ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Philipp Sager

Does Circuit Emulation in Metropolitan Gigabit Ethernets require Service Priority?

Post Diploma Thesis NA-2005-02
November 2004 to February 2005

Supervisor: Dr. Ulrich Fiedler
Co-Supervisor: Rainer Baumann
Professor: Prof. Bernhard Plattner

Chapter 1

Abstract

Recent developments in transmission and switching technologies create new opportunities to converge data and telephony services. Given the emerging deployment of metropolitan Gigabit Ethernets, one aspect of interest is to employ these networks to connect legacy PBX and GSM base stations to the core telephony network. This protects investment in existing infrastructure and creates new revenues for network providers. However, to make this happen, these networks have to be configured in a way that customary QoS requirements for telephony can be met. This is a crucial problem given that the data cross traffic in these networks is bursty. This burstiness can cause excessive queuing delays and frame losses due to buffer overflows and is a widely investigated phenomenon. Moreover, the burstiness of data traffic in these networks is known to be self-similar over a wide range of time scales. This means that there is very little smooth out when aggregating this traffic over time which in turn means that buffering has little effect in mitigating the burstiness. However, it is also known that the burstiness is caused by the heavy-tail in the distribution of transfer sizes which is known to be limited at sizes due to the limits for file sizes in operating systems. Since Gigabit Ethernets operate at enormous transmission rates, the problem whether metropolitan Gigabit Ethernet can accommodate the burstiness of expected traffic patterns becomes real.

Therefore, in this thesis we conduct a simulation study to investigate whether TDM E1 telephony traffic can meet the METRO Ethernet Forum's QoS requirements for delay and loss when transported over a metropolitan Gigabit Ethernet. As a simulation environment we use OPNET Modeler which is widely used in industry and academia. To generate the data cross traffic, we employ OPNET's Raw Packet Generator and as well as a ON/OFF model which we have implemented by ourselves.

In the network, we configure switches to use FIFO and tail drop. Varying the switch buffer limitation by number of packets and by size (in terms of bytes), we find that even for data traffic utilizations as low as 1 % there is little chance to meet the QoS requirements. When the switch buffers are limited by the number of packets, none of our simulations of a bottleneck link could meet the QoS requirements. When the switch buffers are limited by a size larger than 1 MB, then QoS requirements could not be met in our simulations of a bottleneck link because of too many late frames. Only when the switch buffers are limited by a size smaller than 1 MB, QoS requirements could be met. The reason for this is the different size of the frames that transport data and telephony traffic. Frames that transport data are as large as 1'500 bytes, whereas frames that transport telephony traffic are as small as 78 bytes. This leads to a lock-out phenomenon at the end of a saturated queue in a switch. This means there is still capacity for one or more small frames that carry telephony traffic although there is no capacity left for large frames that carry data traffic. However, we doubt that it is reasonable to rely on this effect when employing metropolitan Gigabit Ethernets to transport telephony traffic. Therefore, we conclude that it may be a good idea to have a closer look at the IEEE 802.1Q/p standards which offer service priority when configuring metropolitan Gigabit Ethernets to transport telephony traffic.

Contents

List of Figures

List of Tables

Chapter 2

Introduction

Currently, dedicated infrastructures are used to provide telephony and data services. The infrastructure for telephony services is entirely based on *circuit-switching* and for data services on *packet-switching* technology. Available capacities for packet-switched (data) networks are currently largely expanding due to widespread use of optical transmission technologies. Then emulation of telephony circuit-switching (*circuit emulation (CE)*) can enable to use packet-switched networks to connect telephony equipment to the telephony core network. Circuit emulation is of particular interest in emerging metropolitan *Gigabit Ethernets*, where existing PBX[1] and GSM[2] base stations can be connected. Presumably, this stimulates investments in these existing infrastructures and creates additional revenue for the data network service providers.

However, research on traffic patterns in Ethernets [6] [9] [8] suggests that circuit emulation may have a *Quality of Service* (QoS) problem. Bursty patterns in the data traffic can cause excessive queuing delays and frame losses due to buffer overflows. The burstiness over a wide range of time scales in wide-area and local-area network traffic is a widely investigated phenomenon and can be described statistically using the notion of *self-similarity*. Self-similarity is the property we associate with fractals: the object appears the same regardless of the scale at which it is viewed. Self-similarity in communication networks is principally caused by the *heavy-tailed* distributions of the transfer object sizes. In heavy-tailed distributions, the probability for extremely large values is not negligible. However, measurements in communication networks indicate that the tail of the transfer object size distribution deviates from a heavy tail at some limit [8]. As a consequence, the self-similarity in data traffic and thus the burstiness is also limited. Since Gigabit Ethernets operate at enormous transmission rates, the problem whether metropolitan Gigabit Ethernets can be over provisioned to accommodate the burstiness of the expected traffic becomes real.

Since measurements in local area and corporate networks show that average utilization rates are as low as 1 % [15], this leads to the question if these networks are actually capable of accommodating the burstiness of the expected self-similar traffic and enable circuit emulation without violating QoS requirements. The METRO Ethernet Forum [11] defines QoS requirements for circuit emulation services in metropolitan Ethernet networks. They include boundaries on delay, jitter and the total number of errored frames and can be used to exploit if simulated circuit emulation traffic does meet the QoS requirements.

The QoS performance of circuit emulation traffic is expected to improve if service priority is applied. The virtual LAN (VLAN) and traffic class standards IEEE 802.1Q/p [16] [17] enable service priorization of dedicated traffic on the link layer (MAC layer). These standards can be exploited to provide preferential treatment of circuit emulation traffic in Gigabit Ethernets and thus improve its QoS. The objective of this post diploma thesis is to perform network simulation studies to assess whether and under which conditions service priority is inevitable to enable circuit emulation in metropolitan Gigabit Ethernets.

The problem is investigated with the aid of the discrete event simulation environment OPNET Modeler. Workload data traffic and circuit emulation traffic of a E1 carrier are synthetically generated and sent over a metropolitan Gigabit Ethernet bottleneck network. Two self-similar data

[1] Private Branch Exchange. A privately owned telephone switch, often used in large corporations to provide inside telephone connectivity and access to the Public Switched Telephone Network (PSTN).

[2] Global System for Mobile Communications. World's most widely used mobile system.

traffic models have been tested before using one of them in the simulations. The Raw Packet Generator (RPG), a integrated function of OPNET Modeler, is identified to be inappropriate for this work. The ON/OFF Model of Willinger [9] [8] is used instead to generate the self-similar data workload in all simulations. This model exploits that the superposition of a large number of independent on/off sources with heavy-tailed on- and/or off-periods leads to self-similarity in the aggregated process.

Tests of 44 simulated traffic hours reveal that the results of this thesis strongly depend on the applied buffer size and limitation methods in the switches, given that the buffer queues use first-in-first-out (FIFO) queue scheduling and tail drop queue management algorithms. Large buffer sizes ($> \sim$1 MB) introduce too much delay so that the QoS requirements are clearly violated by exceeding the limit by almost three magnitudes. Small buffer sizes ($< \sim$1 MB) do improve the QoS, but if the buffer capacity is limited by packets, the high number of dropped frames also violates the QoS requirements. Only when the buffer capacity is limited by bits, the circuit emulation traffic does meet the QoS requirements in the simulations of this work. In this case, not a single circuit emulation frame is dropped during all simulation periods. The *lock-out* phenomenon allows the circuit emulation traffic to monopolize queue space, preventing data traffic from getting room in the queue. This phenomenon is the result of spatial effects: the small circuit emulation traffic packets have a higher chance to be accommodated in the saturated buffer than the about 20 times larger data traffic packets. However, we doubt that it is reasonable to rely on this effect when employing metropolitan Gigabit Ethernets to transport telephony traffic. Since the simulation environment OPNET Modeler does not support priority service standards IEEE 802.1Q/p, no statement can be made if service priority enables circuit emulation.

The thesis is structured as follows: chapter 3 introduces the reader to the problem of this thesis. Chapter 4 contains background topics about random variables, heavy-tailed distributions, data analysis, the ON/OFF Model, circuit-switching, differential services with virtual LAN and the simulation environment OPNET Modeler. Chapter 5 explains the simulation design. Starting with the topology, the workload traffic models for the circuit emulation and the self-similar data traffic are introduced. Then, QoS requirements and other simulation parameters are elaborated. The simulation results are presented and discussed in chapter 6. A outlook about further works is finally given in chapter 7. In the appendix, the original problem, program codes and plots can be found.

Chapter 3

Problem

Measurements in local area and corporate networks show that the average utilization rates are as low as 1 % [15]. This enormous over-provisioning raises the question if it enables circuit emulation. Research on provisioning of intra-nets to accommodate voice over IP (VoIP), real-time and best-effort data suggests that without deploying diffserv[1], intra-nets may need considerable over-provisioning [3]. Service priority (diffserv), in contrast, can save capacities by reducing the over-provisioning up to 62 %. Thus the objective of this post diploma thesis is to perform network simulation studies with OPNET Modeler to asses whether and under which conditions service priority is inevitable to enable circuit emulation on metropolitan Gigabit Ethernets.

The answer strongly depends on the traffic patterns of the workload data, the Quality of Service (QoS) requirements of the circuit emulation traffic and the buffer size and management in the switching nodes. To limit the scope of the thesis, the topology on the problem is modeled as a metropolitan bottleneck network. All simulation run on the link (MAC) layer.

Bursty data traffic patterns can cause excessive queuing delays and frame losses due to buffer overflows. The burstiness over a wide range of timescales in wide-area and local-area network traffic is a widely investigated phenomenon and can be described statistically using the notion of *self-similarity*. Self-similarity in communication networks is principally caused by the *heavy-tailed* distributions of the transfer object sizes. In heavy-tailed distributions, the probability for extremely large values is not negligible. However, measurements in communication networks indicate that the tail of the transfer object size distribution deviates from a heavy tail at some limit [8]. As a consequence, the self-similarity in data traffic and thus the burstiness is also limited. One problem to deal with in this work is therefore how the anticipated self-similar workload data traffic can be modeled to use it for the network simulation inputs. The simulation platform OPNET Modeler already features a package to generate self-similar network traffic, called *Raw Package Generator (RPG)*. If this approach is appropriate must be clarified. Another option to model self-similar data traffic is the ON/OFF data model. With this approach, the traffic can be generated externally and then played into the network simulation. This method must also be tested first.

Circuit emulation traffic is subject to high QoS requirements in terms of loss, delay and jitter. Traditionally, switched circuits are used to meet these high service qualities. Suitable QoS metrics and limits must be found to define QoS requirements in packet-switched metropolitan communication networks for telephony services to determine if circuit emulation is possible.

Buffers in network switches absorb data bursts and transmit them during the ensuing bursts of silence. Buffers can help to accommodate (bursty) self-similar data traffic in a network, but they also affect the QoS of circuit emulation traffic. Different queue size limitations in buffers may distinctive influence the QoS. What buffer sizes and limitation methods are inevitable to enable circuit emulation on Gigabit Ethernets with or without service priority is another problem attacked in this work.

[1] DIFFerentiated SERVices. Framework for delivering quality of service (QoS) in IP networks. Proposed standard from the IETF working group.

Chapter 4

Background

The simulated metropolitan Gigabit Ethernet network in this thesis transports two different types of workload traffic: constant bit rate (CBR) **circuit emulation (CE)** traffic and variable bit rate (VBR), self-similar **data** traffic. The modeling of the CE traffic is straight forward and will be provided in the Design section 5.2. The modeling of the data traffic, in contrast, is quite difficult and complex. The anticipated data traffic in this thesis represents all in a common network transported data except the CE traffic. This data originates from traffic generated by computers directly or indirectly connected to the observed network. It may has various origins, like WWW traffic (HTTP over TCP), file transfers (FTP over TCP), real-time data (UDP), mail transfers (SMTP and POP over TCP) and resource (files, printers, . . .) sharing. Each of these individual data traffics has its own characteristic in terms of its active/passive periods and transfered object sizes.

One possibility to generate data traffic in a network simulation would be to record all the traffic patterns[1] transported in real network and replay it into the simulation. This approach has various disadvantages. First, data recording in a network is not as easy as it may looks like. It requires special software and hardware capable of reading, analyzing and storing the generally huge amount of data. Further, a suitable network has to be found and permission granted. Not everyone is delighted about someone ”sniffing” in all the data sent over a network. Second, the data traffic to be replayed in the simulation is limited to the traffic patterns recorded in the real network. In a simulation, one is generally also interested in how the observed variables respond to changes of the input data. This enables researchers to draw additional conclusions about the mechanism of the system and possible limitations. For instance, such changes could be in the characteristics of active/passive periods or simply the average network utilization rate. Introducing these variations in a real network where the traffic is recorded, can be very difficult.

Those reasons are the motivation to generate the data workload synthetically with a set of suitable parameters to adjust the desired characteristics. Although the real world data traffic may exhibit certain statistical properties like self-similarity, it is not completely predictable and varies with time. Hence, random processes play a import role when generating synthetic, self-similar data traffic. A detailed description of one approach, the ON/OFF Model, is given in this Background chapter in section 4.7. To set the ground, the beginning of this chapter is about statistics, data analysis, heavy-tailed distributions and self-similarity. A short introduction to circuit-switching, Virtual LAN (VLAN) and network simulations, introducing the simulation environment of choice (OPNET Modeler), is then given at the end of this chapter.

4.1 Random Variables and their Distributions

A **random variable** is essentially a random number. There are **discrete** random variables that can only take on a finite or at most a countably infinite number of values and there are **continuous** random variables that can take on a continuum of values. Random variables are denoted in this work by italic uppercase X and its i-th outcomes by indexed italic lowercase x_i. All possible outcomes of a random process is called **sample space**.

[1] For network simulations like in this work, the exact content of the data is not relevant, just the amount and the timing - that’s what the word ”pattern” stands for.

Besides the sample space, the probability that the random variable takes on a certain value (in the sample space) is the other property that describes a random variable. More generally, a function $p(x)$ called **probability mass function (pmf)** or **frequency function** is used to define all probabilities of all possible outcomes of a *discrete* random variable.

$$p(x_i) = P(X = x_i) \tag{4.1}$$

In the case of a **continuous** random variable, the probability function is taken by a **density function** $f(x)$ and the probability that X falls in the interval (a, b) is the area under the density function between a and b.

$$P(a < X < b) = \int_a^b f(x)dx \tag{4.2}$$

From this equation it becomes clear - although it may seems strange - that the probability that a *continuous* random variable X takes on any particular value is 0 because eq. 4.2 is zero for the interval (a,a).

One fundamental property of the pmf is that the sum of probabilities of all values in the sample space must equal to one. The same property holds for the density function of a continuous random variable.

$$\sum_i p(x_i) = 1 \tag{4.3}$$

$$\int_{-\infty}^{\infty} f(x)dx = 1 \tag{4.4}$$

In addition to the probability mass function (pmf), it is sometimes convenient to use the **cumulative distribution function (cdf)** of a random variable, which is defined to be

$$F(x) = P(X \leq x), \qquad -\infty < x < \infty. \tag{4.5}$$

Its **complementary cumulative distribution function (ccdf)** $\overline{F}(x)$ is

$$\overline{F}(x) = P(X > x) = 1 - F(x), \qquad -\infty < x < \infty. \tag{4.6}$$

The probability mass function (pmf) of a *discrete* random variable and the density function of a *continuous* random variable are also called **distributions**. There is a number of frequently used distributions which are well known and named. Some of them are the Normal distribution, the Binomial distribution, the Poisson distribution, the Exponential Density and the Pareto distribution. The last one falls into the category of the **heavy-tailed** distributions and is of special interest for modeling and understanding self-similar data traffic. Heavy-tailed distributions are introduced and discussed in the next sections.

4.2 Heavy-Tailed Distributions

Heavy-tailed distributions are also known as **scale-invariant** distributions [8]. A random variable X has a heavy-tailed distribution if its ccdf $\overline{F}(x)$ obeys

$$\overline{F}(x) = P(X > x) = 1 - F(x) \sim cx^{-\alpha}, \qquad 0 < \alpha \leq 2, \quad c > 0, \tag{4.7}$$

where $a(x) \sim b(x)$ is defined by the relation

$$\lim_{n \to \infty} \frac{a(x)}{b(x)} = 1 \tag{4.8}$$

and c is a positive constant. Parameter α is called **tail index**. For a heavy-tailed distributions, α has to be in the range of $0 < \alpha \leq 2$. The characteristic of a heavy-tailed distribution is that its tail decays *hyperbolically*[2] and therefore slowly. That is why the outcomes of heavy-tailed distributed random variables show a extreme variability. Park et al. [8] explains well what a heavy-tailed distribution means in practice:

[2] In contrast to the *exponentially* decreasing tail of a light-tailed Gaussian or Normal distribution.

"Practically speaking, a heavy-tailed distribution gives rise to very large values with non negligible probability so that sampling from such a distribution results in the bulk of values being "small" but a few samples having "very" large values."

As will be shown later, heavy-tailed distributions have big impacts on statistically processing of its sample values.

4.3 Pareto Distribution

The **Pareto** distribution is the simplest and frequently used heavy-tailed distribution [8]. Its pmf is

$$p(x) = \alpha k^{\alpha} \cdot x^{-\alpha-1}, \qquad k \geq x, \quad \alpha, k > 0. \tag{4.9}$$

Parameter α is called **shape parameter**. It can be shown that it equals to the tail index of heavy-tailed distributions. The **location parameter** k is the smallest possible value of the random variable x.

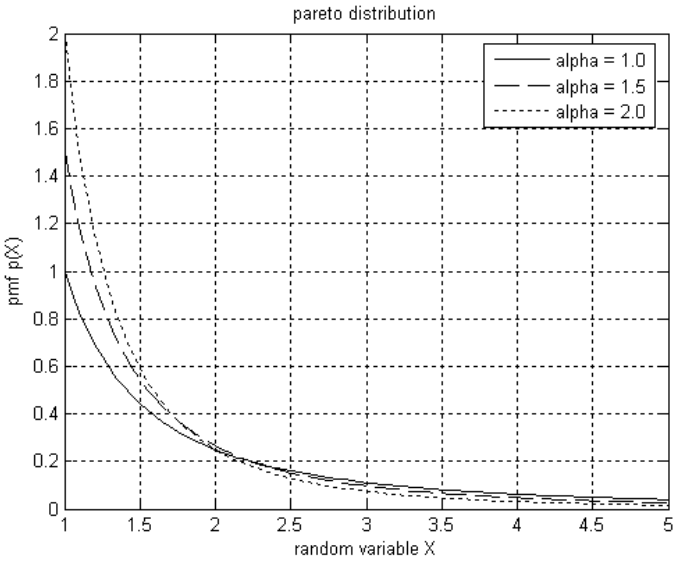

Figure 4.1: Pareto pmf with different shape parameters α and constant location parameter $k = 1$ for small x (normal scale).

Fig. 4.1 shows the Pareto pmf of eq. 4.9 with different shape parameters α in normal scale for small x and fig. 4.2 in log-log scale for a large x. In those figures, the slowly decaying tail of the distribution becomes visible. The influence of the location parameter k on the Pareto pmf is shown in in fig. 4.3 (normal scale) and fig. 4.4 (log-log scale).

The cdf of the Pareto distribution is

$$F(x) = P(X \leq x) = 1 - \left(\frac{k}{x}\right)^{\alpha} \tag{4.10}$$

and is illustrated in fig. 4.5 for different shape parameters α.

The inverse cdf function F^{-1} is well defined under the assumption that F is the cdf of a *continuous* random variable and is strictly increasing on some interval I and that $F = 0$ to the left of I and $F = 1$ to the right of I; I may be unbounded. For the Pareto distribution, the inverse cdf is

$$F^{-1} = k(1-x)^{-\frac{1}{\alpha}}. \tag{4.11}$$

F^{-1} from eq. 4.11 can be used to obtain Pareto-distributed sample values with a random random generator producing uniformly distributed random values in the interval [0,1]. If the

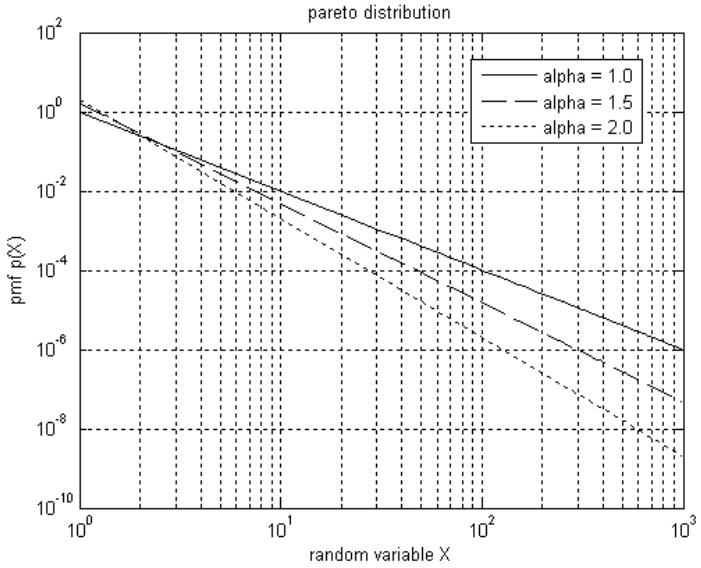

Figure 4.2: Pareto pmf with different shape parameters α and constant location parameter $k = 1$ for large x (log-log scale).

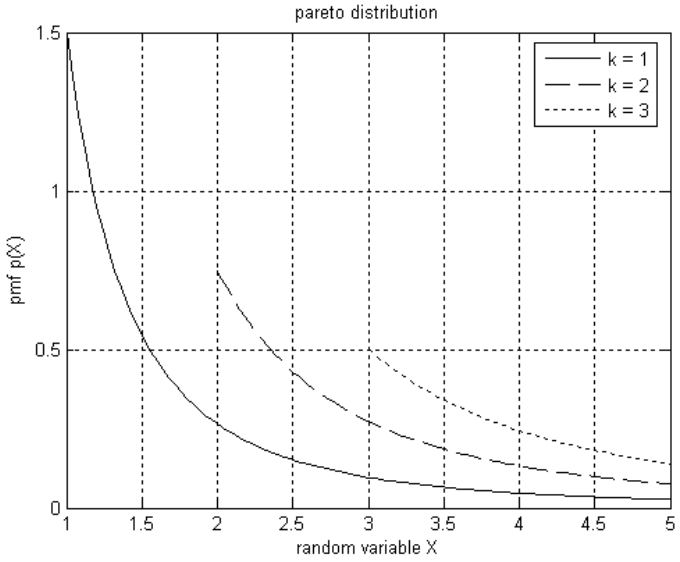

Figure 4.3: Pareto pmf with different location parameters k and constant shape parameter $\alpha = 1.5$ for small x (normal scale).

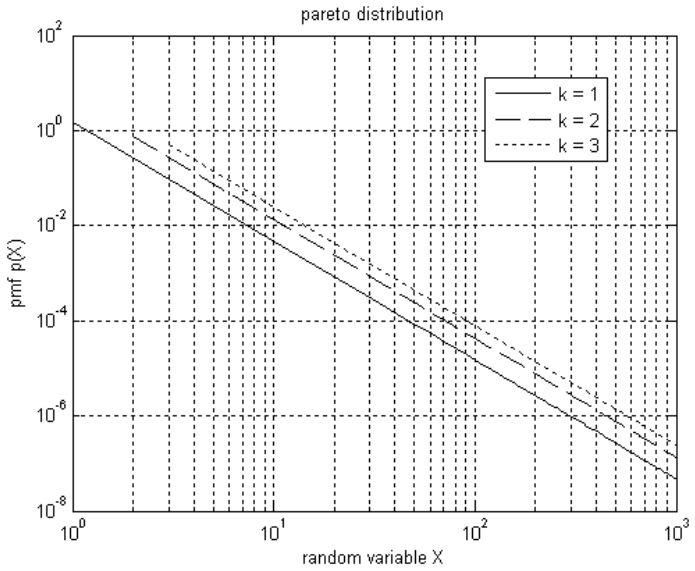

Figure 4.4: Pareto pmf with different location parameters k and constant shape parameter $\alpha = 1.5$ for large x (log-log scale).

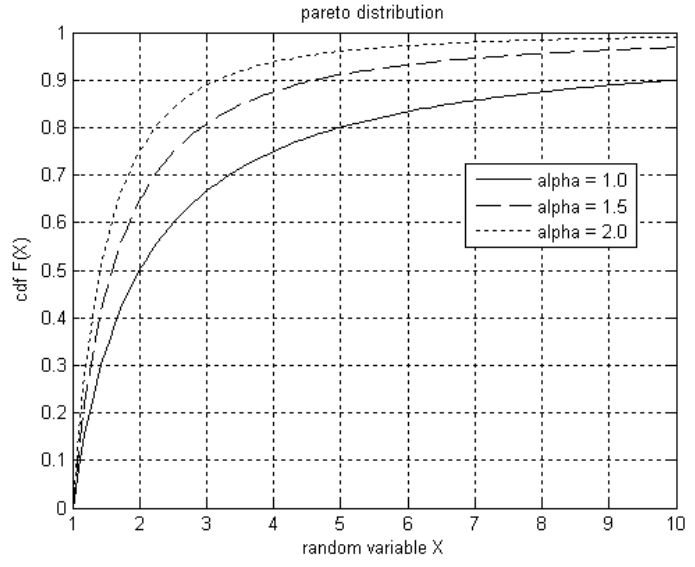

Figure 4.5: Pareto cdf with different shape parameters α and constant location parameters $k = 1$ (normal scale).

outcome of the uniformly distributed random generator is y_i (to distinguish from x_i - the Pareto-distributed random variable), then the corresponding Pareto-distributed random value x_i is

$$x_i = k(1 - y_i)^{-\frac{1}{\alpha}}. \tag{4.12}$$

Fig. 4.6 shows the outcome of 10'000 Pareto-distributed random values x_i obtained from eq. 4.12 and a random generator producing uniformly distributed random values y_i in the interval [0,1].

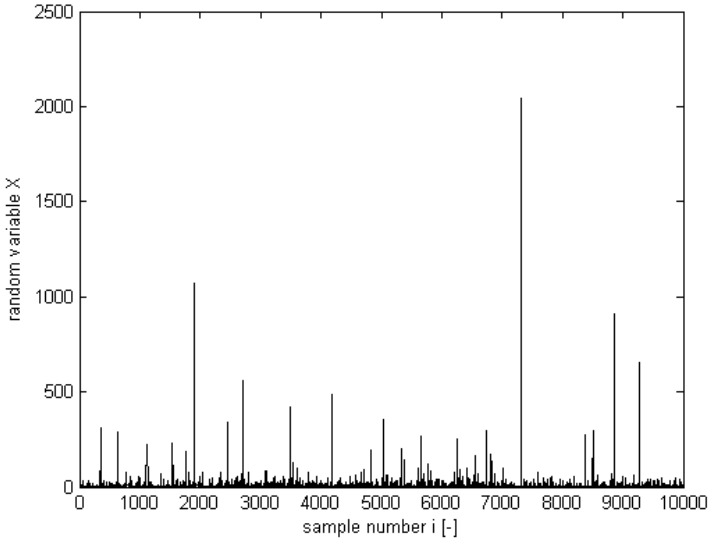

Figure 4.6: Outcome of 10'000 Pareto-distributed random variables with shape parameter $\alpha = 1.2$ and location parameter $k = 1$.

Expected Value

The **expected value** $E(X)$ or **mean value** μ of the Pareto distribution with pmf $p(x)$ from eq. 4.9 is the integral

$$\begin{aligned} E(X) = \mu &= \int_k^l p(x) \cdot x \, dx \\ &= \int_k^l (\alpha k^\alpha \cdot x^{-\alpha-1}) \cdot x \, dx \\ &= \alpha k^\alpha \Big[\frac{x^{-\alpha+1}}{-\alpha+1} \Big]_k^l \\ &= \frac{\alpha k^\alpha}{-\alpha+1} (l^{-\alpha+1} - k^{-\alpha+1}) \end{aligned} \tag{4.13}$$

with the location parameter k as the lower and l the upper limit of the integral. If the maximum of the random variable X is not limited, l is *infinite* and the expected value from eq. 4.13 can be further simplified to

$$\begin{aligned} E(X) = \mu &= \lim_{l \to \infty} \frac{\alpha k^\alpha}{-\alpha+1} (l^{-\alpha+1} - k^{-\alpha+1}) \\ &= \begin{cases} \frac{\alpha k}{\alpha - 1} & \alpha > 1 \\ 0 & \alpha = 1 \\ \infty & \alpha < 1 \end{cases} \end{aligned} \tag{4.14}$$

Given the shape parameter $\alpha > 1$ and the mean value μ for the *infinite* distribution, the location parameter k results from eq. 4.14.

$$k = \frac{\mu(\alpha - 1)}{\alpha}. \tag{4.15}$$

In case of a *limited* distribution with maximum value l, no analytical methods are found to solve eq. 4.13 for k. The numerical method to find the zero point of

$$\frac{\alpha}{-\alpha + 1}(l^{-\alpha+1}k^{\alpha} - k) - \mu = 0 \tag{4.16}$$

is used to obtain k in this work.

Variance

The very bursty behavior of the Pareto distribution and generally all heavy-tailed distributions becomes obvious in the statistical property of its **variance** σ^2. It is infinite for $0 < \alpha < 2$.

$$Var(X) = \sigma^2 = E([X - E(X)]^2) = \infty, \qquad 0 < \alpha < 2 \tag{4.17}$$

In the network context, primarily distributions with tail indices $1 < \alpha < 2$ and infinite variance are of interest. Typically, α is in the range of 1.1...1.2 [8].

4.4 Summarizing Data

In the previous sections, the *continuous* Pareto distribution function is introduced in terms of its mathematical and statistical properties. For network simulations, the Pareto distribution function is only used indirectly to generate random outcomes of a Pareto-distributed variable. The following section gives a general overview how samples of data can be summarized with a special focus on Pareto-distributed random variables.

Note: Many authors use the terms for the statistical properties of the distribution functions and its sample outcomes interchangeable.

4.4.1 Arithmetic Mean

What the *expected value* is for a random variable's distribution, is the **arithmetic mean** for a sample of its outcomes and is denoted with a bar on top of the variable.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.18}$$

The relation of the arithmetic mean $\bar{x}$ of n samples of a random variable X with expected or mean value $E(X) = \mu$ is

$$E(\bar{x}) = \mu. \tag{4.19}$$

The running arithmetic mean for 10'000 Pareto-distributed random variables from fig. 4.6 together with its expected value of 6 is illustrated in fig. 4.7.

4.4.2 Measures of Dispersion

The standard deviation σ of the distribution function is the **standard deviation** s for its sample outcomes. The standard deviation is a measure of dispersion and gives a numerical indication of the "scatteredness" of a sample of numbers. It is the square root of the sample variance. There are two common textbook definitions for the standard deviation s of a data vector X:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.20}$$

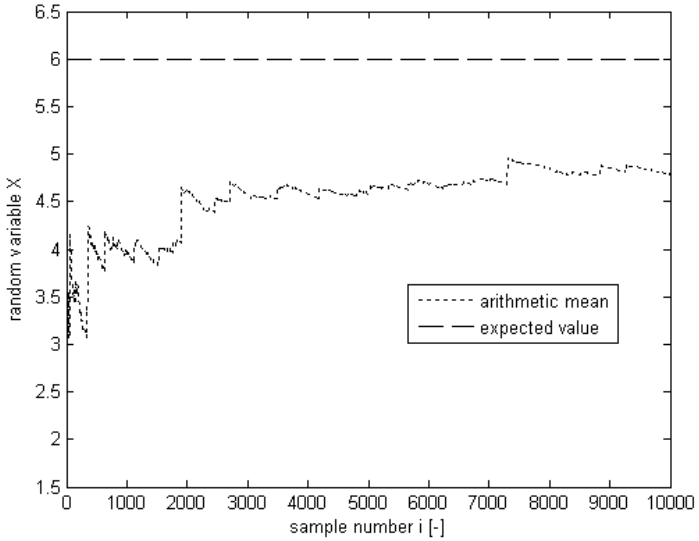

Figure 4.7: Running arithmetic mean $\bar{x}$ and expected value μ for 10'000 Pareto-distributed random variables of fig. 4.6 with shape parameter $\alpha = 1.2$ and location parameter $k = 1$.

and

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{4.21}$$

where n is the number of elements in the sample. Eq. 4.20 uses $n - 1$ as divisor. Its result s is the square root of an unbiased estimator of the variance of the population from which X is drawn, as long as X consists of independent, identically distributed samples. The divisor of eq. 4.21 is n. Its result s is the second moment of the sample about its mean.

4.4.3 Quantiles

The p**th quantile** (or "p-quantile") of the cumulative distribution function (cdf) F is defined to be that value x_p such that

$$F(x_p) = p, \tag{4.22}$$

or

$$P(X \leq x_p) = p. \tag{4.23}$$

If F is the cdf of a continuous random variable and is strictly increasing on some interval I and that $F = 0$ to the left of I and $F = 1$ to the right of I (I may be unbounded), then x_p is uniquely defined with the inverse cdf as

$$x_p = F^{-1}(p). \tag{4.24}$$

For the Pareto distribution, the pth quantile x_p is given by

$$x_p = F^{-1}(p) = k(1-p)^{-\frac{1}{\alpha}}. \tag{4.25}$$

The influences of the parameters α and k are shown in fig. 4.8 and 4.9. Quantile sizes increase very fast for heavy-tailed distributions if $p \to 1$. Names for special cases of p are listed in table 4.1.

Example: If the first percentile ($p = 0.01$) of a data series containing packet delays in a data network is 0.1 second ($x_p = 0.1\ s$), then 1 % of the data points are less or equal 0.1 second and 99 % of the data points are greater than 0.1 second.

p**th Quantile**	**Quantile Name**
0.5	median
0.25	lower quartile
0.75	higher quartile
k*0.01	k-th percentile

Table 4.1: Quantile names for special values of p.

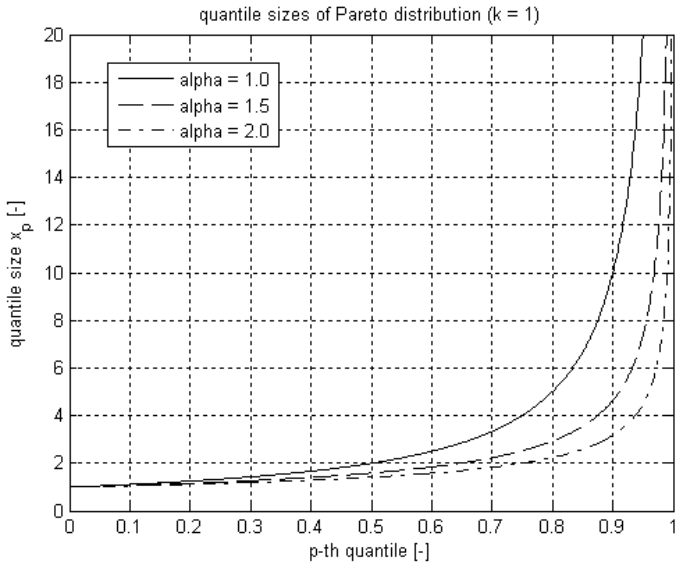

Figure 4.8: Quantile sizes x_p as a function of the pth quantile and shape parameter α of a Pareto distribution with constant location parameter $k = 1$.

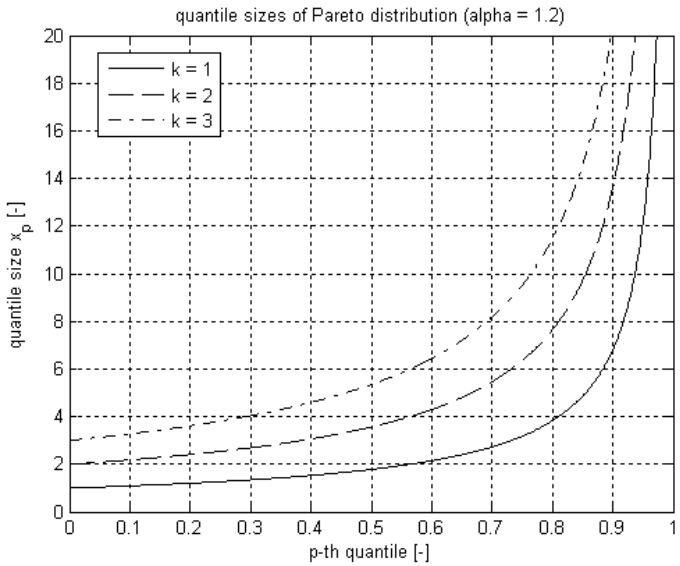

Figure 4.9: Quantile sizes x_p as a function of the pth quantile and location parameter k of a Pareto distribution with constant shape parameter $\alpha = 1.2$.

To apply the quantile measure to a sample of data, the data series has to be ordered first for its increasing values as shown in fig. 4.10 for $n = 10$ samples. In this figure, the median (= 0.5-th quantile) is the $0.5 \cdot n$-th sample's value (= 3) of the ordered series. That is, 50 % (= 5) of the samples are less or equal to 3 and 50 % are larger than 3.

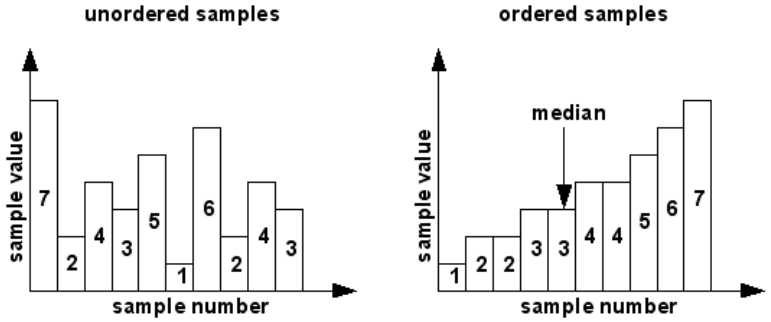

Figure 4.10: 10 unordered (left graph) sample values are ordered (right graph) to determine its median (= 0.5-th quantile).

Quantile calculations are also useful to determine the expected maximum value of any randomly distributed, finite series with cdf $F(x)$. If the series consists of n values, then the expected maximum value corresponds to the $(n/n+1)$-th quantile of the cdf. The p-th quantile of the Pareto distribution is given in eq. 4.25. Replacing the quantile size p by $(n/n+1)$ yields to

$$x_p = F^{-1}(p) = k(\frac{1}{n+1})^{-\frac{1}{\alpha}}. \tag{4.26}$$

Example: If a series contains 100'000 Pareto-distributed file sizes with location parameter $k = 2'167$ bytes and shape parameter $\alpha = 1.2$, then the largest expected file size in the series is 30 MB.

4.5 Convergence of Statistical Properties of Heavy-Tailed Distributions

4.5.1 Convergence of the Arithmetic Mean

Fig. 4.7 shows that the *arithmetic mean* $\bar{x}$ of a population of Pareto-distributed random values does not converge fast to the distribution's expected value μ. Even with 10'000 samples as in fig. 4.7, the arithmetic mean is just about 80 % of the expected value of 6. The slow convergence of the arithmetic mean of a random variable that follows heavy-tailed distributions can be understood quite easily: heavy-tailed distribution outcomes are characterized as exhibiting many small observations mixed in with a few large observations. In such data sets, most of the observations are small, but most of the contribution to the sample mean (or variance) comes from the few large observations. Heavy-tailed distributions behave quite differently from the distributions more commonly used, such as the Normal distribution or the exponential distribution, which have tails that decline exponentially (or faster). In contrast, because their tails decline relatively slowly, the probability of very large observations, occurring when sampling random variables that follow heavy-tailed distributions, is non-negligible.

The instability that is expected in simulations with heavy-tailed inputs may exhibit two features: first, they will be very slow to converge to steady state and second, they will show highly variable performance at steady state. Crovella and Lipsky [25] analyzed the behavior of the arithmetic mean $\bar{x}$ of Pareto-distributed random samples (sample mean). They formulated the equation

$$|\bar{x}_n - \mu| \sim n^{\frac{1}{\alpha-1}} \tag{4.27}$$

which shows how slow the arithmetic mean $\bar{x}_n$ converges to the expected value μ. If the tail index α is close to 1, the arithmetic mean does not converge at all, reflecting the fact that the mean is infinite.

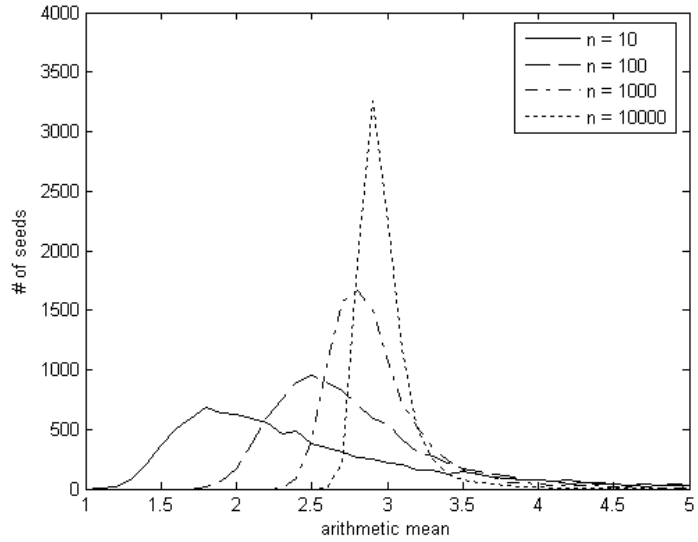

Figure 4.11: Arithmetic mean distributions for 10'000 values, each calculated from n Pareto-distributed random variables with $\alpha = 1.5$, $k = 1$ and $\mu = 3$.

Based on a Pareto-distributed random variable with $\alpha = 1.5$, $k = 1$ and expected mean $\mu = 3$, fig. 4.11 is drawn by plotting the distribution of 10'000 arithmetic mean values for n random variables. For larger samples sizes n the mean distributions converge slowly toward the expected value μ.

Example: From fig. 4.11 can be read that if taking the arithmetic mean from 100 samples ($n = 100$), most of these values will be 2.5, where the curve for $n = 100$ has its peak.

Crovella and Lipsky [25] also formulated a approximation to estimate how many samples n are necessary, to "highly likely" approximate the arithmetic mean to the expected values of its distribution with a accuracy of k digits.

$$n \geq c \cdot 10^{\frac{k}{1-\frac{1}{\alpha}}} \tag{4.28}$$

Supposing $c \approx 1$, table 4.2 lists the necessary number of samples n to achieve k-digits accuracy for different tail indices α. This table also shows that as α is close to 1, the number of samples necessary to obtain convergence in the arithmetic mean explodes. Thus, it is not feasible in any reasonable amount of time to observe steady state in simulation with Pareto-distributed variables. Over any reasonable time scale, such simulations are always in transient state when α is close to 1 and they also show high variability in steady state as analyzed in [25].

α	$n\,(k=1)$	$n\,(k=2)$
2.0	10^2	10^4
1.7	$2.7 \cdot 10^2$	$7.2 \cdot 10^4$
1.5	10^3	10^6
1.2	10^6	10^{12}
1.1	10^{11}	10^{22}

Table 4.2: Number of samples necessary to achieve k-digit accuracy of the arithmetic mean $\bar{x}$ to the expected value μ for different tail indices α.

4.5.2 Convergence of the Quantile

Fiedler and Plattner [4] explored the convergence of quantiles of heavy-tailed object size distributions. From probability theory it is known that the object size *quantile* of a heavy-tailed distribution converges to a normal distribution at rate $n^{-\frac{1}{2}}$, where n is the sample size. This convergence is fundamentally different from the convergence of the sample's average (arithmetic mean) object size to an α-stable distribution at rate $n^{\frac{1}{\alpha-1}}$ (eq. 4.27), where α is the tail index of the heavy-tailed object size distribution. The two convergence rates of the quantile and the average are shown in fig. 4.12 on a log-log scale as a function of the sample size n.

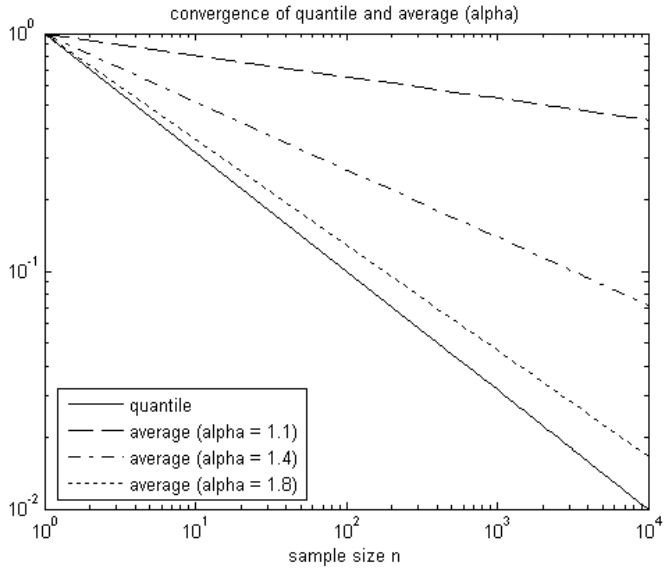

Figure 4.12: Convergence rates of the quantile and the average for different tail indices α (log-log scale).

As a consequence, the amount of time required to converge object size quantiles of interest is magnitudes smaller than the amount of time required to converge the average object size. Based on a Pareto-distributed random variable with $\alpha = 1.5$, $k = 1$ and expected mean $\mu = 3$, fig. 4.13 is drawn by plotting the distribution of 10'000 median (= 0.5-th quantile) values for n random variables. Comparing this plot with the distribution of the arithmetic mean in fig. 4.11, the faster convergence of the quantile becomes visible (attention: different x-scales!).

4.6 Data Analysis

Any data, simulation inputs and outputs, is almost useless if not meaningfully interpreted for properties of interest. Analysis methods discussed in this section can help to interpret data by extracting the relevant information.

4.6.1 Model Validation

Regardless of the approach, validation is a key step before using results to draw conclusions [19]. In fact, validation is a step that is performed repeatedly during the model development, as one continues to add enhancements. By verifying fundamental behaviors of the model at each step, one can maintain a high degree of confidence, and easily identify which particular changes are responsible for new, unexplained behavior. In contrast, if many simultaneous changes are made before examining the model's behavior, it will be impossible to determine which changes are responsible for certain observations, and to what extent each change has contributed.

Therefore, model validation is done for each change or enhancement of the project. Its results are not discussed in this work, unless they deviate from the expectations.

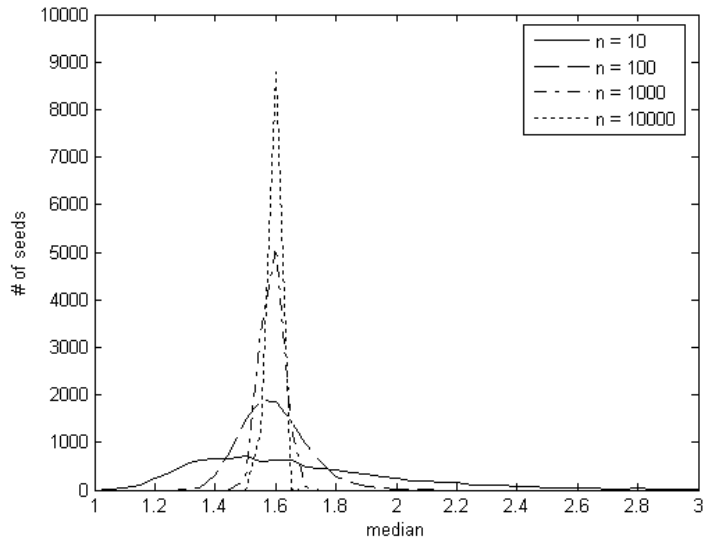

Figure 4.13: Median distributions for 10'000 values, each calculated from n Pareto-distributed random variables with $\alpha = 1.5$, $k = 1$ and $\mu = 3$.

4.6.2 Data Analysis Fundamentals

If model input data underlie random processes, every simulation run with a different seed[3] for the input data produces different output data (results). This is identical to the variations in real systems where each measurement is different. Because the analyzed statistic may be subject to fluctuation, the results of one simulated activity can not be completely trusted. How many total samples are needed is a difficult question to answer. One of the reasons it is difficult is that this depends on the variability of the quantity of interest. Confidence can be established for a set of gathered samples by extracting a range from the samples and giving a likelihood that the real average lies somewhere in that range. By real average is meant the value one would obtain if a vast numbers of samples would be measured in the real system and computed their arithmetic average. Therefore, gathering more samples enhances the confidence, but never provides a 100 % guarantee. To increase the certainty, the range of the seeking average value can be increased, or more samples can be obtained.

While one can rely on its intuition to tell if enough samples have been obtained, or some rules of thumb (many practitioners recommend the number 30 as a good number of samples [19]), there are mathematical methods for determining confidence: confidence intervals. When using confidence intervals, there is an important fact to be aware of concerning the relationship between the samples: these samples must be independent. In other words, the fact that the first sample was obtained should not influence the likelihood that the second sample would have been obtained.

4.6.3 Variance-Time (V-T) Statistics

Self-similarity or **Hurst** parameter H captures the intuitive notion of burstiness in a mathematically rigorous manner. Its value lies in the range of $0.5 \leq H \leq 1$, where 1 characterizes very bursty traffic (perfect self-similar traffic) and 0.5 less bursty traffic (poisson traffic). A interesting feature of self-similar traffic is that aggregating traffic, i.e. adding several self-similar traffics, produces not a smoother traffic but even burstier one [6]. Another conclusion from measured traffic analysis in Leland et al. [6] is that burstiness of traffic increases as traffic intensity increases.

A self-similar time series has the property that when aggregated (leading to a shorter time series in which each point is the sum of multiple original points), the new series has the same

[3] A seed can be interpreted as the initial state of a random number generator. Identical seeds lead to identical series of random numbers.

autocorrelation function as the original [6]. That is, given a (wide-sense stationary) stochastic process

$$X = (X_t; t = 0, 1, 2, \ldots), \tag{4.29}$$

as the first series ($m = 1$) in fig. 4.14, the new *m-aggregated* series denoted

$$X^{(m)} = (X_k^{(m)}; k = 1, 2, 3, \ldots) \tag{4.30}$$

is given for each $m = 1, 2, \ldots, X^{(m)}$ by

$$X_k^{(m)} = \frac{X_{k \cdot m - m + 1} + \ldots + X_{k \cdot m}}{m}; k = 1, 2, 3, \ldots, \tag{4.31}$$

summing for each $m = 1, 2, 3, \ldots$ the original series X over non-overlapping blocks of size m as shown in fig. 4.14 for $m = 2$ in the second series and $m = 4$ in the third series.

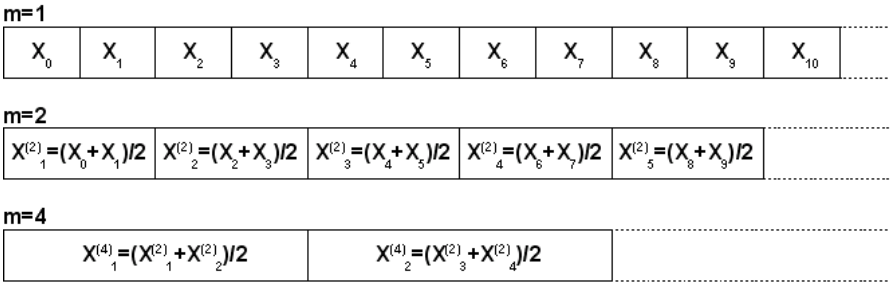

Figure 4.14: Aggregation of time series for aggregation levels m at power of 2.

Then, if X is (exactly second-order) self-similar, the corresponding aggregated processes $X^{(m)}$ have the same autocorrelation structure

$$R^{(m)}(\tau) = R(\tau), m = 1, 2, \ldots; \tau = 1, 2, \ldots \tag{4.32}$$

for all m. The autocorrelation function $R(\tau)$ is given by

$$R(\tau) = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)]}{E[(X_t - \mu)^2]}, \tau = 0, 1, 2, \ldots \tag{4.33}$$

Note: This means that the series is *distributionally* self-similar: the distribution of the aggregated series is the same (except for changes in scale) as that of the original.

The **Variance-Time (V-T) Plot** relies on the slowly decaying variance of a self-similar series. The variance

$$Var(X^{(m)}) = E\big([X^{(m)} - E(X^{(m)})]^2\big) \tag{4.34}$$

of $X^{(m)}$ is plotted against m on a log-log plot. A straight line with slope β greater than -1 is indicative of self-similarity, and the Hurst parameter H is given by

$$H = 1 - \frac{\beta}{2}. \tag{4.35}$$

Leland et al. [6] suggest to ignore the small values for m to estimate the degree of self-similarity. Lamparter [20] describes the Variance-Time Plot for small time series not very reliable, but when many data points are available (i.e. millions), this kind of data analysis is very useful and delivers a quite accurate picture of the existent self-similarity.

In the Design section 5.5, the implementation of a Variance-Time Plot calculation in MATLAB will be provided.

4.7 The ON/OFF Model

Park and Willinger [8] and many other researchers in the area of self-similarity describe the heavy-tailed transfer size distributions the most simple, distilled, yet high-level physical explanation of network traffic self-similarity. In this section, a self-similar traffic model based on heavy-tailed transfer object sizes is introduced, called the **ON/OFF Model**.

The ON/OFF Model of Willinger [9] [8] establishes that the superposition of a large number of independent on/off sources with heavy-tailed on- and/or off-periods leads to self-similarity in the aggregated process - a fractional Gaussian noise process - whose long-range dependence is determined by the heavy-tailedness of the on- or off-periods. The model has its theoretical foundation in a certain renewal reward process together with the empirical evidence of heavy-tailed on/off durations.

The ON/OFF Model considers n independent traffic sources $X_i(t), i \in [1, n]$, where each is a 0/1 renewal process with independent and identically distributed on- and off-periods. This just means that $X_i(t)$ takes on the values 1 (”on”) and 0 (”off”) on alternating, non-overlapping time intervals called on- and off-periods. $X_i(t) = 1$ is interpreted as there being a packet transmission. Thus, an on-period can be viewed as constituting a ”packet train”. Three of such on/off sources and their superposition are depicted in fig. 4.15.

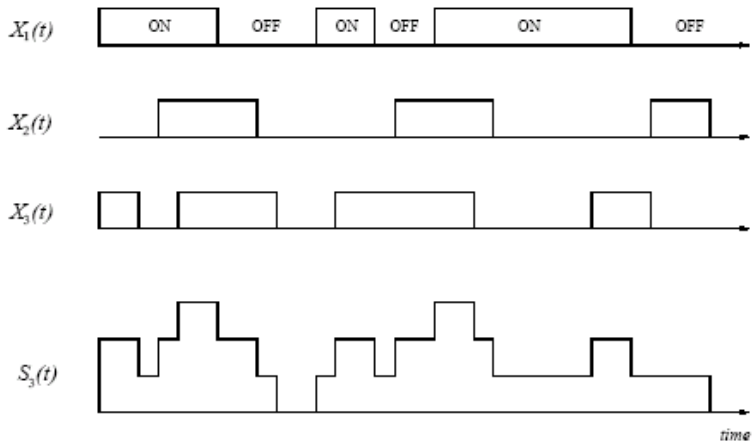

Figure 4.15: $n = 3$ on/off sources $X_1(t), X_2(t), X_3(t)$ and their aggregation $S_3(t) = X_1(t) + X_2(t) + X_3(t)$ (taken from Willinger [8]).

Let $S_n(t) = \sum_{i=1}^{n} X_i(t)$ denote the super positioned traffic at time t and $Y_n(T) = \int_0^T S_n(t)dt$ the cumulative process. $Y_n(T)$ measures the total traffic up to time T. If the on- and off-periods are heavy-tailed distributed random variables and n and T are large, then $Y_n(T)$ behaves asymptotically as fractional Brownian motion and is long range dependent [8] [9] with Hurst parameter

$$H = \frac{3 - \alpha}{2} \tag{4.36}$$

where α is the tail index of the heavy-tailed on- and/or off-periods.

4.8 Circuit-Switching

Since the beginning of the telephony technology, its circuits have always been switched. **Circuit-switching** means to physically connect both communication endpoints with a communication line by switches. When a person places a telephone call, the switching equipment within the telephone system seeks out a physical path all the way from the caller’s to the receiver’s telephone. Circuit-switching has the advantage that both communication endpoints exclusively are using the line and therefore a fixed bandwidth is guaranteed as long as the circuit is established. One of the disadvantages is, if the communication partners are not exchanging any data (e.g.,

nobody is talking on the phone), the line is idle and link capacity is wasted. In contrast to circuit-switching, for data transport between computers another technique to share common resources is mostly used, called **packet-switching**. More information about circuit- and packed-switching can be found in general communication network books like Tanenbaum [12] or Peterson and Davie [13].

In the past, transmission throughout the telephony system was completely analog. With the emerging of digital technologies, all lines and switches are now digital, leaving the *local loop*, that's the two-wired connection between each subscriber's telephone and the *end office*[4], as the last piece of analog technology in the system. In the end office, the analog signal arriving from the local loop is converted to a digital signal. Because there are no high-fidelity claims to the speech quality, a moderate sample rate of 8 kHz (e.g., samples are taken every 125 μs) was chosen which is able to code analogue frequency of up to 4 kHz according to Shannon's sampling theorem. 8 bits = 1 byte are taken to represent the sampled value, resulting in a data rate of 64 kbits/s = 8 kHz $\cdot$ 8 bits for one telephone channel.

The end office is just one place where calls are aggregated. Traditionally, the next aggregation level is an area and then a entire country. **Time division multiplexing (TDM)** is used to collect multiple calls and transmit them over one single line with sufficient capacity, like an optical fiber. Outside North America and Japan, 32 channels are multiplexed to 1 single carrier, the **E1** CCITT[5] standard. 32 $\cdot$ 8 bits = 256 bits is then the amount of data transmitted in a frame of 125 μs and 256 bits $\cdot$ 8 kHz = 2.048 Mbps gives the gross data rate of a E1 carrier. Actually, there are only 30 of the 32 channels used for voice traffic, the remaining 2 channels are used for synchronization purpose and other control data. **T1** is the name of the corresponding carrier system in North America and Japan, multiplexing 24 voice channels with a gross data rate of 1.544 Mbps. TDM allows multiple E1/T1 carriers to be multiplexed into higher-order carriers. The T multiplexing family is called 1.5 Mbit/s hierarchy and the E multiplexing family 2 Mbit/s hierarchy [31]. The common name for both hierarchies is the **Plesiochronous Digital Hierarchy (PDH)** [31]. The T1 and E1 carrier are also called Nx64 kbit/s services [11], where N is the number of channels and 64 kbit/s the bit rate of a single channel.

In this work, one E1 carrier is used to model CE traffic. Design section 5.2 provides a traffic model for the E1 carrier in the network simulations.

4.9 Differential Service with Virtual LAN (VLAN)

Differential services allow to distinguish traffic flows and assigning them individual treatment. In particular, dedicated traffic can be given preferential treatment. This is of special interest to provide higher priority treatment to QoS sensitive traffic like circuit emulation traffic over QoS non-sensitive traffic like best-effort data. In the light of this work, preferential treatment of CE traffic could improve its QoS by reducing end-to-end delays and loss rates, the primary QoS requirements for CE services (see section 5.4). End-to-end delay can be reduced by preferential treatment of higher prioritized CE traffic in buffer queues. Losses mainly originate from packet drops by buffer overflows. Preventing higher prioritized packets from dropping and deleting low priority traffic instead, can reduce losses of the CE traffic.

One implementation of a differential service on the (Ethernet) link layer is the **Virtual LAN (VLAN)** standard **IEEE 802.1Q** [16] together with the priority standard **IEEE 802.1p** [17]. The name "Virtual LAN" comes from the main purpose of a VLAN which is to decouple the logical topology from the physical topology, but as a matter of fact, VLAN introduced something that is surprisingly similar to a connection. The VLAN standard 802.1Q changed the Ethernet header. The new format contains a VLAN tag. In order to make the new format compatible with old Ethernet adapters, the VLAN fields are only actually used by the bridges and switches and not by the individual user machines. New Ethernet adapter generally support the 802.11Q standard and write themselves the VLAN ID in the tag. This way, VLAN supporting bridges and switches can learn the VLAN topography and do not have to be individually programmed.

The new tag (see fig. 4.16), inserted between the source address and the length field of the "old" Ethernet header, is called **Tag Control Information (TCI)**. The *user_priority* field is

[4] The telephone company's nearest concentration point, also called *local central office*.

[5] Commité Consultatif International Télégraphique et Téléphonique. The new name for this organization is ITU-TSS (International Telecommunications Union - Telecommunications Standardization Sector).

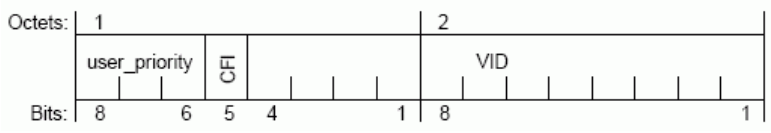

Figure 4.16: IEEE 802.1Q standard Tag Control Information (TCI).

three bits in length, interpreted as a binary number. It is therefore capable of representing eight priority levels, 0 through 7. The use and interpretation of this field is defined in ISO/IEC 15802-3 [18].

user_priority	Acronym	Traffic type
1	BK	Background
2	—	Spare
0 (Default)	BE	Best Effort
3	EE	Excellent Effort
4	CL	Controlled Load
5	VI	"Video," < 100 ms latency and jitter
6	VO	"Voice," < 10 ms latency and jitter
7	NC	Network Control

Figure 4.17: Correspondence between traffic types and default *user_priority* (ISO/IEC 15802-3 standard [18]).

The forwarding process may provide more than one transmission queue for a given bridge port. Frames are assigned to storage queue(s) on the basis of their *user_priority* using a traffic class table that is part of the state information associated with each port. The list in fig. 4.17 indicates, for each possible value of *user_priority*, the corresponding value of traffic class that shall be assigned. Priority value 0 represents the lowest value of user priority, and 7 the highest value. Queues correspond one-to-one with traffic classes.

4.10 Network Simulation Environment

Simulation Modeling is becoming an increasingly popular method for network performance analysis. Generally, there are two forms of network simulation: **analytical modeling** and **computer simulations**. The first method is a mathematical analysis that characterizes a network as a set of equations. The main disadvantage is its simplistic view of the network and inability to simulate the dynamic nature of a network [30]. Thus, the study of a complex system always requires a **discrete event simulation** package, which can compute the time that would be associated with real events in a real-life situation [30]. Software simulator is a valuable tool especially for today's network with complex architectures and topologies. Designers can test their new ideas and carry out performance related studies without "trial and error" hardware implementations.

A typical network simulator can provide the programmer with the abstraction of multiple threads of control and inter-thread communication. Functions and protocols are described either by finite-state machine, native programming code, or a combination of the two. A simulator typically comes with a set of predefined modules and user-friendly graphical user interfaces (GUIs). Some network simulators even provide extensive support for visualization and animation.

4.10.1 OPNET Modeler

For the network simulations in this thesis, the commercial software OPNET[6] Modeler®Release 10.5A PL3 (Build 2570) [19] is given in the assignment as the network simulator of choice. A educational-use license has been purchased permitting exact one active simulation runtime and giving access to OPNET online support. For all simulations, a standard desktop computer with AMD Athlon XP 2800+ (2.07 GHz) CPU, 1 GB RAM and 80 GB hard disk running operation system Window XP (SP2) is employed.

OPNET (Optimized Network Engineering Tool) Modeler was originally developed at MIT and introduced in 1987 as the first commercial network simulator. OPNET provides a comprehensive development environment for the specification, simulation and performance analysis of communication networks. A large range of communication systems from a single LAN to global satellite networks can be supported. *Discrete event simulations* are used as the means of analyzing system performance and their behavior. The following are some of OPNET Modeler's key features (taken from http://www.opnet.com/products/modeler/home.html):

- Most **scalable and efficient simulation** engine, enabling the fastest simulation runtimes using advanced acceleration techniques for wired and wireless models.
- **Hierarchical network models**. Manage complex network topologies with unlimited subnetwork nesting.
- **Object-oriented modeling**. Nodes and protocols are modeled as classes with inheritance and specialization.
- **Clear and simple modeling paradigm**. Model the behavior of individual objects at the "Process Level" and interconnect them to form devices at the "Node Level"; interconnect devices using links to form networks at the "Network Level". Organize multiple network scenarios into "Projects" to compare designs (see fig. 4.18).
- **Finite state machine modeling** of protocols and other processes.
- **Wireless, point-to-point, and multi point links**. Link behavior is open and programmable. Accurately account for delay, availability, bit errors, and throughput characteristics of links. Incorporate physical layer characteristics and environmental effects using a library that includes enhanced TIREM, Longley-Rice, and Free Space, or interface with custom propagation models.
- **Total openness**. APIs for program-driven construction or inspection of all models and result files. Easily integrate existing code libraries into your simulations. Source code provided for all standard models.
- **Integrated analysis tools**. Comprehensive tools to display simulation results. Easily plot and analyze time series, histograms, probability functions, parametric curves, and confidence intervals. Export to spreadsheets or use XML.
- **Import data** from text files, XML, and popular tools from Cisco, HP, NetScout, BMC, Concord, Sniffer, Infovista, MRTG, cflowd, tcpdump, and others.
- **Comprehensive library of detailed protocol and application models**. Including Multi-Tier Applications, Voice, HTTP, TCP, IP, OSPF, BGP, EIGRP, RIP, RSVP, Frame Relay, FDDI, Ethernet, ATM, 802.11 Wireless LANs, MPLS, PNNI, DOCSIS, UMTS, IP Multicast, Circuit Switch, MANET, Mobile IP, IS-IS, and many more. Provided as FSMs with open source code.
- **. . . and network devices**: The Standard Model Library includes hundreds of vendor specific and generic device models including routers, switches, workstations, and packet generators. Quickly assemble your own device models using the "Device Creator." Aggregate traffic from LANs or "Cloud" nodes.
- **Windows NT, 2000, XP, and UNIX**. Models can be shared across hardware and platforms transparently without modification.

[6] OPNET Technologies, Inc. OPNET is a registered trademark of OPNET Technologies, Inc., 2004

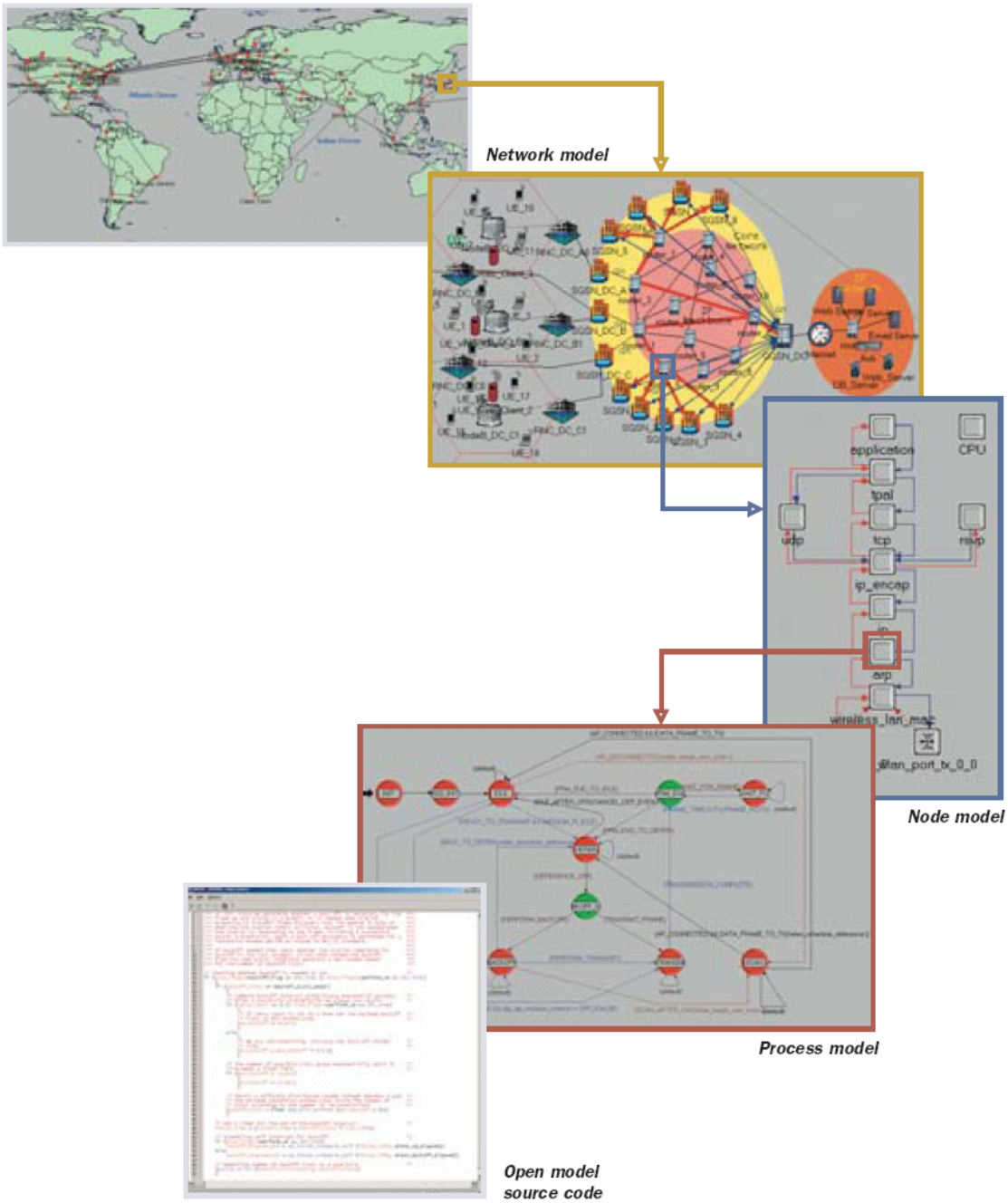

Figure 4.18: Hierarchical modeling structure of a ”Project” (top) in OPNET Modeler.

Chapter 5

Simulation Design

Before running network simulations to investigate the problems identified in chapter 3, the entire simulation setup has to be designed, tested and analyzed. The network simulator of choice, given in the assignment, is OPNET Modeler. OPNET Modeler is a widely used commercial discrete event simulator. More details about OPNET Modeler is given in the Background chapter in section 4.10.1. The following sections describe how the problem is attacked and provide solution proposals which are used in the simulations.

5.1 Topology

The topology for all simulations is given in the assignment and is a metropolitan Gigabit Ethernet *bottleneck* network. The bottleneck link may be viewed in a more general sense, since many have argued that there is always a single bottleneck link on any network path which is usually not fast moving [28]. The networks simulations are limited to the physical and the link (MAC) layer.

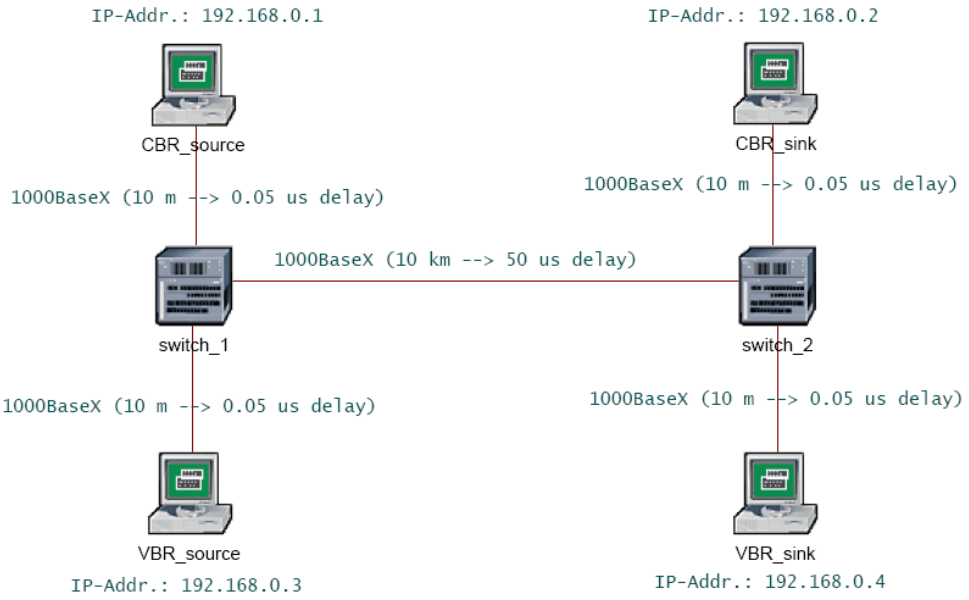

Figure 5.1: Bottleneck topology for all network simulations with OPNET Modeler.

The network technology for all simulations is switched Gigabit Ethernet with optical links (1000BaseX) running at 1 Gbps. In switched Ethernets, all links are full-duplex and no medium contention takes place. The bit error rate (BER) of optical fibers are almost zero. This is in favor of the focus of this work which lies on frame losses due to buffer overflows rather than due to transportation errors. The propagation delays to and from the bottleneck link to the sending

and receiving nodes shall be negligible compared with the delay over the bottleneck link itself. Hence, as shown in fig. 5.1, 10 m access links to the bottleneck switches are chosen and 1 km for the bottleneck link. Frame bursting is disabled for all network adapters.

The **variable bit rate (VBR)** source node *VBR_source* sends variable, best effort workload data traffic (see section 5.3) to the VBR destination node *VBR_sink*. The **constant bit rate (CBR)** source node *CBR_source* sends CE traffic (see section 5.2) of a E1 carrier at constant bit rate to the CBR destination node *CBR_sink*. At the bottleneck entry *switch_1*, VBR and the CBR traffic are concentrated on a shared link - the bottleneck link. At the other side of the shared link, the bottleneck exit *switch_2* splits off the VBR and CBR traffic and sends each one to its destination node. Because in a switched Ethernet there is no competition for the medium (full-duplex), all simulations are done with traffic in only one direction to keep the simulation effort at a minimum without loosing any important model characteristic.

5.2 Circuit Emulation Traffic Generation

The circuit emulation (CE) traffic originates from connection-oriented, circuit-switching telephony services. A general introduction to circuit-switching is given in the Background chapter in section 4.8. This section provides a specific model for a E1 carrier representing the CE traffic in the simulations.

Knowing the E1 carrier properties from section 4.8, the next step is to send it over a Gigabit Ethernet. Since the delay has to be kept as small as possible, it's certainly better to take each individual E1 carrier packet of 125 μs and send it immediately instead of accumulating them to fill a Ethernet frame up to its maximum payload size of 1'500 bytes. Because the Ethernet frame should also make it through network (IP) layer routers, a IP header of 20 bytes is first attached to the E1 packet of 32 bytes, giving 32 bytes + 20 bytes = 52 bytes payload for the Ethernet frame, just 6 bytes more than the minimum Ethernet payload size of 46 bytes. In a switched, full-duplex Gigabit Ethernet there is no medium contention possible. Therefore, the IEEE 802.3z standard for *carrier extension* (padding the normal frame to 512 bytes) is not necessary. The Ethernet adapter attaches then a total of additional 26 bytes (14 bytes header + 4 bytes checksum + 8 bytes preamble) to the 52 bytes of payload, resulting in 78 bytes for one entire Ethernet frame carrying 32 channels E1 telephone traffic of 125 μs (fig. 5.2). This leads to a gross data rate of $78 \cdot 8$ bits$\cdot$ 8 kHz = 4.992 Mbps. In the case of a VLAN topology, 2 more bytes (from TCI) are added to the Ethernet header, increasing the frame size to 80 bytes and the gross data rate to 5.120 Mbps. The results of these calculations are briefly summarized in table 5.1.

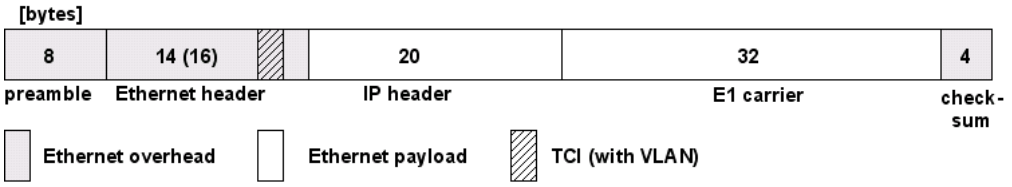

Figure 5.2: Ethernet frame carrying 32 channels E1 telephone data of 125 μs.

No. of channels	32
E1 frame size	32 bytes
Ethernet frame size (without VLAN)	78 bytes
Ethernet frame size (with VLAN)	80 bytes
Ethernet gross data rate (without VLAN)	4.992 Mbps
Ethernet gross data rate (with VLAN)	5.120 Mbps
Frame length	125 μs

Table 5.1: Characteristics of a E1 carrier running over Ethernet with IP header.

Taking into account the 96 interframe bits, total 720 bits (without VLAN) are send over a Gigabit Ethernet to transport 32 bytes of telephone data which gives a "transport efficiency" of

only 36 %. This may look like a waste of link capacity, but in consideration of the enormous capacity of 1 Gbps of the Gigabit Ethernet, it still uses only moderate 0.5 % of the total link capacity.

5.3 Data Traffic Generation

In the Introduction (chapter 2), the workload data traffic is identified to be bursty within a wide range of time scales. The mathematical notion for this property is *long-range dependency* or *self-similarity*. For the network simulations with OPNET Modeler, different approaches are possible to generate self-similar traffic. They are characterized by the location of the generation process (inside or outside OPNET Modeler), the mathematical model (including the parameter set) and the involved layers of the standard model. The following methods are considered in the network simulations to generate self-similar data traffic:

- **Heavy-tailed statistics**: In most places where statistical properties can be chosen in OPNET Modeler, heavy-tailed distributions are among the selection. This model can be applied on the link, network, transport and application layer.
- **Application traffic models**: OPNET Modeler includes a set of models for generating traffic based on standard applications such as FTP, HTTP, voice, and e-mail. This model is restricted to the application layer.
- **Raw Packet Generator** (RPG): Built-in OPNET traffic source model that generates self-similar traffic. This model can only be applied on the link and the network layer.
- **Scripted file playback**: External files (with self-similar data traffic) can be used as ”scripted” input data. This method can be used on the link, network, transport and application layer.

All of these methods are discussed in more detail in the next sections.

5.3.1 Heavy-Tailed Statistics

In the distribution dialog box, heavy-tailed distributions such as *Pareto* and *Weibull* can be used to model self-similar traffic patterns. Unfortunately, these heavy-tailed distributions are restricted to a given set of parameters. For the *ethernet_ip_station* node traffic generation, these distribution parameters are as in fig. 5.3 *Packet Inter-Arrival Time* and *Packet Size*. Unlimited number of rows can be added, each defining a independent traffic source. The totally generated traffic is the superposition of all traffic sources in every row.

There exist models to generate parameterized self-similar data traffic by superposing independent Bernoulli sources [24] or Lognormal distributions [23]. The main problem with these models is to find the right parameters to achieve the desired characteristic of the superposed traffic. This approach is not followed in this work in favor of the RPG and the scripted ON/OFF Model.

5.3.2 Application Traffic Models

OPNET Modeler includes a set of models for generating traffic based on standard applications such as FTP, HTTP, voice, and e-mail. The generic custom application model can also be used to represent a broad range of applications (such as multi-tier ERP transactions) that do not correspond to the traffic patterns of standard network applications. These traffic models are based on a server-client architecture and represent what is closest to a real scenario. Bursty data traffic can be generated in numerous ways. For example, the object size distribution of the page property in the HTTP application can be selected to be heavy-tailed.

Tremendous complex profiles can be set up, each containing infinite numbers of applications. Every application consists of (infinite) numbers of tasks and every task of (infinite) numbers of phases. Additionally, there are many more parameters for the serving nodes like CPU performance and cache sizes. All together, dozens of parameters can and must be selected to set

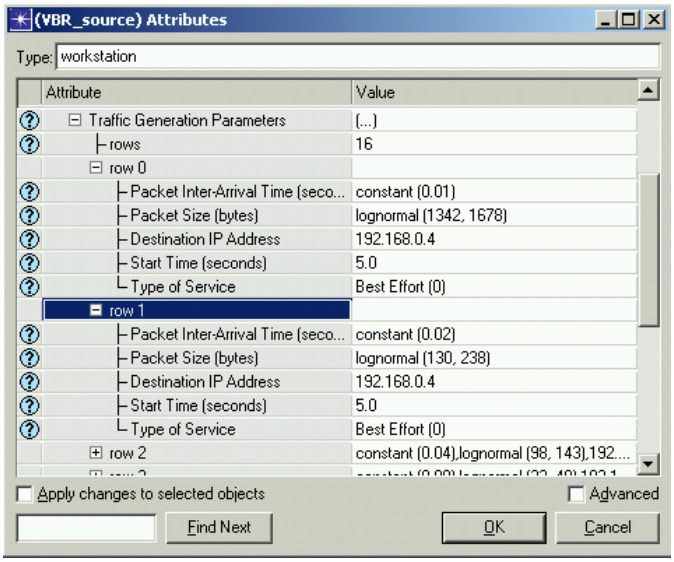

Figure 5.3: Traffic generation parameters of a *ethernet_ip_station* node

up a running simulation. This makes this approach complicated to generate well-defined self-similar data traffic. Furthermore, it only supports UDP and TCP transport protocols. Simulations directly over the Ethernet data link layer or IP network layer like in the simulations in this project are not possible. Therefore, this approach is also not followed in this work.

5.3.3 Raw Packet Generator (RPG)

The **Raw Packet Generator (RPG)** is a traffic source model implemented in OPNET Modeler that generates self-similar traffic. According the documentation, the RPG model can be used to generate any types of self-similar (background) traffic. The RPG module can be used over the network (IP) and link (MAC) layer of the standard model. A variety of **Fractal Point Processes (FPPs)**, based on a research paper by Ryu et al. [14], are implemented in the RPG model. For more information see *17–RPG Model User Guide* in the *Standard Models User Guide* of [19].

In the Project Editor, the RPG Object Palette (fig. 5.4) contains all available RPG nodes. The *ethernet_rpg_station* node runs directly over Ethernet, the *ethernet_rpg_wkstn* node over IP. The *RPG Traffic Generation Parameters* of a *ethernet_rpg_wkstn* node are shown in fig. 5.5. *Promoted* parameters are made globally configurable and can be easily changed for each simulation run. The following FPPs are integrated in the RPG (to be selected in the *Arrival Process Name* of fig. 5.5): ”Sup-FRP”, ”PowON-PowOFF”, ”PowON-ExpOFF”, ”ExpON-PowOFF”, ”Fractal Power-Law Filter”, ”Exponential Rectangular Filter” and ”Fractal Rectangular Filter”.

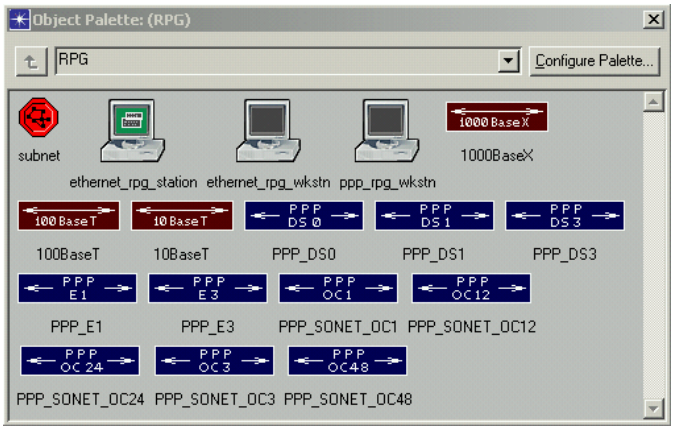

Figure 5.4: RPG Object Palette.

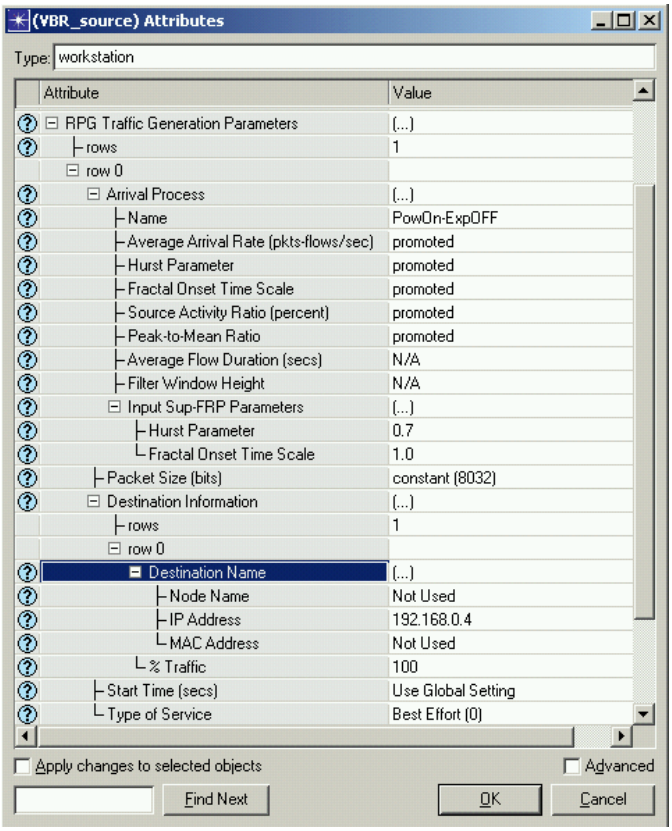

Figure 5.5: RPG Traffic Generation Parameters of a *ethernet_rpg_wkstn* node.

For the simulations in this work, only the ”PowON-ExpOFF” and ”PowON-PowOFF” arrival processes are of interest. Because the ”PowON-PowOFF” model does not allow to set the *Peak-to-Mean Ratio*, only the ”PowON-ExpOFF” function is considered for the simulations. The ”PowON-ExpOFF” model is characterized by the following *RPG Traffic Generation Parameters*:

- **Average Arrival Rate (AAR)**: Defines the average arrival rate for the aggregate traffic that is generated by all sources of the arrival process. For the ”PowON-ExpOFF” model, it is defined in packets/s. Together with the Packet Size it specifies the average generated traffic.

  To achieve an average network utilization of 1 % (see section 5.8), the product of (Packet size + IP header + Ethernet header) and *Average Arrival Rate* must give 1 % of 1 Gbps for the Gigabit Ethernet, i.e. 10 Mbps. Since the *Packet Size* (1’004 bytes) with IP header (20 bytes) is set to the constant value of 1 kB, the *Average Arrival Rate* results with $\frac{10\ Mbps}{(1\ kB+26\ byte)\cdot 8}$ to be 1’190.

- **Hurst Parameter (H)**: Defines the Hurst characteristic of the self-similar traffic source. This parameter determines the shape parameter for the Pareto distribution.

  As a reference value 0.90 is taken for all simulations for reasons discussed in section 5.7.

- **Fractal Onset Time Scale (FOTS)**: Used with the Hurst characteristic to determine the location parameter for the Pareto distribution. It is the smallest time unit from which the power-law behavior or fractal begins to appear.

  This value depends on the Ethernet traffic characteristic and lies usually between 0.1 ms and 1 s. A rule of thumb says the self-similar traffic starts at time scales of the dimension of the round-trip time (RTT). In this work, the RTT is approx. 0.1 ms.

- **Source Activity Ratio (SAR)**: Defines the percentage of time that at least one of the independent on/off traffic sources is active. The range is from 0 to 100.

  This parameter is difficult to determine. Many people recommend values around 75 %. Simulation results shall give further indications.

- **Peak-to-Mean Ratio (P2MR)**: Defines the ratio of peak traffic rate (bursts) over the mean traffic rate. The range is from 1 to infinity. This parameter should be used to adjust the desired variance of the generated traffic.

  The considered target of the (normalized) variance in this work should be higher than 10. With a trial-and-error method the P2MR shall be adjusted to reach the appropriate variance.

- **Packet Size**: Size of generated packets in bits. It is specified by a probability density function (pdf). Together with the *Average Arrival Rate* it defines the average generated traffic.

  For this work, only constant packet sizes between 26 and 1'480 bytes (46 and 1'500 bytes with added IP header) are of interest. Packets smaller than 46 bytes would automatically be padded to the minimum packet size of 46 bytes, packets bigger than the Ethernet Maximum Transfer Unit (MTU) of 1'500 bytes would be fragmented into several consecutive packets adding additional load to the network with its additional headers. Therefore, the *Packet Size* including the IP header of 20 bytes is more or less freely chosen somewhere in the upper region of its range to be 1 kB = 8'192 bits. Subtracting the IP header of 20 bytes, gives 8'032 bits for the *Packet Size*.

- **Destination Information**: Specifies the destination node of the traffic generated by the arrival process.

  In all simulations, a single data source (*VBR_source* node) sends its traffic over a bottleneck link to a single data destination node. Therefore, the destination for all simulations is the data sink (*VBR_sink*) node.

- **Start Time**: Specifies when the arrival process starts generating traffic.

  In this work, data traffic should be present during the entire recorded simulation period. Since data logging is started after 10 seconds, this parameter is given the fix value of 5 seconds for all simulations, allowing the system 5 seconds for stabilization before data is captured for analysis.

Data traffic generation simulation results with the RPG are presented and discussed in the Results section 6.1. Since the results show that the RPG is not suitable for the simulations in this work, it is not employed for the main network simulations.

5.3.4 Scripted File Playback

For the *Packet Generation Arguments* of a *ethernet_station* node, the *Interarrival Time* and the *Packet Size* can be read from a external file. The model reads in the file and replays its content line by line in cyclic order, that is when the last line is read, the process continues reading at the first line again. To specify such an input file, the "scripted" *Distribution name* must be selected as in fig. 5.6 for the *Packet Generation Arguments*. One has to be careful not to have multiple devices in the network using the same file, otherwise events may get synchronized. A scripted file can automatically be generated with a external program.

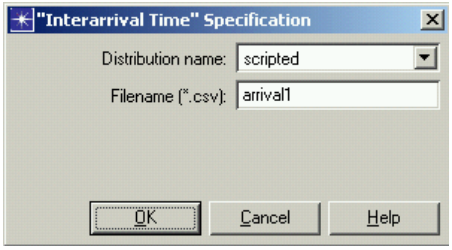

Figure 5.6: Input file 'arrival1.csv' selected for the scripted distribution of the *Interarrival Time*.

In this work, the ON/OFF Model, introduced in the Background chapter in section 4.7, is taken to generate self-similar data traffic with a MATLAB script to write the files 'arrival.csv' with

the interarrival time series and 'size.csv' with the file size series. The implementation of the ON/OFF Model in the MATLAB code `traffic_gen.m` is explained in the next section.

Data traffic generation simulation results with the scripted file playback using the ON/OFF Model is presented and discussed in the Result section 6.2. The results show that the ON/OFF Model is suitable for the simulations in this work and is employed for the main network simulations.

Implementation of the ON/OFF Model

In this section, the implementation of the ON/OFF Model as a self-similar data traffic source in a MATLAB script is explained. With this traffic generation model, n traffic sources send files which sizes are heavy-tailed (Pareto) as well as the waiting times between two consecutive files. This ON/OFF Model deviates from the ON/OFF Model by Willinger [9], introduced in the Background chapter in section 4.7, in two aspects: first, the number of sources n is finite and second, the file *sizes* are superposed rather than the traffic *flows*. In other words, a file is always send at the speed of the link. If two or more sources send a file at the same time ($X_1(t)$ and $X_2(t)$ in fig. 5.7), the superposed traffic is not sent faster ($X'(t)$ in fig. 5.7), but the file size is increased and consequently the transmission time ($X''(t)$ in fig. 5.7).

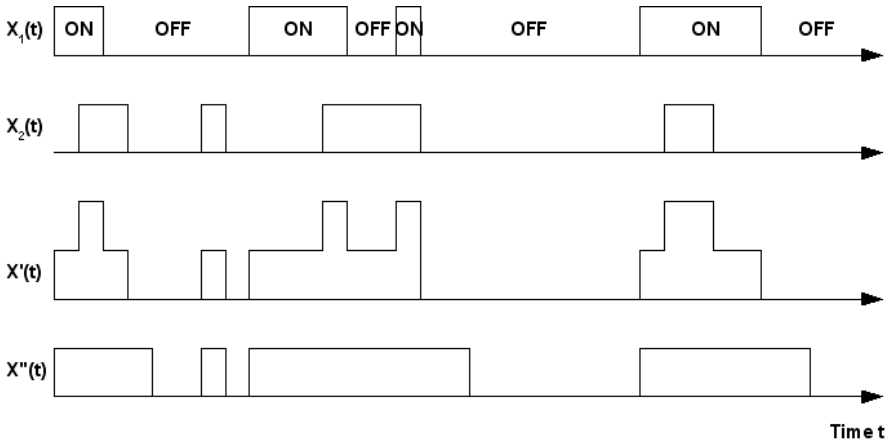

Figure 5.7: Superposed traffic flows $X'(t)$ and transmission times $X''(t)$ of two traffic sources $X_1(t)$ and $X_2(t)$.

The effect of the different superposition can be interpreted as a smoothing of peaks or put it differently, low-pass filtering of the superposed traffic. The simplification can be justified with the characteristics of real networks: link speeds are always limited and traffic peaks from many sources are filtered out by the buffers in the sending processes of traffic aggregation points such as network switches. Besides, it is in favor of the CE traffic because data traffic peaks can only decrease the QoS of the CE service but never increase it. In the light of the simulation results, this can also be formulated as follows: if the CE traffic does violate the QoS requirements with the traffic model discussed here, then it would also do it with Willinger's (original) model and the QoS is even expected to be worse. The following parameters are inputs of the model

- Link capacity
- Average link utilization
- Shape parameter α of file size Pareto distribution
- Shape parameter α of waiting time Pareto distribution
- Average file size
- Maximum file size

used to calculate the location parameters k of the file size and waiting time Pareto distributions. Eq. 4.15 is used to determine the location parameter k of the *unlimited* waiting time

distribution and the numerical method from eq. 4.16 to calculate k of the *limited* file size distribution. The maximum files size are limits in end systems such as the 4 GB file size limit inherent to most popular operating systems (OS). The Microsoft Windows OS, for instance, operates on three file systems with different file size limits: FAT16 and FAT32 with 4 GB and NTFS with 16 TB (Terabyte) limit. The average file size is 12 kB for all simulations with the ON/OFF Model according the paper from Feldmann et al. [38].

An open question is how large the number of sources n has to be. Willinger [9] states n has to be "large" in order that the resulting, superposed traffic becomes long range dependent with Hurst parameter $H = \frac{(3-\alpha)}{2}$. Student paper [36] gives indications that n must be higher than 10'000. Therefore, 100'000 sources are taken in this thesis. Higher number of sources start to cause problems with the soft- and hardware used for the simulations.

The MATLAB code `traffic_gen.m` in appendix D.2 generates random outcomes of the proposed ON/OFF traffic model. The traffic trace is then written into two files (file- or packet sizes and interarrival times) which the simulation software OPNET Modeler reads in as "scripted" data source. Unfortunately, the scripted data source in OPNET Modeler can not read waiting times between sent files but only interarrival times, that is the time between the sending start of two consecutive files as shown in fig. 5.8.

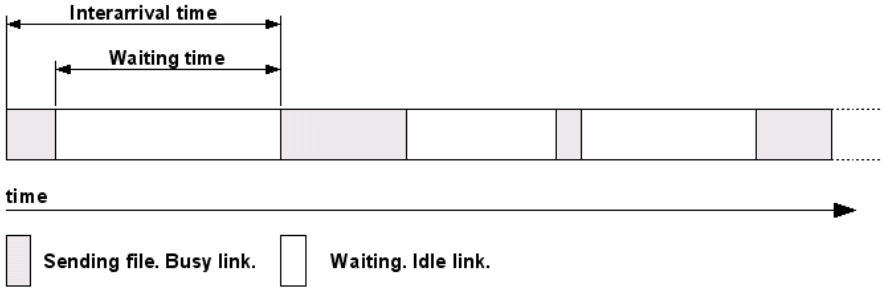

Figure 5.8: Difference between waiting time and interarrival time in a traffic flow.

Additionally, the traffic superposition algorithm only works with on- and off-times, but not with file sizes and off- or interarrival times. Therefore, there is no by-passing of the transformation from file sizes to transmission times and vice versa. The exact transformation would make the calculations considerably more complicated because the sending time of each file has to be calculated by fragmenting the file into frames of maximum and minimum sizes and adding headers and interframe gaps. The ON/OFF Model implementation in MATLAB code `traffic_gen.m` does therefore the following simplification: for the average sized file, a exact transformation from file size to transmission bits is done and the ratio (file size)/(transmission bits) used to transform all file sizes to transmission bits and the inverse ratio to transform any transmission bits back to file sizes. The relation of transmission bits to transfer time is given by (transmission bits)/r, where r is the transmission rate.

Example: If the average file size is 12 kB, the IP header 20 bytes, the Ethernet overhead 26 bytes and the interframe gap 96 bits as in this work, the exact number of transmission bits is 102'480 which leads to a ratio (file size [byte])/(transmission bits [bits]) of 8.33984375. The factor 8 comes from the byte-to-bit transformation and the digits after the point from the overhead and eventually the padding to the Ethernet minimum payload.

Because most files are smaller than the average sized file, this simplified transformation produces slightly too small transmission times for a entire file population. This is due the fact that the overhead ratio (including padding to minimum payload) is generally larger for small files. Nevertheless, the simplification can be justified with the back-conversion from transmission times to file sizes, which uses the same (inverse) ratio and therefore neutralizing to a large extend the introduced error. The error is only then not neutralized, when two or more small file transmissions are superposed to one larger transmission, but with average network utilization rates considered in this work of around 1 % (see section 5.8), these events are very seldom.

The traffic trace generation for a substantial number of traffic sources runs fairly quickly in MATLAB because it is purely matrix operation. Matrix calculations run fast in MATLAB (= MATrix

LABoratory) because it is specially designed to do so although it is a high-level programming language. The superposition of the traffic traces from all sources, in contrast, is not a matrix operation but a loop, which runs very slowly if the number of sources and the traffic trace length become large. Thus the superposition calculations are written in C-code (see appendix E.1). MATLAB features a special function called ”MEX-file” to integrate compiled C-code in the form of a dll-file (MS Windows) as if it was a generic MATLAB function.

Use `(X_file, X_arrival)=superpos(X_on,X_off,t_tot_min)` to call the C-code from MATLAB. The compiled C-code ”superpos.dll” has to reside in the same file directory as the calling MATLAB m-file. This code takes the sorted vectors `X_on` and `X_off` with on- resp. off-times of any number of traffic sources and returns the on- and interarrival-durations (`X_file` and `X_off`) of the superposed traffic up to time `t_tot_min`. Fig. 5.9 shows an example of two superposed source traces.

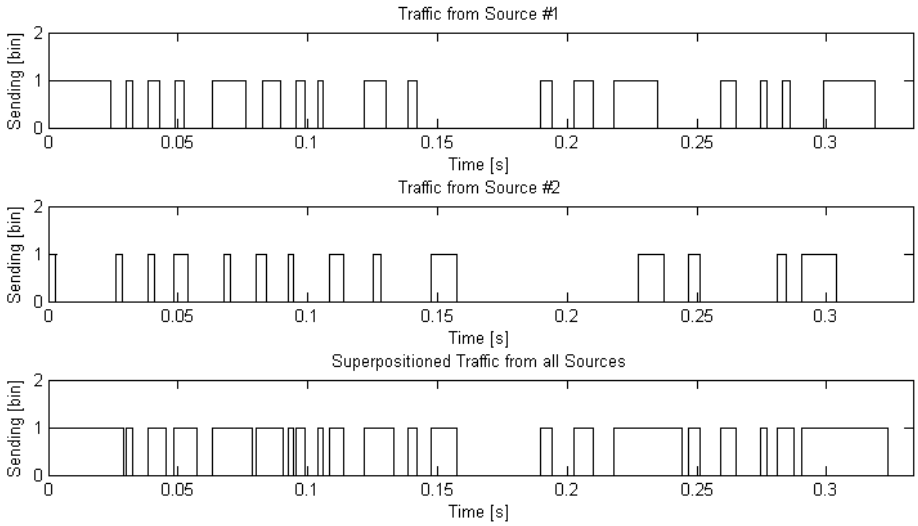

Figure 5.9: Superposition of two traffic traces from two sources with C code `superpos.c`. (High utilization rates are used for illustrating purpose.)

OPNET Modeler's standard settings of the *ethernet_ip_station* node for the attributes *IP/IP Processing Information/Memory Size (bytes)* is 16 MB. This memory size has to be increased to allow files (or packets) larger than 16 MB to be processed and transmitted correctly. Although this limit can be set to any value, 256 MB can be recommended because files exceeding this limit can not be handled anymore by the process `ip_traf_gen`, causing the program to abort. Thus the MATLAB code `traffic_gen.m` has been modified to by-pass that limitation: whenever a file size larger than 256 MB shows up, it is split into consecutive files of max. 256 MB with zero waiting time in between.

Simulation results with the scripted file playback using the ON/OFF traffic model are presented and discussed in the Result chapter in section 6.2.

5.4 Quality of Service (QoS)

Quality of Service (QoS) is the totality of features of a service that influences its ability to satisfy given needs. What these needs are and what metrics and limits are suitable to measure these features to determine the QoS of the CE traffic is the subject in this section.

5.4.1 Delay and Jitter

CE traffic contains primarily voice data of telephone conversations (fax and other analog or digital data services is the other data). Both conversation partners expect to be able to converse in such a way that one person can respond to something said by the other and be heard almost immediately. Applications such as circuit emulation that are sensitive to delay, are called **real-time**

applications. **Non-real-time** applications, in contrast, can work without guarantees of timely delivery of data. They include most popular applications like Telnet, FTP, e-mail, Web browsing, and so on. The data transmitted by these applications are call ”traditional” or ”best-effort” data. The workload data traffic in this work belongs to that category and therefore does not claim for any QoS requirements. CE traffic, however, requires certain boundaries on the packet delay to ensure a ”good” service. Therefore, **delay** is a QoS metric for CE traffic.

Jitter is the variance of delay in a connection. The problem is that audio devices or connection-oriented systems (e.g. ISDN[1] or PSTN[2]) need a continuous stream of data [33]. In order to compensate for this, CE terminals and gateways implement a jitter buffer that collect the packets before relaying them onto their audio devices or connection-oriented lines, respectively. An increase in the jitter buffer size decreases the likelihood of data being missed but also has the drawback that it increases delay of a connection. If the delivery time of a packet exceeds the length of the receiver’s jitter buffer, then this packet ”arrives too late” with respect to its intended play-out time, and will be discarded. Consequently, the speech carried in this packet is lost for the decoding process. This ”packet loss” impairs speech transmission quality [32]. Therefore, jitter is also a QoS metric for CE traffic.

The total delay of the CE traffic in a network topology like in fig. 5.1 consists of **propagation**, **transmission** and **queuing delays**. In the next sections, these delays are further explained with a special focus on the simulations in this work. The total delay experienced by a frame from point to point is often also called **latency**.

Propagation Delay

Signal *propagation delays* over links (due to distance) are an important cause of communication delays. In OPNET Modeler, the propagation delay can be set ”Distance Based” (propagation delay will be determined based on the distance between the two nodes) or with a fixed time attribute. In the ”Distance Based” case, additionally the attribute *Propagation Speed* must be specified. The speed of propagation in coaxial cable is known to be equal to 0.65·c, in unshielded twisted pair to 0.59·c and in optical fibers to 0.68·c, where c is the speed of light in vacuum (299’792’458 m/s). The minimum propagation speed for worst case analysis is 0.585·c.

In this work, all propagation delays are based on fixed time values rather than on ”Distance Based”. This way, communication nodes can freely moved around on the project background without affecting any propagation delays. For the bottleneck link of the network simulation topology in fig. 5.1, the propagation delay is set to 50 μs which corresponds to a distance of 10 km for the optical link. The bottleneck ingress and egress link propagation delays are very short compared with the bottleneck link, so 10 m is the choice derived for these optical links with the fixed propagation delays set to 0.05 μs. All propagation delays together from the source nodes to the sink nodes are 50.1 μs for both types of traffic, CE frames from the CBR source and data traffic from the VBR source.

Transmission Delay

Transmission delay is the second form of delay incurred by each packet as it is sent over a link. Transmission delay is the amount of time between the start of processing of the first bit to the completion of processing of the last bit.

Because the simulated network in fig. 5.1 is collision free, there are no additional delays introduced due to resent frames after experienced collisions. The OPNET model makes use of the data rate attribute of the transmitting channel and the length of the transmitted packet. A CE traffic frame consists of 720 bits (see section 5.2) that are to be transmitted over the 1 Gbps link. The transmission delay at every processing node is therefore 0.72 μs. OPNET’s switch models are not supporting *cut-through*[3] technology, so any frame sent from source to sink node in the

[1] Integrated Services Digital Network

[2] Public Switched Telephone Network

[3] LAN switches can be characterized by the forwarding method that they support. In the *store-and-forward* switching method, the LAN switch copies the entire frame into its onboard buffers and computes the cyclic redundancy check (CRC). The frame is discarded if it is erroneous. If the frame does not contain any errors, the LAN switch looks up the destination address in its forwarding, or switching, table and determines the outgoing interface. It then forwards the frame toward its destination. With the *cut-through* switching method, the LAN switch copies only the destination address into its onboard buffers. It then looks up the destination address in its switching table, determines the outgoing interface, and

network is processed 3 times (*CBR_source*, *switch_1* and *switch_2*), introducing total 2.16 μs of transmission delay.

Queuing Delay

Queuing delays occur when data is made to wait for resources to become available, typically a communications link or a processor. This type of delay can in some cases account for the majority of a frame's end-to-end delay from its source to its destination node. Most queuing delay in OPNET occurs within the node domain, in either queue or transmitter modules. Transmitter modules incorporate a FIFO queue scheduling for frames to wait while frames that arrived before them complete transmission. Queue modules use a process model to determine when frames should be dequeued and forwarded. Therefore, the process domain is also involved in determining queuing delays.

Queuing delays are the only delays of the CE traffic in this work's simulated network that depend on the (variable) workload data traffic. When no data traffic is sent, the total end-to-end delay of a CE frame is at its minimum of 55.972 μs as shown in the Results chapter in section 6.3. Subtracting the propagation delay (50.1 μ) and the transmission delay (2.16 μs), the minimum queuing delay in the network results to be 3.712 μs, which accounts for the processing delays of all nodes of the given network. In the simulation setup of this work, CE and data traffic is sent independently over a 1 Gbps link to the bottleneck entry node *switch_1*. As long as the incoming traffic at the *switch_1* is less than the output link's capacity (which is also 1 Gbps), no relevant queues build up in this node and the delay of the CE traffic is constant at it's minimum value of approx. 56 μs.

If the VBR source sends continuously, the queue grows at the rate of the output link's exceeding capacity which is equal to the CBR traffic rate of 624'000 bytes/s. With growing queue size, also the delay increases. It takes 5.76 ms to send 624'000 bytes of CE traffic which also stands for the delay increase per second. If the maximum acceptable jitter in the CE traffic receiving node is 10 ms, according the MEF QoS requirements (see next section 5.4.3), then after 1.736 s the buffer size has increased that much that the introduced delay does exceed 10 ms. At this point, the buffer would be filled with 1.033 MB of data. Therefore, if the VBR source node sends for more than 1.736 s continuously, CE traffic frames start to arrive too late to be accommodated in the receiver node's jitter buffer and are discarded. To occupy the link for at least that time, a file at the VBR sending node needs to be at least 199 MB. This corresponds to the 0.99999895-th quantile of the ON/OFF Model's file size distribution or in other words, only 1 out of 0.96 million files is expected to be 199 MB or larger. With a file flow of 100 files/s for the reference parameters in table 6.2, this is supposed to happen every 2.7 hours on an average.

The findings in the previous paragraph can also be extended to a general Gigabit Ethernet topology: if the added queue length in all the buffers of all nodes that a CE traffic frame passes on its way from its sender to its destination node is more than 1.033 MB, then the introduced queuing delay is more than 10 ms. If some frames make it through the network without queuing delays, then the MEF QoS limits for jitter (10 ms) may prevent the CE traffic to fulfill the QoS requirements.

Hence, the simulations in this work, two buffer sizes are of interest: more than 1.033 MB and less than 1.033 MB. For reasons of convenience, 1 MB and 2 MB are taken if the buffer is limited by bits and 500 or 1'000 packets if it is limited by packets.

5.4.2 Losses

In networks, packets are sometimes lost or damaged due to buffer overflows or transmission errors. If the transported data is audio information, one lost sample or packet can be interpolated from the surrounding samples with relatively little effect on the perceived audio quality [13]. If more samples are lost, above all consecutive samples, the quality declines to the point that the audio becomes incomprehensible. A substantial part of the CE traffic is real-time voice data. Therefore, losses can negatively affect the QoS of CE traffic. Similar statements may be also true for other telephony services than audio.

---

forwards the frame toward its destination. A cut-through switch provides reduced latency because it begins to forward the frame as soon as it reads the destination address and determines the outgoing interface. (Taken from CISCO Systems' online *Internetworking Technology Handbook* on http://www.lsiinc.com/univercd/cc/td/doc/cisintwk/ito_doc/.)

One may oppose that there exist technologies (like the TCP transport protocol) to detect lost or damaged packets that are then retransmitted. These services are called *reliable*. As long as the total delay including the retransmission is less than the playback point, the QoS is actually not reduced. But generally, detection and retransmitting delays are too large so that for real-time applications these reliable services are generally not used. The network technology in this work is Gigabit Ethernet on the link layer. This layer does not support a reliable service.

In the simulation setup of this work, losses mainly originate from buffer overflows in the bottleneck entry node *switch_1* (fig. 5.1). Losses due to transmission errors are negligible small and buffer overflows in other nodes do not occur unless the simulation setup is badly configured. The point of buffering in the network is to absorb data bursts and to transmit them during the (hopefully) ensuing bursts of silence. This is essential to permit the transmission of bursty data [35]. OPNET Modeler's switch queues use **first-in-first-out (FIFO)** queue scheduling and **tail drop** queue management algorithms like most of the QoS unaware equipment today [34]. Tail drop is the traditional technique for managing router queue lengths. It sets a maximum length for each queue, accept packets for the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted. This technique is known as "tail drop", since the packet that arrived most recently (i.e., the one on the tail of the queue) is dropped when the queue is full. This method has served the Internet well for years, but it has two important drawbacks [35]:

1. **Lock-Out**: In some situations tail drop allows a single connection or a few flows to monopolize queue space, preventing other connections from getting room in the queue. This "lock-out" phenomenon is often the result of synchronization or other timing effects.

2. **Full Queues**: The tail drop discipline allows queues to maintain a full (or, almost full) status for long periods of time, since tail drop signals congestion (via a packet drop) only when the queue has become full. It is important to reduce the steady-state queue size, and this is perhaps queue management's most important goal.

OPNET Modeler allows to set limits on the bit and packet capacity for every queue, but this feature is not documented in OPNET Modeler's manuals and by default, these limits are set to infinity. The technical support of OPNET provided the following procedure to set limits on the queue sizes:

1. Go to the process model `ethernet_mac_v2.pr.m`

2. Go to the Interfaces Menu → Process Interfaces

3. Go to the *Subqueue* attribute and click on the initial value of this attribute

4. If you are not using etherchannel, then ethernet only uses the first subqueue (first row) to queue the packets coming from the higher layers. You can set a finite value for the bit capacity or packet capacity or both. This will make the queue size for all ethernet nodes finite.

Now, the question arises if the queues should be limited by packets or bits. RFC 2309 [35] states that Internet routers set a maximum queue length in terms of packets. However, it is not clear how queue lengths in real Ethernet switches are generally limited. Leaving this subject open, simulations with queue sizes limited by bits and simulations with queue sizes limited by packets are done. Bit capacity limitation is what can be called "best case scenario" for the CE traffic. If the queues are limited by bits, a consequence of tail-dropping is when the queue is almost full, small frames have a bigger chance to be accommodated than large frames (see fig. 5.10) This phenomenon does implicitly give higher priority to small frames from not being dropped and belongs to the lock-out drawbacks of tail-dropping.

In the standard simulation setup in this work, every CE traffic frame needs 78 bytes (without VLAN) of buffer space to be stored in a queue. The data traffic consists primarily of maximum size frames of 1'526 bytes. These frames are almost 20 times larger than the CE traffic frames (see fig. 5.10). Therefore, it is justified to assume that in case of a bit limitation of the buffer, data traffic frames are dropped more frequently than CE frames. Hence, limiting bits or bytes rather than packets is in favor of CE traffic's QoS.

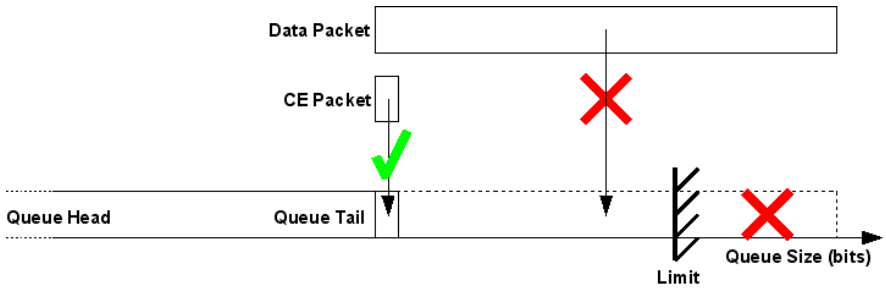

Figure 5.10: Small CE frames have higher chances to be accommodated in saturated buffers with bit capacity limitation than large data frames.

5.4.3 QoS Requirements for CE Traffic

In the sections before, QoS metrics have been identified and discussed with a focus on this work's simulations. This section is about limits on these metrics to qualify if the QoS is acceptable or not. This section starts with an general overview about delay requirements of voice traffic and then introduces a proposal based on the METRO Ethernet Forum (MEF) requirements for CE services over Metro Ethernet Networks (MEN).

Overview

ITU-T's recommendation G.114 [32] provides end-to-end limits for mean one-way delay of voice traffic, independent of other transmission impairments (see table 5.2).

One Way Delay [ms]	Description
0–150	Acceptable for most user applications.
150–400	Acceptable provided that administrations are aware of the transmission time impact on the transmission quality of user applications.
>400	Unacceptable for general network planning purposes; however, it is recognized that in some exceptional cases this limit will be exceeded.

Table 5.2: ITU-T's G.114 [32] recommendation on one-way voice delay.

Cisco Systems [29] classifies a cumulative transmission delay of up to 160 ms as "high quality" as shown in fig. 5.11. Both guidelines refer to the delay from one far end to the other far end, including all delays introduced on the entire path. For voice data, this end-to-end transmission delay is also called "lip-to-ear" or "mouth-to-ear" delay.

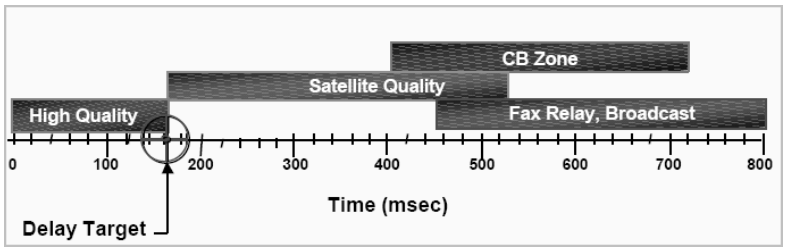

Figure 5.11: Cisco Systems's [29] recommendation on cumulative transmission path delay.

The simulated metropolitan network in this work contributes just a portion to the total end-to-end delay of the CE traffic. Thus the delay introduced by the network must be smaller than the total acceptable end-to-end delay. An approach for QoS requirements for Voice over IP (VoIP) on IP intra-nets provides Fiedler et al. [3]. They defined three requirements from the practice: one for the end-to-end delay and the other two for packet loss. 50 ms is defined as an upper

limit for end-to-end network delay. Successive losses of packets are grouped into *outages*. They are usually perceived as a crackling sound at the receiver's side. The requirements on outages is defined as follows: the number of outages must not exceed 5 per minute and the durations of an outage must not exceed 50 ms.

Metro Ethernet Networks (MEN)

The approach of Fiedler et al. [3] is not applicable to the CE traffic transported over a metropolitan packet-switched network as in this work because VoIP is a different service than circuit emulation. VoIP typically originates and terminates at the network sending resp. receiving node. Circuit emulation services, in contrast, use the network only as intermediate transportation path from the telephone service origin to its destination. For the metropolitan network portion, the upper delay limit can therefore just be a fraction of the cumulative, end-to-end delay limit. The same holds for losses. The **METRO Ethernet Forum (MEF)** outlines requirements of CE services over **Metro Ethernet Networks (MEN)** in its Technical Specification MEF 3 [11]. For US standards, it generally adapts the ANSI T1.510-1999 requirements and for Europe the ITU-T G.826 (02/99) [31] recommendation to address the specific need of MEN.

The main difference between the MEF 3 specification [11] and the ITU-T G.826 [31] recommendation is that the later document distinguishes between errored frames and lost (or defect) frames. An errored frames contains one or more bit errors. On a E1 carrier, for example, one bit error affects only one of the 32 channels. If the traffic is voice data, and depending if the errored bit is in the higher or lower order of the 8 bit voice samples, the effect on the voice quality may be small. In contrast, the Ethernet detects a bit error with very high probability due to the 32 bit CRC **Error Detecting Code (EDC)** and discards the entire frame. This leads to a complete information loss for the duration of one frame length for all channels and consequently to a burst of bit errors in the reconstructed E1 carrier. Therefore, a bit error in the CE traffic, as modeled in this work and in the MEF specification [11], has a distinct influence on the QoS performance than in the constant bit rate digital paths used in the ITU-T G.826 recommendation [31]. Exactly at this point is where the MEF paper [11] goes beyond the ITU-T or ANSI recommendations: it does the conversion from bit errors to frame losses, resp. **Frame Error Ratio (FER)**. The FER is defined as the total number of lost frames over the total number of frames sent [11]. The following effects lead to the loss of or discarding of a frame: frame loss, excessive frame delay or jitter and any bit error in the data stream (which will appear as lost frames).

The MEF uses the equation FER = ES/(CE frame rate), where **ES** is the **Errored Second** which is defined in [31] as a one-second period with one or more errored frames or at least one defect. This conversation has the following two consequences: first, when ITU-T G.826 [31] "accepts" a frame error, MEF [11] "accepts" a entire frame loss. This is certainly a weakening of ITU-T's requirements. Second, maybe to compensate that weakening, MEF converts limits from the one-second period metric ES to a FER, which is defined over the entire measurement period. This is a implicit assumption that not more than one bit error occurs within any one-second period. Therefore, frame error bursts are higher weighted in the MEF metric.

The MEF also converts **Severely Errored Second (SES)**[4] to FER. Since the resulting FER limits are less strict for small "packing densities" (the CE traffic in this work has the smallest possible "packing density" of 1) than the FER limit from the ES conversion, only the last limit is relevant in this work. This FER limit (from the ES conversion) is derived by the MEF as follows: they use the ES Ratio (ESR) limit for a international digital Hypothetical Reference Path (HRP) of 27'500 km from table 1 in ITU-T G.826 [31], which is 0.04, take 17.5 % of it according [31] to get the national portion of it, which is then 0.007. If only one E1 frame is put into one Ethernet frame ("packet density" is 1) as in this work's simulations, the FER is 0.007 divided by the CE frame rate of 8'000 which results to be $8.75 \cdot 10^{-7}$.

The MEF [11] also provides limits on frame delay and jitter. Frame exceeding these limits will be treated as lost and contribute to the FER. In order to avoid the need for voice echo cancellation, the frame delay across the MEN for an emulated circuit SHOULD[5] be less than 25 ms [11]. Additionally, the CES between the two CES end points SHOULD be capable of

[4] A one-second period which contains $\geq$ 30 % errored frames or at least one defect. SES is a subset of ES [31].

[5] This key word has to be interpreted according RFC 2119: "SHOULD" means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

functioning correctly with a frame jitter of up to 10 ms [11].

For the network availability metrics, MEF uses the worst case of the ITU-T G.826 [31] definition: the ”unavailable state” is entered at the start of a period of 10 consecutive seconds with at least one lost CE frame per second. The ”available state” is resumed at the start of a period of 10 consecutive seconds without any CE frame losses [11]. The availability ratio SHALL[6] be 99.95 % or better [11].

The MEF metrics and limits for a ”packet density” of 1 and 8’000 packets per second can be summarized as follows:

- The Frame Error Ratio (FER) must not exceed $8.75 \cdot 10^{-7}$.
- The Frame Error Ratio (FER) is the total amount of lost CE frames over the total amount of sent CE frames within the measurement period.
- CE frames delayed more than 25 ms are treated as lost frames.
- CE frames exceeding the jitter limit of 10 ms are treated as lost frames.
- CE frames dropped by any network node are treated as lost frames.
- The availability ratio must not exceed 99.95 %.
- The ”unavailable state” is entered at the start of a period of 10 consecutive seconds with at least one lost CE frame per second. The ”available state” is resumed at the start of a period of 10 consecutive seconds without any CE frame losses.

As long as the minimum delay is less than 15 ms (as in this work’s simulations, see section 5.4.1), then the jitter limit of 10 ms ”dominates” over the delay limit of 25 ms. For simulation durations and network utilization rates considered in this work, there will always be CE frames passing the network at its minimum (end-to-end) delay of 55.972 μs. Therefore, CE frames delayed more than 10.055972 ms are treated as lost frames in this work and contribute to the FER. Availability consideration are beyond the scope of this work. So, the revised MEF metrics and limits for the simulations in this work can be summarized as follows:

- The Frame Error Ratio (FER) must not exceed $8.75 \cdot 10^{-7}$.
- The Frame Error Ratio (FER) is the total amount of lost CE frames over the total amount of sent CE frames within the measurement period.
- CE frames delayed more than 10.055972 ms are treated as lost frames.
- CE frames dropped by any network device are treated as lost frames.

5.5 Simulation Data Analysis

5.5.1 Data Traffic Analysis

The MATLAB code `trace_analysis.m` in appendix D.1 reads in a data file containing traffic flow data (called ”traffic trace”) captured by the simulation software OPNET Modeler. The data must be recorded in the ”bucket mode”, i.e. stochastic information for small time intervals. The code then generates a Variance-Time (V-T) Plot (see section 4.6.3 for theoretical background) for the traffic trace and also draws the time series in two multi-plots: the aggregated traffic traces for different aggregation levels and the traffic trace at different time scales (time fraction traces). Least square line fit, featuring out layer rejection and interval selection, generates a line through the data points in the V-T Plot. The Hurst parameter H is then derived from the line’s slope with eq. 4.35.

For practical purposes, the aggregation levels are always at power of two as in fig. 4.14. This way, the points in the Variance-Time Plot are equally spaced and calculation time and memory usage is reduced. The variance is normalized to make it independent from the unit quantity.

[6] This key word has to be interpreted according RFC 2119: ”SHALL” means that the definition is an absolute requirement of the specification.

The two multi-plots with the aggregated time series and the traffic traces at different time scales are line plots, i.e. all data points are interconnected by lines. Bar plots, i.e. every data point is visualized by a bar, do required too much memory space in MATLAB but would look somewhat ”nicer”.

An example of the V-T Plot is shown in fig. 5.12. The analyzed traffic trace contains one million data points captured in 0.01 second time intervals (”smallest time unit”) as can be seen at the x-label. Data points which lie too far away from the linear approximation (out layers) are excluded from the least-square line fit and are drawn with a ”x”, whereas the other points used for the least-square line fit are depicted with a ”+”. The estimated Hurst parameter H, calculated from the fitted line's slope β, is shown in the plot title. X- and y-scales are equal to simplify comparison between different V-T Plots. The normalized variance from the unaggregated trace can be read from the first data point at log10(m) = 0 which is in the example of fig. 5.12 approx. $10^{1.2} = 16$.

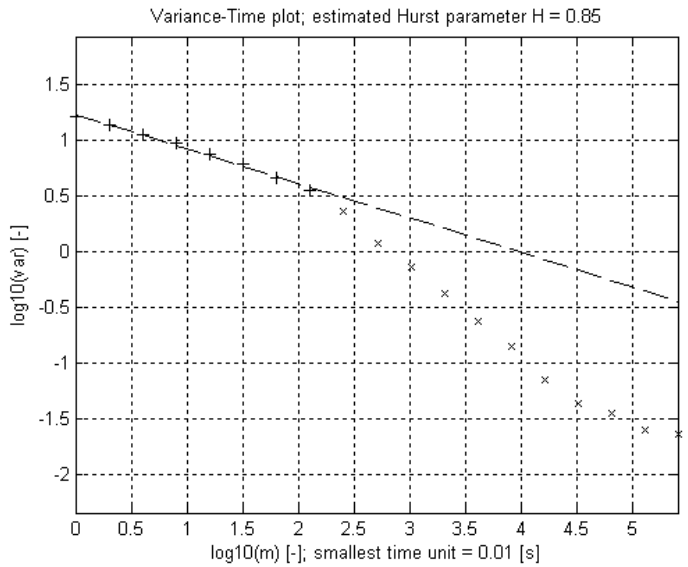

Figure 5.12: Variance-Time Plot generated with MATLAB code `trace_analysis.m` for example traffic trace.

Fig. 5.13 shows an example of a traffic trace at different aggregation levels. The x-labels show the aggregation level m and the corresponding time units. As mentioned earlier, the data points are interconnected by lines rather than drawn in the bar-style.

The last plot generated with the MATLAB code `trace_analysis.m` is the multi-plot at different time scales in fig. 5.14. The first plot illustrates the original traffic trace from the file and is identical with the first plot from fig. 5.13. From this plot's x-scale can be seen that the data file contains one million points and the interval time is 0.01 seconds, which gives a length of 10'000 seconds of the traffic trace. The plots below show one tenth of the time from the plot above. All plots start at the same point. So, if the first plot shows 10'000 seconds of a traffic trace, the next plot below shows only the first 1'000 seconds, the third plot 100 seconds and the last plot 10 seconds from the original traffic trace. This magnifier behavior is emphasized in fig. 5.14 by additional lines. These time fraction plots give a good visual impression about the traffic shape.

5.5.2 QoS Analysis

To measure the QoS performance of the CE traffic according the in section 5.4 elaborated metrics and requirements, the MATLAB code `qos_analysis.m` is written. This code in appendix D.3 reads in two data files and requires a couple of parameters as inputs.

The first input file, named 'rb_output_file_delay.txt', contains the delays of each CE frame received at the *CBR_sink* node exceeding a adjustable threshold. The threshold attribute allows to reduce the amount of captured data. During a frequently used simulation duration of 10'000 s, 80 million CE traffic frames pass the network. Recording the delay of each frame

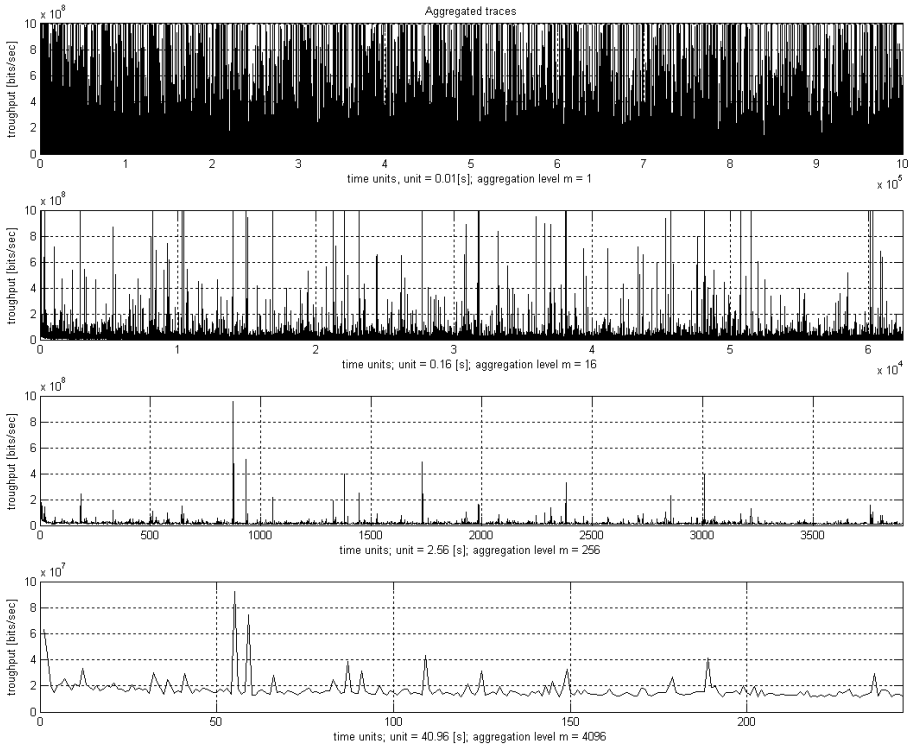

Figure 5.13: Multi-plot with different aggregation levels generated with MATLAB code `trace_analysis.m` for example traffic trace.

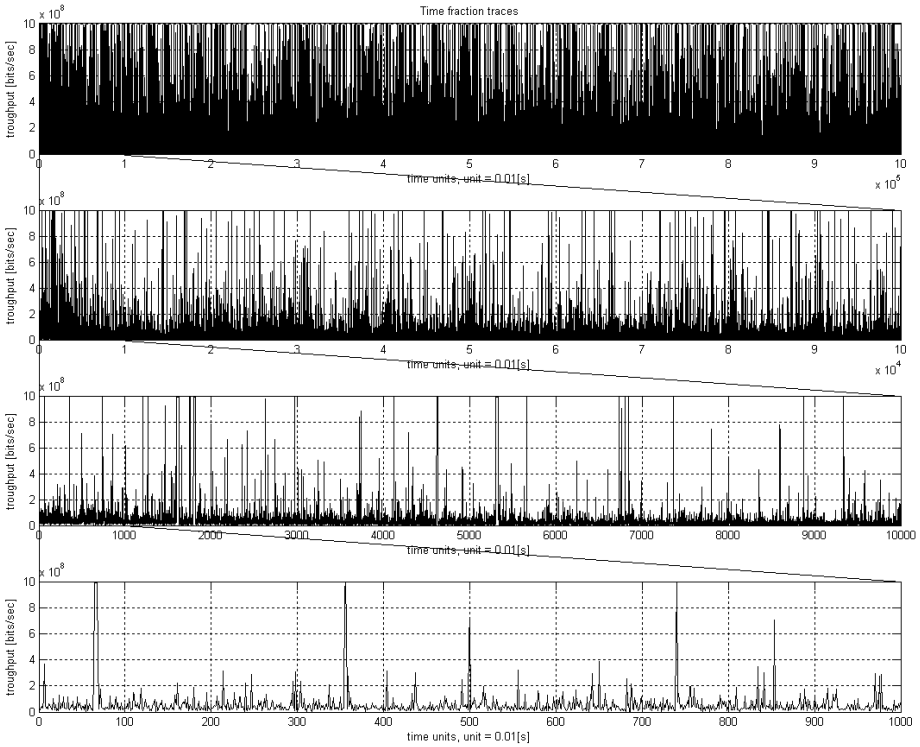

Figure 5.14: Multi-plot with different time scales generated with MATLAB code `trace_analysis.m` for example traffic trace.

would produce a file of approx. 2.7 GB. By choosing a appropriate threshold, this data amount can be reduced by magnitudes. For instance, the 1 ms threshold leads for the same simulation duration to file sizes of approx. 6 MB. Given that the CE frame frequency and total simulation duration is provided as input parameters, this selective data reduction does not impair the applied QoS metrics and can therefore be justified. The threshold feature is not part of OPNET Modeler's standard distribution. Rainer Baumann, this thesis' co-supervisor, modified the OPNET source code to add the threshold attribute for frame delays. This feature can be enabled by setting the new *traf_gen.rb_stat_delay* attribute of the *ethernet_ip_station* node to "1". The other new attribute *traf_gen.rb_delay_threshold* specifies the threshold value. When enabled, the resulting data is automatically written into OPNET Modeler's binary directory 'C:\Program Files\OPNET\10.5.A\sys\pc_intel_win32\bin'.

The second input file, named 'rb_output_file_drop.txt', contains queue drop information of specified network adapters. In particular, it lists the source and destination information for all dropped frames. This makes it possible to associate dropped packets with a traffic flow. This feature has also been added by Rainer Baumann because OPNET Modeler does not support it. Without this functionality, it would not be possible to distinguish dropped data frames from dropped CE frames. The new feature can be enabled for any interface *X* by setting the new *mac_X.rb_drop_mode* node attribute to "1". When enabled, the resulting data is automatically written into OPNET Modeler's binary directory 'C:\Program Files\OPNET\10.5.A\sys\pc_intel_win32\bin'. To associate now the dropped frames with a traffic flow, MATLAB code `qos_analysis.m` just requires the node IDs of the traffic's corresponding source and destination nodes.

5.6 Service Priority in OPNET Modeler

Only when having started working with OPNET Modeler, we realized that the software does not support VLAN with service priorities (standard IEEE 802.1Q/p [16] [17]). The technical support of the software vendor did confirm that this feature is missing in the current version 10.5A of OPNET Modeler. Therefore, one of the mayor objective of this thesis, if circuit emulation requires service priority, can not be answered in this work.

5.7 Hurst Parameter

The Hurst parameter H is introduced in the Background chapter in section 4.6.3. To select appropriate values of H for the data traffic in the simulations, estimates originated from measurements of real LAN traffic are considered. Many people have done traffic analysis and estimated Hurst parameters so far. They conclude that real, live data traffic on LANs exhibit Hurst parameter in the range of 0.70 to 0.98. The early measurements in the years from 1989 to 1992 in Leland et al. [7] are a important guideline. Their estimates of the Hurst parameter of Ethernet LAN traffic are around 0.9.

For this study, measurements of Oliver Lamparter [20] are taken as reference. The data was recorded on September 2002 at the connection to the ISPSS[7] network of the service provider Swisscom. The traffic of more than 5.5 hours originates from 255 LAN-interconnections, allowing companies to interconnect their local networks over the backbone of Swisscom. The estimated values of the Hurst parameter are for a number of estimating methods in the a range of 0.90 to 0.95.

Thus for the synthetic workload generation in this work a Hurst parameter of 0.90 is taken as the target value of the resulting data traffic. Applied to the ON/OFF Model in section 5.3.4, the shape parameter α of the underlying Pareto distribution is 1.2, obtained from eq. 4.36.

5.8 Network Utilization Rate

The burstiness over different time scales, expressed with the Hurst parameter, is one important property of the synthetic data workload. The **average network utilization rate** is another one.

[7] Internet Protocol Standard Service. Internet protocol platform of Swisscom.

Like the Hurst parameter, it should be a realistic value, i.e. the average network utilization rate for the simulation of the synthetic workload should be chosen similar to rates in comparable, real networks. Estimates of Odlyzko [15] are taken in this work. Odlyzko investigated utilization rates of private and corporate data networks and their implications. His estimates for the average utilization rates of such networks are consistent with recent measurements and are as low as 1 % [15]. He provides evident reasons for this surprisingly low value.

The main cause of this low utilization rate is to satisfy user needs for low latency rather than high link efficiency [15]. Average throughput is basically irrelevant. It is only the peak rate that matters. This phenomenon became more emphasized as link capacities grow enormously over the past few years whereas the hardware drastically decreased in price. If prices come down, users will optimize for higher quality transmission and thereby lower utilizations. Lightly utilized resources became inexpensive enough compared to the value people place on its availability and their time. Low utilization rates of corporate networks are not a sign of waste, but of the value of high quality data communications and of the complexity of running networks. Odlyzko [15] hereby disproves the common misconception that that the ”bursty nature of data traffic” is the culprit behind low utilization rates of data networks.

5.9 Simulation Time

As discussed in section 4.5, statistical properties of heavy-tailed distributed random variables converge very slowly. Thus simulations with heavy-tailed random variables as input parameters also converge slowly and show high variability in steady state [25]. This difficulties, inherent in simulations with heavy-tailed inputs, are likely to be particularly great when tail index α is less than about 1.7 [25]. Crovella et al. [27] and Fiedler et al. [4] report that the average object size and the distribution of output statistic that depend on all moments of the heavy-tailed object size distribution does not converge during practically feasible simulation periods. Therefore, the simulation remains in transient state for all periods. This statement also holds for finite heavy-tails as long as the tail is ”sufficient long” [4].

The synthetic data traffic of the ON/OFF Model in this work (see section 5.3.4) is based on a heavy-tailed object size distribution with tail index $\alpha = 1.2$ and with limited tail which can be considered as ”sufficient long”. Therefore, convergence within practically feasible simulation periods of the output statistics can not be expected. Distributions and confidence intervals of these statistics from a number of identical simulation periods with different seeds enable to draw a better picture of the results than any resulting statistical value of one single simulation period. Of course, the simulation times should never the less be that long to reach steady state as close as possible to keep the confidence intervals as small as possible. This objective can not only reached by increasing the simulation time but also by choosing appropriate statistics. As shown in section 4.5, using quantiles instead of arithmetic mean values significantly reduce the interval of confidence. Quantiles do not depend on the extreme tail of the distributions and are further better suited to describe numerous properties, such as latency for QoS guarantees of web services [4].

The simulation duration with the ON/OFF Model as input parameters can be estimated. To reach steady state of the simulation, the average object sizes of the object size distributions of the simulation inputs have to converge [2]. Table 5.3, taken from page 91 of Fiedler [2], lists required sample size to reach 5 % accuracy for different tail indices without and with a object size limit of 2.1 GB.

Tail Index α	**Average (w/o limit)**	**Average (2.1 GB limit)**
1.1	$2.0 \cdot 10^{14}$	$1.4 \cdot 10^{12}$
1.2	$6.4 \cdot 10^{7}$	$6.1 \cdot 10^{6}$
1.3	$4.3 \cdot 10^{5}$	$2.9 \cdot 10^{6}$

Table 5.3: Sample sizes at various tail indices α to achieve 5% accuracy. (Taken from Fiedler [2])

With the results from table 5.3, it is possible to estimate the required simulation duration with the ON/OFF Model from section 5.3.4 as simulation input. The ON/OFF Model in this work is built from sources sending Pareto-distributed object sizes (or files) with shape parameter $\alpha = 1.2$

(section 5.7) and 4 GB limit. With the average network utilization rate of 1 % according section 5.8 and a average file size of 12 kB, about 100 objects are sent per minute on an average (average file flow). From table 5.3 can now be read that the required object sample size to reach 5 % accuracy is somewhere between $6.1 \cdot 10^6$ (2.1 GB limit) and $6.4 \cdot 10^7$ (no limit). These values divided by the average file flow give a required simulation duration somewhere between 17 and 178 hours to reach 5 % accuracy in the simulations of this work. Because the 4 GB limit is not magnitudes away from the 2.1 GB limit in table 5.3, the value is expected to be closer to 17 than 178 hours. Thus 28 hours are taken in the next paragraph for further calculations.

With the given hard- and software in this diploma thesis, simulation periods of 10'000 s (2.8 hours) are feasible. The statistical values of interest from one simulation period (say T) allow to be retrieved from multiple (say n) simulation periods T as if it was one simulation with a extended period of $n \cdot T$. Therefore, approximately 10 simulation periods of 10'000 s, corresponding to one period of 28 hours, are necessary in the network simulations to reach 5 % accuracy. The actual number of simulation runs of 10'000 s is 16 for all tested configurations in this work, giving cumulated 44 simulated hours. Therefore, it is justified to assume that accuracy of the simulation results is 5 % or better. Dividing the total simulation period in a number of small periods has a also a advantage: a set of statistical values originated from all individual small simulation periods reveals addition information by its distribution and confidence interval.

Chapter 6

Simulation Results

In the previous Design chapter 5, the simulation setup is explained and tools to generate self-similar data traffic and to analyze simulation results are introduced. In this chapter, results from simulations are presented and discussed.

6.1 Data Traffic Generation with the Raw Packet Generator (RPG)

OPNET's RPG package is introduced in the Simulation Design chapter in section 5.3.3 and its parameters for the simulation defined. In the next sections, the influence of each individual, relevant parameter in table 6.1 of the RPG is analyzed in detail. For all simulations, the parameters in table 6.1, explained in the Simulation Design chapter in section 5.3.3, are taken as reference. No more than one parameter is changed for each individual simulation run. Only the variation of that parameter is separately mentioned. All other parameters are constant according table 6.1. All simulation lengths are 1'000 seconds. With a "bucket size" of 0.001 seconds, 1 million trace logs are generated and analyzed. The Variance-Time Plot in fig. 6.1, the aggregated traces in fig. 6.2 and the time fraction traces in fig. 6.3 are produced with the reference parameters from table 6.1 and serve as reference plots. The use of only one seed can be justified because it is taken for all simulations. Running all simulations with a identical seed does exclude dependencies of the random generator and exhibits better the influence of the input parameters.

Parameter Name	Abbreviation	Value
Hurst parameter	H	0.90
Fractal Onset Time Scale	FOTS	0.0008 s
Average Arrival Rate	AAR	1190
Peak-to-Mean Ratio	P2MR	100
Source Activity Ratio	SAR	75
Simulation length	-	1'000 s
Smallest sample interval	"bucket size"	0.001 s
Seed	-	127

Table 6.1: Reference parameters for the Raw Packet Generator of OPNET Modeler.

6.1.1 Peak-to-Mean Ratio (P2MR)

The first RPG parameter analyzed is the Peak-to-Mean Ratio (P2MR). The P2MR should control the absolute value of the variance. If the peaks are higher while keeping the mean constant, the variance increases and vice versa. But the P2MR parameter has not the effect on the traffic generation as one would expect. Although the variance increases for P2MR values of 50, 100 to 200, it decreases again at the value of 1'000. The (normalized) variance of the unaggregated traffic traces (log10(m) = 0) can be read from the V-T Plots in appendix F.1.1 and fig. 6.1. It is 0.5 for P2MR = 50, 0.7 for P2MR = 100, 1.8 for P2MR = 200 and 1.4 for P2MR = 1'000. Also

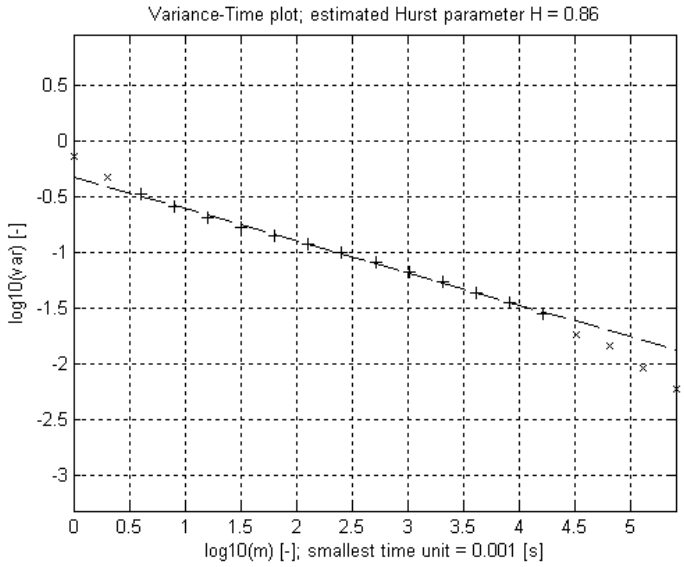

Figure 6.1: Variance-Time Plot of traffic generated with the Raw Packet Generator (RPG) and reference parameters.

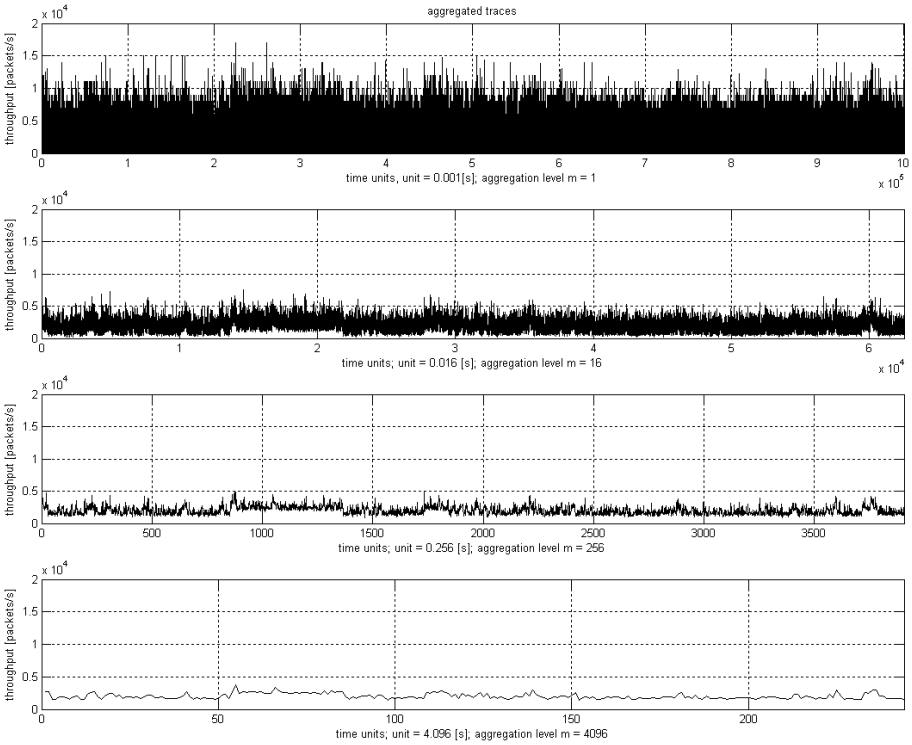

Figure 6.2: Aggregated traffic traces of traffic generated with the Raw Packet Generator (RPG) and reference parameters.

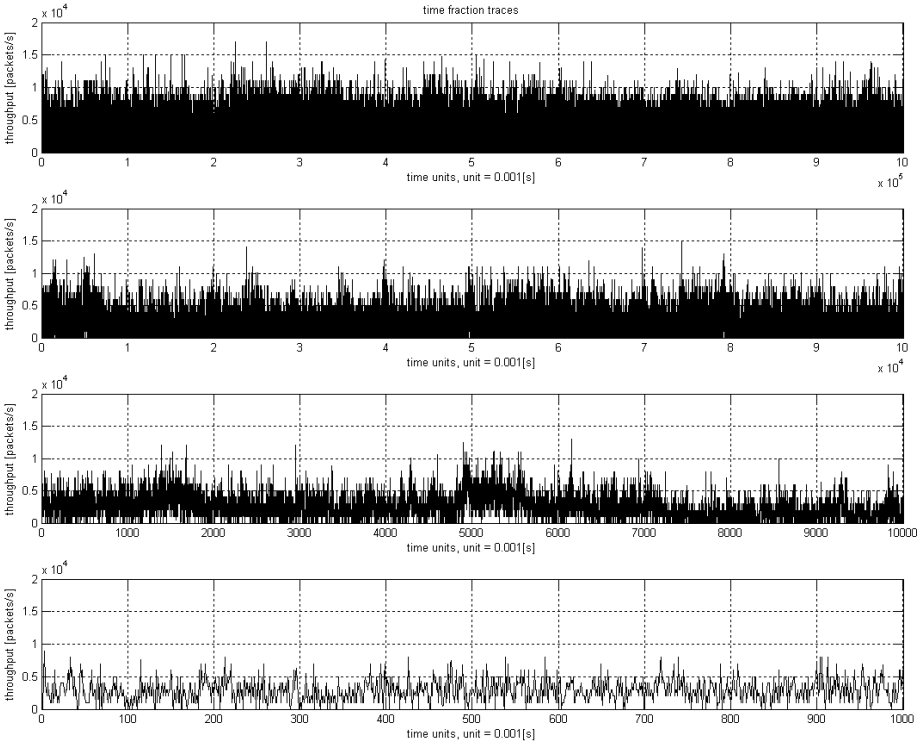

Figure 6.3: Time fraction traces of traffic generated with the Raw Packet Generator (RPG) and reference parameters.

interesting is that, although the variance slightly increases for P2MR = 50 to 200, the peaks in the traffic for P2MR = 50 are higher than for P2MR = 200. A P2MR of 10'000 is also tested without any results because the RPG refuses to generate any packets. For even higher values, packets are generated again.

6.1.2 Hurst Parameter (H)

Simulations are run with two different Hurst parameters H: 0.02 higher and 0.02 lower than the reference value of 0.90. According to the definition of the Hurst parameter (see section 4.6.3), one would expect the slope in the V-T Plot for $H = 0.88$ slightly more and for $H = 0.92$ less falling than the reference slope in fig. 6.1. As shown in the V-T Plots in appendix F.1.2, the Hurst parameters remain almost unchanged, but the higher the Hurst parameter, the earlier the curve starts falling from the fitted straight line. This means that the traffic's self-similarity is restricted to smaller time scales when the Hurst parameter of the RPG is increased.

6.1.3 Source Activity Ratio (SAR)

The Source Activity Ratio (SAR) is increased and decreased by absolute 5 % from the reference value of 75 %. The influence of the SAR to the RPG output in theory is difficult to understand. The major impact one would maybe expect is on the absolute values of the (normalized) variance: a smaller SAR may cause the source to send more traffic in less time and therefore increase the variance. But simulation results show a clear trend: decreasing the SAR increases the Hurst parameter H and vice versa (see V-T Plots in appendix F.1.3). For SAR = 70 %, the estimate of H is 0.94, for the reference (SAR = 75 %) 0.86 and for SAR = 80 % it is 0.83. The change of the absolute variance can be neglected.

6.1.4 Fractal Onset Time Scale (FOTS)

The Fractal Onset Time Scale (FOTS) should influence the smallest time scale when the traffic becomes self-similar. FOTS = 1 and FOTS = 0.1 do not work (simulation remains at packet generation start with 100 % CPU workload and does not progress). In the V-T Plot, the start of self-similar traffic shape is visible where the curve's slope starts to increases after falling at the beginning. In appendix F.1.4 there is a very good example for FOTS = 0.0001 s: the curve falls from log10(m) = 0 (0.001 s) to log10(m) = 1 (0.01 s) of the x-axis at a slope of approx. -1, indication for a Hurst parameter H of 0.5 and not self-similar traffic (poisson distributed). Then, at log10(m) = 1 (0.01 s), the curve becomes almost horizontal, indicating highly self-similar traffic with a Hurst parameter close to 1. This example also shows that the FOTS has not the influence on the RPG as expected: the FOTS is decreased and traffic becomes self-similar at higher time scales. On the other hand, when FOTS is increased to 0.002 seconds, the traffic shows self-similarity right from the smallest time interval of 0.001 s.

6.1.5 Conclusion

In theory, OPNET's RPG package seems to be a attractive option to easily generate anticipated self-similar data traffic. There are just a few input parameters to describe the main attributes of a self-similar traffic pattern. However, simulation results reveal that the RPG model does not perform as written in the RPG user guide. Changing the RPG input parameters will not always have the expected effect on the output of the model. Certain input parameter combinations even let the RPG model "crash": the simulation runs very quickly because the RPG produces no traffic at all or the simulation holds at the beginning of RPG traffic generation with 100 % CPU load. There is a complex relation between the input parameters and the generated traffic. Sometimes, the only way to get the desired output is a trial-and-error method. These effects make it necessary to check the RPG output to make sure the traffic fits the desired characteristics. Other OPNET users are reporting similar problems, like the Katholieke Universiteit Leuven in Belgium [21], which participates in *OPNET's University Program*.

Another problem of the RPG for the network simulations in this thesis is that the (normalized) variance is small for short time scales. For the aggregation level 0 (= 0.001 s), for instance, the

normalized variance is 0.7 in normal scale. This is because the generated traffic does only occupy the link for very short periods. As can be seen from the traffic trace in fig. 6.3, there is no time interval of 0.001 s (= ”smallest time unit”) where the RPG source sends its traffic at full speed. Since the size of all packets is constant 1 kB, this yields to a maximum packet flow of 117’702 packets/second. But non of the peaks in fig. 6.3 are higher than 20’000. These traffic patterns are not what can be observed in real networks. If any node sends a object over the network, it sends it at maximum link speed. If the object is large, it occupies the link completely up to several seconds.

For all these reasons, the RPG approach to generate the self-similar data traffic in the simulations of this work is not further followed. The scripted file playback with the ON/OFF Model is better suited as will be shown with simulation results in the next section.

6.2 Data Traffic Generation with scripted ON/OFF Model

After OPNET’s Raw Packet Generator (RPG), the scripted file playback (see section 5.3.4) with the ON/OFF Model of section 5.3.4 is taken as self-similar data traffic source for network simulations in OPNET Modeler.

The ON/OFF Model performs very well in the simulations. Because the traffic generation is done by a self-written MATLAB code, the model is completely transparent in contrast to OPNET’s RPG. The ON/OFF Model consists of sufficient parameters to tune the traffic source to a variety of desired behaviors. The main advantage of this model over the RPG is that the variance for small time scales of the generated traffic is much higher. This is because the source sends its data in less time but at higher speed than the RPG. The weakness of this model is the decaying variance at higher aggregation time scales.

If not mentioned separately, the parameters in table 6.2 are taken as reference for the simulations. The choice of these parameter is explained in the Simulation Design chapter, section 5.3.4.

Parameter Name	Abbreviation	Value
Shape parameter of file size distribution	α_S	1.2
Shape parameter of waiting time distribution	α_T	1.2
Average file size	S_{avrg}	12 kB
Maximum file size	S_{max}	4 GB
Number of sources	n	100’000
Average link utilization	r_{avrg}	1%
Simulation length	-	10’000 sec
Smallest sample interval	”bucket size”	0.01 sec
Seed	-	0

Table 6.2: Reference parameters for the ON/OFF Model.

The file size sample in fig. 6.4 shows the expected characteristic of a heavy-tailed distribution: most of the files are small and a few are very large. Because these large files contribute considerably to the sample’s arithmetic mean, most of the approx. 1.7 million files are less than the Pareto distribution’s expected mean of 12 kB. The location parameter k, and therefore the smallest file size, is 2’167 bytes. The largest file in fig. 6.4 is 369 MB - much smaller than the upper limit of 4 GB. Nevertheless, it is very close to the largest expected file of 319 MB for this sample size obtained with the method described in section 4.4.3.

The running arithmetic mean of the same file size population as in fig. 6.4 is shown in fig. 6.5. The arithmetic mean over the entire sample population is 12.3 kB, slightly higher than the 12 kB of the expected value. The high variation of the running arithmetic mean for small numbers of files is typical for heavy-tailed distributions. The upwards step between the 400’000th and the 500’000th file in the running mean reflects the influence of the two largest files of the sample population. Because the Pareto distribution is limited at 4 GB, the arithmetic mean does converge faster to the expected value than the unlimited distribution would do. This is mainly due to the increased location parameter k. It is 2’167 bytes for the *limited* distribution and would be 2’048 bytes for the *unlimited* distribution.

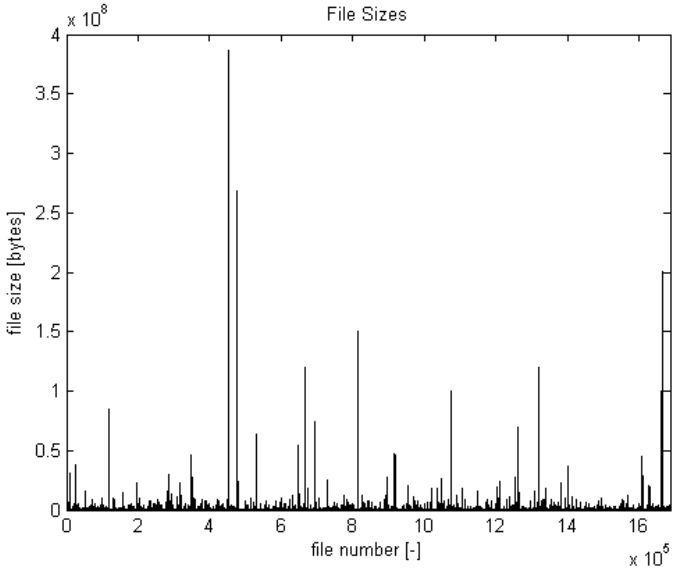

Figure 6.4: File size distribution with ON/OFF Model and reference parameters.

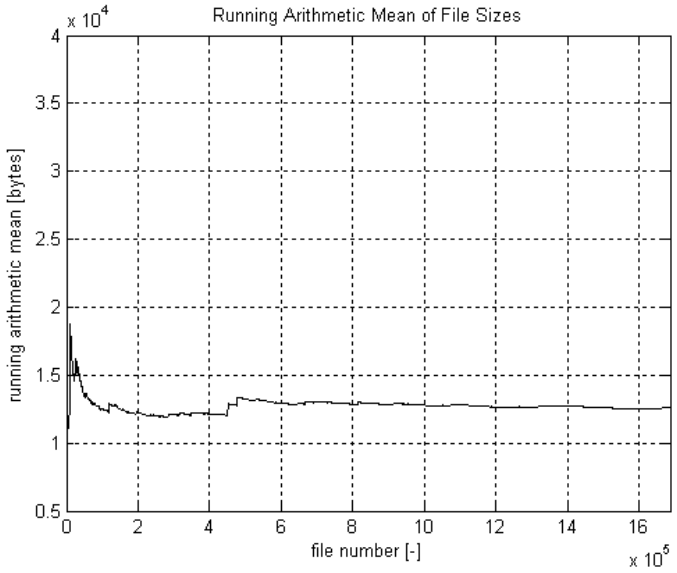

Figure 6.5: Running arithmetic mean of file size distribution with ON/OFF Model and reference parameters.

The average waiting time (OFF period) between two consecutive file transfers results from the link capacity and the average utilization rate. It is 1’025 s for one of the total 100’000 sources or 10.1 ms if only one source would generate the same amount of traffic. The OFF periods are shown in fig. 6.6. The maximum value of all periods is 3.2 s.

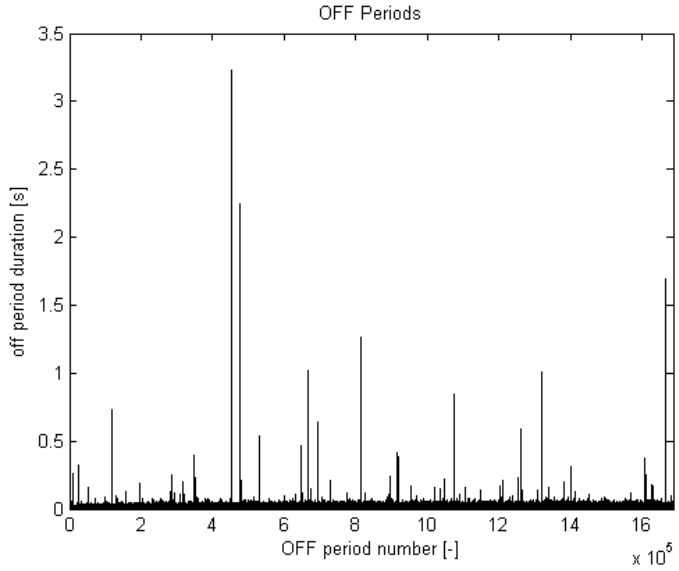

Figure 6.6: Off-period distribution with ON/OFF Model and reference parameters.

Because the OFF period distribution is unlimited, the running arithmetic mean does not converge fast as can be seen in fig. 6.7. Consequently, the arithmetic mean remains below the expected value - even for large sample sizes like 1.7 millions. Shorter waiting times lead to higher utilization rates. This is exactly what can be observed with the ON/OFF Model and simulation periods of 10’000 s. The traffic generated with the reference parameters from table 6.2 does therefore have a larger average utilization rate of 1.72 %. The utilization could be corrected to fit exactly the reference value of 1 %, but the idea is to approximate the reference parameters for a simulation period of infinite length.

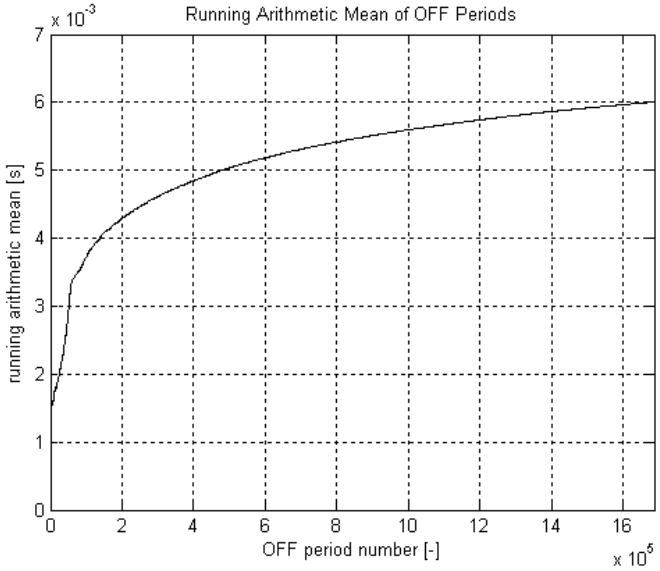

Figure 6.7: Running arithmetic mean of off-period distribution with ON/OFF Model and reference parameters.

A deeper insight into the generated traffic characteristic allow the time fraction traces of fig.

F.10 in appendix F.2.1. These plots show clearly that the link is working many times for at least 10 ms periods (smallest statistical time interval = ”unit”) at its full capacity close to 1 Gbps.

The aggregated traffic traces of fig. F.11 in appendix F.2.2 exhibit the decaying variance with increasing aggregation levels m. Considering that the y-scale in the figure is smaller for higher aggregation levels, the traffic variation becomes smoother as the aggregation increases. A better view of the variance as a function of the aggregation levels m gives the V-T Plot in fig. 6.8. The normalized variance decreases linearly for small aggregation levels up to the aggregation that corresponds approximately to the sending time of the largest file, which is 3.2 s in this example. The estimated Hurst parameter, derived from the linear curve fit as shown in the title of the V-T Plot in fig. 6.8, is 0.85. The expected Hurst parameter with the ON/OFF Model and reference parameters is 0.90.

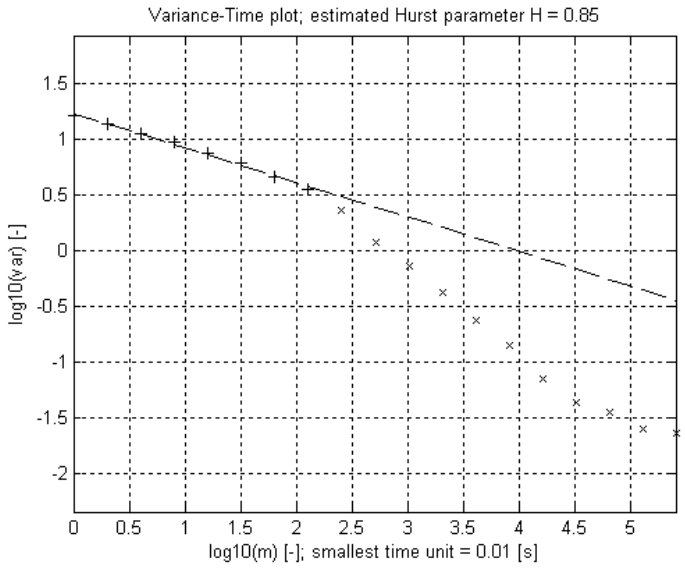

Figure 6.8: Variance-Time Plot of traffic from ON/OFF Model with reference parameters.

The data traffic discussed so far is obtained from only one seed for a simulation duration of 10’000 s. The V-T Plots and the Hurst parameters from 16 different seeds, giving cumulated 44 simulated hours, are analyzed and its results depicted in table 6.3.

A interesting finding from table 6.3 is that the Hurst parameter is higher for larger maximum file sizes. This relation can be better seen if the Hurst parameters and the maximum file sizes are plotted against each other as in fig. 6.9.

The V-T Plots for the traffic of all seeds in table 6.3 are packed into fig. 6.10. This multiple V-T Plot exhibits that for all seeds the variance trend is linear up to aggregation levels m of 2 (= 1 s) to 3.5 (= 32 s). This corresponds to the sending time for the smallest resp. the largest file in column *Max. File [MB]* in table 6.3. The V-T Plot further leads to the assumption that for smaller aggregation levels, not shown in fig. 6.10 (less than 0.01 s), the V-T Plots for all seeds become similar.

The multiple V-T Plot with recorded traffic over one second periods in fig. 6.11 continues 4 more decades to the left (smaller time scales) where the V-T Plot in fig. 6.10 ended. It shows the expected, high (normalized) variance of more than 10 at small time scales (1 μs), which remains almost constant up to aggregation periods of 10 to 100 μs, demonstrating the self-similar traffic characteristic with a Hurst parameter close to 1. Then, between 10 to 100 μs and 1 ms, the curves start falling and continue in the V-T Plot of fig. 6.10.

6.3 Circuit Emulation Traffic Delay

The theoretical background of traffic delay is already given in section 5.4.1 of this report. In this sections we have a look at the simulation results of CE traffic delay when CE *and* data traffic are sent over the bottleneck link of the simulation topology in fig. 5.1.

Seed	Max. File [MB]	Utilization [%]	Hurst Parameter [-]
0	369	1.71	0.85
1	1’366	1.84	0.90
2	619	1.69	0.86
3	149	1.63	0.78
4	224	1.63	0.80
5	271	1.66	0.83
6	135	1.59	0.77
7	2’300	1.98	0.92
8	690	1.71	0.86
9	496	1.69	0.83
10	198	1.61	0.79
11	641	1.74	0.87
12	741	1.75	0.86
13	388	1.62	0.80
14	232	1.66	0.83
15	4’005	1.99	0.94
Avrg.:	798	1.72	0.84

Table 6.3: Max. file sizes, average utilization rates and Hurst parameters H with the ON/OFF Model and reference parameters for 16 different seeds.

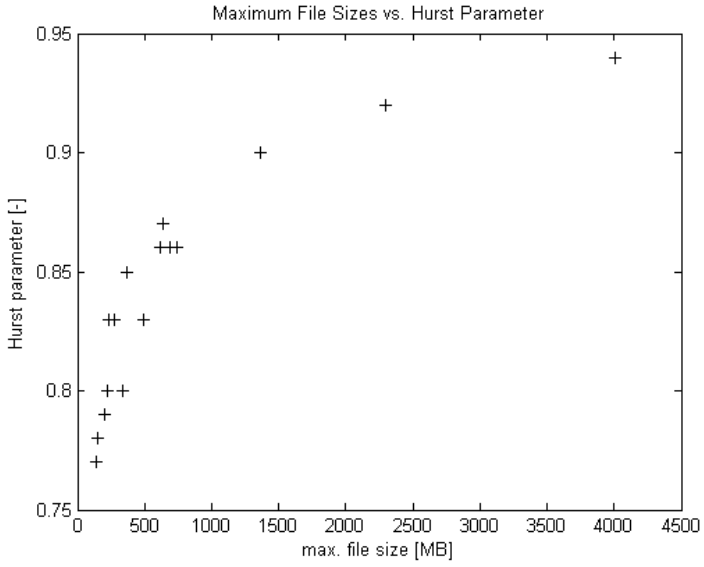

Figure 6.9: Maximum file sizes and their corresponding estimated Hurst parameters H with the ON/OFF Model and reference parameters for 16 different seeds.

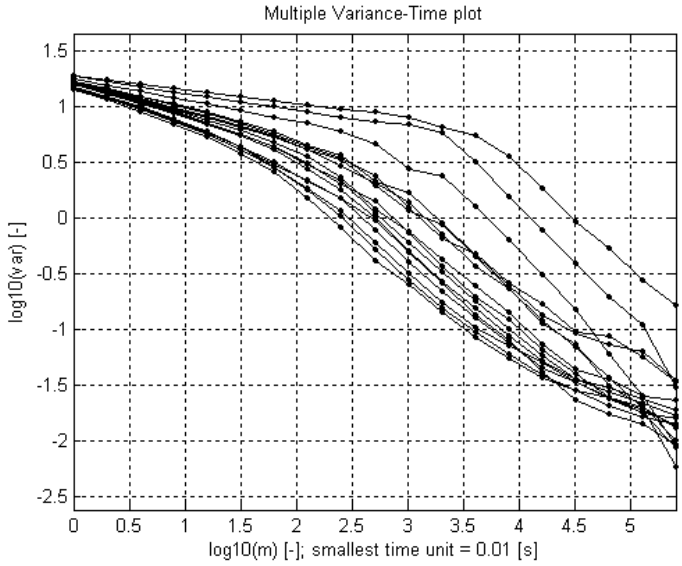

Figure 6.10: Multiple Variance-Time Plot with the ON/OFF Model and reference parameters for 16 different seeds.

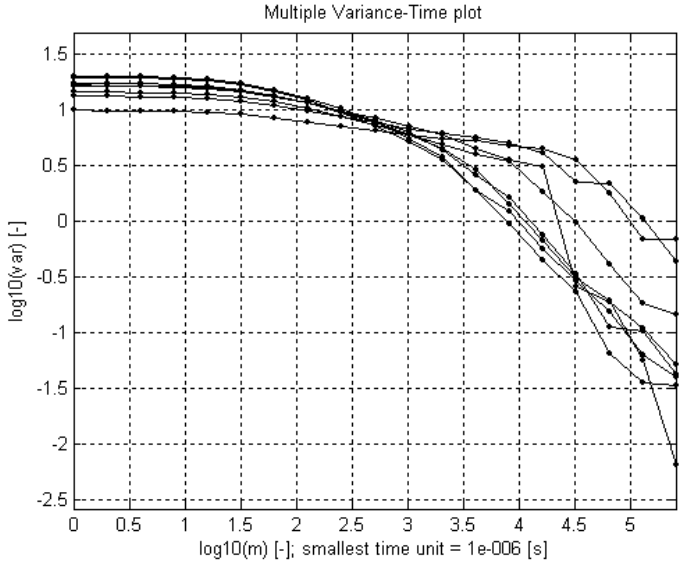

Figure 6.11: Multiple Variance-Time Plot with the ON/OFF Model and reference parameters for small time scales (1 s periods).

As long as the incoming traffic at the bottleneck link entry *switch_1* is less than the output link's capacity, no queues build up in this node and the delay of the CE traffic is constant at it's minimum value of approx. 56 μs as can be seen in the delay trace end in fig. 6.12.

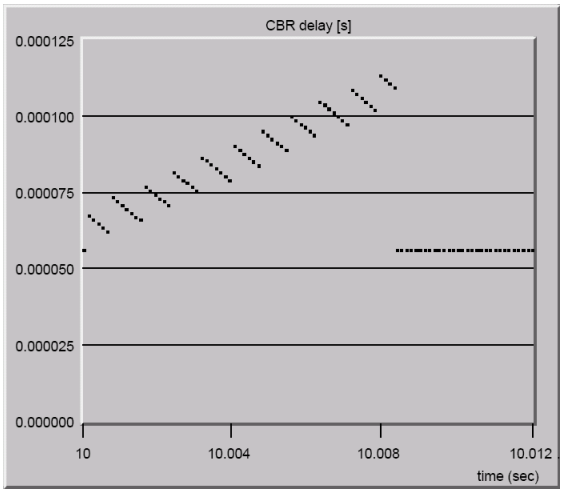

Figure 6.12: CE traffic delay when *VBR_source* sends one file of 10^6 bytes.

Before feeding the simulation with self-similar data traffic, the CE traffic delay is observed in a simulation with a special traffic pattern. For the data traffic, just one file of 10^6 bytes ($\approx$ 1 MB) is chosen. All buffers are of unlimited size. The VBR source sends the file at full link speed to *switch_1*. At the same switch, also the CE traffic arrives continuously and both traffic flows have to be sent over the bottleneck link to *switch_2*. Because the source link from the *VBR_source* to the *switch_1* transports data at its full capacity - the same capacity as the bottleneck link - the switch is not capable of sending the CBR *and* the VBR traffic over the bottleneck link without building up a queue at the sending interface. Queuing frames lead to additional delay. This is exactly what happens with the CE traffic shown in fig. 6.12. To send 10^6 bytes over a Gigabit Ethernet link takes a little more than 8 ms. During this time, the CE traffic delay ramps up. This delay is introduced by the growing queue size of the switch's *sending* buffer as shown in fig. 6.13. The delay and the queue size increases are consistent with the theoretical considerations in section 5.4.1, which are 5.76 ms per second for the delay and 624'000 bytes/s for the queue size.

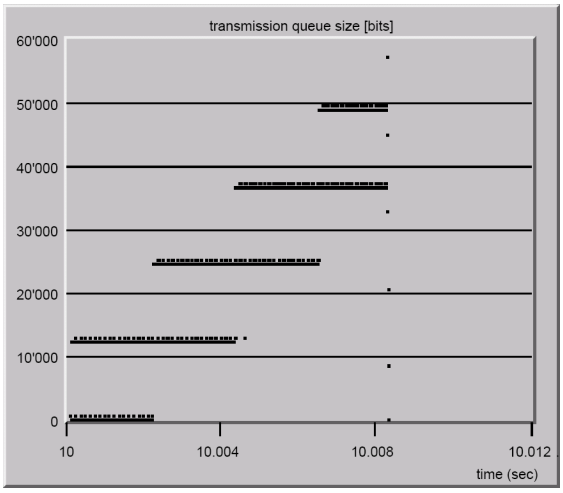

Figure 6.13: *Switch_1* sending queue size when *VBR_source* sends one file of 10^6 bytes.

The delay of the VBR traffic grows linearly as shown in fig. 6.14 because the delay counter starts to work as soon as the entire file is passed to the network layer. The queuing delays are very small compared to the transmission delay.

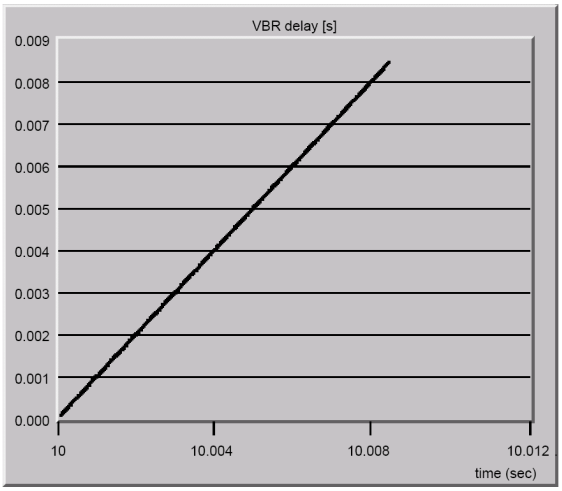

Figure 6.14: VBR traffic delay when *VBR_source* sends one file of 10^6 bytes.

6.4 Circuit Emulation Traffic Loss

Theoretical considerations and general information about data losses is already given in section 5.4.2 and 5.4.3. In this section, we focus on simulation results of CE traffic losses when CE *and* data traffic are sent over the bottleneck link of the topology in fig. 5.1. How the buffer queue builds up in *switch_1* is already discussed in the previous section 6.3. This section is about what happens when the buffer is full and starts to overflow.

Fig. 6.15 shows a common situation of a simulation in this work, where the buffer runs out of free space and works at its saturation point. The small variations in the queue size come from CE frames that are entering or leaving the queue whereas the large variations come from enqueued or dequeued large data traffic frames. One interesting observation is that the queue size just jumps between a few, repeating states. This synchronization effect leads to the lock-out phenomenon and allows a flow to monopolize queue space (see section 5.4.2). As a matter of fact, in the example of fig. 6.15, only data traffic frames are dropped and not a single CE frame. The lock-out phenomenon in the simulations in this work is expected to especially prevent CE frames from dropping when the queue size is limited by bits rather than by packets, because the CE frames are much smaller than the majority of the data traffic frames (see also section 5.4.2). Simulations with self-similar data traffic according the ON/OFF Model and reference parameters from table 6.2 actually reveal that the lock-out phenomenon is in massive favor of the CE traffic when the queue size is limited by bits. During 88 simulation hours, buffer overflows occur many times - with 1 MB buffer space but also with 2 MB. 99'822 frames are totally dropped, but not a single CE frame is among them - only data traffic.

If the buffer capacity is limited by packets, the probability that CE frames are dropped is much higher. Small frame sizes is no longer a advantage in the "battle" for buffer space. Only timing effects may still allow one traffic flow to monopolize buffer queues. Simulation results with self-similar data traffic show a interesting result: when the buffer capacity is limited by 500 packets, during 44 simulation hours, 61'298 data frames and 29'709 CE frames are dropped, giving a ratio of 0.485 of dropped CE frames to data frames. The same simulation shows a different result when the buffer is limited by 1'000 packets: 47'403 data frames and only 193 CE frames are dropped, giving the ratio of 0.004. These results make clear that timing effects can have a big impact on the drop rate of a traffic flow and that it can be difficult to predict them. Fig. 6.16 shows the queue length of a buffer, limited by 500 packets, operating at its saturation point. The queue size does vary differently than in the case of a bit capacity limitation (see fig. 6.15).

6.5 Quality of Service (QoS)

Simulation results of the QoS performance of the CE traffic is the subject in this section. The employed CE traffic model of a E1 carrier is explained in section 5.2 and the synthetic ON/OFF

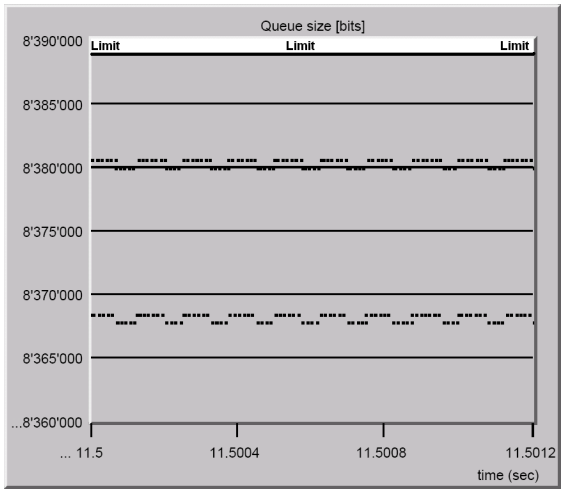

Figure 6.15: Queue size of *switch_1* when buffer with 8’388’608 bit (= 1 MB) limit is saturated.

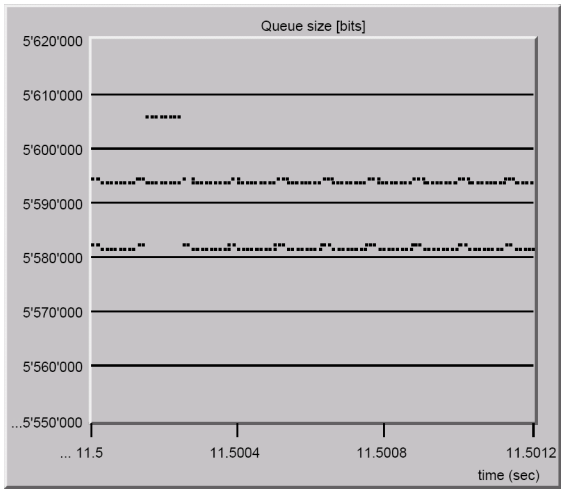

Figure 6.16: Queue size of *switch_1* when buffer with 500 packet limit is saturated.

workload data traffic model in 5.3. The reference parameters of the ON/OFF Model are according table 6.2 and the network topology is shown in fig. 5.1. QoS metrics and requirements are taken from the METRO Ethernet Forum (see section 5.4.3). The simulation time for one seed is always 10'000 s (= 2:47 h). For each switch buffer size limitation of 1 MB, 2 MB, 500 packets and 1'000 packets, 16 simulation runs are done with different seeds, giving cumulated 44:27 hours of simulated traffic time for each setup. The data traffic performance results are already presented and discussed in previous section 6.2.

6.5.1 Buffer Limitation by Bit Capacity

The simulation results for 1 MB buffer capacity limitation in table 6.4 can be summarized as follows: the CE traffic does fulfill for all tested seeds and the cumulative simulation period the QoS requirements. One reason is because the buffer size is small and therefore no frames can possibly be delayed more than 10 ms, which is the applied delay, resp. jitter limit. The maximum delay for 1 MB buffer size is 8.53 ms. When frames exceed the jitter limit, they would be discarded at the CE traffic receiving node. The total number of these frames is listed in column *Delayed CE Frames* in table 6.4 and its value is always 0. The second reason for the good QoS performance is because no CE frames are dropped due to buffer overflows (column *Dropped CE Frames*). In 14 of the total 16 runs the buffer queue reaches at least once its limit and frames are dropped, but without exception data frames. Over all simulation runs, total 53'907 data frames are dropped. The reason for that finding lies in the lock-out phenomenon and is introduced in the Simulation Design section 5.4.2 and analyzed, based on simulation results, in the Result section 6.4.

Seed	Dropped Data Frames	Dropped CE Frames	Delayed CE Frames	QoS
0	1'176	0	0	+
1	7'639	0	0	+
2	2'080	0	0	+
3	0	0	0	+
4	295	0	0	+
5	755	0	0	+
6	0	0	0	+
7	13'594	0	0	+
8	3'001	0	0	+
9	1'336	0	0	+
10	121	0	0	+
11	3'263	0	0	+
12	2'714	0	0	+
13	692	0	0	+
14	487	0	0	+
15	16'754	0	0	+
Total:	53'907	0	0	+

Table 6.4: QoS performance of CE traffic with ON/OFF Model and reference parameters for 16 different seeds and **1 MB** switch buffer limitation.

If the switch buffer limitation is increased to 2 MB, the QoS performance of the CE traffic is completely different. This is because when the queue size exceeds the theoretical limit of 1.033 MB (see section 5.4.1), the introduced delay does also exceed the jitter buffer time limit on the receiving side. Therefore, frames arrive too late and would be discarded. The numbers of discarded CE frames due to jitter buffer time violations are listed for each simulation run in column *Delayed CE Frames* of table 6.5. These values contribute to the Frame Error Ratio (FER). The applied FER limit $8.75 \cdot 10^{-7}$ corresponds to 70 errored CE frames per simulation run of 10'000 s or 1'120 errored CE frames for the cumulative period to fulfill the QoS requirements. In all simulation runs, where the queuing delays increase at least once to 10 ms, the CE traffic does not pass these QoS requirements (column *QoS* of table 6.5). This is the case for 13 of the total 16 simulation runs or 81 %. If all 16 simulation runs are taken together as if it was one

long, cumulative run of 44:27 h, the FER would be $6.7 \cdot 10^{-4}$ for the entire period - still far over the QoS requirement. As in the simulations before with 1 MB switch buffer, no CE frames are dropped (column *Dropped CE frames* of table 6.5) and therefore do not contribute to the FER value. But as one would expect, the amount of dropped data frames is reduced from total 53'907 to 38'915 due to the buffer enlargement.

Seed	Dropped Data Frames	Dropped CE Frames	Delayed CE Frames	QoS
0	129	0	15'891	-
1	5'275	0	122'178	-
2	1'150	0	31'102	-
3	0	0	0	+
4	0	0	1'736	-
5	0	0	8'655	-
6	0	0	0	+
7	12'220	0	228'198	-
8	1'625	0	47'125	-
9	648	0	20'742	-
10	0	0	0	+
11	1'246	0	49'502	-
12	1'648	0	39'997	-
13	4	0	9'739	-
14	0	0	3'628	-
15	14'970	0	280'088	-
Total:	38'915	0	858'581	-

Table 6.5: QoS performance of CE traffic with ON/OFF Model and reference parameters for 16 different seeds and **2 MB** switch buffer limitation.

6.5.2 Buffer Limitation by Packet Capacity

If the buffer size of *switch_1* is limited by packets, the QoS requirements are only met for one of the tested seeds but for none of the cumulative periods. Simulations are run with 500 and 1'000 packet limits. 500 packets is so small that no CE frames can possibly be delayed more than the 10 ms limit. Therefore, this scenario is comparable with the 1 MB buffer size limitation in previous section. But, in contrast to the bit capacity limitation, many CE frames are dropped. As shown in table 6.6, the number of dropped CE frames (column *Dropped CE Frames*) is about half of the dropped data frames (column *Dropped Data Frames*). This is a interesting result since the data traffic source sends about 10 times more frames per time than the CE source. If the drop probability was the same for both traffic flows, then also the drop rate for the data traffic should be about 10 times higher. Because it is not, the lock-out phenomenon works against the CE traffic. But in this cause, timing and not spatial effects allow one traffic flow to partially monopolize the buffer queue.

If the queue size limit is increased to 1'000 packets, a lot of CE frames arrive too late at the receiver's jitter buffer as in the simulation with 2 MB capacity limitation in previous section. The number of these discarded frames, listed in column *Delayed CE Frames* of table 6.7, is much higher than the dropped frames. Therefore, the delay dominates the QoS statistic. Another interesting observation is the ratio of dropped CE to data frames is much lower than with the 500 packet limitation. This indicates that the lock-out phenomenon is now working for the CE traffic. This further shows how unpredictable the underlying effects are.

6.5.3 Summary of all Buffer Limitations

The simulation results of the QoS performance for all 16 seeds and all buffer size limitations is summarized in fig. 6.17. Additionally, on the second y-axes the largest file size is drawn for every seed. Because the number of dropped and delayed frames are spread over many magnitudes, the values are drawn in logarithmic scale. The QoS limit for the total number of errored CE frame

Seed	Dropped Data Frames	Dropped CE Frames	Delayed CE Frames	QoS
0	1’732	832	0	-
1	8’524	4’137	0	-
2	2’955	1’428	0	-
3	243	112	0	-
4	733	349	0	-
5	1’235	599	0	-
6	91	42	0	+
7	13’923	6’767	0	-
8	3’366	1’631	0	-
9	1’653	798	0	-
10	339	164	0	-
11	3’850	1’857	0	-
12	3’474	1’681	0	-
13	898	431	0	-
14	1’303	632	0	-
15	16’979	8’249	0	-
Total:	61’298	29’709	0	-

Table 6.6: QoS performance of CE traffic with ON/OFF Model and reference parameters for 16 different seeds and **500 packet** switch buffer limitation.

Seed	Dropped Data Frames	Dropped CE Frames	Delayed CE Frames	QoS
0	721	6	15’825	-
1	6’728	28	122’005	-
2	1’612	7	31’049	-
3	0	0	0	+
4	0	0	1’736	-
5	299	5	8’620	-
6	0	0	0	+
7	13’137	46	228’060	-
8	2’546	11	47’022	-
9	1’108	3	20’693	-
10	0	0	0	+
11	2’580	17	49’351	-
12	2’107	5	39’946	-
13	464	3	9’690	-
14	33	1	3’624	-
15	16’068	61	279’916	-
Total:	47’403	193	857’537	-

Table 6.7: QoS performance of CE traffic with ON/OFF Model and reference parameters for 16 different seeds and **1’000 packet** switch buffer limitation.

per seed (70) is the dashed line labeled ”QoS Limit”. The number of errored CE frames is the sum of dropped and delayed CE frames, represented by the top value of every bar.

Fig. 6.17 allows a new insight into the relation of the largest file size to the number of errored frames: there are generally more errored frames when the maximum file size is larger and vice versa. This also shows that the maximum-sized file is the main cause for the errored frames. Another result from the comparison of different buffer size limitations is that the total number of errored frames for 2 MB and 1’000 packet buffer limitations are almost equal because not the dropped, but the delayed frames dominate the errored frame statistic.

The distributions of the number of errored frames per seed for all buffer capacity limitations is depicted in fig. 6.18. Because the number of seeds is only 16, there are actually too few seeds to draw founded conclusions like confidence intervals or distribution shapes. One can probably

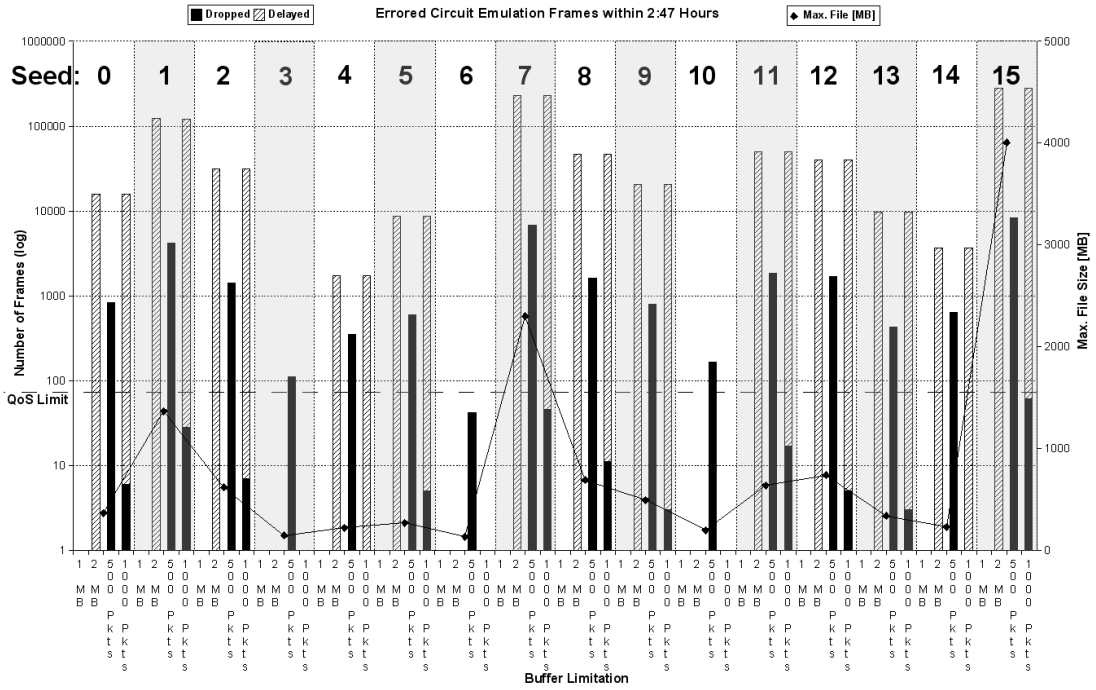

Figure 6.17: QoS performance of CE traffic with ON/OFF Model and reference parameters for 16 different seeds and different switch buffer limitations.

speculate that the distributions for the 2 MB and 1’000 packet buffer limits are heavy-tailed.

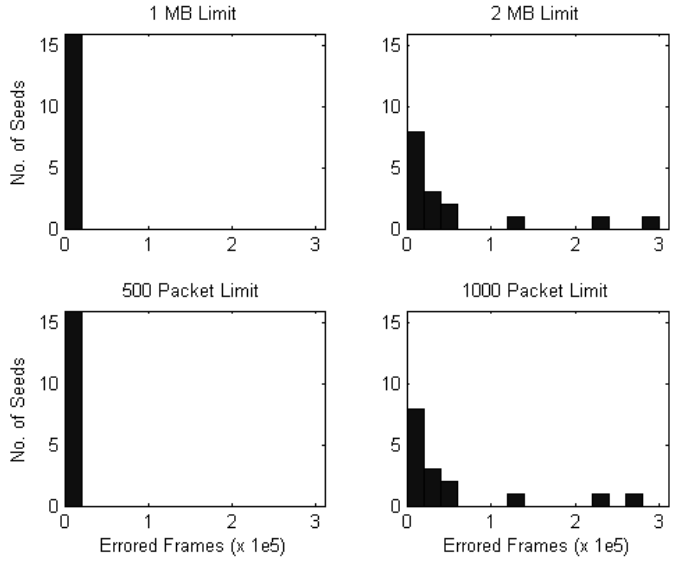

Figure 6.18: Distributions of errored frames per seed with ON/OFF Model and reference parameters for different switch buffer limitations.

Fig. 6.19 shows the results for the cumulative simulation period, i.e. all 16 simulation runs of 10’000 s with different seeds are cumulated to one period of 160’000 s (= 44:27 h). For buffer size limitations of 2 MB, 500 and 1’000 packets, the number of errored frames is 1 to almost 3 magnitudes higher than the QoS limit (1’120). Only in the case of 1 MB buffer size limitation, the QoS requirements are clearly fulfilled with zero errored frames. fig. 6.19 also shows one more time that no CE frames are dropped in the case of bit capacity buffer limitation. This certainly reflects the expected lock-out phenomenon because of spatial effects, but the result is probably over-estimated because the data traffic consisted mainly of maximum-sized frames.

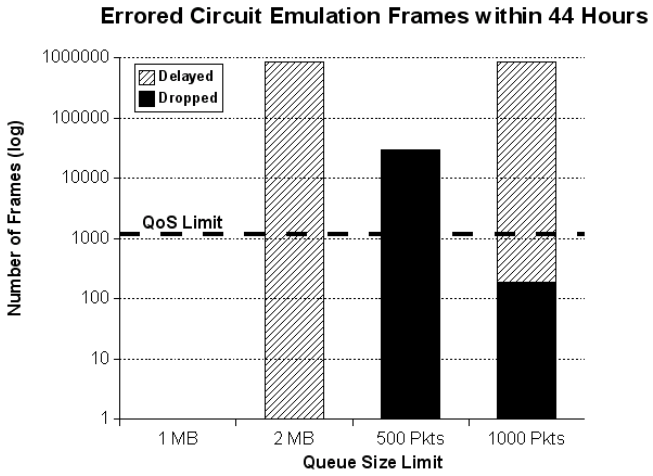

Figure 6.19: QoS performance of CE traffic with ON/OFF Model and reference parameters for cumulative period and different switch buffer limitations.

Chapter 7

Outlook

The main subject that could not be investigated in this thesis is the CE traffic performance with service priority. This is because OPNET Modeler 10.5A does not (yet) support service priority standard IEEE 802.1Q/p [16] [17]. If future versions of the software are supporting it, further simulations can be done to investigate also that subject.

All network simulations in this thesis are run with FIFO queue scheduling and tail drop queue management to control buffer spaces. Simulation results show that tail-dropping allows in some situations a single traffic flow to monopolize queue space, preventing other connections from getting room in the queue. This lock-out phenomenon is observed intensified if the queue capacity is limited by bits rather than by packets. Spatial effects did massively prioritize the small CE frames from not being dropped in saturated switch buffers. Further works could include a market analysis of common queue scheduling and management algorithms in commercial switches. Depending on these results, extended simulations with different algorithms, like active queue management with Random Early Drop (RED) could eventually be done.

The ON/OFF Model is employed for all workload data simulations in this thesis. Since the data traffic model does delicately influence the CE traffic performance, other self-similar data traffic models could be considered for future works. Also recorded data from real networks could be played into the network simulations.

Chapter 8

Summary

In the Simulation Design chapter of this thesis, a solution proposal to the given problem is elaborated. It is a framework of a OPNET Modeler Project for the network simulations and a combination of MATLAB and C codes to generate the simulation input data and to analyze the simulation results. OPNET Modeler is a widely used commercial discrete event simulator for communication networks. The integrated Raw Packet Generator (RPG) of OPNET Modeler is first tested and identified to be inappropriate to generate self-similar data traffic for the simulations in this work. The second approach is the ON/OFF traffic model from Park and Willinger [9] [8]. MATLAB code generates with the ON/OFF Model workload data traffic and writes it to files. The network simulation environment OPNET Modeler then reads these files as ”scripted” input data. Quality of Service (QoS) requirements are derived from the METRO Ethernet Forum [11] to measure the circuit emulation performance.

Tests of 44 simulated traffic hours reveal that the results of this thesis strongly depend on the applied buffer size and limitation methods in the switches, given that the buffer queues use first-in-first-out (FIFO) queue scheduling and tail drop queue management algorithms. Large buffer sizes ($> \sim$1 MB) introduce too much delay so that the QoS requirements are clearly violated by exceeding the limit by almost three magnitudes. Small buffer sizes ($< \sim$1 MB) do improve the QoS, but if the buffer capacity is limited by packets, the high number of dropped frames also violates the QoS requirements. Only when the buffer capacity is limited by bits, the circuit emulation traffic does meet the QoS requirements in the simulations of this work. In this case, not a single circuit emulation frame is dropped during all simulation periods. The *lock-out* phenomenon allows the circuit emulation traffic to monopolize queue space, preventing data traffic from getting room in the queue. This phenomenon is the result of spatial effects: the small circuit emulation traffic packets have a higher chance to be accommodated in the saturated buffer than the about 20 times larger data traffic packets. However, we doubt that it is reasonable to rely on this effect when employing metropolitan Gigabit Ethernets to transport telephony traffic. Since the simulation environment OPNET Modeler does not support priority service standards IEEE 802.1Q/p, no statement can be made if service priority enables circuit emulation. The conclusion is therefore that without service priority, circuit emulation in metropolitan Gigabit Ethernets is very difficult or eventually not possible.

Bibliography

[1] John A. Rice, *Mathematical Statistics and Data Analysis*, University of California, Berkeley, USA. Duxbury Press, 2nd ed., 1995.

[2] Ulrich Fiedler, *Evaluating Performance in Systems with Heavy-Tailed Input. A Quantile-based Approach*, Swiss Federal Institute of Technology, Zuerich, Switzerland. Diss. ETH No. 15233, 2003.

[3] Ulrich Fiedler, Polly Huang, Bernhard Plattner, *Towards Provisioning Diffserv Intra-Nets*, Swiss Federal Institute of Technology, TIK, Zuerich, Switzerland. In *Proceedings of IWQoS'01*, pages 27–43, Springer, June 2001.

[4] Ulrich Fiedler and Bernhard Plattner, *Using Latency Quantiles to Engineer QoS Guarantees for Web Services*, Swiss Federal Institute of Technology, TIK, Zuerich, Switzerland. In *Proceedings of IWQoS'03*, pages 345–362, Springer, May 2003.

[5] Anja Feldmann, Anna C. Gilbert, Polly Huang, Walter Willinger, *Dynamics of IP traffic: A study of the role of variability and the impact of control*. In *Proceedings of the ACM/SIGCOMM'99*, Cambridge, Massachusetts, USA, August 29–September 1, 1999.

[6] Will E. Leland, Walter Willinger, Murad S. Taqqu, Daniel V. Wilson, *On the Self-Similar Nature of Ethernet Traffic (extended version)*. In *IEEE/ACM Transactions on Networking (TON)*, Volume 2 ,Issue 1, pp. 1–15, February 1994.

[7] Will E. Leland, Walter Willinger, Murad S. Taqqu, Daniel V. Wilson, *On the Self-Similar Nature of Ethernet Traffic*. In *Proceedings SIGCOM'93*,pp. 183–193, San Francisco, USA, September 1993.

[8] Kihong Park and Walter Willinger, *Self-Similar Network Traffic: An Overview*. In *Self-Similar Network Traffic and Performance Evaluation*, Chapter 1, Wiley-Interscience, New York, USA, 15 January, 2000.

[9] Walter Willinger, Murad S. Taqqu, Robert Sherman, Daniel V. Wilson, *Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level*. In *IEEE/ACM Transactions on Networking*, Vol. 5, No. 1, pp. 71–86, April 15, 1997.

[10] METRO Ethernet Forum, *Introduction to Circuit Emulation Services over Ethernet*.
http://www.metroethernetforum.org (2005-02-22)

[11] METRO Ethernet Forum, *Circuit Emulation Service Definitions, Framework and Requirements in Metro Ethernet Networks*., Technical Specification MEF 3, April 13, 2004.
http://www.metroethernetforum.org (2005-02-22)

[12] Andrew S. Tanenbaum, *Computer Networks*, Vrije Universiteit, Amsterdam, The Netherlands. Pearson Education International, 4th ed., 2003.

[13] Larry L. Peterson and Bruce S. Davie, *Computer Networks - A System Approach*. Morgan Kaufmann Publishers, 3rd ed., 2003.

[14] Bo Ryu, Steven Lowen, *Fractal Traffic Models for Internet Simulation*. In *Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, pp. 200–206, Antibes, France, July 04 - 06, 2000.

[15] Andrew Odlyzko, *The Internet and other networks: Utilization rates and their implications*, AT&T Labs - Research, September 12, 1998.

http://papers.ssrn.com/sol3/papers.cfm?abstract_id=132770 (2005-02-22)

[16] IEEE-SA Standards Board, *IEEE Std 802.1Q (1998) - IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA, December 8, 1998.

[17] IEEE-SA Standards Board, *IEEE 802.1p - Traffic Class Expediting and Dynamic Multicast Filtering*, published in *IEEE Std 802.1D (1998) - IEEE Standard for Information technology / Telecommunications and information exchange between systems / Local and metropolitan area networks / Common specifications / Part 3: Media Access Control (MAC) Bridges*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA, 1998.

[18] International Organization for Standardization (ISO), *ISO/IEC 15802-3:1998 - Information technology / Telecommunications and information exchange between systems / Local and metropolitan area networks / Common specifications / Part 3: Media Access Control (MAC) Bridges*, International Organization for Standardization (ISO), Geneva, Switzerland.

[19] OPNET Modeler 10.5 Product Documentation,

http://www.opnet.com/support (2004-03-31)

[20] Oliver Lamparter, *Ein Softwaresystem für die Verkehrsdatenanalyse in Kommunikationsnetzwerken*, Swiss Federal Institute of Technology, Zuerich, Switzerland. Diss. ETH Nr. 15070, TIK-Schriftenreihe Nr. 49, Shaker Verlag, 2003.

[21] P. Leys, J Potemans, B. Van den Broeck, J.Theunis, E. Van Lil, A. Van de Capelle, *Use of the Raw Packet Generator in OPNET*, Katholieke Universiteit Leuven, Division ESAT-Telemic, Leuven-Heverlee, Belgium.

http://www.esat.kuleuven.ac.be/telemic/networking/opnet.php (2005-02-22)

[22] J. Potemans, B. Van den Broeck, Y. Guan, J. Theunis, E. Van Lil, A. Van de Capelle, *Implementation of an Advanced Traffic Model in OPNET Modeler*, Katholieke Universiteit Leuven, Division ESAT-Telemic, Leuven-Heverlee, Belgium.

http://www.esat.kuleuven.ac.be/telemic/networking/opnet.php (2005-02-22)

[23] J. Potemans, J. Theunis, B. Van den Broeck, Y. Guan, E. Van Lil, A. Van de Capelle, *Modelling Fractal Internet Traffic: Additive dyadic Superposition of lognormal Distributions*, Katholieke Universiteit Leuven, Division ESAT-Telemic, Leuven-Heverlee, Belgium.

http://www.esat.kuleuven.ac.be/telemic/networking/opnet.php (2005-02-22)

[24] J. Potemans, B. Van den Broeck, J. Theunis, P. Leys, E. Van Lil, A. Van de Capelle, *A tunable discrete Traffic Generator based on a Hierarchical Scheme of Bernoulli Sources*, Katholieke Universiteit Leuven, Division ESAT-Telemic, Leuven-Heverlee, Belgium.

http://www.esat.kuleuven.ac.be/telemic/networking/opnet.php (2005-02-22)

[25] Mark E. Crovella and Lester Lipsky, *Long-lasting transient Conditions in Simulations with heavy-tailed Workloads*. In *Proceedings of the 1997 Winter Simulation Conference*, pp. 1005–1012, Atlanta, Georgia, USA, 1997.

[26] Mark E. Crovella and Azer Bestavros, *Explaining World Wide Web Traffic Self-Similarity*, Technical Report TR-95-015, Computer Science Department, Boston University, USA, October 12, 1995.

[27] Mark E. Crovella and Lester Lipsky, *Simulations with Heavy-Tailed Workloads*. In *Self-Similar Network Traffic and Performance Evaluation*, Chapter 3, pp. 89–100. Wiley-Interscience, New York, USA, 2000.

[28] S. Bajaj et al., *Is Service Priority Useful in Networks?*. In *Proceedings of the ACM Sigmetrics '98*, Madison, Wisconsin, USA, June 1998.

[29] Cisco Systems, *Introduction to Voice and Telephone Technology*, Session 401, Cisco Systems Inc., 1999.

http://www.cisco.com/networkers/nw99_pres/401.pdf (2005-02-22)

[30] Xinjie Chang, *Network Simulations with OPNET*. In *Proceedings of the 1999 Winter Simulation Conference*, Vol. 1, pp. 307–314, Phoenix, Arizona, USA, 1999.

[31] ITU-T, *ITU-T Recommendation G.826 (02/99): Error performance parameters and objectives for international, constant bit rate digital paths at or above the primary rate*, International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), February 1999.

[32] ITU-T, *ITU-T Recommendation G.114 (05/2003): One-way transmission time*, International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), May 2003.

[33] NIKOTEL, company for internet based telephony, based in San Diego (CA), USA.

http://www.www.nikotel.com/glossary (2005-02-22)

[34] Oriana Riva, *Analysis of Internet Transport Service Performance with Active Queue Management in QoS-Enabled Network*, Politecnico di Milano, carried out at University of Helsinki, Department of Computer Science, Milan, April 2003.

[35] Network Working Group, *RFC 2309 - Recommendations on Queue Management and Congestion Avoidance in the Internet*, The Internet Society, April 1998.

[36] Hsu-Hui Wang, *Self-Similarity on Network Systems with Finite Resources*, Department of Communications Engineering, National Chiao Tung University, Hsinchu, Taiwan, July 2003.

[37] Wendell Odom, *CCNA Self-Study - CCNA ICND - Exam Certification Guide*, The official self-study test preparation guide for the Cisco CCNA ICND exam 640-811, Cisco Press, *Exam Certification Guide* series, 4th Edition, Aug 4, 2003.

[38] Anja Feldmann, Polly Huang, Anna C. Gilbert, Walter Willinger, *Dynamics of IP traffic: A study of the role of variability and the impact of control*. In *Proceedings of the ACM/SIGCOMM'99*, Cambridge, USA, August 29–September 1, 1999.

Appendix A

Timetable

Date	Tasks / Milestones
2004-11-03	Start of diploma thesis. Hand-over of task description and software.
2004-11-17	First simple network simulations runnable.
2004-11-22	Time schedule discussed with advisers.
2004-12-01	First self-similar data traffic analysis available (Variance-Time Plot).
2004-12-08	First simulation results with the Raw Packet Generator (RPG) available.
2004-12-17	Start of implementation of new self-similar traffic model (ON/OFF Model).
2005-01-09	Hand-in of intermediate report.
2005-02-01	First Quality of Service analysis available.
2005-02-17	All simulations results available and data analysis finished.
2005-02-25	Hand-in of final report and CD-ROM with produced software.
2005-03-04	Presentation of thesis to Communication Systems Group at TIK of ETH Zürich.

Table A.1: Time schedule of diploma thesis

Appendix B

Original Problem

B.1 Introduction

Currently, dedicated infrastructures are employed to provide telephony and data services. The infrastructure for telephone services is entirely based on circuit-switching technology. The infrastructure for data services, in contrast, is based on packet-switching technology. In the near future, available capacities for packet-switched networks will be largely expanded due to widespread deployment of optical transmission technology. Then emulation of telephony circuit-switching (”circuit emulation”) can enable to use packet-switched networks to connect legacy telephony equipment to the telephony core network. This is of particular interest in emerging metropolitan Gigabit Ethernets where existing PBX and GSM base stations can be connected. Presumably this protects investments in existing infrastructure and creates additional revenue for service providers.

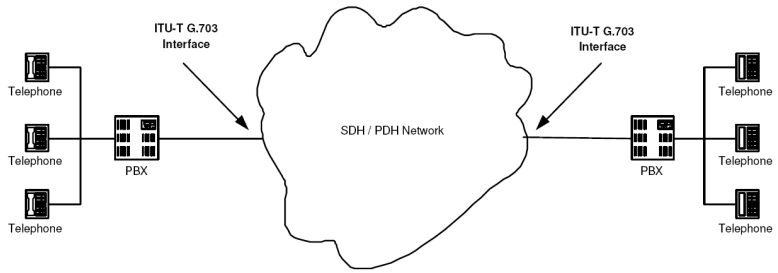

Figure B.1: Circuit Switching.

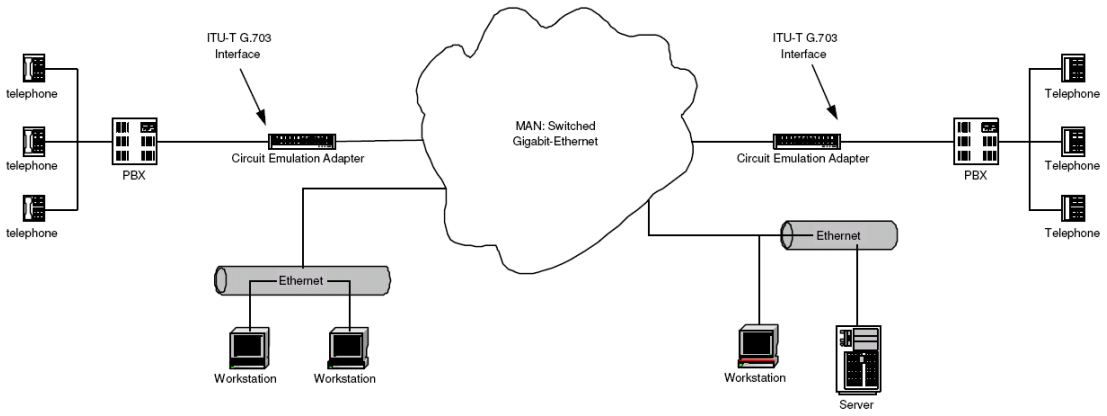

Figure B.2: Emulation of Circuit Switching in MANs.

However, research on traffic patterns in Ethernets [8] [6] suggests that circuit emulation may

have a Quality of Service (QoS) problem. The bursty patterns of data traffic can cause excessive queuing delays and frame losses due to buffer overflows. Due to the self-similar nature of the data traffic in Ethernets, the negative effect of these burstiness patterns can hardly be alleviated with buffering. Willinger et al. [6] gives indications that the underlying reason of this burstiness is the heavy tail in the size distribution of data transfers. This means that the distribution follows a power law for $x \to \infty$ with an exponent less or equal to two. Formally, this can be expressed as

$$1 - F(x) \sim x^{-\alpha}, \qquad \alpha \in (0, 2] \tag{B.1}$$

where $F(x)$ is the cumulative distribution function (cdf) and $a(x) \sim b(x)$ is defined by the relation

$$\lim_{x \to \infty} \frac{a(x)}{b(x)} = 1 \tag{B.2}$$

We further denote that the distribution has infinite variance since $\alpha < 2$.

Recent measurements indicate that the tail of the transfer size distribution usually only follows this power law for four of five decades (see Park and Willinger [8]). Hence, the tail deviates from a heavy tail at some limit. This limit is induced by limits in end systems such as the 4 GB file size limit inherent to most popular operating systems. As a consequence, the self-similarity in data traffic and thus the negative effects of burstiness are also limited. Moreover, line speeds of access links also limit traffic burstiness. Since Gigabit Ethernets operate at enormous line speeds, the problem whether metropolitan Gigabit Ethernets can be over-provisioned to accommodate the burstiness of the expected traffic becomes real.

With this provisioning issue being unclarified, major equipment vendors [37] recommend to assign dedicated virtual LANs (VLANs) for VoIP and CE traffic. This is a first step to enable preferential treatment to CE traffic since the IEEE 802.1Q VLAN standard [16] requires to support a priority field in the VLAN header. This priority field in turn can be used to implement preferential treatment of VoIP and CE traffic in Ethernet switches.

B.2 Assignment

B.2.1 Objectives

The objective of this post diploma thesis is to perform network simulation studies with OPNET [19] to assess whether and under which conditions service priority is inevitable to enable circuit emulation (CE) on Gigabit Ethernets. Much of the simulation methodology can be taken from Fiedler's IWQoS'01 paper [3] on provisioning DiffServ Intra-Nets. Feldmann's paper on the dynamics of IP traffic [5] can serve as a starting point in workload generation. The impact of the following parameters on the simulation results should be given special attention:

- Traffic burstiness properties and limits thereof. Particularly limits in the size distribution of transfer sizes.
- Traffic composition (data traffic versus CE traffic).
- Overall network utilization.

To limit the scope of the thesis, the impact of the network topology on the problem should not be given special attention. We think it is justified to focus on simulating CE over a bottleneck link. This bottleneck link may be viewed in a more general sense, since many have argued that there is always a single bottleneck link on any network path which is usually not fast moving [28].

Statistical significance of simulation results has to be explicitly shown, e.g. with normality tests, before conclusions are drawn from simulation results. This is necessary since simulations with workloads generated from heavy-tailed distributions inherently require an extremely large amount of input to reach stable state and show a large variability in stable state [27]. For methods on how to draw conclusions from transient simulations see Fiedler's IWQoS'03 paper [4].

B.2.2 Tasks

- Read the intro of the Park Willinger book on self-similar network traffic and performance evaluation [8]. This is to get an overview on notions such as self-similarity, long-range dependence, heavy-tailed distributions, their interrelations and their application in the characterization of network traffic.
- Get familiar with OPNET Modeler [19] as a simulation environment.
- Set up a simple implementation of workload generation for both data traffic and CE traffic in this environment.
- Implement and run a simple simulation of CE over a bottleneck link.
- Develop evaluation concepts to assess the quality of the CE in your simulation based on the documents of the METRO Ethernet Forum [11].
- Vary the parameters in the workload generation as well as the traffic composition and overall network utilization and measure the quality of the CE.
- Document your implementations, evaluation procedures and results.

B.2.3 Deliverables and Organization

- If possible, student and adviser meet or telephone on a weekly basis to discuss progress of work and next steps. If problems/questions arise that can not be solved independently, the student should not hesitate to contact the adviser anytime.
- At the end of the third week, a detailed time schedule of the thesis must be given and discussed with the adviser.
- In regular intervals (e.g. every two or three months) intermediate reports are due. These reports are linked to a short discussion of 15 minutes with the professor and the adviser. The student has to talk about the major aspects of the ongoing work including results, problems and remaining work.
- At the end of this thesis, a presentation of 15 minutes must be given either in teleconference or during the communication systems group meeting. The presentation should carefully introduce settings and background of the work. Moreover, it should contain an overview of the major results and conclusions of the work.
- All reports may be written in English or German. The final report must contain a summary, the assignment and the time schedule. Its structure should include an introduction, a methods/design section, a results section and a conclusion section. Moreover, the final report should include a complete documentation of all produced software. Related work must be correctly referenced. See http://www.tik.ee.ethz.ch/$\sim$flury/tips.html for more tips. Three hard copies of the final report must be delivered to TIK.
- Any software which is produced in relation with this thesis needs to be delivered including complete source code and documentation before ending the thesis. Moreover, the PDF and the complete source code employed to generate the final report including data to draw the figures also have to be delivered. Preferred format for delivery is a CD ROM.

Appendix C

Configuration

Parameter	Setting
Shape parameter α of Pareto-distributed file sizes	1.2
Shape parameter α of Pareto-distributed off-times	1.2
Number of on/off sources	100’000
Average network utilization rate	1 %
Average file size	12 kB
Maximum file size	4 GB
OPNET packet size limitation	268’435’456 bytes
Seeds	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Traffic duration	10’000 s

Table C.1: Configuration of the ON/OFF Model (MATLAB script).

Parameter	Setting
Hurst parameter H	0.90
Fractal Onset Time Scale FOTS	0.0008 s
Average Arrival Rate AAR	1190
Peak-to-Mean Ratio P2MR	100
Source Activity Ratio SAR	75
Seed	127
Traffic duration	1’000 s

Table C.2: Configuration of the Raw Packet Generator (OPNET Modeler).

Parameter	**Setting**
Switch buffer capacity limitation	1 MB, 2 MB, 500 packets, 1'000 packets
Switch buffer queue scheduling algorithm	firs-in-fist-out (FIFO)
Switch buffer queue management algorithm	Tail drop
Switch forwarding method	store-and-forward
Network technology	Switched Gigabit Ethernet (full-duplex)
Link technology	1000BaseX (optical)
Frame bursting	disabled
Bottleneck link propagation delay	50 μs
Access link propagation delays	0.05 μs
Attribute *Processing Information/Memory Size(bytes)* for the *ethernet_ip_station* node	256 MB
CE frame delay threshold for data logging	1 ms
Data logging start	10 s
"Bucket size" for data traffic probe	0.01 s

Table C.3: Configuration of network simulation environment (OPENT Modeler).

Parameter	**Setting**
Minimum end-to-end delay	55.972 μs
Maximum acceptable jitter	10 ms
Frame Error Ratio (FER) limit	$8.75 \cdot 10^{-7}$
Availability ration limit	99.95 %
Number of transmitted CE frames per second	8'000

Table C.4: Configuration of Quality of Service (QoS) assessment (MATLAB script).

Parameter	**Setting**
Aggregation levels n	$n = 2^i \quad , i = 1, 2, 3, \ldots$
Maximum square deviation from least square linear line fit	0.05

Table C.5: Configuration of Variance-Time Plot (MATLAB script).

Appendix D

MATLAB Code

D.1 trace_analysis.m

```
% This MATLAB code reads in a data file containing traffic flow data
% (called 'traffic trace') captured by the simulation software OPNET. The
% data must be recorded in the 'bucket' mode, i.e. stochastic information
% for small time intervals. The code then generates a Variance-Time Plot
% for the traffic trace and also visualizes the time series in two multi-
% plots: the aggregated traffic traces for different aggragation levels and
% the traffic trace at different time scales (time fraction traces). Least
% square line fit generates a line through the data points in the Variance-
% Time Plot and the Hurst parameter is derived from the line's slope.
% Philipp Sager, 2005-02-19.
%********************** parameter definition ******************************
  clear all;
% path and name of data file [string]:
  file='C:\...\traffic.txt';
% data unit [string]:
  unit='throughput [bits/sec]';
% number of ignored header lines in data file [-]:
  n4=8;
% number of multi-plots [-]:
  n3=4;
% aggregation levels for aggregated traces multi-plot (exponent of 2) [-]:
  n5=4;
% max. square deviation from least square linear fit for V-T Plot [-]:
  dmax=0.1;
% min. time for self-similar traffic (for Hurst paremeter calc.) [s]:
  t_min=0.009;
% max. time for self-similar traffic (for Hurst paremeter calc.) [s]:
  t_max=1.8;
%********************** start of calculations ****************************
% Reads ASCII files that contain rows of space separated values. The
% resulting data is placed into an variable with the same name as the file
%(without the extension):
  fid=fopen(file,'r');
  A=textscan(fid,'%f %f','headerlines',n4,'treatasempty','#N/A');
  fclose(fid);
% removal of fields containing 'NaN' values:
  A=[A{1},A{2}]; A(isnan(A(:,2)),2)=0;
% number of samples:
  n=size(A,1)
% first timestamp in data:
  tmin=A(1,1)
```

```
% last timestamp in data:
  tmax=A(n,1)
% smallest time interval:
  delta_t=A(2,1)-A(1,1)
% plot first aggregated trace:
  A=A(:,2)'; l=1;
  aggregate=figure();subplot(n3,1,l);
  plot(A,'k'); title('Aggregated traces');
  xlabel(['time units, unit = ',num2str(delta_t),'[s]',
 '; aggregation level m = 1'])
  ylabel(unit); grid on; box on; yscale=ylim; axis([0 n yscale]);
% plot timefraction traces:
  figure
  for i=1:n3
      subplot(n3,1,i); plot(A,'k')
      if i==1
          title('Time fraction traces')
      end
      xlabel(['time units, unit = ',num2str(delta_t),'[s]'])
      ylabel(unit); grid on; box on;
      yscale=ylim; axis([0 n yscale]); axis([0 n/10^(i-1) yscale])
  end
% normalized variance:
  mu=mean(A); var=mean((A/mu-1).^2);
% result matrix A2:
  A2(1,1)=1; A2(1,2)=var;
% calculating aggregated time series and their normalized variances:
  k=0; n
  while n>=4
      k=k+1;
      A=(A(1:2:n-1)+A(2:2:n))/2;
      n=size(A,2)
    % normalized variance:
      var=mean((A/mu-1).^2);
    % result matrix A2:
      A2(k+1,1)=2^k; A2(k+1,2)=var;
    % plot aggregated traces:
      if (k==n5*l & l<n3)
          l=l+1;
          figure(aggregate)
          subplot(n3,1,l); plot(A,'k')
          xlabel(['time units; unit = ',num2str(delta_t*2^k),' [s];',
           ' aggregation level m = ',num2str(2^k)])
          ylabel(unit); grid on; box on; yscale=ylim; axis([0 n yscale]);
      end
  end
% evaluation of slope and Hurst parameter h with least-square method:
  A2=log10(A2);
  A2pos_init=A2(A2(:,1)<=log10(t_max/delta_t)&A2(:,1)>=log10(t_min/delta_t),:);
  A2neg_init=A2(A2(:,1)>log10(t_max/delta_t) | A2(:,1)<log10(t_min/delta_t),:);
  P=polyfit(A2pos_init(:,1),A2pos_init(:,2),1);
% calculate deviations from linear fit to all points:
  dA2=(A2pos_init(:,2)-(P(1)*A2pos_init(:,1)+P(2))).^2;
  dA2pos=dA2; A2neg=[]; A2pos=A2pos_init;
  while sum(dA2pos>=dmax)>0
      A2pos=A2pos_init(dA2<dmax,:);
      A2neg=A2pos_init(dA2>=dmax,:);
      P=polyfit(A2pos(:,1),A2pos(:,2),1);
```

```
      dA2=(A2pos_init(:,2)-(P(1)*A2pos_init(:,1)+P(2))).^2;
      dA2pos=(A2pos(:,2)-(P(1)*A2pos(:,1)+P(2))).^2;
      max(dA2pos)
  end
  A2neg=[A2neg_init;A2neg]
% calculation of Hurst parameter h from slope P(1):
  h=1+P(1)/2
% variance-time plot:
  figure
  if size(A2neg,1)==0
      plot(A2(:,1),P(1)*A2(:,1)+P(2),'--k',A2pos(:,1),A2pos(:,2),'+k')
  else
      plot(A2(:,1),P(1)*A2(:,1)+P(2),'--k',A2pos(:,1),A2pos(:,2),'+k',
        A2neg(:,1),A2neg(:,2),'xk')
  end
  title(['Variance-Time plot; estimated Hurst parameter H = ',
    num2str(h,'%1.2f')]);
  xlabel(['log10(m) [-]; ','smallest time unit = ',num2str(delta_t),' [s]']);
  ylabel('log10(var) [-]'); axis equal; grid on; box on;
```

D.2 traffic_gen.m

```
% This MATLAB code generates two files containing series of file sizes and
% interarrival times according to the ON/OFF traffic model. The files can
% then be used as inputs for a 'scripted' data source in the simulation
% environment OPNET. The ON/OFF Model generates files which sizes are
% heavy-tailed distributed (Pareto) with a maximum limit and unlimited
% heavy-tailed distributed (Pareto) waiting times.
% Philipp Sager, 2004-12-27.
%********************* parameter definition ****************************
  clear all;
% file name for file inter-arrival times [-]:
  file_arrival='C:\...\arrival.csv';
% file name for file sizes [-]:
  file_size='C:\...\size.csv';
% maximum file size [byte]:
  s_max=4294967296;
% shape parameter of file size pareto distribution [-]:
  alpha_file=1.2;
% shape parameter of off time pareto distribution [-]:
  alpha_off=1.2;
% ethernet bandwith [bits/s]:
  r=1e9;
% average network utilization rate [-]:
  util=0.01;
% average file size [byte]:
  s_avrg=1.2288e4;
% ethernet overhead size (preamble+SoF+addresses+length field+check-sum) [byte]:
  p_eth=26;
% maximum ethernet payload size (ethernet MTU) [byte]:
  p_max=1500;
% minimum ethernet payload size [byte]:
  p_pad=46;
% ethernet interframe gab [bits]:
  p_interframe=96;
% ip header size [byte]:
  p_ip=20;
% approx. length of traffic trace [s]:
  t_max=1300;
% OPNET packet (file) size limitation [bytes]
% (Node attribute "IP/IP Processing Information/Memory Size (bytes)"):
  s_opnet=268435456;
% seed of random generator:
  seed=0;
%********************* start of calculations ***************************
% resets the state of the random generator to its initial seed:
  rand('state',seed);
% average traffic flow [bits/s]:
  r_avrg=util*r;
% maximum size of ethernet payload without ip header (fragment size of files):
  p_frag=p_max-p_ip;
% time to send maximum sized ethernet frame:
  t_frame=(8*(p_max+p_eth)+p_interframe)/r;
% traffic generated with average sized file [bits]:
  p_file=fix(s_avrg/p_frag)*((p_max+p_eth)*8+p_interframe);
  s_remain=mod(s_avrg,p_frag);
  if s_remain ~= 0
    % padding of payload to minimum size:
```

```
      if s_remain < (p_pad-p_ip)
          s_remain=p_pad-p_ip;
      end
      p_file=p_file+((s_remain+p_ip+p_eth)*8+p_interframe);
  end
% avrg number of files (int) to generate traffic trace of length t_max [-]:
  n_max=round(t_max*r_avrg/p_file);
% average file flow [files/s]:
  n_avrg=r_avrg/p_file;
% num. calc. of file size location param. k to fit s_avrg and s_max [byte]:
  f=@(k)(alpha_file/(1-alpha_file))*(k^alpha_file)*
    s_max^(1-alpha_file)-k)-s_avrg;
  k_size=fzero(f,s_avrg)
% calculation of off time location parameter k to fit r_avrg [s]:
% time to send average sized file:
  t_file=fix(s_avrg/p_frag)*t_frame;
  s_remain=mod(s_avrg,p_frag);
% padding of payload to minimum size:
  if s_remain < (p_pad-p_ip)
      s_remain=p_pad-p_ip;
  end
  if s_remain ~= 0
      t_file=t_file+((s_remain+p_ip+p_eth)*8+p_interframe)/r;
  end;
  k_off=((1-n_avrg*t_file)/n_avrg)*(alpha_off-1)/alpha_off
% generating file size vector:
% generating n_max uniformly (0...1) distributed random values:
  rdm=rand(1,n_max);
% calc. file size vector with inverse cdf 1-(k/x)^alpha of pareto function:
  X_file=k_size*(1-rdm).^(-1/alpha_file);
% removing file sizes > s_max and rounding to nearest integer:
  X_file(find(X_file>=s_max))=[]; X_file=round(X_file); n_max=size(X_file,2);
% plot file sizes:
  plot([1:n_max],X_file,'k'); title('file sizes');
  xlabel('file number i [-]');ylabel('file size Xi [byte]');
  grid off; box on; yscale=ylim; axis([0 n_max yscale]);
% generating off time vector:
% generating n_max uniformly (0...1) distributed random values:
  rdm=rand(1,n_max);
% calc. off time vector with inverse cdf 1-(k/x)^alpha of pareto function:
  X_off=k_off*(1-rdm).^(-1/alpha_off);
  mean(X_off)
% total length of off times [s]:
  t_off=sum(X_off)
% transformation of off times to inter-arrival times:
  t_on=0;
  for i=1:n_max
      t_file=fix(X_file(i)/p_frag)*t_frame;
      s_remain=mod(X_file(i),p_frag);
    % padding of payload to minimum size:
      if s_remain < (p_pad-p_ip)
          s_remain=p_pad-p_ip;
      end
      if s_remain ~= 0
          t_file=t_file+((s_remain+p_ip+p_eth)*8+p_interframe)/r;
      end;
      t_on=t_on+t_file;
      X_off(i)=X_off(i)+t_file;
```

```
  end
% total length of generated traffic [s]:
  t_total=sum(X_off)
% average utilization rate of generated traffic [%]:
  util_eff=t_on/t_total
% plot file sizes vs. sending time:
  figure; plot(cumsum(X_off),X_file,'k')
  title('file trace');xlabel('time [s]'); ylabel('file size [bytes]');
  grid off; box on; yscale=ylim; axis([0 t_total yscale]);
% split files >s_opnet into consecutive files of size s_opnet
% (OPNET packet size limitation):
  a=find(X_file>s_opnet);
  for i=1:size(a,2)
      'File too large for OPNET:'
      X_file(a(1))
      n_complete=fix(X_file(a(1))/s_opnet);
      s_partial=mod(X_file(a(1)),s_opnet);
      if mod(X_file(a(1)),s_opnet) ~= 0
          X_file=[X_file(1:a(1)-1),linspace(s_opnet,s_opnet,n_complete),
            s_partial,X_file(a(1)+1:end)];
          X_off=[X_off(1:a(1)-1),linspace(0,0,n_complete),X_off(a(1):end)];
      else
          X_file=[X_file(1:a(1)-1),linspace(s_opnet,s_opnet,n_complete),
            X_file(a(1)+1:end)];
          X_off=[X_off(1:a(1)-1),linspace(0,0,n_complete-1),X_off(a(1):end)];
      end
      a=find(X_file>s_opnet);
  end
% write vectors to comma separated vector files:
  csvwrite(file_arrival,X_off');
  csvwrite(file_size,X_file');
```

D.3 qos_analysis.m

```
% This MATLAB code reads in two files: 'file_drop' contains queue drop
% information and 'file_delay' CE traffic delays. With the file content
% together with the defined parameters, the code calculates QoS metrics
% and evaluates if the recorded CE traffic does fulfil the defined QoS
% requirements.
% Philipp Sager, 2005-02-04.
%********************** parameter definition ******************************
  clear all;
% simulation log start [s]:
  t_start=10;
% simulation log end [s]:
  t_end=10010;
% number of sent CE frames per second [1/s]:
  f=8e3;
% minimum end-to-end delay [s]:
  delay_min=55.972e-6;
% maximum tolerated jitter [s]:
  t_jitter=0.01;
% Frame Error Ratio (FER) limit [-]:
  FER_limit=8.75e-7;
% availability ratio limit [-]:
  avail_limit=0.9995;
% path and name of data file with drop statistic:
  file_drop='C:\...\rb_output_file_drop.txt';
% path and name of data file with drop statistic:
  file_delay='C:\...\rb_output_file_delay.txt';
% number of discarded header lines in data files [-]:
  n_header=1;
% ID of VBR traffic source [INTEGER]:
  id_vbr_src=9;
% ID of VBR traffic sink [INTEGER]:
  id_vbr_dst=8;
% ID of CBR traffic source [INTEGER]:
  id_cbr_src=6;
% ID of CBR traffic sink [INTEGER]:
  id_cbr_dst=7;
% ID of dropping process [STRING]:
  id_proc_drop='top.gigabit_ethernet.switch_1.mac_0';
% ID of delay measuring process [STRING]:
  id_node_delay='top.gigabit_ethernet.CBR_sink';
%********************** start of calculations *****************************
% reading of drop statistic in 'file_drop':
  fid=fopen(file_drop,'r');
  if fid~=-1 % can open the file
      A=textscan(fid,'%f %d32 %d8 %d8 %s %s','headerlines',n_header);
      fclose(fid);
    % put cells into arrays:
      T_drop=A{1}; ID_drop=A{2}; SRC=A{3}; DST=A{4}; PROC=A{5}; MSG=A{6};
      clear A;
    % select only entries logged from process 'id_proc_drop':
      T_drop=T_drop(find(strcmp(PROC,id_proc_drop)));
      ID_drop=ID_drop(find(strcmp(PROC,id_proc_drop)));
      SRC=SRC(find(strcmp(PROC,id_proc_drop)));
      DST=DST(find(strcmp(PROC,id_proc_drop)));
    % select only entries sent from VBR traffic source:
```

```
      T_drop_VBR=T_drop(find(SRC==id_vbr_src));
    % select only entries sent from CBR traffic source:
      T_drop_CBR=T_drop(find(SRC==id_cbr_src));
      ID_drop=ID_drop(find(SRC==id_cbr_src));
    % remove simulation start offset:
      T_drop_VBR=T_drop_VBR(find(T_drop_VBR>=t_start));
      T_drop_VBR=T_drop_VBR-t_start;
      T_drop_CBR=T_drop_CBR(find(T_drop_CBR>=t_start));
      T_drop_CBR=T_drop_CBR-t_start;
  else % can not open the file
      T_drop_CBR=[];
      T_drop_VBR=[];
  end
% number of dropped frames [-]:
  n_drop_CBR=size(T_drop_CBR,1)
  n_drop_VBR=size(T_drop_VBR,1)
% reading of delay statistic in 'file_delay':
  fid=fopen(file_delay,'r');
  A=textscan(fid,'%f %d32 %f %s','headerlines',n_header);
  fclose(fid);
% put cells into arrays:
  T_delay=A{1}; ID_delay=A{2}; DELAY=A{3}; NODE=A{4}; clear A;
% select only entries logged from node 'id_node_delay':
  T_delay=T_delay(find(strcmp(NODE,id_node_delay)));
  ID_delay=ID_delay(find(strcmp(NODE,id_node_delay)));
  DELAY=DELAY(find(strcmp(NODE,id_node_delay)));
% remove simulation start offset:
  ID_delay=ID_delay(find(T_delay>=t_start));
  DELAY=DELAY(find(T_delay>=t_start));
  T_delay=T_delay(find(T_delay>=t_start));
  T_delay=T_delay-t_start;
% quantile calculation with compensation for not recorded delays < threshold:
  quant=quantile(DELAY,1-FER_limit*((t_end-t_start)*f)/size(DELAY,1))*1000
  plot(sort(DELAY))
% calculate number of lost frame due to jitter limit violation [-]:
  T_delay=T_delay(find(DELAY>(delay_min+t_jitter)));
  ID_delay=ID_delay(find(DELAY>(delay_min+t_jitter)));
  DELAY=DELAY(find(DELAY>(delay_min+t_jitter)));
  n_delay=size(DELAY,1)
% calculate total Frame Error Ratio (FER) [-]:
  FER=(n_drop_CBR+n_delay)/((t_end-t_start)*f)
  if FER>=FER_limit
      'Frame Error Ratio (FER) exceeds limit!'
  end
% calculation of availability ratio:
  LOSS=sort([T_drop_CBR;T_delay]);
  if size(LOSS,1)==0
      LOSS=[0,0];
  end
  n_bad=1; % number of consecutive one-second periods with at least one loss
  n_good=0; % number of consecutive one-second periods with no losses
  t=LOSS(1); % begin of (first) one-second period with losses
  t_unavail_start=t; % start time of unavailable status
  t_unavail=0; % accumulated duration in unavailable status
  i=2; % go to second loss
  while i<=size(LOSS,1)
      if LOSS(i) < t+1; % more than one loss in one-second period
         i=i+1; % go to next loss
```

```
    else
        t=t+1;
        if LOSS(i) < t+1 % next one-second period with loss
            n_bad=n_bad+1
            n_good=0;
            if n_bad == 10 % enter in unavailable status
                t_unavail_start=t+1; % store start of unavailable status
            end
        else % next one-second period without loss
            n_good=n_good+1
            if n_bad >= 10 % actual status is unavailable
                if n_good == 10 % enter in available status
                    t_unavail=t_unavail+(t+1-t_unavail_start);
                    n_bad=0; n_good=0; % reset
                  % initialize like a new start:
                    if i < size(LOSS,1)-1
                        t=LOSS(i);
                        n_bad=1;
                        i=i+1;
                    else
                        break
                    end
                end
            else % actual status is available
                n_bad=0; % reset
                if i < size(LOSS,1)-1
                    t=LOSS(i)
                    n_bad=1;
                    i=i+1;
                else
                    break
                end
            end
        end
    end
end
t=t+1;
if n_bad>=10 % unavailable status at end of loss serie
    if t+10 > t_end-t_start
        t_unavail=t_unavail+(t_end-t_start-t_unavail_start+10);
    else
        t_unavail=t_unavail+(t-t_unavail_start+10);
    end
end
avail= 1-(t_unavail/(t_end-t_start))
if  avail <= avail_limit
    'Availability is less than limit!'
end
```

Appendix E

C-Code

E.1 superpos.c

```
/* MATLAB MEX-File. Use                                                    */
/*         (X_file, X_arrival)=superpos(X_on,X_off,t_tot_min)              */
/* to call from MATLAB.                                                    */
/* The compiled C-code "superpos.dll" has to reside in the same file       */
/* directory as the calling MATLAB m-file.                                 */
/* This code takes the sorted vectors 'X_on' and 'X_off' with on- resp.    */
/* off-times of any number of traffic sources and returns the on- and      */
/* interarrival durations ('X_file' and 'X_off') of the superposed         */
/* traffic up to time 't_tot_min'.                                         */
/* Philipp Sager, 2005-02-01.                                             */

#include "mex.h"

/* The gateway routine */
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
    /* Declare variables. */
    double *pX_on, *pX_off, t, *pX_arrival, *pX_file;
    unsigned int dim, n=0, i_off=1, i_on=1;
    double t_on=0, t_start=0, T_on[1000];

    /* Check for proper number of input and output arguments. */
    if (nrhs != 3) {
        mexErrMsgTxt("Three input arguments required.");
    }
    if (nlhs > 2) {
        mexErrMsgTxt("Too many output arguments.");
    }

    /* Check data type of input argument. */
    if (!(mxIsDouble(prhs[0]))) {
    mexErrMsgTxt("Input array must be of type double.");
    }
    /* Check the dimensions of X_on */
    dim = mxGetN(prhs[0]);

    /* Create a matrix for the return argument */
    plhs[0] = mxCreateDoubleMatrix(1, dim, mxREAL);
    plhs[1] = mxCreateDoubleMatrix(1, dim, mxREAL);
```

```
    /* Assign pointers to the various parameters */
    pX_on = mxGetPr(prhs[0]);
    pX_off = mxGetPr(prhs[1]);
    t = mxGetScalar(prhs[2]);
    pX_file = mxGetPr(plhs[0]);
    pX_arrival = mxGetPr(plhs[1]);

    /* Do the actual computations */
    while (*pX_off <= t || n > 0)
    {   if (*pX_on < *pX_off) /* a source starts sending */
        {   n++; /* increment no. of parallel sending sources */
            T_on[n]=*pX_on; /* store n-th ON time */
            if(++i_on <= dim) /* not very last source starts sending */
            {   if ((n==1) && ((t_start+t_on) <= *pX_on))
                {   /* first source starts sending and t_on not
                     * overlapping with this sending period */
                   *pX_arrival++=*pX_on-t_start; /* write interarrival time */
                   t_start=*pX_on; /* store new sending start */
                }
                pX_on++; /* go to next ON time */
            }
            else /* very last source starts sending */
            {   if ((n==1) && ((t_start+t_on) <= *pX_on))
                {   /* first source starts sending and t_on not
                     * overlapping with this sending period */
                    *pX_arrival=*pX_on-t_start; /* write interarrival time */
                    *pX_on = *pX_off;
                }
            }
        }
        else /* a source stops sending */
        {   t_on=t_on+*pX_off-T_on[n]; /* add sending time from that source */
            n--; /* decrement no. of parallel sending sources */
            if(++i_off <= dim) /* not very last source stops sending */
            {   if (n==0 && ((t_start+t_on) < *pX_on))
                {   /* last source stops sending and t_on not
                     * overlapping with this sending period */
                    *pX_file++=t_on; /* write sending time */
                    t_on=0; /* reset sending time */
                }
                pX_off++; /* go to next OFF time */
            }
            else /* very last source stops sending */
            {   *pX_file=t_on; /* write sending time */
                break; /* exit while-loop */
            }

        }
    }
    return;
    return;
}
```

Appendix F

Plots

F.1 Raw Packet Generator

F.1.1 Peak-to-Mean Ratio (P2MR)

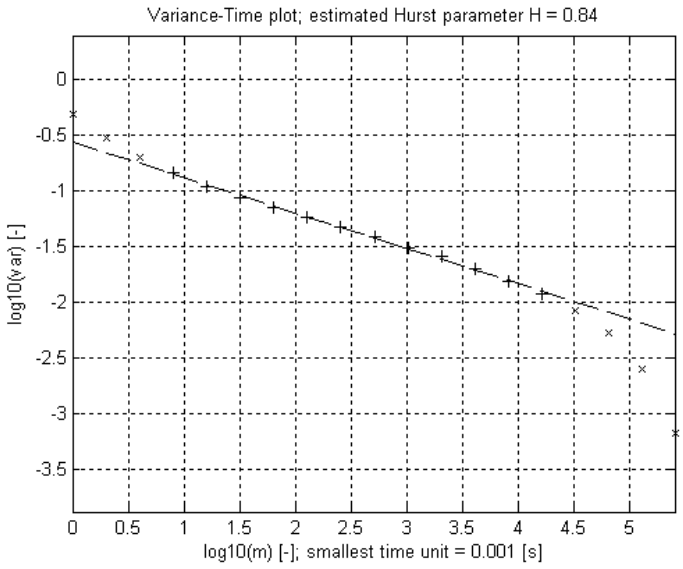

Figure F.1: Variance-Time Plot with RPG and P2MR = 50.

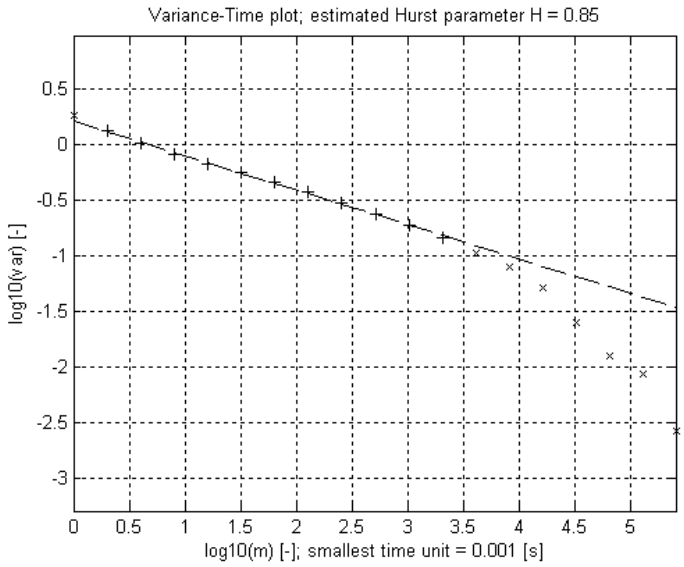

Figure F.2: Variance-Time Plot with RPG and P2MR = 200.

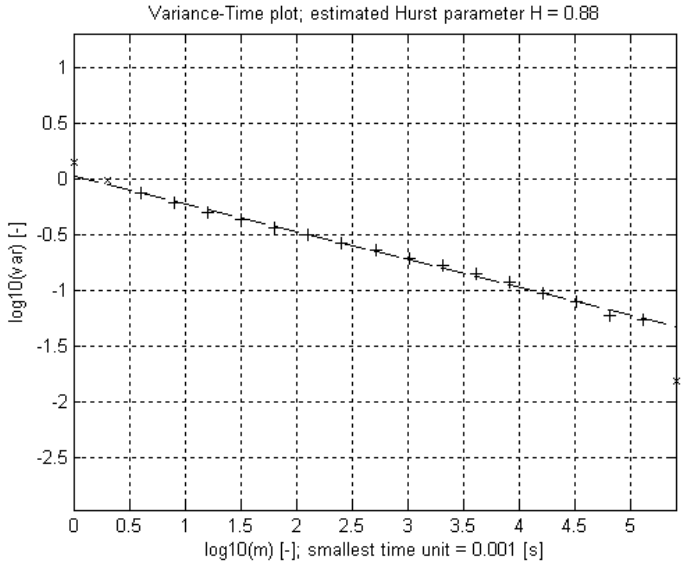

Figure F.3: Variance-Time Plot with RPG and P2MR = 1’000.

F.1.2 Hurst Parameter (H)

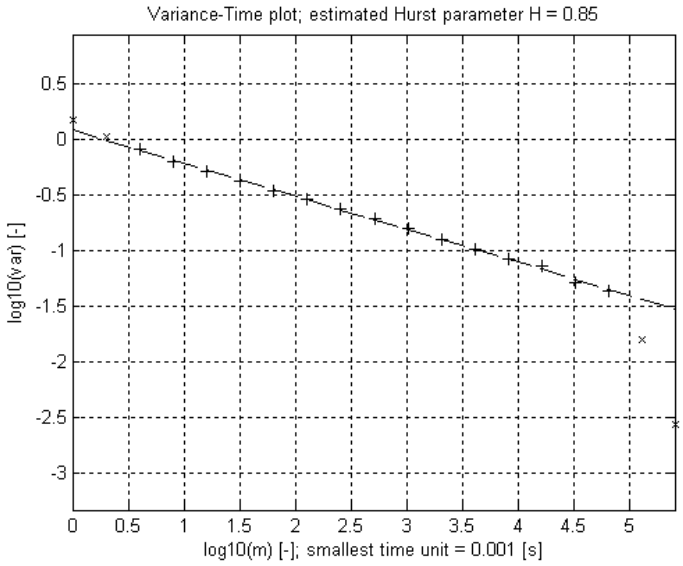

Figure F.4: Variance-Time Plot with RPG and H = 0.88.

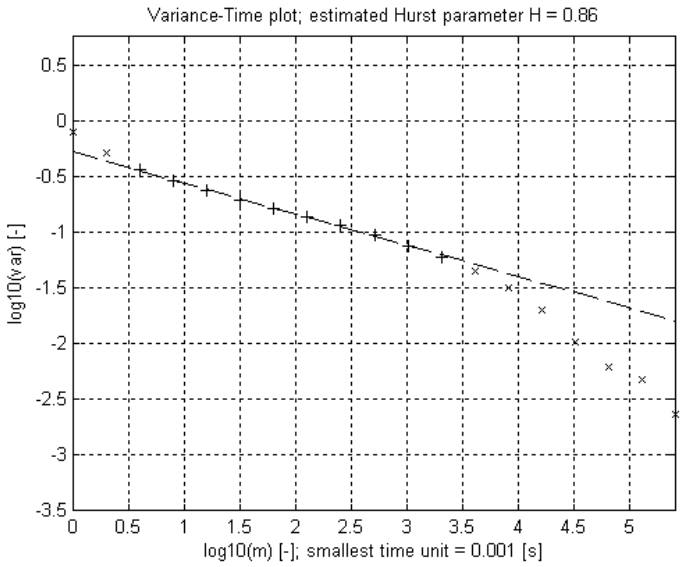

Figure F.5: Variance-Time Plot with RPG and H = 0.92.

F.1.3 Source Activity Ratio (SAR)

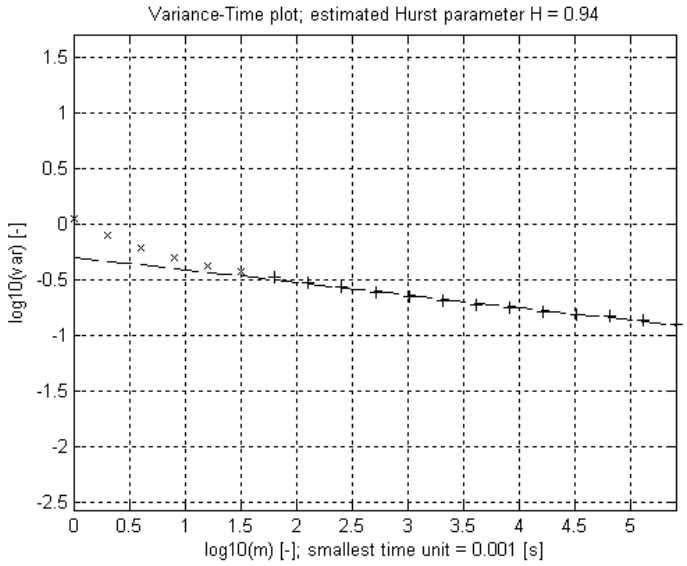

Figure F.6: Variance-Time Plot with RPG and SAR = 70 %.

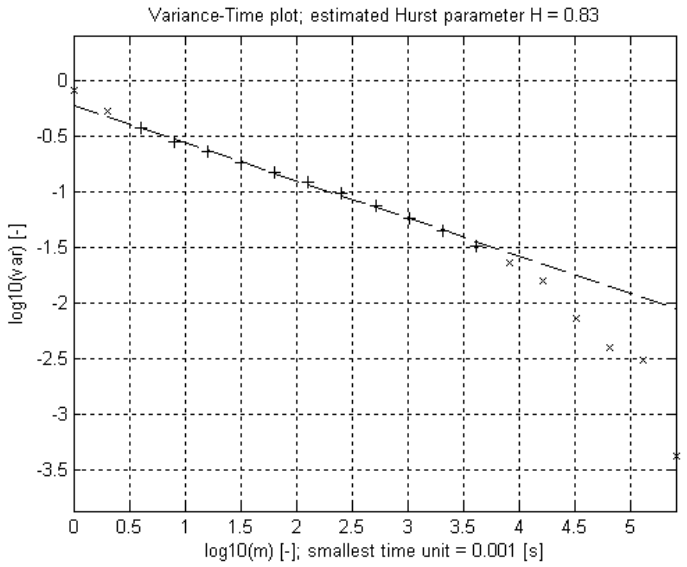

Figure F.7: Variance-Time Plot with RPG and SAR = 80 %.

F.1.4 Fractal Onset Time Scale (FOTS)

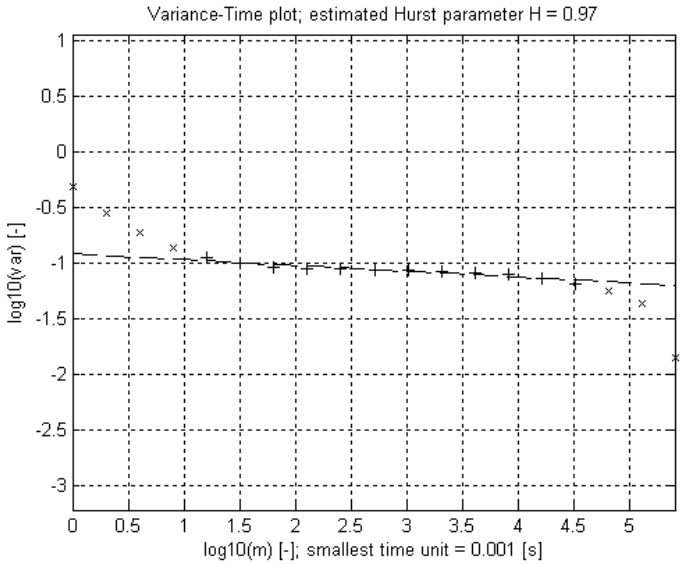

Figure F.8: Variance-Time Plot with RPG and FOTS = 0.0001 s.

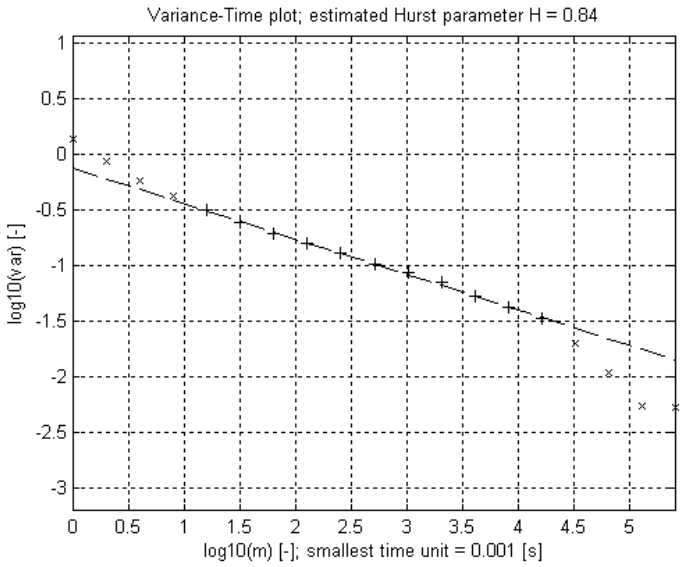

Figure F.9: Variance-Time Plot with RPG and FOTS = 0.002 s.

F.2 ON/OFF Model

F.2.1 Time Fraction Traffic Traces

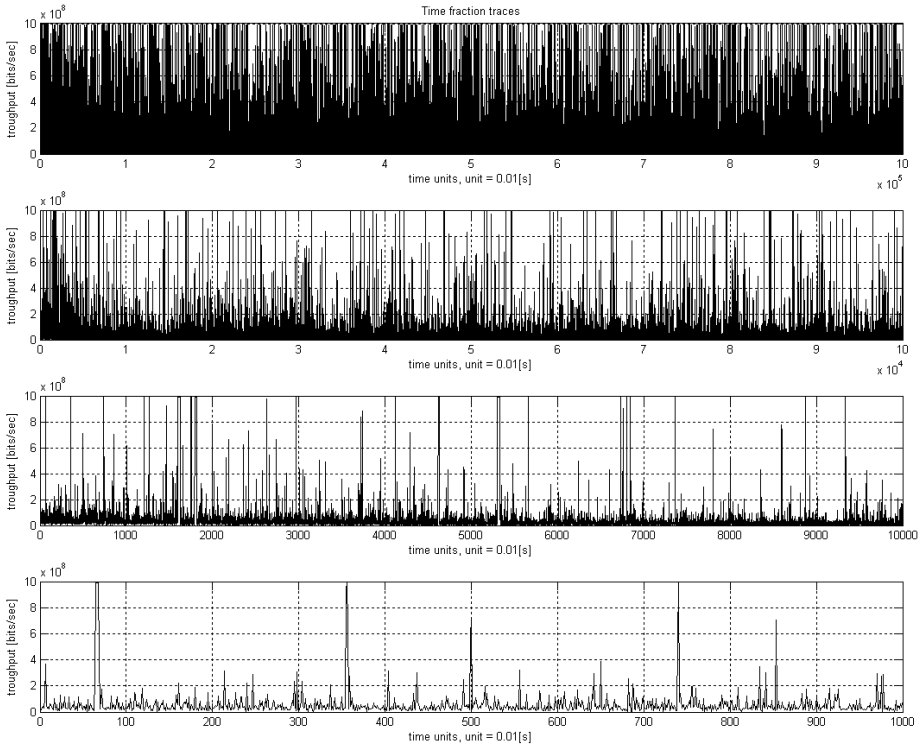

Figure F.10: Time fraction traffic traces with ON/OFF Model and reference parameters.

F.2.2 Aggregated Traffic Traces

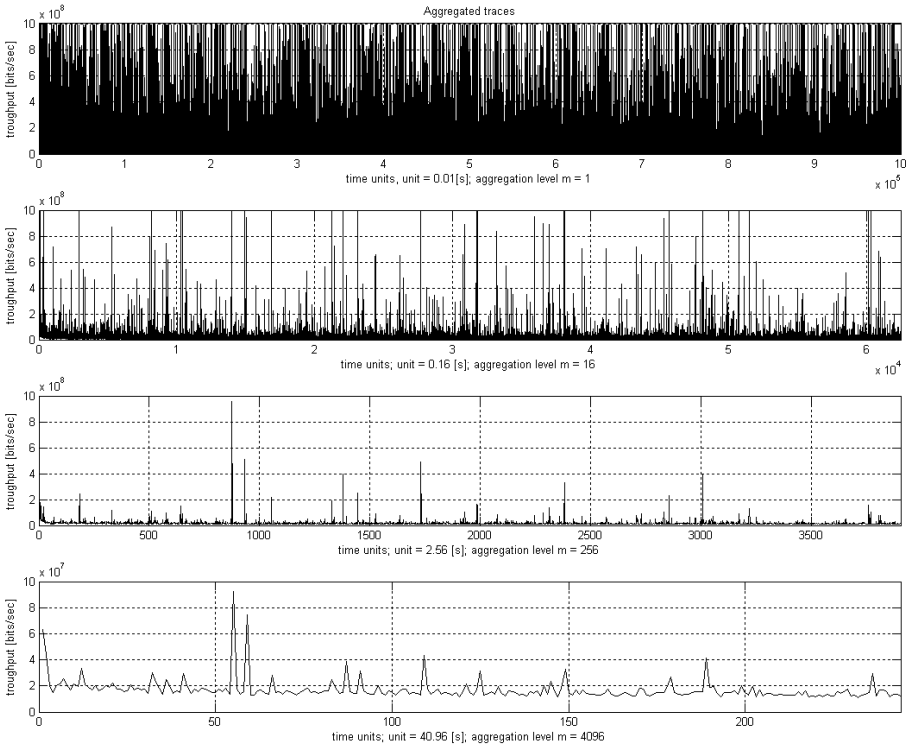

Figure F.11: Aggregated traffic traces with ON/OFF Model and reference parameters.