Computer Network Emulation for Quality of Experience Assessment

Mihai Ivanovici

Editura Universităţii Transilvania din Braşov

Computer Network Emulation for Quality of Experience Assessment

Mihai Ivanovici
Transilvania University of Braşov, România

Editura Universităţii Transilvania din Braşov

2015

EDITURA UNIVERSITĂŢII *TRANSILVANIA* DIN BRAŞOV

Adresa: 500091 Braşov,
B-dul Iuliu Maniu 41A
Tel: 0268 476 050
Fax: 0268 476 051
E-mail : editura@unitbv.ro

Editură acreditată de CNCSIS
Adresa nr.1615 din 29 mai 2002

Referenţi ştiinţifici:

Prof. univ. dr. ing. Gheorghe TOACŞE

Prof. univ. dr. ing. Mihai CIUC

Ilustraţie copertă: Alexandra Liana Stănescu

Design copertă: Vlad Andrei Nica

Descrierea CIP a Bibliotecii Naţionale a României
Ivanovici, Mihai
Computer network emulation for quality of experience assessment / Mihai Ivanovici. - Braşov : Editura Universităţii "Transilvania", 2015
Bibliogr.
ISBN 978-606-19-0586-7

004

Contents

Foreword

This book is mainly based on the PhD thesis entitled *Network Quality Degradation Emulation—An FPGA-based Approach to Application Performance Assessment* publicly defended in on January 26th, 2006 at Politehnica University of Bucharest, Romania. During the research that took place place at CERN, the European Organization for Nuclear Research, Geneva, Switzerland, rougly between 2002 and the end of 2005, we proposed the term of *user-perceived quality* in our research publications and tried to impose it at international level, but finally the term *Quality of Experience* was consacrated. The last two chapters of this book are based on the research carried out at Transilvania University of Braşov, Romania, between 2006 and 2010.

The publication of this book in 2015 celebrates an important achievment for the Department of Electronics and Computers within the Transilvania University of Braşov, România: due to our implication to the ATLAS experiment at CERN, Geneva, Switzerland, our contribution was acknowledged by placing our university on the list of collaborating institutes. This official position places the research in the field of FPGA and ASIC design, as well as PCB design, performed in the forementioned department, on the international map of electronics for particle physics.

The outline of this book is the following: the existing network emulators are presented in Chapter 2. Their architecture is briefly described, along with their features. The approaches are analyzed and their disadvantages are emphasized. Chapter 3 states a set of principles that are the basic guidelines we propose for the emulation of realistic network conditions. The two main approaches that can be taken for the emulation of the degradation that occurs in computer networks are presented. Chapter 4 proposes the mathematical models and tools to be used to achieve the goals presented in the previous chapters. Their consistence with classical queueing theory is demonstrated. Then the hardware platform we used for development of our network emulator and the emulator architecture are described in Chapter

5. Experimental results are shown in Chapter 6. Three applications are discussed: web browsing, voice over IP and video streaming. The reader is invited to prior or further read the book by Răzvan Beuran, entitled *Introduction to Network Emulation*, Pan Stanford Publishing, 2013.

I would like to express my gratitude to my mentor, prof. Vasile Buzuloiu, supervisor of my PhD thesis, for guiding me on my scientific path. My special thanks go to Bob Dobinson and Brian Martin who acquainted me in their ATLAS Networking group at CERN and conducted my research. I wish to thank Neil Davies from Predictable Network Solutions, UK, for his continuous support and involvement in the work carried out at CERN. I would like to thank Răzvan Beuran, not only for sharing the same office, but for both the scientific and philosophical discussions we had during our stay at CERN, as well for his contributions to the development of the network emulator. Special thanks to Matei Ciobotaru, who implemented the low-level library providing the primitives for the memory and PCI access. Many thanks to all those who read the original PhD thesis: their pertinent comments and suggestions helped me refine it. Last but not least, many thanks to Ştefan Savu for his contribution on VoIP QoE assessment.

Mihai Ivanovici, Braşov, România, 2015

Chapter 1

Introduction

The Internet grows and becomes more complex every day. The complexity of the Internet is not only determined by its dimensions, but also by its heterogeneity. Like other complex systems[1], networks exhibit a highly unpredictable behaviour. They are non-linear systems, therefore the overall behaviour is not just the simple sum of the individual network components, as one may expect. Complex non-linear systems are difficult to predict since they exhibit qualitatively different behaviours at different times.

The Internet is *best effort*. One may think that best effort refers to a irreproachable operation, but in fact it says that nothing is guaranteed and the Internet *does its best* to ensure the connectivity and deliver the application traffic between two end points. Networks are degrading environments. They perturb application behavior by delaying and sometimes even dropping the application traffic. In an ideal network, applications should only experience a minimum quality degradation, caused by the propagation delay. In real-world networks things get worse, as application traffic experiences, in addition to the fixed propagation delay, a variable delay due to queueing and the presence of other traffic, and, potentially, loss. The delay and loss are the consequences of the competition between traffic flows for the available network resources. We denote the increased loss and delay experienced by an application by the term *quality degradation*. The quality degradation in the network is reflected in the performance degradation at application level.

As network applications become more and more demanding, the need to be able to quantify and predict the behaviour of computer networks comes to light. Mission or life-critical applications simply cannot rely on the current

[1] Many natural phenomena may be called "Complex systems", and "Complexity science" is highly interdisciplinary. Examples of complex systems include ant hills, ants themselves, human economies, nervous systems, cells and so forth.

Internet. First of all, one must understand application requirements and quantify them in a precise manner [39]. In case the requirements are not met, the deviation from the ideal behaviour should be assessed and the implications analyzed. The next step would be to reconfigure or redesign the network in such a way that it will ensure the desired quality level for the applications. The alternative choice would be to redesign the application in order to make it more robust or more efficient. These two approaches can be combined in order to meet the application requirements.

To refine network applications or protocols, and make them more appropriate for the real-life deployment, one can use an iterative approach: design, implementation, validation. Starting from the initial design or specifications, the application is implemented. The next step is to validate the implementation in a real network scenario or in a laboratory experimental setup. Based on the observed application behaviour, the design stage can be revisited. Using this methodology, applications would be guaranteed to work in a certain range of network conditions. However, network conditions can be very aggressive and even the most robust applications may fail. Guaranteeing bounds on the quality degradation in networks could be a solution to the unpredictable behaviour of networks.

Understanding how application performance depends on network quality degradation is a preliminary step in understanding the behaviour of networks, as well as the behaviour of the network applications. Knowing the relationship between the user-perceived quality—as a measure of application performance—and the quality at network level makes it possible to: (i) predict how an application would behave in given network conditions; and (ii) design the networks in such ways that the applications will perform according to user requirements or expectations.

From the user perspective, quality of experience (QoE) or quality degradation is perceived as an unsatisfactory application behaviour, with respect to expectations. Depending on application, quality degradation may mean an increased time-to-load for a Web page, an increased time-to-complete for a file transfer, or a poor quality of the speech signal for a Voice-over-IP call. From the network perspective, any excessive quality degradation[2] indicates insufficient resources, or ineffective management of the available ones. Quantifying user needs—translated into application minimum expectations—and determining the circumstances under which applications fail are important steps in understanding the application behaviour.

[2]As mentioned before, one should expect at least a minimum quality degradation, due to the propagation delay.

1.1 Quality of Experience

The applications drive the development of networks. The need for transferring huge amounts of data across long-haul connections, or the need for wireless ad-hoc connections, as well as the ordinary e-mail or browsing require continuous refining of networking technologies. New standards and protocols allow for more reliable and faster data handling. However, the user is the only one who can say if data is transferred fast enough or the application behaves the way it should.

In what follows we shall illustrate how user expectations become requirements for a certain application. Let us consider the case of a very simple electrical circuit comprising a light bulb, a power source and a switch. When switching on the circuit, the light bulb will illuminate. Everyone expects this to happen in a blink of an eye. It takes longer for a neon light to turn on, and definitely much longer to switch on the public lights or the illumination system of a football stadium. The person that presses the switch surely expects the light to go on. No matter what the user expects, there will be a delay between the moment the main switch is pressed and the moment when the lights will be on.

Usually everyone's *desire* is that the light go on instantaneously. This is still an *expectation*. A *requirement* is an expectation which is expressed within a time constraint: if light goes on, I want it to happen in less than 1 second. When browsing the Internet, the user expects a certain web page to be loaded in a browser, unless the page is unavailable. It's desirable that this happens in a couple of seconds. If it takes longer than 10 seconds, the page may no longer be of interest. In this case, the requirement is for the page to load in less than 10 seconds.

1.2 Assessing QoE

There are three steps to take in order to assess application performance: (i) observe the application behavior at the end-node level, (ii) accurately measure the quality degradation experienced by the application traffic and (iii) correlate the above. Scientific method requires the use of objective metrics to perform both the network and application level performance assessments.

First of all, one must *observe* the application outcome. A human observer could judge if the application behaved as expected, or objective metrics can be used [72], [37]. At application level the user is unaware of what is happening at network level. Moreover he should not care about the underlying mechanisms. However, the performance of the application and implicitly

the quality of experience strongly depends on network performance. Therefore is mandatory to observe the network conditions or the network quality degradation between the two end points (see Figure 1.1).

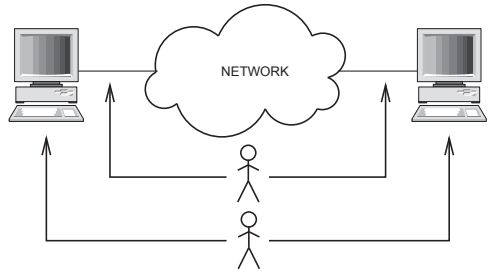


Figure 1.1: Observing both application QoE and network QoS.

Observing is not enough. Accurate *quantification* of both the application performance (QoE) and the network conditions (QoS) should be performed. The following step is to correlate the two results, thus experimentally determining the relationship between the application performance and the quality degradation at network level.

This second step implies defining the application outcome as well as a metric to allow quantify the application performance. The metric can be either *objective* or *subjective*. Then the appropriate method to measure the application outcome should be chosen. Once the method and the outcomes are well defined, the application outcome must be accurately quantified.

In case of network quality degradation there are various widely used metrics [45], [46]: one-way delay [47], one-way packet loss [48] and throughput. However, when application performance must be determined, each application class requires the definition of specific metrics that take into account the application nature. For example, for Voice over IP (VoIP) one can use the Perceptual Evaluation of Speech Quality score [72]. In case of file transfer, useful metrics are transfer time performance and goodput [37].

There are two traditional methods for testing and validating network applications and protocols: (i) *simulation*, i.e., run a model or a representation of the application's code in a completely synthetic environment; and (ii) *real network testing*, i.e., run the application in a real environment.

Simulation has several advantages: usually it is cheap (from the point of view of the resources involved), and it works for large-scale tests if sufficient computing resources are available. Simulation has the great advantage of being controllable, thus experiments are reproducible. One problem related to simulation is the need to rewrite the code in order to match to

the simulation environment, therefore the simulation implementation of the applications may differ from the real one. However, a simulation is only an approximation of the real life, because both applications and networks are mathematically modeled.

In real environments, both applications and network are real, therefore any discrepancy between the application or network behaviour and its model is eliminated. So the most important advantage of real network testing is the fact that the real application is running in a real environment. The major disadvantage is the cost: it is very expensive to create a real test environment, even on a small scale. The tests in a real environment may not to be reproducible, therefore possible observed problems are more difficult to solve. Also the range of network conditions in which applications can be studied is limited. The limitation comes from the impossibility of controlling the other traffic flows that share the same network as the traffic of the application of interest.

Computer network emulation is a hybrid technique between simulation and real network testing, that allows the study of real network applications in a laboratory setup. The application behaviour can be studied in a wide range of network conditions.

1.3 Computer network emulation

An emulator can be seen as a *black box*, which behaves as one particular network for the external observer. The goal is to observe the same application QoE both when using the computer network emulator and the real network. The black box becomes a *network in a box*, as illustrated in Figure 1.2

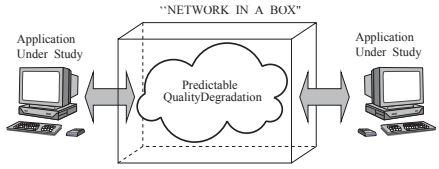


Figure 1.2: Network in a box.

We embraced the computer network emulation technique as the most appropriate one for application performance assessment. It allowed us to study the application behaviour in a wide range of network conditions, using a relatively cheap laboratory setup consisting mainly of off-the-shelf components [39]. Assessing the application performance requires a well-defined method-

ology [39]. The system we proposed is depicted in Figure 1.3 in a typical test setup.

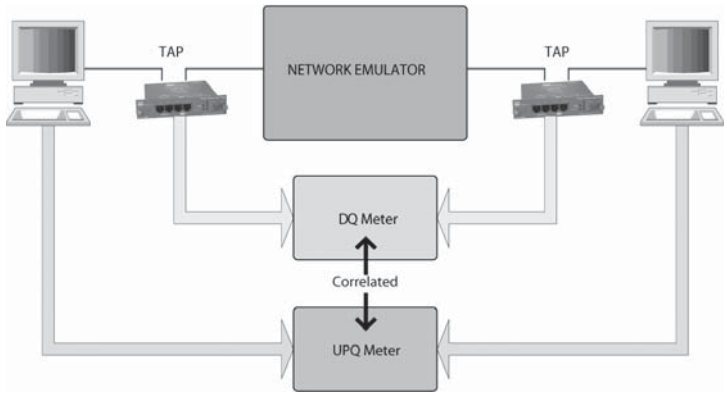


Figure 1.3: The setup for application QoE assessment.

This monitoring system allows the passive, non-intrusive measurement of the network quality and the user-perceived quality (UPQ) or quality of experience (QoE) for a certain application. The novelty of our monitoring system is that we are able to both accurately measure the network quality degradation and objectively assess application UPQ/QoE in parallel. This allowed us to quantify the relationship between the two for a certain network application and identify application's requirements.

Using Fast Ethernet taps we mirror the traffic on the link between two PCs that run the network application under study. This traffic is fed into programmable Alteon UTP network interface cards, which are hosted by two PCs (see the detailed setup depicted in Figure 1.4). From each packet all the information required for the computation of the network QoS parameters is extracted and stored in the local memory as packet descriptors. The host PCs, which control the programmable NICs, periodically collect this information and store it in descriptor files. This data is then used to compute off-line the following network parameters: one-way delay and jitter, packet loss and throughput. Based on the same descriptors, we can calculate instantaneous or average values, and various histograms. See [39] for details on the implementation and the capabilities of the monitoring system.

The monitoring system we designed made use of a software network emulator. From the plethora of existing network emulators we chose NIST Net [15]. It allowed us to implement the most cost effective system. However, given the limitations of the existing network emulators (see Chapter 2) we decided to implement our own network emulator.

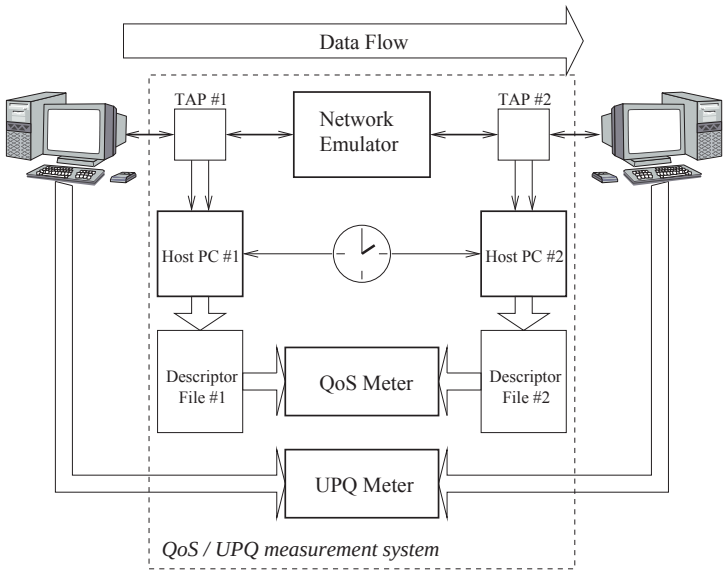


Figure 1.4: The detailed setup used to assess application QoE.

For a VoIP application we established the experimental relationship between the user-perceived quality (UPQ) and the networks degradation (see Figure 1.5). The PESQ score (Perceptual Evaluation of Speech Quality) [72] is an objective metric, proposed by International Telecommunication Union (ITU) to quantify the user satisfaction in the case of a telephone call.

According to [55] the relationship between PESQ scores and audio quality is the following: PESQ scores between 3 and 4.5 mean acceptable perceived quality, with 3.8 being the PSTN[3] threshold this will be termed as good quality; values between 2 and 3 indicate that effort is required for understanding the meaning of the voice signal this will be named low quality; scores less than 2 signify that the degradation rendered the communication impossible, therefore the quality is unacceptable.

One can establish that for loss rates up to 4% the influence is noticeable, but quality remains *good* for low jitter values; quality becomes *low* for higher loss rates, up to 15%. If jitter is high then loss doesn't change significantly the perceived quality which moreover is *unacceptable*. This is due to the high level of induced dejittering loss [38].

In order to have a better insight in the location of the limits of good-to-low and low-to- unacceptable quality, horizontal cross-sections of the surface in Figure 1.5 at the PESQ score levels corresponding to these limits (PESQ

[3]Public Switched Telephone Network.

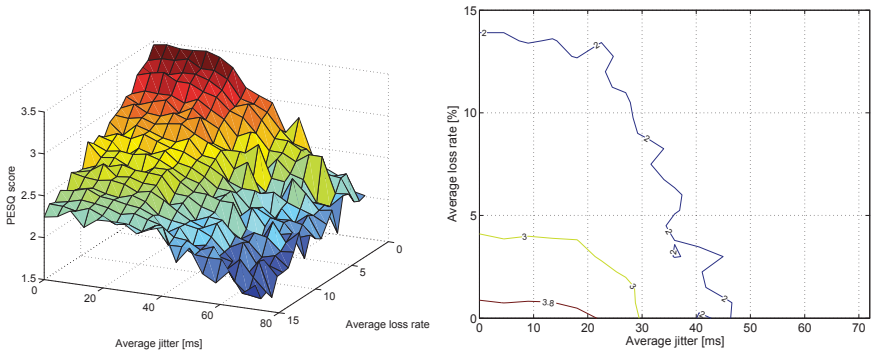


Figure 1.5: UPQ/QoE vs. network quality degradation for a VoIP call using the G.711 codec (surface and contour plot).

score values equal to 3 and 2) were made. For the G.711 codec [56] one can also see the boundary of excellent-to-good quality, at a PESQ score equal to 3.8. This is the only codec we studied for which quality can be also excellent: the corresponding area lies roughly within 0 to 1% loss rate and 0 to 20 ms jitter.

Once that the dependency of the user-perceived quality on network quality degradation is determined, two main tasks can be performed: (i) *predict* how the application will perform in given network conditions; and (ii) *design* or configure the network so that the applications will run at a specific performance level or will deliver the required user-perceived quality or QoE.

To emulate means "to reproduce the action of or behave like (a different type of computer)" [49]. When emulating networks, the emulation refers to the ability of reproducing the effects induced by a live network, as they are perceived by the application traffic. Our goal is to emulate the end-to-end quality degradation likely to appear in large networks, so that same loss in quality has the same effects on application behaviour. The effects on application behaviour are to be judged based on the observable outcomes defined for that particular application.

There are several levels of fidelity to which a network can be emulated. At the most concrete level, emulating a network element means first of all emulating its internal architecture. Basically this comes to emulating a series of queues and the switching fabric that interconnects them. Emulating network elements by reconstructing exact replicas can be very complex. As the system to be emulated increases, the growth of its complexity may require huge processing power and the time to simulate or emulate the system skyrockets. Therefore an equivalent model should be built, as an abstraction

of the entire system.

To emulate a complete end-to-end path, one could emulate the complexity of each of the individual components along that path. However, in this approach, there would be no hope of generalization and hence the extraction of principles on how applications and networks interact. We aim to emulate this interaction, and also the interaction between different traffic flows. The consequence of these interactions is the degradation in the network quality as it is experienced by the traffic flows. There are two possible scenarios:

1. the case when the traffic of interest (that we also call *foreground traffic*) is the only traffic flowing through the network. In this the interaction between the traffic of interest and the network is emulated,

2. the more realistic case is when other traffic flows – denoted by *background traffic* – share the same resources. The competition between different streams for the limited network resources leads to saturation, the prime cause of degradation in quality.

Network quality degradation can be characterized by two parameters: the loss and the delay. Intrinsically they are random variables, characterized by their probability density functions. They can be seen as two degrees of freedom, still highly correlated. We use these two coordinates to assess the degree of realism of our models. For a delay-insensitive application, different delay distributions can have the same effect on the application outcome, whereas the loss distributions can have completely different effects, given the specific packet loss pattern. E.g., for TCP/IP, it makes a significant difference which packets get lost (data or control packets) to the time-to-complete a data transfer. Also the loss in bursts (several packets are lost in a row) has a more dramatic impact to the TCP/IP rate, than if only one packet is lost, if the simplest acknowledgment method is deployed [57].

This books describes the approach we took for the emulation of the quality degradation likely to occur in computer networks. The framework we propose makes it possible to create an emulation tool, which permits the emulation of realistic network conditions. Mathematical models and computer simulation, as well as measurements in a testing environment are used to validate the real implementation on a hardware platform. The hardware platform we used is presented in Chapter 5.

The approach comprises three levels. At the theoretical level, mathematical models of the objects to be emulated are created. Their intrinsic properties are analyzed and their bounds are estimated. At the simulation level, the objects is simulated and their behaviour is observed. The outcome

is quantified and compared against the mathematical model.The first two levels allow us to extract the ideal behaviour that will be used as reference for the validation of the implementation. Then, a calibration of the real implementation follows. Given the constraints usually imposed by real implementations, the degradation introduced may differ from the expectation. A possible cause can be the fixed-point representation and the fact that time values are represented in quanta. Also the time spent to make decisions regarding the packet handling should be taken into account, as part as the intended delay.

There can be several possible levels of equivalence, depending on the degree of approximation. Identity can be the strongest level of equivalence, if the output of the theoretical model and the one of the real implementation match exactly. The two outputs could also match in terms of averages over arbitrarily-long intervals of time, or in terms of probability density functions. A measure that characterizes the difference between the two probability functions could be used to indicate the level of equivalence.

1.4 A taxonomy of computer networks

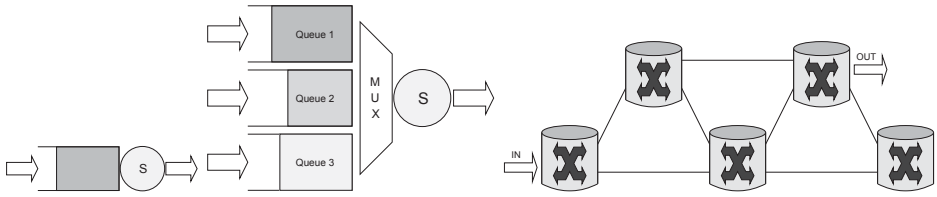
In this section we propose a taxonomy of networks and relate this to the ΔQ concept. Given that a network can be any system that interconnects two end-points, from a simple wire to the whole Internet, we can classify the networks into the following categories (refer also to Figure 1.6). Because of the quality degradation it introduces, any network is a quality degradation object. Each category of networks is characterized by the loss and delay they introduce.

***Order 0* networks.** At the most abstract level, the class of order 0 networks corresponds to an empty set of network elements or devices. The two end points are communicating through a *wire*. The wire can correspond to a UTP cable or an optical fiber. To emulate order 0 networks simply means to emulate the fixed delay caused by propagation through the communication channel and the loss caused by the bit error rate (BER) characteristic of the transmission medium.

***Order 1* networks.** This corresponds to a set comprising only one network element, that has only a queue or a FIFO. This implies emulating the variable delay caused by queueing and the loss caused by resource exhaustion. The *intra-stream contention* is therefore modeled, as the packets in a flow compete for the same resources.

***Order 2* networks.** In this case, the set is formed out of a network element comprising multiple queues. The switches and the routers are order 2 networks. These queues are serviced according to a scheduling mechanism, and may imply the use of traffic differentiation. One must emulate the *inter-stream contention* and the effects of different scheduling mechanisms.

***Order 3* networks.** This set includes multiple network elements of order 0, 1 and 2, and correspond to LANs, MANs or WANs. The quality degradation they induce correspond to the accumulation of multiple hop effects.



(a) order 1 (b) order 2 (c) order 3

Figure 1.6: The proposed taxonomy illustrated.

The "ΔQ" concept. Networks can be viewed as the sum of basic degradation elements. Every element induces delay and, potentially, loss. They degrade the experienced quality of the traffic flowing through them, i.e. they are quality degraders. Each basic element contributes with a certain amount—denoted by ΔQ—to the total amount of degradation along a network path. For every switch and router along that particular path, mathematical models can be built. Those models are thereafter aggregated in network models of the end-to-end quality degradation (see such an example in Figure 1.7). The aggregation process may be more complex than the simple addition of the per-hop effects. Depending on the complexity of switch and router models, the emulation of QoS mechanisms, like scheduling algorithms and traffic shaping techniques, can be performed as well.

The initial step in our approach is to first create emulations of simple building blocks, such as wires and queues, reproducing the quality degradation they introduce. In this view, a complex network is the composite of such basic components. Based on these simple models, one can synthesize an approximation of the behaviour of a complex network into one aggregate single model.

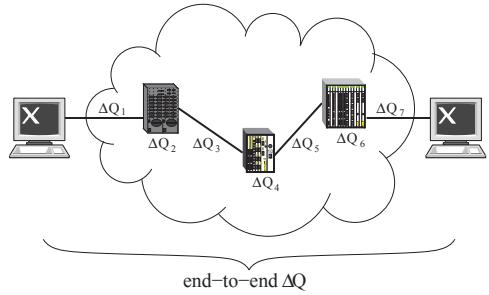


Figure 1.7: The end-to-end ΔQ.

The main goal of our approach was to create a tool that is able to emulate several network objects, like queues and wires, switches and wide-area networks. The *wires* allow the emulation of the fixed delay introduced by the propagation through transmission media. The emulator makes it possible to study, for example, how applications perform over large-delay connections, the consequences of using long optical fibers. The *queues* allow to study the simplest contention management mechanisms, like tail drop. No traffic differentiation applies, therefore all traffic flows get the same treatment. The scheduling mechanisms implemented in *switches* and routers, like Strict Priority or Weighted Round Robin, enforce differential treatment. Application traffic experiences different quality degradation based on its priority and the other traffic flows in the system. The *wide-area networks* are a collection of queues and wires, switches and routers. The quality degradation they introduce is the result of local quality degradations in each of these elements. Finding a single compact model of such a network, based on the aggregation of simple models, would be the ultimate goal of any emulation tool. Using this tool, we are able to assess application performance by experimentally establishing the relationship between the user-perceived quality and the experienced network quality degradation.

Chapter 2

Existing Computer Network Emulators

In this chapter the existing network emulators will be presented. Their features will be described and discussed. For some of them, the architecture is presented, and the concepts behind it, together with with their main advantages and disadvantages.

There exist several network emulators, in both hardware and software implementations. The hardware network emulators are usually available as commercial products, while the software network emulators are freely available. The hardware network emulators can be more precise, in the sense that the degradation they introduce (e.g., delay) is more accurate than in a case of a software implementation. Software network emulators can be installed and run on ordinary personal computers, which makes them very attractive for building cheap experimental setups.

Most of the existing network emulators are network architecture oriented, i.e. they emulate a specific network by deploying a complete architectural representation of it. The main disadvantage that emerges from this is the fact that as the complexity of the emulated network increases, its description or representation becomes larger and more difficult to deal with. There is no ability to abstract and to use a more compact and simple model of the network to be emulated.

2.1 Commercial Network Emulators

The description of the following commercial network emulators is solely based on the information available on Internet, which is mainly market oriented. Therefore their description may lack details on the internal archi-

tecture, the emulation concepts that are implemented or the mathematical models that are used.

`Anuē` network emulators are used to validate optical network operation during pre-deployment testing. It can precisely emulate signal delays and impairments that occur during Gigabit-rate transmission over optical fiber of other long-haul media. It supports several protocols, at data rates up to 10.7 Gbps. The `Anuē` network emulator is a *programmable spool of fiber* [9].

The following product versions are available: Sonet/SDH, Sonet/SDH Path Layer, Gigabit Ethernet, 10 Gigabit Ethernet, Fibre Channel and Combo Units, based on the type of the interfaces they are equipped with and the protocol they support. A functional `Anuē` system consists of a hardware platform and one or more software loads. All `Anuē` software loads include a Graphical User Interface (GUI) and TCL script interface, which allow for easy automation of common tasks.

The `Hammer PacketSphere` network emulator from Empirix [10] has a network processor architecture. It can introduce latency, jitter, packet loss, bandwidth limitations, bit errors, duplicate packets and packet reordering, at line speed on Fast Ethernet and Gigabit interfaces. Over 1.4 million packets per second (pps) can be processed per direction.

The latency and jitter can be constant or with uniform, normal, or random distribution. Packet loss be expressed in percentages, 1 packet out of N, or in size of a burst. Packet duplication and packet reordering can be also specified in percentages. Higher-level effects like congested router or packet re-route can be emulated. The two parameters to configure are the re-route rate and the re-route duration.

`PacketStorm 4XG` IP network emulator reproduces the unfavorable conditions of IP network in a controllable and repeatable laboratory setup. `PacketStorm 4XG`[1] can accommodate 16 Gigabit Ethernet ports and two 10 Gigabit Ethernet ports, its sustained packet rate being of over 64 million packets per second.

Its user interface allows to create the IP network to be emulated, by specifying the impairments for every link. The user can use Differentiated Serviced by specifying the priority of each application traffic.

[1] http://www.packetstorm.com/

`Shunra Virtual Enterprise.` Shunra[2] takes an empirical–rather than a purely mathematical–approach to network emulation. They believe that purely mathematical modeling is prone to error, because it assumes that the behaviour of networks can be accurately predicted. This assumption cannot stand, since in the real world, the networks and the applications often behave in unexpected ways. Network devices may not be ideally configured and software code may have bugs. The models may also require constant updating in order to fit the ever-changing network conditions.

Shunra's *empirical modeling* makes use of recorded conditions in the network to be emulated. The captured data is then used to define the parameters of the emulation, like latency, error rates and bandwidth utilization.

The `Virtual Enterprise` [19] from Shunra is based on StormAppliance and Cloud WAN emulation technology. It creates an exact replica of the network to be emulated, including end users and live application traffic. Figure 2.1 shows the architecture of the Shunra `Virtual Enterprise`.

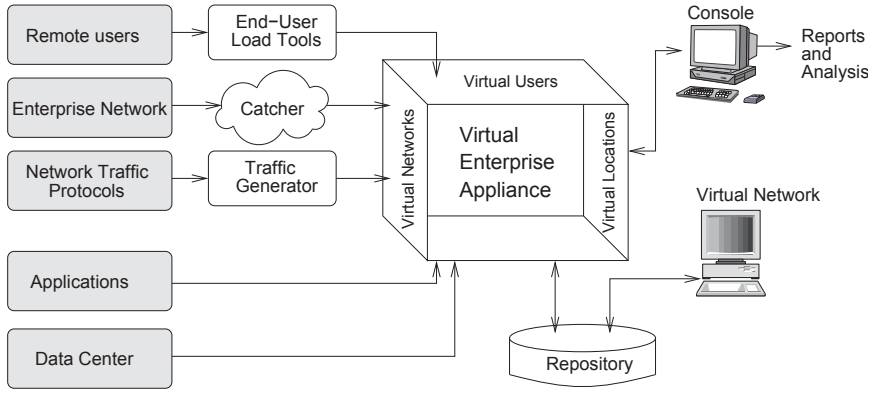


Figure 2.1: The Shunra Virtual Enterprise architecture.

At the heart of Shunra `Virtual Enterprise` is the Virtual Enterprise Appliance, functioning as a bridge or a router, that alters the speed at which traffic passes through and subjects packets to impairments likely to occur in wide-area networks. It can emulate symmetric and asymmetric links, and it can introduce latency, jitter and packet loss, as well as packet effects, such as duplication and fragmentation. The Virtual Enterprise Appliance can recreate complex network configurations, including different prioritizing schemes.

The "Console" controls the entire Shunra Virtual Enterprise through its XML/HTTP interface. This GUI, based on Microsoft Visio, allows you to

[2]http://www.shunra.com/

define the network to be emulated, by dragging and dropping components as clouds, endpoints, gateways and end-users. The "Catcher" records the network conditions, like latency and packet loss, for the network that is being monitored. The "End User Load Tool" and "Traffic Generators" act as application traffic generators, in order to produce the background traffic.

`Simena` . The network emulators produced by Simena [8] are available in three versions: NE2000, NE1000 and NE500, depending on the features implemented. The `Simena` network emulators work at Ethernet level, the packets being forwarded between the two network interfaces as soon as the unit is powered on. This makes them easy to use in "Plug & Play" ad-hoc setups.

The `Simena` network emulators offers packet filtering capabilities, in order to emulate different network impairments for different traffic flows. It can simultaneously perform up to 32 emulation tasks. The emulation can be either bi-directional or unidirectional. The quality degradation introduced can be: fixed, uniform distributed or normal distributed latency, fixed, dynamic or burst packet loss, bandwidth throttling, fixed or dynamic duplicate packets, random out-of-order packets. It can also emulate different network conditions like congestion, carrier loss or fragmentation.

`Simena` network emulators have a Web-based GUI, which allows users to remotely configure them from anywhere in the network. The real-time packet analysis permits decoding any packet flowing through.

2.2 Freely-available Network Emulators

The freely-available network emulators are implemented in software. They can be installed on ordinary PCs, which makes them a very attractive solution for building low-cost laboratory setups. They are usually designed to run on Unix-like operating systems, such as Linux or Free-BSD.

Another advantage of the software network emulators is the possibility of connecting them to a variety of end points, by simply using the appropriate network interface cards. This makes them able to be interconnected with a multitude of technologies, like copper or optical fiber Ethernet (Fast or Gigabit Ethernet), 622 Mbps OC-12 etc.

`Delayline` is an application-level emulation tool [1], implemented in C on a UNIX platform. It offers the possibility of describing the network to be emulated, by specifying a "map" (the network architecture) and the parameters for the links between the nodes of the network. This makes

Delayline an network-architecture-oriented emulation tool. The parameters that can be specified for each link are loss and delay.

`Delayline` intercepts application system calls, and replace them with its own system-like calls. Figure 2.2 shows how Delayline is integrated into an application, as an intermediate level between the user application and the operating system:

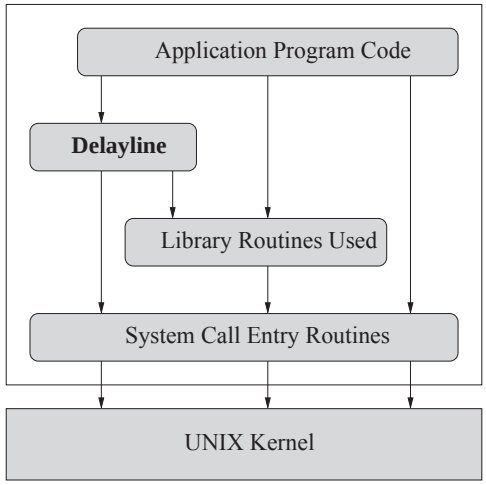


Figure 2.2: Integration of `Delayline` into an application program.

In order to use `Delayline`, one must recompile the network application which is under study, using a certain number of header files, and link it to the `Delayline` library provided. This is a major inconvenience, since usually the applications are only available in a binary format, therefore the user cannot recompile their source code. Because `Delayline` is designed to operate only with applications that use Berkeley sockets to communicate, the number of applications that can be studied using this emulator is therefore restricted.

In order to introduce quality degradation, one can specify the distribution (the probability density function) of the delay and the random loss percentage. However, these two parameters are independent random variables, so the quality degradation the application would experience is far from realistic. The precision of this tool is not satisfactory, given that the delay overhead that `Delayline` introduces for each intercepted message is of 500 μs [1].

`Dummynet` is a software-based emulation tool, that can be integrated into an existing protocol stack, allowing experiment to be run on a standalone

system. It is implemented on a FreeBSD operating system. `Dummynet` intercepts the communications of the protocol layer under test and can emulate the effects of finite queues, bandwidth limitations and communication delays [2].

`Dummynet` inserts a simple network model in an operational protocol stack, between the application and the network layer. Since `dummynet` introduces almost no overhead, tests can be run with rates up to the maximum operating speed allowed by the system in use [2].

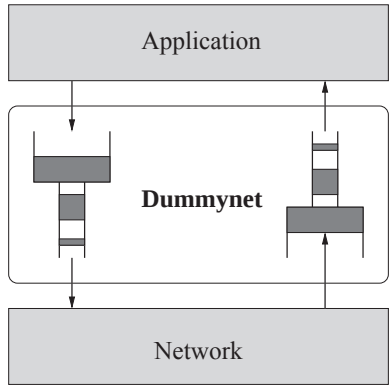


Figure 2.3: The principle of operation of `dummynet`.

The quality degradation that `dummynet` is able to introduce consists of random loss, fixed delay, bandwidth limitation and packet reordering.

`ENDE` [3] is a software-based emulation tool, that allows protocol testing on a single machine [3]. It can emulate end-to-end delays between two hosts by observing the status of a particular connection using ICMP packets. The basic architecture is presented in Figure 2.4.

`ENDE` consists of two single server queues with finite buffers, one queue for each direction. Application traffic flowing between the client and the server (see Figure 2.4) experiences loss and delay.

The delay introduced has two components: a fixed component due to propagation and a variable component due to queueing. The variable component is determined by the other traffic flows in the Internet. `ENDE` estimates the characteristics of the background traffic through ICMP probes.

`ENDE` intercepts the TCP and UDP packets sent to a specific port number and delays them with the value measured on the connection between the local and the remote host (see Figure 2.5). In parallel with the emulation,

[3] An End-to-end Network Delay Emulator

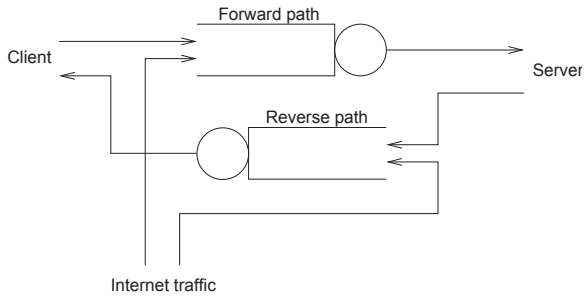


Figure 2.4: The internal architecture of ENDE.

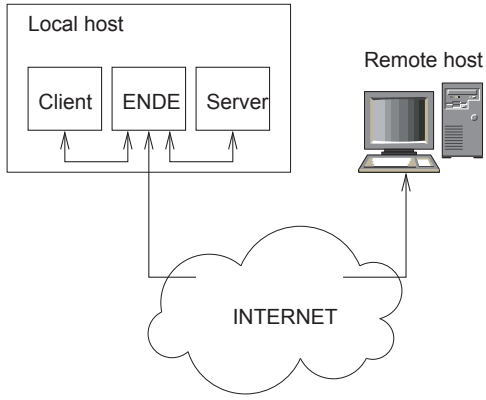


Figure 2.5: A typical setup for ENDE.

it sends ICMP packets at regular intervals in order to observe the current state of the emulated connection.

The main disadvantage of this emulator is the fact that experiments are not reproducible–they strongly depend on the network conditions of the connection which is emulated. The accuracy of the emulated delay depends on the clock resolution and on the accuracy of the ping measurements.

ENTRAPID is a protocol development environment [11], that can aid the design and implementation of "correct, efficient, scalable, and robust protocols". It allows the user to intuitively specify large test topologies and their associated workload.

ENTRAPID protocol development environment combines the features of a multi-kernel approach and general-purpose network simulation. In this approach, a single FreeBSD kernel maintains multiple copies of kernels, called Virtualized Networking Kernels (VKNs), one for each machine in the emu-

lated network configuration. This eases the development of routing protocols, which, by their nature span multiple machines.

At the top level, `ENTRAPID` is a process running in the user space and can interact with other processes and with the network interface cards. It supports several hundreds of VKNs, therefore large network topologies can be emulated.

`NIST Net.` The `NIST Net` network emulator [15] [16] is developed by the National Institute of Standards and Technology (NIST)[4]. It is a network emulation package that runs on Linux, allowing any ordinary PC[5] to emulate a wide range of network conditions. `NIST Net` emulates "common network effects" such as packet loss, duplication or delay, router congestion and bandwidth limitations. It can emulate "performance dynamics" in IP networks and the "critical end-to-end performance characteristics imposed by various wide-area network situations".

`NIST Net` is implemented as a kernel module extension to the Linux operating system and an X Windows System-based user interface application. Its architecture is presented in Figure 2.6.

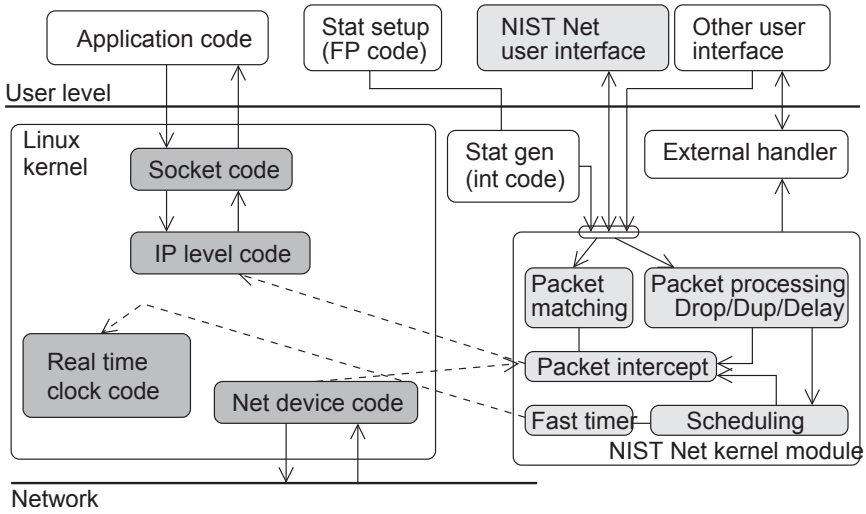


Figure 2.6: NIST Net architecture.

The `NIST Net` kernel module intercepts the IP packets at kernel level and classifies them based on user-defined rules. After classification, it determines if the packets are to be processed (i.e., delayed or dropped) and then it

[4] http://www.nist.gov/

[5] The PC must have 2 network interface cards and must be configured as a router.

forwards the packets to the Linux IP level code. For better time accuracy, NIST Net makes use of it own fast timer, which takes control of the system real-time clock.

NIST Net uses a set of statistical routines to generate the parameters for the impairments it applies to network traffic. The user can specify the average and the standard deviation for the delay and the loss the emulator can introduce.

ONE. The Ohio Network Emulator (ONE) is implemented on a Sun workstation, running the Solaris OS. It can emulate a certain network by specifying its topology.

ONE alters the network traffic based on several user-configurable parameters. The delay introduced by the emulator has three components:

- Transmission Delay - is the amount of time it takes to transmit a packet onto a given channel, and is computed as the ratio between the packet size and the bandwidth of the channel,
- Queueing Delay - for a given packet is the sum of the service times of the precedent packets already in the queue.
- Propagation delay - is the time needed by a packet to travel along a physical channel and is dictated by the speed of light through that particular medium.

The user can specify the linespeed and the propagation delay for the links between the emulated nodes. The size of each queue of a given interface can also be specified. Packet loss introduced by the emulator occurs when the queues are full.

2.3 Discussion

Anuē is the simplest network emulator, able to only emulate constant delays. Network emulators like Hammer PacketSphere, Simena, Delayline, Dummynet and NIST Net allow the user to chose between predefined distributions for the loss and delay they introduce. Unfortunately, the delay and the loss are random and uncorrelated. PacketStorm 4XG, Shunra Virtual Enterprise, ENDE and ENTRAPID are topology-based network emulators. They emulate specific networks which are completely defined by the user. Albeit the fact that the ONE network emulator is also topology-based, it introduces a more realistic degradation: the loss occurs when its internal

queues are full, and the delay takes into account the queueing delay, the propagation and the transmission of the packets. For further reading, various network emulation testbeds, like Emulab, PlanetLab or ORBIT, are described in [63]. The book also describes the emulators already presented in this book, but a different perspective is proposed.

The software network emulators lack precision, i.e. the degradation they introduce cannot be accurate. In any system, the packet loss is the result of a critical race for resources: a loss occurs at a certain moment depending on the relative timing of the packets. Any shift in this relative timing, due to lack of precision, generates a different loss pattern, resulting in a different application behaviour. Another drawback of the software network emulators is the fact that their performance strongly depends on the characteristics of the PC that is used. The frequency of the CPU, the data rate of the PCI or the efficiency of the network interface cards are only few examples of PC characteristics that influence the performance of the software network emulators.

Most of the existing network emulators are network topology oriented. They emulate a particular network, by using a node-by-node representation. The main drawback of all emulators is the fact that the loss and the delay they introduce is not correlated, therefore the degradation that a certain traffic flow will experience is not realistic. Because each packet is treated independently, the delay variation may lead to packet reordering which in real network equipment cannot occur when there is only one way from the input port to the output port. The delay variation should be naturally induced as the consequence of variations in the experienced service time, due to the presence of other traffic flows. Also the original packet sequence should be preserved inside a certain stream, so that no artificial packet reordering is introduced.

Chapter 3

Emulation. Principles and Techniques

In this chapter we present the basic principles of network emulation, based on observations on how the quality degrades in the order 1 networks (see the taxonomy introduced in section 1.4). These principles are to be seen as general guidelines that would ensure the realism of network emulation. They create the context for the emulation techniques presented afterwords.

We identified two main techniques for the emulation of networks. The first approach is to directly emulate the background traffic within the queues of the network emulator, in order to alter the foreground traffic. The second approach is to view the capacity available to service the foreground traffic as being diminished by the processing for the background traffic, by using a *server with vacations*. We shall present each of these approaches and then, in the following chapter we shall illustrate their time-trace equivalence.

3.1 Principles of network emulation

The loss and delay can characterize any network. They can be the parameters of the quality degradation introduced by networks. According to IETF IPPM, the one-way delay [47] is defined as the difference in time between the reception of the last bit of a packet and the transmission of its first bit. The one-way loss [48] is defined to be zero when a packet is received within a certain finite time interval.

We start this section by observing an order 0 network. Packets are transmitted over the wire from one end to the other, and during this process they experience a certain quality degradation. Transmission degrades quality in two ways: (i) by introducing a constant delay due to propagation

through the transmission media, and (ii) by potentially introducing a fixed amount of loss, as consequence of the bit error rate (BER) characteristic of that particular media. This amount is a worst-case value over a specific time interval. The BER of the transmission media can be neglected, given its usually insignificantly small values–e.g., 10^{-12} for Gigabit Ethernet on optical fiber [58]. These observation lead to the first two principles:

P1: Delay has a fixed component due to propagation through transmission medium.

P2: Loss has a fixed component due to transmission impairments.

We shall analyze the degradation likely to be introduced by one queue, i.e., an order 1 network. Assuming the input rate is larger than the output rate, the queue will start to fill up. As the queue fills, the experienced delay will increase and when the queue becomes full, the delay experienced by packets reaches a maximum value, d_{max} (see Figure 3.1 (a)). The moment when the first packet loss occurs coincides with the moment when the queue becomes full, t_{full} (see Figure 3.1 (b)).

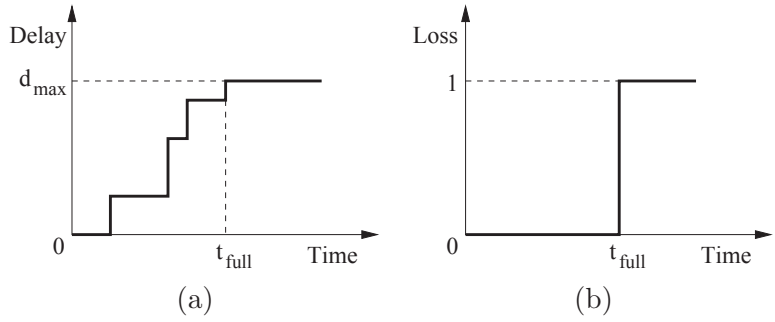


Figure 3.1: Delay (a) and loss (b) vs. time for a queue getting full.

The loss is therefore correlated with the delay. In the presented scenario loss cannot occur when the delay is less than d_{max}. The example above illustrates the third principle of network emulation:

P3: Loss and delay are correlated.

This third principle applies to order 1 networks or higher-order networks, where the intrinsic correlation between loss and delay reflects the competi-

tion for resources. For order 0 networks, the loss and the delay induced in the process of transmission are not necessarily correlated.

Treating the probability of loss and the experienced delay as independent random variables, as most of the existing network emulators do, may lead to unrealistic scenarios, unlikely to appear in real networks. When emulating networks, packets in a flow should not be treated independently, but as part of the context created by the traffic flow they belong to. Each packet experiences the degradation induced by the history created by the previous packets of the traffic flow.

The packets that get enqueued wait for service. The time spent by a packet in the queue is determined by the service time of the previous packets already enqueued. This time varies, depending on the traffic pattern and the distribution of the packet service time.

P4: Delay has a variable component induced by the competition for service.

Loss generally occurs when resources are exhausted. The competition between packets for getting into the buffer space is the source of loss.

P5: Loss is induced by the competition for buffer space.

The observations above along with the fore mentioned principles lead to the following two corollaries:

C1: Delay has two components—the fixed delay due to transmission through media and the variable delay induced by contention for service.

C2: Loss has two components—the fixed amount of loss due to transmission impairments and the variable loss induced by contention for service.

If only one traffic flow enters one queue, that the competition is between consecutive packets. This is the case of *"intra-stream contention"*[1]. If multiple traffic flows compete for having access to the same queue or the same resources, than the case is of a *"inter-stream contention"*.

The packets in a flow create a certain history of the queue or network element, therefore any new packet experiences the effects induced by precedent

[1]The terms *intra-stream contention* and *inter-stream contention* are from a personal communication with Neil Davies from Predictable Network Solutions, UK.

packets. The consequence is a variation in packet experienced delay. The IP Packet Delay Variation, IPDV [52], is defined as the difference between the one-way delay of two packets: $IPDV = D_i - D_j$. The two packets can be consecutive packets or can be chosen according to a selection function. IPDV can also be computed as the difference between the delay of the current packet and a reference delay value, like the average or the minimum delay.

The term *jitter* is used to refer to the absolute value of IPDV, computed for consecutive packets. Jitter is the natural consequence of the other traffic packets being serviced, to the detriment of the packet belonging to the traffic of interest. This stands for order 1 and order 2 networks.

P6: Jitter is consequence of presence of other traffic sharing the resources.

When emulating network elements like switches, packet reordering is in contradiction with the Ethernet standard [42], which specifies that packet order within a traffic flow should be preserved during the switching process. Artificially introduced jitter may lead to packet reordering. This can happen in most of the existing network emulators, if packets are treated independently so that the delay is applied independently.

Let's consider the following example. Two consecutive packets arrive at t_{a1} and t_{a2} (see Figure 3.2). If independent delay values are applied independently without preserving the initial order of packets, the two packet can be swapped.

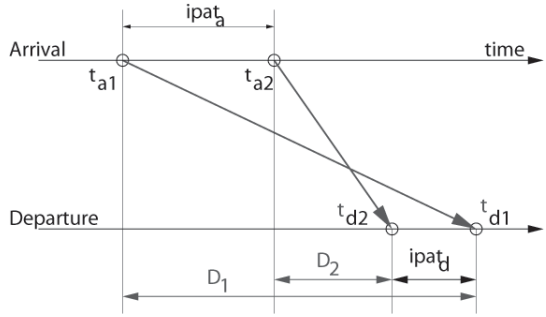


Figure 3.2: Independently applied delay.

In Figure 3.2 $ipat_a$ is the initial inter-packet arrival time, $ipat_d$ is the inter-packet arrival time at departure, D_1 and D_2 are the two delay values and t_{d1} and t_{d2} are the departure times of the two packets. The instantaneous jitter is:

$$j = |D_2 - D_1| = D_1 - D_2 = ipat_a + ipat_d > ipat_a \tag{3.1}$$

If the instantaneous jitter is larger than the initial inter-packet arrival time, the packets will be re-ordered.

In order 3 networks, packet reordering may appear when the route changes, so packets belonging to the same traffic flow arrive at destination on different routes. However, there exist network elements that have multiple paths from one input port to the same output port. This architecture is likely to reorder packets in the same traffic flow when the input load exceeds a certain threshold. Out-of-order packets were observed in tests in real networks over long-haul connections for which the route was statically configured and remained the same during the measurements [51], [50].

3.2 Background traffic technique

The background traffic can be considered to be either *passive* or *active*. The background traffic is considered to be passive if its sources send only UDP-like traffic or *inelastic* traffic. Passive background traffic can also be seen as a good approximation for the case when the network element is being shared by a large number of flows. In this situation, no individual flow will have a substantive effect on the sum of the other traffic flows and as such we can appeal to the statistical properties of large populations. On the other hand, should one wish to study the interaction between the foreground traffic and TCP-like traffic, the background traffic has to be *elastic* or reactive[2] to the foreground traffic. This requires a closer emulation of the end point's behaviour.

3.2.1 Passive background traffic

The background traffic is said to be "passive" when it doesn't modify its properties with respect to the pattern of the traffic of interest. Therefore, the emulation of the background traffic is independent on the pattern of the input traffic, whatever that pattern may be.

Constant-load background traffic. This is the simplest algorithm which is implemented in our network emulator, and corresponds to the emulation of order 1 networks. It uses one queue for both the foreground traffic and

[2]In the same sense as the TCP modifies its transmission rate to compete with other traffic flows in order to occupy the entire available bandwidth.

the background traffic (see Figure 3.3). A fixed amount of the available bandwidth is occupied by the background traffic. The queue is serviced at a constant rate, so a fixed amount of bandwidth is available for the foreground traffic. This is equivalent to using rate limitation mechanisms from the point of view of the effects on the foreground traffic.

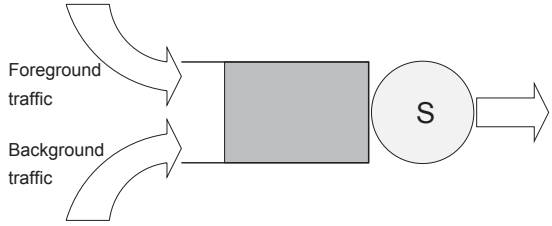


Figure 3.3: One queue scenario.

Variable-load background traffic. This algorithm emulates the case when the available bandwidth for the foreground traffic varies because the other traffic flows that share the same resources have a variable load. When more traffic flows share the same network, one can differentiate between them and service them according to the importance of each of them. Service classes refer to the possibility of defining different types of traffic and apply different treatment to them.

Depending on the service class to which the background traffic belongs to, there are several cases to be emulated. The first case is when both the foreground traffic and the background traffic belong to the same service class. The competition for getting into the same queue will cause loss, while the foreground traffic will experience a variable queueing delay and service rate as a result of servicing the background traffic.

The second case is when the foreground traffic and the background traffic belong to different service classes and are assigned to separate queues (see Figure 3.4). The background traffic will then influence the service provided to the foreground traffic in a manner that depends on the scheduling mechanism implemented in the MUX.

To give an example of the possible influence of the scheduling mechanism, let's consider the case of Strict Priority being implemented in MUX. Two cases can be distinguished: (i) the background traffic has higher priority – the foreground traffic will not be serviced while the background traffic queue is non-empty; and (ii) the background traffic has lower priority – the foreground traffic will only be affected by the background traffic if such

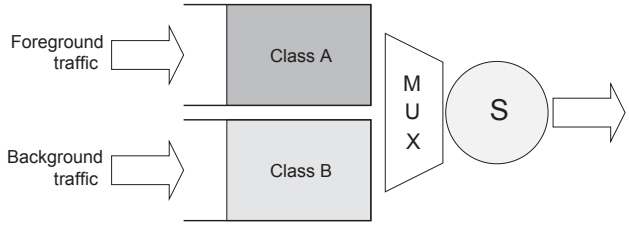


Figure 3.4: Different service classes scenario.

packets arrive during the transmission of background traffic packets that started when the foreground traffic queue was empty.

The general model combines all the previously mentioned cases in a situation when background traffic is assigned to classes with the same priority and different (higher and lower) priorities than the foreground traffic. We aim at building this general model based on the simple models described above. This corresponds to the emulation of order 2 networks.

3.2.2 Active background traffic

The background traffic is said to be "active" when it does adjust its temporal properties, as a function of the varying temporal properties of the foreground traffic. This kind of emulation permits the study of the interaction between "elastic" traffic flows, i.e. flows that adapt their rate to network conditions in order to make better use of the available bandwidth.

The emulation of an active background traffic requires the use of a feedback mechanism, one that adjusts the rate or the burst size of the background traffic source by means of a rate control system (RCS). An outline basic scheme is presented below.

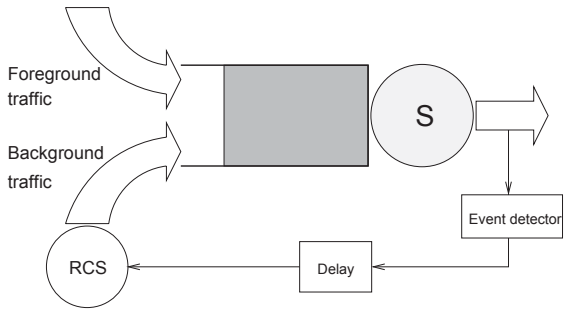


Figure 3.5: The active background traffic emulation.

The feedback on the *reverse path* emulates the rate control mechanism, such as the acknowledgment-based TCP feedback. A module will detect the significant events (e.g. packet loss) triggering the modification of the output rate of the source of background traffic. This makes it possible to emulate the effects of the TCP slow start or congestion avoidance algorithm or the modification of the rate of an UDP stream (e.g. decrease of transmission rate by change of codec during congestion, for a VoIP application). An additional module introduces a user-configurable delay that represents the length of the reverse path and provides a realistic delayed feedback.

3.2.3 Multiple background traffic sources

A general scheme for the emulation of an order 1 network is presented in Figure 3.6. Multiple sources of both passive and active background traffic are used in order to have a realistic emulation.

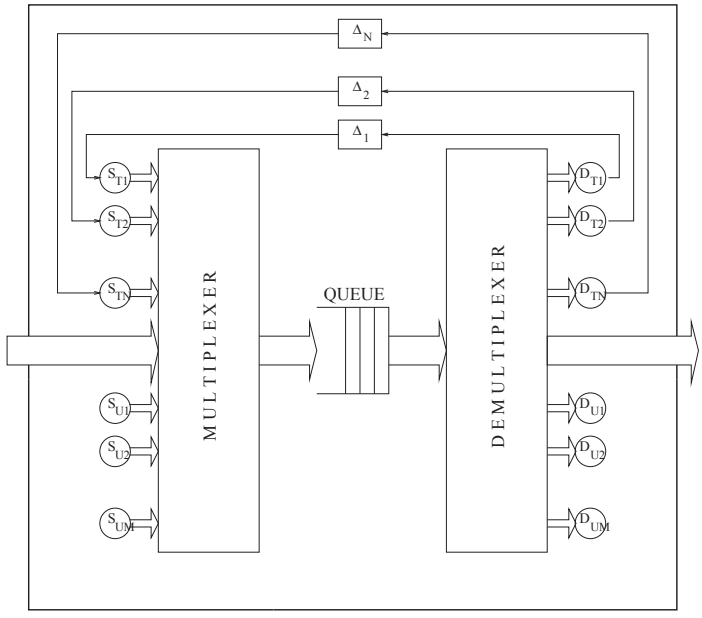


Figure 3.6: General architecture for the emulation of order 1 networks.

S_{Ti} are N sources of TCP-like traffic, D_{Ti} are N destinations for the TCP-like traffic, Δ_i are elements that delay the feedback from the destinations of the TCP-like traffic, emulating different RTTs, S_{Ui} are M sources of UDP-like traffic and D_{Ui} are M destinations for the UDP-like traffic.

The complexity of this general architecture increases if the number of sources increases. Such an architecture is difficult to implement and may

require the dynamic allocation of the resources. A more compact architecture can be imagined (see Figure 3.7).

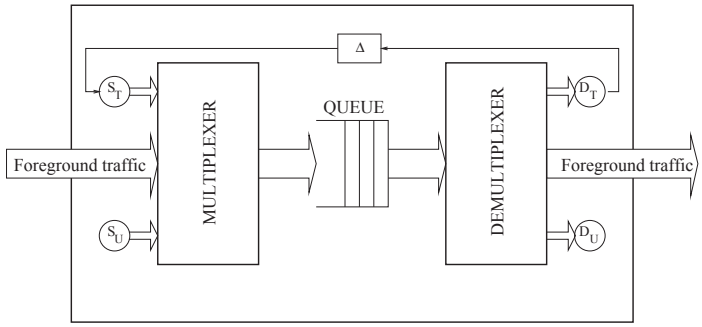


Figure 3.7: Compact architecture for the emulation of order 1 networks.

The sources of background traffic are replaced by only two sources, one for active and one for passive background traffic, respectively. The equivalent sources are obtained by aggregating all the sources into one single model. This architecture is more appropriate to implement, regardless the number of the desired background traffic sources. It can also be used to emulate order 2 networks, by implementing more queues and a scheduling scheme.

3.3 Server with vacations technique

An alternative approach to emulating the behaviour of order 1 and order 2 networks is the one of server with vacations[3]. In this approach the effects of the background traffic on the foreground traffic are emulated by the service being temporarily unavailable (the server S is "on vacation"). From the point of view of the foreground traffic, the server is not available for certain intervals of time, equivalent to the period when other traffic is being serviced.

Figure 3.8: Server with vacations.

[3]The term "server with vacations" is used in the queueing theory-related literature to denote a server whose services are unavailable for the traffic of interest because of various reasons.

The main advantage of this approach is that it is not necessary anymore to generate virtual background traffic, the effect of the large number of flows being emulated by the rate and duration of the vacations taken by the server. This is useful given that the number of such flows can be very large in complex scenarios, which would require a high computational complexity.

The discussion above is valid for the case of passive background traffic. More elaborate vacation models must be employed when emulating active background traffic. A possible solution may use a vacation predictor to determine the moments when the server is on vacation, depending on how the equivalent active background source adapts its transmission rate. In other words, the vacation predictor varies the apparent service rate, as a consequence of the interaction with other traffic flows.

3.4 Approach equivalence

The server with vacations approach is more attractive due to its lower computational costs. However, one must make sure that there exists an equivalence between this approach and the background traffic approach. In this case, the properties of the equivalent server with vacations will be determined. There is the potential for many different levels of equivalence:

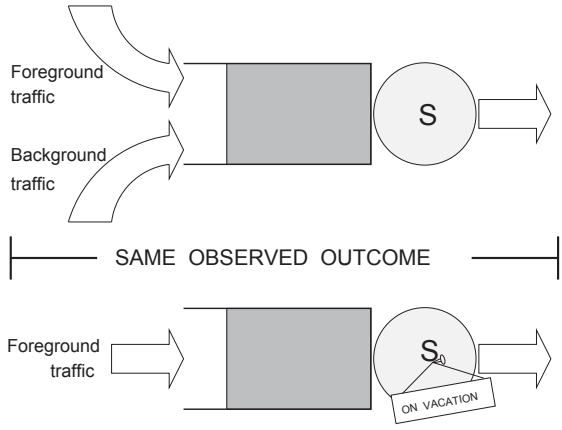


Figure 3.9: Background traffic versus server with vacations.

1. trace-level equivalence – the time traces of the foreground traffic flows at the output of the two systems are identical

2. statistical-distribution-level[4] equivalence – the probability density func-

[4]e.g. probability density function (PDF).

tions of the output foreground traffic flows in the two cases are "equal" by a goodness-of-fit test, like chi-square [18],

3. application-level equivalence – the observable behaviour of the application is the same for both cases

The equivalence (1) is a strict-sense equivalence, whereas the last two are equivalences on the basis of a certain aggregate measure. Hence, two systems equivalent in the first acceptation will also be equivalent in acceptations (2) and (3). However, the last two equivalence measures do not imply each other, as illustrated in Figure 3.10. One can envisage a case when two traffic flows with equivalent PDFs, produce different application behaviour because of the different time pattern of the flows.

For example, for TCP-based applications, the outcome may differ significantly, depending on the type of packets that are lost: if data packets are lost, they are retransmitted as soon as their loss is detected, i.e. after one RTT. Control packets, like connection establishment packets are retransmitted after a timer, usually of 1 second, expires.

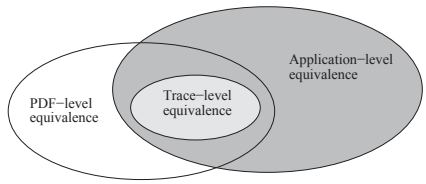


Figure 3.10: Levels of equivalence.

Our main interest lies in the application-level equivalence, i.e. end-user perceives the same application performance. We shall assume that the background traffic is passive, that allows us to simplify the task of finding the relationship between the distribution of the vacations and the distribution of the background traffic (see Section 4.4).

3.5 Emulating multiple hops

More hops can be emulated by concatenating queues and wires. In the spirit of background traffic approach, two queues can be concatenated as shown in Figure 3.11. Two scenarios can be identified: (i) the background traffic from source 1 goes along with the foreground traffic through the second queue ; (ii) the background traffic from source 1 exits the first queue, i.e., it shares only the first segment with the foreground traffic.

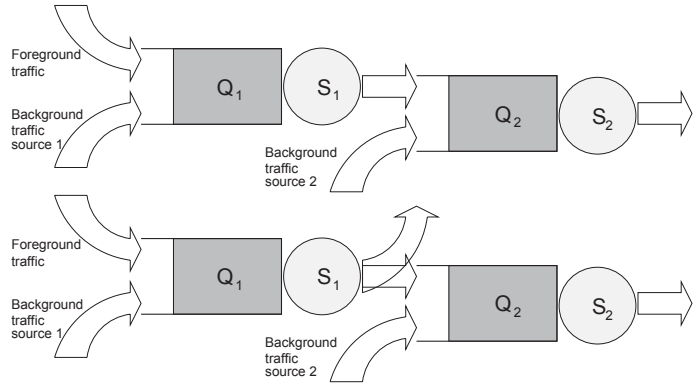


Figure 3.11: Emulating two concatenated queues - scenarios.

Another example illustrates the case when only certain segments of the same network are shared by the traffic flows. The following figure shows an order 3 network, with four routers and six end nodes. Consider the case when there are three sources that send traffic to three destinations.

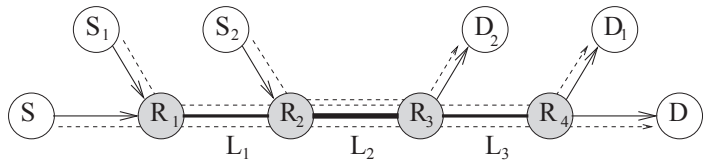


Figure 3.12: Example of traffic flows sharing routes.

The traffic of interest would be from source S to the destination D. This traffic shares the same network with the background traffic (i.e. traffic from S_1 to D_1 and traffic from S_2 to D_2). In this particular situation, only the link L_2 is shared between all three traffic flows. When emulating order 3 networks the path of each traffic flow has to be taken into account. The quality degradation introduced by each order 1 and order 2 network depends on the traffic pattern. The overall behaviour of the order 3 network will depend on the routes chosen for the traffic flows.

Chapter 4

Emulation. The Models

Queueing theory makes use of probabilistic techniques to study waiting line phenomena. It allows computing mean values that characterize a queue, like the expected steady-state number of customers in the queue or the mean steady-state time a customer spends in the queueing system, based on the arrival traffic pattern and the service time distribution.

We present here a temporal model, that permits to determine the time trace of the output traffic, when the time trace of the input traffic and the service time for each packet are known. The results will be validated by simulating the queues in Python programming language. The consistency of this approach with classic queueing theory will be proved by comparing the results of our model and of simulations against the results of queueing theory.

Having this model of a queue, it is possible to combine queues and to obtain a single model exhibiting the same properties as a complex system. As an example, the equivalent model for two concatenated queues will be determined. The same model will be used to determine the server vacations, thus emulating the effects of the background traffic being serviced.

4.1 Queueing theory

We briefly present here the M/M/1 and the M/M/1/K models, representing an infinite and a finite queue, respectively. According to Kendall notation [18], M/M/1 represents a system with exponential inter arrival distribution (the first M), exponential service time distribution (the second M) and 1 server. K indicates the system capacity, i.e., the maximum number of customers allowed in the system, when the queue is finite.

Using the formulae in [18] we plot the average delay and loss character-

istics of these two queue models[1], as functions of offered load (see Figures 4.1 and 4.2).

4.1.1 M/M/1

The steady state probability that there are n customers in the system is:

$$p_n = (1 - \rho)\rho^n \tag{4.1}$$

where ρ is the server utilization, expressed as the ration between λ, the mean arrival rate of customers in the system, and μ, the mean service rate (the mean rate of service completion while the server is busy).

Expected steady state time a customer spends in the system is:

$$W = \frac{W_S}{1 - \rho} \tag{4.2}$$

where W_S is the expected customer service time, $\frac{1}{\mu}$. The expected number of customers in the queueing system, L is given by:

$$L = \frac{\rho}{1 - \rho} \tag{4.3}$$

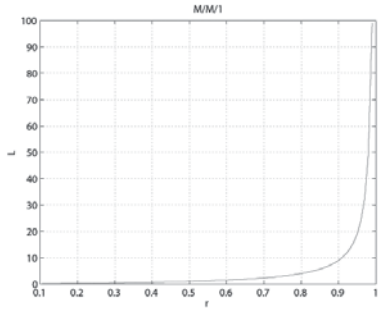


Figure 4.1: M/M/1 model queue occupancy as function of server utilization.

4.1.2 M/M/1/K

In this case, p_n, the steady state probability that there are n customers in the system is:

$$p_n = \begin{cases} \frac{(1-a)a^n}{(1-a^{K+1})} & \text{if } \lambda \neq \mu \\ \frac{1}{K+1} & \text{if } \lambda = \mu \end{cases} \tag{4.4}$$

[1] M/M/1 does not have a loss characteristic, since the queue is infinite.

where a is the traffic intensity or the offered load.

$$a = \lambda W_S \tag{4.5}$$

The international unit of traffic intensity is the *erlang*, named for A. K. Erlang, a queueing theory pioneer.

$$a = \lambda W_S = \frac{\lambda}{\mu} = \rho \tag{4.6}$$

The probability of loss is equal to the probability of having the queue full, i.e. of having K customers in the system. Remember that K is the capacity of the system (the length of the queue). In Figure 4.2 the probability of queue full was plotted as function of server utilization, for several queue sizes: 1, 2, 5, 10, 20 and 50. One can notice that as the size of the queue increases, the probability of loss decreases. The queue occupancy is given by:

$$L = \begin{cases} \frac{a[1-(K+1)a^K+Ka^{K+1}]}{(1-a)(1-a^{K+1})} & \text{if } \lambda \neq \mu \\ \frac{K}{2} & \text{if } \lambda = \mu \end{cases} \tag{4.7}$$

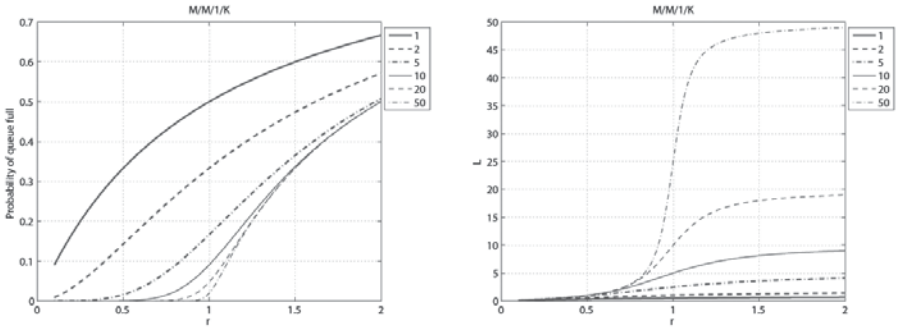


Figure 4.2: M/M/1/K model - probability of queue full as function of server utilization (left) and queue occupancy as function of offered load (right).

4.2 A temporal model

Let us consider a generic queue (see Figure 4.3). The input traffic is characterized by the arrival times of the packets, denoted by X, and the associated service times, S.

For given X and S we want to determine the departure times, denoted by Y. X, S and Y represent discrete time values.

$$X = \{x(n)|n = \overline{0, N-1}\} \tag{4.8}$$

$$S = \{s(n)|n = \overline{0, N-1}\} \tag{4.9}$$

$$Y = \{y(n)|n = \overline{0, N-1}\} \tag{4.10}$$

The service time of the packets is expressed in the same time units as the arrival and departure time. All these time values can be expressed in quanta of 1 ns, for instance. For 10 Gigabit Ethernet, the first bit of a packet can be transmitted at any moment in an interval of 1 ns [43]. However, one could determine the distance error between the model and a real transmission mechanism, but this is not our purpose.

Imagine the following model of a queue: the queue is a tube, and inside this tube there is a disk that can slide. The disk slides to the left when a packet arrives into the queue, and slides to the right each time a packet leaves the queue (see Figure 4.3): $w(n)$ is the waiting time for the n^{th} packet, or the work-load of the queue at the moment when the n^{th} packet arrives.

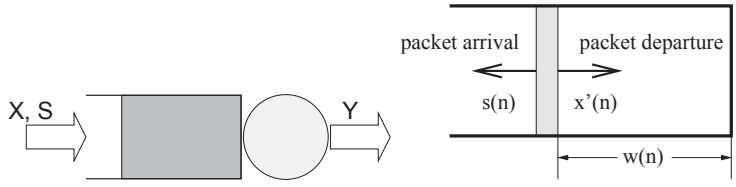


Figure 4.3: A queue and its temporal model.

When a packet is enqueued, the disk moves to the left with the amount of time needed to service that packet (i.e. the service time for that packet, $s(n)$). Then the disk slides to the right with the amount of time between two packet arrivals, $x(n) - x(n-1)$. This means that during that time interval, the queue gets serviced and its work-load decreases with the corresponding amount of time.

The inter-packet arrival time for the n^{th} packet is:

$$x(n) - x(n-1) = x'(n) \tag{4.11}$$

which is the discrete first derivative of the arrival time,

We shall use this model to analyze a finite and an infinite queue. The queues will be also simulated and the results—the measured loss and delay—compared against the results presented in sections 4.1.1 and 4.1.2. Then, the model will be used to prove the equivalence between the background traffic generation and the server with vacation approach (see Section 4.4).

4.2.1 An infinite queue

Let us consider the case of an infinite queue. The queue has two states: *empty* and *non-empty*. The initial position of the disk indicates an empty queue. If we make the assumption that the queue is never empty, the departure times are given by:

$y(0) = x(0) + s(0)$
$y(1) = x(1) + s(0) + s(1) - [x(1) - x(0)]$
$y(2) = x(2) + s(0) + s(1) + s(2) - [x(1) - x(0)] - [x(2) - x(1)]$

$$y(n) = x(n) + \sum_{i=0}^{n} s(i) - \sum_{i=1}^{n} x'(i) \tag{4.12}$$

From the last expression we can determine the waiting time for the n^{th} packet, which is:

$$w(n) = \sum_{i=0}^{n} s(i) - \sum_{i=1}^{n} x'(i) \tag{4.13}$$

The formula 4.12 can be further simplified to:

$y(0) = x(0) + s(0)$
$y(1) = x(0) + s(0) + s(1)$
$y(2) = x(0) + s(0) + s(1) + s(2)$

$$y(n) = x(0) + \sum_{i=0}^{n} s(i) \tag{4.14}$$

We shall now determine the condition for the queue to be empty. The n^{th} packet finds the queue empty if its arrival time, $x(n)$, is larger than the departure time of the previous packet, $y(n-1)$:

$$x(n) > y(n-1) \tag{4.15}$$

Using the formula 4.13, we get:

$$x(n) > x(0) + \sum_{i=0}^{n-1} s(i) \tag{4.16}$$

$$x(n) - x(0) > \sum_{i=0}^{n-1} s(i) \tag{4.17}$$

i.e. the total time elapsed between the arrival of the first packet and the arrival of packet n^{th} becomes larger than the sum of the service time for all

the packets, meaning that the server finished servicing all the jobs in the queue and it idles.

We can conclude that the departure times are given by:

$$y(n) = \begin{cases} x(n) + s(n) & \text{if the queue is empty} \\ x(k) + \sum_{i=k}^{n} s(i) & \text{if the queue is not empty} \end{cases} \tag{4.18}$$

where k is the index of the packet that finds the queue empty.

The general formula for the departure time would be:

$$y(n) = x(k) + \sum_{i=k}^{n} s(i) \;\mid\; x(k) - x(j) > \sum_{i=j}^{k} s(i) \tag{4.19}$$

where j, as well as k, are the indices of the packets that found the queue empty consecutively.

An M/M/1 queue was simulated in Python and the measured average delay over five runs is presented in Figure 4.4 as function of load. The time traces of the simulation, for the arrival times and service times were used to compute the departure times, using formula 4.19. The results of the simulation matched exactly the results given by the formula.

In the M/M/1 model, packet arrivals and departures are seen as point-processes, i.e., packets are enqueued and dequeued instantaneously. The service time is spent entirely in the service facility. The same conditions apply as well to the simulation in Python.

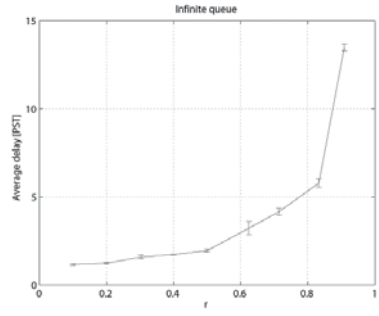


Figure 4.4: The average delay as function of offered load.

The average delay is expressed in packet service time (PST). This plot is similar to 4.1 showing the average queue occupancy predicted by the M/M/1 model. Multiplying the average queue occupancy, L, with the average PST, one can obtain the average delay for that queue.

4.2.2 A finite queue

A finite queue can be described as an algorithmic state machine (ASM) which has three states: *empty*, *non-empty* and *full*. In this case, the departure times are given by:

$$y(n) = \begin{cases} x(n) + s(n) & \text{if the queue is empty} \\ x(k) + \sum_{i=k}^{n} s(i) & \text{if the queue is not empty and not full} \\ \infty & \text{if the queue is full} \end{cases} \quad (4.20)$$

When the queue is full, packets are dropped. We can consider that a dropped packet will never leave the queue, i.e., its departure time is ∞.

The n^{th} packet is enqueued if:

$$\sum_{i=k}^{n} s(i) \leq queue\_length \quad (4.21)$$

or dropped, otherwise. The length of the queue is expressed in time. A packet is not enqueued unless there is sufficient space (service time) in the queue for the entire packet.

An M/M/1/K queue was simulated in Python, for several queue sizes: 1, 2, 5, 10, 20 and 50. The average delay and the loss are plotted as functions of the offered load in Figure 4.5.

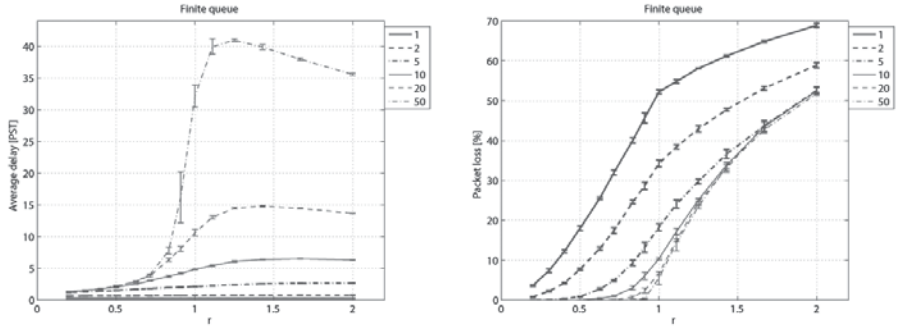


Figure 4.5: The average delay (left) and loss percentage (right) as function of offered load.

The time traces of the simulation, for the arrival times and service times were used to compute the departure times, using formula 4.20 given by the temporal model presented in this chapter. In the simulation the length of the

queue was expressed in bytes and the dequeueing of packets was performed incrementally in quanta of a certain number of bytes. The results of the simulation matched exactly the results given by the formula, the two time traces of the packet departure times were observationally identical.

4.3 Model comparison

In this section, the simulation results are compared against the theoretical values predicted by the queueing theory.

The average delay measured in the process of simulating an infinite queue is compared with the one predicted by the M/M/1 model (see Figure 4.6). One can notice that, in simulation, the average delay is slightly larger than the one predicted in theory.

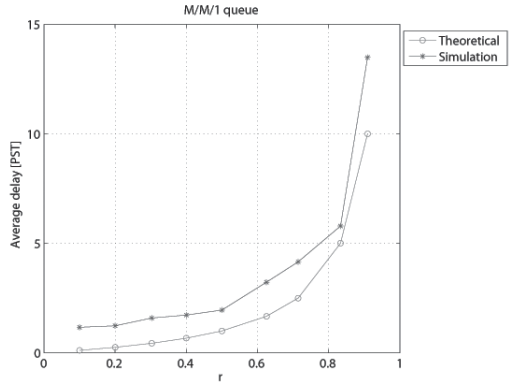


Figure 4.6: Average delay for M/M/1 and a simulated infinite queue.

The average delay and loss percentage measured during the simulation of a finite queue are compared with the corresponding values predicted by the M/M/1/K model (see Figure 4.7).

The average delay is slightly larger for loads smaller or equal to 1, similar to the case of an infinite queue. For loads larger than 1, the average delay is smaller than the one predicted in theory. The answer to these discrepancies resides in the different behaviour of the loss (see Figure 4.7). When $\rho \leq 1$, the loss is smaller than in theory, therefore more packets are enqueued and the average delay increases with the queue occupancy. For $\rho > 1$ the loss in simulation is larger, leading to a decreased queue occupancy and average delay.

However, the simulation is closer to what happens in a real system that

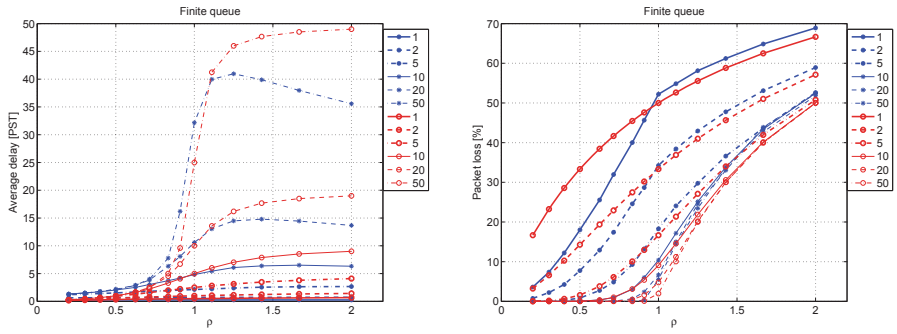


Figure 4.7: Average delay and packet loss for M/M/1/K and a simulated finite queue.

handles packets, for at least two reasons: (i) packets are serviced in data chunks and (ii) the packet arrival is not a point process like the assumption made by the queueing theory. Given that our temporal model matches the results of the simulation, we consider it to be a good approximation for the quality degradation that occurs in order 1 networks.

4.4 Server with vacations

Using the temporal model described in section 4.2 we shall determine the time and duration of the server vacations, for a given background traffic pattern. In other words, starting from a background traffic scenario, we can find the equivalent server with vacations that will cause the same quality degradation to the traffic of interest.

4.4.1 Emulating the delay

We shall start with an infinite queue, in order to determine the equivalent server with vacations from the point of view of delay. The fact that the queue is infinite is a necessary condition so that there will be no loss.

Determining the vacations means determining the moments in time when the server in on vacation and their durations. For the foreground traffic, the equivalent delay will be induced by these vacations. The background traffic source will be replaced by a server vacation source. A background packet generated at time t_0 will be replaced by a server vacation at time $t_0 + w$, where w is the waiting time in the queue for the background packet, given by the service times of the packets already in the queue. The duration of the

server vacation is equal to the service time of the background traffic packet.

Let us consider now the case with both foreground traffic, $x(n)$, and background traffic, $b(n)$, entering in the same queue, like in Figure 4.8.

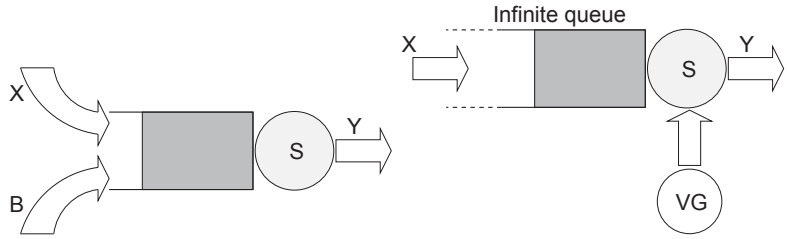


Figure 4.8: Foreground and background traffic (left) and infinite queue with a server vacation generator (right).

Given a certain foreground traffic pattern—characterized by its arrival times, $x(n)$, and the corresponding service times, $s_x(n)$—and a certain background traffic pattern—characterized by its arrival times, $b(n)$, and the corresponding service times, $s_b(n)$—we wish to determine the moments when the server is on vacation, denoted by $v(n)$, and the duration of the vacations, $s_v(n)$, so that the effects are the same for the foreground traffic.

From the point of view of the foreground traffic, servicing the k^{th} packet from the background traffic, characterized by the its arrival time $b(k)$ and its service time $s_b(k)$ is equivalent to a server vacation. This vacation has the same duration as the background packet service time, so $s_v(k) = s_b(k)$. The moment when the server will take this vacation is given by the arrival time of that background packet plus the necessary waiting time, i.e., $v(k) = b(k) + w(k)$.

We shall start from the formula 4.13 of the waiting time and take into account that in the same queue packets from both the foreground and the background traffic are eunqueued. The sum of the service times is composed of two sums: the sum of the service time corresponding to the foreground traffic, $\sum_{i=j}^{k} s_x(i)$, and the sum corresponding to the background traffic, $\sum_{i=j}^{k} s_b(i)$.

$$w(n) = \sum_{i=j}^{k} s_x(i) + \sum_{i=j}^{l} s_b(i) - \sum_{i=j+1}^{n} a'(i) \tag{4.22}$$

$$k + l = n \tag{4.23}$$

where k is the number of packets in the queue that belong to the foreground traffic and l is the number of packets representing the background

traffic. j is the index of a packet that found the queue empty. Depending on the pattern of the foreground and the background traffic, the inter-packet arrival times $a'(n)$ are given by:

$$a'(n) = \begin{cases} x(n) - x(n-1) & \text{if } n^{th} \text{ packet} \in X \text{ and } n-1^{th} \text{ packet} \in X \\ x(n) - b(n-1) & \text{if } n^{th} \text{ packet} \in X \text{ and } n-1^{th} \text{ packet} \in B \\ b(n) - b(n-1) & \text{if } n^{th} \text{ packet} \in B \text{ and } n-1^{th} \text{ packet} \in B \\ b(n) - x(n-1) & \text{if } n^{th} \text{ packet} \in B \text{ and } n-1^{th} \text{ packet} \in X \end{cases}$$

The equivalent server with vacation is therefore characterized by:

$$\begin{cases} s_v(n) & = s_b(n) \\ v(n) & = b(n) + w(n) = \sum_{i=j}^{k} s_x(i) + \sum_{i=j}^{l} s_b(i) - \sum_{i=j+1}^{n} a'(i) \end{cases} \tag{4.24}$$

4.4.2 Emulating the loss

In order to emulate the loss, the system must emulate the resource occupation due to the presence of the background traffic. This implies varying the length of the queue, i.e., the capacity of the system, so that the induced loss is the consequence of resource exhaustion. Therefore the model is a finite queue with variable length. The instantaneous value is determined based on the pattern of background traffic, whose effects are to be emulated.

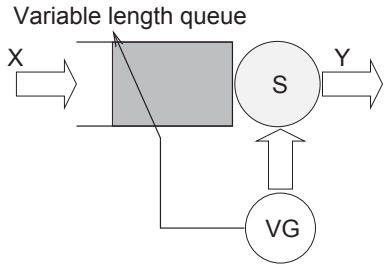


Figure 4.9: Finite variable length queue with a server vacation generator.

The vacation generator also dynamically adjusts the size of the queue, based on the queue history created by the foreground packets and the server vacations. The duration of the vacations correspond to the service times of an imaginary background traffic which no longer occupies space in the queue. The space that is virtually occupied by packets from a background traffic

has to subtracted from the queue maximum size, so that the foreground traffic experiences the same lack of resources.

The instantaneous queue length is computed as:

$$l = L - \sum_{i=0}^{k} s_v(i) \tag{4.25}$$

where L is the maximum size of the queue, expressed as service time, k is the number of server vacations to be taken at the current time, and $s_v(i)$ is the duration of i^{th} vacation.

4.5 Ideal behaviour of scheduling mechanisms

Strict Priority (SP) and Weighted Round Robin (WRR) scheduling algorithms are implemented in most of the current Gigabit Ethernet switches and routers. Their behaviour is very well known, but the quality degradation they introduce is only qualitatively studied. We accurately quantified the quality degradation induced by these two scheduling mechanisms. Results and detailed test configuration can be found in [36]. Here we only mention that there are eight identical sources of traffic, that send packets to the same destination. Each traffic flow has a different priority, i.e. goes in different priority queues. All the sources have the same instantaneous load, which is varied from zero to a certain percentage of the line speed.

4.5.1 Strict Priority

In this section we present a simple model for the behaviour of a system that has two and four queues, serviced according to the SP scheduling algorithm, for a particular input traffic. By definition, SP means that as long as there are packets in a high priority queue, this queue is serviced, regardless of the content of the low priority queues. Knowing the probabilities of having one packet at the input of each queue, we compute the probabilities of having at the output of the system a packet from a certain input queue.

We start by defining some basic terms. Having one queue and a constant bit rate (CBR) ingress traffic, we want to determine the probability of having one packet at the input of a queue, at a certain moment in time, knowing the average rate of the incoming traffic.

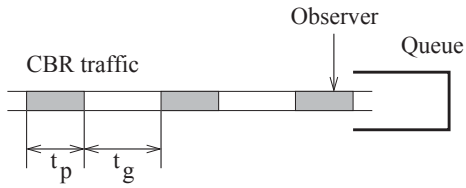


Figure 4.10: The CBR traffic at the input of a queue.

Let us note this probability with p_i. The duration in time of a packet is t_p and the duration of the inter-packet gap is t_g. At limit, the probability of having a packet at the input of the queue is given by the ratio:

$$p_i = \frac{t_p}{t_p + t_g} \tag{4.26}$$

When the input rate is zero, the inter-packet gap is infinity, so $p_i = 0$. When the input rate is line speed, the inter-packet gap has a minimum value. For simplification we shall consider this minimum value to be zero and therefore $p_i = 1$. In Figure 4.11(a) we plot p_i as function of the input rate, which takes values from zero to 1, where 1 represents line speed (LS).

Consider a system with four queues, each one being serviced according to a different priority: Queue #1 - highest priority, Queue #2 - high priority, Queue #3 - low priority and Queue #4 the lowest priority.

We shall use the following notations: p_{ij} is the probability of having a packet at the input of queue j, p_{sj} is the probability to service the queue j and p_{oj} is the probability of having a packet from queue j at the output of the system of queues. $j = \overline{1..4}$. Consider that there are four transmitters that send CBR traffic with four different priorities and their transmission rate is varied from 0 to 1 in the same time. This means that:

$$p_{i1} = p_{i2} = p_{i3} = p_{i3} = p_i \tag{4.27}$$

By definition, Strict Priority scheduling mechanism will service the highest priority queue as long as there are packets in it. The lower priority queues will be serviced only at the moments of time when the highest priority queue is empty. Making the assumption that the packet service time is the same for all the packets in the system and the time needed to dequeue a packet is zero, than the probability of having a packet serviced from the highest priority queue equals the probability of having a packet at the input of the highest priority queue:

$$p_{o1} = p_{i1} = p_i \tag{4.28}$$

The probability of servicing the high priority queue will be 1 minus the probability of having a packet at the output of the system from the highest priority queue.

$$p_{s2} = 1 - p_{o1} = 1 - p_i \tag{4.29}$$

The probability to have packets from the high priority queue is the minimum between the probability of having a packet in the hight priority queue, p_{i2}, and the probability of servicing that queue, p_{s2}:

$$p_{o2} = min\{p_{i2}, p_{s2}\} \tag{4.30}$$

If $p_{s2} > p_i$, the queue can be serviced with a larger probability than the probability of having packets in that queue. This means that the probability of having a packet serviced is p_i. If $p_s2 < p_i$, the probability of having packets enqueued is larger than the probability of servicing the queue, then having packets serviced from that queue will happen with the same probability of servicing the queue. Therefore:

$$p_{o2} = \begin{cases} p_i & \text{, for } p_i \leq 0.5 \\ 1 - p_i & \text{, for } p_i > 0.5 \end{cases} \tag{4.31}$$

Then p_{o1} and p_{o2} versus rate are depicted in Figure 4.11.

The probability of servicing the low priority queue p_{s3} is

$$p_{s3} = 1 - p_{o1} - p_{o2} \tag{4.32}$$

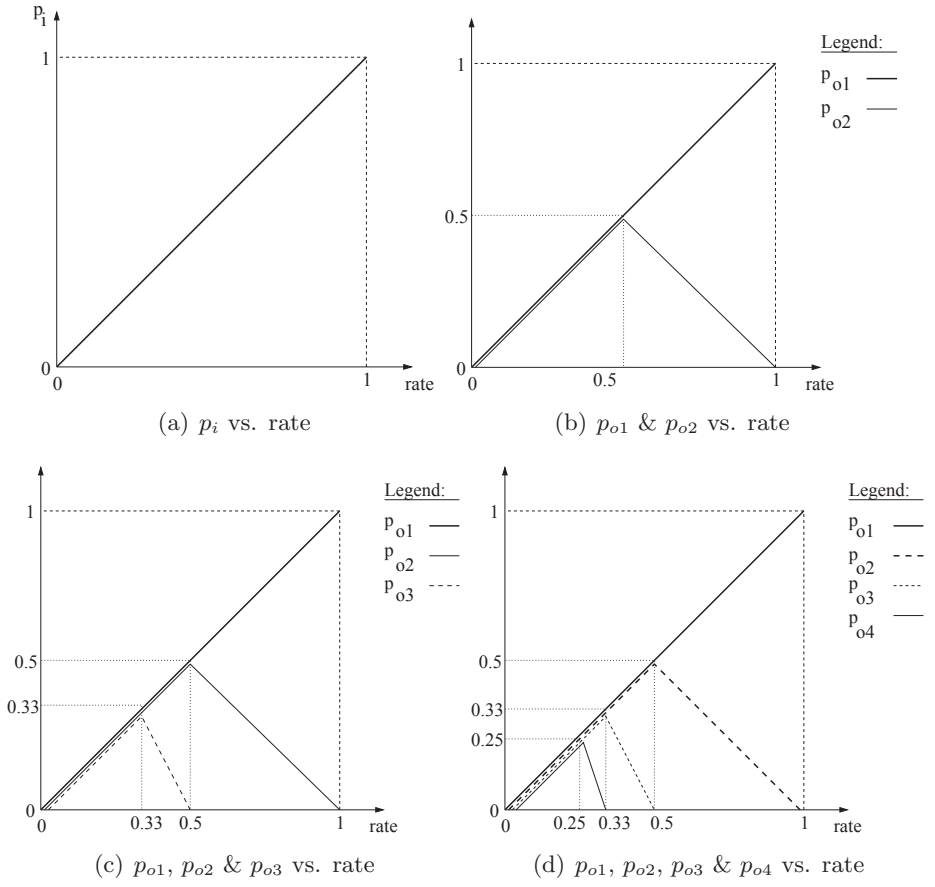
The probability p_{o3} is:

$$p_{o3} = min\{p_{i3}, p_{s3}\} \tag{4.33}$$

Following the same idea, $p_{s4} = (1 - p_{o1} - p_{o2} - p_{o3})$.

The SP mechanism for four queues was simulated in Python. Eight sources of traffic were used. The results are plotted in Figure 4.12 as the dependency of received load on transmitted load, per priority.

The points that define the behaviour of SP were denoted by Ⓐ, Ⓑ, Ⓒ and Ⓓ. They indicate the place where the decrease in quality starts, as experienced by the traffic in queues q_1 to q_4, respectively. Point Ⓐ represents the moment when the total offered load for all queues (q_1 to q_4) reaches 1 Gbps. Past Ⓐ, q_4 starts experiencing loss. Point Ⓑ indicates that the total load for queues q_1 to q_3 is 1 Gbps; thereafter, q_3 loses packets as well and q_4 is starved. Point Ⓒ shows the moment when the total amount of traffic in queues q_2 and q_3 reaches 1 Gbps. From this onward, packet loss occurs



(a) p_i vs. rate

(b) p_{o1} & p_{o2} vs. rate

(c) p_{o1}, p_{o2} & p_{o3} vs. rate

(d) p_{o1}, p_{o2}, p_{o3} & p_{o4} vs. rate

Figure 4.11: Probabilities vs. rate for SP.

for q_2. Starting at point Ⓓ q_1 is saturated and the received traffic is limited to 500 Mbps for each of the priorities 6 and 7. Everything else is lost.

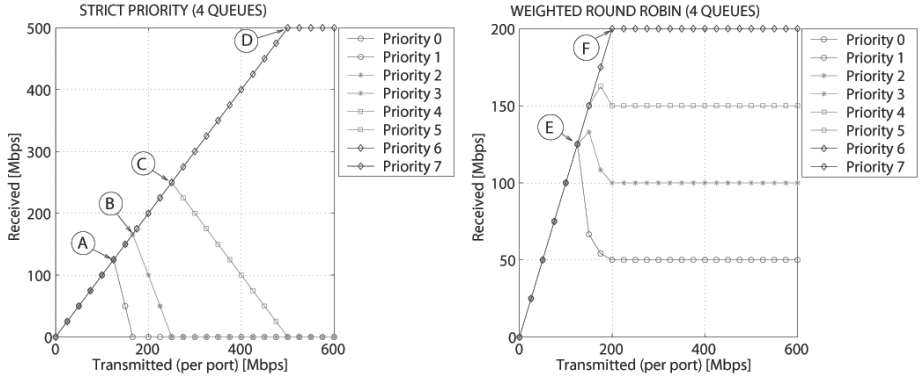
If the maximum transmission rate was 500 Mbps, the points Ⓐ (125, 125), Ⓑ (165, 165), Ⓒ (250, 250) and Ⓓ (500, 500) have the same coordinates as in figure 4.11. The results of the simulation therefore match the results theoretically determined.

For the highest priority, in saturation, the delay introduced is given by the size of the queue. For the other queues, the delay introduced depends on the pattern of the traffic with the higher priority. For example, the packets in the second priority queue will wait until the highest priority queue empties before they are serviced.

4.5.2 Weighted Round Robin

WRR is a variant of the Round Robin mechanism [44] that services each priority queue in a cyclic manner, proportionally to its associated weight. This allows control of the minimum amount of bandwidth guaranteed for each priority queue. The average delay for each traffic flow is bounded and can be estimated *a priori*. The values that represent the relative weights for the queues, also represent the expected ratios between the throughput of each queue.

The WRR algorithm was simulated in Python and the results are shown in Figure 4.12. They represent the expected ideal behaviour of WRR scheduling (a work-conserving mechanism). Again, we assume the case of a switch with Gigabit Ethernet ports and four priority queues. The corresponding weights are as follows: 0.4 for q_4, 0.3 for q_3, 0.2 for q_2 and 0.1 for q_1. Saturation occurs when all transmitters send traffic at 125 Mbps (point Ⓔ), thereafter service differentiation appears. Starting at point Ⓕ bandwidth guarantees are enforced.



(a) SP (b) WRR

Figure 4.12: Simulation results - received vs. transmitted load.

4.6 A Quality Degradation Algebra

We have found so far a mathematical representation for order 0 and order 1 networks. For the emulation of order 2 networks we analyzed the ideal behaviour of two widely-used scheduling algorithms. For SP, a theoretical model is presented and showed that its result match the ones obtained in simulation. We shall use these results further on as reference for our hard-

ware implementation of the two mechanisms.

For the emulation of order 3 networks we need a formal way to aggregate these objects. In this section we propose a degradation emulation algebra, as a bottom-up compositional approach, starting from basic elements that are aggregated into more complex representations. An algebra is a formal way of manipulating objects to obtain other objects with known properties. We propose an algebra to combine models of order 0 and order 1 networks into an equivalent model of order 3 networks built from these elements.

Sequences of degraders can thus be collapsed into a single degrader, that is easier to implement and manipulate. Defining an algebra makes it possible to recombine the elements due to properties of commutativity and associativity, obtaining minimal equivalent representations. The conditions under which such properties hold must be determined.

The degradation emulation algebra is defined over a set of two basic degradation elements: *queues* and *wires*. A wire $W(L, D)$ is an object that introduces constant loss (L) and constant delay (D). The constant amount of loss is given by the effective bit error rate (BER) characteristic for a particular transmission medium. The constant delay is due to propagation through the medium. However, the wire is just an abstract object which exhibits deterministic known properties.

A queue $Q(l, d)$ is an object that introduces variable loss (l) and variable delay (d). The variable loss and delay depend on several parameters like the length of the queue, the pattern of the input traffic, the distribution of the service time etc.

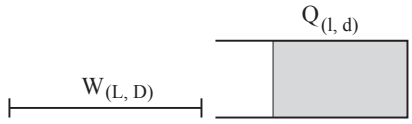


Figure 4.13: The *wire* and *queue* elements.

We denote the operation used to combine these elements by "$\circ$". This is equivalent to cascading two elements. In this notation, a queue $Q(l, d)$ followed by a wire $W(L, D)$ will be represented by $Q(l, d) \circ W(L, D)$. In the following relationships, one must determine $\hat{L}$, $\hat{D}$, $\hat{l}$ and $\hat{d}$:

$$W(L_1, D_1) \circ W(L_2, D_2) \Leftrightarrow W(\hat{L}, \hat{D}) \tag{4.34}$$

$$Q(l_1, d_1) \circ Q(l_2, d_2) \Leftrightarrow Q(\hat{l}, \hat{d}) \tag{4.35}$$

The relationships above will determine whether two element models of the same type can be combined into a single one that has the same behaviour. The next two equivalences are still to be validated:

$$Q(l,d) \circ W(L,D) \Leftrightarrow Q(\hat{l},\hat{d}) \tag{4.36}$$

$$W(L,D) \circ Q(l,d) \Leftrightarrow Q(\hat{l},\hat{d}) \tag{4.37}$$

We shall determine the conditions under which associativity and commutativity of "$\circ$" hold.

When two wires are concatenated, the propagation delays D_1, D_2 sum up and the equivalent system is a wire for which the propagation delay D is the sum $D_1 + D_2$:

$$D = D_1 + D_2 \tag{4.38}$$

Using the same notations that were used for the temporal model descibed in Section 4.2, the departure times are:

$$y(n) = x(n) + D_1 + D_2 \tag{4.39}$$

Let's assume that the loss percentage characteristic of the two wires are L_1 and L_2. If M is the number of packet at the input of the first wire and N is the number of packets that arrive at the output of second wire:

$$N = M(1 - L_1)(1 - L2) \tag{4.40}$$

$$= M(1 - L_1 - L_2 + L_1L_2) \tag{4.41}$$

$$= M - M(L_1 + L_2 - L_1L_2) \tag{4.42}$$

Then the equivalent wire will be characterized by the loss percentage:

$$L = L_1 + L_2 - L_1L_2 \tag{4.43}$$

The operation of concatenating wires is commutative, based on the commutativity of sumation and multiplication.

For three concatenated wires, $W_1(L_1, D_1)$, $W_2(L_2, D_2)$, $W_3(L_3, D_3)$, the delay characteristic of the equivalent system is:

$$D = D_1 + D_2 + D_3 \tag{4.44}$$

The departure times when N wires are concatenated is:

$$y(n) = x(n) + \sum_{i=1}^{N} D_i \tag{4.45}$$

For the same M and N as above:

$$N = M(1 - L_1)(1 - L2)(1 - L_3) \tag{4.46}$$
$$= M(1 - L_1 - L_2 - L_3 + L_1L_2 + L_1L_3 + L_2L_3 - L_1L_2L_3) \tag{4.47}$$
$$= M - M(L_1 + L_2 - L_1L_2 - L_1L_3 - L_2L_3 + L_1L_2L_3) \tag{4.48}$$

Then the equivalent loss characteric is:

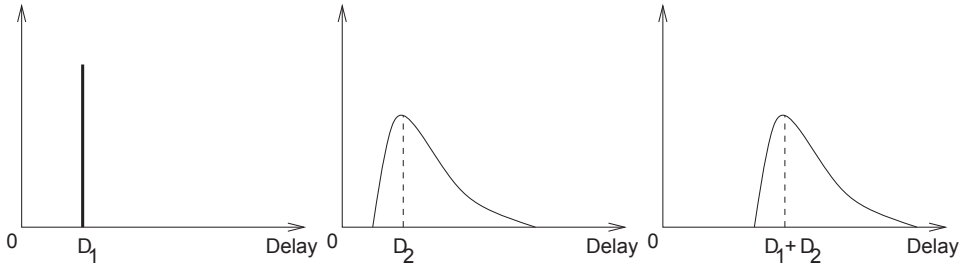
$$L = L_1 + L_2 - L_1L_2 - L_1L_3 - L_2L_3 + L_1L_2L_3 \tag{4.49}$$

When concatenating a queue and a wire, the wire is the one responsible for the constant delay of the equivalent system. The delay distribution that characterizes the queue will be shifted to the right, with the amount given by the constant delay introduced by the wire.

The equivalent model is determined using the following formula:

$$y(n) = D_1 + x(n) + \sum_{i=0}^{n} s_1(i) - \sum_{i=1}^{n} x'(i) \tag{4.50}$$

The resulting delay histogram is presented in Figure 4.14 (c).



(a) wire (b) queue (c) wire + queue

Figure 4.14: The delay histogram characteristics.

Using the temporal model described in Chapter 4 Section 4.2, one can determine the vacancies of an equivalent server, in order to emulate two queues:

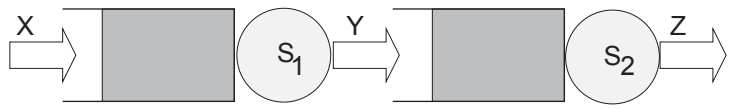


Figure 4.15: Two queues concatenated.

We shall assume the case of two infinite queues. The output of the first queue can be expressed as:

$$y(n) = x(n) + \sum_{i=0}^{n} s_1(i) - \sum_{i=1}^{n} x'(i) \tag{4.51}$$

The output of the second queue can be expressed as:

$$z(n) = y(n) + \sum_{i=0}^{n} s_2(i) - \sum_{i=1}^{n} y'(i) \tag{4.52}$$

Then,

$$z(n) = y(n) + \sum_{i=0}^{n} s_2(i) - \sum_{i=1}^{n} y'(i) \tag{4.53}$$

$$= x(n) + \sum_{i=0}^{n} s_1(i) - \sum_{i=1}^{n} x'(i) + \sum_{i=0}^{n} s_2(i) - \sum_{i=1}^{n} y'(i) \tag{4.54}$$

As:

$$y'(n) = \left[x(n) + \sum_{i=0}^{n} s_1(i) - \sum_{i=1}^{n} x'(i) \right]' \tag{4.55}$$

$$= [x(n)]' + \left[\sum_{i=0}^{n} s_1(i) \right]' - \left[\sum_{i=1}^{n} x'(i) \right]' \tag{4.56}$$

$$= x'(n) + 0 + \sum_{i=2}^{n} x"(i) = x'(n) + \sum_{i=2}^{n} x"(i) \tag{4.57}$$

The departure times for the equivalent system to the two concatenated queues are:

$$z(n) = x(n) + \sum_{i=0}^{n} s_1(i) - \sum_{i=1}^{n} x'(i) + \sum_{i=0}^{n} s_2(i) - x'(n) - \sum_{i=2}^{n} x"(i) \tag{4.58}$$

Chapter 5

An FPGA-based Emulator

This chapter describes the FPGA-based hardware platform we used and the internal architecture of the network emulator, as well as the design process. The advantages of a hardware implementation are emphasized, as well as the benefits emerging from the use of a message passing paradigm and its implementation in Handel-C.

5.1 Hardware vs. software implementation

One of the problems of the software network emulators is the lack of accuracy of the degradation they induce. Although the network applications are usually not sensitive to a very precise delay, accurate timing is needed for the correct emulation of effects depending of the packet sequential order. Such an effect is the competition for the last available buffer between two streams, the cause of packet loss. In any system, loss is the result of a critical race for resources. The fact whether packet loss occurs or not at a certain moment depends on the relative timing of the packets. Any shift in time results in a different packet being dropped. Therefore, accurate emulation of these effects is needed in order to get correct loss rates and distributions, a mandatory feature of an emulator since they are crucial for the performance of most applications. A hardware implementation, such as ours, ensures a correct behavior regarding timing.

The network emulator is intended to sustain 1 Gb/s rates, which means a packet rate of approximately 1.5 million packets per second, for the smallest size Ethernet packets. This gives a worst-case processing time between two consecutive packets of $0.67\mu s$. In order to be able to sustain such a packet rate and to benefit of enough processing time for each packet, a hardware approach is mandatory. In a software approach, the packet arrival events

are not deterministic. This is because there is a variable delay between the moment of time when the packet arrives at the network interface card and the moment the packet data is delivered to the application level. This delay depends on when the interruption from the network interface card occurs, how fast the driver fetches the packet data over the PCI interface, and then how fast the kernel copies the packet data to the user space. In hardware all these uncertainties are eliminated, given that the packet data can be read from the medium access controller (MAC) the moment when the start of packet bit is set [59].

5.2 The hardware platform

Our implementation is based on a custom-design PCI platform [20] [21]. This hardware platform uses one Altera Stratix FPGA [53] with 25k logic elements, two Gigabit Ethernet RJ45 ports and memory: 64 MB of synchronous dynamic RAM (SDRAM) and 1 MB of synchronous static RAM (SSRAM). The schematic is presented in Figure 5.1:

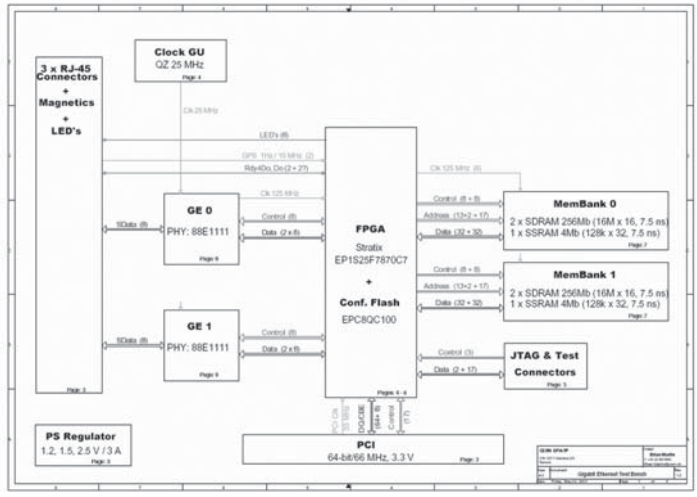


Figure 5.1: Schematic of the hardware platform.

The FPGA manages all the resources and contains the intellectual properties (IPs) for the two Ethernet MACs and the SDRAM and PCI controllers. The user-defined higher-level functionality is implemented on the same FPGA, using the Handel-C programming language [6]. A low-level library written also in Handel-C is used to provide primitives for memory and PCI access. The dual-port on-chip RAM blocks provided by the Stratix

FPGA are extensively used to implement queues between concurrent processes.

The SDRAM is used to temporarily store packet data. The SSRAM is used to store the configuration of the emulator. Packet data flows between the two Gigabit Ethernet ports, allowing for the integration of the hardware platform into a network. The board can be hosted by PCs that have a standard 3V3 PCI connector, which facilitates the deployment. The PCI is used to configure the application firmware and to collect statistics.

The whole system is driven at the PC level, by a Python-based control system. This allows to create automated procedures for performing experiments in a very simple and flexible manner. The low-level communication with the hardware platform via PCI is performed through a custom Linux kernel module [22].

5.3 Implementation philosophy

The emulator was implemented using the Handel-C language [6] from Celoxica[1]. Handel-C is hardware description language that has a syntax which is similar to the syntax of ANSI[2] C [7]. While ANSI C is a merely sequential programming language, Handel-C offers the possibility of executing instructions in parallel. We chose Handel-C because it is readily accessible to non-hardware specialists while at the same time offers constructs that allowed us to use the natural parallelism of the hardware. Handel-C had been employed in other networking applications at CERN [54].

Given its roots in CSP[3] and occam[4], Handel-C provides specific constructs for developing systems with concurrent processes.

The most obvious construct of use is the `par`, which allows for the parallel execution of statements or complete processes. In practice every module shown in Figure 5.3 runs as a process under a top-level `par`. The architecture and the Handel-C have a one-to-one correspondence. Each arrow in the diagram has a direct correspondence with a Handel-C channel in the implementation.

Channel objects allow data to be communicated between processes. The data transfer occurs only when both processes are ready and forces the syn-

[1] http://www.celoxica.com/

[2] American National Standards Institute.

[3] Communicating Sequential Processes, CSP, is a formal language used to describe parallel systems created by C. A. R. Hoare in the early 1980s and described in [60].

[4] occam is a parallel processing language designed by a team at INMOS in conjunction with the design of the transputer processor, and based on T. Hoare's ideas of CSP.

chronization of parallel processes. Their existence proved especially useful for several reasons. During the development phase, we could independently debug and validate individual modules given that whatever logical or timing changes happened as the result of an optimization, the module would still fit back into the full design and communicate with its adjacent modules as before. If, in the worst case of a design error, there is some channel mismatch then the whole system freezes and allows the debugger to retrieve state and correct the error. This independent development is especially powerful when one considers that for the full design a compile, place and route cycle is at least half an hour.

Channels also resolve the problem of passing data across clock domain boundaries. The design cannot avoid different clock domains since the PCI interface requires a 33 MHz frequency clock and the MAC interface requires a 25 MHz frequency clock. However having the facility to easily cross clock domains meant that we could choose something close to the optimum frequency for several different tasks. For example, the SDRAM server runs at 125 MHz and the main core at 62.5 MHz. Without this option we would have been unable to meet the design requirements of the project. Again however there are limits. The channel is a complex structure with up to a four-cycle overhead which becomes the limit for very high speed transfers. For the fastest transfer logic we needed to use an internal hardware feature, the dual-port RAM, as shared memory between two clock domains. The shared memory acts as a mailbox while the requests for transfer are still sent over channels. This came at the cost of having to ensure the synchronization with our own logic, an error-prone process that consumed a considerable debugging time.

Although ordered and synchronous operations have clear advantages, there are cases where data has to be retrieved as fast as possible—such as the ingress from the MAC interfaces which must be cleared irrespective of the state of the modules that will consume the data. We used queues in this case to asynchronously accept the incoming data.

The use of multiple channels in each module lead to yet another problem: the arbitration of the access to all these channels. Handel-C provides a solution by means of the `prialt` instruction. This instruction allows the channel that is first ready to perform a transfer and it even works between different clock domains. This instruction was used, for example, in the Packet Data Storage module, to which "free reference" messages are addressed from more sources (from Degradation Emulation Engine on packet discard, and from the Packet Data Forwarder module on packet transmission).

In retrospect the choice of language for this project was fully justified. It

allowed for a formal approach to the design process and we found that with each iteration we moved further from the shared memory constructions and closer to channel-based ones that facilitated both debugging and execution. As we became more competent with using channels and CSP, the code we wrote became smaller and simpler. This leads to the fact that maintenance and modifications are also easier.

5.4 Network emulator architecture

The general architecture of a network emulator is depicted below (see Figure 5.2). The incoming traffic is classified in order to apply different quality degradation to it. Then the experienced quality of each class of traffic gets degraded in a system of queues, according to a degradation model. At the output, packets are scheduled in order to be transmitted to the output port.

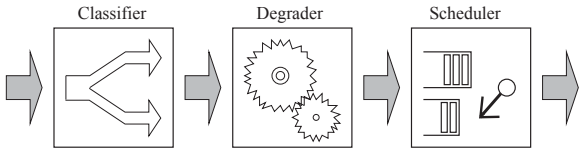


Figure 5.2: Basic network emulator architecture.

In our implementation, the Classifier allows the user to specify up to 16 classification rules. The classification can be performed based on the information contained in the following fields from the packet headers: type of service (ToS), the protocol number, source and destination IP addresses. The input traffic can be classified in eight classes.

The Degrader maintains eight queues, one for each class of traffic. The length of each queue can be configured by the user. The Scheduler can implement any scheduling algorithm. We implemented two scheduling algorithms, that are used in most of the Gigabit Ethernet switches: Strict Priority and Weighted Round Robin.

The emulator system may be seen as a three-level architecture. At the upper level, there is a user interface written in Python[5]. This Command Line Interface (CLI) allows the user to configure the emulator and collect statistics. The user can configure the fixed delay, the sizes of the queues, specify the output rate and upload the classification criteria. The intermediate level is constituted by a Linux kernel module based on the IO_RCC

[5]http://www.python.org/

[22] library developed at CERN, which is the interface between the high-level Python routines and the FPGA. The communication takes place over the PCI. At the lowest level of the system is the FPGA-based hardware platform.

The network emulator is a *packet processor*, with a generic architecture composed of a *packet data path* and a *control path*. This architecture was inspired from the architecture of general-purpose processors, that comprise a data path and a control path [41]. The detailed system architecture is depicted in Figure 5.3. The communication channels between the modules are represented as arrows. The packet data path consists of a storage block (the Packet Data Storage) and receiving/forwarding modules (i.e. the Packet Data Receiver and the Packet Data Forwarder). The control path (in shades of gray) processes the packet references (structures that allow the identification of packet inside the system) by applying different kinds of degradation, like dropping or delaying a packet. It also interacts with the packet path and has the knowledge about the place where packets are stored in memory. The modules of the control path can be easily replaced, due to the channel-based emulator architecture, in order to change the functionality of the hardware platform.

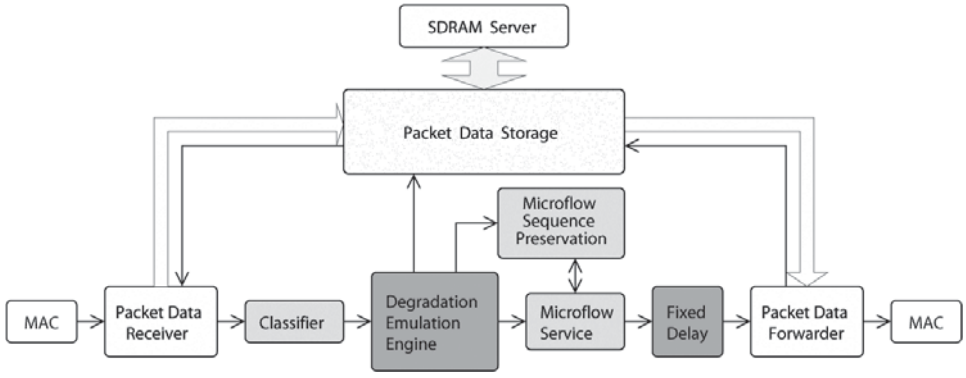


Figure 5.3: The emulator architecture: data (blue) and control (green) path.

The core of the emulator is depicted in Figure 5.4. The Degradation Emulation Engine module maintains eight queues which are serviced in a SP or WRR manner, user configurable. The maximum size of each queue is of 128 packets. This hardware limit can be changed by allocating more on-chip RAM and recompiling the design.

Each queue has a background traffic source associated with it. The source can be started or stopped from the user interface, and the pattern of

each background traffic source can be configured in real-time, by uploading the descriptor tables on the on-chip RAM. The descriptor tables contain information about the size of the background traffic packets and the inter-packet gaps between them.

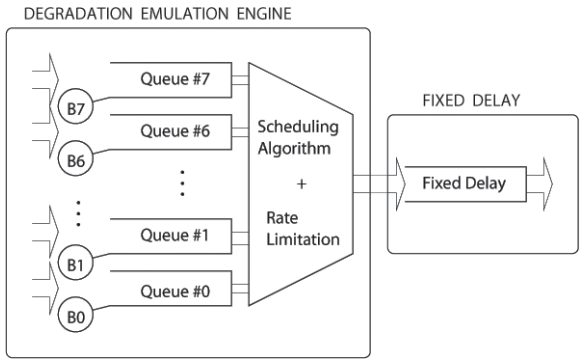


Figure 5.4: The core architecture.

In the next section, each module from the architecture presented in Figure 5.3 is briefly described.

5.5 Module Description

The MAC Receiver. This process provides packet data to the Packet Data Receiver in chunks of 32 bit words. The MAC Receiver is configured and controlled by the Packet Data Receiver. The Packet Data Receiver starts reading 32-bit words from MAC Receiver when the Start of Packet (SoP) event is generated, until the End of Packet Event (EoP) is detected. The Packet Data Receiver also reads from the MAC Receiver the value of the register indicating errors that may occur: CRC errors, alignment errors etc. The MAC Receiver is an IP macro and the source code is not available. Therefore its capabilities cannot be changed.

The Packet Data Receiver. This process manages the reception of complete packets of data from the MAC interface. The technique used to retrieve packets from MAC is polling. The Packet Data Receiver (PDR) maintains two queues–one for storing full packet data and another one for maintaining packet descriptors. If one packet is correctly received, a descriptor (containing the size of the packet and a value indicating how many bytes are valid from the last 32-bit word) is placed in the descriptor queue.

The packet data queue is 32 bits wide and it has 4 K elements. The descriptor queue is 11 bits wide (9 bits for the packet size expressed in 32-bit words + 2 bits for indicating the valid bytes) and it can host 128 descriptors.

As long as there are descriptors in the descriptor queue, the packets that have been received are sent to the Packet Data Storage (PDS) module. A memory address is received upon storage. This memory address is used to construct a packet reference that uniquely identifies the stored packet. The packet reference also contains the following fields from the packet header: the protocol number, the Type Of Service (TOS), the source IP address and the destination IP address. The packet reference is then sent to the Classifier module.

The Packet Data Storage. This process manages the storage of complete packets of data as received from the MAC interface. In parallel, the address of the next available memory slot is calculated. The Packet Data Storage maintains a bit map for indicating which memory slots are free and which are occupied.

The packets are stored in the SDRAM. The available capacity (64 MB) is divided in 32 K slots of 512 32-bit words. Once packets have been received they can be either marked as free (usually as the result of some policing action), or read out (some time later) and then marked as free–this would be the normal outcome for packets that have passed through the degrader.

The Classifier. This process classifies packets based on the fields retained from their header, which are received from the Packet Data Receiver. Once packets have been classified, the corresponding packet reference and the id specifying the degradation that will be applied are sent to the Degradation Emulation Engine (DEE) module.

The Degradation Emulation Engine. This process emulates network degradation as configured by external means. Upon receiving a packet reference and a degrader id, the process determines the appropriate queue id and sends it along with the packet reference to Micro-flow Sequence Preservation for enqueueing. Under certain conditions the decision to immediately free the packet can be taken; in this case no enqueueing takes place. The next queue to be serviced is indicated to Micro-flow Service by its id. A second parallel process implements the rate limiting for the queues in the Micro-flow Sequence Preservation. The rate for each queue can be configured.

The Micro-flow Sequence Preservation. The Micro-flow Sequence Preservation (MSP) module stores and manages in a FIFO manner (according to queue id) the packet references received from the Degradation Emulation Engine module. The Micro-flow Sequence Preservation module maintains two queues–one for the traffic of interest (classified traffic) and one for the unclassified traffic, which is serviced on a *best effort* basis. The size of each queue can be configured. Upon reception of a queue id from the Micro-flow Service module, a packet reference is dequeued from the corresponding queue and returned to this module.

The Micro-flow Service. The Micro-flow Service (MS) module manages the retrieval of packet references from Micro-flow Sequence Preservation based on queue ids received from Degradation Emulation Engine. This is the module that implements the scheduling scheme for the queues of the Micro-flow Sequence Preservation module. The packet reference is subsequently sent to Packet Data Forwarder.

The Fixed Delay. The Fixed Delay (FD) module introduces a constant delay value, by enqueueing the descriptors in a queue. Each time a descriptor is enqueued, the current time is read from a special register called the clock register. A fixed amount, that can be specified in the user interface is added to the time read from the clock register and the sum is also enqueued along with the descriptor. At the output of the Fixed Delay queue a process reads the timestamps and decides when to forward the next packet descriptor.

The Packet Data Forwarder. The Packet Data Forwarder (PDF) module manages the transmission of complete packets to the MAC interface. It maintains two queues–one for full packet data and another one for packet descriptors. When a packet reference is received, the corresponding packet is retrieved from the Packet Data Storage. Upon the retrieving of a packet, a packet descriptor is placed in the descriptor queue.

As long as there are descriptors in the descriptor queues, a parallel process empties the packet data queue. Once packets have been transmitted a "free reference" message is sent to Packet Data Storage.

The MAC Forwarder. This process receives packet data from the Packet Data Forwarder in chunks of 32-bit words. The MAC Forwarder is configured and controlled by the Packet Data Forwarder. As for the MAC Receiver, the MAC Forwarder is also an IP macro and the source code is not available. Therefore its capabilities cannot be changed.

5.6 Implementation details

In this section, some details regarding the memory management, the access to SDRAM and the random number generation are given. These details are intended to highlight the implementation choices that were made due to constraints imposed by the usage of an FPGA. Complexity was traded for fast implementation, as the 1 Gb/s operation requires a processing speed of approximately 1.5 million packets per second.

Different-width mpram ports . Handel-C allows defining multi-ported RAMs that have ports with different width. This feature was used to build queues that have different widths for input and output. The declaration of such a multi-ported RAM is presented below:

```
mpram {
   wom <32> input[ 1024 ];
   rom <64> output[ 512 ];
} qram;
```

This mpram has a write-only port (`wom`), representing the input of the queue, and a read-only port (`rom`) representing the ouput of the queue. For example, in PDR, one queue that has a 32-bit wide input and a 64-bit wide output was used. 32-bit words read from the MAC core are enqueued and then 64-bit words are dequeued in order to increase the bandwidth for the communication with the next module, PDS. The only drawback would be the word padding, which is needed for odd numbered queues, required one clock cycle.

Pipelining. "Pipelining is an implementation technique whereby multiple instructions are overlapped in execution" [17]. In Handel-C, a two-stage pipeline can be easily implemented using the `par` statement, as follows:

```
unsigned  32   register;

read_one_word_from_MAC( register );
while( !end_of_packet )
   par {
      enqueue( register );
      read_one_word_from_MAC( register );
   }
enqueue( register );
```

On the first cycle, one word is read from the MAC and stored in the variable called register. Then until the end of packet, the word previously

read is enqueued and another one is read from the MAC, in parallel. On the last cycle, the last value of the register is enqueued.

Pipelining was used in various places in the emulator, for example in PDR and PDF, to handle data to and from the MAC core. The 32-bit words are read and in the same cycle enqueued in the input queue. Given that each word must be read from the MAC on every cycle, and in the same time enqueued, pipeline was mandatory.

The pipeline for the instructions in a general-purpose processor has to be flushed in case of a jump instructions, which affects the performance. In our design, this does not occur, so the same rate is maintained for the statements that take place in parallel.

The Memory management. The SDRAM is divided in slots of 256 64-bit words (2048 bytes). The address is determined as follows: the index of the slot, multiplied by 256 is added to the base address. This mechanism was preferred for its simplicity, which implies fast logic on the FPGA. The main disadvantage is the poor utilization of the memory space, resulting in internal fragmentation [40]. For example, a 64-byte packet will occupy only 4 words in the 256 word slot, resulting in 99 % waste. A more efficient algorithm could be used, in order to increase the memory space utilization. For example one could divide the available space in two or more classes of slots having different sizes. However, the memory management and the computation of the memory address will have an increased complexity, that will affect the performance.

A bitmap is maintained to determine the occupancy of each slot. One process in Packet Data Storage is responsible to determine the address of the next available slots. It implements the *First Fit* algorithm [40]. The total amount of memory existing on the board (64 MB) allows for 32 k slots. This means that for maximum Ethernet size (1518 bytes) packets a delay of 3.8 s at Fast Ethernet operation or 380 ms at Gigabit Ethernet operation can be emulated. For the minimum Ethernet size (64 bytes) packets the maximum delay that can be emulated is 214 ms and 21 ms for Fast Ethernet and Gigabit Ethernet, respectively.

The SDRAM access. The access to the SDRAM was implemented using a Client/Server architecture, with one server and two clients. The architecture is depicted in Figure 5.5.

The two clients (the read and write clients) use two dual-port on-chip RAMs of 8 words, to temporarily store the data which is read/written from/to the memory. Those dual-port RAMs are alternatively used in a

double-buffering algorithm to better interleave the read and write operations. The clients initiate a memory transfer by sending a read/write request to the server. When a transfer finishes, the server send an acknowledgment to the client that requested the transfer. The two SDRAM controllers are configured to operate with burst of 8 words, the maximum available length for a burst. Due to use of channels and the client/server architecture, the SDRAM server is elegantly implemented in Handel-C.

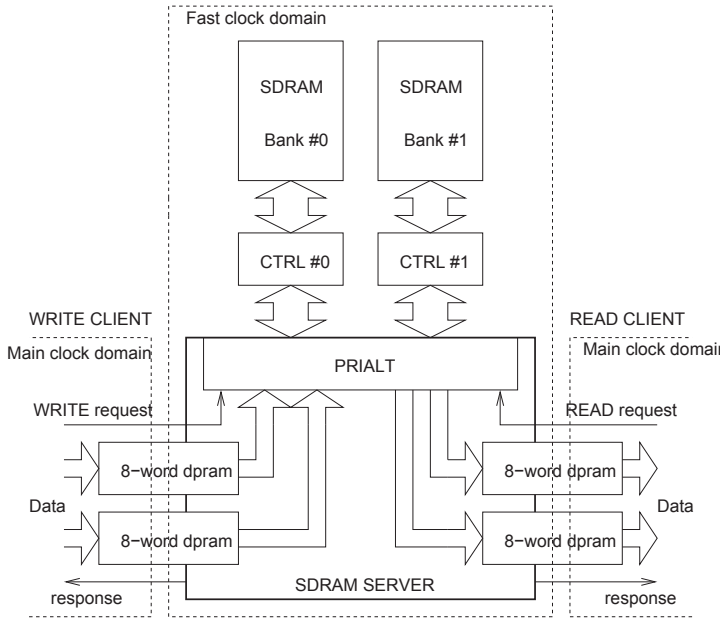


Figure 5.5: The SDRAM server architecture.

The random number generation. The generation of random numbers is a key element for any hardware implementation of stochastic algorithms. The paper [4] contains a review of the existing hardware implementations of random number generators (RNG). The most common type of RNGs used in cryptography is the Linear Feedback Shift Register (LFSR) described in [5]. The LFSR used in our network emulator was implemented as a macro procedure in Handel-C:

```
macro proc rand(Variable) {
  do {
    Variable = (Variable<-21) @ (Variable[12] ^ Variable[30]) @ \\
               (Variable[10:1]^Variable[29:20]);
  } while (1);
}
```

A feedback shift register consists of two parts: a shift register and a feedback function. The feedback function for a LFSR is a exclusive or (XOR) of certain bits in the register. Combining this uniformly distributed random number generator with look-up tables in the on-chip RAM, it is possible to implement any given distribution. The tables are loaded over the PCI with values generated on the host PC.

5.7 Implementation validation

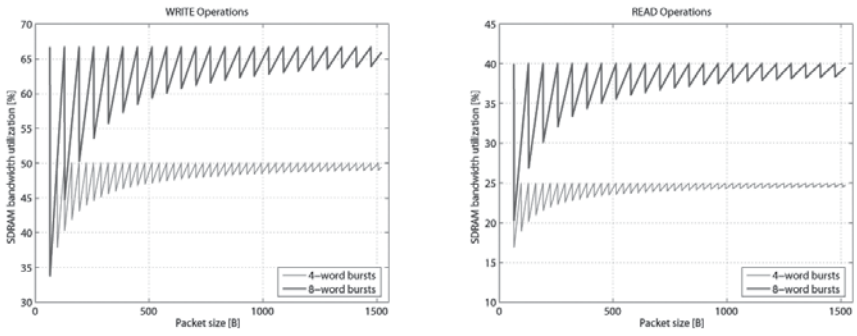
We performed various tests for the validation of our hardware network emulator. We analyzed the performance of our implementation from several points of view: memory fragmentation, the SDRAM access, achieved maximum packet rate as a function of packet size, as well as the performance of the scheduling algorithm implementation.

Because of the use of slots of 2048 bytes, internal fragmentation of the SDRAM memory occurs. Consequently, the average slot utilization is of only 38.6 %. One optimization could be the use of different size slots, at different base addresses. For example, if slots of 512 bytes would be used for packets with sizes between 64 and 512 bytes, and 2048-byte slots for packet sizes larger than 512 bytes, the average slot utilization would increase to 51.54 %. More complex memory allocation could be imagined, but this can only be achieved to the detriment of execution speed.

The SDRAM controller can be configured to operate in two modes: burst and page mode. Burst mode was preferred, so that the read and write operations can be interleaved. Operation interleave allows storing and retrieving packet data to/from SDRAM, regardless of the size of packets. The use of a larger transfer granularity such as page mode is more efficient from the point of bandwidth utilization, but causes blocking between input and output that is unacceptable.

The length of one burst can be 1, 2, 4 or 8 words. According to specifications of the SDRAM controller, the overhead for a write operation is of 4 cycles, while for read operations the overhead is 12 cycles. After the address setup, it takes one cycle to read/write one word.

In our implementation, the SDRAM operations take place in bursts of 8 words. In Figure 5.6 the SDRAM bandwidth utilization is presented, for 8-word and 4-word burst length, as function of packet length. It is clear that higher bandwidth utilization is obtained when the length of burst is 8 words. The ideal SDRAM bandwidth utilization would be achieved when one cycle per word would be used for read or write operation.

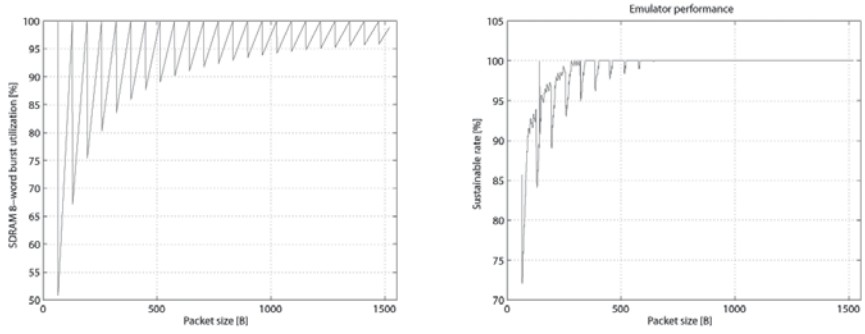


(a) WRITE ops. (b) READ ops.

Figure 5.6: SDRAM bandwidth utilization for WRITE and READ ops.

For example, for a 65-byte packet, two write burst operations are needed– one for storing 64 bytes and the second one to store the 65th byte. Because another 8 words are written for only one more byte, the efficiency of the operation drops to 51 % (see Figure 5.7).

We start now showing the results of a series of performance and validation tests. Our network emulator will be compared against another emulator in order to demonstrate the realism of the our approach to network quality degradation emulation. We propose a set of two tests to be performed for comparison of the two emulators.



(a) (b)

Figure 5.7: The 8-word burst utilization and sustainable packet rate vs. packet size.

Using the network tester described in [21], we experimentally determined the performance of our hardware implementation from the point of view of

the sustainable packet rate. In Figure 5.7 the sustainable rate expressed in percentages of the 1 Gb/s line speed is presented as function of the packet size. Note that for most packet sizes the implementation is capable to sustain line speed. For packets smaller than 512 bytes, the performance is usually larger than 85%, except for the 65-byte packets when the performance droppes to approximately 73%.

This uneven performance is entirely due to the limitations of the interface to the SDRAM memory, which represents the bottleneck of our system. An increase in performance proportional to the increase of the frequency of the SDRAM Server clock domain was observed. However, the overall performance is satisfactory given that short bursts can be dealt with, depending on the size of the internal queues of the emulator. The interface to SDRAM can be further improved in order to achieve line speed even for packets smaller than 500 bytes, if the SDRAM access would be redesigned. However this only becomes an objective for production implementations.

We shall demonstrate that our approach to the emulation of network quality degradation offers the possibility of reproducing realistic network scenarios. For this, we propose two sets of tests that were peformed in order to realize the comparison between our network emulator and a representative off-the-shelf emulator. The proposed tests are: (i) packet sequence preservation, or out-of-order test, and (ii) delay auto-correlation test for consecutive packets.

For comparison, we chose the NIST Net network emulator (see Chapter 2). We consider it to be representative for the currently existing network emulators for two reasons: it is implemented in software, and it allows the loss and the delay to be independently set.

We configured our emulator to limit the output rate at 10 Mb/s and to classify all the input traffic into only one queue. The input traffic was UDP, generated by an application running on a PC. The average load was 5 Mb/s and the traffic pattern constant bit rate. The same queue is shared by the background traffic. One source of background traffic sent Poisson traffic with negative exponential departure times, at an average load of 5 Mb/s. The background traffic was a mixture of 70% 64-byte packets, 15% 576-byte packets and 15% 1518-byte packets, representative for the Internet traffic [61], [62].

In this scenario we measured the delay experienced by the application foreground traffic. There was no loss during the experiments. The same measured average of 14 ms and the same variance of 2.5 ms were set for the delay to be introduced by the NIST Net network emulator; the same foreground traffic was generated through NIST Net. The following observations

were made. When using NIST Net, the packets are re-ordered and the original packet sequence is not preserved. This unrealistic reordering has strong negative effects on application performance. In our network emulator, the order of the packets at the output of the emulator is always the same with the one at the input. The average delay auto-correlation for consecutive packets measured for the delay traces obtained with NIST Net was smaller than the same average measured on delay traces through our network emulator. This happened consitently throughout several experiments. We conclude these two tests show that the delay introduced by our network emulator has a more realistic variation, with correlated consecutive values. The correlation comes from the fact that the original packet order is preserved by means of a queue. The variation of the delay is the natural consequence of the competition between the foreground and the background traffic.

We conclude that the emulator is able to sustain line speed for most of the packet sizes. Its performance is satisfactory given the 85% sustainable rate for minimum size Ethernet packets. Comparing our network emulator against the NIST Net network emulator showed that the quality degradation introduced by our solution is more realistic, from two points of view: (i) the preservation of original packet sequence; and (ii) the delay auto-correlation for consecutive packets.

5.7.1 Scheduling algorithm performance assessment

We present the test results of our scheduling mechanisms implementation. We show that our implementation of Strict Priority and Weighted Round Robin scheduling algorithms is in accordance with the theoretical behaviour of those mechanims.

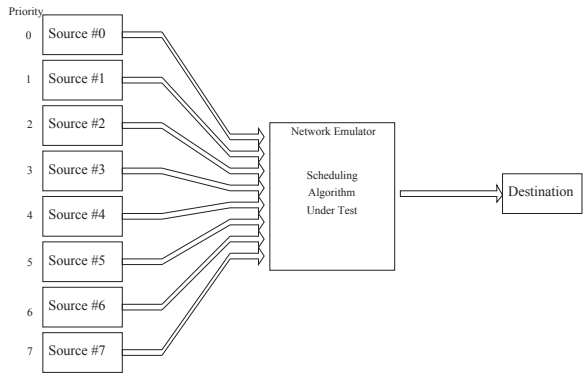
The degradation induced by the scheduling mechanisms implemented in the emulator was assessed. The setup we used for our experiments is shown in Figure 5.8. With this apparatus we measured the quality degradation experienced by different traffic flows when scheduling mechanisms are deployed. We drove the system of queues inside the Degradation Emulation module into saturation, which is the point where quality degradation occurs.

The traffic flows in our tests were generated by eight sources that send packets to the same destination. All packets from a source have the same priority, marked in the ToS field of the IP header. Priorities are distinct between sources. The traffic we used to drive the system into saturation was CBR or Poisson (i.e., traffic with a negative exponential inter-packet gap distribution). The CBR traffic is in accordance with the recommended Constant Load traffic for benchmarking Ethernet devices [23], which is a

traffic with a constant inter-packet gap and a constant packet size. The traffic was CBR, with 64-byte packets. It was classified based on the ToS field from the IP header, according to the mapping presented in Figure 5.8.

ToS	Class	Priority
0x08	0	0, 1
0x18	1	2, 3
0x28	2	4, 5
0x38	3	6, 7

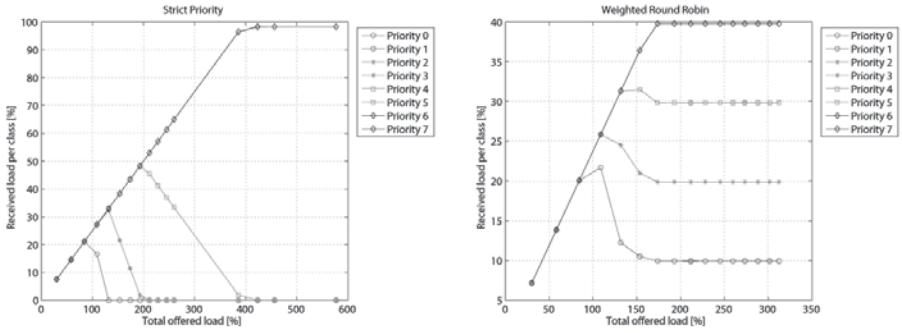
(a) mapping



(b) setup

Figure 5.8: Scheduling mechanism test setup.

Our SP implementation behaves according to the ideal model presented in Chapter 4. The tests were performed according to the methodology we proposed in [36]. Results are presented in Figure 5.9. Note that the behaviour is the same as the expected one (see Figure 4.12). Our implementation of WRR works on a per-packet basis, i.e. the weights represent the actual ratios between the number of packets service from the queues. In our tests, the weights were: 4, 3, 2, and 1. The results are presented in Figure 5.9. Note that the behaviour is the same as the expected one (see Figure 4.12).



(a) SP

(b) WRR

Figure 5.9: Received load as function of offered load.

Both scheduling algorithms behave in accordance with the theoretical behaviour. This allows us to emulate in a realistic way order 2 networks with up to eight classes of traffic.

Chapter 6

Application QoE assessment

In this chapter we present the results of several experiments that are aimed at assessing application performance and QoE through the use of the network emulator. For the assessment of application QoE, we analyze the behaviour of three representative and widely-used applications: web browsing, VoIP and video streaming. The former uses the HTTP protocol, based on TCP, therefore the traffic it generates is elastic, while the latter two have an inelastic traffic, being based on UDP. The setup we developped and used to assess the applications' QoE is illustrated in Figure 1.4 and intimatelly described in [35] and [64].

6.1 Web browsing

Web browsing is an HTTP-based application, characterized by short-lived TCP transfers, which makes it very sensitive to packet loss. For emulation, we used the approach of background traffic generation (see Section 3.2), thus the traffic of interest (HTTP) competes with the background traffic generated inside the Degradation Emulation Engine to occupy queue space (which induces loss) and for being serviced (which induces delay) - see Figure 3.3. We compare two cases, when the background traffic source has a CBR or a Poisson pattern. For all the tests, the emulator was configured to introduce a fixed delay of 12.5 ms in each direction, equivalent to a 25 ms RTT. The available bandwidth was limited to 10 Mb/s and the size of the queue was 128 packets.

In the setup presented in Figure 1.4, the end PCs run Linux with kernel 2.4.21, the HTTP server was Apache[1] 2.0 (httpd-2.0.46) and the client was `wget` (wget-1.8.2), a non-interactive network retriever that allows the

[1] http://www.apache.org/

automation of tests. The interconnection used Fast Ethernet, because the taps used work at 100 Mb/s. For the Apache server all the parameters had default values, including the Timeout[2] of 300 s. The KeepAlive[3] parameter was "on" and "off" in turn. When "off", a new TCP connection is open and closed for each file transfer. This represents the most inefficient case. When "on", the same TCP connection is used for up to MaxKeepAliveRequests = 100, separated by no more than KeepAliveTimeout = 15 s. The load generated by the background traffic source was varied from 0 to 100% of the available output rate, which was set to 10 Mbps.

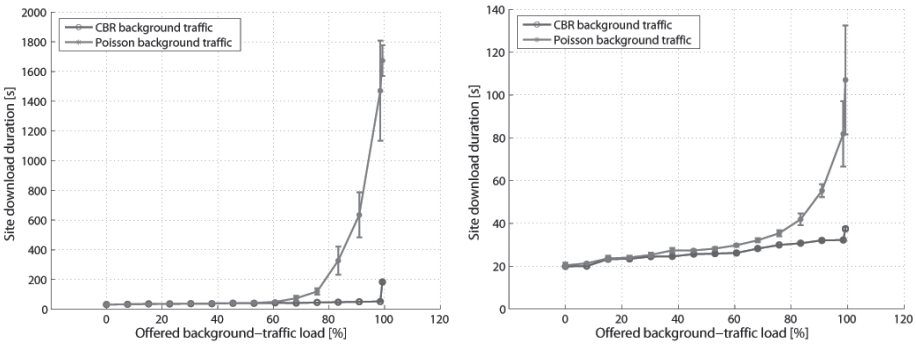
We chose a representative web page to use in our tests, that contains both images and text, consisting of 499 files, with a total size of 1.6 MB. The average file size is approximately 3 kB, representative as the average value of the file sizes on the Web [12]. For a test without loss, we measured the instantaneous rate at the server, with an averaging interval of 0.1 s. We noticed the resemblance between the histogram of instantaneous rate and the histogram of file sizes: small files are sent at low rates, while larger files are sent at higher rates, given the window-based rate control of TCP. The KeepAlive was off for this test, so for each file to be transferred a new TCP connection was initiated.

For the assessment of QoE for web browsing we measured the site download duration [37]. The results in Figure 6.1 show the dependency of site download duration on the offered background traffic load, for KeepAlive "off" and "on", respectively. Note that the influence of the CBR background traffic on the performance is not significant (the download duration doubles from 0 to 100% background traffic load); the explanation is this case is equivalent to a constant diminution of the bandwidth available for the application. This is not important since the foreground traffic only uses approximately 0.5% of the available capacity. When the background traffic load approaches 100 %, there is a rapid increase of the download duration followed by denial of service, leading to complete application failure.

When the background traffic is Poisson (thus more realistic) noticeable performance degradation starts occurring from loads of 60 %. At loads larger than 80 % the degradation becomes significant and reaches values with more than one order of magnitude, compared to the CBR case. The intrinsic burstiness of the Poisson traffic determines the different shape of the graphic, and the fact that different Poisson distributions were used explains the larger standard deviations of the results.

[2]Timeout: The number of seconds before receives and sends time out.

[3]KeepAlive: Whether or not to allow persistent connections (more than one request per connection).



(a) KA off (b) KA on

Figure 6.1: Site download duration vs. background traffic offered load.

One can observe the improvement of the site download duration by one order of magnitude when the same TCP connection is maintained open for the transfer of all the files of the site. Opening a TCP connection for each file to be transferred required more time for each connection establishment, and is affected more by loss (losing one control packet implies the setup of a timer—usually of 3 seconds—while one data packet which is lost is retransmitted after one RTT—usually in the order of tens of milliseconds). Denial of service occurs when the background traffic rate exceeds 100 % of the available bandwidth, leading to complete application failure.

For the representation of the results the loss and delay coordinates could be used instead of the background traffic load. However, in the case of TCP-based traffic like HTTP, the loss at network level is hidden by the retransmission mechanism to the application. It can be measured, for example by means of taps, but the dependence of loss on the background traffic load is not linear and therefore the graphs would be more difficult to interpret. As for the delay, the variable delay induced in the emulator is of order of hundreds of μs, compared to the fixed delay of 25 ms. Therefore a measure like the average would not be conclusive.

6.2 VoIP

Wireless networks and VoIP applications are two major coordinates of the telecommunications landscape. Wireless networks are used everywhere, from home users to industrial applications. VoIP had a long adoption curve compared to wireless networks, but it represents the present and future of the

telephony systems. The idea of using IP technology for voice transport was proposed in the 70s, though the routing technologies of that time had neither the capacity to support voice traffic nor the required quality of service capabilities. The interest in the wide area packet-based voice technology quickly shifted from the enterprise market to consumer VoIP applications like Skype, where IP technology could provide cheap phone calls for those who were interested in price rather than quality. In the enterprise space, the focus shifted to local IP voice in the form of an IP PBX [65]. The essential idea was to eliminate the separate, stand-alone PBX system that had existed for almost 100 years and move the voice switching function on to the LAN switch infrastructure, which can be wireless even. The wireless VoIP has a major drawback: the lack of security [66]. Therefore encryption should be used in order to protect the privacy of the VoIP calls. Encryption introduces an additional computation that represents an overhead, leading to a diminished bandwidth and increased delay that are not desirable for an application [67].

Several metrics are widely-used for measuring QoE for VoIP applications. ITU has defined standards that allow an evaluation of the quality of voice communication: MOS (Mean Opinion Score) [68], PSQM (Perceptual Speech Quality Measure) [69], the E-Model [70], PAMS (Perceptual Analysis/Measurement System) [71] and PESQ (Perceptual Evaluation of Speech Quality) [72]. The first of them (MOS) was a subjective metric, but successive attempts have been made to define objective metrics as well. For our experiments we used the PESQ score, which represents the most advanced objective metric for measuring speech quality.

There exist several studies focused on the performance of a VoIP application in wireless networks [38]. Our work aims at quantifying the cost of enabling encryption for the wireless LAN, in terms of voice signal quality drop. We experimentally determined the maximum number of VoIP connections that can be performed in parallel and the quality of the voice signal as a function of the number of calls. The maximum quality levels are achieved when the wireless LAN is not secured and those values will be considered as reference. Our study makes it possible to determine the conditions for which the wireless technology is suitable for VoIP telephony systems.

The experimental setup is depicted in Figure 6.2. It comprises three computers: two for generating the VoIP traffic (left) and one to receive the VoIP calls (right). The senders are equipped with two wireless network cards of 54 Mbps and they talk to a wireless router (2.4 GHz-802.11g) with Fast-Ethernet connections. The router is connected to the destination computer by a UTP Fast Ethernet connection. All the computers have an alternative

Fast Ethernet connection to a main switch, used to implement the so-called control network, which is not depicted in Figure 6.2. This interface was used to send/receive Linux commands/responses during the test, in order to avoid loading the wireless interface with other traffic than VoIP and also to have quick and reliable answers from the computers. In this way we implemented a control network apart from the wireless network used for the VoIP experiments.

All talkers send VoIP traffic using the UDP. The wireless router becomes the bottleneck of our wireless network, when the router reaches the limits of its hardware resources. For the tests we used a wave file recorded at 8 kHz with 8 bits per sample, resulting in a data rate of 64 kbps. This voice-coding scheme is standardized in ITU Recommendation G.711 [56]. We used a freeware VoIP application Speak Freely [73] v.7.6a for Linux which sends voice data over the network using a certain encoding, and ensures decoding and playback at the receiving end. The software implements a series of codecs that are all available using the built-in protocol: G.711, G.726, GSM, LPC, LPC-10, CELP. For the tests we used the G.711, G.726 [74], GSM [75] codecs: G.711 is the codec that needs the largest bandwidth (64 kbps) and offers the best quality (maximum PESQ score of 4.5).

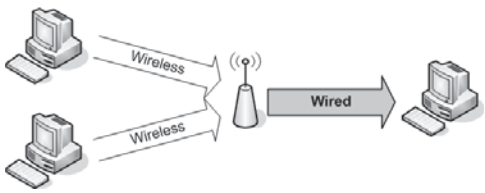


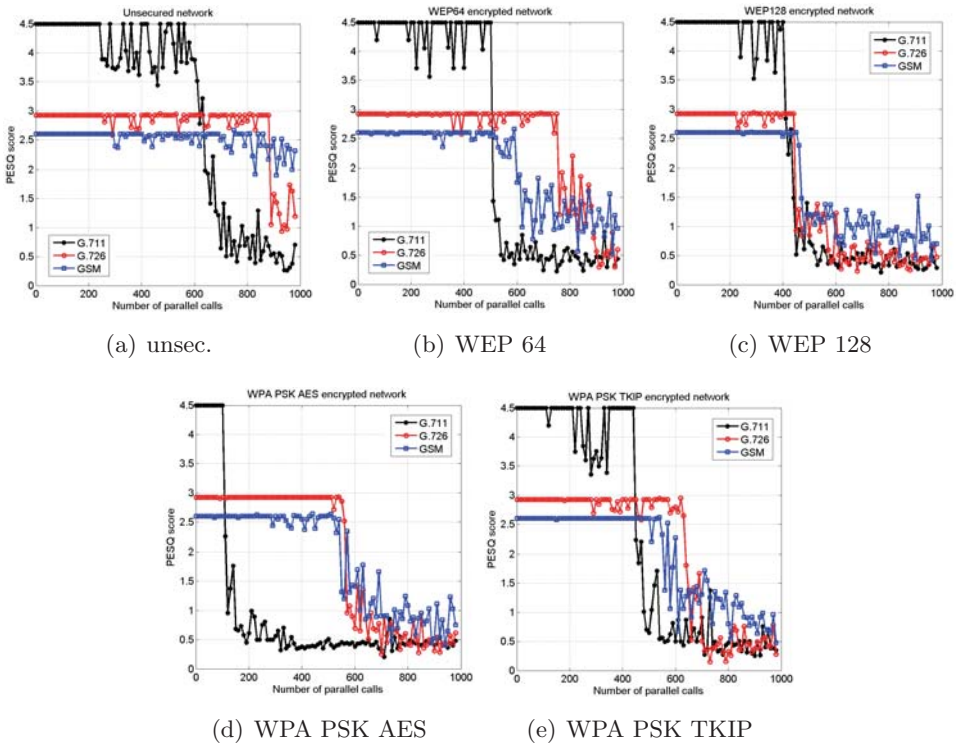
Figure 6.2: Experimental setup.

The testing procedure was automated, all the commands being sent from a control PC via preconfigured ssh sessions, in order to avoid human interaction. At startup, there is only one VoIP call initiated, in the best quality transmission conditions of the given test network. The control script runs in a loop and with each iteration another 10 VoIP parallel calls are added. One of the calls was sent with the Speak Freely application (automatic using an audio file), the rest of the calls were emulated using home-made software applications (by generating UDP traffic using the same packet size as the codec we used G.711, G.726 or GSM). The call that was received with the VoIP application was recorded in an audio file. The loop ends with a maximum of 1000 parallel VoIP calls. Each resulting wave file was compared with the original one. For the software sound comparison we used the PESQ (Perceptual Evaluation of Speech Quality) metric [72].

The tests were repeated five times. The points of the plots depicted below were obtained by averaging the results from these experiments. Once configured, the parameters of the VoIP application, i.e. chosen codec, generated throughput, packet size and inter-packet time remained the same for the entire duration of the particular test run. We compared the performance of the VoIP application in five encryption scenarios, depending on the type of the security enabled for the wireless network: no-encryption at all, WEP 64, WEP 128, WPA PSK AES and WPA PSK TKIP [13]. The results are presented in Figure 6.3a. We can observe the differences regarding the maximum value of the PESQ score, values less that 4.5 for G.726 and GSM as a consequence of the loss of information due to compression. We recommend using the G.711 in order to obtain a maximum of quality of the audio signal. We must take into consideration the fact that if we exceed 650 calls in parallel the quality problems will rise very fast. As an alternative, one can use also the G.726 codec that has constant results (of maximum PESQ score) until a number of 880 VoIP calls in parallel is reached. The codec with the smallest data rate (from our experiment), GSM , has a constant evolution, similar to G.726. The signal quality modifies only at the end of the testing interval.

For the WEP64 encryption technique the maximum number of VoIP calls at good quality can be 500 in the case of the G.711 codec, 750 for G.726 and approximately 600 for GSM (Figure 6.3b). A good quality / number of users ratio is provided by G.726 when using a WLAN with WEP64 encryption. We observe the fact that although GSM codec has a much smaller data rate than G.711 or G.726, it does not provide good results. It would be expected to offer at least a result similar with G.726 or even better. WEP 128 encryption is not recommended for a network dedicated to VoIP calles, because of the PESQ score values that are fall off rapidly compared to other encryption standards (stronger encryption schemes have better results). A reason for this conclusion is the large processing overhead. For the WEP 128 scenario, the maximum number of good-quality VoIP calls in both cases is comprised between 400 and 450 regardless the codec (Figure 6.3 c.).

The codec that provides the best voice quality, G.711, in a WPA PSK AES secured network allows a maximum of only 100-110 VoIP calls in parallel (Figure 6.3d.). In this case the combination best encryption best quality of the audio signal produces the most unfavorable results regarding the number of total VoIP calls in parallel on the same access point. A possible solution is to use the same codec G.711 for the best audio quality in conjunction with a WPA PSK TKIP encryption. In this way, we can make a compromise between the quality of the audio signal, security and



(a) unsec. (b) WEP 64 (c) WEP 128

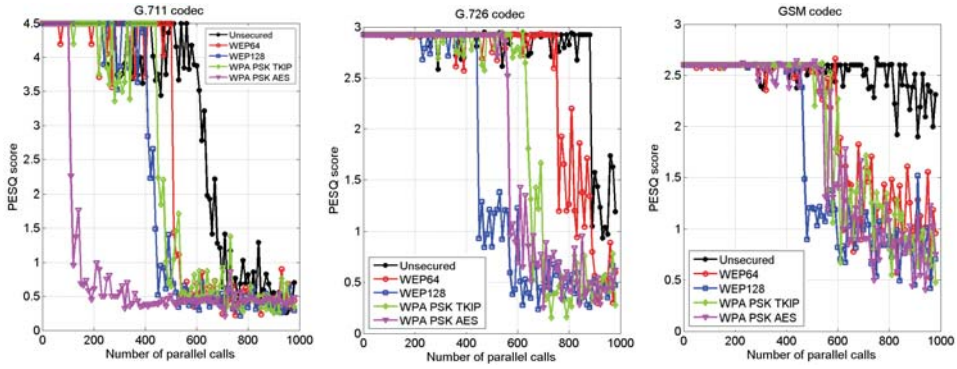
(d) WPA PSK AES (e) WPA PSK TKIP

Figure 6.3: Codec comparison for the five encryption scenarios.

the maximum number of VoIP users in parallel. By using this solution we can have an enhancement by four times the number of VoIP calls then when use used the WPA PSK AES algorithm. In the WPA TKIP scenario, when the G.726 codec is used, a maximum of 620 VoIP connections in parallel with a good quality of the audio signal is achieved. In this case the GSM codec does not provide a great performance because there are just 550 VoIP calls in parallel (see Figure 6.3e.). As expected G.711 a great bandwidth consumer (comparative to the other two codecs that we analyzed) does not have an extraordinary performance, providing only 450 calls at a good quality. The same codec used in a WEP64 and WEP128 encrypted network provides almost similar results 500 and 450 calls in parallel, respectively. The G.726 and GSM codecs have good performances with this type of encryption compared to G.711. The two codecs used with the TKIP algorithm offer almost the same number of VoIP calls in case of the AES encryption, so we recommend that in practice to use the AES algorithm in order to have the best security.

In Figure 6.4 we present the PESQ score as a function of the number of parallel VoIP calls, for each of the three codecs and each of the five

encryption scenarios.



(a) G711. (b) G726 (c) GSM

Figure 6.4: Encryption comparison for three audio codecs.

For the G.711 codec, the best ratio quality number of users is obtained in the case of using an unsecured network, 610 VoIP calls with good and very good audio quality of the speech. Between 610-640 calls we observe a score that is satisfying for some users and from that point the quality is unacceptable. The encryption offered by WPA PSK AES algorithm provides the maximum security, but with the price of about 100 possible VoIP calls in parallel. WPA PSK TKIP encryption represents a good alternative when an optimum ratio between security and the number of users is desired, so in this case we can have up to 450 VoIP calls in parallel at good quality on the same access point. If we compare the TKIP and the AES technologies, we observe that we can have up to four times more calls with TKIP. For the WEP standards, the differences consist in only 100 calls (500 for WEP64 and 400 for WEP128).

For the G.726 codec the PESQ score starts from 3 because of the quality that is lost due to the coding process. The maximum performance is reached in the case of using an unsecured network, with a number of 880 calls in parallel. The high security provided by the WPA PSK AES or TKIP techniques allows for 560 respective 630 VoIP calls at good quality, due to the small data rate of the codec. The WEP128 encryption mechanism is not recommended in this case because it has a low performance (maximum 450 VoIP calls). This result comes from the fact that the 128 bits used for encryption produce an important overhead. The WEP alternative on 64 bits ensures 750 VoIP calls in parallel at good quality. Using the WEP64 algorithm and G.726 we obtain a good solution if we need a minimum security and a great number of VoIP calls.

GSM is the codec with a low transmission rate of just 13 kbps, but unfortunately with a maximum PESQ score of approximately 2.6. The unsecured network offers good results (800 calls). Between 800 and 1000 the quality decreases, so the users are not satisfied with a PESQ score between 2 and 2.6. The WEP128 allows only 450 VoIP calls in parallel, while WEP64 gives us a maximum of 600 connections in parallel using the same router. For the systems where security is a key factor is recommended to use the WPA technology: approximately 500 users can have a VoIP call in parallel at acceptable quality, both for AES and TKIP. The difference between WEP64 (590 calls) and WPA (550 calls) is insignificant, so its recommended, where the hardware supports, to use the WPA encryption standard. Despite the smaller rate than G.726 (2.5 times), the GSM does not have the same performances (or better), in the case of an encrypted network.

We experimentally determined the maximum number of good quality VoIP calls that can be initiated in parallel in an encrypted wireless network. The drop points in the curves representing the results clearly indicate the exact moments when the congestion occurs in the experimental network. Using our results one can configure a wireless network by choosing the appropriate combination of codec / encryption algorithm in order to obtain the desired maximum number of parallel VoIP calls at good or acceptable quality.

6.3 MPEG-4 video streaming

Video streaming is a very demanding application in terms of network services, due to its real-time characteristics. The amount of bandwidth it requires depends on the compression rate, and implicitly on the quality of the video signal. Video streaming also requires high reliability for the data transfer between the streaming server and the client. The MPEG[4] standards describe the most frequently used compression algorithms for video sequences. The high compression rate MPEG-4 achieves reduces the required bandwidth for a video streaming application, which makes the application vulnerable to any packet loss at network level. MPEG-4 is intended to be the standard for videoconferencing quality at extremely low bit rates (from a few to a few dozen of Kb/s) [14].

In case of an MPEG video-streaming application, the server packetizes the MPEG stream in order to send it to the client that requested it. The protocol used for the control of the streaming is RTSP (Real-time Streaming

[4]Motion Picture Experts Group, http://www.mpeg.org/

Protocol) or RTP (Real-time Protocol). Details on the RTP payload for MPEG-4 streams can be found in [24] and [25]. The transmission protocol used by the application under study is User Datagram Protocol (UDP). UDP traffic is an inelastic, i.e., the application doesn't adjust its transmission rate to network conditions. In addition, lost packets are not retransmitted. Therefore, packet loss at network level will cause gaps in the MPEG video stream.

The MPEG-4 streaming server used in our experiments was the Helix[5] streaming server from Real Networks[6]. The free version has a limitation of 2 Mb/s for the output transmission rate. However, a quality equivalent to that of VHS [7] videocassette only requires a transmission bit rate in the order of 1.5 Mb/s [14]. The MPEG-4 client `mpeg4ip`[8] was modified so that every video frame that is rendered on screen is saved on disk as an individual bitmap file. Based on these files, one can recreate the video sequence at reception, as it would have been seen by a potential user. Using QoE metrics, we then compute the user-perceived quality for those video sequences. What follows is the correlation of the measured network quality degradation (QoS) with the calculated QoE.

There are various metrics for the assessment of image and video quality [76]: (i) *full-reference* or *reference-based*, when both the video sequence at the transmitter and the video sequence at the receiver are available, then the sequence at receiver is compared to the original sequence at transmitter; (ii) *no reference* or *without reference*, when the video sequence at the transmitter is not available, therefore only the video sequence at the receiver is being analyzed; and (iii) *reduced-reference* [77] [78] which are based on the sequence at the receiver and on some features extracted from the original signal at the transmitter. This is the case of the fractal measures we propose below.

For the quality assessment of an image or a video sequence, the metrics can be also divided into *subjective* and *objective*. During the last decade, several quality measures, both subjective and objective, have been proposed, especially for the assessment of the quality of an image after lossy compression, image rendering on screen or for digital cinema [79]. Most of them use models of the human visual system to express the image perception as a specific pass band filter (to be more precise, a pass band filter for the

[5] https://helixcommunity.org/

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

[7] The acronym for Video Home System, a recording and playing standard for video cassette recorders, developed by JVC and launched in 1976.

[8] http://mpeg4ip.sourceforge.net/

achromatic vision and a low pass filter for the chromatic one) [80]. In this chapter we explore a well-known property of the human visual system, i.e. to be *sensitive* to the visual complexity of the image. We use fractal features [81] to estimate this complexity, relying on the hypothesis that the fractal features are capable of characterizing the image complexity in its whole, i.e. both the space-frequency complexity and the color content.

First, we propose two objective reference-based metrics [34] for the assessment of the user-perceived quality for video streaming applications: the number of dropped video frames (NDF) and the number of altered video frames (NAF). NDF is computed as the difference between the number of frames in the original video sequence at transmitter and the number of video frames that are effectively rendered at receiver. NAF indicates how many frames—from the ones received and rendered—are affected by impairments.

The most complex metrics are based on models of the human visual system, but some of them are now classical signal fidelity metrics like the signal-to-noise ration (SNR) and its variant peak SNR (PSNR), the mean squared error (MSE) and root MSE (RMSE) which are simply distance measures. These simple measures are unable to capture the degradation of the video signal from a user perspective [82]. On the other hand, the subjective video quality measurements are time consuming and must meet complex requirements (see the ITU-T recommendations [26], [27], [28], [29]) regarding the conditions of the experiments, such as viewing distance and room lighting. However, the objective metrics are usually preferred, because they can be implemented as algorithms and are human-error free.

The Video Quality Experts Group (VQEG)[9] is the main organization dealing with the the perceptual quality of the video signal and they reported on the existing metrics and measurement algorithms [32]. A survey of video-quality metrics based on models of the human vision system can be found in [83] and several no-reference blockiness metrics are studied and compared in [33]. A state-of-the-art of the perceptual criteria for image quality evaluation can be found in [84]. OPTICOM[10] is the author of one metric for video quality evaluation called "Perceptual Evaluation of Video Quality" (PEVQ), which is a reference-based metric used to measure the quality degradation in case of any video application running in mobile or IP-based networks. The PEVQ Analyzer [85] measures several parameters in order to characterize the degradation: brightness, contrast, PSNR, jerkiness, blur, blockiness etc. Some of the first articles that proposed quality metrics inspired by the human perception [86] [87] drew also the attention

[9] http://www.vqeg.org
[10] http://www.opticom.de

on some of the drawbacks of the MSE and the importance of subjective tests. Among the unanimously-accepted metrics for the quantification of the user-perceived degradation, the ones proposed by Winkler use image attributes like sharpness and colorfulness [30] [31], [88]. In [89] the authors propose a no-reference quality metric also based on the contrast, but taking into account the human perception and in [90] the hue feature is exploited. Wang proposes in [91] a metric based on the structural similarity between the original image and the degraded one. The structural similarity (SSIM) unifies in its expression several aspects: the similarity of the local patch luminances, contrast and structure. This metric was followed by a more complex one, based on wavelets, as an extension of SSIM to the complex wavelet domain, inspired by the pattern recognition capabilities of the human visual system [92]. Together with Wang, Rajashekar is the author of one of the latest image quality metric based on an adaptive spatio-chromatic signal decomposition [93] [94]. The method constructs a set of spatio-chromatic function basis for the approximation of several distortions due to changes in lighting, imaging and viewing conditions. Wavelets are also used by Chandler & Hemami to develop a visual signal-to-noise ratio (VSNR) metric [95] based on their recent psychophysical findings [96] [97] [98].

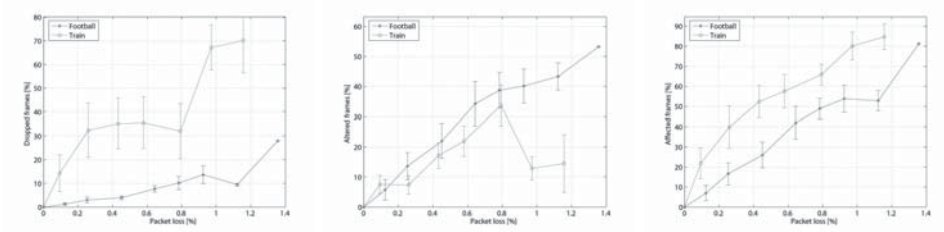
Most of the existing metrics for the video quality are used to quantify the degradation introduced by the compression algorithm itself, as a consequence of the reduced bit rate. We are interested in objectively assess the degradation in video quality caused by the packet loss at network level [99]. In our experiments, we identified two kinds of degradation: (i) the degradation that affects the sequence, i.e. the temporal component of the signal and (ii) the degradation that affects the frames, i.e. the spatial component.

The degradation that affects the video frames (see Figure 6.6) is a mixture of several impairments, including blockiness and occurrence of new colors. The modifications of the image content reflect both in the color histograms—a larger spread of the histogram due to the presence of new colors—and the spectral representation of the luminance and chrominance (high frequences due to blockiness, see [100]). Given all the above considerations, metrics like blur, contrast, brightness and blockiness are not fully able to reflect the degradation, thus they cannot be applied for such degraded video frames. Metrics able to capture all the aspects of the degradation that reflect the color spread—the amount of new colours occurring in the degraded video frames would be more appropriate. We therefore consider that the approaches based on on multiscale analysis and image complexity are more adapted to the video quality assessment, because vision is a complex process that integrates multiple aspects of an image: the spatial

frequencies and the topology, as well as the color.

Fractal analysis-based approaches offer the possibility to synthesize into just one measure adapted to the human visual system, all the relevant features for the quality of an image (e.g. colourfulness and sharpness) instead of analyzing all image characteristics independently and then to find a way to combine the intermediate results. Due to its multi-scale nature, the fractal analysis is in accordance with the spirit of all multi-resolution wavelet-based approaches mentioned before, which unfortunately work only for gray-scale images. Therefore, one of the advantages of our approach would be the fact that it also takes into account the colour information. In addition, the fractal measures are invariant to any linear transformation like translation and rotation.

In the experiments we performed, prior to the development of the emulator presented in this book, we introduced artificial packet loss using the NIST Net network emulator [15]. Packet loss was introduced in the server-client direction, with values between 0 and 1 %. We ran the tests using various MPEG-4 video sequences, "football" being a widely-used one. The video sequences are 10 seconds long, with 250 frames, each of 320 × 240 pixel size. The average transmission rate was approximately 1 Mb/s. The decreased number of altered frames, for loss rates exceeding 0.8 % (see Figure 6.5), is a consequence of the increased number of dropped video frames. A possible explanation could be the fact that the severely degraded frames are no longer rendered, so the number of dropped frames increases. By putting together the two metrics, one can plot the total number of affected frames (TNAF)—both dropped and altered—as a function of packet loss at network level. The relatively monotonic increase of TNAF can be observed in Figure 6.5.



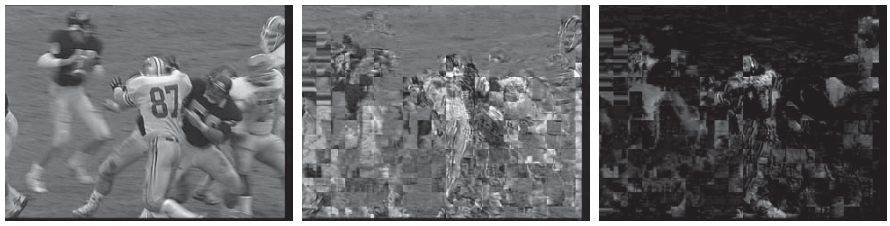
(a) NDF (b) NAF (c) TNAF

Figure 6.5: The percentage of (a) dropped (b) altered and (c) total number of affected video frames vs. packet loss.

In Figure 6.5 one can observe that, for example, a 0.6% packet loss causes 60% of the video frames to be affected. This shows how vulnerable to packet loss an MPEG-4 video streaming application may be. Losing one packet containing the information of an I (intra) frame from the MPEG-4 stream implies the degradation of all the following P (predictive) or B (bi-directional predictive) frames. Altering the information of a P frame implies only the degradation of another adjacent frame. Altered B frames do not cause the degradation of other video frames.

Further on, we focus on the QoE assessment by fractal analysis means. The color fractal dimension (CFD) and lacunarity are the two most-known and widely-used fractal analysis tools. The fractal dimension characterizes the complexity of a fractal set, by indicating how much space is filled, while the lacunarity is a mass distribution function indicating how the space is occupied [101]. These two fractal properties are successfully used to discriminate between different structures exhibiting a fractal-like appearance [102], [103], [104], for classification and segmentation, due to their invariance to scale, rotation or translation.

In Figure 6.6 we present two video frames: one from the original video sequence and the corresponding degraded video frames from the sequence at the receiver, along with the pseudo-image representing the absolute difference between the former two. The computed colour fractal dimensions are 3.14, 3.31 and 3.072, respectively. One can see that the larger fractal dimension reflects the increased complexity of the degraded video frame. The increased complexity comes from the blockiness effect, as well as from the augmented color content.



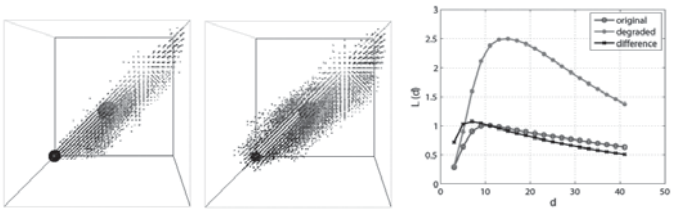
(a) original CFD=3.14 (b) degraded CFD=3.31 (c) absolute CFD=3.072

Figure 6.6: Original video frame (a), corresponding degraded received video frame (b) and absolute difference (c).

The corresponding lacunarity curves are depicted in Figure 6.7. One can see that the curve for image 6.6(b) is placed highly above the curve for the image 6.6(a) indicating a more lacunar and heterogeneous image.

Surprisingly enough, the difference image 6.6(c) has a very similar lacunarity to the one of the original image, but the difference pseudo-image is more lacunar than the original for small values of δ: $\delta \leq 10$—indicating that the degradation mainly takes place in blocks of 8×8 pixels—while for larger values of δ it is less lacunar—more uniform, clearly seen and justified by the smaller variations of colours. The complexity revealed by the lacunarity curves is in accordance with the fractal dimension: the original unaffected video frame being a less lacunar image than the degraded one.

Because the lacunarity is a measure of how the space is occupied, we present in Figure 6.7 the 3D histograms in the RGB colour space, as a visual justification. One can see that the histogram of the degraded video frame is more spread than the one of the original video frame, indicating a more rich image from the point of view of its color content (see more details in [100]).



(a) original (b) degraded (c) lacunarities

Figure 6.7: The 3D RGB histograms for the two video frames and the corresponding lacunarity curves.

In Figure 6.8 we depict the block diagram that illustrates the use of the color fractal dimension and lacunarity as video quality metrics in a reduced reference scenario. At the source, the two fractal measures are computed for each video frame and sent along with the coded video frames over the network. At destination, the same fractal measures are computed for the received video frames and compared with the references. Details about the experimental setup are to be found in [34], [105], [64].

In Figure 6.9 one may see three type of degradation that occurs in our tests: *important* or *severe* degradation (top 2 lines); *less-affected* frames (middle) and *special* or green degraded frames (bottom). The difference ΔCFD between the colour fractal dimension of the degraded and the original corresponding video frame will be considerable for the first two images that exhibit an important degradation—i.e almost the entire image is affected by severe blockiness, and the scene cannot be understood. ΔCFD

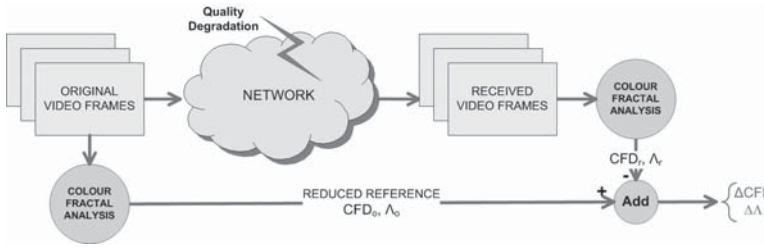


Figure 6.8: The block diagram.

will be small, but still positive for less affected images (the football players may no longer be identifiable, but the rest of the scene is unchanged). For the "green" images the colour fractal dimension is smaller than the one of the corresponding original frames, therefore the ΔCFD will be negative.

In order to analyse the degradation in time, in Figure 6.10 the evolution of the color fractal dimension in time is depicted. One can see that the original "football" sequence is characterized by a large variation in the complexity of the image, due to the fact that the scene changes and also due to the high dynamicity. Therefore the variation of the color fractal dimension due to degradation is almost insignificant. In addition, due to the lost video frames, the two curves will get more and more desynchronized in time, which makes the analysis more difficult. However, it is possible to create a reference-based metric by using the color fractal dimension (note the grey zones that indicate a slight increase of the fractal dimension due to quality degradation at network level).

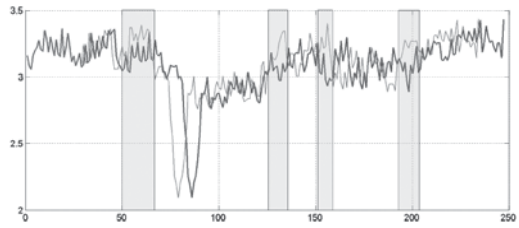
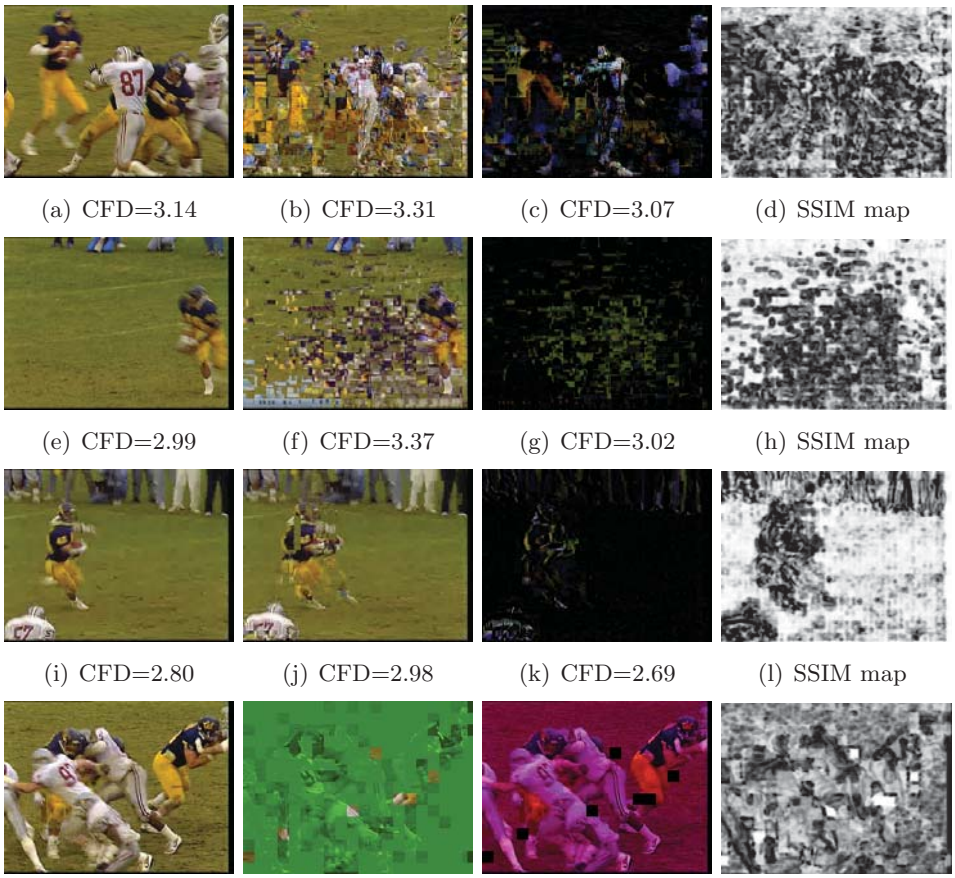


Figure 6.10: Color fractal dimension vs. time, original (thick line) and received (thin line), for the "football" video sequence.



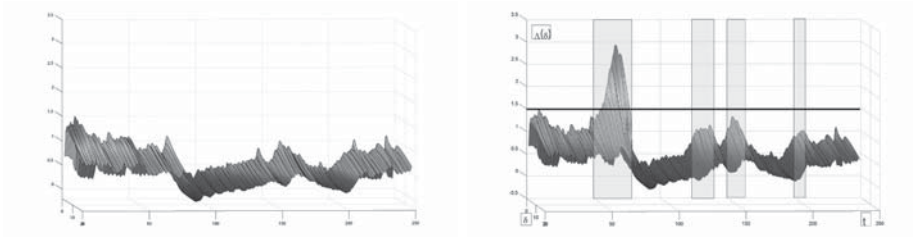
(a) CFD=3.14 (b) CFD=3.31 (c) CFD=3.07 (d) SSIM map

(e) CFD=2.99 (f) CFD=3.37 (g) CFD=3.02 (h) SSIM map

(i) CFD=2.80 (j) CFD=2.98 (k) CFD=2.69 (l) SSIM map

(m) CFD=3.22 (n) CFD=2.28 (o) CFD=2.97 (p) SSIM map

Figure 6.9: Original video frames (1^{st} column) from the "football" sequence, degraded frames exhibiting different levels of degradation (2^{nd} column), absolute differences (3^{rd} column) and the SSIM map [91] (4^{th} column).



(a) original video seq (b) degraded video seq

Figure 6.11: Colour lacunarity curves vs. time for the "football" sequence.

One can note that for the original "football" video sequence the color lacunarity has also an important variation (see Figure 6.11) from frame to frame, but its values are comprised between 0 and 1.5. For the degraded video sequence (b) we can see that the lacunarity skyrockets up to 3.0 for the interval of video frames affected by important degradation (the first interval market with grey). The less important degradation (the next grayed intervals) can only be detected if we take as reference the lacunarity of the original video sequence. In order to implement a no-reference metric, $lacunarity \geq 1.5$ can indicate the severe degradation.

Further on, we performed a comparison with the following metrics: SNR, PSNR, MSE, SSIM and VSNR (see [100] for more details). For the computation of the SSIM we used the Matlab code[11] provided by the author of the metric proposed in [91] and for VSNR the Matlab implementation available[12] provided by the authors of [95]. For color images, the MSE, SNR and PSNR metrics are often computed independently for the red, green and blue (RGB) colour channels and averaged together in order to compute the final distortion. We chose to compute these classical signal fidelity measures in the RGB color space, to be consistent with the definition of the color fractal approach, which was developed based on the RGB colour space.

We compute the difference ΔCFD between the colour fractal dimension of the degraded video frame and the colour fractal dimension of the original video frame, along with the metrics mentioned above. The values of ΔCFD are very well correlated to SNR, PSNR and MSE, and well correlated to VSNR, but they are not at all correlated to SSIM. However, for the minimum visible degradation for which $\Delta CFD = 0.178$ is small, the SSIM indicates the largest similarity, as well as PSNR, and VSNR has also a large value. For the largest visible degradation the VSNR well captures it, while SSIM does not reach its minimum values. SSIM and VSNR were mainly used to assess the quality degradation induced by the image compression algorithms, case in which the image degradation is not as violent as in our experiments.

The original hypothesis was that the quality perceived is directly proportional to the fractal complexity of an image. In order to validate from a subjective point of view the approach we proposed for the assessment of the video quality, we performed several subjective tests, on different video frames from video sequences - sport videos of football matches, in particular. The aim of the experiments was to prove that the complexity of colour fractal images is in accordance with the human perception, therefore the color fractal analysis-based tools are appropriate for the development of video

[11] http://www.ece.uwaterloo.ca/~z70wang/research/ssim/
[12] http://foulard.ece.cornell.edu/dmc27/vsnr/vsnr.html

streaming QoE metrics.

We ran our experiments on a set of 27 individuals, guided by the general recommendations from [26]. In the experiment we used video frames—original and degraded—from the standard test "football" video sequence. Pairs of images were presented, thus the experiments were *reference-based*. After presenting the minimum and the maximum degradation that may affect the video frames, the individuals were asked to grade the perceived degradation with a score comprised between 0 and 5, according to the levels of degradation presented in Table 6.3, in accordance with the quality levels specified by the ITU.

0	no degradation at all
1	imperceptible
2	perceptible, but not annoying
3	slightly annoying
4	annoying
5	very annoying

Table 6.1: Levels of video QoE.

We computed the mean opinion score and the standard deviation, σ_{MOS}, based on the 27 responses, as well as the colour fractal dimension (CFD) and its variation, ΔCFD. The correlation coefficient between the MOS and ΔCFD is 0.8523, if we exclude the green frames. Despite of the fact that these results must be extended to a bigger image set, the approach creates a new perspective on QoE assessment using the perception of color image complexity. We conclude that the fractal dimension reflects the perceived visual complexity of the degraded images, as long as the degradation is not extreme and ΔCFD is not negative.

Bibliography

[1] D. B. Ingham, G. D. Parrington, "Delayline - A wide-Area Network Emulation Tool".

[2] L. Rizzom "Dummynet: a simple approach to the evaluation of network protocols".

[3] I. Yeom, A. L. Narasimha Reddy, "ENDE: An End-to-end Network Delay Emulator".

[4] P. Martin, "An Analysis of Random Number Generators for a Hardware Implementation of Genetic Programming using FPGAs and Handel-C", Technical Report CSM-358, January 2002.

[5] B. Schneier, "Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C", John Wiley & Sons, Inc., January 1996.

[6] Celoxica, "DK3.1 Handel-C Language Reference Manual", Celoxica Limited, 2005.

[7] B. W. Kernighan, D. M. Ritchie, "The C Programming Language, Second Edition", Prentice Hall, Inc., 1988.

[8] Simena, http://www.simena.net/company.htm

[9] Anue, http://www.anuesystems.com/index.htm

[10] Empirix, http://www.empirix.com

[11] X. W. Huang, R. Sharma, S. Keshav, "The ENTRAPID Protocol Development Environment", Proceedings of IEEE INFOCOM'99, March 1999.

[12] M. F. Arlitt, C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants", Proc. SIGMETRICS, Philadelphia, PA, USA, April, 1996.

[13] J. Geier, "Deploying Voice over Wireless LANs", Cisco Press, Indianapolis, 2007.

[14] F. Fluckiger, "Understanding Network Multimedia–Applications and Technologies", ISBN 0-13-190992-4, Prentice Hall, 1995.

[15] National Institute of Standards and Technology, NIST Net network emulator, http://www-x.antd.nist.gov/nistnet/

[16] M. Carson, D. Santay, "NIST Net - A Linux-based Network Emulation Tool", to appear in Computer Communication Review.

[17] D. A. Patterson, J. L. Hennessy, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, Inc., 1990.

[18] A. O. Allen, "Probability, Statistics, and Queueing Theory with Computer Science Applications", 2nd edition, Academic Press, Inc., 1990.

[19] Shunra, "The Virtual Enterprise: Eliminating the risk of delivering distributed IT services", white paper, 2004.

[20] M. Ciobotaru, M. Ivanovici, R. Beuran, S. Stancu - "Versatile FPGA-based Hardware Platform for Gigabit Ethernet Applications" , 6th Annual Postgraduate Symposium, Liverpool, UK, June 27-28, 2005.

[21] M. Ciobotaru, S. Stancu, M. LeVine, B. Martin - "GETB - A Gigabit Ethernet Application Platform: its Use in the ATLAS TDAQ Network", Real Time 2005, Stockholm, Sweden, June 10, 2005.

[22] M. Joss, "IO_RCC - A package for user level access to I/O resources on PCs and compatible computers", CERN, Technical report ATL-D-ES-0008, October, 2003.

[23] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, July 1991.

[24] D. Curet, E. Gouleau, S. Relier, C. Roux, P. Clement, G. Cherry, "RTP Payload Format for MPEG-4 FlexMultiplexed Streams", IETF draft, July, 2002

[25] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, P. Gentric, "Transport of MPEG-4 Elementary Streams", IETF draft, November, 2002.

[26] ITU-R Recommendation BT.500, "Subjective quality assessment methods of television pictures", ITU, 1998.

[27] ITU-T Recommendation P.910, "Subjective Video Quality Assessment Methods for Multimedia Applications", ITU, 1996.

[28] ITU-R Recommendation J.140, "Subjective assessment of picture quality in digital cable television systems", ITU, 1998.

[29] ITU-T Recommendation J.143, "User requirements for objective perceptual video quality measurements in digital cable television", ITU, 2000.

[30] S. Winkler, "Visual Fidelity and Perceived Quality: Towards Comprehensive Metrics", Proc. SPIE Human Vision and Electronic Imaging, vol. 4299, pp. 114-125, San Jose, California, January, 2001.

[31] S. Winkler, "Issues in Vision Modelling for Perceptual Video Quality Assessment", Signal Processing, vol. 78, no. 2, pp. 231-252, October, 1999.

[32] VQEG, "Final report from the Video Quality Experts Group on the validation of objective models of video quality assessment". Vision Models", presentation, August, 1997.

[33] S. Winkler, A. Sharma, D. McNally, "Perceptual Video Quality and Blockiness Metrics for Multimedia Streaming Applications", Proc. 4th International Symposium on Wireless Personal Multimedia Communications, pp. 553-556, Aalbord, Denmark, September, 2001.

[34] M. Ivanovici, "Objective Performance Evaluation for MPEG-4 Video Streaming Applications" , Scientific Bulletin of University "POLTEHNICA" Bucharest, C Series (Electrical Engineering), submitted for publication.

[35] M. Ivanovici, R. Beuran, N. Davies, "Assessing Application Performance in Degraded Network Environments - an FPGA-based approach", Communicating Process Architectures, Eindhoven, Netherlands, September 18-21, 2005.

[36] R. Beuran, M. Ivanovici, N. Davies, B. Dobinson, "Evaluation of the Delivery QoS Characteristics of Gigabit Ethernet Switches" , CERN-OPEN-2005-002, CERN, Geneva, Switzerland, December 2004.

[37] R. Beuran, M. Ivanovici, V. Buzuloiu, "File Transfer Performance Evaluation" , Scientific Bulletin of University "POLTEHNICA" Bucharest, C Series (Electrical Engineering), vol. 66, no. 2-4, 2004, pp. 3-14.

[38] R. Beuran, M. Ivanovici, "User-Perceived Quality Assessment for VoIP Applications", technical report (delivered to U4EA Technologies), CERN-OPEN-2004-007, CERN, Geneva, Switzerland, January 2004.

[39] R. Beuran, M. Ivanovici, B. Dobinson, N. Davies, P. Thompson, "Network Quality of Service Measurement System for Application Requirements Evaluation" , International Symposium on Performance Evaluation of Computer and Telecommunication Systems, July 20-24, 2003, Montreal, Canada, pp. 380-387.

[40] A. S. Tanenbaum, "Modern Operating Systems", second edition, Prentice Hall, 2001.

[41] J. L. Hennessy, D. A. Patterson, "Computer Architecture: A Quantitative Approach", second edition, Morgan Kaufman Publishers, Inc., 1996.

[42] IEEE 802, "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", IEEE Computer Society, http://standards.ieee.org/getieee802/802.html

[43] IEEE 802.3ae, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE Computer Society, http://standards.ieee.org/getieee802/802.3.html

[44] H. J. Chao, X. Guo, "Quality of Service in High-Speed Networks", John Wiley & Sons, Inc., 2002.

[45] ITU-T Recommendation I.380, "Internet Protocol (IP) Data Communication Service—IP Packet Transfer and Availability Performance Parameters", ITU-T, February, 1999.

[46] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics", IETF RFC 2330, May, 1998.

[47] G. Almes, S. Kalidindi, M. Zekauskas, "A One-way Delay Metric for IPPM", IETF RFC 2679, September, 1999.

[48] G. Almes, S. Kalidindi, M. Zekauskas, "A One-way Packet Loss Metric for IPPM", IETF RFC 2680, September, 1999.

[49] Oxford English Dictionary, the online version, http://www.oed.com.

[50] K. Korcyl, G. Sladowski, R. Beuran, R. W. Dobinson, C. Meirosu, M. Ivanovici, M. L. Maia, "Network performance measurements as part of feasibility studies on moving part of the ATLAS Event Filter to off-site Institutes" , First European Across Grids Conference, Santiago de Compostela, Spain, February 2003.

[51] K. Korcyl, G. Sladowski, R. Beuran, R. W. Dobinson, C. Meirosu, M. Ivanovici, M. L. Maia, "Network performance measurements for massive data transfers between CERN Geneva and Cyfronet Cracow", Cracow Grid Workshop, Cracow, Poland, December 2002.

[52] C. Demichelis, P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", IETF RFC 3393, November 2002.

[53] Altera, "Stratix Device Handbook", http://www.altera.com/literature/hb/stx/stratix_handbook.pdf

[54] F. R. M. Barnes, R. Beuran, R.W. Dobinson, M.J. LeVine, B. Martin, J. Lokier, and C. Meirosu, "Testing Ethernet Networks for the ATLAS Data Collection System", IEEE Trans. Nucl. Sci., Vol. 49, No. 2, April 2002, pp. 516-520.

[55] V. Servis, "Measuring speech quality over VoIP networks, The TOLLY Group, December 2001.

[56] ITU-T Recommendation G.711, Pulse Code Modulation (PCM) of voice frequencies, ITU-T, 1993.

[57] W. Richard Stevens, "TCP/IP Illustrated, Volume 1. The Protocols", Addison Wesley, ISBN 0-201-63346-9, July 1997.

[58] L. B. James, A. W. Moore, M. Glick, "Structured errors in optical Gigabit Ethernet Packets, PAM, Nice, April 2004.

[59] More than IP, "10/100/1000 Mbps Ethernet MAC Core with FIFO", Release notes, September 2003.

[60] C. A. R. Hoare, "Communicating Sequential Processes", Prentice-Hall, 1985.

[61] K. Mochalski, J. Micheel, S. Donnelly, "Packet Delay and Loss at the Auckland Internet Access Path", PAM2002 Passive and Active Measurement Workshop, Fort Collins, Colorado, USA, March 25-26th, 2002.

[62] J. Pongsiri, M. Parikh, M. Raspopovic, K. Chandra, "Visualization of Internet Traffic Features", Center for Advanced Computation and Telecommunications, University of Massachusetts Lowell, http://morse.uml.edu/

[63] R. Beuran, "Introduction to Network Emulation", Pan Stanford Publishing, 2013.

[64] M. Ivanovici, R. Beuran, "Correlating Quality of Experience and Quality of Service for Network Applications", in Quality of Service Architectures for Wireless Networks: Performance Metrics and Management, pag. 326-351, IGI-Global, USA, 2010

[65] Asterisk Home Page, open source PBX, http://www.asterisk.org/

[66] Price Waterhouse Coopers Home Page, Information security breaches survey 2006 - technical report, http://www.pwc.co.uk/

[67] D. Collins, "Carrier Grade Voice Over IP Second Edition", McGraw-Hill, 2004

[68] ITU-T Recommendation P.800, "Methods for subjective determination of transmission quality", ITU-T, August 1996.

[69] ITU-T Recommendation P.861, "Objective quality measurement of telephone-band (300-3400 Hz) speech codecs", ITU-T, February 1998.

[70] ITU-T Recommendation G.107, "The E-model, a computational model for use in transmission planning", ITU-T, May 2000.

[71] Malden Electronics Ltd., "PAMS A Perceptual Analysis/Measurement System", http://www.malden.co.uk/products/dsla/pams.htm.

[72] ITU-T Recommendation P.862, "Perceptual evaluation of speech quality (PESQ), an objective method for end to end speech quality assessment of narrow-band telephone networks and codecs", ITU-T, February 2001.

[73] B. C. Wiles, J. Walker, Speak Freely VoIP application, http://www.speakfreely.org.

[74] ITU-T Recommendation G.726, "40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)", ITU-T, 1990.

[75] M. Rahnema, "Overview of the GSM system and protocol architecture", IEEE Communications Magazine, 1993.

[76] C. Fernandez-Maloigne, "Fundamental Study for Evaluating Image Quality", Annual Meeting of TTLA, ITRI, Taiwan (invited paper), December 2008.

[77] T. Yamada, Y. Miyamoto, M. Serizawa, H. Harasaki, "Reduced-Reference based Video Quality Metrics using Representative-Luminance Values", Image Communication, vol. 24, no. 7, pag. 525-547, August 2009.

[78] T. Oelbaum, K. Diepold, "Building a Reduced Reference Video Quality Metric with Very Low Overhead using Multivariate Data Analysis", The 4th International Conference on Cybernetics and Information Technologies, Systems and Applications: CITSA 2007.

[79] C. Fernandez-Maloigne, M.C. Larabi, G. Anciaux, "Comparison of subjective assessment protocols for digital cinema applications", plenary talk, QoMEX First International Workshop on Quality of Multimedia Experience, San Diego, USA, 2009.

[80] V. Rossell, M.C. Larabi, G. Anciaux, C. Fernandez-Maloigne, "Objective Quality Measurement Based on Anisotropic Contrast Perception", 4th European conference on Color in Graphics, Imaging and Vision, June 2008.

[81] M. Ivanovici, "Color and Multispectral Texture Image Analysis - Models, Features and Applications", Transilvania University printing house, 2015.

[82] Z. Wang, A.C. Bovik, "Mean Squared Error: Love It or Leave It?", IEEE Signal Processing Magazine, pag. 98-117, January 2009.

[83] C. J. van den Branden Lambrecht, "Survey of Image and Video Quality Metrics based on Vision Models", presentation, August 1997.

[84] T. N. Pappas, R. J. Safranek, J. Chen, "Perceptual Criteria for Image Quality Evaluation" in Handbook of Image and Video Processing, 2nd ed., pag. 669-686, Academic Press, 2000.

[85] OPTICOM GmbH Germany, "PEVQ - Advanced Perceptual Evaluation of Video Quality", white paper, 2005.

[86] P.C. Teo, D.J. Heeger, "Perceptual Image Distortion", Proceedings of the IEEE International Conference of Image Processing, pag. 982-986, 1994.

[87] S.A. Karunasekera, N.G. Kingsbury, "A Distortion Measure for Blocking Artifacts in Images Based on Human Visual Sensitivity", IEEE Transactions on Image Processing, vol. 4, no. 6, pag. 713-724, June 1995.

[88] S. Winkler, "Digital Video Quality - vision models and metrics", John Wiley & Sons, Ltd., 2005.

[89] B. Bringier, N. Richard, M.C. Larabi, C. Fernandez-Maloigne, "No-Reference Perceptual Quality Assessment of Colour Image", 14th European Signal Processing Conference (EUSIPCO), Florence, Italy, September 4-8, 2006.

[90] L. Quintard, M.C. Larabi, C. Fernandez-Maloigne, "No-Reference Metric Based on the Hue Feature: Application to Quality Assessment of Color Displays", 4th European conference on Color in Graphics, Imaging and Vision, June 2008.

[91] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", IEEE Transactions on Image Processing, vol. 13, no. 4, pag. 600-612, April 2004.

[92] M. H. Sampat, Z. Wang, S. Gupta, A.C. Bovik, M.K. Markey, "Complex Wavelet Structural Similariy: A New Image Similarity Index", IEEE Transactions on Image Processing, vol. 18, no. 11, pag. 2385-2401, November 2009.

[93] U. Rajashekar, Z. Wang, E.P. Simoncelli, "Quantifying Color Image Distortions based on Adaptive Spatio-Chromatic Signal Decompositions", IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, November 2009.

[94] U. Rajashekar, Z. Wang, E.P. Simoncelli, "Perceptual Quality Assessment of Color Images using Adaptive Signal Representation", Human Vision and Electronic Imaging XV, Proc. of the IS& T/SPIE Annual Symposium on Electronic Imaging, vol. 7527, San Jose, California, 2010.

[95] D.M. Chandler, S.S. Hemami, "VSNR: A Wavelet-Based Visual Signal-to-Noise Ratio for Natural Images", IEEE Transactions on Image Processing, vol. 16, no. 9, pag. 2284-2298, September 2007.

[96] D.M. Chandler, S.S. Hemami, "Effects of natural images on the detectability of simple and compound wavelet subband quantization distortions", Journal of the Optical Society of America, vol. 20, no. 7, pag. 1164-1180, July 2003.

[97] D.M. Chandler, S.S. Hemami, "Suprathreshold image compression based on contrast allocation ang global precedence", SPIE Human Vision and Electronic Imaging VIII, Santa Clara, CA, 2003.

[98] D.M. Chandler, S.S. Hemami, "Effects of spatial correlation and global precedence on the visual fidelity of distorted images", SPIE Human Vision and Electronic Imaging XI, San Jose, CA, 2006.

[99] M. Malkowski, D. Claßen, "Performance of Video Telephony Services in UMTS using Live Measurements and Network Emulation", Wireless Personal Communications Journal, September 2007.

[100] M. Ivanovici, N. Richard, C. Fernandez-Maloigne, "Towards Video Quality Metrics Based on Colour Fractal Geometry", Journal of Image and Video Processing, Hindawi Publishing Corp., January 2010.

[101] C.R. Tolle, T.R. Mc Junkin, D.T. Rohrbaugh, R.A. LaViolette, "Lacunarity definition for ramified data sets based on optimal cover", Physical D, vol. 179, no. 3, pag. 129-15, 2003.

[102] W.S. Chen, S.Y. Yuan, H. Hsiao, C.M. Hsieh, "Algorithms to estimating fractal dimension of textured images", IEEE International conferences on Acoustics, Speech and Signal Processing (ICASSP), vol. 3, pag. 1541-1544, May 2001.

[103] W.L. Lee, Y.C. Chen, K.S. Hsieh, "Ultrasonic liver tissues classification by fractal feature vector based on M-band wavelet transform", IEEE Transactions on Medical Imaging, vol. 22, pag. 382-392, 2003.

[104] G.W. Frazer, M.A. Wulder, K.O. Niemann, "Simulation and quantification of the fine-scale spatial pattern and heterogeneity of forest canopy structure : a lacunarity -based method designed for analysis of continuous canopy heights", Forest ecology and management, vol. 214, pag. 65-90, 2005.

[105] M. Ivanovici, R. Beuran, "User-Perceived Quality Assessment for Multimedia Applications", The 10th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), vol. IV, pag. 55-60, 18-19 May 2006.

The book deals with a very important aspect of network research: the assessment of the quality of experience (QoE), that is quantifying the manner in which network conditions influence user satisfaction. The approach used for this purpose is that of network emulation, which allows the author to evaluate real applications and protocols in a wide range of realistic network conditions.

Prof. Ivanovici' s presentation is very thorough, starting from a characterization of the issues related to QoE assessment and an overview of existing network emulation systems, and continuing with a solid description of the principles, techniques and models for network emulation. Of paramount importance are the practical aspects of this book, including a detailed presentation of an FPGA-based network emulator and of a series of actual QoE assessment results for various network applications and protocols. All these contribute to providing readers with both the theoretical and practical knowledge and skills so that they can tackle the topical but challenging task of network emulation with confidence.

Dr. Eng. Razvan Beuran
Research Associate Professor
School of Information Science
Japan Advanced Institute of Science and Technology

ISBN 978-606-19-0586-7