IP Traffic Statistics - A Markovian Approach

by
Thorsten R. Staake

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical Engineering

May 2002

APPROVED:

Professor M. C. Bromberg ECE Department Major Thesis Advisor	Professor D. R. Brown ECE Department Thesis Committee
Professor M. L. Claypool CS Department Thesis Committee	Professor J. A. Orr ECE Department Department Head

Abstract

Data originating from non-voice sources is expected to play an increasingly important role in the next generation mobile communication services. To plan these networks, a detailed understanding of their traffic load is essential.

Recent experimental studies have shown that network traffic originating from data applications can be self-similar, leading to a different queueing behavior than predicted by conventional traffic models. Heavy tailed probability distributions are appropriate for capturing this property, but including those random processes in a performance analysis makes it difficult and often impossible to find numerical results.

In this thesis three related topics are addressed: It is shown that Markovian models with a large state space can be used to describe traffic which is self-similar over a large time scale, a Maximum Likelihood approach to fit parallel Erlang-k distributions directly to time series is developed, and the performance of a channel assignment procedure in a wireless communication network is evaluated using the above mentioned techniques to set up a Markovian model. Outcomes of the performance analysis are blocking probabilities and latency due to restrictions of the channel assignment procedure as well as estimations of the overall bandwidth that the system is required to offer in order to support a given number of users.

Acknowledgements

Nothing is ever done alone. Thus, I would like to thank certain people who helped me along the way during my studies at the Worcester Polytechnic Institute.

Thanks to my advisor, Professor Matthew C. Bromberg, who, as I feel, was responsible for advancing my knowledge in the area of applied statistics, network traffic modeling and performance evaluations, and doing so invested a lot of time and effort in me.

Thanks to my thesis committee, Professor Donald R. Brown and Professor Mark L. Claypool, whose questions and comments greatly helped to improve the quality of this work.

Moreover, I want to express my gratitude to those who supported me financially: My parents, the German Academic Exchange Service (DAAD), and the *Zentrum für Interdisziplinäre Technikforschung* at the Darmstadt University of Technology, Germany.

Thorsten Staake

May, 2002

Contents

List of Figures

List of Tables

1 Introduction

The information exchange industry that includes fixed and wireless telephone as well as Internet access providers is by far the largest industry in the world [PK02]. Its continuous growth mainly results from the increasing number of Internet and cellular telephone users, and high hopes still lay on broadband wireless Internet access which is expected to develop a large market in the near future. Advantages of wireless systems are the gain in convenience for the end user and the efficient deployment of network access points in areas where the existing wired infrastructure is not highly developed. However, wireless broadband networks are difficult to design and it is questionable if the emerging third generation (3G) cellular systems as described in the IMT-2000 standard will fulfill the expectations. Nevertheless, enormous investments have been made in this field, and a large number of companies are currently working on 3G systems.

For those companies, an important questions to answer is how to dimension certain parts of the network in order to support a desired number of users. During the development process, statistical models can be used to predict system performance or to compare several alternatives.

When setting up a statistical model of a network, it is essential to understand the nature of traffic the system has to accommodate, and to choose an appropriate mathematical description of the traffic; here appropriate means with the desired level of detail and computationally feasible. For voice applications, detailed and relatively simple models exist, but traffic originating from certain data applications is found to be self-similar and hence more complicated to model [LWTW93]. Several approaches to describe network traffic are summarized in [WTE97], and we will introduce two more to model traffic with a large number of exponential wait states later: One known as the Prony method [dP95] where a Hyperexponential distribution is used to approximated long tailed prob-

ability distribution functions which can be used to generate self- similar time series, and a new technique to directly fit parallel Erlang-K distributions to observed data using Maximum Likelihood estimation. We will use the latter approach in a performance analysis.

The object under study in this thesis is a channel assignment procedure in a wireless network similar to the systems specified in the IMT-2000 standard, but optimized for data traffic. Of primary interest is the network's access scheme. For the performance analysis, a Markovian approach with a very large state space is used. However, an ongoing dispute is whether such conventional models can be used in the presence of self-similar traffic pattern: Some researchers feel it is necessary to develop completely new techniques to describe network traffic [CB95], other have the opinion that more sophisticated models using conventional techniques are appropriate in most cases [PKC97]. Both points of view are discussed later. To contribute to the discussion, we show that it is possible to generate traffic which appears self-similar over a large time scale with extended Markovian models, supporting the assumption that conventional traffic models are still useful when dealing with data networks.

Outcomes of the performance evaluation are blocking probabilities due to restrictions of the network's access scheme, where different numbers of possible simultaneous channel requests and different connection speeds are taken into account.

The remainder of this thesis is organized as follows. In Section 2, a review of the mathematics needed in the following work is provided, including definitions of the used time series distributions and self-similarity, as well as an overview of Maximum Likelihood estimation and Markovian statistics. Moreover, the essentials of the IMT-2000 code division multiple access (CDMA) system are summarized. In Section 3, an excerpt of the literature dealing with network traffic characterizations and modeling is given and the role of self-similarity in network traffic is discussed. In Section 4, an algorithm to

approximate long tailed distributions using Hyperexponentials is stated and the quality of the approximation is examined. In Section 5, an Maximum Likelihood approach to fit parallel Erlang-K stages directly to observed data is introduced, its Matlab implementation is given and the goodness of fit is examined. In Section 6, our traffic measurement on the Internet Protocol layer is described, where packet sizes and packet interarrival times were recorded. In Section 7, a simple ON/OFF model with long tailed state transitions is used to generate self- similar traffic; it is shown that with the Hyperexponential approximations of those transitions, the generated traffic is still self-similar over a large time scale. In Section 8, the model of the access scheme under study is introduced, using the traffic traces collected as described in Section 6 and the Erlang-K fitting approach developed in Section 5 to obtain a Markovian description of the model. Performance measures for different network configurations are derived. The thesis is concluded in Section 9, followed by the Appendix with the commented source code of all developed programs and a complete list of the considered references.

2 Mathematical Background and Essentials of CDMA

This chapter provides a review of the mathematics used in the following work: The time interval distributions needed are defined and the methodology of Maximum Likelihood parameter estimation is introduced. Moreover, the properties of Markovian state transition diagrams with their matrix form solution are given, and a definition of self-similarity and long range dependency are stated. Finally, the essentials of code division multiple access (CDMA) techniques used in communication networks are summarized.

2.1 Time Interval Distributions

Arrival processes, such as telephone calls reaching a central exchange, are described mathematically as point processes, consisting of arrivals at instants T_0, T_1 $\ldots T_n$. A mathematically equivalent description is the interarrival time process $\{A_n\}_{n=0}^{\infty}$, where the continuous function $A_n = T_n - T_{n-1}$ is the time separating the n^{th} arrival from the previous one, often referred to as waiting time. If the A_n are identically and independently distributed, one speaks of a renewal process. To describe the statistics behind the interarrival times, the random processes introduced in the next two subsections will be used.

2.1.1 Combining Exponential Distributions: Erlang-K, Hyper- and Hypoexponentials

Exponential distributions are characterized by a single parameter λ, often referred to as the transition rate. The probability density function (pdf) is given by

$$f(t) = \lambda e^{-\lambda t} \quad \lambda > 0,\ t \geq 0 \tag{1}$$

and the cumulative density function (CDF) by

$$F(t) = 1 - e^{-\lambda t} \quad \lambda > 0,\ t \geq 0. \tag{2}$$

The expected value is

$$m = \frac{1}{\lambda} \tag{3}$$

and the variance is

$$\sigma^2 = \frac{1}{\lambda^2} \tag{4}$$

Given the observed value vector is $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, the parameter λ can be estimated with

$$\lambda = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{5}$$

The most important property of an exponentially distributed random process is that it is memoryless, meaning time until the next event occurs does not depend on the time passed after the previous event; this characteristic is unique among continuous distributions and makes the random process attractive for analytical uses, for example in Markovian state descriptions, which are introduced later.

If a random process cannot be accurately modeled by a single exponential state, it can be useful to combine a number of exponentials leading to the more general class of phase type distributions; three special cases of phase type distributions, namely Erlang-K, Hyper- and Hypoexponentials, are introduced in the following. **Hypoexponential** distributions are obtained by combining K independent processes with exponential waiting time in series, as shown in Figure 1; this corresponds to addition of k independent stochastic variables and the probability density function is obtained by convolution.

These distributions are referred to as steep because the variance decreases when adding more stages while holding the overall mean fixed. In the most general case

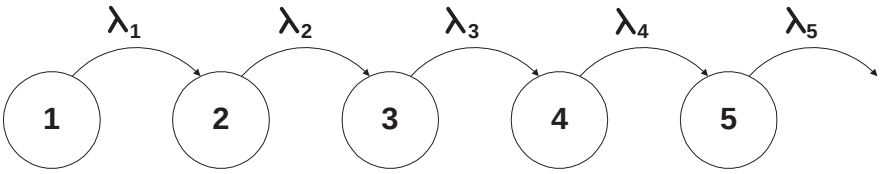

Figure 1: State Transition Diagram of Hypoexponential with 5 Stages

Hypoexponentials have as many parameters as exponential stages. Restricting the parameter space to a single transition rate λ yields to a **Erlang-K** distribution, where K is the number of exponential wait states. The corresponding probability density function is given as

$$f(t) = \frac{(\lambda t)^{K-1}}{(K-1)!}\lambda\, e^{-\lambda t}, \quad \lambda > 0,\ t >= 0,\ K = 1, 2, ... \tag{6}$$

and illustrated in Figure 2 with $K = 5$ and $K = 50$; λ_5 and λ_{50} are chosen so that the mean is 1 sec. The expected value and variance of an Erlang-K distribution are

$$m = \frac{K}{\lambda} \tag{7}$$

and

$$\sigma^2 = \frac{K}{\lambda^2}, \tag{8}$$

respectively. For an increasing number K of exponentials the variance becomes smaller when the expectation value m is held constant

$$\sigma^2 = \frac{m^2}{K}, \tag{9}$$

and the probability that a waiting time T lies within a given interval around m increases. For large K this property makes the Erlang-K distribution suitable for modeling deterministic processes.

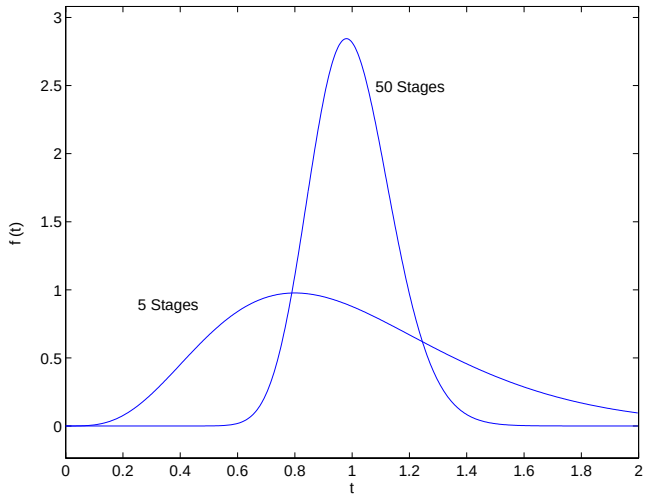

Figure 2: Probability Density Function of Erlang-K distributions with 5 and 50 Stages

A weighted sum of exponential distributions (exponentials in parallel, see Figure 3) is called a **Hyperexponential** distribution.

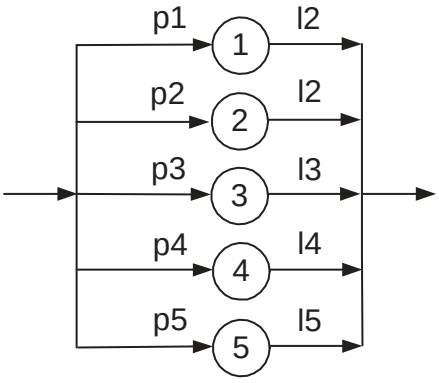

Figure 3: Hyperexponential

For this type of distribution the variance increases when adding more stages, therefore it is referred to as a flat distribution. This type of random process will be used to approximate probability density functions which decay slower than exponentially.

A Hyperexponential with N stages has N parameter tuples (p_i, λ_i), where the weight factors p_i have to be non-negative and sum up to one, and the λ_i are individual transition rates greater than zero. The probability density function is given by

$$f(t) = \sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i t}, \quad \lambda > 0,\ t >= 0,\ k = 1, 2, ... \tag{10}$$

The j-th non-central moment can be found with

$$m_j = j! \sum_{i=1}^{k} \frac{p_i}{\lambda_i^j} \tag{11}$$

Due to the large parameter space it is difficult to fit Hyperexponential distributions directly to observed data.

2.1.2 Long Tailed Distributions: Pareto and Weibull

In the following discussion, long tailed (or equivalent: heavy tailed) distributions will be used to describe packet sizes and interarrival times. Therefore the definition is stated shortly:

> A distribution is said to be long- or heavy tailed if its complementary cumulative distribution function (CCDF), regardless of its shape for small values of the random variable, has the following asymptotic behaviour:

$$P(T \geq t) \sim t^{-\alpha}, \quad for \;\; t \to \infty, \;\; and \;\; 0 < \alpha < 2. \tag{12}$$

Pareto and Weibull distributions are, for certain parameters, heavy tailed.

The Pareto Distribution

The Pareto CDF is a power curve that can be easily fit to observed data. It will be used to describe packet sizes later. Its CDF is given by

$$F(t) = \begin{cases} 1 - (\frac{t}{k})^{-\alpha-1} & \text{if } t > k \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The probability for $t < k$ is zero; k is the minimum value which can occur in a sample set. The probability density function (pdf) is

$$f(t) = \begin{cases} \alpha k(\frac{t}{k})^{-\alpha-1} & \text{if } t > k \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The mean and variance are given as follows:

$$m = \frac{k\alpha}{\alpha - 1} \quad \alpha > 1 \tag{15}$$

$$\sigma^2 = \frac{k\alpha}{(\alpha-1)^2(\alpha-2)} \quad \alpha > 2, \tag{16}$$

For $\alpha \leq 1$ the mean, and for $\alpha \leq 2$ neither the variance nor the mean does exist. This property makes it difficult to use Pareto distributions in queuing network analysis.

Referring to Jain [Jai91], the following equation can be used to estimate α for a given sample vector $\mathbf{x} = (x_1, x_2, \ldots, x_l)^T$, assuming that the smallest possible value k is known:

$$\alpha = \frac{1}{(1/n)\sum_{i=1}^{l} ln(\frac{x_i}{k})}. \tag{17}$$

The Weibull Distribution

Weibull distributions will be needed to describe packet interarrival times. Their CDFs are given by

$$F(t) = 1 - e^{-(\frac{t}{\alpha})^b}, \quad with \;\; \alpha, b > 0, \;\; and \;\; t \geq 0; \tag{18}$$

consequently, the density function is

$$f(t) = \frac{bt^{b-1}}{a^b} e^{-(t/a)^b}, \tag{19}$$

where a is the scale parameter, and b the shape parameter. For $b < 3.602$ Weibull distributions have a long right tail, for $b > 3.602$ a long left tail. The parameter can be estimated by solving the following set of equations for a and c:

$$a = [\frac{1}{n}\sum_{i=1}^{n} x_i^c]^{\frac{1}{c}} \tag{20}$$

$$c = [\ (\sum_{i=1}^{n} x_i^c log(x_i))\ (\sum_{i=1}^{n} x_i^c)^{-1}\ - \frac{1}{n}\sum_{i=1}^{n} log(x_i)\ ]^{-1} \tag{21}$$

2.2 Parameter Estimation and Maximum Likelihood Approaches

To describe a phenomenon with random processes, the parameters of the underlying distributions must be obtained. An optimal tool for this is Maximum Likelihood estimation which will be used to fit exponential, Weibull and Pareto distribution to data, as given above.

Generally speaking, if $f_{\mathbf{x}}(\mathbf{y} \mid \theta)$ is the pdf of a random vector **x** dependent on the unknown parameter vector θ, then the optimal parameter choice is those which maximizes $f_{\mathbf{x}}(\mathbf{y} \mid \theta)$ for a given sample vector **y**. In a later chapter, a Maximum Likelihood approach is introduced to fit a large number of parallel Erlang-K distributions to data. Referring to [FH98], the maximum likelihood estimation is asymptotically efficient, meaning that as the number of samples grow to infinity, the mean square error between the estimated parameters and the true parameters becomes as small as theoretically possible.

2.3 Markov Processes

In a later chapter a Markovian model of an access scheme for a wireless network will be developed; therefore the important properties of such processes are stated in the

following:

A set of random variables $\{X(t)\}$ with a discrete state space form a Markov chain if the probability that the next state is $x(t_j)$ depends only upon the current state $x(t_{j-1})$ and not upon any previous value. In other words, a Markov chain is a random sequence in which the dependency extends backward one unit in time. Therefore, the entire past history which affects the future must be summarized in the current state of the process. This implies that the distribution of time that the process remains in a state must be memoryless. For this reason, in continuous time Markov chains, the state time must be exponentially distributed. Usually each state of a Markov process is referred to as an integer number $0 \ldots n$. The exponential waiting time in state i is specified by the parameter λ_i. The probability that, when leaving state i the next state is j is defined as p_{ij}. A highly useful feature is that the process with n states can be described completely by a nxn matrix **P** containing the probabilities for going from one state to the other and by a vector of length n containing the waiting time of every state. A nxn transition rate matrix **Q** can be set up using the relationship

$$q_{ij} = \begin{cases} \lambda_i p_{ij} & i \neq j \\ -\lambda_i & i = j \end{cases} \tag{22}$$

The steady state probabilities are found by solving the following set of equations for π

$$\pi \mathbf{1} = 1 \tag{23}$$

$$\mathbf{0}^T = \pi \mathbf{Q} \ , \tag{24}$$

where **0** is the all zero vector; **1** is the all ones vector and the i-th element of vector π represents the steady state probability for being in state i. It is possible to find solutions for models with a very large state space.

2.4 Markov Modulated Processes and Burstyness

In a Markov modulated process the parameters of a model are determined by the state of an overlaying Markov chain. An example is a two state Markov modulated Poisson process, as illustrated in Figure 4:

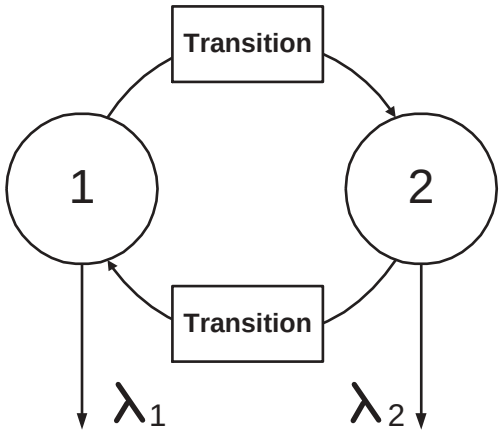

Figure 4: A simple 1-Burst Process

Lets assume that the system is initially in state 1; then the modulated process generates packets which are separated in time regarding to an exponential random process with parameter λ_1. After a random time, again exponentially distributed but with parameter μ_1, the modulating process transitions to state 2, causing the modulated process to generate packets with rate λ_2, until the system goes back to state 1 and repeats itself. More generally speaking, in a Markov modulated Poisson process the transition matrix of the modulated system depends on the state of the modulating process, which introduces a considerable degree of freedom when describing a system.

A special case is the so-called ON/OFF process, where packets are transmitted during the ON state and the generation rate while in the OFF state is set to zero, leading to a bursty traffic pattern. Bursty traffic is more difficult to handle in a queueing system than traffic generated from non-bursty sources which produce a more continuous workload.

Transmission burstyness is often measured by the following expression:

$$b = 1 - \frac{\kappa}{\lambda_p} \tag{25}$$

$\lambda_p =$ peak rate while ON

$\kappa =$ average rate over the entire process

Clearly $0 \leq b \leq 1$; for b equal to zero, the source is not bursty, for b approaching 1 we have a bulk arrival process.

2.5 Self-Similarity and Long Range Dependency

Self-similarity is a property used when describing fractals which appear the same regardless the scale at which they are viewed. When talking about stochastic time series, these series were referred to as self-similar when their aggregated distribution remains unchanged (except for changes in time scale) compared to the original distribution. This sort of behavior is different from simple Markovian models: When aggregating traffic generated by simple Markovian models, the resulting traffic stream is less bursty whereas the aggregation of self-similar traffic sources results in a bursty traffic stream leading to a different queueing behavior. A summary of the relevant papers covering this issue is given in chapter 3.

Definition of Self-Similarity

A stationary time series $X = (X_t;\ t = 1, 2, 3, \ldots)$ is statistically exact second-order self-similar if it has the same autocorrelation function $r(k) = E[(X_t - \mu)(X_{t+k} - \mu)]$ as the series X^m for all m, where X^m is the m-aggregated series $X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, \ldots)$ obtained by summing the original series X over nonoverlapping blocks of size m, $X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + X_{km-m+2} + \ldots + X_{km})$.

A stationary time series X is statistically asymptotically second order self-similar if autocorrelation $r^{(m)}(k)$ of X^m agrees asymptotically, i.e. for large k, with the autocorrelation $r(k)$ of X.

A detailed explanation of self-similarity is given by Crovella and Bestravos, [CB95]. Referring to [Cox84], a self-similar process has the following related properties:

- The variance of the sample mean decreases more slowly than the reciprocal of the sample size, $var(X^m) \sim m^{-\beta}$ as $m \to \infty$ with $0 < \beta < 1$
- The autocorrelation decays hyperbolically rather than exponentially fast. This shows that a self-similar process is long range dependent.

2.5.1 Determining the Degree of Self-Similarity

Self-similar traffic patterns can be detected by visual observation of traffic plots on different time scales. Self-similar traffic appears similar across many time scales whereas short range dependent time series look like noise after aggregating them. The visual test should be supplemented with a second, more mathematical technique which is discussed later.

The degree of self-similarity can be defined using the so-called Hurst parameter H, which expresses the speed of decay of the autocorrelation function. For a self-similar process, 1/2 < H <1; for H = 1/2 the time series is short range dependent, for H $\to$ 1 the process becomes more and more self-similar.

Since slow decaying variance and long range dependence (i.e. slow decaying autocorrelation functions) are both related to self-similarity, it is possible to determine the degree of self-similarity using either of those properties. In this work, the so called variance-time plot will be utilized, which relies on the slow decaying variance of every self-similar process.

For a self-similar time series, the variance of an aggregated process decreases linearly (for large m) in log-log plots over m. The slope β can be estimated using linear regression, leading to he Hurst parameter which is determined by the relation

$$H = 1 - \frac{\beta}{2}. \tag{26}$$

Figure 5 shows a variance-time plot with $\beta = 0.233$ leading to a $H = 0.88$.

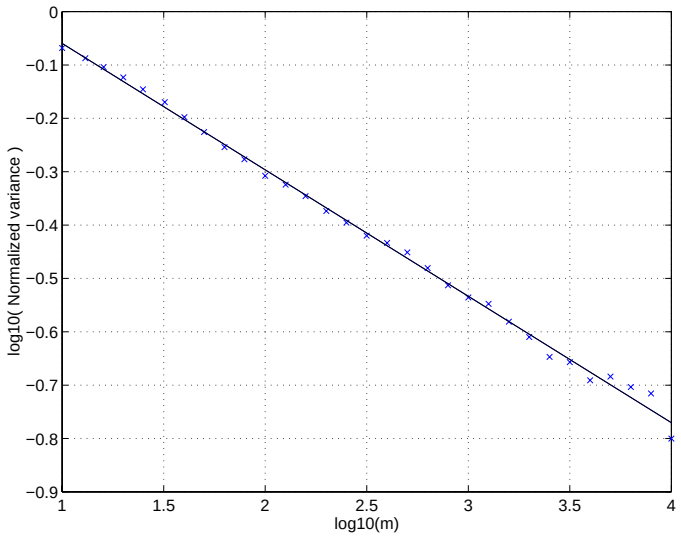

Figure 5: Variance-Time Plot

A second graphical method to determine H is the so-called R/S plot. The ratio R(n) over S(n) is determined by the following equation:

$$\frac{R(n)}{S(n)} = \frac{1}{\sigma(n)}[max(0, W_1, W_2, ..., W_n) - min(0, W_1, W_2, \ldots, W_n)], \qquad (27)$$

where σ is the square root of the sample variance

$$\sigma^2(n) = \frac{1}{1-n}\sum_{k=1}^{n}(X_k - \bar{X(n)})^2 \qquad (28)$$

and W_k is a measure for the deviation of the process X_k,

$$W_k = (X_1 + X_2 + \ldots + X_k) - k\bar{X(n)}. \qquad (29)$$

When plotting log[R(n)/S(n)] versus log(n), the slope equals H. A detailed explanation is given in [Rob00].

The drawback of the variance-time and the R/S plot is that both cannot be used to

derive confidence intervals of H. This is possible using Whittle's approximate Maximum Likelihood approach, which uses the property that any long-range dependent process approaches fractional Gaussian noise when aggregated to a certain level. The technique is explained in [LWTW93]. A paper comparing the various H estimation techniques is [TTW95].

2.6 Essentials of CDMA

When more than one user share a common media in a communication network, a technique must be employed to manage the access to this common media. In wireless systems, common techniques are frequency division multiple access, FDMA, where a certain frequency band is assigned to a user, or time division multiple access, TDMA, where data is transmitted separated in time. A different approach is code division multiple access, CDMA. Using CDMA, multiple simultaneous transmissions at the same frequency are separated using coding theory so that data streams which "collide" are added linearly and can be separated at the receiver.

A technique which can be employed for code division multiple access is direct sequence spread spectrum, DSSS. Using DSSS, each transmitted bit is mapped into N smaller pulses, referred to as a chip sequence, and sent using a conventional digital modulator. At the receiver the signal is demodulated and correlated with the chip sequence; a peak in the correlation function represents the original bit. The name *spread spectrum technology* expresses that after the mapping process an N-fold increase in bandwidth is necessary to transmit the bit in the same time.

In a multiuser environment, different unique spreading codes ("keys") are assigned to different users. The codes are chosen such that during decoding of the superimposed data stream, the frame originating from a single user can be obtained by a correlation

with the user specific chip sequence[1]. The correlations of data encoded with other keys do not show a distinct peak, but are a source of noise when detecting the desired data, which limits the number of users who can simultaneous transmit data. Therefore, in order to increase the system capacity, interference is typically reduced with power control mechanisms minimizing the transmission power per terminal.

Spread spectrum CDMA (DS-CDMA) systems have a number of advantages over conventional TDMA or FDD techniques: DSSS is resistant against multipath and frequency selective fading, and also against jamming, which makes the technique interesting for military applications. Moreover, CDMA systems provide a higher capacity[2] and a higher flexibility for integration of services with different bandwidth requirement, such as telephony and Internet access. For third generation (3G) cellular systems, CDMA is the technology of choice.

Ideally, no channel requests are necessary in CDMA systems where every user has its own spreading code and enough bandwidth is available. However, both, bandwidth and the number of spreading codes are limited, such that prior to a longer transmission a key has to be requested and the decision has to be made if enough bandwidth is available for another user. In this work, a channel allocation procedure of a CDMA network is evaluated. The system under study follows the IMT-2000 standard for 3G cellular implementations, but has minor modifications to optimize it for computer data traffic. The channel request procedure of a WCDMA mobile station as defined in [Ins97] is summarized and the differences to the modeled network are pointed out in the following: In WCDMA, one uplink access channel is provided and announced by the base station[3].

[1] Commonly used spearing sequences are orthogonal Walsh codes for synchronized systems and Gold codes, which are non-orthogonal but show an excellent correlation behavior even if no precise synchronization is available.

[2] The gain in capacity is far less than expected when comparing the second generation cellular phone systems IS-95 (CDMA) and GSM (FDD/TDD), for example.

[3] To be precise: One access channel is available in every 5MHz frequency band that the available spectrum is divided into.

The mobile stations use a random access technique similar to slotted ALOHA to transmit small amounts of data to the base station. Such packets are referred to as random access bursts and have the structure illustrated in Figure 6:

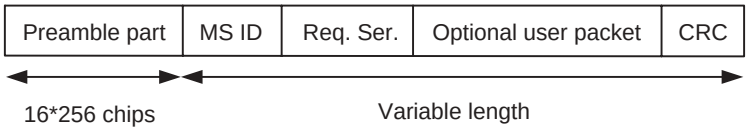

Figure 6: IMT-2000 Random Access Burst

- The preamble part contains a chip sequence, randomly chosen from a set of 16 orthogonal code words.
- The MS-ID sequence specifies which device is requesting service.
- The data part contains the optional payload.
- CRC is the checksum over the whole packet.

Before a packet is sent, the mobile station has to synchronize itself with the time slots of the base station. The mobile station then transmits the random access burst with a $2n$ ms time offset to the slot boundary, where n is chosen randomly between 0 and 4. An example is depicted in Figure 7.

With this scheme, 16x5 mobile stations may transmit within a 10 ms frame without collision, where 16 represents the different spreading codes and 5 the possible time offsets.

The system under study is optimized for data traffic, for example originating from Internet browsers. Due to its bursty nature, channels are assigned more often and for shorter durations than in networks which carry mostly voice traffic, leading to a higher utilization of the access channel. For the model discussed in Section 8.1, the number of

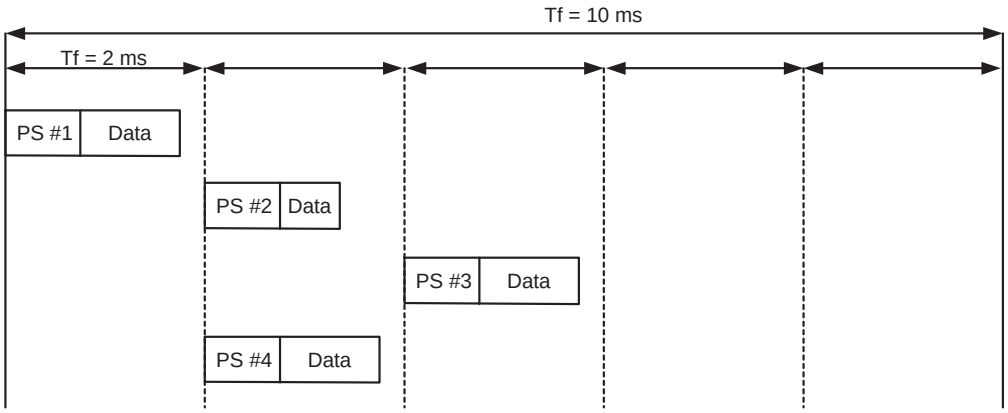

Figure 7: Random Access Burst Scheduling

unique spreading codes used for the access channel is increased to 256, and the burst duration is conjectured to be 3.33 ms; a mobile station then selects a random time of $3.33n$ ms, $n = 0, 1, 2$ as offset to a frame boundary. With that modification, 256x3 stations may transmit within a 10 ms frame[4].

[4]The frame durations were suggested by Beamreach Networks, Inc. for the performance analysis

3 Literature Research

Studies performed in the last couple of years have presented convincing evidence that network traffic often cannot be accurately described by standard Poisson models, implying that a number of existing traffic models lead to false predictions. In the following, the major results in this field of research are presented.

- First, papers examining the traffic structures occurring in communication networks are summarized.
- After that, different studies pointing out either the importance of self-similarity to network performance or the irrelevance of the need for capturing self-similarity in traffic modeling are discussed.
- Finally, an outline of different existing traffic models is given.

3.1 Traffic Structures in Communication Networks

Prior to 1993, network traffic was exclusively modeled using Poisson processes, which were well understood and capable of precisely describing traffic patterns originating from conventional telephone systems. In the paper "On The Self-Similar Nature of Ethernet Traffic" [LWTW93], Leland et al. demonstrate that local area network traffic can show fractional behavior, which is different from both, conventional telephone traffic and from Poisson related packet traffic models used at this time. The main difference is that Poisson traffic becomes smoother (less bursty) as the number of sources increases, whereas this is not the case for the traffic traces Leland et al. observe: The aggregated traffic stream remains bursty even for a large number of superimposed sources. Moreover, the traffic streams look similar when viewed at different time scales, as shown in

Figure 8[5]. A behavior like this indicates self-similar traffic patterns in the underlying processes.

To support his hypothesis, Leland recorded traffic traces on various local area networks. These measurements have a resolution of 100 μsec spanning 136 hours and capturing more than 40 TB of Data.

For the collected data the degree of self-similarity was measured using an R/S plot, a variance-time plot, and utilizing Whittles approximate Maximum Likelihood estimation, as described in Section 2.2. With a confidence of 95 percent the Hurst parameter H is found to lay within 0.8 and 0.95.

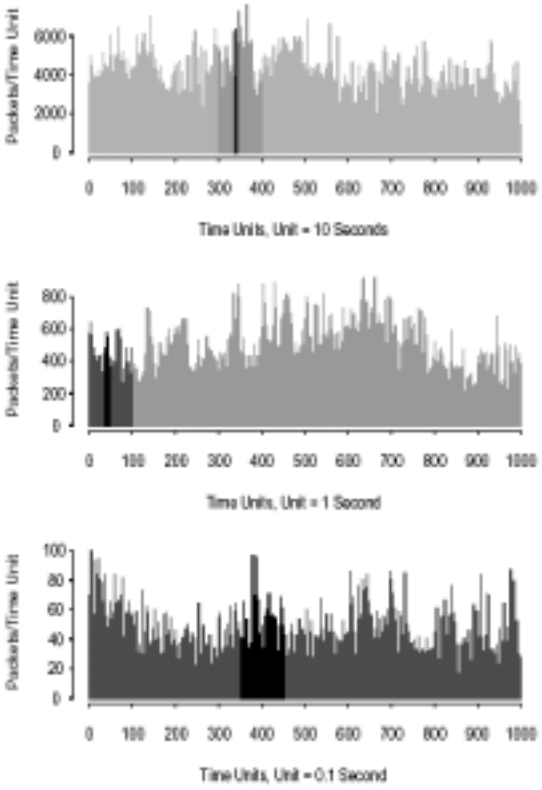

Figure 8: Pictorial Proof for Self-Similarity

Leland et al. conclude that the inability of recent traffic models to capture network traffic's self-similar behavior is the reason for obtaining different packet delays and packet loss probabilities between trace driven measurements and those based on conventional traffic models. The results presented in this paper led to a number of other publications in this field.

Another significant work in this area was published under the title "Explaining World Wide Webs Traffic Self-Similarity" by Crovella and Bestravos [CB95] in 1995. The authors examined WWW traffic with similar techniques Leland et al. used for local area

[5]This graph is taken from [LWTW93], Figure 1(b),1(c),1(d)

network traffic.

Crovella collect client based traces of HTTP traffic on the application level using a Mosaic browser which was modified to log all users access to the Web. The size of each file accessed, including the protocol overhead as well as the time required to transfer the file from a server is recorded. Files contained in Mosaic's cache are excluded from the measurement since only transferred files were of interest. Overall, the traces contain approximately 600 users and capture the transfer volume of 2 TB of data. Timestamps are accurate to 10 msec. To convert the traffic trace to a time series, the bytes transferred in each request are allocated equally into bins spanning the transfer duration, using a granularity of 1 sec. This time series is used to show the burstyness of the trace at varying time scales, indicating self-similar traffic patterns. The Hurst parameter H was estimated with the techniques as described in Section 2.5.1 of this thesis. With a confidence of 95 percent H is determined to lay within the interval (0.77, 0.87) for the examined traffic trace.

To explain the fractional structure of Web traffic, the authors use a method of constructing self-similar processes described by Mandelbrot, stating that a self-similar process can be constructed by superimposing many simple renewal processes, in which the inter-renewal times are drawn from a heavy tailed distribution. The connection between Mandelbrot's result and the traffic structure is the following: A traffic source can be approximated as an ON/OFF process which is in turn a simple renewal process. The essential heavy "taildness" of the inter-renewal times is found to result from heavy tailed "file transmission times" (due to the distribution of file sizes) and from heavy tailed "silent times" (introduced by user behavior). Both, transmission times and silent times are examined in greater detail, and found to be heavy tailed; their complementary cumulative distribution functions are given in Figure 9[6]

[6]This graph is taken from [CB95], Figure 8(b) and Figure 13

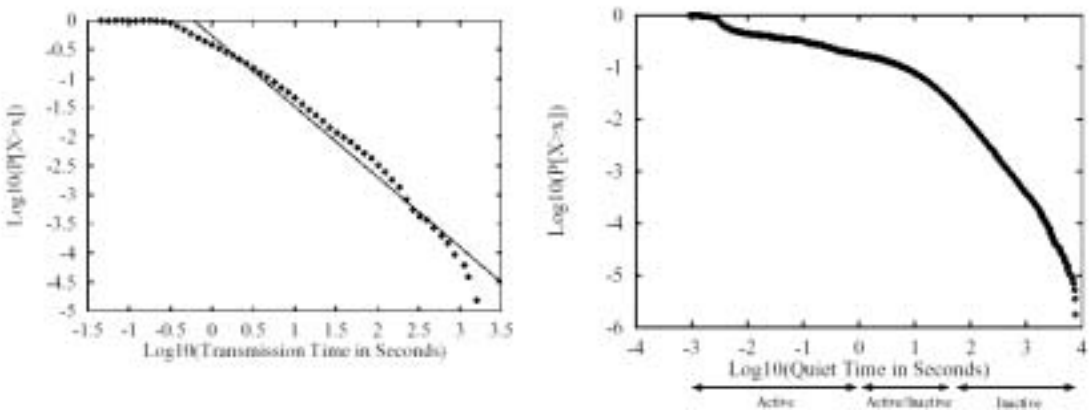

Figure 9: Distribution of Transmission and Silent Times

The transmission time is found to be much heavier tailed than the silent time; for that reason Leland et al. assume that the file sizes are a major reason for World Wide Web traffic's self-similarity. This conclusion is augmented in the work of Taqqu, which shows that the value of the Hurst parameter H is determined by whichever distribution is heavier tailed. Related to this work, Park, Kim and Crovella showed that it is sufficient to draw file sizes from a heavy tailed distribution to generate self-similar network traffic.

Other papers using traffic traces from physical network measurements to identify scale invariant burstyness and proposing models capable of generating synthetic traffic with matching characteristics are:

- [GW94] and [HDLW95] for variable bit rate video traffic,
- [PF94] for wide area traffic and
- [WTSW95] for local area network traffic on the source level.

3.2 Self-Similarity: Important or Irrelevant?

The presence of self-similarity in network traffic is widely accepted. The question which arises now is how prevalent such traffic patterns are in network performance studies.

Beyond researchers, two different opinions are common: One is that nearly all existing models have to be rewritten using completely different approaches, the other is that more sophisticated but conventional models are sufficient for traffic modeling and performance analysis. In the following, reasons for both points of view are summarized.

The papers discussed so far, especially [CB95] and [LWTW93], point out that conventional models are not sufficient for performance analysis. [LWTW93] mention that packet loss and delay behavior differ between trace driven simulations based on their traffic measurement and those based on former traffic models. However, Leland et al. do not specify the extent of discrepancy. In [CB95] Crovella and Bestravos suggest the use of multiple ON/OFF sources with long tailed renewal times for modeling applications at the source level, but also without quantifying the mismatch of conventional models.

Numerical results are given in "On The Effect of Traffic Self-Similarity on Network Performance" by Park et al. [PKC97]. The authors examine the influence of long range dependent traffic on different transport layer protocols. The connectionless Unified Datagram Protocol (UDP) and the connection oriented Transfer Control Protocol (TCP) with its implementations Tahoe, Reno and Vegas are evaluated.

To generate synthetic self-similar network traffic, the ON/OFF model proposed by Crovella and Bestravos is used as an input for the network simulator ns [LXP+95]. The results obtained indicate that measures such as mean queue length, packet loss and delay depend on the degree of self-similarity. Figure 10[7] relates the mean queue length for different link buffer sizes to α, which is in turn related to the Hurst parameter $H = (3 - \alpha)/2$. The Protocol under study in this Figure is TCP Reno, the offered average throughput is held constant and the link bandwidth is set to 1.5Mbps.

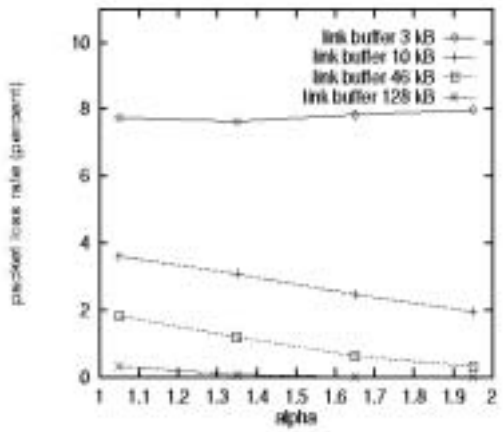

Figure 10: Influence of Self-Similarity on Packet Loss Rates

For small link buffers (3 kbyte) H has no significant influence on the mean queue length, whereas for large link buffers (128 kbyte) the mean queue length is by factor 7 higher for $H \approx 1$ than for $H \approx 0.5$. Figure 11[8] shows the packet loss ratio over α, again for constant average offered traffic and TCP Reno. Again the influence of α (or $H = (3 - \alpha)/2$) depends on the link buffer size: TCP implementations with small buffers (3kbyte) are invariant against the Hurst parameter, whereas the packet loss rate increases with H for large link buffers (128kbyte).

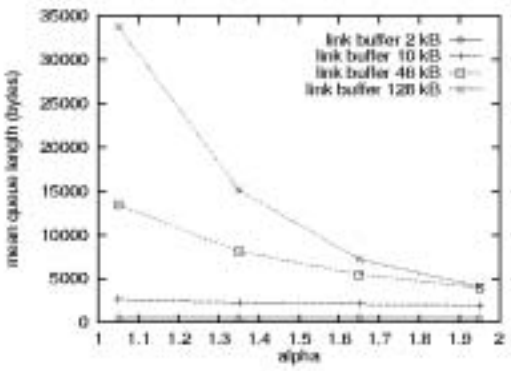

Figure 11: Influence of Self-Similarity on Mean Queue Length

The comparison of different TCP implementations Tahoe, Reno and Vegas shows

[7]This graph is taken from [PKC97], Figure 18
[8]This graph is taken from [PKC97], Figure 20

that even under highly self-similar traffic conditions the performance gap is preserved through the three congestion control algorithms. The performance gain from Tahoe to Reno is relatively minor while the performance gain from Reno to Vegas is more pronounced, as it is under non long range dependent traffic.

In [ST99] Sahinoglu and Takinay present a survey on the self-similar phenomenon observed in multimedia traffic and its implications on network performance. The authors concentrate on Quality of Service (QoS) in the network layer.

It is said that, in the presence of self-similar traffic, buffers needed at switches and multiplexers to obtain a desired overflow or cell loss probability must be bigger than those predicted by traditional queuing analysis, which is in accordance to the results described in [PKC97]; larger buffers lead to greater delays, which are an important measure for real time applications such as video conferencing and Internet telephony.

Sahinoglu and Takinay point out that, for self-similar multimedia traffic, increasing the buffer size is far less effective for decreasing the cell or packet loss probability: With non long range dependent traffic, a linear increase in buffer size will produce nearly exponential decrease in packet loss. For self-similar traffic they find that these assumptions do not hold. The decrease in packet loss with buffer size is far less than expect. Figure 12[9] shows the relationship between queue size and utilization for different degrees of self-similarity. For H = 0.5, representing traffic which shows no long range dependence, the queue size grows exponentially with the utilization. For H = 0.9, which is approximately the Hurst parameter Leland et al. [LWTW93] found in Ethernet traffic, the mean queue size is approximately the same as for H = 0.5 for a utilization smaller than 35 percent, but for higher utilizations the degree of self-similarity has major influence on the queue size.

[9]This graph is taken from [ST99], Figure 2

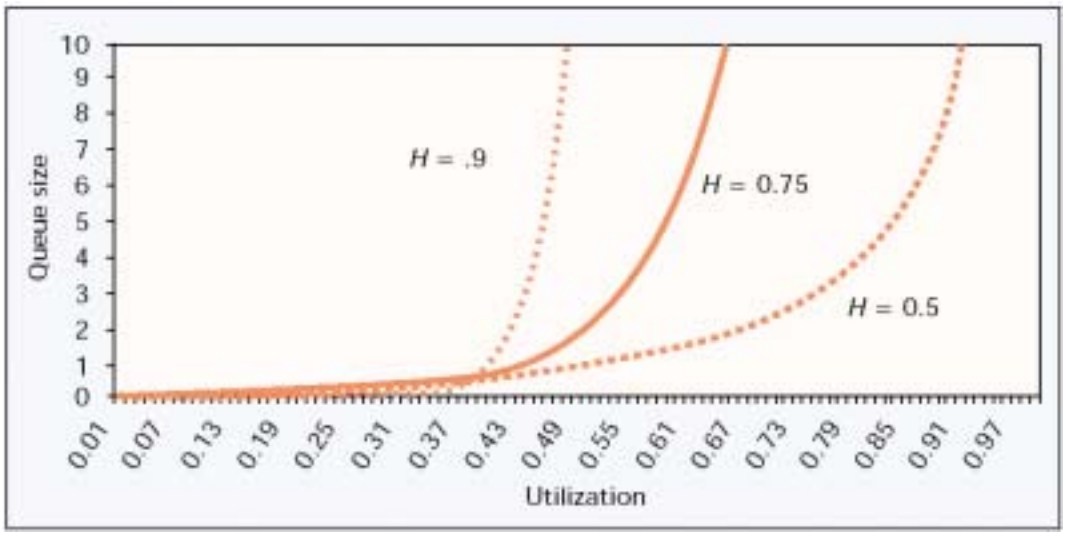

Figure 12: Queue Size over Utilization for Different Hurst Parameters

In accordance to [PKC97], Sahinoglu and Takinay use this plot to support their thesis that network performance, as captured by throughput, packet loss rate and packet retransmissions under certain conditions degrades gradually with degree of long range dependency. To possibly overcome this performance loss, the authors conclude that a tool capable of detecting self-similarity in real time is necessary to optimize adaptive resource allocation techniques, but without suggesting a certain approach.

Contrary opinions are stated in the following:

In [RE96] Ryun and Elwaid defend their thesis that it is unnecessary to capture the long term correlations of a real variable bit rate (VBR) video source under "realistic ATM buffer dimensioning scenarios" [RE96], as far as cell loss rates and delays are concerned as quality of service metrics. The authors define realistic buffer dimensioning as buffers with a capacity of approximately 20 msec to 30 msec times the average data rate of a single source and a utilization chosen so that the cell loss rate is less than 10^{-6}. A simulation is used to show that, for the parameters they have chosen, even in the presence of self-similar traffic long range correlations do not have a significant impact on

the cell loss rate. Moreover they show that simple Markov models, which capture the short term correlations, provide good approximations of cell loss rates.

In accordance to Ryun et al. Heyman and Laksman [HL96] show that even for sources with a large Hurst parameter, Markov chains could be used to estimate buffer occupancy when the buffers sizes were no larger than 10 ms for a single source. Lin and Suda [LS98] show that for ATM frame assembly operations, were also small per flow buffers are used, both, the model capturing long range dependency and a conventional Markov model shows similar queueing behavior.

3.3 Include Long Range Dependency in a Network Model?

There seems to exist no single answer to this question; however, the following conclusion can be drawn from the discussion above:

- Self-similar traffic is needed for a precise performance analysis if the buffers in the system are big, or if the utilization is higher than 35 percent. (35 percent is a rule of thumb taken from [ST99])
- Self-similar traffic patterns can be neglected in case of a Hurst parameter smaller than H = 0.7, in case of small buffers for a given source bandwidth, or when the utilization of a system is low.

However, researchers who question the usage of conventional approaches tend to use rather simple Poisson models to support their thesis. We will justify the use of extended Markovian models in Section 7.2 where we show that they can be used to generate traffic which is self similar over a large time scale.

3.4 Existing Traffic Models

To reduce the design complexity, most communication networks are hirachically organized using a layer concept. A layer has defined interfaces, and offers a number of services to the layer above. An Example is the TCP/IP reference model, illustrated in Figure 13:

The layer on top is the **application layer**. It contains the higher level protocols such as the File Transfer Protocol (FTP), the Simple Mail Transfer Protocol (SMTP) and the Hypertext Text Transfer Protocol (HTTP). The **transport layer** offers a reliable end-to-end connection using the transmission control protocol (TCP), embedding flow control, connection management and congestion control. The second protocol in the **transport layer** is the User Datagram Protocol (UDP), which offers a unreliable and connectionless service.

The **Internet layer** defines a packet format and protocol called IP (Internet Protocol). Features such as packet routing and congestion avoidance are implemented in this layer. Not defined in greater detail within the TCP/IP reference is the **Host-to-Network layer**. It has to be able to inject and or to accept IP packets to or from the layer above.

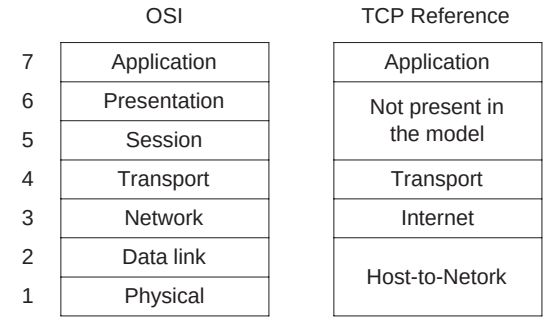

Figure 13: ISO OSI and TCP Reference model

Due to the complexity of a communication network, it makes sense to use the layered structure not only for designing but also for modeling a network. In the following, several examples of models dealing with mechanisms on different layers are introduced.

3.4.1 Modeling the Application Layer

An empirical model for traffic originating from Web browsers is proposed by Deng [Den96]. In this work, a simple ON/OFF model is developed, with the ON state describing a user who is downloading content from the Internet and no traffic generated while in the OFF state.

A traffic trace spanning 4.5 hours and containing 293 active users, each connected to a service provider with a T1 line, is used to derive the random variables describing the duration of the ON and OFF states as well as the interarrival times, i.e. the time difference between two consecutive document downloads.

An ON period is considered a sequence of arrivals with less than 60 seconds between any two events. If arrival A is separated by more than 60 seconds form arrival B, then A marks the beginning of an OFF period. Two OFF periods divided by a single event are considered a single OFF period. Although the 60 second time period is chosen heuristically, the resulting distributions are, referring to Deng, insensitive to time periods on the order of 30 to 120 seconds. Deng found that the ON period and the interarrival time of documents can be described by a Weibull distribution, while the OFF period can be modeled by a Pareto distribution. Sizes of the transferred files are not examined but taken from [CB95]. The distributions and their parameter are summarized in Table 3.4.1.

With these parameters, both, ON and OFF times are heavy tailed. Problems arise due to the infinite mean and variance of the OFF distribution: It is, for example, not possible to derive the average time spent in the ON state.

A more detailed model for Web traffic is introduced by Choi and Limb in[CL99]; its basic structure is illustrated in Figure 14 [10]: A Web request occurs when a user clicks

[10]The figure is taken from [CL99]

Period	Distribution	Model (PDF)	Model (CDF)
ON period	Weibull	$p(x) = \alpha\beta x^{\beta-1}e^{-\alpha x^{\beta}}$ $\beta = 0.88, \alpha = e^{-4.5\beta}$	$F(x) = 1 - e^{-\alpha x^{\beta}}$
Interarrival	Weibull	$p(x) = \alpha\beta x^{\beta-1}e^{-\alpha x^{\beta}}$ $\beta = 0.5, \alpha = e^{-1.5\beta}$	$F(x) = 1 - e^{-\alpha x^{\beta}}$
OFF period	Pareto	$p(x) = \frac{\beta\alpha^{\beta}}{x^{\beta+1}}$ $\beta = 0.5, \alpha = 60$	$F(x) = 1 - (\alpha/x)^{\beta}$

Table 1: Parameter of Deng's Empirical Model

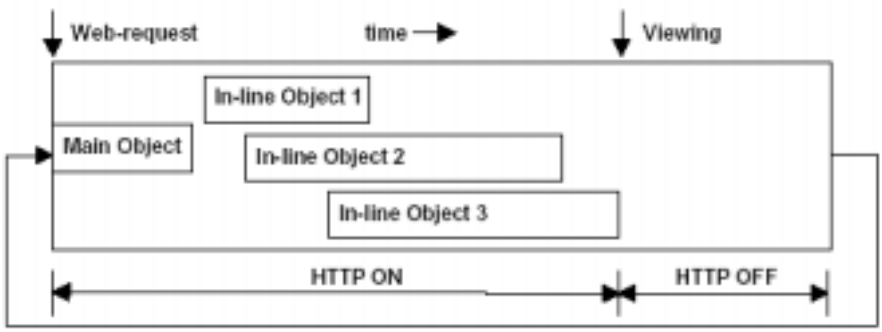

Figure 14: Overview of Choi's Basic HTML Traffic Model

on a link or enters a URL [BLMM94]. The first object transferred is referred to as main object, most likely a HTML document containing references for pictures or a java script applet. These documents, whose download is initiated by the browser, are called in-line objects, each using its own TCP connection, which can be either back-to-back or in parallel, depending on the HTML protocol version. The authors utilize a traffic trace containing HTML and TCP header information to derive the statistical parameters used to describe the model, such as main and in-line object sizes and interarrival times, parsing time, page viewing time and number of Web requests. The state transition diagram for Web traffic generation is given in Figure 15 [11]: When a new Web request is initiated

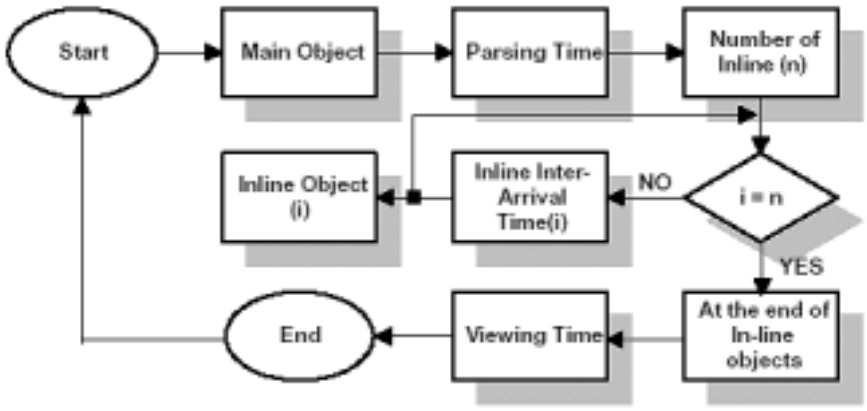

Figure 15: Choi's State Transition Diagram for Web Traffic Generation

by the user, the main object is loaded, containing the information for the browser on how to proceed (e.g. from where to load the inline objects). After the parsing time (the time needed by the browser to process the main object), the inline objects are loaded. When all downloads are completed, the user usually works with the page for a certain time, introduced as viewing time. The process starts again with a new Web request. Referring to Choi and Limb, the model generates traffic which closely matches traffic patterns measured on a real network.

[11]The figure is taken from [CL99]

3.4.2 Modeling at the Transport and Internet Layer

For the Transport layer a large number of models exist, describing connection arrivals or performance of features such as routing and congestion control. [GCMM01] uses complex queueing theory to derive approximations for network performances such as latency and TCP packet loss probabilities. However, the underlying Internet layer is only roughly approximated by M/M/1 queues, providing a good example how a layered network structure can be used to simplify a modeling approach.

An analytic model of wide area TCP connections for telnet, nntp, snmp and ftp traffic is given by Paxon [Pax93]. Referring to Paxon, ideally, an analytical model has few bound parameters, which makes it easy to understand, and no free parameters, which makes it predictive. However, most analytical models also have free scale or offset parameters which have to be estimated before the model can be used to derive numerical results.

The author used a traffic trace containing 2.5 million TCP connections to obtain the parameter for his model. A complete description of the random variables associated with the wide area connections is provided in [Pax93], as well as a comparison with empirically derived models, showing that the analytical approach describes connection interarrivals well and provides a good estimation of the amount of data transferred.

Very many other traffic models exist, all concentrating on different features at different levels of detail. A comprehensive summary of several approaches for traffic modeling is provided in [WTE97] by Willinger and Taqqu.

4 Approximating Long Tailed Distributions with Hyperexponentials

This chapter deals with the approximation of long tailed distributions, which are often used in performance evaluations, with a large number of exponential stages. It is organized as follows:
First, reasons for the usefulness of exponential approximations are given, and the structure of the exponential fit is explained. Secondly, an algorithm capable of fitting a group of exponential stages to a large class of distributions is introduced. Thirdly, the goodness of fit is examined and the drawbacks of the approach are summarized. In the last part of the chapter, the algorithm is extended such that it becomes applicable for a larger class of Pareto distributions.

4.1 Why Approximate Long Tailed Distributions with Hyperexponentials?

The reason why exponential approximations of long tailed time interval distributions are useful is simple: Performance models tend to be easier to analyze when phase type distributions are used. In some cases, an exponential approximation makes it possible to solve queueing models that could not be solved before, for example by using Markovian statistics. Another advantage is that convenient Laplace transforms exist for the approximation, which is not the case for most long tailed random processes.

How does one put single exponential wait stages together to mimic long tailed behavior?
The CDF of long tailed distributions for large t decays slower than the tail of a single exponential. The same is true for a Hyperexponential distribution, as introduced

in Section 2.1.1, which makes them suitable for an approximation. In a more mathematical sense, referring to [FW97]: The class of probability distributions with a completely monotone pdf can be approximated arbitrary closely with Hyperexponentials. A function is completely monotone if all derivatives exist and the following inequality is satisfied:

$$(-1)^n f^{(n)}(t) \geq 0, \quad t > 0 \ , \ n \geq 1. \tag{30}$$

The following statement is referred to as Bernstein Theorem:

> If F is a CDF with a completely monotone pdf, then there are Hyperexponential CDFs with n stages, $F^{(n)}, n \geq 1$, i.e. CDFs of the form
>
> $$F^{(n)}(t) = \sum_{i=1}^{k_n} p_{ni}(1 - e^{-\lambda_{ni} t}), \quad t \geq 0 \tag{31}$$
>
> with $\lambda_{ni} \leq \infty$ and $p_{n1} + p_{n2} + \ldots + p_{nk_n} = 1$, such that $F^{(n)} \to F$ as $n \to \infty$.

An example for random variables which are completely monotone are Weibull distributions, which are examined now.

4.2 Deriving the Parameters of the Exponential Fit

The following algorithm was first described by de Prony in 1795 [dP95] and recently by Feldmann and Whitt [FW97]. After introducing the basic procedure, we extend the algorithm so that it can be used for a larger class of Pareto distributions. Its implementation in Matlab is given in Appendix A.1.1.

Definitions

Let $F^c(t)$ be the complementary cumulative density function, CCDF, $F^c = 1 - F(t)$ of

a completely monotone pdf which is to be approximated, and let

$$h(t) = \sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i t} \tag{32}$$

be the pdf of the Hyperexponential distribution; k is the approximation order (the number of exponential wait states), p_i and λ_i, $i = 1 \ldots k$ are the parameters which are to be adjusted to achieve the best fit.

A region of interest, i.e. the time interval which is important for later analysis, is chosen as $[T_{min},\ T_{max}]$. Then k points $c_1, \ldots, c_k$ are selected for which the fitting algorithm will compute parameters such that the Hyperexponential CCDF matches the CCDF of the given distribution on this points. They are chosen according to Equation 33 to 36 such that they are spaced equally at a logarithmic scale.

$$c_k = T_{min}, \tag{33}$$

$$c_1 b = T_{max}, \tag{34}$$

$$c_i = c_1 \tilde{x}^{-(i-1)}, \quad k = 2 \ldots k-1 \tag{35}$$

with

$$\tilde{x} = e^{\frac{ln(\frac{c_k}{c_1})}{1-k}}. \tag{36}$$

The free parameter b, $b < \frac{c_i}{c_{i+1}}$ for all i, can be used to adjust the mean of the exponential fit to the expected value of the original function.

The Fitting Algorithm

Having defined the parameters, the fitting routine can be introduced. The pair (λ_i, p_i)

are, beginning with $i = 1$ and ending with $i = k - 1$, chosen such that

$$p_i e^{-\lambda i t} = F_i^c(t) \quad \text{for } t = c_i \tag{37}$$

and

$$p_i e^{-\lambda i t} = F_i^c(t) \quad \text{for } t = bc_i \ , \tag{38}$$

where $F_i^c(t)$ is the given CCDF minus the already computed parts of the Hyperexponential fit, if there is any, i.e.

$$F_i^c(t) = F^c(t) - \sum_{j=1}^{i-1} p_j e^{-\lambda_j t_i} . \tag{39}$$

The last weight factor p_k is determined by the relationship $\sum_{i=1}^{k} p_i = 1$ after $p_i, i = 1, \ldots, k-1$ have been found. λ_k is chosen so that

$$p_k e^{-\lambda_k t} = F_k^c(t) \quad \text{for } t = c_k \ . \tag{40}$$

This can be done for several b until the mean is sufficiently approximated.

As an example, a Weibull distribution with $a = e^{4.5} = 90$ and $b = 0.5$[12], leading to a CCDF $F^c(t) = e^{-(\frac{t}{90})^{0.5}}$, is approximated with a 3 stage Hyperexponential (k=3). The region of interest is assumed to be the interval from $T_{min} = 2$ to $T_{max} = 2500$ seconds, the free parameter b is set to b = 5. With Equation 34 and Equation 33, one gets $c_3 = 2$ and $c_1 = 500$. Using Equation 35 and Equation 36, $\tilde{x} = 15.81$ and $c_2 = 31.62$ is obtained. Now the fitting procedure can be used: The first step (i=1) is to

[12]These parameters are taken from Deng's model

solve Equation 37 and Equation 38 for p and λ, leading to

$$\lambda_i = \frac{1}{(b-1)c_i} ln(F_i^c / F_i^{bc}) \tag{41}$$

$$p_i = F_i^{c_i} e^{\lambda_i c_i} \tag{42}$$

With Equation 37 and $i = 1$, F_1^t is just the CCDF of the given function, $F^c(t) = e^{-(\frac{t}{90})^{0.5}}$, leading to $\lambda_1 = 0.001547$ and $p_1 = 0.1962$. For $i = 2$, one can proceed as before, except that $F_2^t = F^c(t) - p_1 e^{-\lambda_1 t}$; $\lambda_2 = 0.009502$ and $p_2 = 0.4935$. Now the last tuple (λ_3, p_3) can be determined: $p_3 = 0.3103$ is found using Equation 42; with Equation 41, where $F_2^t = F^c(t) - p_1 e^{-\lambda_1 t} - p_2 e^{-\lambda_2 t}$, $\lambda 3 = 0.2677028$ is obtained. The given Weibull distribution and the Hyperexponential fit are compared in Figure 16; for only 3 exponential stages, the obtained approximation is not very accurate. However,

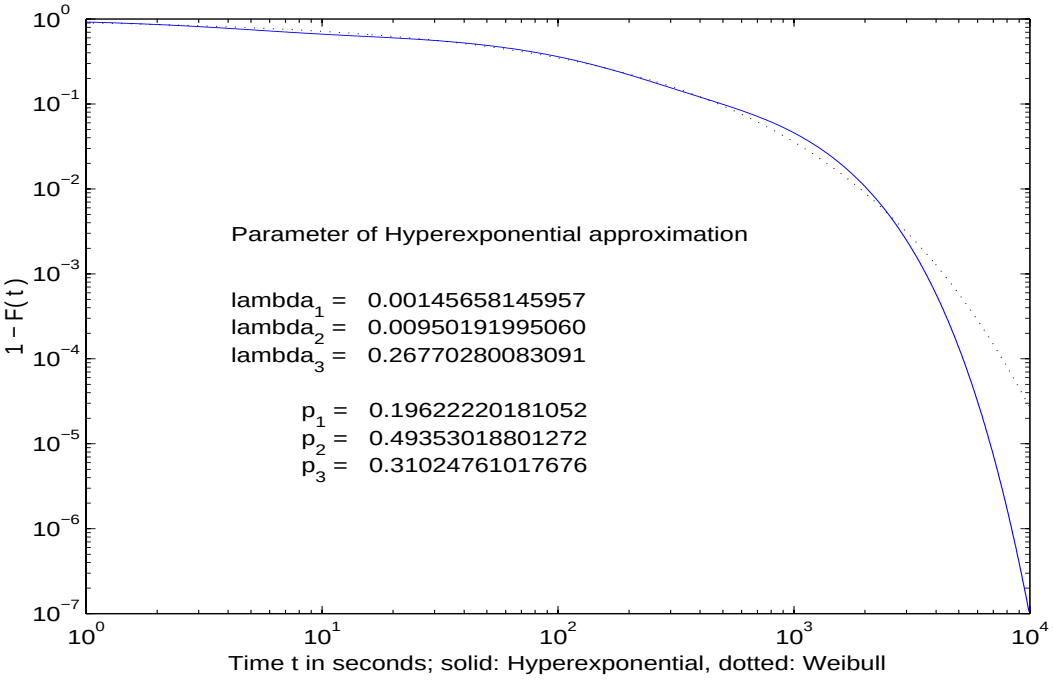

Figure 16: 3-Stage Hyperexponential Fit to Weibull(exp(4.5),0.5)

with a larger number of stages, it is possible to approximate a long tailed distribution

very closely. Figure 17 and Figure 18 show the CCDF and the CDF of a 16-stage Hyperexponential fit to a Weibull distribution with $a = 245$ and $b = 0.75$. The relative error for both, CCDF and CDF, is below 3 percent within the large interval $t = 5 * 10^{-4}$ to $3 * 10^{-3}$ sec. This accuracy should be sufficient for most applications.

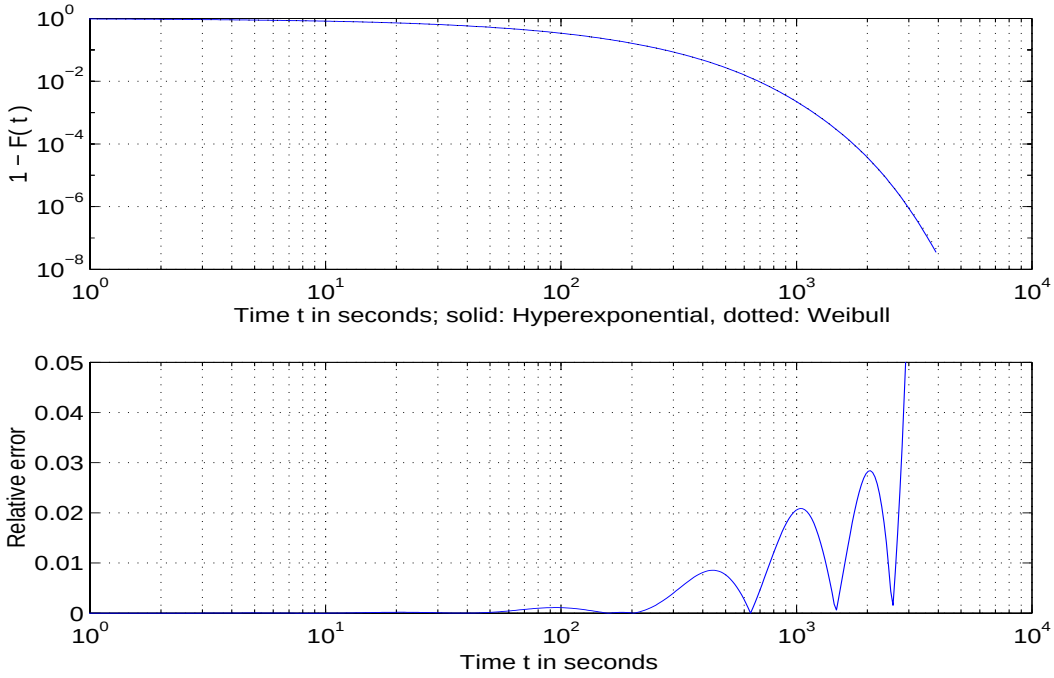

Figure 17: 16-stage Hyperexponential Fit to Weibull(exp(5.5),0.75), CCDF

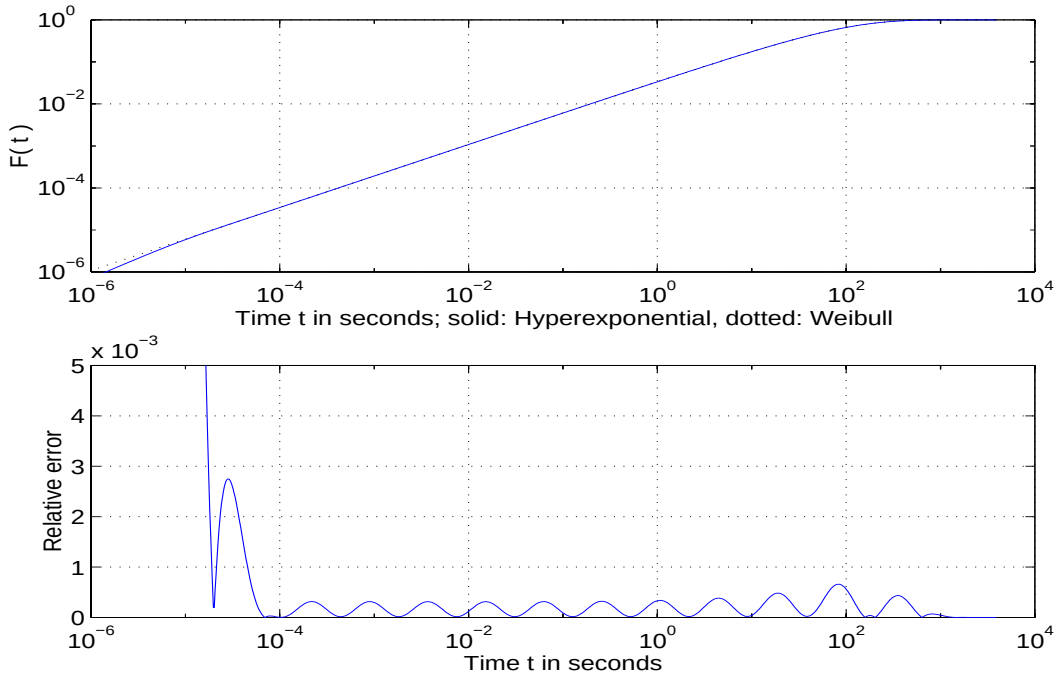

Figure 18: 16-stage Hyperexponential Fit to Weibull(exp(5.5),0.75), CDF

Transition rates	Weightfactors
λ_{01} = 0.00000001832904*1.0e+5	p_{01} = 0.00000000006819
λ_{02} = 0.00000002613571*1.0e+5	p_{02} = 0.00031139594117
λ_{03} = 0.00000003800343*1.0e+5	p_{03} = 0.06823208434681
λ_{04} = 0.00000006377714*1.0e+5	p_{04} = 0.40150703463163
λ_{05} = 0.00000014956361*1.0e+5	p_{05} = 0.35479903100208
λ_{06} = 0.00000052083900*1.0e+5	p_{06} = 0.12232600606005
λ_{07} = 0.00000217207189*1.0e+5	p_{07} = 0.03596635460883
λ_{08} = 0.00000923601009*1.0e+5	p_{08} = 0.01123820157313
λ_{09} = 0.00003876731793*1.0e+5	p_{09} = 0.00370485880446
λ_{10} = 0.00016096097697*1.0e+5	p_{10} = 0.00125604354496
λ_{11} = 0.00066450851037*1.0e+5	p_{11} = 0.00043133215001
λ_{12} = 0.00273719956837*1.0e+5	p_{12} = 0.00014891554052
λ_{13} = 0.01127039192866*1.0e+5	p_{13} = 0.00005150636801
λ_{14} = 0.04642587104368*1.0e+5	p_{14} = 0.00001781920947
λ_{15} = 0.19137571400107*1.0e+5	p_{15} = 0.00000616238589
λ_{16} = 1.00129050187363*1.0e+5	p_{16} = 0.00000325376479

Table 2: Parameter of 16-Stage Hyperexponential Fit to Weibull(exp(5.5),0.75)

4.3 Describing Pareto Distributions with Exponentials

A drawback of the Hyperexponential approximation is that it is not applicable to general Pareto distributions, which have a pdf of the form

$$f(t) = \begin{cases} \alpha k(\frac{t}{k})^{-\alpha-1}; & \text{if } t > k \\ 0 & \text{otherwise.} \end{cases}$$

At t=k, the derivative of f(t) does not exist and therefore the Bernstein Theorem does not hold. However, a Hyperexponential approximation does exist for the shifted version

$$\tilde{f}(t) = f(t+v) \;\; t > 0 \tag{43}$$

For $\tilde{f}(t)$ the fitting routine can be used as described before; to obtain the approximation of $f(t)$, the time shift is reversed by combining the Hyperexponential fit of $\tilde{f}(t)$ with a deterministic wait state. But how does one obtain an approximation for the non-shifted version by using only exponential stages? The answer is to approximate the wait state with an Erlang-K distribution. The state transition diagram is given in Figure 20. λ_{Erlang} is chosen so that the expectation value of the added stage equals the time v the

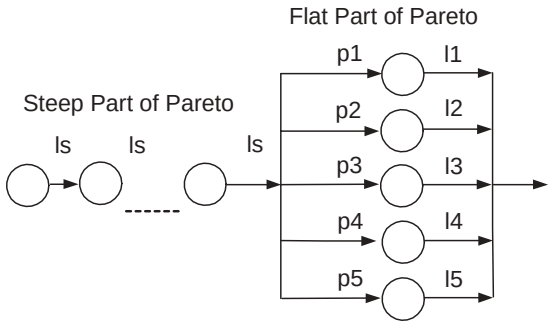

Figure 19: State Transition Diagram for an Approximation of a Pareto Distribution

original function was sifted with. This leads to $\lambda_{Erlang} = \frac{K}{v}$. For $K \to \infty$, Erlang-K distributions behave like deterministic wait states. In the model which is introduced

later, $K = 200$ was empirically chosen; with a probability of 90 percent the waiting time lies within 5 percent around the mean.

An example to demonstrate the quality of the overall approximation is given now, where an Erlang-K distribution with 200 stages together with an Hyperexponential consisting of 10 stages is fit to an Pareto distribution with $\alpha = 1.5$ and $\beta = 60$. In Figure 20 the CDF and the CCDF of the original function and approximation is given. The CDF plot shows the behavior for small t, whereas the CCDF shows the goodness of fit at the tails. For small t, a better approximation would have been achieved if the number of exponential states of the Erlang-K distributions had been chosen to be larger, for example 500 instead of 200, but for a larger number of states one encounters numericall difficulties when computing the convolution to obtain the probability density function which is necessary to generate the plot.

The tail behavior of the fit approximates the original Pareto distribution closely, as illustrated in the CCDF plot. From $t_1 = 10^3$ till $t_2 = 3 * 10^6$ both curves are hard to distinguish. For larger t, the asymptotic behavior of the fit becomes those of a short tailed distribution. However, if the number of exponential stages in the Hyperexponential part of the fit is increased, for example from 10 to 20 which would not have a significant impact on the mathematical complexity of the solution, the upper bound t_2 can be made larger if necessary. The parameter of the fit are given in Tabel 4.3; λ_i and p_i for $i = 1 \ldots 10$ are computed with the algorithm as described in Section 4.2, K_{er} was empirically chosen, and λ_{er} was computed with Equation 7.

The Hyperexponential fitting procedure can very accurately approximate the tails of given slow decaying random processes. When a phase type distribution is needed, one would first match a long tailed distribution to the data, what is easily done due to the small parameter space of such a distribution, and then apply the fitting routine to it. If the underlying statistic of the data is truly those of a long tailed distribution -for example

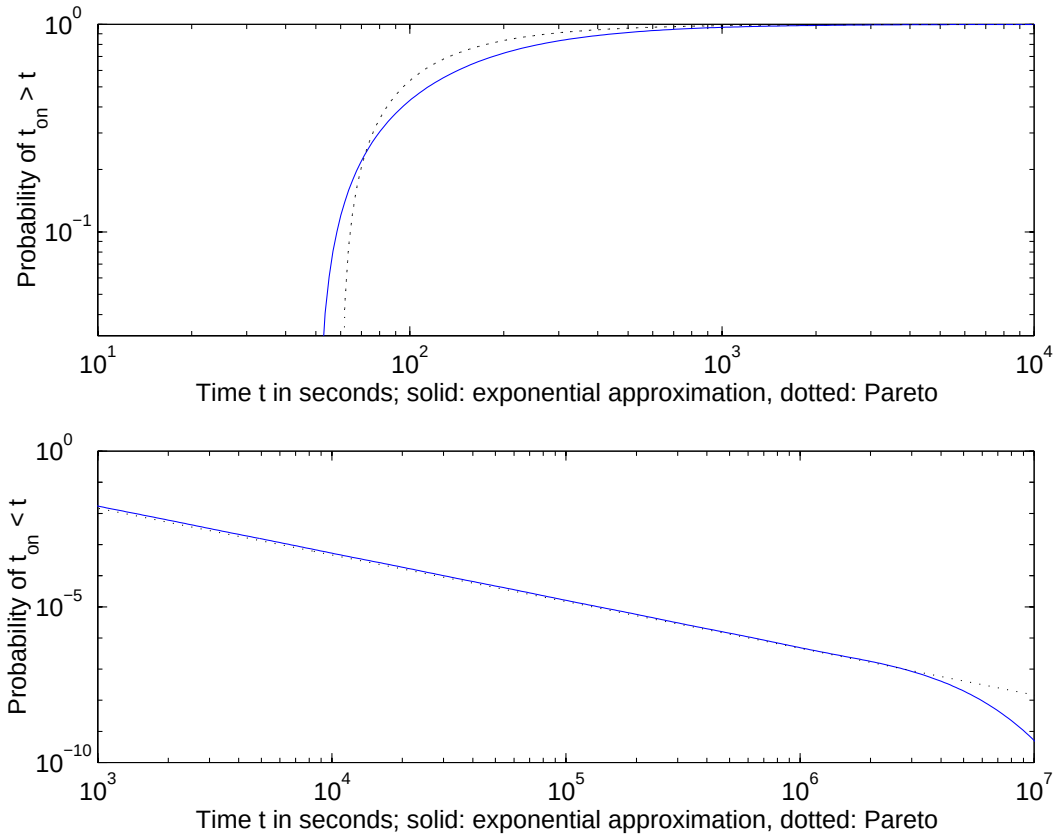

Figure 20: 10-stage Hyperexponential and 100 stage Erlang-K Fit to Pareto(1.5,60), CDF (top) and CCDF (bottom)

Transition rates	Weightfactors
λ_{er} = 0.3	K_{er} = 200
λ_{01} = 0.00000074257647	p_{01} = 0.00000072553115
λ_{02} = 0.00000393676652	p_{02} = 0.00000567131482
λ_{03} = 0.00001425439220	p_{03} = 0.00003439450937
λ_{04} = 0.00004706387335	p_{04} = 0.00019822230376
λ_{05} = 0.00015126708993	p_{05} = 0.00112123759216
λ_{06} = 0.00048142934558	p_{06} = 0.00620495447972
λ_{07} = 0.00151932541277	p_{07} = 0.03241586287740
λ_{08} = 0.00470791523456	p_{08} = 0.14288407402354
λ_{09} = 0.01398707060832	p_{09} = 0.39768178349492
λ_{10} = 0.04044703155438	p_{10} = 0.41945307387316

Table 3: Parameter of Markovian Fit to Pareto(1.5,60)

Pareto or Weibull- this method provides good results. However, if this is not the case, what is very likely because complex technical processes are rarely fully described by only two parameters, the first fit will not precisely describe the data resulting in an inaccurate Hyperexponential fit. In other words, in such a case the algorithm cannot utilize the great flexibility of the large number of exponential stages with its large parameter space. The method we have developed provides this flexibility. It is introduced in the next chapter.

5 Fitting Data to Parallel Erlang-K Distributions

In this chapter, a routine is introduced to fit various phase type distributions to random processes. It is possible to find the parameters of arbitrarily grouped exponentials, for example Hyperexponentials, Hypoexponentials or combinations of both, as long as the probability density function can be written down in closed form. However, we obtained the best results by using parallel Erlang-K distributions.

In the following, the properties of parallel Erlang-K distributions are summarized, a Maximum Likelihood approach to find the parameters of the fit is introduced, and the optimization problem and its Matlab implementation is explained. Equipped with the new tool, an example is given to examine the quality of the fitting procedure.

5.1 Parallel Erlang-K Random Processes

We define the weighted sum of parallel Erlang-K distributions as parallel Erlang-K process, where a single exponential state is contained as a special case. An example for a state transition diagram is illustrated in Figure 21: A token which enters the process

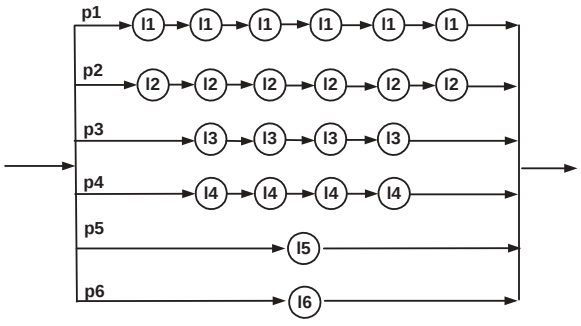

Figure 21: Structure of Parallel Erlang-K Stages

selects branch j with probability p_j and then passes through N_j serial exponential wait states with a homogeneous waiting time λ_j. Such a system has a probability density

function of a the form

$$f(t) = \sum_{i=1}^{m} p_i \frac{(\lambda_i t)^{N_i - 1}}{(N_i - 1)!} \lambda_i \, e^{-\lambda_i t}, \quad \lambda_i > 0, \; t \geq 0, \text{ for all possible i} \tag{44}$$

where m denotes the model order, i.e. the number of parallel stages and, as before, N_i denotes the number of exponentials in a branch i, each with transition rate λ_i. It can be seen that Equation 44 is the weighted sum of pdf's of single Erlangian stages given by Equation 6. The j-th non-central moment of the overall process is given by

$$m_j = \sum_{i=1}^{m} p_i \frac{(j + N_i - 1)!}{(N_i - 1)!} \lambda_i^{-1} \tag{45}$$

This equation will be used to determine the quality of an approximation once a fit is obtained.

5.2 A Maximum Likelihood Approach for Parameter Estimation

If $f_t(\mathbf{t} \mid \theta)$ is the random value t dependent on the unknown parameter vector θ, then the optimal parameter choice is those which maximizes $f_t(\mathbf{t} \mid \theta)$ for a given sample vector $\mathbf{y}$. In our case, θ contains two parameter vectors λ and $\mathbf{p}$, and $\mathbf{t}$ represents the durations of time intervals. The function which is to be maximized can be denoted as

$$f_{ML}(\mathbf{p}, \; \lambda) = \sum_{n=1}^{N} f(t_n | (\mathbf{p}, \; \lambda)), \tag{46}$$

where N is the length of the sample vector. Valid constraints are that the sum of non-negative weightfactors p_i has to equal one, and that the transition rates have to be bigger than zero:

$$\text{C:} \quad \textstyle\sum_i p_i = 1, \text{ and for all i} \quad p_i \geq 0, \quad \lambda_i > 0 \tag{47}$$

The mathematical notation of the problem is

$$\begin{pmatrix} \mathbf{p} \\ \lambda \end{pmatrix} = arg \quad \max_{(\mathbf{p},\,\lambda)^T \epsilon \;\; C} \; f_{ML}(\mathbf{p},\;\lambda) \tag{48}$$

With the probability density function specified by Equation 44, one obtains the final expression which is to be solved to find the optimal parameter set:

$$\begin{pmatrix} \mathbf{p} \\ \lambda \end{pmatrix} = arg \quad \max_{(\mathbf{p},\,\lambda)^T \epsilon \;\; C} \; \sum_{n=1}^{N} \; ln(\; \sum_{i=1}^{m} p_i \frac{(\lambda_i t)^{N_i - 1}}{(N_i - 1)!} \lambda_i \, e^{-\lambda_i t} \;) \tag{49}$$

With Equation 49 the optimal set of parameters can be found by solving an nonlinear constrained optimization problem. However, this problem is computationally intensive; assuming a sample size of N = 2000 and a model order of m = 5 (i.e. 5 parallel Erlang-K stages), the innermost part of the sum in Equation 49 has to be computed 10,000 times, just to test for one set $(\mathbf{p},\;\lambda)$. Therefore, special care is taken to speed up the computation of the two sums and to reduce the number of times the equation needs to be evaluated. An implementation in Matlab is introduced in the following.

5.3 Matlab Implementation

The constrained non-linear maximization problem of Equation 49 is solved using the Matlab function FMINCON, which is part of the optimization toolbox. The function is called with the following command:

X = FMINCON(FUN, X0, Aeq, Beq, LB, UB, NONLCON, h, N) FMINCON finds the argument vector X for which the function specified in "FUN" has its minimum; it has the following input variables:

- X0 is a vector of the same length m as the argument vector X; it contains the

starting point of the optimization.

- Aeq, Beq specify the linear equality constraints; for this problem, Aeq is a vector of length m and Beq is a scalar, leading to the condition

  Aeq(1) X(1) + Aeq(2) X(2) + . . . + Aeq(m) X(m) = Beq .

- LB and UB are vectors of length m; they define the lower and upper bounds of the elements given in X

  LB(i) $\leq$ X(i) $\leq$ UB(i) for i = 1,2,. . .,m .

- NONLNCON specifies the name of the m-file containing expressions for non-linear constraints. This option is not needed here.

- h and N are optional parameters; h represents the name of the file containing the sample data t_n. N is a vector which describes the structure of the random process; as before, the i-th element of the vector N represents the number of exponential stages in the i-th branch of the parallel Erlang-K distribution.

For this optimization problem, "FUN" specifies the name of an m-file which takes the arguments X, h and N as input and returns the scalar result of Equation 46. X contains the parameter vectors which is tested by FMINCON. The function is so flexible that results for every possible combination of Erlang-K stages can be obtained without changing the source code.

A straight forward implementation to calculate Equation 49 would make use of two *for* loops, one for every sum. However, *for* loops in Matlab are very slow. Surprisingly it is possible to express f_{ML} by using only matrix operations while keeping $\mathbf{N}$ (and with $\mathbf{N}$ the structure of the process) a flexible system parameter. The crucial part of the Matlab implementation is given in the following lines, starting with the inner part of the

sum and evaluating the overall sum of Equation 49:

$$\mathbf{inner} = \mathbf{p}.*(\lambda.\hat{}\mathbf{N})./fractvec(\mathbf{N}-1)*(exp((\mathbf{N}'-1)*ln(\mathbf{t})).*exp(-\lambda'*\mathbf{t}) \tag{50}$$

$$f = -\,ones(1,\ length(\mathbf{inner})\ )*ln(\mathbf{inner})' \tag{51}$$

Element wise multiplication, division and exponentiation are denoted as .*, ./ and .^, respectively, **facts** computes the element wise factorial of a vector, and ' denotes a transposition. $\mathbf{p}$, λ and $\mathbf{N}$ are row vectors of length m, where k is the number of parallel Erlangian stages. $\mathbf{t}$ is the data vector and can have arbitrary length l.

The result of $\mathbf{p}.*(\lambda.\hat{}\mathbf{N})./fractvec(\mathbf{N}-1)$ is a row vector of length m:

$$\left(\ \frac{p_1\lambda_1^{N_1}}{(N_1-1)!}\ ,\ \frac{p_2\lambda_2^{N_2}}{(N_2-1)!}\ ,\ \ldots\ ,\ \frac{p_m\lambda_m^{N_m}}{(N_m-1)!}\ \right) \tag{52}$$

The last part of Equation 50, $(exp((\mathbf{N}'-1)*ln(\mathbf{t})).*exp(-\lambda'*\mathbf{t})$, produces a k x l matrix

$$\begin{pmatrix} t_1^{N_1-1}e^{-\lambda_1 t_1} & t_2^{N_1-1}e^{-\lambda_1 t_2} & \ldots\ldots & t_l^{N_1-1}e^{-\lambda_1 t_l} \\ t_1^{N_2-1}e^{-\lambda_2 t_1} & t_2^{N_2-1}e^{-\lambda_2 t_2} & \ldots\ldots & t_l^{N_2-1}e^{-\lambda_2 t_l} \\ \ldots\ldots\ldots & & & \\ t_1^{N_m-1}e^{-\lambda_m t_1} & t_2^{N_m-1}e^{-\lambda_m t_2} & \ldots\ldots & t_l^{N_m-1}e^{-\lambda_m t_l} \end{pmatrix} \tag{53}$$

After the matrix multiplication of Equation 52 and Equation 53 one obtains the following row vector of length l:

$$
\begin{array}{l}
(\ \dfrac{p_1\lambda_1^{N_1}}{(N_1-1)!}t_1^{N_1-1}e^{-\lambda_1 t_1} + \dfrac{p_2\lambda_2^{N_2}}{(N_2-1)!}t_1^{N_1-1}e^{-\lambda_2 t_1} + \ \ldots \ + \dfrac{p_m\lambda_m^{N_m}}{(N_m-1)!}t_1^{N_1-1}e^{-\lambda_1 t_1} \ , \\
\ \ \dfrac{p_1\lambda_1^{N_1}}{(N_1-1)!}t_2^{N_1-1}e^{-\lambda_1 t_2} + \dfrac{p_2\lambda_2^{N_2}}{(N_2-1)!}t_1^{N_1-1}e^{-\lambda_2 t_2} + \ \ldots \ + \dfrac{p_m\lambda_m^{N_m}}{(N_m-1)!}t_2^{N_1-1}e^{-\lambda_1 t_2} \ , \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\ \ \dfrac{p_1\lambda_1^{N_1}}{(N_1-1)!}t_l^{N_1-1}e^{-\lambda_1 t_l} + \dfrac{p_2\lambda_2^{N_2}}{(N_2-1)!}t_1^{N_1-1}e^{-\lambda_2 t_l} + \ \ldots \ + \dfrac{p_m\lambda_m^{N_m}}{(N_m-1)!}t_l^{N_1-1}e^{-\lambda_1 t_l} \ )
\end{array}
\tag{54}
$$

In Equation 51 f is the negative sum of the logarthmized elements of the vector $\mathbf{inner}$; it is taken times (-1) so that Matlab's minimization routine can be used for maximization. By replacing the loops with the matrix form solution, the speed is increased by factor 8 to 10.

It is possible to reduce the number of function calls by providing the optimization routine with a gradient of the function which is to maximize. For Equation 55, the gradient is

$$
(\ \frac{\partial f(t)}{\partial p_1}, \ \ldots \ , \frac{\partial f(t)}{\partial p_m}, \frac{\partial f(t)}{\partial \lambda_1}, \ \ldots \ , \frac{\partial f(t)}{\partial \lambda_m} \ ) \tag{55}
$$

with

$$
\frac{\partial f(t)}{\partial p_j} = - \ \frac{t^{(N_j-1)}}{(N_j-1)!} \ \lambda_j^{N_j} \ e^{-\lambda j t} / \ q \tag{56}
$$

and

$$
\frac{\partial f(t)}{\partial \lambda_j} = - \ \frac{p_j \ t^{(N_j-1)}}{(N_j-1)!} \ (-t\lambda_j^{N_j} \ + \ N_j\lambda_j^{N_j-1} \ ) \ e^{-\lambda j t} / \ q \tag{57}
$$

The negative sign again results from the fact the a minimization routine is used for maximization, and q is the inner part of the deviation, i.e. the inner argument of the logarithm:

$$
q = \sum_{i=1}^{m} p_i \frac{(\lambda_i t)^{N_i-1}}{(N_i-1)!}\lambda_i * e^{-\lambda_i t} \tag{58}
$$

The computation of the derivatives is also done in matrix from. Our routine allows to fit

8 parallel Erlang-K stages to 50,000 data points within about 5 minutes.

5.4 Examining the Goodness of Fit

The remainder of this chapter concerns itself with the quality of our parallel Erlangian approximation. In two examples the obtained fit of processes matching the statistics of packet interarrival times and packet transmission durations are analyzed.

Example one: Transfer Times

The measurement of packet arrivals and their description as interval time processes is explained in the next chapter and is of no relevance for now. So far it is only important to know that the statistics of those interval lengths are to be determined.

To do so, the first step is to chose the structure of the parallel Erlang-K process. Normally, no prior knowledge of an appropriate structure is available. In this case, an Erlang-K process as shown in Figure 21 can be used: If one branch is not needed to describe the underlying statistics of the random vector, the weight factor for this branch is simply found to be zero, so no problem results from offering more parallel stages than necessary.

The second step is to chose a starting point for the optimization problem. During our experiments it turned out that, due to the robustness of the optimization procedure, the algorithm converges for almost every starting point.

After the structure and the starting point have been chosen, no further user interaction is required; the function which is to optimize is automatically generated and the parameters are computed. Figure 22 shows the results of a fit: The Erlang CDF is painted in red, approximating the CDF of the data (in blue) remarkably well; for comparison, a simple one moment fit is given in black. The first 6 moments are compared in Table 4: The difference between the first moments is smaller than $2.6 * 10^{-5}$ percent, the

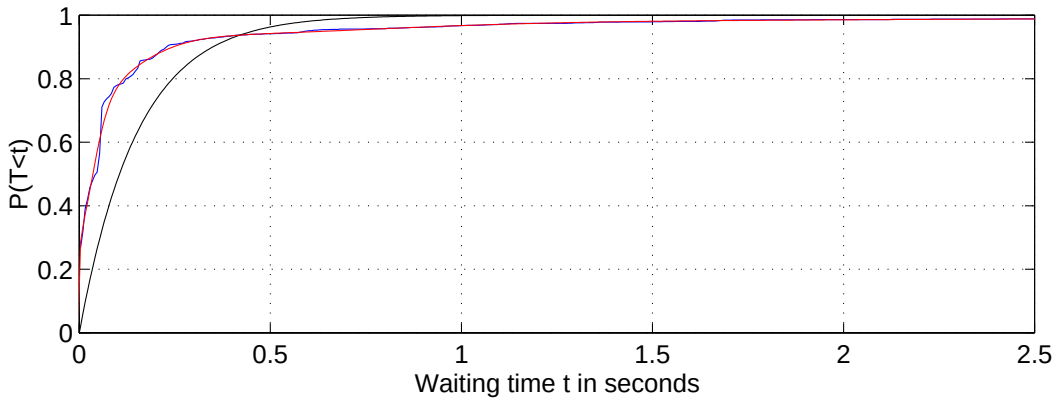

Figure 22: Erlangian Fit of Interarrival Times

Non-central moments of measured data	Non-central moments of exponential fit
$m^{(1)}$ = 0.1517498	$m^{(1)}$ = 0.1517494
$m^{(2)}$ = 0.2658355	$m^{(2)}$ = 0.2635139
$m^{(3)}$ = 0.9767670	$m^{(3)}$ = 0.9723445
$m^{(4)}$ = 4.4134095	$m^{(4)}$ = 4.5612237
$m^{(5)}$ = 21.829472	$m^{(5)}$ = 24.517985
$m^{(6)}$ = 113.54659	$m^{(6)}$ = 146.20628

Table 4: Comparison: Moments of Data vs. Moments of Fit, Interarrival Times

difference between the second and the third moment is smaller than 0.9 and 0.5 percent, respectively. The set of calculated parameters is summarized in Table 5.

Example two: Interarrival Times

The second example is intended to show the robustness of the fitting procedure. The same Erlang-K structure and the same starting point as in the previous example are used to determine the statistics of transfer duration times.

For this dataset it turns out that no single exponential stages are needed to describe the random process: The weightfactors p_5 and p_6 converge to zero. The parameters of

Structure of parallel branch	Transition rates	Weight factors
Erlang K with one stage	λ_1 = 1144.32934951	p_1 = 0.22255139784
Erlang K with one stage	λ_2 = 58.9633706868	p_2 = 0.20445402521
Erlang K with four stages	λ_3 = 67.6736889003	p_3 = 0.34693781019
Erlang K with four stages	λ_4 = 21.0484523593	p_4 = 0.16424534436
Erlang K with six stages	λ_5 = 6.55865724787	p_5 = 0.04485583223
Erlang K with six stages	λ_6 = 1.83856848483	p_6 = 0.01695559018

Table 5: Parameter of Erlangian Fit, Interarrival Times

the fit are given in Table 6. The goodness of fit is visualized in Figure 23.

Structure of parallel branch	Transition rates	Weight factors
Erlang K with four stages	λ_1 = 15.23759	p_1 = 0.129225
Erlang K with four stages	λ_2 = 55.35309	p_2 = 0.701139
Erlang K with six stages	λ_3 = 5.627248	p_3 = 0.072976
Erlang K with six stages	λ_4 = 1.130448	p_4 = 0.096659

Table 6: Parameter of Erlangian Fit, Transfer Duration

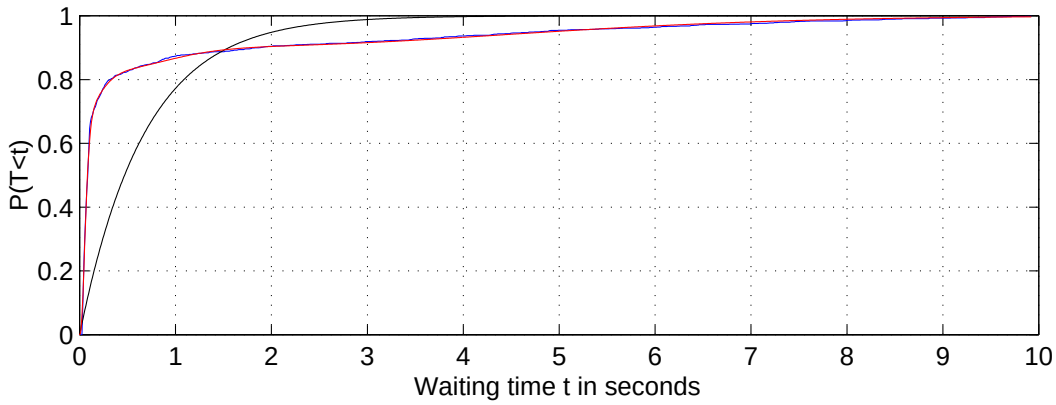

Figure 23: Erlangian Fit of Transfer Durations

The approximation again is very accurate. Both CDFs are hard to distinguish, and the moments of the fit, given in Table 7, also match those of the data very precisely.

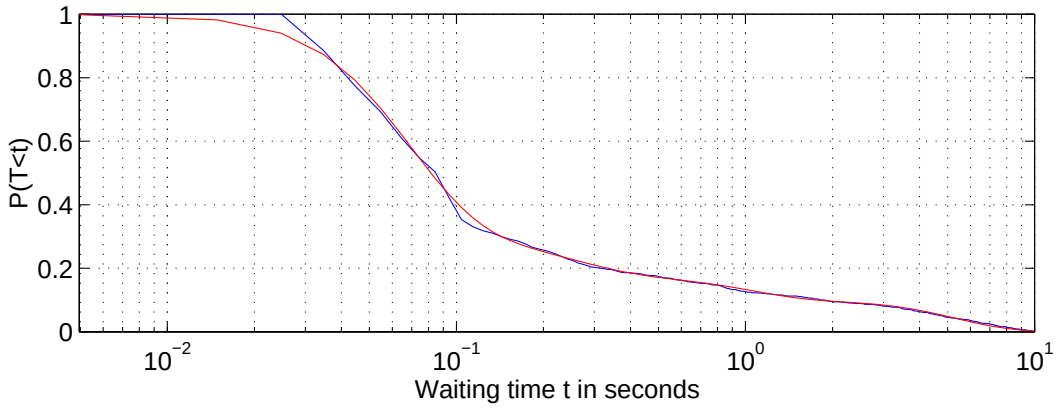

Figure 24: Erlangian Fit of Transfer Durations, CCDF on Log Scale

Non-central moments of measured data	Non-central moments of exponential fit
$m^{(1)}$ = 0.67543	$m^{(1)}$ = 0.67543
$m^{(2)}$ = 3.31491	$m^{(2)}$ = 3.28933
$m^{(3)}$ = 22.5708	$m^{(3)}$ = 22.6244
$m^{(4)}$ = 170.651	$m^{(4)}$ = 179.211
$m^{(5)}$ = 1371.05	$m^{(5)}$ = 1583.74
$m^{(6)}$ = 11472.7	$m^{(6)}$ = 15407.7

Table 7: Comparison: Moments of Data vs. Moments of Fit, Transfer Duration

6 Traffic Measurement

In the previous chapters two approaches have been stated making it possible to describe random time intervals which are characteristics for communication networks with Markovian statistics. The data collection necessary to obtain those time intervals is introduced now: First, techniques to measure Internet traffic are summarized; after this the tools used to conduct those measurements and the user group participating in this survey is described. Finally, the results of the measurement are given.

6.1 Internet Traffic Measurement Techniques

Three general techniques are used to measure Internet traffic: Server logs, client logs and packet traces. All techniques have certain advantages and disadvantages which are stated shortly.

6.1.1 Server logs

Server logs are easy to obtain: A Web server is set up to record times of all file requests, together with file sizes and individual file transfer times. The obtained statistics can be used to create a work load model for file servers, but no general model can be developed since the statistics change significantly depending on the content of the Web server. Server logs are used in [AW96], [Mog95].

A drawback is that no single-user behavior can be obtained because access patterns across multiple servers are not captured. Moreover protocol overheads are not included in the statistics.

6.1.2 Client logs

For Web traffic, client logs are obtained by using modified Internet browsers. They can capture single user behaviour characterized by online session duration, number of requested pages or request times and frequencies, and record file sizes and individual file transfer times. Moreover it is possible to find statistics of document cashing. Client logs are used in [CP95], [CBC95] and [CB95], for example.

A problem with client logs is that no source code widely used browsers is available anymore, making it impossible to modify popular programs like Netscape or Microsoft Internet Explorer. Moreover it is difficult to provide a large number of user with new browsers.

6.1.3 Packet Traces

A third approach is to collect packet traces. This can be done either on a subnet which carries HTTP traffic, or, similar to client logs, on individual computers. Typically a trace file consists of packet headers, sizes, addresses and timestamps for all packets of a specified type. An advantages of this method is that protocol overhead[13] and document caching are already included in the statistics. It is possible to record all traffic which is sent over the PC's ethernet card, and by listening only to a certain port, for example port 80 for HTTP traffic or 21 for FTP traffic, packet traces for specific applications can be obtained.

In a large number of papers packet traces recorded on subnets [Bru97], [Den96], [LPCE99], [CDJM91] or on dial in routers [Vic97], [FBC99] are used.

In this work, a packet-based approach was chosen to collect data. A trace recording tool was installed on a number of individual clients, not necessarily in a single subnet. This

[13]To be precise: The overhead of the higher level protocols is included

approach has several advantages: Without making the user change his or her browser, as it is necessary when using traditional client logs, it is possible to obtain per user statistics. Moreover, the collected traces can capture activities on the IP layer, which is close to the hardware interface modeled in the next chapters. The downside of this approach is that not 100 percent of the packets are captured when the processor utilization of the client is high, and that the trace recording process may slow down the computer. This issues are discussed in detail in [Deg00].

6.2 Tools for the Trace Record

As mentioned before, the aim was to collect traffic traces on individual clients. This is possible with the packet capturing software *tcpdump*, which was developed at the University of Berkeley. *Tcpdump* is a command-line tool for filtering and capturing network traffic from protocols such as IP, ICMP[14], ARP[15], RARP[16], UDP[17] and TCP. It is possible to specify the protocols and ports of interest *tcpdump* is supposed to listen to, as well as the detail of information which is recorded.

An example for a typical program call is *tcpdump* -n -tt ip and port 80 » dump.txt. Unformatted timestamps (-tt) of all IP packets, incoming and outgoing, (ip), which carry HTTP traffic (port 80) are written in the local file dump.txt, including source and destination addresses (-n). A complete list of the available options and commands for *tcpdump* is given in the Appendix.

The major drawback of *tcpdump* is that it does not run under Microsoft operating systems, excluding a large number of potential users from the traffic observations. This problem was most recently addressed by Loris Degioanni, [Deg00], who introduced a

[14] Internet Control Message Protocol, [SPE81]
[15] Address Resolution Protocol, [Plu82]
[16] Reverse Address Resolution Protocols, [Plu82]
[17] User Datagram Protocol, [BLMM94]

Windows port named windump which provides the same functionally and uses the same commands as *tcpdump*.

However, windump is still a command-line based tool; in order to make it easy to use, a number of batch files for Windows 98, WinNT and Win2000 were written during the research project which allow the user to start and stop the packet capturing process with a single mouse click. Moreover the program's installation is automated and the user is asked to enter the following information for the completeness of the statistics when the program is started the first time:

- age
- gender
- private, educational or professional use
- speed of Internet connection
- time zone

Windump was configured to record source and destination addresses, packet sizes, and the arrival or departure times for all incoming and outgoing IP packets which carry HTTP traffic. The timestamps are given with a resolution of 100 microseconds.

Since the traffic traces are saved on the local machines, it is necessary to transfer, them to a central server. This is done automatically: When a new traffic measurement is started, the recent log file is concatenated with the personal information, compressed and given a unique name. For the file transfer the MS-DOS ftp client is used. The commented scripts are provided in Appendix B.

On the central server a batch script is used to append the individual user statistics to a database which allows queries to obtain packet traces of a subset of users or separation of incoming and outgoing packets. The database is used to generate files serving as input to Matlab routines for further processing of the data.

6.3 User Profile

In previous studies, traffic measurements were conducted on corporate or university networks. One can assume that user behaviour in those surroundings differs from the habits of private users for several reasons: The time for a browsing session as well as the downloaded content is likely to be different. This is can be explained as a result of different costs for being online, and because it is not recommended to visit certain pages during work. This motivated us to examine traffic traces which originate from private surroundings rather than using existing measurements.

Windump together with the automated trace upload can be installed on individual private computers, which makes it possible to obtain packet statistics for users of all ages, in all surroundings, and after working hours. This is a major advantage for service providers with a large number of private customers. Therefore our scripts together with *windump* my be of interest for other researchers in this field.

Since the network which is modeled later works at 768 kbps, traffic traces captured at Internet connections with similar speeds are of primary interest, so that assuming similar user behaviour on the network under study and the trace generating network is realistic. Two different types of network connections were available which fulfill the bandwidth requirement: The 1 Mbps Internet access in the WPI dormitories, and DSL connections operating at a maximum speed of 768 kbps.

During a period of two months, from beginning of February until the end of March, 30 users with desired connection speeds, most of them students, were constantly recording their traffic traces.

6.4 Results

Trace records represent 30 users transferring 760 MB of data. The two hours with the highest bandwidth demand, referred to as busy hours, were found to be those from 3 am to 5 am. With a probability of 20 to 33 percent a user was online at least once during this two hour period.

Since the behaviour of a network under highest realistic traffic loads is of primary concern, the traces generated during busy hours were used to derive statistics for packet interarrival times and packet sizes. The traces contain two million data points, which is sufficient to obtain precise results for those short term characteristics; techniques to find the interarrival times and to calculate transfer durations for packets on the network under study are discussed in the next two chapters.

However, a larger number of users had been necessary to compare traffic patterns of different user groups, or to find reliable statistics for changes of traffic demand during a week. Moreover the time of the busy hour and the traffic intensity is likely to be biased because the group of users is not representative.

Nevertheless, *windump* together with the automated trace upload can be used to obtain the necessary amount of data even for the long term statistics. Perhaps these issues may be addressed in a future research project when more time for the packet collection is available.

7 The Outer ON/OFF Model

In the remaining two chapters, the performance of a wireless network is evaluated by employing the statistical tools which have been introduced in the previous part of the thesis. For the performance analysis, the traffic originating from a single source is described using two ON/OFF models, where the outer process modulates the inner one: When the outer process is in its OFF state, no traffic is generated, representing a user which is reading a previously downloaded document, for example. This state is referred to as active OFF - active because the user has at least once generated network traffic during the measurement, OFF because he or she is not causing network traffic at the moment. A user who is currently down- or uploading content with a browser is represented by the outer ON state, referred to as active ON. The traffic generated while being in this state is inherently bursty and not sufficiently modeled as a simple Poisson arrival process. For this reason, a second ON/OFF process, in the following called access model, is introduced, modeling packet arrivals while being in the active ON state in greater detail. The model is illustrated in Figure 25.

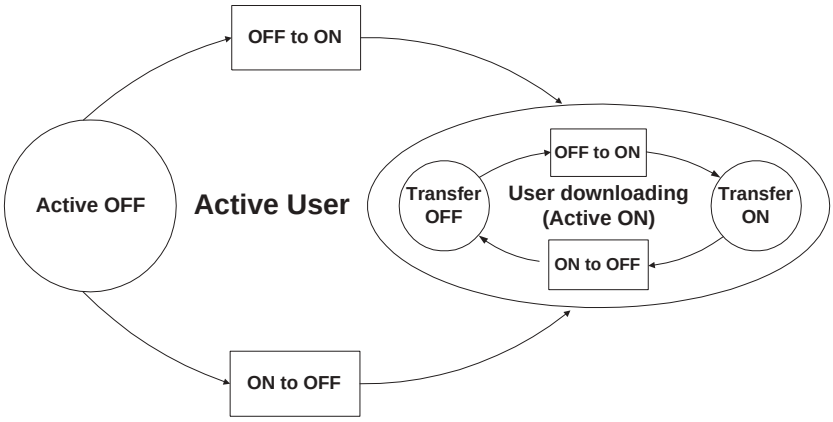

Figure 25: Transition Diagram Complete Model

The underlying statistics of the state transitions for the access model were obtained

from traffic measurements. For the outer process, the collected traces were only sufficient to derive first order statistics; not enough data was available to accurately fit distributions to it. However, reasonable estimates for the probability of being in the active ON or active OFF state are obtained which can be used to find overall probabilities when examining the access model.

7.1 State Probabilities of the Outer Model

An active user is defined as a user who has at least once generated network traffic during a measurement period. For measurements taken during the busiest two hours of the day, the probability of a user to be categorized as active was found to lay within 25 and 33 percent. These numbers must be used with care; traffic traces spanning a longer time period and capturing a more representative user group are necessary to obtain more accurate results. However, the main emphasis lies on the evaluation of the access model in Section 8. The outer ON/OFF model is basically intended to show that traffic which is self-similar over a large time scale can be generated with a Markovian model. This done in the following sections.

7.2 Generating Self-Similar Traffic

Referring to [LWTW93], [WTE97] and [Den96], ON/OFF models can be used to generate self-similar traffic when the time being in either state is described by long tailed distributions. This statement is supported by the following experiment:

A single traffic source is approximated using the model illustrated in Figure 26; network traffic is generated with continuous rate only while being in the ON state, and transitions from ON to OFF and OFF to ON are described by Pareto distributions with $\alpha = 1.2$ $\beta = 5$ $\alpha = 1.5$ $\beta = 60$, respectively. The tail parameters are taken from [LWTW93]. Traffic from 100 identical and independent sources is superimposed. A resulting synthetic trace is given in Figure 27.

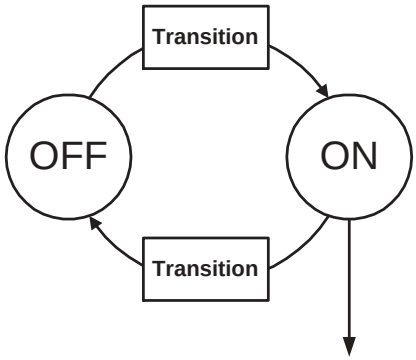

Figure 26: Simple Two State Model

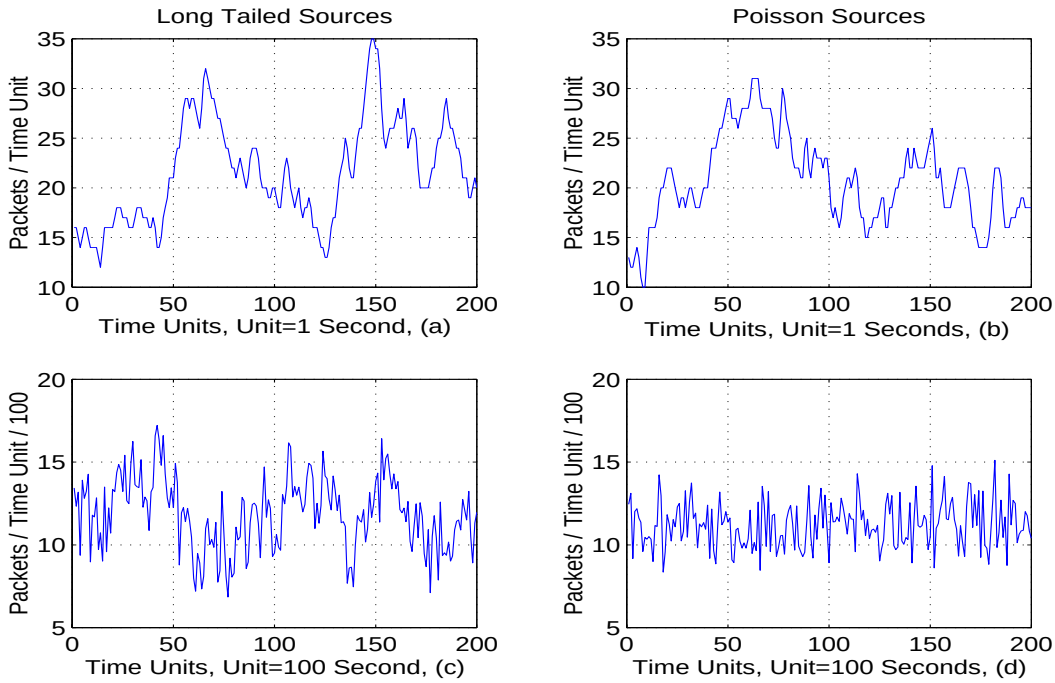

Figure 27: Aggregation of Synthetic Traffic Trace

The simulation is written in Matlab with the source code given Appendix A.3; care was taken that initial conditions do not have an influence on the measurement by ignoring the first 10,000 transitions.

For comparison the same model but with exponentially distributed state transitions

was set up. The exponential waiting times were chosen such that the throughput of a single source is equal in both models.

The difference between the traffic pattern becomes apparent when the synthetic traces are aggregated: When the underlying statistics are heavy tailed, illustrated in Figure 27 (a) and (c), the aggregated traffic remains bursty but smooths out for exponential holding times, given in Figure 27 (b) and (d). The corresponding variance-time plot are shown in Figure 28 and Figure 29.

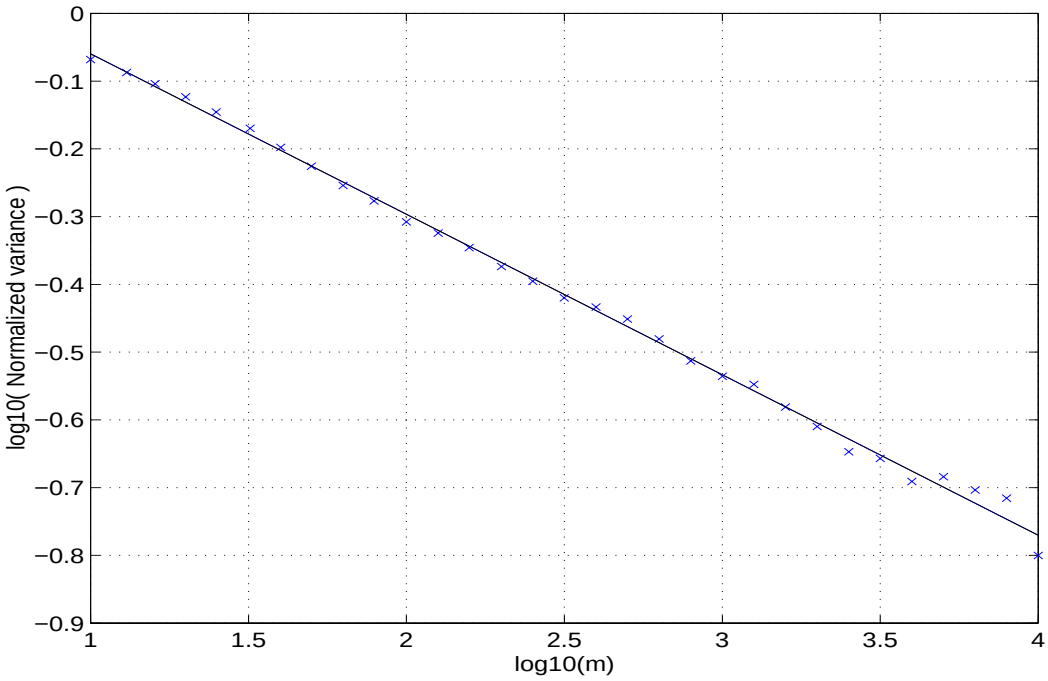

Figure 28: Variance-Time Plot of Traffic Originating from Heavy Tailed Sources

For the long tailed model, the slope of the variance-time plot is found to be $\beta = 0.233$ using linear regression. This yields to a Hurst parameter of $H = 1 - \frac{\beta}{2} = 0.88$ which is the same result [LWTW93] obtained when working with the real traffic trace characterized by the afore mentioned Pareto distributions. For the exponential model, H is found to be $H = 0.52$, which is close to the theoretical value of $H = 0.5$ for short range dependent processes.

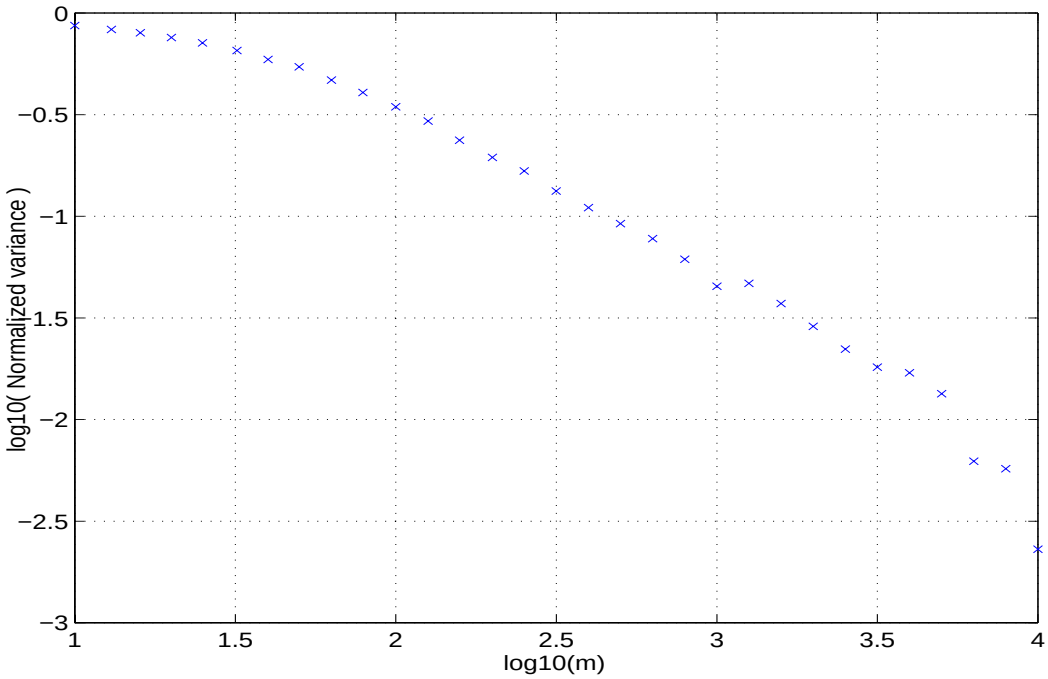

Figure 29: Variance-Time Plot of Traffic Originating from Exponential Sources

The previous example shows that the "long tailed model" is capable of describing self-similar characteristics of network traffic and the two-state exponential model is not. The advantage of the exponential model is that the traffic model can be defined by using only Markovian statistics, leading to feasible computations in performance analysis; the same is true for Hyperexponential holding times.

A question we want to answer now is whether self-similar network traffic can be generated by using the Hyperexponential approximation as introduced in Section 4.2. Therefore the Pareto distributions in the previous example are replaced by their 16-stage Hyperexponential fit. The state transition diagram corresponding to Figure 26 with Hyperexponential approximations is illustrated in Figure 30.

The degree of self-similarity of the synthetic traffic trace originating from the Hyperexponential model is again determined by a variance-time plot, shown in Figure 31.

As expected, the traffic is not truly self-similar and smooths out after a certain level

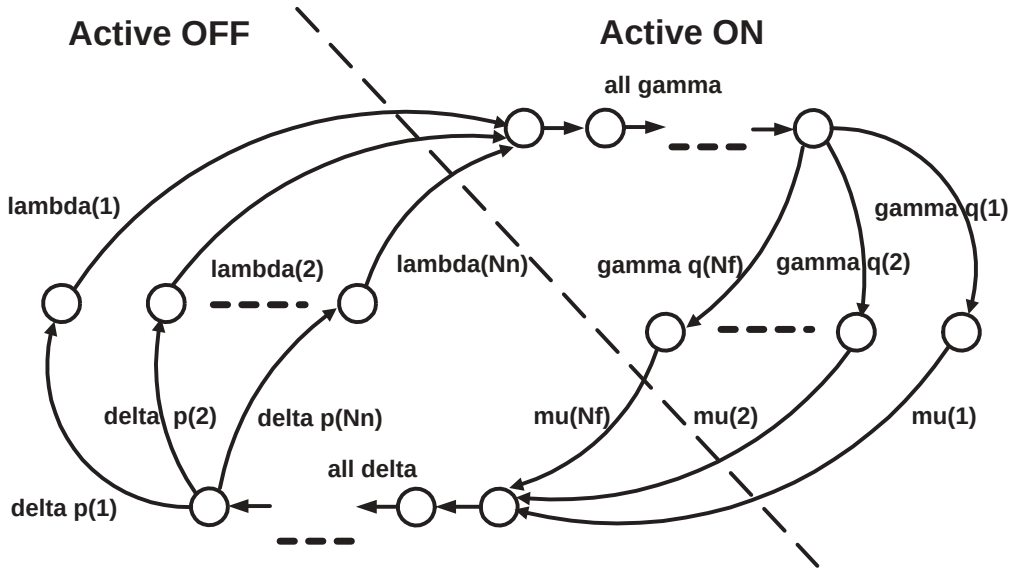

Figure 30: State Transition Diagram of Hyperexponential ON/OFF Model

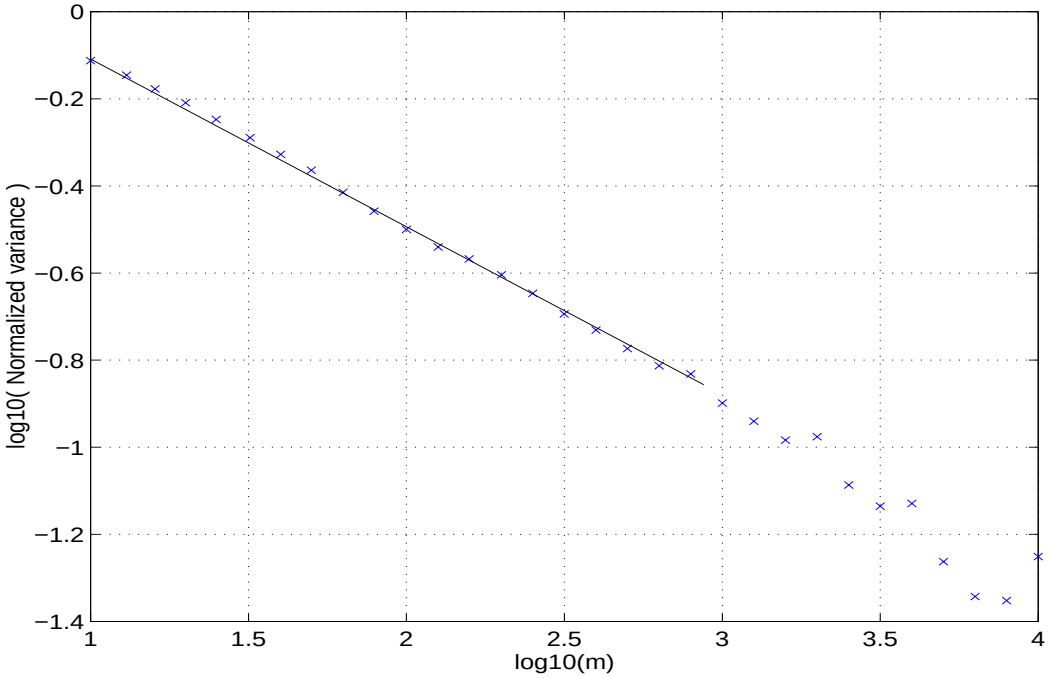

Figure 31: Variance-Time Plot of Traffic Originating from Hyperexponential Sources

of aggregation, reflecting the not truly long tailed underlying distributions. However, for an aggregation level of up to 1000 (i.e. for up to 1000 independently distributed sources) the trace shows self-similar characteristics. If the Hurst parameter is computed for less than 1000 sources, it is found to be $H = 0.87$ which is very close to its original value of $H = 0.88$. When approximating long tailed distributions with a larger number of exponential stages, the self-similarity and with it the burstyness of a trace can be maintained for a larger number of sources.

From the last example one can conclude that extended models using standard techniques are capable of describing network traffic even in the presence of self-similarity. This result can be seen as an argument in favor of the opinion that it is *not* necessary to rewrite the queueing theory for data networks as predicted in [PKC97].

8 The Access Model

Two ON/OFF processes are used to model traffic originating from a single source. In the previous chapter the outer process, which modulates or, in other words, activates the inner process, was introduced. The inner process is discussed in the following; it models the channel requests in a simplified wireless network and is based on the following assumptions which are valid for a system currently developed at Beamreach Network Inc. described below:

- The number of access servers, i.e. the number channels which can be assigned simultaneously is restricted.
- The channel assignment takes a deterministic time on the order of 3 msec.
- When a mobile station asks for a channel, a random access channel similar to those of an IMT-2000 compliant system as introduced in Section 2.6, but with a larger number of orthogonal codes such that collisions of requests are negligible.
- The overall system bandwidth is assumed to be unrestricted, i.e. a user is blocked only due to limitations of the channel assignment process.

In the following chapter, the access model is introduced, and its state transitions are approximated using the parallel Erlang-K fitting method from Section 5. Having done that, a Markovian state diagram is developed such that the access model can be described in matrix form. With the matrix representation, blocking probabilities or the utilization of the transmitter are derived.

8.1 A Detailed Description of the Model

The access model consists of five states, named Transfer OFF, Access Request (AR), Access, Wait and Transfer ON. Its state transition diagram is given in Figure 32.

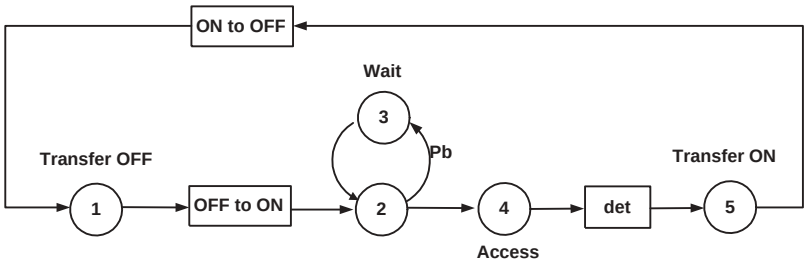

Figure 32: Inner Traffic Model

One can think that a token circulates in the state diagram: When a download begins, the token is inserted in the transfer OFF state. After a random time described by the OFF to ON transition a packet or a group of packets is ready to send and the token passes to state AR, indicating that an access channel is requested. With probability P_b no access server is available and the token enters the Wait state, where it stays for an exponentially distributed random time until a new request is made, again with a blocking probability of P_b. The token leaves state AR to the Access state with $1 - P_b$, the probability that a server is available. The duration of assigning a channel is deterministic, so after a fixed time the token passes to the Transmission state, indicating that data can be transferred without prior scheduling. After a random time defined by the ON to OFF transition, the token enters the Transfer OFF state and the channel is released. The process starts again when the next packet or group of packets is ready to be sent. For a complete model description, the state transitions have to be determined, which is done now.

8.1.1 Determining the State Transitions

The exponential waiting time in case of blockage and the deterministic time until a free access channel is assigned are given system parameter, whereas the ON to OFF and OFF to ON transitions also depend on factors such as user behavior and file size distributions. For these transitions, the underlying statistics are obtained using traffic traces which were recorded on Internet connections with similar bandwidth constraints

and pricing policies as present in the system under study. Therefore one can expect similar user behavior resulting in approximately the same traffic pattern.

The traffic trace contains packet arrival times and sizes of individual sources; to derive the statistics for the state transitions it is necessary to first find the time intervals of being in the transmission ON and transmission OFF state if the recorded trace had been a network load on the system under study. The trace contains per user packet arrival times and packet sizes. With this information it is possible to calculate the time between the end of a transfer and the next access request, referred to as inter transfer time T_{id}, and the time for being in the transfer state T_d, referred to as transfer time. For this calculation it is necessary to take into account that, if possible, packets are grouped together into larger units to reduce the number of channel assignment procedures. This is done using the following heuristic:

- When a packet becomes ready to send while no access channel is assigned, a request for a channel is generated, independent of events in the future or in the past.

- Packets which are generated while a channel is requested by, or assigned to the packet generating station use the same channel, i.e. the system does not need to allocate a new one.

- After a packet is sent and no further packet is already generated, the system stays in the transmission state for a fixed time to wait for an eventually generated packet. We assume a additional waiting time of 30ms[18]. If a packet is generated during this waiting time, it is sent using the open channel.

In Figure 33 an example is provided illustrating the transmission process at a wireless connection.

[18]The waiting time was a suggestion from Beamreach Networks Inc.

At time T_{ar1}, the first packet is to be sent and a request for an access channel is generated. At time T_{aa1} the access channel is assigned and at T_{s1} the sender starts the transmission. At T_{ae1} the transfer of the packet is complete. The duration of the transmission T_d depends on the size of the packet being transferred and on the speed of the connection.

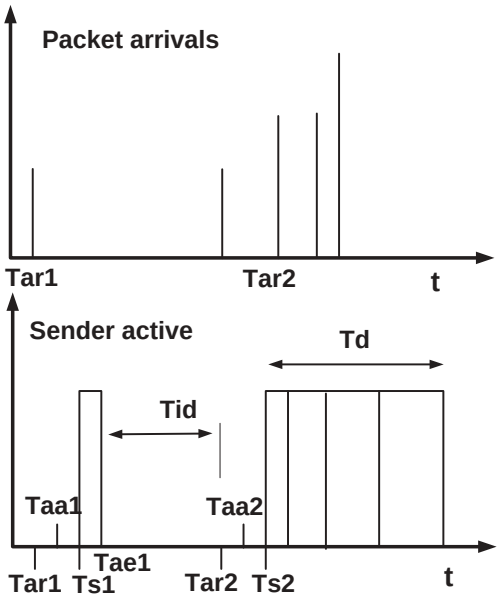

Figure 33: Time Line of Packet Arrivals

At time T_{ar2} a second packet is to be sent. Time T_{id} between the previous release of the access channel, T_{ae1}, and the next request, T_{ar2}, is referred to as inter transfer time.

Let's consider the case where a number of other packets become ready to send during the channel allocation, the actual sending process, or a short time after the transmission of the first packets. These packets are sent using the assigned channel, therefore it is not necessary to go to the access state again. In this case the duration being in the transmission state T_d is determined by the sum of packet sizes.

The Matlab routine which is used to find the time intervals of a given traffic trace is given in Appendix A.4.

8.2 Fitting Data to Erlang-K Distributions

Having obtained the intervals it is now possible to find probability distributions to describe the time spent in the Transfer ON and Transfer OFF state. However, it is not possible to accurately fit a common distribution to the data. With Weibull, Pareto, Log-

normal or exponential distributions it is only possible to match the first one or two moments. On the other hand, using the Erlang-K approximation as introduced in Section 5 provides good results. The difference between the first moment is less than 0.01 percent, between the second moments less than 1 percent and between the third moments less than 0.6 percent. The pdfs are approximated very precisely; as an example, two such pdfs are given in Figure 34, with the data's pdf in blue, the Erlang-K fit in red.

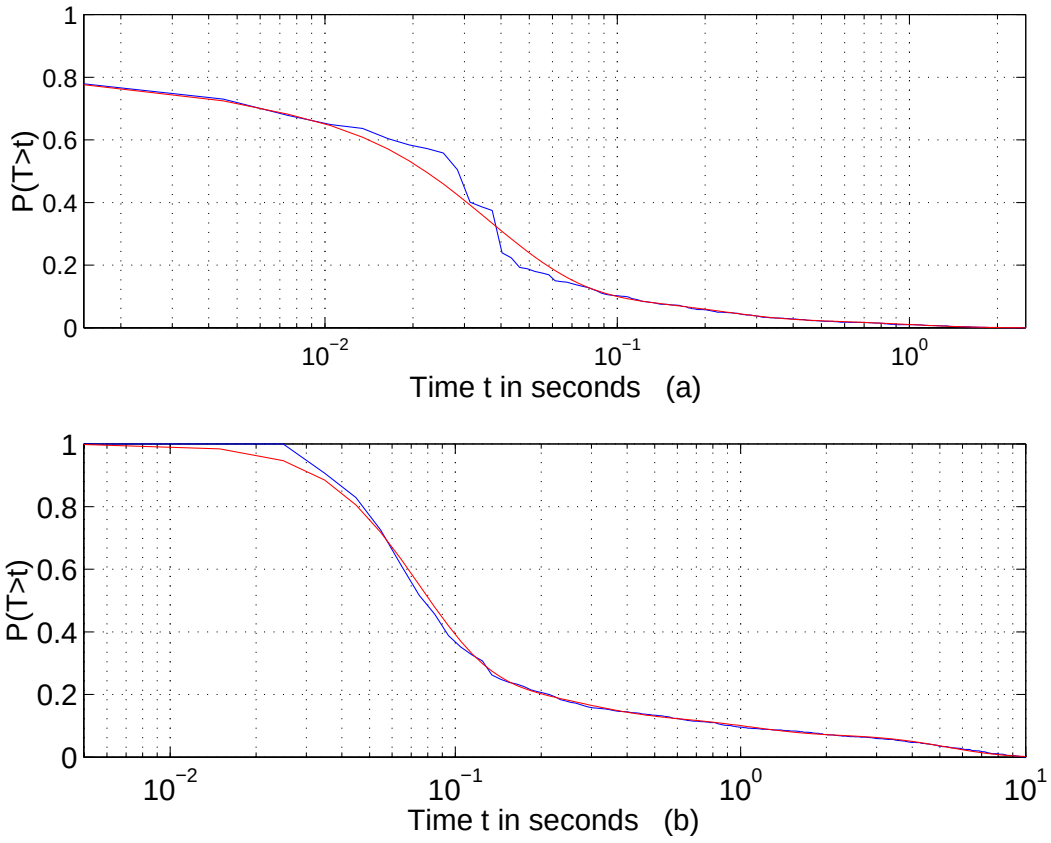

Figure 34: Erlangian Fit for Incoming Data on a 768 kbps Network: (a) Transmission Duration; (b) Interarrival Times

8.3 Markovian Description and Solutions for Steady State Probabilities

It is desirable to use only exponential wait states when describing the transitions in Figure 32. Doing so, it is possible to represent the access model with a transition matrix and derive steady state probabilities by solving a matrix equation. Utilizing the parallel Erlang-K fit for the Transmission OFF and Transmission ON times, the only non-exponential waiting time is used to model the channel assignment, which is a deterministic process.

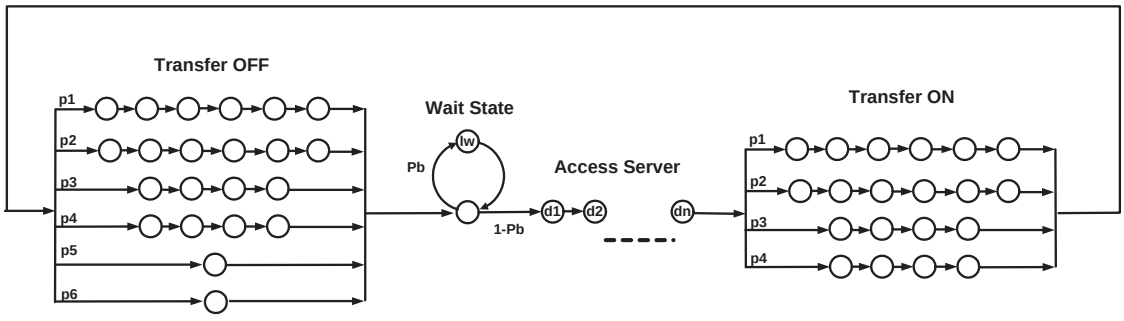

Figure 35: Transition Diagram of Access Model

As mentioned in Section 2.1.1, deterministic processes can be approximated arbitrary closely by a single Erlang-K random process if the number of exponential stages goes to infinity. 500 equal exponential waitstates are used to model the channel assignment, therefore the time spent in the system with a confidence of 90 percent lies 2.5 percent around the mean.

Now the Markovian state transition diagram can be drawn: The exponential wait states 1 until 6 correspond to the Transfer ON, states 9 until 13 represent Transfer OFF. Blockage is modeled with states 7 and 8, the deterministic channel assignment is given by 14 until 14+N. The model we used describes the Transfer states with a higher number

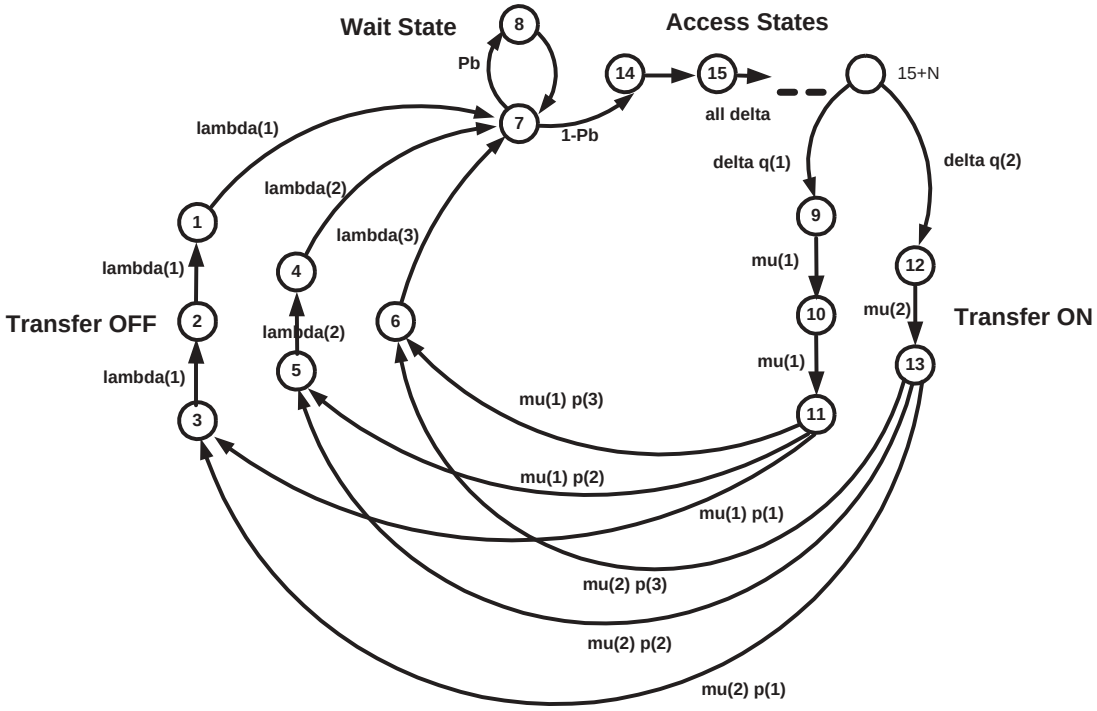

Figure 36: Markovian State Transition Diagram of the Access Model

of exponentials. A nxn transition matrix **P** can be written down with the elements

$$q_{ij} = \begin{cases} \lambda_i p_{ij} & i \neq j \\ -\lambda_i & i = j \end{cases} \tag{59}$$

as introduced in Section 2.3. A Matlab routine is used to set up **P**; the matrix contains about 260,000 elements, mostly due to the approximation of the deterministic part, and depends on the structure of the parallel Erlang-K approximation.

The steady state solution for the probability is obtained by solving

$$\pi \mathbf{1} = 1 \tag{60}$$

$$\mathbf{0}^T = \pi \mathbf{Q} \ , \tag{61}$$

for π. With the Matlab implementation given in Appendix A.4, this is done in less than 10 seconds. Knowing which exponential state corresponds to which state in the non Markovian access model in Figure 32, the steady state probabilities of being in the Transfer OFF, Transfer ON, Access and Wait state, can be obtained.

8.4 Blocking Probabilities due to Restrictions of the Access Server

So far a single user was modeled independently from other sources. This assumption no longer holds for a large number of users in the system: The access scheme can only process up to N_c channel requests at the same time; if more requests occur, they are blocked. Therefore the probability of blockage, P_b, i.e. the probability that more than N_c users request a channel within a time slot, depends on the total number of users. The computation of P_b is discussed in the following: First it is shown how to compute P_b under the condition that N users are in the active ON state. Then the pdf is derived relating the number of subscribed users N_s to the number of active ON users N, and finally P_b is given as a function of N_s. For now, bandwidth is of no concern and the only restricted resource is the maximum number of channel requests the system can process at a time. We show that the utilization of the access scheme is low, and following the discussion in [ST99], we conclude that the self-similar property of network traffic is of no significant importance for the evaluation of the blocking probability, at least for the multiserver case, allowing us to use binomial distributions to compute the number of users in the Transfer ON state. The utilization of the access scheme is shown in Figure 37.

The probability P_{access} for a single active ON user to have a channel request being processed at a given time can be found with the matrix Equation 61, assuming that blocking probability P_b is known. However, this is not the case, but it is possible to derive P_b and P_{access} recursively:

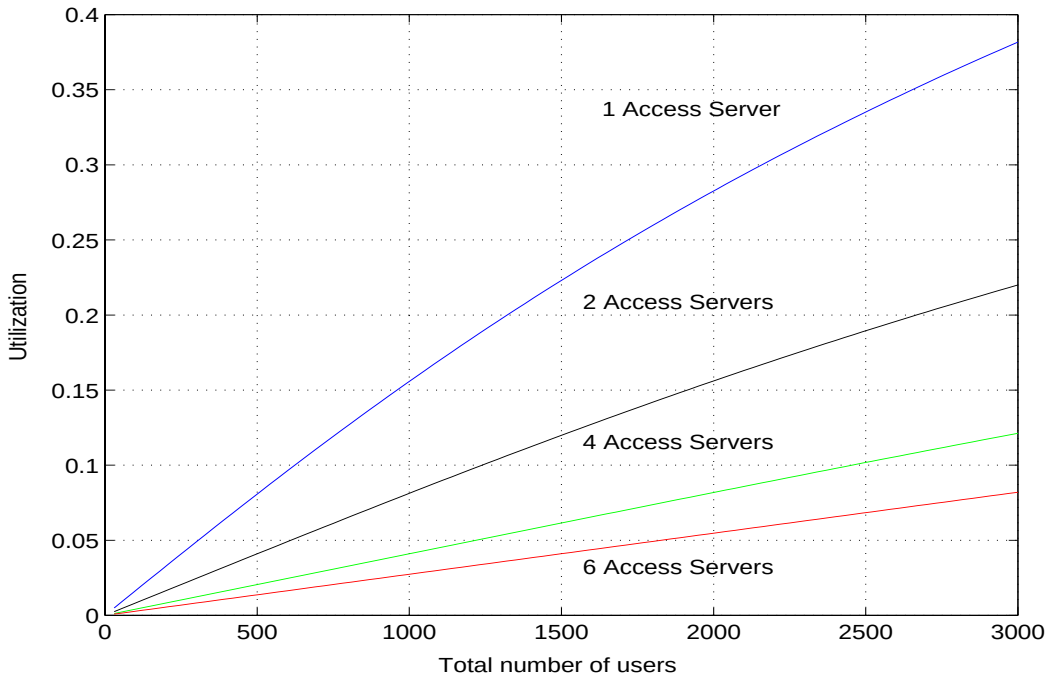

Figure 37: Utilization of the Access Scheme for 2, 4 and 6 Access Servers

$P_{b,0} = 0$ is chosen for the first calculation of P_{access}. With P_{access} and N users in the active ON state, each of them described by an individual access model, the probability density function of n out of N users asking for a channel can be given by a binomial distribution, given in Equation 62.

$$f(n \mid (P_{access} = p \mid P_{b,0})) = \begin{pmatrix} N \\ n \end{pmatrix} p^n (1-p)^{N-n} \tag{62}$$

With Equation 63 the probability that users ask for a channel when all N_c access servers are busy can be derived:

$$p(P_{access} \mid P_{b,0}) = 1 - \sum_{n=0}^{N_c} f(n \mid P_{access \mid P_{b,0}} = p), \tag{63}$$

which is the probability of blocking $P_{b,1}$ under the condition $P_{b,0}$. In the next step, $P_{b,1}$

is used to calculate $P_{b,2}$ and so on. The recursive relation can be expressed as:

$$P_{b,i} = p(n|(P_{access} \mid P_{b,i-1})). \tag{64}$$

Equation 64 converges fast. Usually, for $i = 10$ the difference between $P_{b,i}$ and $P_{b,i-1}$ is smaller than 0.01 percent, leading to a very good approximation of P_b.

So far the probability of blocking P_b for an individual source is expressed as a function of the number of users in the active ON state. In the following, P_b is related to the number of subscribed users.

Lets assume that the access scheme should support N_s subscribed users. In the worst case of all traffic logs collected during the two busiest hours of the day, less than 1/3 of the subscribed users are working with their Internet connection in a way that they are described by the outer ON/OFF model. From the N_a active users 25 to 37 percent were found to be in the active ON state, indicating that they are currently down- or uploading content from the Internet.

The probability of 37 percent for an active user to be in the active ON state was chosen to compute the probability density function of the number of traffic sources being described by the access model,$f_a(n)$, assuming a binomial distribution. The pdf $f_a(n)$ is then scaled by 3 so that it relates the number of subscribed users to the number of users described by the access model.

$$f(n_{trans}) = \begin{pmatrix} N_a \\ n_{trans/3} \end{pmatrix} p_{on}^{n_{trans}/3}(1 - p_{on})^{N_a - n_{trans}/3} \tag{65}$$

The blocking probability can be expressed as

$$p_b(n_{sub}) = \sum_{n_{act}} p(n_{act}|n_{sub}) p_b(n_{act}) \tag{66}$$

8.4.1 Probability of Blockage for Different Network Configurations

In the following, the influence of per user bandwidth and number of access servers on the blocking probability will be examined. The underlying statistics were obtained from the traffic trace described in Section 6.4, where the Matlab routine extract_2 is used to find the durations of all ON and OFF periods, depending on the per user bandwidth of the target network. For all transmission speeds of the up- and downlink which are examined in the following, an individual parallel Erlang-K description of the transfer ON and OFF state is derived. The mean of the exponentially distributed waiting time in case of blockage is assumed to equal 30 msec, and the service time of an access server is set to 3 milliseconds.

Figure 38 shows the blocking probability over the number of subscribed users for a downlink with 768 kbps per user bandwidth. The systems are assumed to have only one and two access servers.

If only one access request can be processed every 3 milliseconds, the blocking probability due to restrictions of the access server is bigger than $P_b = 0.1$ for 900 subscribed users and $P_b = 0.2$ for 1500 users. Since in this case the low utilization assumption does not hold, the graph only provides an estimate for the blocking probability in the single server case. For all other network configurations the low utilization assumption is satisfied.

In a system which can process two requests at once, P_b is significantly smaller: For 900 subscribed users, $P_b = 0.025$, increasing to $P_b = 0.06$ for 1500 users. However,

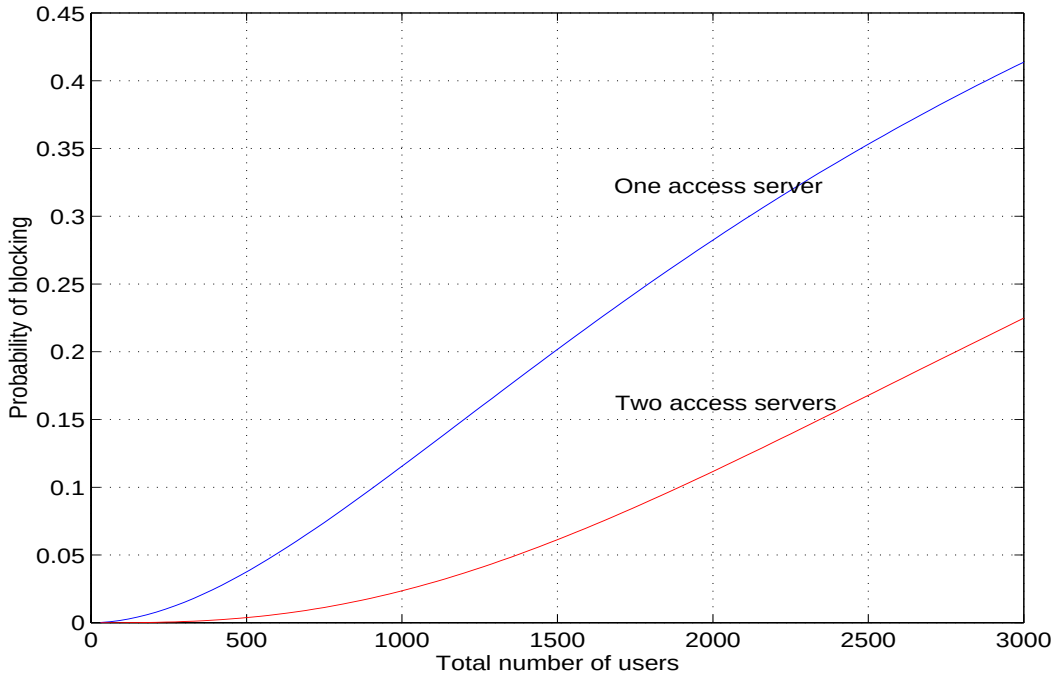

Figure 38: Downlink, One and Two Access Servers, 768 kbps

when aiming at 3000 customers, two access servers are not sufficient.

In Figure 39 the performance of systems with 2, 3, 4 and 6 servers is compared. Again, the downlink with a per user bandwidth of 768 kbps is evaluated.

If the target is to dimension the access scheme so that 3000 customers can be served without degrading the performance due to restriction of the access scheme, it has to be possible to process four to six channel requests at the same time. Figure 39 indicates that with four access servers the probability of blocking is smaller than 4 percent, and for six servers even less than 0.5 percent.

The utilization of the access scheme and with it the blocking probability also depends on the speed of the network connection. Figure 40 indicates why:

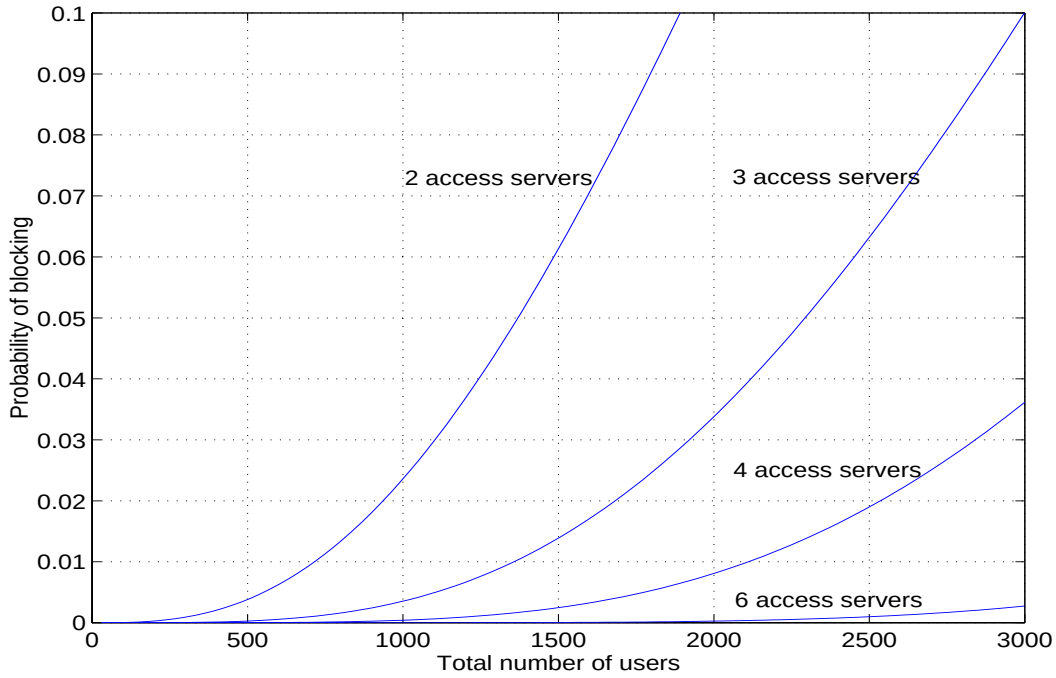

Figure 39: Downlink, 2,3,4,6 Access Servers, 768 kbps

If two consecutive packets are generated separated by a time which depends mostly on the speed of a Web server, the time gap between end of packet one and start of packet two is larger if the connection speed is higher. This increases the probability that a channel is released and has to be assigned new for the next packet, making more channel requests necessary. Another important factor is that the user behaviour depends on the available bandwidth. However, the traffic logs were recorded on a network with about 500 to 800 kbps, so the statistics for the slower connections, especially for the 192 kbps link, should be treated with care. Figure 41 relates the blocking

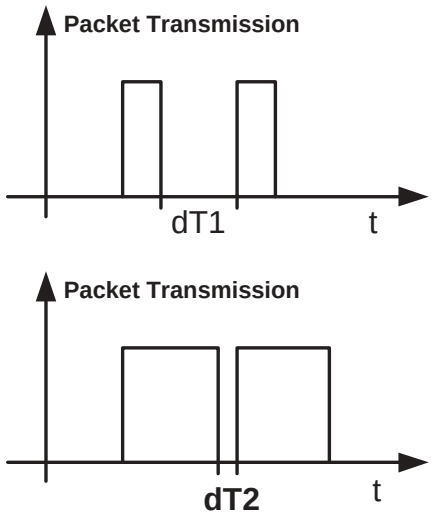

Figure 40: Time Line of Packet Arrivals

probability for different network speeds to the number subscribed users. The system is assumed to have three access servers.

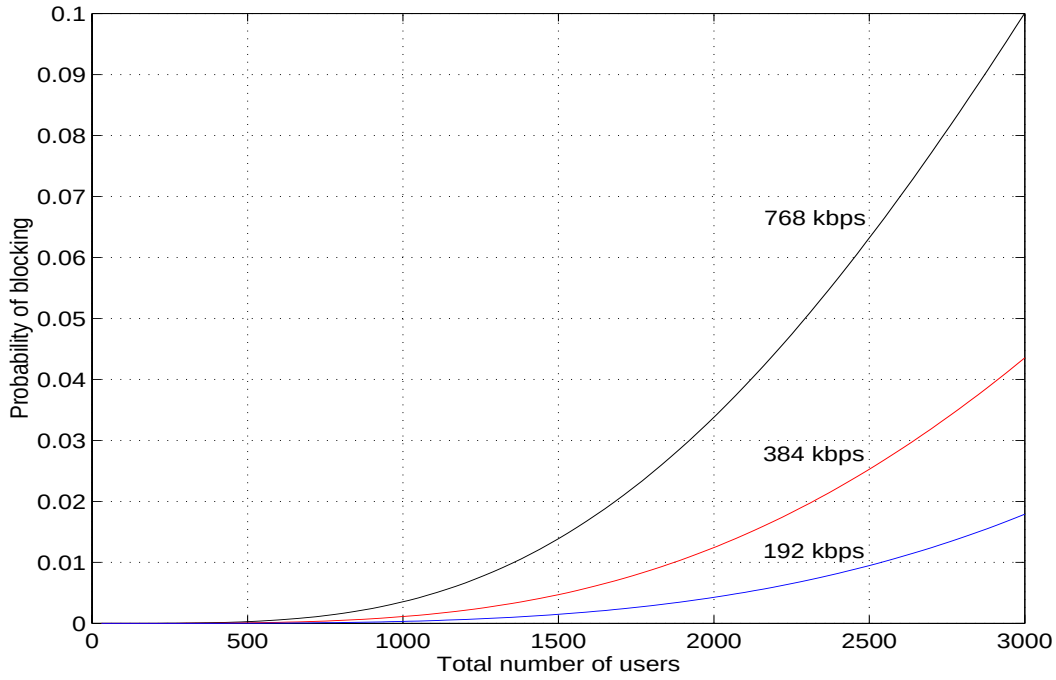

Figure 41: Three Access Servers, 768 kbps (black), 384 kbps (red), 192 kbps (blue),

The blocking probability is more than twice as high for the 768 kbps link compared to the 384 kbps source. For the 192 kbps connection, P_b is much lower than expected, most likely because the traffic trace from which the statistics are obtained describes a user behavior which is not realistic for such a slow connection: During a download a user stays almost all the time in the transfer ON state, making new channel requests unnecessary.

In the following the blocking probability of up- and downlink are compared. It turns out that the uplink shows a higher probability of blockage, even for an asymmetric network connection.

In Figure 42, P_b is given for two different networks: One with a per user bandwidth of 768 kbps and 384 kbps for down- and uplink, respectively, and one with a transmis-

sion rate 384 kbps and 192 kbps. In both cases, three access servers are assumed for either direction.

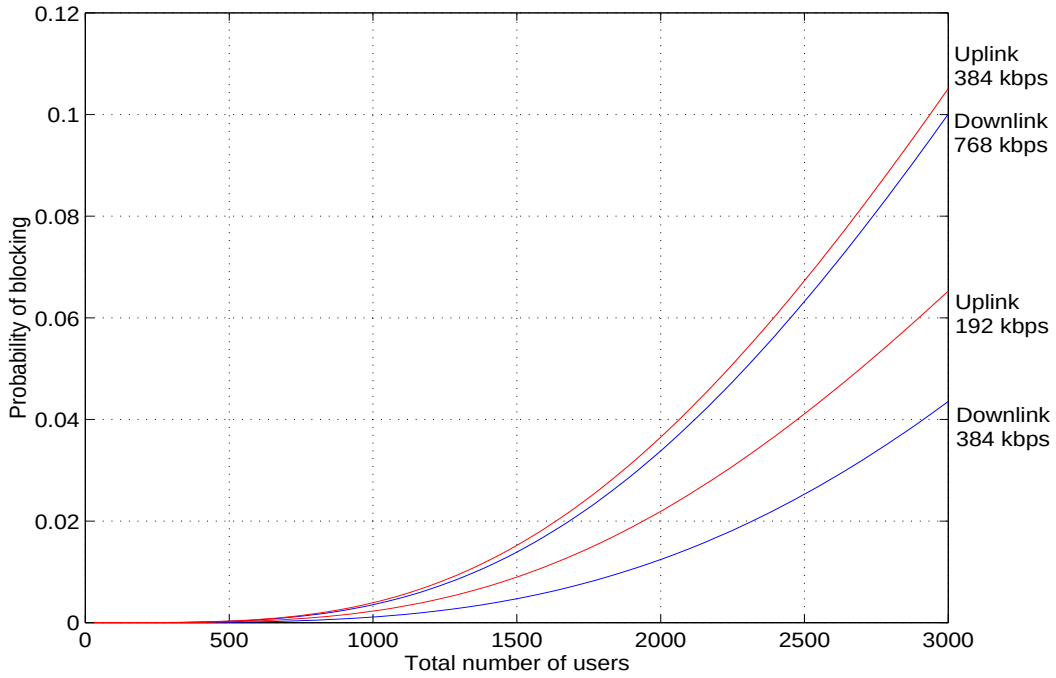

Figure 42: Downlink 768 and 384 kbps; Uplink (red) 384 and 192 kbps, Three Access Servers

Why is the uplink more difficult to handle than the downlink? A possible explanation is that, during a file download, the models for up- and downlink stay almost the same time in the active ON state, and nearly the same number of packets are sent in either direction. The difference is that the packets transferred to the client during a download tend to be larger than the acknowledgment packets sent in the reverse direction. Therefore the time gap between the end of one packet transmission and the beginning of the next is likely to be larger, resulting in a higher number of channel releases and new channel requests when the channel is only kept alive for a certain fixed time after the last packet has been transmitted. This causes a higher utilization of the uplink access servers and with it a higher probability of blocking.

A way to reduce the utilization of the access scheme could be to leave the channel

open for a longer time, waiting for packets to be sent. A welcome side effect would be that latency is reduced because less time consuming channel requests were necessary. However, by doing so, resources are wasted (e.g. spreading codes in a code division multiple access system or frequency bands in a frequency division multiple access system), resulting in a lower available aver all bandwidth.

8.5 Latency due to Restrictions of the Access Server

The time between the completed channel request and the first packet which can be sent is referred to as latency. In our access model, latency is the sum of two time periods: The first is a integer multiple of 3 milliseconds and reflects the random process of assigning a channel dependent on the blocking probability, the second is a constant system dependent parameter describing the time it takes until the first packet can be sent after the channel has been assigned, and assumed to be 10 msec.

When a large number of access servers is used, a channel is almost always assigned within 3 milliseconds: For a 768 kbps downlink, with a confidence of $1 - P_b = 0.95$ and $1 - P_b = 0.97$ the latency equals 13 milliseconds on a system with four and six access servers, respectively.

However, the blocking probability on a system which can only process one request at a time can be as high as 42 percent for 3000 subscribed users. In this case it is likely that a channel request is blocked several times which makes it necessary to consider the backoff strategy of the wireless network when computing the latency.

The system under study uses a binary exponential backoff scheme to reschedule blocked requests. After a station's i^{th} collision, a random number between 0 and $2^i - 1$ is chosen until the packet is sent again. Assuming a constant blocking probability for each slot, a percentile plot of the latency for the single server case is given in Figure 43.

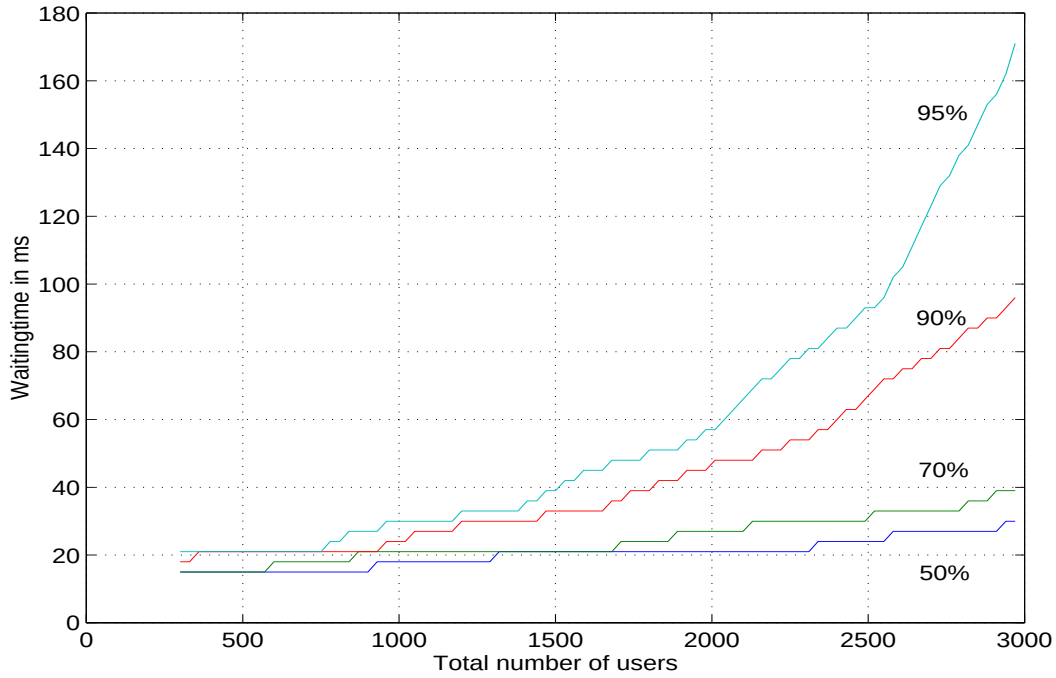

Figure 43: Percentile Plot Latency, 1 access server, 768 kbps

8.6 Determining the Target System Throughput

The network model was designed to evaluate the performance of a wireless access mechanism. No restrictions concerning the total system throughput γ are taken into account. Nevertheless is it possible to determine an approximation of the required bandwidth, referred to as total target system throughput R, which is needed to support N_s customers. This is done using the probability density function of the number of users n in the Transfer ON state, $p(n,\ N_s)$. Assuming that every user in this state is continuously transferring data at rate r, the probability that the total target system throughput R is sufficient for N_s customers can be determined by the relation

$$Prob(\gamma < R \mid N_s) = \sum_{n:\, n<\frac{R}{r}} r\ n\ p(n,\ N_s), \tag{67}$$

where sufficient means that no user is blocked due to bandwidth constraints of the system.

Figure 44 shows the desired total target system throughput for 500, 1000 and 3000 subscribed users. The network is assumed to have 3 access servers, and the numbers are derived for the downlink.

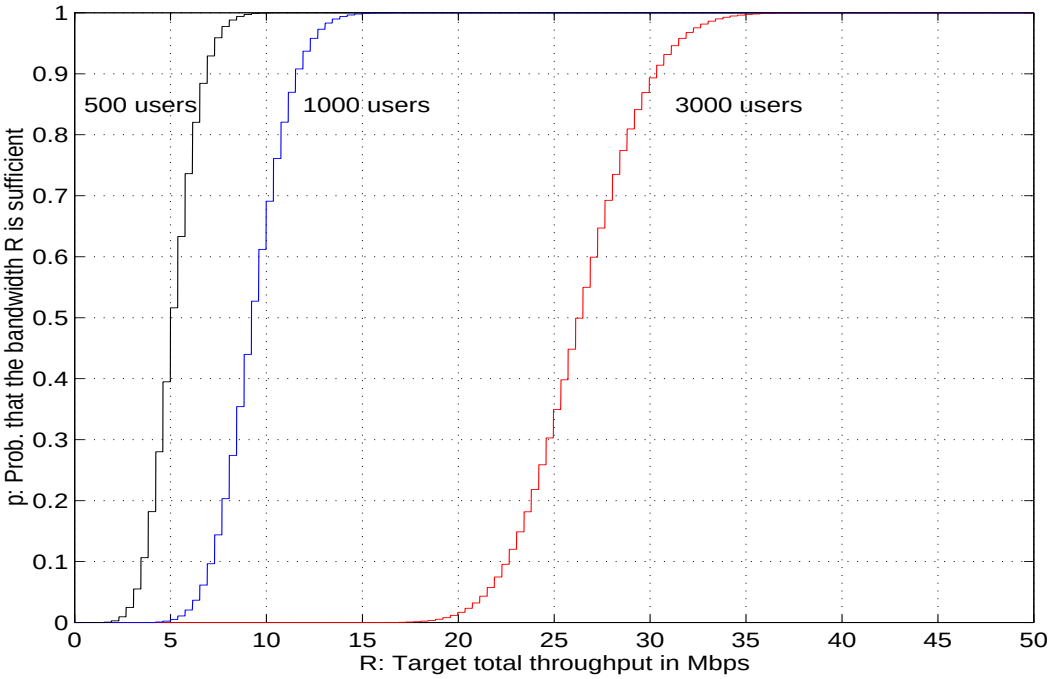

Figure 44: Target Total System Throughput for N = 500, 1000, 3000 User with an Individual Bandwidth of 384 kbps

With a probability of 90 percent, a total bandwidth of 12 Mbps is sufficient to support 1000 customers. For the same confidence, 30.5 Mbps are needed for 3000 customers, which is less then three times the requirement of a 1000 user pool. A reason for this is that the blocking probability is not negligible for 3000 users, which makes it more likely that a user is in the waitstate and not transferring data.

Another remarkable result is that a small difference in the target total system bandwidth leads to a significant change in system performance: Increasing the available bandwidth by 22 percent from 25 to 30.5 Mbps increases the probability that the band-

width is sufficient from 30 to 90 percent.

The required system throughput for different per user bandwidths is shown in Figure 45. Again the downlink of a system with 3 access servers is examined; the individual bandwidth is 384 kbps and 768 kbps, the total number of customers is 3000.

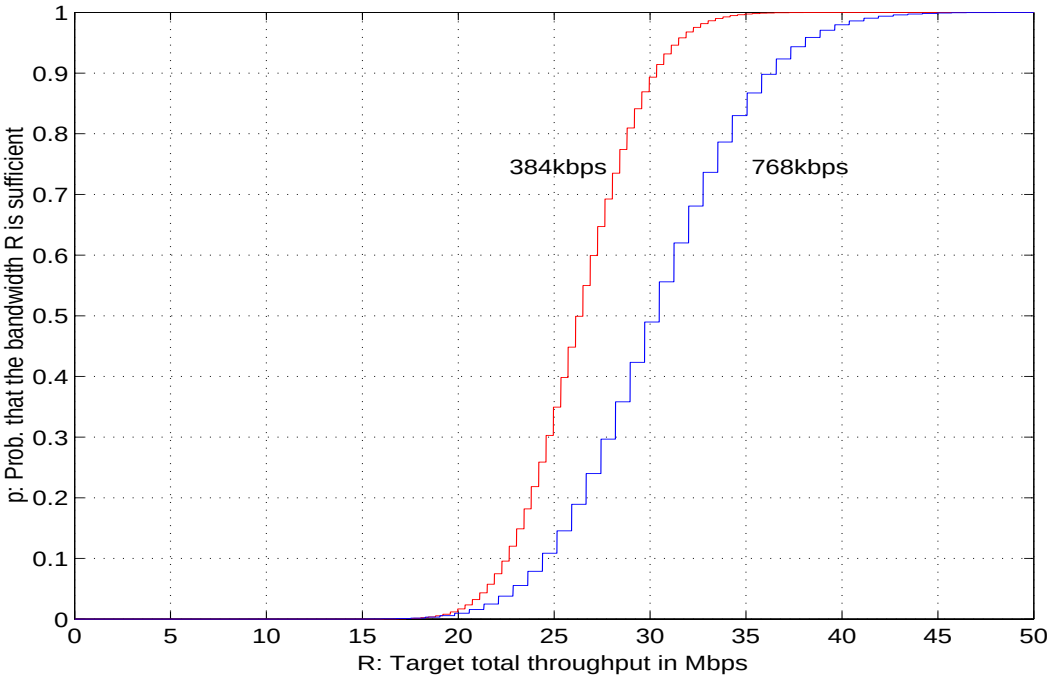

Figure 45: Target Total System Throughput for N = 3000 User and an Individual Bandwidth of 768 kbps and 384 kbps

Figure 45 indicates that the target total system throughput is similar for both link bandwidths. With a probability of 90 percent, about 30.5 Mbps are necessary to support 384 kbps connections whereas 37 Mbps are needed to for link speeds of 768 kbps. One may expect the difference in bandwidth demand to be almost twice as high for individual link speeds of 768 kbps compared to the 384 kbps system. However, taking into account that after a certain bandwidth threshold the user behavior for HTML browsing, and only

HTML traffic is modeled, does not change significantly, the result is realistic.

The plots provided in this chapter, with the main emphasis on the blocking probabilities due to limitations of the access server, are intended to help dimensioning the channel assignment facility in a wireless network. With the Matlab code provided in the Appendix, plots for other network settings can be generated; it may be of interest to examine the influence of different access server speeds or to evaluate the performance of shared access servers for up- and downlink. With minor modifications the model can also be used to describe random access channel request in the IMT-2000 systems UMTS and CDMA2000.

9 Summary and Conclusion

The self-similar behavior of network traffic has been supported by numerous experimental studies. This finding raised the fundamental question whether classical telegraphic statistical modeling methods are still useful.

One contribution of this thesis is to provide a positive answer to this question: An algorithm for approximating long tailed probability distributions by Hyperexponentials was extended to become applicable for general Pareto random processes. Equipped with this tool, an ON/OFF model with long tailed Pareto transition rates was approximated using a large number of exponential stages. We showed that the extended Hyperexponential model is capable of generating traffic which is self-similar over a large time scale. The Hurst parameter calculated with a variance-time plot for an aggregation level of 1000 indicates that the traffic's degree of self-similarity of 32 state Hyperexponential model matches those of the long-tailed ON/OFF model very closely, at least for up to 1000 traffic sources, showing that extended but conventional models can be used to describe long range dependent traffic. This result gives strong support to the use of classical telegraphic methods for analyzing network behavior.

The second contribution of this thesis is a new approach to obtain a Markovian description of a random process: A Maximum Likelihood estimation is used to fit parallel Erlang-K distributions to observed data.

Where the extended Hyperexponential fitting algorithm is intended to approximate one probability distribution by another, the Erlang-K approach is used to directly describe observed data. This is a major advantage when data cannot be accurately described by a conventional distribution function. It was shown that parallel Erlang-K approximations provide very precise fits of both, the moments of a random process and the tail behaviour of the underlying distribution.

An application of the latter fitting technique is introduced in the last part of the thesis: A wireless CDMA network is modeled to derive the performance of its channel assignment procedure by using parallel Erlang-K approximations to describe the processes within the network. These processes, such as channel request interarrival times and packet transmission durations, are obtained using measurements on real networks. For this purpose the traffic analyzing tool *windump* was extended with scripts for an automatic installation and measurement upload procedure, making it possible to collect traffic traces on individual PCs using a cable or DSL Internet connection. *Windump* together with our scripts may be of interest for researcher who want to examine network traffic in a home environment.

Equipped with fitting procedure and network data, a Markovian description of the model is presented, such that standard queueing theory techniques can be used to evaluate the network. Outcomes of the performance analysis are estimations of blocking probabilities and latencies due to restriction of the access server for different network configurations.

A Matlab Source code

A.1 Hyperexponential Fitting Routines

A.1.1 ip_on_weib_to_hyp

The following routine fits a Hyperexponential to a Weibull distribution.

```
function X = ip_on_weib_to_hyp(PHAT)

%
% Input:  PHAT is a vector with two elements;
%         the first element is the Parameter a, the
%         the second the parameter b of the Weibull
%         distribution as defined in the paper.
% Output: X is (step,2) matrix X = [p,lamda] with
%         containing the parameter of the Hyperex-
%         ponential distribution; each row describes
%         a exponential with weightfactor p_i and
%         and waitingtime lamda_i. steps is the
%         order of approximation.


Weib_para_a = PHAT(1)     % Prints out the parameter
Weib_para_b = PHAT(2)     % of Weibull distribution

b = 1.2                   % width between approximation points

steps = 12                % Number of fitting points

                          % initialisation
lamda = zeros(steps,1);   % lamda contains the waiting time of
                          % single exponential;
p = zeros(steps,1);       % p contains the weigthfactors of the
                          % exponentials

von = 0.0001;             % Borders of drawing
bis = 200000;


ap(1) = 10000;            % Borders of approximation
ap(steps) = 0.001;
```

```
first_fit_point = ap(1)         % first
last_fit_point = ap(steps)      % and last fitting point
for i = 2:steps                 % seperation into intervals
   ap(i) = ap(1) * ( exp( log( ap(steps)/ap(1) ) / (1-steps)) )
                                                    ^(-(i-1));
end

% obtaining first parameter lamda and p
lamda(1) = 1/((b-1)*ap(1))*log(getFi_weib(ap(1),1,lamda,p,PHAT)/
                              getFi_weib(b*ap(1),1,lamda,p,PHAT));
p(1) = getFi_weib(ap(1),1,lamda,p,PHAT) * exp(lamda(1)*ap(1));

% iteration to obtain parameter 2 till max-1
for i = 2:(steps-1)
   lamda(i)= 1/((b-1)*ap(i))*log((getFi_weib(ap(i),i,lamda,p,PHAT)
                           /getFi_weib((b*ap(i)),i,lamda,p,PHAT)));

  p(i) = getFi_weib(ap(i),i,lamda,p,PHAT) * exp(lamda(i)*ap(i));
end

% last parameter p using that sum of all p must sum to 1
p(steps) = 1;
for i = 1:(steps-1)
  p(steps) = p(steps) - p(i);
end
% calculation of last lamda
lamda(steps) =1/ap(steps) * log(p(steps) /
                        getFi_weib(ap(steps),steps,lamda,p,PHAT));


% Outprint of original CDF and approximation

% speeding up the output routine by reducing the number of points
% for plotting
t = zeros(10000,1);   % Length(t) = Number pf points when plotting
const = exp( -(log(von/bis)/length(t)));

for i = 1: length(t)
  t(i) =  bis * const^(-i);
end

hyp_ex = 0;           % computing values for approximation
for i = 1:steps
  hyp_ex = hyp_ex + p(i)*exp(-(lamda(i)*t));
end
```

```
hyp_ex = 1-hyp_ex;


figure(3);
subplot(2,1,1);       % Outprint of CDF
loglog(t,hyp_ex,'b', t,(1- exp(- (t./PHAT(1)).^PHAT(2) )),'k:');
grid on;
title('CDF of a Hyperexponential with 12 stages fit
                                          to Weibull(1.12, 0.54)');
xlabel('Time t in seconds; solid: Hyperexponential,
                                                dotted: Weibull');
ylabel('Probability of t_{on} < t');

subplot(2,1,2);       % outprint of relative errors of CDF
semilogx(t,abs(hyp_ex-((1-exp(-(t./PHAT(1)).^PHAT(2))))) ./
                    (min((1-exp(-(t./PHAT(1)).^PHAT(2))),hyp_ex)))
grid on;
xlabel('Time t in seconds');
ylabel('Relative error');

figure(4);
subplot(2,1,1);       % outprint of CCDF
loglog(t,1-hyp_ex,'b', t,1-(1-exp(-(t./PHAT(1)).^PHAT(2))),'k:');
grid on;
title('CCDF of a Hyperexponential with 12 stages fit to
                                            Weibull(1.12, 0.54)');
xlabel('Time t in seconds; solid: Hyperexponential, dotted:
                                                    Weibull');
ylabel('Probability of t_{on} > t');

subplot(2,1,2);       % outprint of relative errors of CCDF
semilogx(t,abs(1-hyp_ex-(1-(1-exp(-(t./PHAT(1)).^PHAT(2))))) ./
           (eps+min(1-(1-exp(-(t./PHAT(1)).^PHAT(2))),1-hyp_ex)))

grid on;
xlabel('Time t in seconds');
ylabel('Relative error');

lamda_hyp = lamda     % output of computed parameters
p_hyp = p

mom_1_hp = p'*(lamda.^-1)       % output of moments
mom_2_hp = 2*p'*(lamda.^-2)
mom_3_hp = 6*p'*(lamda.^-3)
mom_4_hp = 24*p'*(lamda.^-4)
```

```
mom_5_hp = 120*p'*(lamda.^-5)

var_hp = mom_2_hp-mom_1_hp^2
X = [p,lamda];        % return values of Hyperexponential


function [X] = getFi_weib(x,i,lambd,p,PHAT)

% used in the Feldmann fittingrourine
% computes the difference between the inverse
% CDF of Weibull(a,b) and the Hyperexponential
% with the parameters given in the vectors
% lambd and p

a = PHAT(1);              % Weibull parameter a
b = PHAT(2);              % Weibull parameter b
help = 0;

% in the for loop the value of x pluged in the
% already existing part of the Hyperexponential "CCDF" is computed
for k = 1:(i-1)
 help = help +  p(k) * exp(-(x*lambd(k)));
end

% The difference between the Weibull CCDF and the Hyperexponential
% CCDF (or a part of the Hyperexponential if it is not already
% completely developed) is computed and returned.
X = (exp(-((x./a).^b))) - help;
```

A.1.2 ip_on_par_to_hyp

The following routine fits a Hyperexponential to the shifted version of a Pareto distribution. The time shift is reversed by concatenating the Hyperexponential and an Erlang-K distribution.

```
function X = ip_on_weib_to_hyp(PHAT)

%
% Input:  PHAT is a vector with two elements;
%         the first element is the Parameter a, the
%         the second the parameter b of the Weibull
%         distribution as defined in the paper.
% Output: X is (step,2) matrix X = [p,lamda] with
%         containing the parameter of the Hyperex-
%         ponential distribution; each row describes
```

```
%         a exponential with weightfactor p_i and
%         and waitingtime lamda_i. steps is the
%         order of approximation.


Weib_para_a = PHAT(1)     % Prints out the parameter
Weib_para_b = PHAT(2)     % of Weibull distribution

b = 1.2                   % width between approximation points

steps = 12                % Number of fitting points

                          % initialisation
lamda = zeros(steps,1);   % lamda contains the waiting time of
                          % single exponential;
p = zeros(steps,1);       % p contains the weigthfactors of the
                          % exponentials

von = 0.0001;             % Borders of drawing
bis = 200000;


ap(1) = 10000;            % Borders of approximation
ap(steps) = 0.001;

first_fit_point = ap(1)         % first
last_fit_point = ap(steps)      % and last fitting point
for i = 2:steps                 % seperation into intervals
   ap(i) = ap(1) * ( exp( log( ap(steps)/ap(1) ) / (1-steps)) )
                                                      ^(-(i-1));
end

% obtaining first parameter lamda and p
lamda(1) = 1/((b-1)*ap(1))*log(getFi_par(ap(1),1,lamda,p,PHAT)/
                              getFi_par(b*ap(1),1,lamda,p,PHAT));
p(1) = getFi_par(ap(1),1,lamda,p,PHAT) * exp(lamda(1)*ap(1));

% iteration to obtain parameter 2 till max-1
for i = 2:(steps-1)
   lamda(i)= 1/((b-1)*ap(i))*log((getFi_par(ap(i),i,lamda,p,PHAT)
                           /getFi_par((b*ap(i)),i,lamda,p,PHAT)));

  p(i) = getFi_par(ap(i),i,lamda,p,PHAT) * exp(lamda(i)*ap(i));
end
```

```
% last parameter p using that sum of all p must sum to 1
p(steps) = 1;
for i = 1:(steps-1)
  p(steps) = p(steps) - p(i);
end
% calculation of last lamda
lamda(steps) =1/ap(steps) * log(p(steps) /
                        getFi_par(ap(steps),steps,lamda,p,PHAT));


% Outprint of original CDF and approximation

% speeding up the output routine by reducing the number of points
% for plotting
t = zeros(10000,1);   % Length(t) = Number pf points when plotting
const = exp( -(log(von/bis)/length(t)));

for i = 1: length(t)
  t(i) =  bis * const^(-i);
end

hyp_ex = 0;           % computing values for approximation
for i = 1:steps
  hyp_ex = hyp_ex + p(i)*exp(-(lamda(i)*t));
end
hyp_ex = 1-hyp_ex;

figure(1);

subplot(2,1,1);     % outprint of CDF (original and approximation)
loglog(t+PHAT(2),hyp_ex,'b',t,1 - (PHAT(2)./(t)).^(PHAT(1)),'k:');
grid off;
title('CDF of a Hyperexponential with 7 stages fit to flat part
                                        of Parto(0.82, 0.0016)');
xlabel('Time t in seconds; solid: Hyperexponential, dotted:
                                                    Pareto');
ylabel('Probability of t_{on} > t');

subplot(2,1,2);     % outprint of relative errors of CDF
semilogx(t,abs( hyp_ex-(1-(PHAT(2)./(t+PHAT(2))).^(PHAT(1)))) ./
              min((eps+min( 1- (PHAT(2)./t).^(PHAT(1)),hyp_ex))));
grid on;
xlabel('Time t in seconds');
ylabel('Relative error');
```

```
figure(2);         % outprint of CCDF (original and approximation)

subplot(2,1,1);
loglog(t+PHAT(2),1-hyp_ex,'b',t,1-(1-(PHAT(2)./(t))
                                                .^(PHAT(1))),'k:');
grid off;
title('CCDF of a Hyperexponential with 7 stages fit to flat part
                                            of Parto(0.82, 0.0016)');
xlabel('Time t in seconds; solid: Hyperexponential, dotted:
                                                          Pareto');
ylabel('Probability of t_{on} < t');

subplot(2,1,2);          % outprint of relative errors
semilogx(t, abs((1-hyp_ex)-(1-((1-(PHAT(2)./(t+PHAT(2))).^
(PHAT(1))))).(eps+min(1-(1-(PHAT(2)./t).^(PHAT(1))),(1-hyp_ex))))
axis([10^0 10^7 0 0.05])
grid on;
xlabel('Time t in seconds');
ylabel('Relative error');


lamda_hyp = lamda       % output of computed parameters
p_hyp = p
X = [p,lamda];


function [X] = getFi_par(x,i,lamd,p,PHAT)
% used in the Feldmann fittingrourine
% computes the difference between the inverse CDF of Pareto(a,k)
% and the Hyperexponential with the parameters given in the
% vectors lamd and p a delay of PHAT(2) is necessary to obtain
% the correct approximation

a = PHAT(1);
b = PHAT(2);
help = 0;
for k = 1:(i-1)
 help = help +  p(k) * exp(-(x*lamd(k)));
end

X = (b/(x+b)).^(a) - help;      % 1 - f(x) - help
```

A.2 Maximum Likelihood Estimation

A.2.1 ml_fit_05

The following routine computes the Maximum Likelihood estimation of the parameters for a parallel Erlang-K description. The function *objfun_05* is called; it derives the numerical result of Equation 46. After the fit is obtained, the first five moments of the original data and of the Erlangian approximation are calculated.

```
function [XX, kk, fname] = ml_fit_05

% fits data to parallel flexible structured Erlang K

k = [2,2,3,5,5];                    % order of parallel
                                    % Erlang K stages
 fname = 'acc_in_trans_384_01';     % Name specifying the File
                                    % containing the time intervalls
                                    % which are to describe

 % initial guess for parameter
x0 = [0.1 1.0 0.1 1.8 0.3 1.0 0.1 2.1 0.1 0.5 0.1 0.5];
k = [1,1,4,4,6,6];

 h = load(fname);                   % loads the sample vector

 constraints_vec = [1,0];           % defines the vecor whis is
 for i = 2:length(k)                % used to force that all p
                                    % must sum up to 1
    constraints_vec = [constraints_vec , [1,0]];
 end

 t =h.dT;       % copying the datastructure given in h in an array

options = optimset(options,'GradObj','on');
% 'GradObj','on': use the gradients as supplied in the objfun
%                                                        routine

% defines that all parameter must be bigger than 0
lb = zeros(1,length(x0))+2*eps;
% no upper bound (but still all p must sum to 1)
ub = [];

% Starts the optimization process
```

```
[x,fval,exitflag,output] = fmincon('objfun_05',x0,constraints_vec,
                                   [1],[],[],lb,ub,[],options,h,k);
% print out the obained values:
fval
output

% comutation of the 1st to 6th central moment
fitt_mean = 0;
i = 1;
for s = 1:(length(x)/2)
   fitt_mean = fitt_mean + x(2*s-1)*factorial(i+k(s)-1) /
                                   factorial(k(s)-1)*x(2*s)^(-i);
end
fitt_mean_1 = fitt_mean
samp_mean = mean(t)

fitt_mean = 0;
i = 2;
for s = 1:(length(x)/2)
   fitt_mean = fitt_mean + x(2*s-1)*factorial(i+k(s)-1) /
                                   factorial(k(s)-1)*x(2*s)^(-i);
end
fitt_mean_2 = fitt_mean
samp_m2 = var(t)+(mean(t)^2)

fitt_mean = 0;
i = 3;
for s = 1:(length(x)/2)
   fitt_mean = fitt_mean + x(2*s-1)*factorial(i+k(s)-1) /
                                   factorial(k(s)-1)*x(2*s)^(-i);
end
fitt_mean_3 = fitt_mean
samp_m3 = 1/(length(t)-1)*ones(1,length(t))*((t.^3)')

fitt_mean = 0;
i = 4;
for s = 1:(length(x)/2)
   fitt_mean = fitt_mean + x(2*s-1)*factorial(i+k(s)-1) /
                                   factorial(k(s)-1)*x(2*s)^(-i);
end
fitt_mean_4 = fitt_mean
samp_m4 = 1/(length(t)-1)*ones(1,length(t))*((t.^4)')

fitt_mean = 0;
i = 5;
```

```
for s = 1:(length(x)/2)
   fitt_mean = fitt_mean + x(2*s-1)*factorial(i+k(s)-1) /
                                    factorial(k(s)-1)*x(2*s)^(-i);
end
fitt_mean_5 = fitt_mean
samp_m5 = 1/(length(t)-1)*ones(1,length(t))*((t.^5)')

fitt_mean = 0;
i = 6;
for s = 1:(length(x)/2)
   fitt_mean = fitt_mean + x(2*s-1)*factorial(i+k(s)-1) /
                                    factorial(k(s)-1)*x(2*s)^(-i);
end
fitt_mean_6 = fitt_mean
samp_m6 = 1/(length(t)-1)*ones(1,length(t))*((t.^6)')

XX = x;             % prints out the parameter of the fit
kk = k;
```

A.2.2 objfun_05

In the following function, the numerical result of Equation 46 is derived.

```
function [f, G] = objfun_05(x,h,k)

% x contains the parameters wich are varied;
% h contains the sample vector
% k describes the number of exponential stages
% in each parallel exponential stage


alpha = x(1:2:end);  % separating the parameter given in x
                     % intp beta representing the transistion
beta = x(2:2:end);   % rate and alpha representing the probability


t =h.dT ;       % copying the datastructure given in h in an array
md = 0 ;        % if the error massage log of zero occures set md
                                                        to eps


% computes the evaluation of the function which is to maximize;
q = (((alpha.*(beta.^k)./fact_vec(k-1))) * (exp((k-1)'*log(t)) .*
                                              exp(-(beta)'*t)));
```

```
% gives back the negative sum to that the minimization algorithm
% can be used for maximisation
f = -(ones(1,length(q))*log(q+md)');

% computes the gradiant at for the parameter
% (variables in this case) for given h and k
G_help = zeros(1,length(x));
for u = 1:length(k);
  fg = (beta(u)^k(u)/factorial(k(u)-1) .* t.^(k(u)-1) .
                                        *exp(-beta(u).*t))./(q+md);
  G_help(2*u-1) = ones(1,length(fg))*fg' ;
  fg = ((alpha(u)/factorial(k(u)-1)) .* (k(u)*beta(u)^(k(u)-1)-t *
         beta(u)^(k(u))) .*exp(-beta(u)*t) .* t.^(k(u)-1))./(q+md);
  G_help(2*u)   = ones(1,length(fg))*fg' ;
end

G = -G_help;            % gives back the negative value so that the
                        % minimization algorithm can be used for
                        % maximization
```

A.3 Outer ON/OFF Model

A.3.1 outer_on_off_get_para

The following routine computes the steady state probabilities for being in active OFF and active ON state based on the traffic trace saved in the file **masterincoming.ext**.

```
function outer_on_off_get_para

% separate user in the input stream are separated by (0,0)
% data is fit to longtailed distributions describing the data as
% a two state ON/OFF model. For the outer ON/OFF process

offset = 12;                        % determines the maximum time
                                    % between to packets for which
                                    % the user is still assumed to
                                    % be in the ON state
S1 = load('masterincoming.ext');    % load S with incoming trace

number_of_datapoints_incoming = length(S1);
inlength = ones(1,length(S1))* S1(:,2);   % print out sum of sizes
                                          % ofthe recorded packets
S2 = load('masteroutgoing.ext');    % load S with outgoing trace
```

```
number_of_datapoints_outgong = length(S2);
outength = ones(1,length(S2)) * S2(:,2)  ;

S = S2; % CHOOSE IF PROCESSING IN OR OUTGOING DATA

duration = S(end,1) - S(1,1);       % print out duration of trace

% counter variables for loops
k = 1;
i = 1;
j = 1;

% determing the duration of the ON and OFF periods:
 while k < length(S(:,1)),     % while the index k is smaller than
                               % the length of the sample vector S
   if S(k,1) == 0              % do avoiding to record data
      k = k+1   ;              % twice of the user changes
      %i = i-1;
      %j = j-1;
   end
   help = S(k,1);              % remember the possile start of an
                               % ON perid

   while ((k<length(S(:,1))) & (S(k+1,1)-S(k,1) < offset) &
                                                 not(S(k+1,1) == 0))
      k = k+1   ;      % repeat until two consecutive packeges are
                       % seperated by a period longer than offset
                       % seconds or until end of string is reached
                       % erg = S(k+1,1)-S(k,1)
   end
   k = k
   S(k,1);
   if ((S(k+1)-S(k))>offset)   % if two consecutive packeges are
                               % seperated by a period longer than
      kold = k;                % offset seconds find the OFF time
      %while (((k+1)<length(S(:,1))) & (S(k+2,1)-S(k+1,1)>offset))
       % k = k+1;              % singel packets within a 15 second
                               % interval are ignored (not ending
      %end                     % the off period)

      if S(k+1)-S(kold) < 900
         toff(j) = S(k+1)-S(kold);    % off time appended to the
         j = j+1;                     % vector toff
      end
```

```
   end

   if ((S(k) - help)>0)        % if there is more than one packet
      ton(i) = S(k) - help;    % to send on periods are appended
      i = i+1;                 % to the vector ton
   end
   k = k+1;
end

toff = toff(1:end)              % the last on and off periods are
ton = ton(1:end)                % not detected

t_off_sum = sum(toff);
t_on_sum = sum(ton);

p_aktive = t_on_sum / (t_off_sum + t_on_sum)

% Estimating the parameter
% Parameter for ON period, Weibull distributed
 [PHAT_on] = Weibfit(ton);  % fit Weibull to inter transfer time
% Print out results for Weibullfit:
Weib_on_a = PHAT_on(1)
Weib_on_b = PHAT_on(2)


% Parameter for ON period, Pareto distributed

% toff normalized so that toff >= 1
% to use Jain's Max. Likelyhood method
% Maximum likelyhood estimate for parameter of Pareto distribution
ton_sum =  log(ton/1) * ones(length(ton),1);
% Print out result of Paretofit
a_Pareto_on = 1/ ( 1/length(ton) * ton_sum )
b_Pareto_on = offset

% Parameter for OFF period, Pareto distributed

% toff normalized so that toff >= 1
% to use Jain's Max. Likelyhood method
% Maximum likelyhood estimate for parameter of Pareto distribution
toff_sum =  log(toff/offset) * ones(length(toff),1);
% Print out result of Paretofit
a_Pareto_off = 1/ ( 1/length(toff) * toff_sum )
b_Pareto_off = offset
```

A.3.2 on_off_sim

This routine produces a vector representing the number of users active over time using a two state ON/OFF model with long tailed state durations; it can be emploied to simulate user behavior for a large number of sources.

The obtained vector is used in the routine aggreg to test for the degree of self-similarity.

```
function on_off_sim

user = 100;
gran = 1;                       % points per  second
tmax = 200000;                  % time in seconds

t = zeros(1,(gran*tmax));
counterh = 0;

told = 0;
ton = 0;
toff = 0;
for n = 1:user
   n
   told = 0;
   while told<tmax,
     counterh = counterh +1;
      dton = 5*(1/(1-rand(1,1)))^(1/1.2);
     dtoff = 60*(1/(1-rand(1,1)))^(1/1.5);          % works fine!
     %ton(counterh) = dton;
     %toff(counterh) = dtoff;
     a=round(told*gran)+1;
     b=round((told+dton)*gran);
     if (told+dton) < tmax
        t(a:b) = t(a:b) + 1;
     else
        t(a:end) = t(a:end) + 1;
     end
     told = told+dton+dtoff;
    end
end

t = t(100001:end);
save on_off_sim_nou_02 t;
%ton = sum(ton)
%toff = sum(toff)
```

```
counter = counterh
```

A.3.3 on_off_hyper_sim

This routine produces a vector representing the number of users active over time using a two state ON/OFF model with Hyperexponential distributed state durations; it can be emploied to simulate user behavior for a large number of sources. Uses the random generator **hyperrnd**.

The obtained vector is used in the routine aggreg to test for the degree of self-similarity.

```
function on_off_hyper_sim

user = 100;

gran = 1;           % points per  second
                    % time in seconds
tmax = 200000;

[phat] = ip_off_par_to_hyp2_for_vt([1.5,5],5);
pon=phat(:,1);
lambdaon=phat(:,2);

[phat] = ip_off_par_to_hyp2_for_vt([1.2,60],60);
poff=phat(:,1);
lambdaoff=phat(:,2);

t = zeros(1,(gran*tmax));
counterh = 0;

told = 0;
ton = 0;
toff = 0;
for n = 1:user
   n
   told = 0;
   while told<tmax,
     counterh = counterh +1;
     dton = hyperrnd(pon,lambdaon)+5;
     dtoff = hyperrnd(poff,lambdaoff)+60;
     a=round(told*gran)+1;
     b=round((told+dton)*gran);
     if (told+dton) < tmax
        t(a:b) = t(a:b) + 1;
```

```
     else
        t(a:end) = t(a:end) + 1;
     end
     told = told+dton+dtoff;
    end
end

t = t(100001:end);
save on_off_hyper_sim_nou_02 t;
ton = sum(ton)
toff = sum(toff)
counter = counterh


function X = hyperrnd(p,lambda);

p = cumsum(p);
p = p;
help = rand;

for i=1:length(p),
   if help<p(i)
      X = exprnd(1/lambda(i));
      break;
   end
end
```

A.3.4 var_time_plot

The following routine produces a variance-time plot of the datavecor which is loaded.

```
function var_time_plot

h = load('on_off_hyper_sim_nou_02');
t = h.t;
lt = length(t)
vart = var(t)
st = 10
for i = st:40   % 35:62 was good
   i
   m(i) = round(10^(0.1*i));
   tagg = 0;
   for j = 1:floor(lt/m(i))
      tagg(j) = sum(t( (j-1)*m(i)+1 : j*m(i)) )/m(i);
   end
```

```
   length(tagg)
   taggv(i) = log10(var(tagg)/vart);
end
%plot(t(1:100:end))
figure(3);
plot(log10(m(st:end)),taggv(st:end),'x')
%lsline
grid on
title('Variance-time plot');
xlabel('log10(m)');
ylabel('log10(Normalized variance)');
```

A.4 Access Model

A.4.1 extract_2

In the following routine a measured traffic trace is used to generate a time series which can be used to describe the access model. Moments of the data are computed.

```
function extract_2

S1 = load('masterincoming.ext');      % load S with incoming trace

length(S1);
inlength = ones(1,length(S1)) * S1(:,2);  % print out sum of sizes
                                          % of recorded packets

S2 = load('masteroutgoing.ext');      % load S with outgoing trace

length(S2);
outength = ones(1,length(S2)) * S2(:,2) ;

time = S1(1,1)-S1(end,1);


% outlength = ones(1,length(S)) * S(:,2) % print out sum of sizes
                                         % of recorded packets

% throwing away last and first values (sometimes not  recorded
% correctly) and taking into account that \textit{tcpdump} only gives
% out the length of the payload
```

```
S1 = [ S1(2:length(S1)-1,1), S1(2:length(S1)-1,2)+24 ]; %
S2 = [ S2(2:length(S2)-1,1), S2(2:length(S2)-1,2)+24 ]; %

S = S1;     % CHOOSE IF PROCESSING IN OR OUTGOING DATA


duration = S(end,1) - S(1,1) ;       % print out duration of trace

fixdelay = 0.012                     % time lost in access scheme
bandwidth = 192000/8;                % average bandwith of trans-
                                     % mitter in byte per second
offset = 10;

gap = 0.03                           % time waited in send state
                                     % for another packet to come
n = 1;                               % counter variables for loops
k = 1;

while k <= (length(S)),          % loop to find time for going
  if not(S(k,1) == 0)            % from off to on (Tar)
                                 % and from ON to OFF ( T2 )

  Tar(n) = S(k,1);           % OFF to ON request when arrival
                             % occures while not sending
                             % the absolute time of nth request
                             % is recorded in Tar(n)
  PLength(n) = S(k,2);       % record packet length

  dTs = fixdelay + PLength(n)/bandwidth; % dTs is the time from
                                         % access request till
                                         % packet transfer of the
                                         % first packet ends
  T2(n) = Tar(n) + dTs;           % absolute time of end of packet
                                  % transfers is recorded in T2(n)
  k = k+1;                        % loop counter increased

  % Packets are grouped together when generated while sending
  % and within a certain time after sending:
  while ((k<=(length(S)))&(S(k,1)<(T2(n)+ gap))& not(S(k,1)==0)),
    Tgap = 0;
    if S(k,1) > T2(n);                   % Tgap is time waited for
      Tgap = S(k,1) - T2(n);             % a possible arrival
    end
    PLength_new(n) = S(k,2);
    PLength(n) = PLength(n)+PLength_new(n);
```

```
    dTs = PLength_new(n)/bandwidth + Tgap;
    T2(n) = T2(n) + dTs;          % abs time of end of transfer of
    k = k+1;                      % actual packet is written in T2
  end
  n = n+1;
  else
  k = k+1;
  end
end

% Time delta between download and next request:
for n = 1 : (length(Tar)-1)
dT1(n) = Tar(n+1) - T2(n);
end

% Time deltas for download:
for n = 1 : (length(T2))
    dT2(n) = T2(n) - Tar(n) - fixdelay;
end

%Print out first 30 computed times:
downloadtime_dT2 = dT2(1:30)
interdownloadtiem_dT1 = dT1(1:30)
Packetlength = PLength(1:30)

% sorting out the times for not beeing in the active state:
n = 1;
for i = 2 : length(dT1)
  if ((dT1(i) < offset) & (dT1(i) > 0))
                              % if a timegap greater than 15
                              % seconds occures the corresponding
                              % deltaT is excluded (it belongs to
                              % the inactive state)
    dT1_on(n) = dT1(i);
    n = n+1 ;
  end
end

length_data = length(dT1_on)

dT = dT1_on;
save acc_in_inter__01 dT;

dT = dT2;
save acc_in_trans__01 dT;
```

```
mom_1_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^1)')
var_inter = var(dT1_on)
mom_2_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^2)')
mom_3_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^3)')
mom_4_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^4)')
mom_5_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^5)')
mom_6_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^6)')
mom_7_inter = 1/(length(dT1_on)-1)*ones(1,length(dT1_on))*
                                                    ((dT1_on.^7)')
mom_1_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^1)')
var_trans = var(dT2)
mom_2_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^2)')
mom_3_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^3)')
mom_4_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^4)')
mom_5_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^5)')
mom_6_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^6)')
mom_7_trans = 1/(length(dT2)-1)*ones(1,length(dT2))*((dT2.^7)')
```

A.4.2 single_on_prob_7

In the following function, the Markovian state transition matrix describing the access model is set up and the steady state probabilities for a single user is derived, assuming that the blocking probability is known.

```
function [p_act_off, p_acc_wait, p_acc, p_send] =
                    single_on_prob_7(p_block,para_pro,l,para_int,k)

% l represents the structure of the approximation of the Erlang K
% distributon for of the packet sending time. For example
% l = [2,1,4] consists of 2 Erlang K=1, 1 Erlang K=2 and 4 Erlang
%  K=3
% weight factor and service rate of Erlang K approximation for packet
% interarrival time

send_offset = 0.0025; % needed to transfer a minimumsized packet
                      % 24 byte, 384kbit/5
```

```
% p_int contains the parameter of the inter transfer time
% approximation of the downlink p_pro contains the parameter
% of the download time approximation of the downlink


% waiting time for exponentials of inter transferetime
lamda = para_int(2:2:length(para_int));

% weight factors for exponentials of inter transferetime
p = para_int(1:2:length(para_int));

% waiting time for exponentials of transfer duration
mu = para_pro(2:2:length(para_pro));

% weight factors for exponentials of inter transferetime
q = para_pro(1:2:length(para_pro));

Nd = 100;   % number of states for Erlang-K for deterministic
            % access scheme
Nt = 100;   % number of states for Erlang-K for threshold

Td = 0.012; % time "request access" to "begin of transfer"
Tt = 0.0016;    % steep part of Pareto representing minimum
                % packet size

delta = Nd/Td; % computation of service rates for
trash = Nt/Tt; % exponential series

Pb = p_block;   % probability of blocking
Tb = 0.03; % blocking waiting time


kb = sum(k)+2;                % short hand for state representing
                              % the blocking probability
Ns = kb+Nd+Nt;                % short hand for state representing
                              % the last state of steep
                              % part download time
Q = zeros(Ns+sum(l),Ns+sum(l));% initialize transistion matrix

% writing the transistion probabilities in the Q Matrix:
% setting up the Erlangien description of the interarrival time
for j = 1:length(k)           % for Erlang K, K = 1..length(k)
   for i = 1:(k(j)-1)         % transistion to following state
                              % within one Erlangian stage
```

```
        Q(sum(k(1:(j-1)))+i,sum(k(1:(j-1)))+i+1) = lamda(j);
   end
                        % transistion from last exponential state
                        %  of Erlangian stage to access servers
    Q(sum(k(1:(j))),kb-1) = lamda(j);
end

% setting transistions to waitstate if access server blockes
Q(sum(k)+1,kb) = Pb*delta;          % Pb is the probability that
Q(sum(k)+1,kb+1) = (1-Pb)*delta;    % all servers are busy
Q(sum(k)+2,sum(k)+1) = 1/Tb;        % the waiting time Tb is set


for i = 1:(Nd)           % Exponentials representing the waiting
                         % time while in the access server
    Q(kb+i,kb+i+1) = delta;
end

for i = 1:(Nt-1)         % Exponentials representing the steep
                         % part of the downloadtime
   Q(kb+Nd+i,kb+Nd+i+1) = trash;
end

for i = 1:length(l)                % Transistion to the flat
                                   % part of the downloadtime
    Q(Ns,Ns+sum(l(1:(i-1)))+1) = trash*q(i);
end

for j = 1:length(l)
    for i = 1:(l(j)-1)
        Q(sum(l(1:(j-1)))+Ns+i,sum(l(1:(j-1)))+Ns+i+1) = mu(j);
   end
   for m = 1:length(k)
        Q(Ns+sum(l(1:(j))),1+sum(k(1:(m-1)))) = mu(j)*p(m);
   end
end

% computing the diagonal of the matrix; each row of the matrix
% must sum to one
help = Q * ones(Ns+sum(l),1);
for i= 1:(Ns+sum(l))
  Q(i,i) = -help(i);
end
                                   % Q has rank length(Q_column)-1
QQ = [Q';ones(1,Ns+sum(l))];       % the fact that the resulting
```

```
                                    % probability vector must sum
                                    % to one is used; the equation
                                    % pi_1+...pi_n =1 is appended
p_ss = QQ\[zeros(Ns+sum(l),1);1];  % solving the matrix equation


% printing out the results:
                                % probability for beeing in the...
p_act_off = ones(1,sum(k))*p_ss(1:sum(k)); %... active off state

p_acc_wait = ones(1,1)*p_ss(kb:kb);       %... access waite state
                                          % due to blocking

p_acc = ones(1,Nd)*p_ss(kb+1:kb+Nd)/4;    % ... access state
                      % the prob. is devided by 3 because we are
                      % interested in only in the three four
                      % of the access process miliseconds
% probability for beeing in the transfer state
p_send = ones(1,sum(l))*p_ss(Ns+1:Ns+sum(l)) + ones(1,Nt) *
                                          p_ss(kb+Nd+1:kb+Nd+Nt);
```

A.4.3 erg

In the following routine the steady state probabilities of being in the Transfer OFF, Transfer ON, Wait and Access state are derived for different numbers of users in the system.

```
function erg = find_block

% determines the blocking probability for the traffic model

% parameter of Erlangian approximation

l = [1,1,4,4,6,6];          % acc_in_trans_384_01:

k = [2,2,4,4,6,6];          % acc_in_inter_384_01:
nu_max = 500;               % Number of users in active state
ns = 3;                     % Number of servers

erg_mat_act = zeros(nu_max,5);

for nu = 1:nu_max

% Initialization:
p_b = 0.1;          % start value for probability
```

```
p_b_alt = 1;        % start value for comparison of
                    % consecutive iterations
i = 0;              % iteration counter

%iteration for finding p_b
while ((abs(p_b_alt - p_b) > p_b/1000) & (i < 20)),
   % Repeat until desired precision is reached, i.e. until
   % blocking probability in model equals the blocking
   % probability for nu users and ns servers

   i = i+1;
   if i == 20
      i = i
   end
   p_b_alt = p_b;

   % for a blocking probablity p_b find the probability of
   % beeing in the access server
   [p_act_off, p_acc_wait, p_acc, p_send] =
                          acc_mod_prob(p_b,para_pro,l,para_int,k);

   %p_acc = p_acc                    % for debugging purpose
   %p_acc_wait = p_acc_wait

   % find the blocking probability for nu users and ns servers
   % with the new probaility for beeing in the access server
   p_b = acc_mod_comp_block(p_acc,nu,ns);
   %P_b_over_all = p_b
end

p_act_off = p_act_off;
p_acc_wait = p_acc_wait;
p_acc = p_acc;
p_send = p_send

computing_for_nu = nu
erg_mat_act(nu,1) = p_b;
erg_mat_act(nu,2) = p_acc_wait;
erg_mat_act(nu,3) = p_send;
erg_mat_act(nu,4) = p_acc;
erg_mat_act(nu,5) = p_act_off;
end
erg = erg_mat_act;
save erg_act_out_3_192_test erg;
```

B Dos Scripts and Installation Routines

B.1 wpiwinnt.bat

This batch file for Win2000/NT starts the whole data collection and file processing.

- Change into C:/Wpiipres directory
- Launch userinfo.exe if userinfo.exe exists to collect user information
- Call tokennt.bat, which defines variables used for file processing
- Call wpidumpnt.bat, which processes the data files and uploads them to the WPI-IP Research FTP site
- Rename userinfo.exe to userid.exe, so that userinfo.exe only runs once
- Delete unnecessary ZIP files to clean the directory
- Launch windump to collect IP Packets and save data into a text file

```
@echo wpiwin.bat
cd c:\wpiipres
if exist userinfo.exe call userinfo.exe
call tokennt.bat
call wpidumpnt.bat
rename userinfo.exe userid.exe
del *.zip
del *.ZIP
windump -n -q -tt ip and port 80 >>%Computername%%Username%.txt
```

B.2 wpiwin98.bat

This batch file for Win98 starts the data collection and file processing.

- Change into C:/Wpiipres directory
- Launch userinfo.exe if userinfo.exe exists to collect user information
- Call wpidump98.bat, which processes the data files and uploads them to the WPI-IP Research FTP site
- Rename userinfo.exe to userid.exe, so that userinfo.exe only runs once

- Delete unnecessary ZIP files to clean up the directory
- Launch windump to collect IP Packets and save data into a text file

```
@echo Off
cd c:\wpiipres
if exist userinfo.exe call userinfo.exe
call wpidump98.bat
rename userinfo.exe userid.exe
del *.zip
windump -n -q -tt ip and port 80 >>data.txt
```

B.3 tokennt.bat

This batch file for Win2000/NT defines ID informationused in addidnt.bat.

- Set IP Address into variable "IP"
- Set system date into variable "StrDate"
- Set system time into varible "StrTime"
- Set both system time and username into variable
- "UnqStr" for use as Zip file name in wpidump.bat
- Set both system date and time into variable "ntdate"
- Add IP Address to user_id.txt

```
@echo off
for /f "skip=5 tokens=1,2,12*" %%A in ('ipconfig') Do set %%A=%%C
for /f "tokens=2" %%I in ('Date /T') Do set StrDate=%%I
for /f "tokens=1,2* delims=:" %%K in ('Time /T') Do set StrTime=%%K%%L
set UnqStr=%StrTime%%Username%
set ntdate=%StrDate%%StrTime%
```

B.4 wpidumpnt.bat

This batch file for Win2000/NT gets text files ready for upload to the FTP site and cleans up the directory.

- Call addidnt.bat, which adds ID information to the original data text file
- Compress user_id.txt file into a ZIP file. Variable "Username" is used as the ZIP file name
- Compress ntdata.txt into a ZIP file. Variables "OP" and "UnqStr" are used as the ZIP file name

- Delete unneeded text files "user_id.txt", "ntdata.txt" and "%computername%%Username%.txt" to clean up the directory
- Call transfer.bat, which opens the FTP connection and uploads the ZIP files.

```
@echo off
call addidnt.bat
pkzip %Username%.ZIP user_id.txt
pkzip %UnqStr%.zip ntdata.txt
del user_id.txt
del ntdata.txt
del %Computername%%Username%.txt
call transfer.bat
```

B.5 wpidump98.bat

This batch file for Win98 gets text files ready for upload to the FTP site and cleans up the directory.

- Call addid98.bat, which add ID information to the original data text file
- Compress user_id.txt file into a ZIP file. Variable "Uname" is used as the ZIP file name
- Compress 98data.txt into a ZIP file. Variables "OP" and "D" are used as the ZIP file name
- Delete unneeded text files "user_id.txt", "data.txt" and "98data.txt" to clean up the directory
- Run transfer.bat, which opens the FTP site and ready to upload ZIP Files

```
@echo Off
call addid98.bat
pkzip %uname%.ZIP user_id.txt
pkzip %op%%d%.zip 98data.txt
del user_id.txt
del data.txt
del 98data.txt
call transfer.bat
```

B.6 wpiftp.bat

This batch file containing the FTP commands is used when transfer.bat opens the WPI-IP Research FTP site.

- Open FTP session

- Enters FTP site username
- Enters FTP site password
- Changes to the public FTP directory
- Set the upload mode to binary
- Upload ZIP files
- Close FTP session

```
user
ftp
mattb@wpi.edu
cd pub/mattb/incoming
bin
mput *.zip
mput *.ZIP
quit
```

B.7 transfer.bat

This batch file opens the WPI-IP-Research FTP site.

- Open the WPI-IP Research FTP site
- Call wpiftp.bat to get the commands for FTP site

```
user
ftp
mattb@wpi.edu
cd pub/mattb/incoming
bin
mput *.zip
mput *.ZIP
quit
```

References

[AW96] Martin Arlitt and Carey Williamson. Web server workload characterization: The search for invariants. In *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Philadelphia, USA, May 1996.

[BLMM94] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform resource locators (url), internet rfc 1738, December 1994.

[Bru97] Mah Bruce. An empirical model of HTTP network traffic. In *Proceedings of the IEEE INFOCOM 97*, pages 592–600, Kobe, Japan, April 1997.

[CB95] M. E. Crovella and A. Bestravos. Explaining world wide web traffic self similarity. Technical Report TR-95-015, Boston University, August 1995.

[CBC95] Carlos Cunha, Azer Bestravos, and Mark Crovella. Characteristics of www client based traces. Bu-cs-95-010, Boston University, July 1995.

[CDJM91] Ramon Caceres, Peter Danzig, Sugih Jamin, and Danny Mitzel. Characteristics of wide area tcp/ip conversations. In *ACM SIGCOMM '91*, Zurich, Switzerland, September 1991. ACM SIGCOMM '91.

[CL99] H. Choi and J. Limb. A behavioral model of web traffic. In *International Conference of Networking Protocol (ICNP '99)*, September 1999.

[Cox84] D. R. Cox. *Statistics: An Appraisal*, chapter Long Range Dependence: A Review, pages 55–74. Iowa State University Press, 1984.

[CP95] Lara Catledge and James Pitkow. Characterizing browsing strategies in the world-wide web. In *Third International World Wide Web Conference*, Darmstadt, Germany, April 1995. Third International World Wide Web Conference.

[Deg00] Loris Degioanni. *Development of an Architecture for Packet Capture and Network Traffic Analysis*. PhD thesis, Politecnico di Torino Facoltà di Ingegneria, Corso di Laurea in Ingegneria Informatica, March 2000.

[Den96] S. Deng. Empirical model of WWW arrivals at access link. In *Proceedings of ICC*, 1996.

[dP95] R. de Prony. Essai experimentale et analytique. pages 24–26, 1795. Ecole Polytechnique.

[FBC99] J. Färber, S. Bodamer, and J. Chrazinski. Statistical evaluation and modelling of internet dial-up traffic. In *SPIE's International Symposium on Voice, Video and Datacommunications*, 1999.

[FH98] James J. Filliben and Alan Heckert. *Engineering Statistics Internet Handbook*, volume 1. http://www.itl.nist.gov/div898/handbook/index.htm, 1998.

[FW97] Anja Feldmann and Wars Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance. In *INFOCOM Kobe*. IEEE INFOCOM, April 1997.

[GCMM01] M. Garetto, R. Lo Cigno, M. Meo, and M. Ajmone Marsan. A detailed and accurate closed queueing network model for many interacting TCP flows. Infocom, 2001.

[GW94] M. Garrett and W. Willinger. Analysis, modeling and generation of self similar vbr video traffic. pages 269–280. ACM SIGCOMM '94, 1994.

[HDLW95] C. Huang, M. Devetsikiotis, I. Lambadaris, and D. Wilson. Modeling and simulation of self-similar variable bit rate compressed video: A unified approach. pages 114–125. ACM SIGCOMM '95, 1995.

[HL96] D. P. Heyman and T. V. Laksman. What are the implications of long-range dependence for vbr-video traffic engineering. Number 4 in 3, pages 301–17. IEEE Trans. Networking, June 1996.

[Ins97] European Telecommunications Standards Institute. Universal mobile telecommunications system (umts), tr 101 146 v3.0.0. Technical report, European Telecommunications Standards Institute, December 1997.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[LPCE99] A. Reyes Lecuona, E. Gonzales Parda, E. Casilari, and A. Diaz Estrella. A page-oriented WWW traffic model of wireless system simulations. In *Proceedings of the 16th International Telegraffic Congress*, Edingurgh, United Kingdom, 1999.

[LS98] G. Lin and T. Suda. On the impact of long-range dependent traffic in dimensoning atm network buffer. pages 1317–1324. IEEE INFOCOM '98, 1998.

[LWTW93] Will E. Leland, Walter Willinger, Murad S. Taqqu, and Daniel V. Wilson. On the self-similar nature of ethernet traffic (extended version). pages 1–15. IEEE / ACM Transactions on Networking, September 1993.

[LXP+95] LBL, Xerox, PARC, UCB, and USC/ISI. The network simulator - ns-2. http://www.isi.edu/nsnam/ns/, 1995.

[Mog95] Jeffrey Mogul. The case for persistent-connection http. In *ACM SIGCOMM '95*, Cambridge, USA, August 1995. ACM SIGCOMM '95.

[Pax93] Vern Paxson. Empirically-derived analytic models of wide-area tcp connections, June 1993.

[PF94] V. Paxon and S. Floyd. Wide area traffic: The failure of poisson modeling. pages 257–268. ACE SIGCOMM '94, 1994.

[PK02] Kaveh Pahlavan and Prashant Krishnamurthy. *Principles of Wireless Networks - A Unified Approach*, volume 1. Prentice Hall PTR, 1 edition, 2002.

[PKC97] Kihong Park, Gitae Kim, and Mark Crovella. On the effect of traffic self-similarity on network performance. Technical report, Pride University, Department of Computer Science, 1997.

[Plu82] David C. Plummer. An ethernet address resolution protocol, internet rfc 826. November 1982.

[RE96] Bong K. Ryun and Anwar Elwaid. The importance of long-range dependence of vbr video traffic in atm traffic engineering: Myths and realities. SIGCOMM '96, 1996.

[Rob00] Thomas G. Robertazzi. *Computer Networks and Systems: Queueing Theory and Performance Evaluation*. Springer Verlag, 3 edition, 2000.

[SPE81] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION. Internet control message protocol , internet rfc 792, September 1981.

[ST99] Zafer Sahinoglu and Sirin Takinay. On multimedia networks: Self-similar traffic and network performance. Number 37, pages 48–52. IEEE Communications Magazin, January 1999.

[TTW95] M. S. Taqqu, V. Teverovsky, and W. Willinger. Estimators for long range dependence: An empirical study. In *Fractals*, number 4 in 3, pages 785–798, 1995.

[Vic97] N. Vicari. Measurement and modeling of WWW-sessions. Technical Report 184, University of Würzburg, 1997.

[WTE97] Walter Willinger, Murad S. Taqqu, and Ashok Erramli. A bibliographical guide to self-similar traffic and performance modeling for modern high speed networks, 1997.

[WTSW95] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high variability: Statistical analysis of ethernet lan traffic at the source level. ACM SIGCOMM '95, 1995.