1–4, Washington, DC, USA, 2003. IEEE Computer Society.

[62] J.G. Apostolopoulos and M.D. Trott. Path diversity for enhanced media streaming. *Communications Magazine, IEEE*, 42(8):80–87, Aug. 2004.

[63] Shu Tao and Roch Guérin. Application-specific path switching: a case study for streaming video. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 136–143, New York, NY, USA, 2004. ACM.

[64] Shu Tao, Kuai Xu, Ying Xu, Teng Fei, Lixin Gao, Roch Guerin, Jim Kurose, Don Towsley, and Zhi-Li Zhang. Exploring the performance benefits of end-to-end path switching. *SIGMETRICS Perform. Eval. Rev.*, 32(1):418–419, 2004.

[65] L. Golubchik, J. C. S. Lui, T. F. Tung, A. L. H. Chow, W.-J. Lee, G. Franceschinis, and C. Anglano. Multi-path continuous media streaming: what are the benefits? *Perform. Eval.*, 49(1-4):429–449, 2002.

[66] Evangelos Vergetis, Eric Pierce, Marc Blanco, and Roch Guérin. Packet-level diversity - from theory to practice: an 802.11-based experimental investigation. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 62–73, New York, NY, USA, 2006. ACM.

[67] S.-J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 10, pages 3201–3205 vol.10, 2001.

[68] A. Tsirigos and Z.J. Haas. Multipath routing in the presence of frequent topological changes. *Communications Magazine, IEEE*, 39(11):132–138, Nov 2001.

[69] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Improved data distribution for multipath tcp communication. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 1, pages 5 pp.–, Nov.-2 Dec. 2005.

[70] Janardhan R. Iyengar, Paul D. Amer, and Randall Stewart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *IEEE/ACM Trans. Netw.*, 14(5):951–964, 2006.

[71] C. Perkins, O. Hodson, and V. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12:40–48, Sep/Oct 1998.

[72] K. Stuhlmuller, N. Farber, M. Link, and B. Girod. Analysis of video transmission over lossy channels. *Selected Areas in Communications, IEEE Journal on*, 18(6):1012–1032, Jun 2000.

[73] M. Zorzi and R.R. Rao. On the statistics of block errors in bursty channels. *Communications, IEEE Transactions on*, 45(6):660–667, Jun 1997.

[74] E.N. Gilbert et al. Capacity of a burst-noise channel. *Bell Syst. Tech. J*, 39(9):1253–1265, 1960.

[75] EO Elliott. Estimates of error rates for codes on burst-noise channels. *Bell Syst. Tech. J*, 42(9):1977–1997, 1963.

[76] Ling-Jyh Chen, T. Sun, M.Y. Sanadidi, and M. Gerla. Improving wireless link throughput via interleaved FEC. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 539–544 Vol.1, June-1 July 2004.

[77] Mark Claypool and Yali Zhu. Using Interleaving to Ameliorate the Effects of Packet Loss in a Video Stream. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 508, Washington, DC, USA, 2003. IEEE Computer Society.

[78] Jin Young Lee and Hayder Radha. Interleaved source coding (ISC) for predictive video coded frames over the Internet. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 2, pages 1224–1228 Vol. 2, May 2005.

[79] Jin Young Lee and H. Radha. Evaluation of Interleaved Source Coding (ISC) over Channel with Memory. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 1152–1157, March 2006.

[80] Suk Kim Chin and R. Braun. Improving video quality using packet interleaving, randomisation and redundancy. In *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, pages 405–413, 2001.

[81] B.W. Web and Ding Lin. Transformation-based reconstruction for real-time voice transmissions over the internet. *Multimedia, IEEE Transactions on*, 1(4):342–351, Dec 1999.

[82] Jianfei Cai and Chang Wen Chen. Fec-based video streaming over packet loss networks with pre-interleaving. In *Information Technology: Coding and Computing, 2001. Proceedings. International Conference on*, pages 10–14, Apr 2001.

[83] Steven Gringeri, Roman Egorov, Khaled Shuaib, Arianne Lewis, and Bert Basch. Robust compression and transmission of mpeg-4 video. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 113–120, New York, NY, USA, 1999. ACM.

[84] P. Salvo Rossi, F. Palmieri, G. Iannello, and A.P. Petropulu. Packet interleaving over lossy channels. In *Signal Processing and Information Technology, 2004. Proceedings of the Fourth IEEE International Symposium on*, pages 476–479, Dec. 2004.

[85] Jey-Hsin Yao and Yao-Min Chen. Experiments of Real-Time MPEG Audio over the Internet. *Fujitsu Scientific and Technical Journal*, 33(2):138–144, 1997.

[86] Yi J. Liang, John G. Apostolopoulos, and Bernd Girod. Model-Based Delay-Distortion Optimization. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, 2002.

[87] S. Wee, Wai tian Tan, J. Apostolopoulos, and M. Etoh. Optimized video streaming for networks with varying delay. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 89–92 vol.2, 2002.

[88] M. Kalman, E. Steinbach, and B. Girod. R-D optimized media streaming enhanced with adaptive media playout. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 869–872 vol.1, 2002.

[89] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. Do you trust your software-based traffic generator? *IEEE Communications Magazine*, In press.

[90] http://www.grid.unina.it/Traffic.

[91] M.E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking (TON)*, 5(6):835–846, 1997.

[92] iperf. http://sourceforge.net/projects/iperf as of November 2009.

[93] mgen. http://pf.itd.nrl.navy.mil/mgen/mgen.html as of November 2009.

[94] Intel ixp4xx network processors. http://www.intel.com/design/network/products/npfamily/ixp4xx.htm as of November 2009.

[95] Alessio Botta, Walter de Donato, Antonio Pescapé, and Giorgio Ventre. Networked Embedded Systems: a Quantitative Performance Comparison. In *Global Telecommunications Conference, 2008. GLOBECOM '08. IEEE*, 2008.

[96] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):00–00, July 2003.

[97] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes In Computer Science*, 2218:329–350, 2001.

[98] L. Subramanian, I. Stoica, H. Balakrishnan, and R.H. Katz. OverQoS: offering Internet QoS using overlays. *ACM SIGCOMM Computer Communication Review*, 33(1):11–16, 2003.

[99] Limin Wang, Vivek Pai, and Larry Peterson. The effectiveness of request redirection on CDN robustness. *SIGOPS Oper. Syst. Rev.*, 36(SI):345–360, 2002.

[100] R. Sherwood and N. Spring. Touring the internet in a TCP sidecar. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 339–344. ACM New York, NY, USA, 2006.

[101] Canonico, Roberto and D'Antonio, Salvatore and Ventre Giorgio. Extending the planetlab usage model for distributed heterogeneous research testbeds. In *Paper presented at the 2nd International Workshop on Real Overlays & Distributed Systems, ROADS'07*, Warsaw, Poland, July 2007.

[102] Himabindu Pucha, Charlie Hu, and Morley Mao. On the impact of research network based testbeds on wide-area experiments. In *Proceedings of IMC '06, the 6th ACM SIGCOMM conference on Internet measurement*, Rio de Janeiro, Brazil, December 2006.

[103] S. Banerjee, T.G. Griffin, and M. Pias. The Interdomain Connectivity of PlanetLab Nodes. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 73–82, 2004.

[104] J. Ledlie, P. Gardner, and M. Seltzer. Network Coordinates in the Wild. In *NSDI, Cambridge, MA, April*, 2007.

[105] Onelab European research project. http://www.onelab.eu as of November 2009.

[106] Alessio Botta, Roberto Canonico, Giovanni Di Stasi, Antonio Pescapé, Giorgio Ventre, and Serge Fdida. Integration of 3G connectivity into a PlanetLab based testbed - A step of an evolutionary path towards heterogeneous large scale network testbeds. Under review.

[107] R.P. Karrer, I. Matyasovszki, A. Botta, and A. Pescapé. MagNets - experiences from deploying a joint research-operational next-generation wireless access network testbed. In *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*, pages 1–10, May 2007.

[108] R.P. Karrer, A. Botta, and A. Pescapé. High-speed backhaul networks: Myth or reality ? *Computer Communications*, 31(8):1540–1550, 2008.

[109] A. Botta, A. Pescapé, and R.P. Karrer. *Wireless Networks Test-beds: when heterogeneity plays with us*, chapter in Heterogeneous Wireless Access Networks, Architectures and Protocols. Springer, 2008.

[110] The Tolly Group. Network processors: Performace analysis of 802.11 broadband routers and access point, 2004. http://www.intel.com/design/network/products/npfamily/tolly_204131.pdf.

[111] Alessio Botta, Antonio Pescapé, and Giorgio Ventre. An approach to the identification of network elements composing heterogeneous end-to-end paths. *Computer Networks*, 52(15):2975–2987, 2008.

[112] Steinmetz, Ralf. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):61–72, Jan 1996.

[113] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi. Tracking Down Skype Traffic. In *IEEE INFOCOM 2008. The 27th Conference on Computer Communications.*, pages 261–265, April 2008.

[114] A. Barbuzzi, F. Ricciato, and G. Boggia. Discovering Parameter Setting in 3G Networks via Active Measurements. *Communications Letters, IEEE*, 12(10):730–732, October 2008.

[115] V. Gambiroza and et al. End-to-end performance and fairness in multihop wireless backhaul networks. In *IEEE MobiCom*, Philadelphia, PA, September 2004.

[116] W. Willinger and M. Garrett. Analysis, modeling, and generation of self similar VBR video traffic. In *ACM SIGCOMM: conference on applications, technologies, architectures, and protocols for computer communications*, London, England, August 1994.

[117] D.W. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979.

[118] G. Iannello, F. Palmieri, A. Pescape, and P. Salvo Rossi. End-to-end packet-channel bayesian model applied to heterogeneous wireless networks. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 1, pages 6 pp.–, Dec. 2005.

[119] Francesco Vacirca, Fabio Ricciato, and Rene' Pilz. Large-Scale RTT Measurements from an Operational UMTS/GPRS Network. In *International Conference on Wireless Internet (WICON)*, 2005.

[120] S. Rugel. On the performance of optimization proxies for data services in mobile networks. In *46th FITCE Congress*, 2007.

[121] Hao Jiang and Constantinos Dovrolis. Passive estimation of TCP round-trip times. *SIGCOMM Comput. Commun. Rev.*, 32(3):75–88, 2002.

[122] Angela Wang, Cheng Huang, Jin Li, and Keith W. Ross. Queen: Estimating Packet Loss Rate between Arbitrary Internet Hosts. In *Passive and Active Measurements Conference*, Lecture Notes in Computer Science, 2009.

[123] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Improving accuracy in end-to-end packet loss measurement. In *ACM SIGCOMM: conference on applications, technologies, architectures, and protocols for computer communications*, 2005.

[124] B. Ribeiro, E.S. e Silva, and D. Towsley. On the efficiency of path diversity for continuous media applications. *UMass CMPSCI Technical Report 05-19*, 2005.

[125] D. Jurca, S. Petrovic, and P. Frossard. Media aware routing in large scale networks with overlay. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 892–895, July 2005.

[126] E. Vergetis, R. Guérin, and S. Sarkar. Improving Performance Through Channel Diversity in the Presence of Bursty Losses. In *Proc. ITC-19*, Beijing, China, August 2005.

[127] Vinh Bui, Weiping Zhu, Alessio Botta, and Antonio Pescapé. An MDP-based Approach for Multipath Data Transmission over Wireless Networks. In *Communications, 2008. ICC 2008. IEEE International Conference on*, May 2008.

[128] Vinh Bui, Weiping Zhu, Alessio Botta, and Antonio Pescapé. A Markovian Approach to Multi-path Data Transfer in Overlay Networks. Under review.

[129] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski. Modeling internet backbone traffic at the flow level. *Signal Processing, IEEE Transactions on*, 51(8):2111–2124, Aug. 2003.

[130] Almudena Konrad and Anthony D. Joseph. Choosing an accurate network path model. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 314–315, New York, NY, USA, 2003. ACM.

[131] Eitan Altman. Applications of Markov Decision Processes in Communication Networks. In *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer, 2002.

[132] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary poisson view of internet traffic. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1558–1569 vol.3, March 2004.

[133] Yin Zhang and Nick Duffield. On the constancy of internet path properties. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 197–211, New York, NY, USA, 2001. ACM.

[134] Long Vu, Indranil Gupta, Jin Liang, and Klara Nahrstedt. Measurement and modeling of a large-scale overlay for multimedia streaming. In *QSHINE '07: The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness & Workshops*, pages 1–7, New York, NY, USA, 2007. ACM.

[135] D. Jurca and P. Frossard. Video packet selection and scheduling for multipath streaming. *Multimedia, IEEE Transactions on*, 9(3):629–641, April 2007.

[136] X. Liu, E.K.P. Chong, and N.B. Shroff. Opportunistic transmission scheduling with resource-sharing constraints in wireless networks. *IEEE Journal on Selected Areas in Communications*, 19(10):2053–2064, 2001.

[137] Leonard Kleinrock. *Queueing Systems Volume 1: Theory*. John Wiley & Son, New York, USA, 1975.

[138] D. R. Cox. *Renewal Theory*. Methuen & Co., London, GB, 1962.

[139] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, USA, 1994.

[140] John Little. A proof of the queueing formula $l = \lambda w$. *Operations Research*, 9:383–387, 1961.

[141] Shane P. Pederson and Mark E. Johnson. Estimating model discrepancy. *Technometrics*, 32(3):305–314, 1990.

[142] Joseph Y-T. Leung. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, USA, 2004.

[143] Bing Wang, Wei Wei, Jim Kurose, Don Towsley, Krishna R. Pattipati, Zheng Guo, and Zheng Peng. Application-layer multipath data transfer via TCP: Schemes and performance tradeoffs. *Performance Evaluation*, 64(9-12):965 – 977, 2007.

[144] David B. Johnson, David A. Maltz, and Josh Broch. Dsr: the dynamic source routing protocol for multihop wireless ad hoc networks. *Ad hoc networking*, pages 139–172, 2001.

[145] Coleman Rodney. *Stochastic Processes*. Allen & Unwin Ltd, London, 1974.

[146] S.B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 227–234 vol.1, Mar 1999.

[147] Mark W. Garrett and Walter Willinger. Analysis, modeling and generation of self-similar vbr video traffic. *SIGCOMM Comput. Commun. Rev.*, 24(4):269–280, 1994.

[148] Traffic identification engine. http://tie.comics.unina.it as of November 2009.

[149] Eitan Altman, Konstantin Avrachenkov, and Chadi Barakat. Tcp in presence of bursty losses. *Perform. Eval.*, 42(2-3):129–147, 2000.

[150] Pasi Sarolahti and Alexey Kuznetsov. Congestion control in linux tcp. In *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*, pages 49–62, Berkeley, CA, USA, 2002. USENIX Association.