Methodology for Parameterisation of Large Scale Network Simulations

Kai Below

2003

Methodology for Parameterisation of Large Scale Network Simulations

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

von
Kai Below
aus Hamburg

2003

1. Gutachter (referee): Prof. Dr. Ulrich Killat

2. Gutachter (referee): Prof. Dr. Volker Turau

Tag der mündlichen Prüfung (date of oral examination): 27.11.2003

Acknowledgement

I would like to thank my supervisor Prof. Dr. Ulrich Killat for giving me the chance to work in his department on the given topic and for various technical and non-technical discussions. I owe him lots of gratitude for generously supporting my research.

I would like to thank all colleagues of the department “Communication Networks” of the TUHH for the good cooperation. Special thanks to Lorenzo Battaglia, Benjamin Forgeau and Sebastian Zimmermann. I would also like to thank “my” students who gave me the chance to learn to become a supervisor.

Last, but not least, I would like to thank my family for their love and encouragement: my parents, my wife and my children.

Abstract

Network providers are confronted with the optimisation or extension of existing networks. This leads to new challenges for simulation-based network dimensioning. The first challenge is the realistic simulation of the existing network, where topology information and traffic measurements have to be considered. The second challenge is to predict how changes of the existing network will affect its performance: link capacities, queue management algorithms etc. are subject to changes to adapt the network to changing traffic requirements. The best strategy for enhancing or extending the network under consideration can be found by comparing the resulting benefits and disadvantages between the existing network and new alternatives.

The simulation of existing networks is an inverse problem: (i) the network description and some measurements are given from the network provider; (ii) the number and the behaviour of clients must be derived from the given parameters. Considering this problem was motivated by an industry project "ERNANI" funded by the "Deutsches Forschungsnetz" (DFN) and the German Telekom.

The author proposes a new methodology to solve the inverse problem. First, an algorithm for the allocation of clients is proposed. This algorithm efficiently controls the average traffic intensity. Second, a method is developed for matching higher-order moments of the simulated traffic to measurements made in existing networks. Two higher moment parameters were selected, that are of major importance in network engineering: the coefficient of variation and the Hurst parameter.

Realistic network simulations are characterised by high complexity because internal states of protocols must be stored for each connection. Therefore, memory requirements and simulation speed are major issues in such simulations. One solution for this problem is to reduce the number of clients by increasing the activity of each client in order to keep the traffic characteristics unchanged. To asses the applicability and performance of this solution, critical network parameters are estimated as a function of the number of clients. The parameters considered are: average link load, loss probability, coefficient of variation of the packet inter-arrival times, Hurst parameter and average end-to-end delay. It is shown in this work that the number of clients, as well as the required memory, could be reduced by a factor of $4-8$ without significant impact on the studied parameters. Reducing the number of clients by a factor of 8 the simulation speed increased by approximately 33 %.

This work represents a major step towards realistic modelling and simulation of existing networks. The simulation results based on the presented methodology are very promising. The successful increase of the simulation efficiency represents one step towards the realistic simulation of current and future multi-Gbit networks.

Zusammenfassung

Netzwerkbetreiber werden mit der Optimierung und Erweiterung von existierenden Netzwerken konfrontiert. Dies führt zu neuen Herausforderungen für die simulationsbasierte Netzwerkdimensionierung: Die erste Herausforderung ist die realistische Simulation des existierenden Netzes unter Berücksichtigung der Topologie und der Verkehrsmessungen. Die zweite Herausforderung ist die Vorhersage, wie sich Veränderungen im existierenden Netz auf die Leistungsfähigkeit auswirken: die Kapazitäten der Verbindungsleitungen, die Algorithmen der Warteschlangen etc. können verändert werden, um den sich verändernden Anforderungen zu genügen. Der Vergleich der resultierenden Vor- und Nachteile zwischen dem existierenden Netz und den Alternativen dient der Entscheidungsfindung für das zukünftige Netzwerk.

Die Simulation von existierenden Netzen ist ein inverses Problem: (i) die Netzwerk-Beschreibung und einige Messwerte sind bekannt; (ii) die Anzahl und das Verhalten der Benutzer muss jedoch von den bekannten Parametern abgeleitet werden. Die Behandlung dieses Problems wurde durch das Industrieprojekt “ERNANI” angeregt, gefördert durch Deutsches Forschungsnetz (DFN) und Telekom.

Der Autor stellt in dieser Arbeit eine neue Methodologie vor, um das inverse Problem zu lösen. Es wird ein Algorithmus vorgeschlagen, der die mittlere Netzlast effizient über die Anzahl und Verteilung der Nutzer kontrolliert. Im weiteren wird eine Methode vorgestellt, mit der höhere Momente der Verkehrsstatistik, hier der Variationskoeffizient und der Hurst Parameter, gemessenen Werten angepasst werden können.

Realistische Netzwerk-Simulationen zeichnen sich durch eine hohe Komplexität aus, da die Protokollzustände für jede Verbindung gespeichert werden müssen. Der Speicherbedarf und die Simulationsgeschwindigkeit überschreiten daher leicht kritische Grenzen. Als Lösung wird die Reduktion der Anzahl der Nutzer durch Steigerung ihrer Aktivität vorgeschlagen. Zur Leistungsbewertung wurden folgende kritische Netzparameter in Abhängigkeit von der Anzahl der Nutzer untersucht: mittlere Last, Verlustwahrscheinlichkeit, Variationskoeffizient der Zwischenankunftszeiten und Hurst Parameter. Die Anzahl der Nutzer, und damit der Speicherbedarf, konnte ohne signifikanten Einfluss auf die untersuchten Parameter um den Faktor $4 - 8$ reduziert werden, wobei die Simulationsgeschwindigkeit um bis zu 33 % zunahm.

Diese Arbeit repräsentiert einen wesentlichen Schritt in Richtung der realistischen Modellierung und Simulation von existierenden Netzen. Die Simulationsergebnisse basierend auf den hier vorgestellten Methoden sind vielversprechend. Die erfolgreiche Steigerung der Effizienz der Simulationen bedeutet einen Schritt in Richtung der realistischen Simulation von existierenden und zukünftigen Hochgeschwindigkeitsnetzen.

Contents

Chapter 1

Introduction

Computer networks and especially the Internet are commonly used media for e-mail, information retrieval, e-commerce, education and games. Furthermore, very significant parts of the economy rely on communication via computer networks. The growing demand for fast Internet access and new multi-media applications (e.g. video-conferencing) lead to a steadily increasing demand for larger network capacity. The demand is currently fulfilled mainly by two technologies: Wavelength Division Multiplexing (WDM), a recent optical fibre technology, provides a cost effective technology to satisfy the bandwidth demand in the core, and Asymmetric Digital Subscriber Line (ADSL) supplies the end-user with a fast Internet access.

However, it is not sufficient to provide enough capacity for the traffic: the user must be provided with a certain Quality of Service (QoS). This could be assured by simple overprovisioning methods. However, this means to provide a significantly larger capacity than required, which is expensive. Dimensioning communication networks while fulfilling the contradicting constraints of economics and QoS is a challenging task.

Analytical solutions for problems in this area would be advantageous: a lot of implementation work could be saved and the solution would not require complex simulations with several seeds for confidential results. However, analytical solutions can only be found for special cases of very simple network topologies, or under simplifying assumptions which can not be used to model reality. Therefore, computer simulations are used here for performance evaluations with realistic simulation models to predict the behaviour of the network under consideration.

Computer simulation allows realistic performance evaluation of large communication networks, as shown later in this work. The complexity of such simulations consists of a significant implementation and optimisation effort on one hand, and large memory requirements and long simulation durations on the other hand. Therefore, realistic network

simulation can only be performed with powerful simulation tools and enough processing power, e.g. with a Linux cluster. The complexity of such simulations is probably the reason why this was not addressed before: most publications utilise rather small simulation scenarios for packet level simulations, e.g. [LMS00, VKJ+98a, VKJ+98b, VFJ+99, PKC96a]. More complex network scenarios are only used in a small number of studies, e.g. [HWJL96, FGHW99], or when the simulation does not cover the packet level and therefore not all details of e.g. the TCP protocol. However, to the best knowledge of the author, there is no publication in the literature that considers the simulation of such complex networks with full protocol details as shown and performed in this study.

Nevertheless, there is a need for complex network simulations in order to understand the full dynamics of end-to-end measures and to predict the performance of e.g. a new protocol. A protocol designed for the Internet should be evaluated with competition between different connections over several hops, not only for a single hop connection. Protocol scalability issues could be identified with complex simulation scenarios, which could possibly not be realised with a simple bottleneck scenario.

The author was confronted with the optimisation of an existing network for a network provider. This task was performed in the industry project "ERNANI" funded by the the "Deutsches Forschungsnetz" (DFN) and the German Telekom. This work, several publications of the author and the final project report [BSK01] (in German) are motivated by this problem or use the knowledge and developed libraries initiated by this project [KB02, ABK02]. The qualitative and quantitative assessment of new alternatives to the current state of a network require a realistic simulation of the status quo as the first step of the performance evaluation. Moreover, measurements of the network provider need to be taken into account when setting up the simulation.

The simulation of existing networks is an inverse problem: the network description (number of nodes, connectivity, capacities, routing) and some measurements (traffic matrix, etc.) are given from the network provider. However, the number of clients and their distribution over the network as well as their behaviour are in most cases unknown. Therefore, the number and allocation of clients has to be derived from the network description and the measurements of the network provider. This inverse problem does not necessarily have a unique solution, especially since the measurements of the provider do not cover all relevant parameters. Therefore, this work provides a methodology to find one plausible solution for the inverse problem.

An early state of the proposed iterative algorithm for the allocation of clients with coarse estimations was presented in [BK02c]. This study focused on the iterations, since the initial client allocation was not accurate enough. An improved estimation of the client allocation with individual treatment of flows with different round-trip times was published in [BK02a] and [BK02b]. An improvement of the throughput estimation for TCP's congestion-avoidance phase and a methodology for adjusting the Hurst parameter of the traffic was presented in [BK03a]. The protocol overhead and the traffic from client to

server was additionally considered in [BK03b], leading to further improvements in the estimation accuracy. Furthermore, a methodology to reduce the simulation complexity by reducing the number of clients was also presented in [BK03b]. An extensive article summarising major findings of this work will be published in [BK04].

This work is structured as follows: a detailed description of the problem considered is presented in Chapter 2. The source model used to generate the traffic is characterised in detail in Chapter 3. The solution of the inverse problem of adapting the simulation to measurements is detailed in Chapter 4. Configuration and implementation aspects of the simulator and the network simulation models are described in Chapter 5. The simulation results are shown and discussed in Chapter 6. The whole work is summarised and the conclusions are presented in Chapter 7.

Chapter 2

Description of the Problem

The motivation and a detailed description of the problem considered and solved in this work are presented in this chapter. The intrinsic features of the WWW traffic are the user behaviour and the HTTP/TCP protocols. Therefore, it is important to use source models that reflect this behaviour [FP01]. A simple example in Sec. 2.1 shows that an aggregated traffic model based on the User Datagram Protocol (UDP) can not be used to model WWW traffic: the reactivity of TCP is not captured by the aggregated model, leading to completely wrong performance estimates.

The lifelike simulation of the current state of a network is treated in Section 2.2. Moreover, the problems associated with finding the appropriate parameters for the setup of the simulation are discussed here. The parameters influencing the TCP throughput are presented in Section 2.2. The estimation of the TCP throughput is required to adjust the average load in the network. An approach for matching the higher order moments of the traffic statistics is presented in Section 2.3. Convergence issues with workloads following a heavy-tail distribution are discussed in Section 2.4.

2.1 Granularity of Modelling

Realistic simulations of large communication networks require a large amount of computer resources. The simulation speed and the required memory for the simulation are the key requirements that can render a realistic simulation impossible. The required memory is proportional to the number of clients in the implementation used in this work. The simulation speed is mainly determined by the number of packets (or events) that are sent across the network.

Being confronted with design, implementation and conduction of very complex network simulations, the question arises whether or not to use aggregated source models, where

one aggregated source model replaces several hundreds or thousands of HTTP/TCP clients. The aggregated approach has several advantages:

- The memory consumption of the simulation is reduced drastically since the state information of the single connection is not taken into account.
- The configuration of a few aggregated source models is easier to perform for the user than for thousands of individual sources.
- The run-time efficiency is higher for aggregated source models which manifests itself in a higher simulation speed.

The advantages seem to be very convincing on one hand. On the other hand, the simplicity of aggregated source models goes hand in hand with major disadvantages: no feedback information about the congestion state of the network is taken into account and the state information is not saved for each connection. The reactivity of the TCP protocol is a key feature of the applications and protocols used in the Internet. The three measurements in Tab. 2.1 underline that TCP is the most important protocol used in the internet: more than 85 % of the packets and more than 93 % of the Bytes are created by applications using the TCP protocol.

Table 2.1: Protocol usage of TCP and UDP, measurement April, 7th 2003, `http://ipmon.sprint.com/packstat/packetoverview.php`.

Location	Protocol	Packets %	Bytes %
nyc-21.0-030407	TCP	90.51	96.22
	UDP	7.31	2.71
rly-23.0-030407	TCP	85.60	93.32
	UDP	11.65	4.56
sj-28.0-030407	TCP	87.10	96.02
	UDP	11.11	3.23

The application usage of the measurements in 2000 is summarised in Tab. 2.2: the majority of the packets and the volume are produced by HTTP applications. The protocol usage did not change significantly in the last three years, but the application usage changed substantially: packet and byte volume of Peer To Peer (P2P) traffic has increased significantly. However, this study started in October 1998, when the vast majority of the traffic was web traffic. Therefore, only web traffic was considered in this study.

The *degree of overload of TCP traffic,* which is required in the following, is defined as follows: the degree of overload is the throughput that could possibly be achieved with a

Table 2.2: Usage of selected applications, August 9th, 2000, `http://ipmon.sprint.com/packstat/packetoverview.php`.

Location	Application	Packets %	Bytes %
sj-09.0-000809	HTTP	70.19	80.78
	P2P	3.36	3.69
	FTP	1.37	1.23
	Email	3.13	2.68
	Other TCP	14.44	8.36
sj-00.0-000809	Web	71.12	83.49
	P2P	3.34	2.44
	FTP	0.75	1.01
	Email	2.44	1.58
	Other TCP	7.46	4.63

sufficient link capacity divided by the throughput achieved with the current link capacity. Example: the current throughput is 49 Mbit/s (for a capacity of 50 Mbit/s) and the throughput increases to 78 Mbit/s (for a capacity of 100 Mbit/s) resulting in a degree of overload of 159 %.

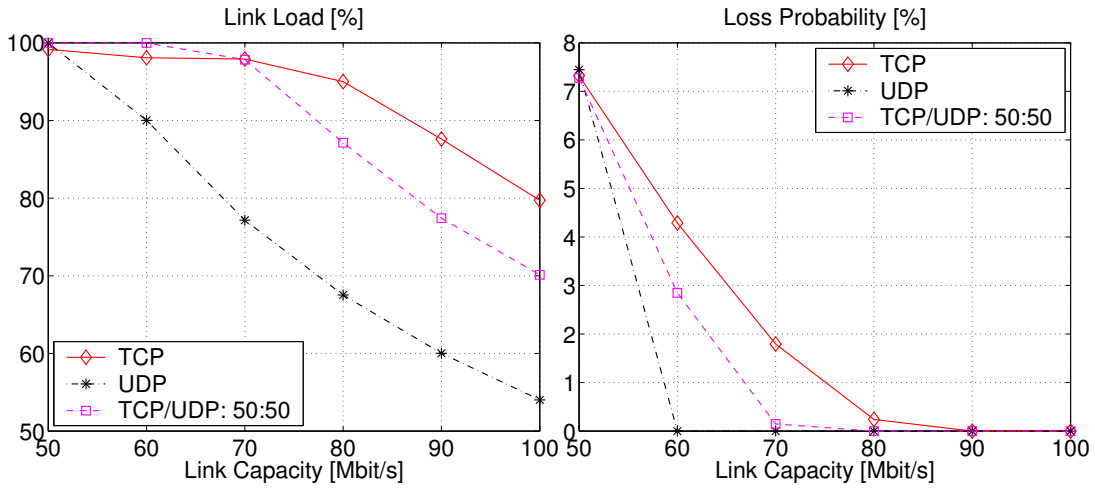
A very simple experiment, conducted with the so called bottleneck scenario, is described in the following (see Sec. 5.6.1 for a full description of the bottleneck scenario). The example supports the argument that the reactivity is a very important feature that needs to be modeled for realistic simulations. The bottleneck model was configured with a link capacity of 50 Mbit/s and 159 % overload: the utilisation was $\rho \approx 98$ % and the loss probability was approximately 7.3 %. The target link capacity is to be found such that the link load conforms to $\rho \leq 80$ %.

The traffic is modeled in three ways: with pure HTTP/TCP connections, with an aggregated UDP traffic model [RL98, RL96, RN96] and with a mixture of both traffic variants. The HTTP/TCP scenario is equipped with a typical web user model (average download volume 60 KB and average off-time 40 s) and 12196 users. Both scenarios use power-tail distributions to induce self-similarity and long-range dependence into the traffic. The UDP source model was tuned to reach approximately the same loss probability in order to establish a comparable situation for HTTP/TCP and UDP traffic at the starting point (capacity 50 Mbit/s). The third variant was set up with a traffic mix of 50 % HTTP/TCP and 50 % UDP traffic. The link capacity was increased by increments of 10 Mbit/s from 50 Mbit/s to 100 Mbit/s.

The simulation with UDP traffic is more than 6 times faster than the simulation of TCP traffic and requires only 8.5 MB RAM while HTTP/TCP clients allocate 240 MB RAM.

The third case with the traffic mixture is still 230 % faster than pure HTTP/TCP traffic and requires 112 MB RAM.

The resulting link load and loss probability measurements for the three cases are visualised in Fig. 2.1. It is obvious that the link load and loss probability match quite well for 50 Mbit/s. However, the values are diverging for larger link capacities: the HTTP/TCP traffic reaches 80 % link utilisation at 100 Mbit/s link capacity whereas the UDP traffic requires a link capacity of less than 70 Mbit/s to bring the utilisation down to 80 %. The loss probability is also descending much slower with the link capacity for TCP traffic as compared to with UDP traffic. The traffic load and loss probability in the case with the traffic mix is closer to pure TCP traffic. However, it is still far away from reproducing the behaviour of pure TCP traffic.



(a) Link Utilisation (b) Loss Probability

Figure 2.1: Overloaded simple bottleneck with TCP and UDP traffic for increasing link capacity.

The reason for the different characteristics of the three experiments is the reactivity of TCP, that is lacking in UDP: the congestion avoidance mechanisms in TCP lead to an adaption to the network state, preventing that the loss probability exceeds reasonable limits over a broad range of offered traffic. Therefore, the link utilisation stays approximately constant when the capacity increases until the offered load does not exceed the link capacity anymore. This threshold is reached for approximately 70 Mbit/s in Fig. 2.1 (a).

Obviously, the UDP traffic can not be used as a complete replacement of HTTP/TCP traffic in order to speed-up the simulation and to reduce the complexity of the simulation. The reactivity is one very important property of the major part of the Internet traffic.

Nevertheless, it might be an option to replace a certain fraction of the TCP traffic by an aggregate UDP traffic. The simulation results show, that the fraction must be significantly smaller than 50 %, if the results should match the TCP traffic closely. However, the gain of using aggregated UDP traffic is directly related to the fraction of UDP traffic. Therefore, also the gain of using aggregated UDP traffic is small if the fraction of UDP traffic is small. Additionally, the validation and parameterisation of other stochastic traffic properties in the case of the traffic mixture is a task, which has to be accounted, when the gain is judged.

More cases can be found where modelling of the reactivity of TCP is also important:

- performance evaluation of different active queue management strategies
- applications and higher layer protocols utilising TCP, where the throughput depends on loss probability and end-to-end delay
- competition between different connections, fairness issues.

Since UDP traffic obviously can not be used for modelling the reactive traffic in the Internet, the focus of this work is the simulation of pure HTTP/TCP traffic.

2.2 Generating Prescribed Traffic Intensities

The classical network simulation experiments build on the following setting: models of network nodes are interconnected to form a certain network model. The nodes are equipped with a routing table, a buffer with a certain capacity and management strategy. A set of connections with a certain behaviour are connected to the network nodes. The input variables of the simulation are the source model parameters and the network model parameters as detailed in Fig. 2.2. The output parameters are (among others) the following performance measurements: the traffic matrix (average throughput for each flow from node i to node j, e.g. Mbit/s), Hurst parameter, coefficient of variation of packet inter-arrival times (IATs) and loss probability, as shown in Figure 2.2.

The optimisation of an existing network, e.g. by the network provider, poses a different type of problem: the optimisation requires a realistic simulation of the current state of the network as reference in the first step. The design alternatives to be considered are then compared to the reference in the second step of the performance evaluation.

The simulation of existing networks is an inverse problem: the network description (number of nodes, connectivity, capacities, routing) and some measurements (traffic matrix, etc.) are given from the network provider. However, the number of clients and their distribution over the network as well as their behaviour are in most cases unknown.

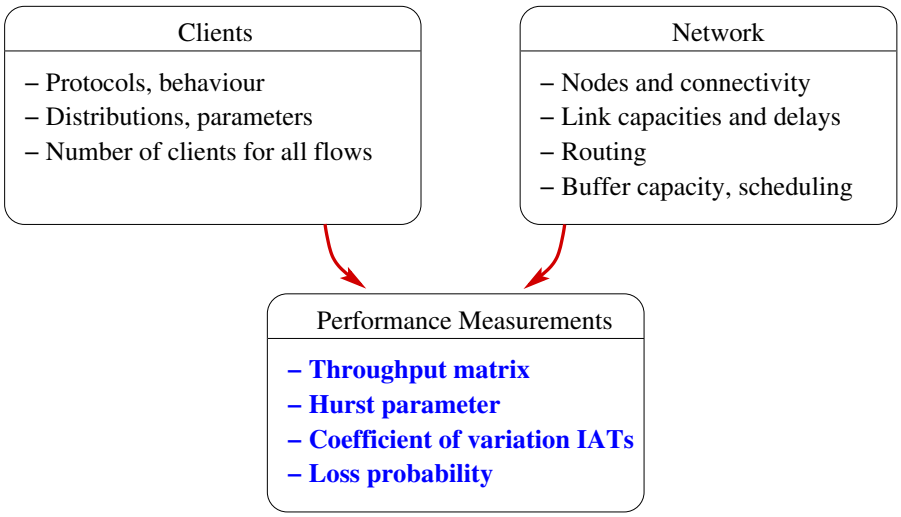


Figure 2.2: Classical simulation paradigm: source and network models are given, the simulation is used for measuring the performance.

Therefore, the number and allocation of clients has to be derived from the network description and the measurements of the network provider.

This problem can be solved by assuming that a selected user model with the corresponding parameters for the HTTP/TCP clients is valid for the network under focus. A user model very similar to the one described in [CL99] is used here (see Sec. 3.1 for a more detailed description). The solution of the inverse problem is then to find the appropriate number of HTTP/TCP clients for all flows to establish the measured throughput matrix (see Fig. 2.3).

A prerequisite for determining the number of clients for all flows is the estimation of the average throughput of a single TCP connection. User models for generating WWW traffic are characterised by an on/off behaviour: the client opens a connection and sends a HTTP request to the server, the server responds with the requested data. The client enters and idle period (reading time) after having received the data. Furthermore, the average download volume is very small (around 60 KB [CL99]) leading to many short-lived TCP connections.

The average throughput of a single connection can not be calculated with the simple steady state equations for the TCP throughput [Mor00] since TCP does not reach the steady state for such short-lived connections that are characteristic for WWW traffic. Models for the TCP throughput (or for download times) for finite download volumes are also existing [CSA98] but they are rather complex and require the knowledge of the packet loss probability, which is not known and hard to estimate a priory. A simple model for the TCP throughput of short-lived connections is developed in this work and

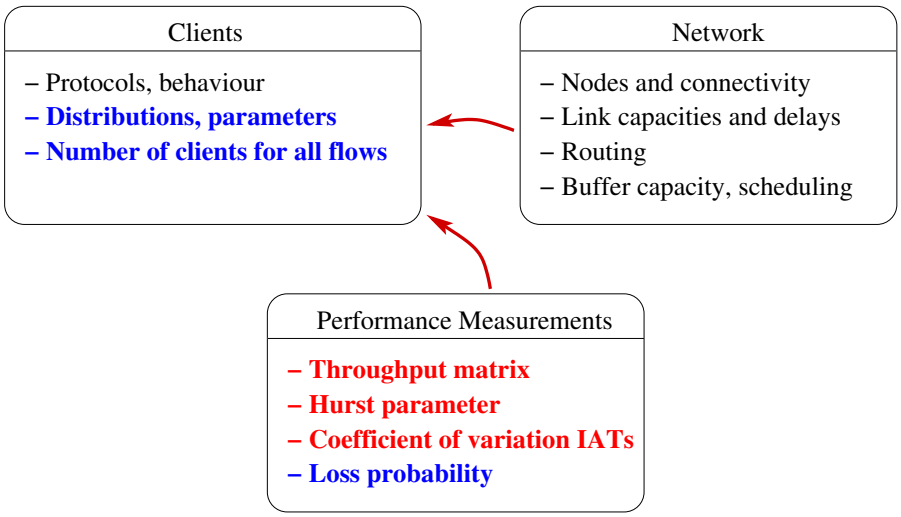


Figure 2.3: Inverse problem: network model and some performance measurements are given, source model parameters and number of clients for all flows are unknown parameters.

presented in Section 4.1. It is shown in Section 6.2 that the model is sufficiently accurate for the purpose of finding the appropriate number of clients for all flows.

A unique solution is not necessarily existing for such an inverse problem. Especially links with a utilisation close to 100 % pose a problem: it could be that the offered load is also about 100 %; but an offered load of 150 % on the other hand would show very similar utilisation values. The two cases can only be distinguished by inspection of further measurements, like queueing delay or loss probability (see Sec. 2.1, p. 6 for the definition of overload with TCP traffic).

Establishing an average traffic load in the simulation that is close to measured values is, however, only the first step for a realistic simulation: the higher order moments of the traffic statistics need to be matched as well to measurements from the real network. This is essential since the performance of many QoS measures depend significantly on higher order statistics, e.g. packet delay variations and packet loss probability. The matching of the higher order statistical parameters Hurst parameter and coefficient of variation are addressed in a Section 2.3.

2.2.1 Parameters Influencing TCP Throughput

A solution for the above mentioned inverse problem should take into account all parameters that have a non-negligible influence on the average throughput of a TCP connection:

- link utilisation (depends on given throughput matrix, routing and link capacities)
- round-trip time (depends on the number of hops, waiting times in the buffer, link capacities and utilisation)
- number of competing TCP connections
- burstiness of traffic, coefficient of variation, Hurst parameter (depends on user model and number of TCP connections)
- packet loss probability (depends on buffer capacity and scheduling, number of connections, utilisation, coefficient of variation and Hurst parameter)

Network congestion is detected by TCP through the feedback via the acknowledgements of the receiver; the sending rate is adapted accordingly. Packet losses are detected by three duplicate acknowledgements of the receiver or by time-outs (cf. Sec. 3.2.4). The currently mostly used TCP variant is called “NewReno”, the congestion control algorithms (fast-retransmission and fast-recovery) are able to maintain an approximately stable throughput up to some percent of packet loss rate. There is one exception to this behaviour: fast-retransmission can only be triggered when the congestion window size is larger than three packets. Otherwise, if the congestion window is smaller or equal to three packets, there are not enough duplicate acknowledgements that could trigger a fast-retransmission and therefore TCP has to wait for a time-out. The duration of TCP time-outs range from some hundred milliseconds up to 64 seconds. This can lead to a crucial reduction of the average throughput, especially if the packet loss occurs in the first and second round-trip, when there are not enough samples to perform a good estimate on the round-trip time. Default time-out values of 6 and 12 s are used in the first and second round-trip, respectively.

An example with two approaches for estimating the required number of connections s_{ij} for the flow from node i to node j illustrates the problem considered: the first, naive approach is to assume that the number of connections is proportional to the prescribed throughput tp_{ij}: $s_{ij} = const \cdot tp_{ij}$. This approach is very simple, but nevertheless, the estimation of the multiplicative constant is still an open issue. The reference is the solution developed in this work (see Sec. 4.1), called “advanced approach” in the following.

The simulation results from the two approaches are visualised in Fig. 2.4. A near optimal solution for the multiplicative constant was derived with the advanced approach; the same number of clients is used in both approaches, only the distribution over the flows is different. The relative difference to the prescribed throughput values is depicted on the ordinate and the flow identification on the abscissa. It is obvious that the difference to the prescribed values is not acceptable with the naive approach in Fig. 2.4 (a). The results achieved with the advanced approach match the target quite well: the maximum deviation is approximately 8 %.

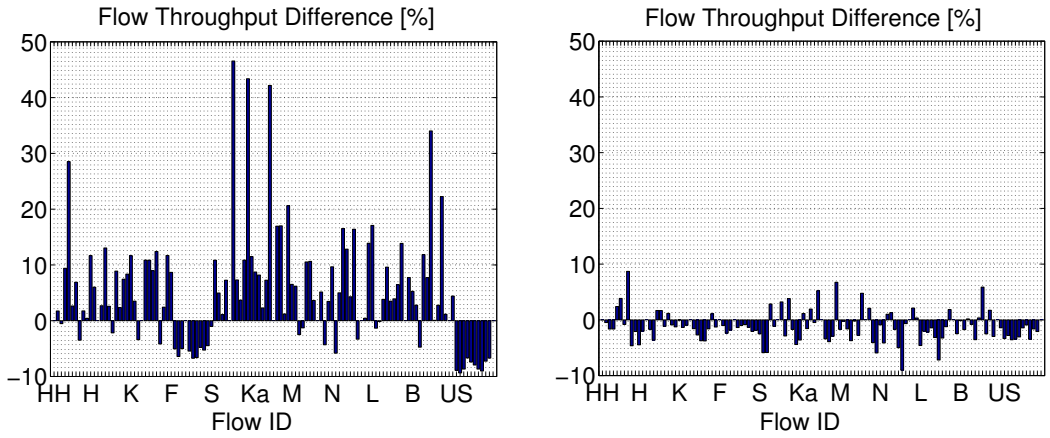


Figure 2.4: Comparison of (a): naive approach $s_{ij} = const \cdot tp_{ij}$ versus (b): advanced approach.

The reason why the naive approach fails, even with the same number of clients, is that this approach does not consider the strong sensitivity of TCP to different round-trip times. The next step would be to incorporate the round-trip time into the calculation. However, another simple approach integrating the round-trip time rtt_{ij} as multiplicative parameter $s_{ij} = const \cdot rtt_{ij} \cdot tp_{ij}$ leads completely into the wrong direction as shown in Fig. 2.5: all flows with a relatively large round-trip time experience a larger throughput (positive value) and all flows with a small round-trip time a smaller throughput than the prescribed value. The reason for this behaviour is that web transfers are typically on-off processes: the client is active sending requests and receiving the web page and is idle when the user reads the content. The transmission time depends on the round-trip time but the passive time does not depend on the round-trip time. The derivation of model of the throughput of a HTTP/TCP client and the required number of clients for all flows satisfying the given traffic matrix (called “advanced approach” here) is presented in Section 4.1.1.

2.3 Matching Traffic Statistics

In addition to adjusting the average throughput as discussed in the previous section, also the higher order statistics in the simulation need to be matched to the measurements from the network existing in the real world. Two very important higher order parameters are discussed in this section: the coefficient of variation of the packet inter-arrival times entering a queueing system (Sec. 2.3.1) and the degree of self-similarity, the Hurst parameter H (Sec. 2.3.2). The coefficient of variation describes the variability of the traffic

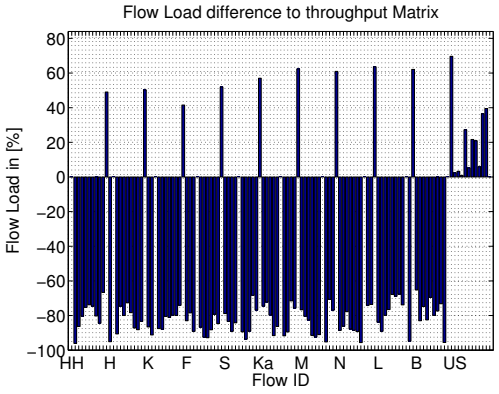


Figure 2.5: Approach $s_{ij} = const \cdot rtt_{ij} \cdot tp_{ij}$.

and the Hurst parameter describes the self-similarity and long-range dependence of the traffic. These two parameters represent very important higher order traffic characteristics.

2.3.1 Coefficient of Variation of Packet Inter-Arrival Times

The coefficient of variation of the inter-arrival time of packets entering the queueing systems is one measure of the burstiness of the traffic. It is defined as the standard-deviation divided by the mean value

$$cv = \frac{\sigma}{\mu}. \tag{2.1}$$

The coefficient of variation depends on the link utilisation, the number of TCP connections, the packet size distribution and the average download volume. A high link utilisation leaves scarce room for traffic variability. Therefore, the coefficient variation decreases when the traffic load increases. The multiplexing gain increases with the number of connections, resulting in a smoother traffic. Hence, the value of *cv* decreases when the number of connections is increasing. Variable packet sizes introduce variability because the transmission time of large and small packets differ. This leads to a higher variance of the inter-arrival times and therefore to a higher value of the coefficient of variation. The average download volume also affects the coefficient of variation: a very small download volume results in very short connection durations and a traffic that tends to Poisson behaviour. Large download volumes on the other hand lead to a large competition among the TCP connections. A strong competition leads to packet losses and time-outs which increase the variability of the traffic.

2.3.2 Self-Similarity

Network traffic is known to be self-similar since the famous Bellcore study [LTWW93]. The comparison of the counting process of Poisson traffic and self-similar traffic for different counting windows in Fig. 2.6 reveals some properties of the two traffic types: the packet count for a time unit of 0.01 s (bottom) is similar for both types – the variability is very high. But also a difference is visible: the Poisson traffic has no correlation whereas the “hills” and “valleys” have larger width for the self-similar traffic, indicating a correlation structure.

The variability of the Poisson traffic is reduced with each step of increasing the counting window from 0.01 s to 100 s. The graph tends to a straight line with constant average value and very small variance for a counting window of 100 s. This is not the case for the self-similar traffic: spikes are visible on all scales and the variance is only reduced marginally for large values of the counting window. Since the burstiness of the traffic is very similar on all time scales (or aggregation levels) this traffic is called self-similar traffic.

The definition of second order self-similarity and the associated properties based on [LTWW93] is given in the following. The stochastic process X is defined as sequence of numbers (e.g. inter-arrival times, counting process):

$$X(i),\ i = 0,1,2,3,\ldots \tag{2.2}$$

The aggregation of the stochastic process X is denoted by $X^{(m)}$

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{k \cdot m} X(i),\ k = 1,2,3,\ldots, \tag{2.3}$$

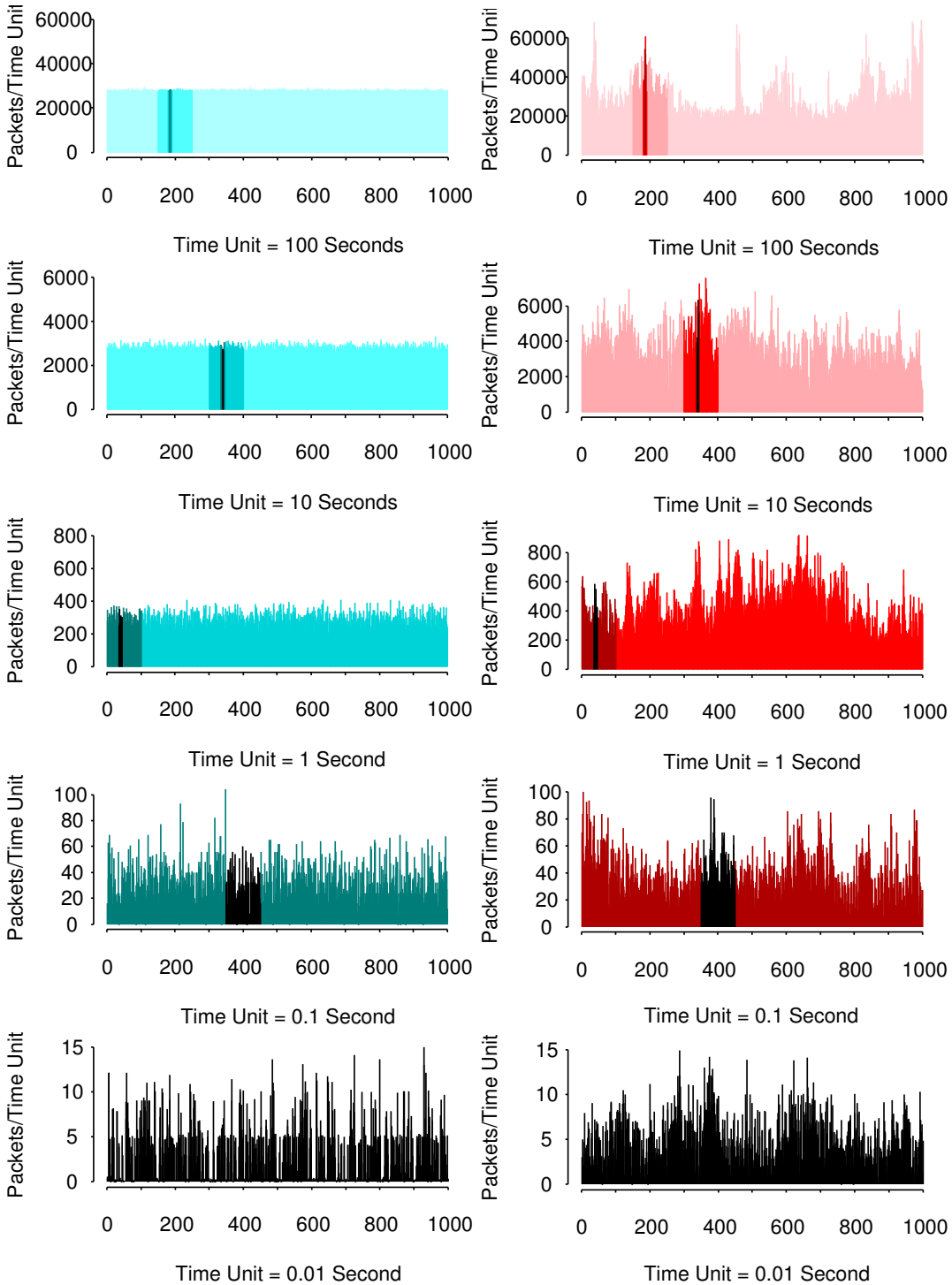
that is, the aggregated process consists of sums non-overlapping intervals of size m.

The Cumulative Distribution Function (CDF) $F(x)$ describes the probability that the value of the stochastic process is smaller or equal to the threshold x:

$$F(x) = P\{X \leq x\}. \tag{2.4}$$

The Complementary Cumulative Distribution Function (CCDF) $R(x)$, also known as reliability function, describes the probability that the value of the process is larger than the threshold x:

$$R(x) = 1 - F(x) = P\{X > x\}. \tag{2.5}$$



(a) Poisson Traffic.

(b) Self-Similar LAN Traffic.

Figure 2.6: Counting process (packets per time unit) of (a): Poisson traffic versus (b): self-similar traffic [WTRW97].

A class of *power-tail* distributed random variables with the property Eq. (2.6), also called *heavy-tail* and *long-tail* distributed random variables, can be used to produce self-similar traffic [CB97, PKC96b].

$$R(x) \sim cx^{-\alpha} \quad \text{as} \, x \to \infty, \, 0 < \alpha < 2 \tag{2.6}$$

Three cases for the shape parameter α can be differentiated:

- $\alpha \geq 2$: the process has finite mean and variance (no heavy tail)
- $1 < \alpha < 2$: process has infinite variance and finite mean
- $0 < \alpha < 1$: process has infinite variance and infinite mean.

A stochastic process is called *exactly second order self-similar,* if the Auto-Correlation Function (ACF) of the m-aggregated process is the same as for the original process:

$$r^{(m)}(i) = r(i). \tag{2.7}$$

The process is called *asymptotically second order self-similar* if Eq. (2.7) holds only for very large values of m and i:

$$r^{(m)}(i) \approx r(i) \quad \text{for} \quad m, i \to \infty. \tag{2.8}$$

Self-similar processes have the following three properties:

- Slowly decaying autocorrelation:

$$r(i) \sim i^{-\beta} \quad \text{with} \quad 0 < \beta < 1. \tag{2.9}$$

- Long Range Dependence (LRD): the sum of the autocorrelation is infinite:

$$\sum_{i=-\infty}^{+\infty} r(i) = \infty. \tag{2.10}$$

- Slowly decaying variance (with increasing aggregation level m):

$$Var\{X_k^{(m)}\} \sim m^{-\beta} \quad \text{with} \quad 0 < \beta < 1. \tag{2.11}$$

The last property, the slowly decaying variance, is the most salient feature of self-similar processes: self-similar processes have a very large variability that manifests itself in a slowly decaying variance. The variance of so-called "well-behaved distributions" (no heavy tail, e.g. negative exponential distribution) follow m^{-1}, which decays considerably faster than $m^{-\beta}$, with realistic values of $\beta \approx 0.5$ (cf. Sec. 2.4). The problem for communication networks stems from the fact that this large variability reduces the multiplexing gain even for a large number of connections since the variations are not smoothing out with the same speed as could be expected from e.g. Poisson traffic as visible in Fig. 2.6.

The degree of self-similarity of network traffic is described by the Hurst parameter H. The parameter β from Eq. (2.11), the shape parameter α from Eq. (2.6) and the Hurst parameter H have the following relationship, if the file size distribution is the only source of self-similarity:

$$H = (3-\alpha)/2 = 1-\beta/2 \tag{2.12}$$

$$\alpha = 3-2H = 1+\beta \tag{2.13}$$

$$\beta = 2(1-H) = \alpha-1. \tag{2.14}$$

Various Hurst parameter estimators are available in the scientific literature. The Variance Time plot (VT-plot) directly uses Eq. (2.11): the variance of the aggregated processes is plotted against the aggregation level on a double logarithmic plot. The slope of a straight line fitted into the curve determines the Hurst parameter. However, there are more reliable methods for estimating the Hurst parameter: it has been shown that the wavelet-based so-called Abry-Veitch estimator outperforms all other Hurst parameter estimators [AV98]. Therefore, this estimator is used in this work, see Appendix A.1 for a detailed description.

The Hurst parameter can be adjusted with the shape parameter α of the heavy-tail distributed file sizes [PKC96a, CB97] and follows approximately Eq. (2.12). To be more precise: the superposition of an infinite number of on-off sources with heavy-tail distributed on- or off-times and heavy-tail shape parameter α fulfil Eq. (2.12) [TWS97].

The Hurst parameter is also not independent of the target load on a specific link in a network. There are mainly two effects that can be observed:

1. The Hurst parameter descends with increasing load, the traffic tends to Poisson [CCLS01, PF95].
2. The Hurst parameter increases with increasing load due to the increased long range dependence introduced by a larger probability of TCP time-outs (see Section 6.4.3).

The significance and dominance of these effects is discussed together with the simulation results in Section 6.4.2.

2.4 Convergence Issues with Heavy-Tail Workloads

The *Generalised Central Limit Theorem* (GCLT) [Fel71, CL97], which includes heavy-tail distributed random variables with infinite variance, describes the effect of addition of random variables on the resulting distribution. The sum A_n of n independent and identically distributed (iid) random variables X_i with mean μ, variance σ^2 and shape parameter $1 < \alpha < 2$ is

$$A_n = \frac{1}{n} \sum_{i=1}^{n} X_i. \tag{2.15}$$

A distribution Z_n with mean zero can be defined as

$$Z_n = n^{1-1/\alpha} (A_n - \mu) \tag{2.16}$$

with

$$Z_n \xrightarrow{d} S_\alpha, \tag{2.17}$$

where S_α is a so-called α-Stable distribution. The notation $Z_n \xrightarrow{d} S_\alpha$ means that the random variable Z_n converges in distribution to S_α (has roughly the same distribution for large n). The α-Stable distribution follows also a heavy tailed distribution with the same shape parameter α but with a smaller standard deviation, see factor $n^{1-1/\alpha}$ in Eq. (2.16).

For the case of finite variance $\alpha = 2$ the GCLT reduces to the well-known *Central Limit Theorem* (CLT):

$$Z_n = n^{1/2} (A_n - \mu) \tag{2.18}$$

with

$$Z_n \xrightarrow{d} N(0, \sigma^2), \tag{2.19}$$

where $N(0, \sigma^2)$ represents the normal distribution with zero mean and variance σ^2.

Comparing the GCLT with the CLT it becomes obvious that the implication with heavy-tailed random variables is, that the standard deviation of the process decreases much slower for the GCLT according to $1/n^{1-1/\alpha}$ as compared to $1/n^{1/2}$ for the CLT (for realistic values of $\alpha \approx 1.5$). Moreover, the distribution S_α is again a heavy-tailed distribution and not the normal distribution, as in the case of the CLT.

The behaviour of the sum of random variables is visualised for different values of n in Fig. 2.7. The mean value is set to one for both cases. It is obvious that the width of the exponential distribution at the left hand side decreases very fast with increasing n and that the mean value is already consistent for small values of n. In contrast, the width of the heavy-tailed distribution at the right hand side is still large for large values of n and the mean value did not yet converge to the mean value of $\mu = 1$, even for $n = 10,000$.

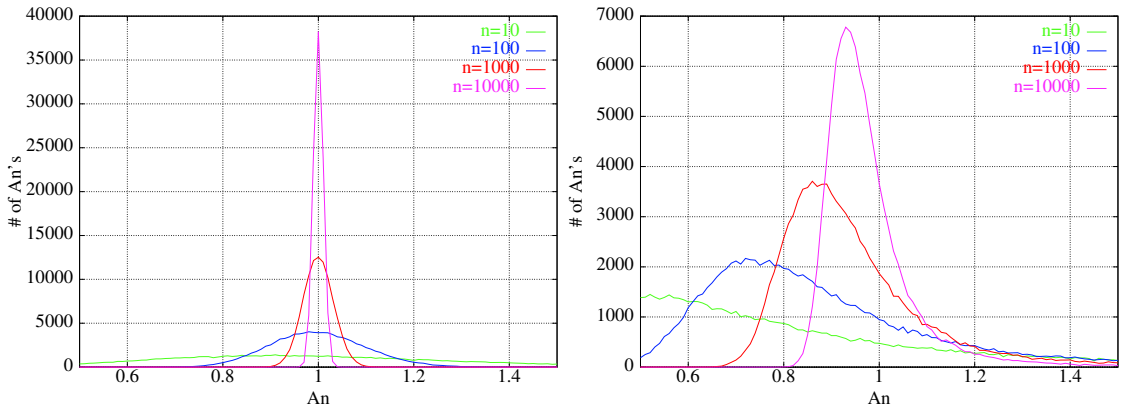


Figure 2.7: Histogram of A_n for different n for exponential (left) and heavy-tailed (right) random variables, both with mean $\mu = 1$ [CL97].

Consequently, simulations with heavy-tail distributed random variables experience a stability issue: the convergence to the mean value is very slow due to the fact that very large values have a non-negligible probability of occurrence [CL97]. Heavy-tail distributions are used here for the file size distribution (HTTP object size distribution).

Crovella and Lipsky have presented an estimation on how many samples n are required in order to reach an accuracy of k digits for a variable value of α, see (2.20) [CL97]. The required number of samples for a two digit accuracy ($k = 2$) is shown in Table 2.3 for selected values of α.

$$n \geq c \cdot 10^{k/(1-1/\alpha)}. \tag{2.20}$$

The required number of samples increases non-linear and very fast for a stable mean value, especially when $\alpha \leq 1.3$. Since actual network traffic has a degree of self-similarity in the range $[0.6, 0.85]$ it is necessary to use values of $\alpha \geq 1.3$ according to Eq. (2.13), for which $4.6 \cdot 10^8$ samples are required to reach an accuracy of two digits on the estimation of the mean value. Approximately $12 \cdot 10^6$ file sizes are drawn in a simulation of 600 s simulated time with the so-called B-WiN scenario (see Sec. 5.6.3 for a detailed description of the scenario). Therefore the sample size is large enough to achieve a two digit accuracy for $\alpha \geq 1.4$ for the mean value of the heavy-tail distributed file size. However, some simulations are nevertheless performed with smaller α-values in order to see the tendency for small shape parameter values. It is important to keep in mind that the accuracy of the estimated mean is less than two digits in this case. The two digit accuracy is not valid for the random numbers of each flow: the $12 \cdot 10^6$ file sizes are the of sum all drawn files in the whole B-WiN consisting of 110 flows. Therefore,

Table 2.3: Number of samples necessary to achieve two digit accuracy for the mean value as a function of α (Eq. (2.20) with $c = 1$).

α	n
2.0	$1.0 \cdot 10^4$
1.75	$4.6 \cdot 10^4$
1.7	$7.2 \cdot 10^4$
1.6	$2.2 \cdot 10^5$
1.5	$1.0 \cdot 10^6$
1.4	$10 \cdot 10^6$
1.3	$4.6 \cdot 10^8$
1.2	$1.0 \cdot 10^{12}$
1.1	$1.0 \cdot 10^{22}$

it could be expected that not all flows reach a two digit accuracy for the mean value, especially for small values of α.

A further increase in convergence speed was gained with two different strategies:

- using Truncated Power-Tail (TPT) distributions (see Fig. 2.8), where the tail of the distribution fades out exponentially after a defined threshold T_1, or
- cutting the tail sharply after the defined threshold.

A combination of both methods is used in this work: the TPT distribution fades out exponentially after the first threshold T_1 and is cut sharply after a second threshold T_2.

A measurement of the tail of the CCDF of the TPT distribution implemented in the simulation environment Ptolemy with $n = 10^{10}$ samples is shown for $\alpha = 1.3$ and $\alpha = 1.5$ in Fig. 2.9 in a double logarithmic plot. An average file size of 10 KB was used for the simulations and the threshold was set to $T_1 = 20$. The exponential fading starts after approximately 60 MB for $\alpha = 1.3$ and after 20 MB for $\alpha = 1.5$.

The sharp cut is not performed in the measurements shown in Fig. 2.9. However, the sharp cut is performed in the simulations at $10^8 = 100$ MB, where the curve reaches a probability of approximately $9 \cdot 10^{-7}$ for $\alpha = 1.3$ and $9 \cdot 10^{-8}$ for $\alpha = 1.5$. These settings provide a file size distribution with a power-tail over more than three orders of magnitude above the mean, a small phase of exponential fade out (depending on α) and a sharp cut at four orders of magnitude above the mean.

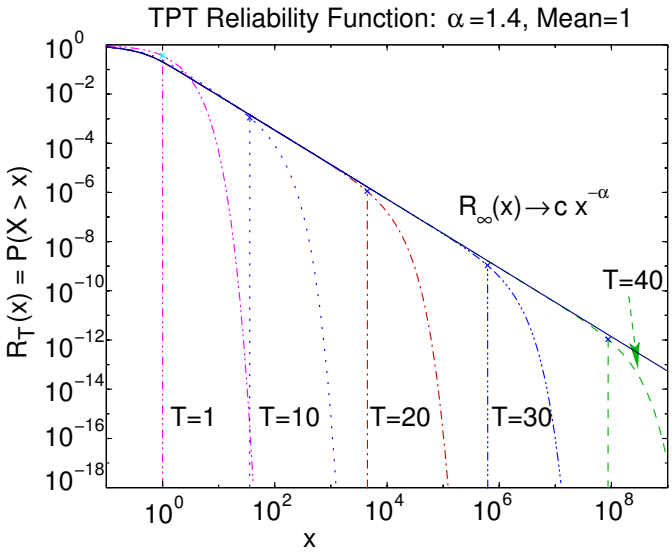


Figure 2.8: CCDF of the Truncated Power-Tail (TPT) distribution [GJL95, GGS97].

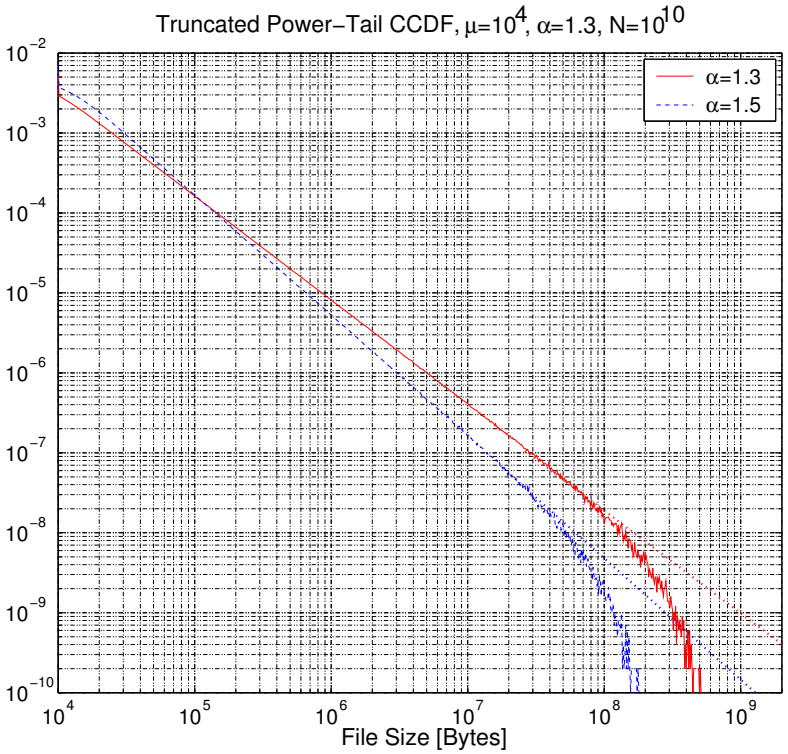


Figure 2.9: Measured tail of CCDF of the Truncated Power-Tail (TPT) distribution, parameters: $\mu = 10^4$, $T_1 = 20$, $T_2 = \infty$, $n = 10^{10}$ samples.

It can be expected that the cutting strategy improves convergence for small values of α: the rare occurrence of very large values determines the convergence behaviour. Limiting the value range improves the convergence behaviour. However, the heavy-tail is maintained for more than three orders of magnitude so that self-similarity can be observed in the measurements. The power-tail is also limited in reality to some orders of magnitude above the mean due to physically limited systems, e.g. limited buffer sizes, limited file sizes on webservers, limited network capacity etc. Hence, this limitation increases the convergence of the simulation and mimics the existing limits of real existing systems.

Chapter 3

Source Model

A detailed description of the source model including the on-off user behaviour with the associated distributions (cf. Sec. 3.1) and the utilised protocols HTTP, TCP and IP are presented in this chapter. An overview over the protocols HTTP, TCP and IP is presented in Section 3.2. The HTTP protocol and the features modelled in this work are presented in Section 3.2.1. The header formats of TCP and IP are sketched and the dynamics of the TCP protocol are discussed. The bandwith limitations at different locations in the network and the corresponding implications are discussed in Section 3.3.

An analytic model for WWW traffic, the so called TCP-modified Engset model [HLN97], is described in Sec. 3.4. This model is based on a birth-death process that models the connection arrival and departure process. It provides solutions for e.g. the probability that j users are active at the same time, the average download time and the average link load. This model is an analytic solution that claims to model reality. It is presented here in order to be able to compare the simulation results with the results from this analytic model.

3.1 On-Off Source Behaviour

The user model used in this work follows the studies in [CL99, PKC96a, CB97, PKC97]. The currently widely used HTTP 1.1 is implemented including pipelining. That is, the full HTTP transfer volume is transmitted in one TCP connection, which is being closed at the end of the transfer, see Fig. 3.1. The usage of parallel TCP connections for one HTTP download is not modelled and implemented here. The additional effort for modelling and implementation of parallel TCP connections is significant. Therefore the author decided to develop a performant solution for HTTP with a single TCP connection, which can be used to develop a solution for parallel connections in future studies.

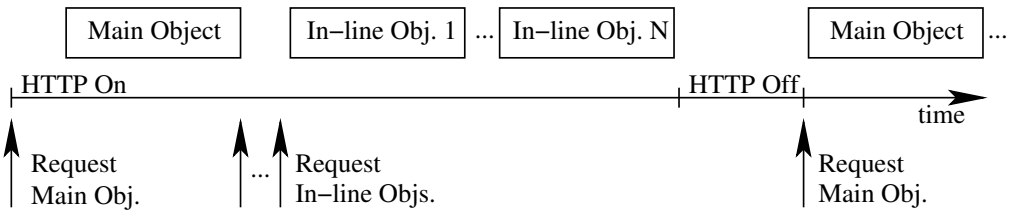


Figure 3.1: HTTP/TCP behaviour, no parallel connections.

The on-off behaviour is described in the following. The TCP connection is opened before data transmission via the three way handshake. The HTTP request for the main object (i.e. a HTML file) is sent by the client directly after the third signalling packet. The HTTP requests for the embedded objects (inlines, e.g. images) are sent from HTTP client to server after receiving the main HTTP object. The TCP connection is closed by the client after receiving all requested objects. The user spends a certain amount of time reading after successful transmission (off-time). The whole process starts from the beginning after the off-time has elapsed. Further discussion of the distributions and their parameter fittings of the user model can be found in [CTB98, PKC97].

The HTTP object size is drawn from a Truncated Power-Tail (TPT) distribution with a truncation level $T = 20$ [GJL95, GGS97] (see Sec. 2.8 for a detailed discussion). The major contribution to the self-similarity of the traffic comes from the distribution of the file-sizes [CTB98, PKC97]. Therefore, a geometric distribution is used for the number of HTTP inline objects and a negative exponential distribution is used for the off-time, deviating from [CL99].

The average value for the HTTP download volume is approximately 60 KB [CL99]. TCP fragments this volume into 42 packets when using a Maximum Segment Size (MSS) of 1460 Bytes which is typically used for Ethernet. The average connection is therefore short-lived; TCP requires only seven round-trip times to transmit the average download volume without packet losses. The download volume is split into a main object and several inline objects (see Fig. 3.1). When the whole HTTP download volume is transmitted in one TCP connection, as it is the case in this model, the differentiation between sizes for main object and includes (as measured in [CL99]) is of minor relevance. Therefore, the size of main object and includes is drawn with the same parameters from the TPT distribution. The average number of HTTP objects was measured to be approximately 6 and the average off-time was 40 s in [CL99]. The average size of the HTTP objects was set to 10 KB such that the resulting average download volume is $6 \cdot 10$ KB.

It was easier to keep the webserver very simple and to implement the distributions and parameters at the client side, from an implementation point of view. Therefore, the client transmits the size of the data object, that the server should send back to the client, with the HTTP request (in this implementation).

The complete on-off behaviour can be summarised as follows:

1. An off-time is drawn from the uniform distribution in $[0, t_{off}]$ in order to desynchronise the start of the clients.
2. The number of objects is drawn from the geometric distribution.
3. The size of the first object is drawn from the TPT distribution.
4. The client opens the TCP connection and sends the HTTP request for the first object to the webserver.
5. The webserver sends the first object to the client.
6. The size for all remaining includes is drawn from the TPT distribution and the requests are sent to the webserver after successfull receiving the first object.
7. The server sends the data for all remaining objects.
8. The client closes the TCP connection after successfully receiving all objects.
9. The client draws an off-time from the negative exponential distribution. The procedure starts again from 2 when the off-time is elapsed.

3.2 Protocol Overview

The Open System Interconnection (OSI) reference model and the TCP/IP protocol stack are sketched in Fig. 3.2 [Tan02]. Ethernet and the Point to Point Protocol (PPP) are the mostly used implementations of the “Physical” and “Data Link” layers of the OSI protocol stack. The Internet Protocol (IP) serves as “Network” layer. The “Transport” layer is implemented by the Transmission Control Protocol (TCP), which assures a reliable transmission, and the User Datagram Protocol (UDP), which offers only unreliable transmission. The “Session” and “Presentation” layer of the OSI model are not present in the TCP/IP layer model. The application level protocols used in the TCP/IP model are HTTP, FTP, SMTP and others.

The source model discussed in Section 3.1 uses the Protocols HTTP, TCP and IP, which are the mostly used protocols in the Internet. The data encapsulation is visualised in Fig. 3.3. The application layer data (HTTP payload) is prepended with a TCP header and an IP header. The TCP and IP header formats are discussed in Section 3.2.3 and 3.2.2, respectively. The Ethernet layer is neglected here since it plays only a minor role when using the nowadays widely distributed switched Ethernets in the LAN where no collisions can occur.

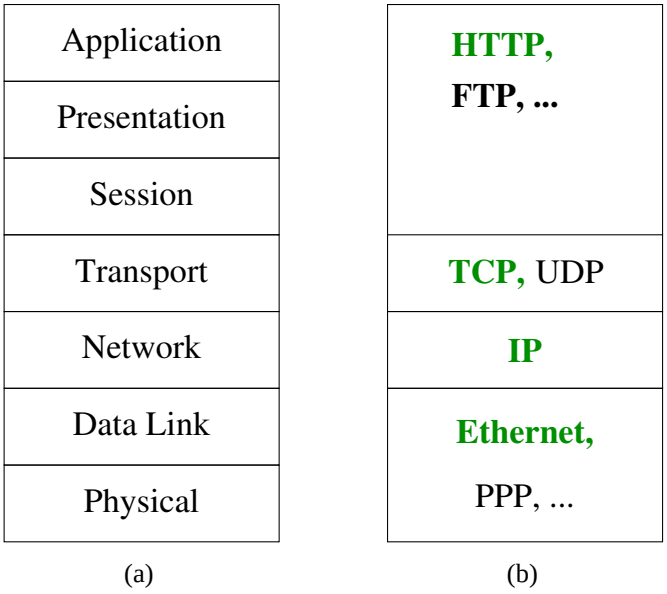


Figure 3.2: (a) OSI reference model and (b) Internet protocol stack.

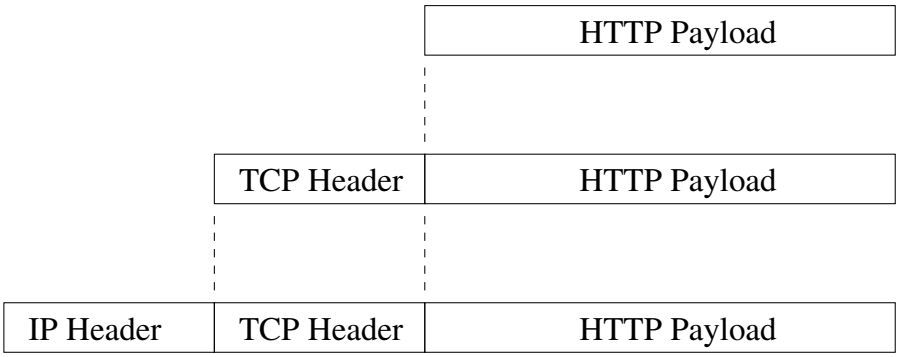


Figure 3.3: Data encapsulation Internet protocol stack.

3.2.1 HTTP Protocol

The HyperText Transfer Protocol (HTTP) is used in the Internet as an application level protocol (see Section 3.2, Fig. 3.2) for retrieval of WWW data, e.g. HTML pages. The protocol has been improved over the time with the target of a better throughput and smaller latency. Currently there are two major variants existing, HTTP 1.0 [BLFF96] and HTTP 1.1 [FGM+97], both variants can be used with or without persistent connections.

The oldest variant, HTTP without persistent connections, opens a new TCP connection

for the download of each HTTP object. The download of multiple objects was scheduled serially, the download of the next object could be started only after the current object was transmitted successfully and after closing the current TCP connection. Always closing and re-opening the TCP connection introduces a large delay and signalling overhead. TCP always has to start with minimum rate in slow-start in this case. This leads to poor performance in terms of long delays and small throughput, especially when a download consists of a large number of objects, as it is apparent for many web pages nowadays.

Therefore, persistent connections have been implemented for HTTP 1.0 to overcome this performance problem (option keep-alive); HTTP 1.1 uses persistent connections by default. The requests from the client and the responses from the server can be pipelined in a single TCP connection when this option is enabled. Another option implemented for both HTTP versions is the usage of parallel connections. This option does not necessarily increase the download speed, but it always increases the load on server and client. However, current web browsers use HTTP 1.1 with persistent and parallel connections. Nevertheless, the source model implemented in this work uses persistent but no parallel connections, as discussed already in Sec. 3.1.

3.2.2 TCP Protocol

The format of the TCP segment header is displayed in Fig. 3.4. The 16-bit source and destination port are used to address special applications at the server side and to differentiate several concurrently running client applications. The sequence number identifies the individual packet in the data flow of one connection. The receiver signals the last successfully received packet with the acknowledgement number to the sender. The "Window" field contains the number of bytes that the sender of this segment is willing to accept, beginning with the number indicated in the acknowledgement field. The data field is optional, it is not used for e.g. signalling. See [Tan02] for more information about specific header fields of the TCP segment.

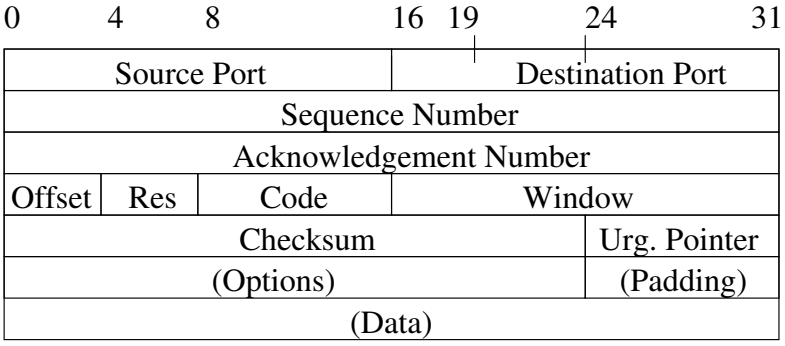


Figure 3.4: TCP segment.

3.2.3 IP Protocol

Ethernet serves as the physical layer for computers connected to a Local Area Network (LAN) and PPP is used for dial-up connections. This work concentrates on Ethernet since most users in the German research institutes are connected via an Ethernet LAN to the B-WiN network (see Section 5.6.3). It can be expected that all networks are switched networks, without broadcast problems like collision detection. Therefore, the functionality of Ethernet reduces to adding a header to the IP packet which can therefore be easily considered by adjusting the bandwidth limitations accordingly.

The IP protocol is the lowest layer modelled in the simulations in this work. The main tasks performed by the IP protocol are addressing, performed at client and server side, and the routing of the packets, performed by a router module in the node model. Fragmentation of IP packets is very unlikely since most Internet devices support the maximum size of Ethernet packets of 1500 Bytes and e.g. PPP uses smaller maximum packet sizes to increase the responsiveness of the connections due to the small bandwidth of e.g. modem connections. The IP datagram header format is displayed in Fig. 3.5, the length of the fields are displayed with bit-counts at the top. Source and destination address are used for routing the packet through the network. The maximum size of an IP datagram is 65535 Bytes. See [Tan02] for more information about specific header fields of the IP datagram.

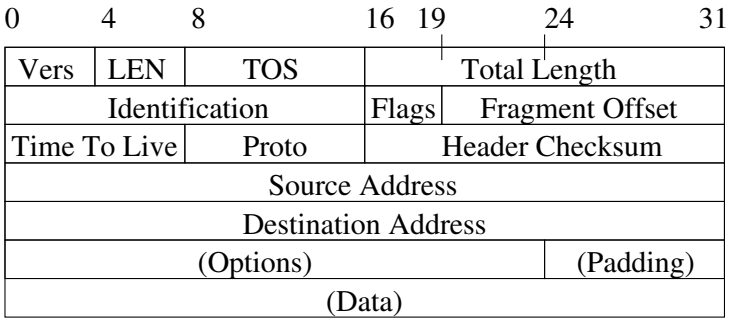


Figure 3.5: IP datagram.

3.2.4 Dynamics of TCP

The dynamics of the TCP protocol are very important in order to understand performance problems in the Internet. There are several variants of TCP existing: Tahoe, Reno, NewReno, RenoSACK, Vegas, etc. This work concentrates on the currently mostly used TCP NewReno. The implementation TCP NewReno for Ptolemy was validated with the simulator ns-2 [UCB].

The dynamic behaviour can be separated into three main parts:

- Slow-Start (SS)
- Congestion Avoidance (CA)
- Time-Out / Fast Retransmission / Fast Recovery

The congestion window (CWND) denotes the number of bytes (or segments) the sender is allowed to send without waiting for an acknowledgement from the receiver. The dynamic behaviour of the congestion window is depicted in Fig. 3.6. The CWND is normalised here to the maximum segment size and the time representation is simplified as a point process that has only events at integer multiples of the Round-Trip Time (RTT). The congestion window is initialised to one segment before a connection is established. Every acknowledgement received from the client, that indicates a successful delivery of a packet, provokes a doubling of the congestion window in slow-start phase.

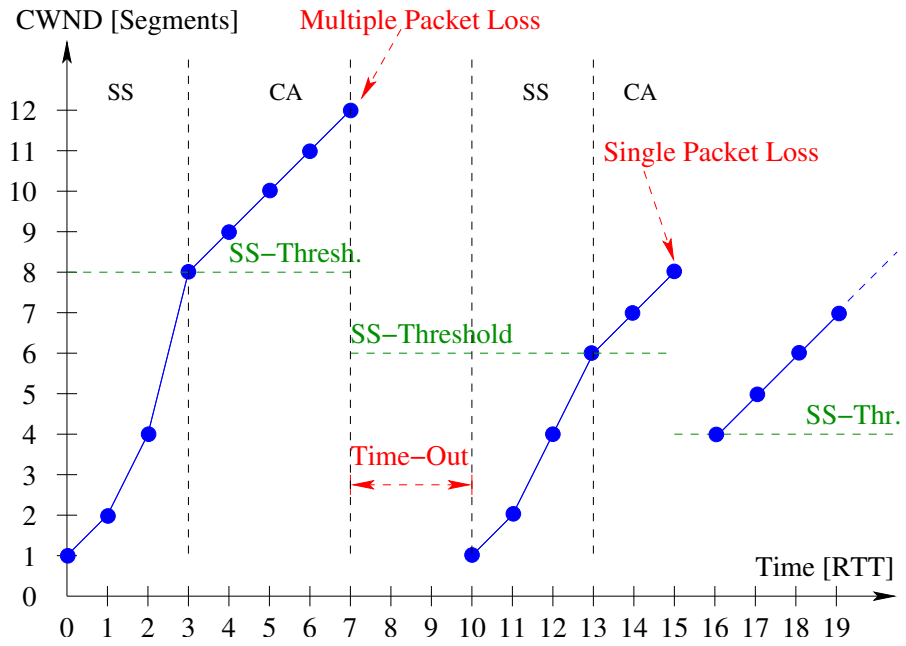


Figure 3.6: TCP dynamics: congestion window (CWND) over time, slow start (SS), congestion avoidance (CA) and time-out.

TCP migrates to congestion avoidance once the slow-start threshold (SSTH) has been reached. The value of SSTH is initialised with the maximum congestion window size, usually 65535 Bytes, before opening a connection. The threshold is reduced to half of the

CWND for each detected packet loss. Packet loss can be detected by receiving three duplicate acknowledgements or by a time-out. Duplicate acknowledgements are acknowledgements for a packet that has already been acknowledged; therefore three duplicate acknowledgement mean four acknowledgements for the same packet.

If the packet loss was detected by three duplicate acknowledgements, fast retransmission and fast recovery are used, which can effectively retransmit the lost packet(s) (cf. Fig. 3.6, time 15). The throughput does not go down to zero, as for a time-out.

TCP experiences a time-out if too many packets are lost or if the congestion window was too small such that three duplicate acknowledgements could not be received (cf. Fig. 3.6, time 7). The duration of a time-out is variable, it depends on the RTT estimated by Karn's algorithm [KP87]. For the first round-trip, when no measurements of the RTT are available, a default time-out value of 6 s is used. The value of 6 s from the first round-trip is used to calculate the time-out for the second round-tip. The default value for the RTT variance, for which no measurements are existing in the second round-trip, lead to a time-out value of 12 s for the second round-trip. Real measurements are used from the third round-trip on to estimate a useful value for the time-out. The default granularity for RTT and time-out calculations can be 500 ms or 100 ms, depending on the implementation. The latter value is used in this work.

A time-out duration of several seconds leads to a crucial throughput collapse of the TCP connection. Short-lived connections have a relatively larger probability to collapse than long-lived connections, since their congestion window is rather small: the average HTTP download volume of 60 KB is transmitted in 7 RTTs. A packet loss in the first two of the seven round-trips leads to the collapse. Therefore, short-lived connections require a small packet loss probability, otherwise the user has to wait for several seconds for a download that could theoretically be finished in less than one second. See [SW01] for more detail about TCP dynamics.

3.3 Bandwidth Limitation

In the real world there are several types of bandwidth limitations in the network:

- the capacity of the customer link (e.g. Modem, Integrated Services Digital Network (ISDN), Digital Subscriber Line (DSL), Ethernet),
- the capacity of the webserver,
- the capacity of the webserver's link to the Internet,
- the capacity of the links between the customers and the webservers network provider.

The limited customer capacity is implemented as a rate limitation at the IP layer. It is possible in this implementation to adjust different limits for upload and download rate such that ADSL can be modelled. The capacity limitation of the webserver itself is modelled by a maximum number of acceptable connections and a throughput limitation. Delays due to disk or database access in the webserver are not modelled yet, but could be easily implemented. The capacity of the webserver's link to the Internet, or of some servers sharing this link, is modelled by a queue and server which model a switch or access router with a limited outgoing capacity. The link capacities between the customers and the webservers network provider are also limited, the implementation is discussed in Section 5.4.

The speed of the core network is nowadays in most cases larger than a single connection can usually occupy. Therefore, it is important for the estimation of the throughput of a single client to take into account the limitations on the edges (webserver and client link). In most relevant cases, the webserver will also have a larger speed than the client and therefore the clients access speed remains the important limitation of the throughput per connection. The limiting effect on the client side is not only the limited bandwidth, but also the delay introduced by i.e. a slow Modem line can be of importance when estimating the throughput of a TCP connection.

This work is focussing on the B-WiN network model, as discussed in Section 5.6.3. It is a model of a network connecting the research institutes in Germany. Most of the traffic is generated by researchers (or students in a residential accommodation) connected via Ethernet to the core. In this case it is sufficient to model the bandwidth limitation – the delay in the local network is negligible as compared to the delay in the Internet. Therefore, only the throughput limitation of a 10 Mbit/s Ethernet line is modeled. Nevertheless, the change of the parameters to incorporate lower bandwidth limitations is very easy, the simulator is prepared for this task.

3.4 TCP-modified Engset Model

The TCP-modified Engset model was proposed by Heyman et. al. [HLN97] to provide an analytical solution for WWW traffic, that is characterised by short transfer volumes and idle periods. It describes the performance for a single bottleneck topology for the case of a limited number of customers. The model is based on a birth-death process which describes the number of simultaneously active users. Different queueing schemes are considered such that also IP over ATM with e.g. early packet discard can be modeled. The model assumes a fair share of the bandwidth between the TCP connections. Furthermore, the model is insensitive to the distributions used for on- and off-phases, that is, only the mean values are significant for the performance parameters. The model

provides analytic solutions for the following performance parameters: steady state probability that j users are active at the same time, average download time, average throughput on the bottleneck link.

A summary of the model with the focus on the case of First In First Out (FIFO) queues is presented in the following. Furthermore, only the case of packet loss events as the loss of the full packet are covered here, as opposed to when modelling e.g. ATM. The bottleneck scenario is sketched in Fig. 3.7. The capacity of the bottleneck link is denoted with C and the propagation delay with D. A total of N TCP clients $S_1 \ldots S_N$ are connected through symmetrical, bi-directional links with capacity c and propagation delay d to the bottleneck router R_1. The connection of the destinations $D_1 \ldots D_N$ to the router R_2 are not limited in capacity and delay. The destinations model the webservers here, which usually have a high-speed connection to the internet and the main limitation is modeled in the access lines of the user and the one bottleneck link in the internet.

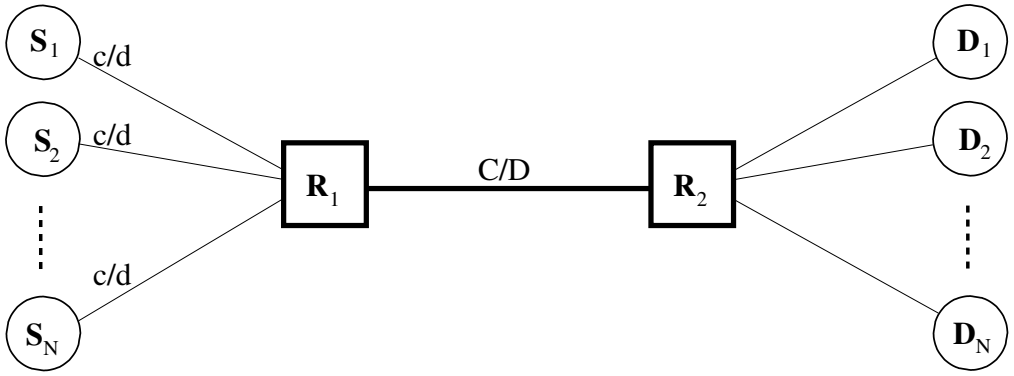


Figure 3.7: Single bottleneck, Engset model.

The ratio between access line capacity and bottleneck capacity describes the number of users, that can be served without congestion on the bottleneck link

$$s = \left\lfloor \frac{C}{c} \right\rfloor . \tag{3.1}$$

The normalised buffer capacity b can be specified with the buffer size B in packets, the minimum RTT (empty queues) and the service time for a data packet $\tau = MTU/C$ as

$$b = \frac{B}{RTT_{min}/\tau}. \tag{3.2}$$

The user behaviour is characterised with the average idle time (e.g. $t_{off} = 40$ s)

$$t_{off} = 1/\lambda, \tag{3.3}$$

with the idle time rate λ, i.e. the number of idle-times per time unit. The average download time without congestion $1/\mu$ is defined similarly with the average download volume v (e.g. 60 KB) as

$$\frac{1}{\mu} = \frac{v}{c}. \tag{3.4}$$

The average efficiency ρ of the bottleneck captures the case that the number of packets in the system are not enough to fill the link. A throughput degradation is occurring in this case, due to e.g. buffer capacity which is not sufficient. The efficiency is defined as

$$\rho = 1 - \frac{\left(1 - \frac{1}{2}(b+1)\right)^2}{b+1}, \tag{3.5}$$

with the positive-part operator defined as $x^+ = max\{x, 0\}$.

The assumption of a fair bandwidth share between all TCP connections leads to the rate $r(j)$ for j simultaneously active users

$$r(j) = \begin{cases} c & \text{for } j \leq s \\ \rho \cdot C/j & \text{for } j > s \end{cases}. \tag{3.6}$$

The slow-start behaviour of TCP is not captured here: the TCP connection starts with one packet per RTT and doubles the number of packets per RTT if all packets have been transmitted successfully.

The steady state probabilities of the birth-death process J, where j is the number of simultaneously active users, can be specified as

$$P_j = P_0 \cdot \underbrace{\frac{\beta_0 \cdot \beta_1 \cdot \cdots \cdot \beta_{j-1}}{\delta_1 \cdot \delta_2 \cdot \cdots \cdot \delta_j}}_{\gamma_j}, \quad j = 1, \ldots, N \tag{3.7}$$

with the birth rate

$$\beta_j = (N - j) \cdot \lambda, \quad j = 0, \ldots, N \tag{3.8}$$

and the death rate

$$\delta_j = \begin{cases} j \cdot \mu & \text{for } j = 0, \ldots, s \\ \rho \cdot C/v & \text{for } j > s \end{cases}. \tag{3.9}$$

The parameter P_0 can be calculated with the fact that the sum of the probabilities is always one

$$\sum_{j=0}^{N} P_j = 1 \tag{3.10}$$

as

$$P_0 = 1/\sum_{j=1}^{N} \gamma_j. \tag{3.11}$$

The solution for P_j can now be calculated with Eq. (3.1)–(3.11). The steady-state solution P_j and the rates $r(j)$ are used in the following to derive the parameters describing the performance of the network model.

The average number of active users can now be specified with the result of P_j as the expectation value of the process J (number of active users)

$$E[J] = \sum_{j=1}^{N} j \cdot P_j. \tag{3.12}$$

The total average throughput on the bottleneck link is then given by

$$E[J \cdot r(J)] = \sum_{j=1}^{N} j \cdot P_j \cdot r(j). \tag{3.13}$$

Finally, the average download time for a web page is the average volume v multiplied with the average number of active users divided by the total average throughput:

$$t_{on} = \frac{v \cdot E[J]}{E[J \cdot r(J)]}. \tag{3.14}$$

The average throughput of a single user follows as average page size divided by the sum of on- and off-time:

$$r_{avg} = \frac{v}{t_{on} + t_{off}} = \frac{v \cdot E[J \cdot r(J)]}{v \cdot E[J] + t_{off} \cdot E[J \cdot r(J)]}. \tag{3.15}$$

The slow-start behaviour of TCP is not taken into account here, the throughput of a real TCP connection does not start directly with rate c. Therefore, it can not expected that short downloads are modelled accurately. Furthermore, TCP time-outs are not covered. Hence, the accuracy for high packet loss probabilities might be limited.

Chapter 4

Algorithm for Simulation Setup

The solution for the inverse problem characterised in Chapter 2 is presented in this section. The first part of the solution – the iterative algorithm determining the number and distribution of clients – is presented in Sec. 4.1. The input parameters for the algorithm are the target throughput matrix, the average HTTP transfer volume, the maximum segment size of TCP, the maximum bandwidth of the path from client to the first hop, the link capacities, the propagation delays and the routing table. The second part – adjusting the higher order statistics to prescribed values, especially the Hurst parameter and the coefficient of variation – is addressed in Sec. 4.2 and 4.3.

The algorithm and equations presented in this chapter propose a solution for the inverse problem of how to setup the simulation such that the simulation model produces results which closely match a corresponding network in reality. It does not claim to be a complete model for the performance as the Engset model presented in Sec. 3.4. The simulation results of scenarios configured with the equations presented in the following are compared with the analytic results from the Engset model in Chapter 6.

4.1 Iterative Algorithm for Allocation of Clients

The algorithm determining the number of clients per flow, required for generating traffic according to a prescribed throughput matrix, is described in this section. The number of clients in the flow from node i to node j is denoted by s_{ij} in the following. The outline of the iterative algorithm for the allocation of sources is given as follows, see Algorithm 1. An initial estimation of the required number of clients per flow is performed following the equations given in Sec. 4.1.1. The first simulation is performed with the resulting allocation of clients. The throughput matrix established in the simulation is compared to the prescribed throughput matrix. An error measure must be defined to quantify the

deviation of the two throughput matrices (cf. Sec. 6.2). Subsequent simulations are performed, which reduce the error by correcting the number of clients based on the measurements of the previous simulation, until the error is smaller than a certain prescribed error limit ε or a maximum number of iterations k_{max} is reached. The maximum absolute error of all flows is in the first iteration in most cases already below 10 % so that no iterations might be required, depending on the error that one is willing to accept.

An early state of this algorithm with coarse estimation of the client population $s_{ij,1}$ was presented in [BK02c]. This study focused on the iterations, since the initial client population was not accurate enough. An improved estimation of $s_{ij,1}$, leading to smaller errors, was published in [BK02a] and [BK02b]. An improvement of the throughput estimation for TCP's congestion-avoidance phase and a methodology for adjusting the Hurst parameter of the traffic was published in [BK03a]. The protocol overhead and the traffic from client to server was additionally considered in [BK03b] leading to further improvements in estimating $s_{ij,1}$. Furthermore, a methodology to reduce the simulation complexity by reducing the number of clients was also published in [BK03b]. An extensive article summarising major findings of this work will be published in [BK04].

The equations for the estimation of $s_{ij,1}$ are described in Sec. 4.1.1. The incorporation of a bandwidth limitation at the client side into the estimation of $s_{ij,1}$ is discussed in Sec. 4.1.2. The equations describing how to adapt the number of clients per flow for further iterations are described in Sec. 4.1.3.

---

1. The initial allocation of clients $s_{ij,1}$ is calculated based on the equations derived in Sec. 4.1.1.
2. An initial simulation is performed with client matrix $s_{ij,k}$ with $k = 1$.
3. The error measure is evaluated comparing measured and prescribed throughput matrix.
4. While (error $> \varepsilon$ AND $k < k_{max}$):
   (a) Determine allocation of sources $s_{ij,k+1}$ from $s_{ij,k}$, based on the difference of prescribed traffic matrix and the corresponding results of simulation k as detailed with equations in Sec. 4.1.3.
   (b) Perform simulation $k+1$ with $s_{ij,k+1}$ to determine max. error.
   (c) Determine the error ε of simulation $k+1$.
   (d) Increment k by one.

Algorithm 1: Algorithm for allocation of clients.

4.1.1 Estimation of the Client Allocation

The traffic matrix TP is typically given by measurements, the elements tp_{ij} denote the throughput of the flow between node i and node j, e.g. in Mbit/s. The target is to estimate a client allocation S with elements s_{ij} generating traffic according to TP. An estimation of the average throughput of a single HTTP/TCP client is derived that can be applied for all flows so that the client allocation S can be derived. The average throughput of a single HTTP/TCP client in the flow from node i to node j is

$$r_{server,ij} = \frac{v+o}{t_{on,ij}+t_{off}}, \tag{4.1}$$

where v and o represent the average HTTP transfer volume and the protocol overhead, respectively. The parameters $t_{on,ij}$ and t_{off} are the average HTTP on- and off-times in seconds, respectively (download time and reading time). The distributions of t_{off} and v and their respective parameters are considered to be sufficiently secured by measurements of the user behaviour [CL99].

The value of $t_{on,ij}$ depends on the download volume v and on the competition among all connections about the resources in the network. Measurements of the user behaviour have shown that the value of t_{off} is much larger than $t_{on,ij}$ [CL99], so that t_{off} and v dominate Eq. (4.1). A simple estimation of $t_{on,ij}$ can already be sufficient in this case for a good estimate of the average throughput $r_{server,ij}$. However, it might not be always the case that t_{off} is so much larger than $t_{on,ij}$ that the latter can be completely ignored. Therefore, $t_{on,ij}$ can be calculated as

$$t_{on,ij} = n_{RTT} \cdot rtt_{ij}, \tag{4.2}$$

where rtt_{ij} is the average RTT and n_{RTT} is the average number of RTTs required by TCP to transmit the volume v. With the simplifying assumption of a uniform packet loss probability for all flows, the parameter n_{RTT} does not depend on the flow. The average number of RTTs n_{RTT} can consequently be calculated as constant for the whole network. The average round-trip time rtt_{ij} depends on the networks dynamics and can not be known exactly in advance. Furthermore, the value of rtt_{ij} depends strongly on the considered flow.

A lower bound for the RTT can be calculated by adding up the corresponding link propagation delays for each flow. This lower bound is used as an approximation for the average RTT here. The queueing delay can hardly be estimated, since the packet arrival process can not be expressed as a simple Poisson process, for which the solution for queueing systems are available.

The number of required round-trip times n_{RTT} for the transmission of v Bytes can be calculated as follows:

$$n_{RTT} = 1 + n_{RTT,SS} + n_{RTT,CA}. \tag{4.3}$$

The value 1 RTT in (4.3) represents the first two packets of the TCP connection establishment with the three-way handshake. The third packet of the three-way handshake can be directly followed by the HTTP request, so that it should not be accounted for in (4.3). The on-time is finished when the client receives the last data packet; the last acknowledgement and the signalling packets for closing the connection are counted as off-time here, since they do not contribute significantly to the throughput. The parameters $n_{RTT,SS}$ and $n_{RTT,CA}$ represent the number of RTTs that TCP requires to transmit v Bytes in slow-start and congestion avoidance phase, respectively, including the HTTP requests. The total number of packets n_P for transmitting v Bytes is

$$n_P = \left\lceil \frac{v}{MSS} \right\rceil \tag{4.4}$$

with the Maximum Segment Size MSS of TCP packets (the standard value of $MSS =$ 1460 Bytes for Ethernet is used here). All the packets and requests contain an overhead of h Bytes and are acknowledged with the same overhead. Therefore the overhead is the same for client and server. The protocol overhead in Eq. (4.1) can be calculated as

$$o = (4 + n_P + n_{Req}) \cdot h, \tag{4.5}$$

where n_{Req} represents the average number of HTTP requests and h stands for the header size of each packet ($h = 40$ Bytes for TCP/IP packets). The constant value 4 in Eq. (4.5) represents the number of packets required for connection setup and release.

Combining (4.1), (4.2), (4.3) and (4.5) the result for the average throughput in download direction of a single HTTP/TCP client is:

$$r_{server,ij} = \frac{v + o}{(1 + n_{RTT,SS} + n_{RTT,CA}) \cdot rtt_{ij} + t_{off}}. \tag{4.6}$$

The throughput from client to server due to signalling, acknowledgements and HTTP requests has the opposite direction to the download, which can be considered by transposing the matrix. The throughput from client to server can be calculated as (please note the swapped indices at the left hand side)

$$r_{client,ji} = \frac{n_{Req} \cdot s_{Req} + o}{(1 + n_{RTT,SS} + n_{RTT,CA}) \cdot rtt_{ij} + t_{off}}, \tag{4.7}$$

with the average number of requests n_{Req} and the average size of a HTTP request s_{Req}.

The maximum number of packets that can be transmitted in one RTT is the ratio of the maximum congestion window and the maximum segment size:

$$n_{P,CWNDmax} = \left\lfloor \frac{CWND_{max}}{MSS} \right\rfloor . \tag{4.8}$$

The maximum number of RTTs that TCP stays in slow-start phase (without losses, cf. Sec. 3.2.4) is:

$$n_{RTT,SS,max} = \lceil \log_2(n_{P,CWNDmax}) \rceil . \tag{4.9}$$

The maximum number of packets that can be transmitted in slow-start, can be calculated as

$$n_{P,SS,max} = 2^{n_{RTT,SS,max}+1} - 1 \tag{4.10}$$

where $CWND_{max}$ denotes the maximum TCP congestion window, the default value is $CWND_{max} = 65535$ Bytes. The number of RTTs in slow-start is given by

$$n_{RTT,SS} = \begin{cases} \lceil log_2(n_P) \rceil & n_P \leq n_{P,SS,max} \\ n_{RTT,SS,max} & n_P > n_{P,SS,max} \end{cases} . \tag{4.11}$$

The number of RTTs in congestion-avoidance phase can be written as the remaining number of packets after slow-start phase divided by the average congestion window size:

$$n_{RTT,CA} = \left\lceil \frac{\max\left\{0, n_p - n_{P,SS,max}\right\}}{CWND_{CA,avg}} \right\rceil . \tag{4.12}$$

The average congestion window in congestion-avoidance phase depends on the packet loss probability p_{loss} [MSM97]

$$CWND_{CA,avg} = \left\lfloor \sqrt{\frac{3}{2 \cdot p_{loss}}} \right\rfloor . \tag{4.13}$$

However, since the actual loss probability p_{loss} is not known before carrying out a specific simulation, previous simulations have been evaluated. The packet loss probability with pure HTTP traffic was in most cases for all three simulation scenarios smaller than 0.1 % (with a buffer capacity of 2500 packets). A packet loss probability of 0.1 % might seem very small. However, the average connection duration is very small for HTTP/TCP

connections. Already small loss probabilities can lead to severe performance issues in this case, as opposed to e.g. long-lasting FTP connections. Using $p_{loss} = 0.001$ yields a value of $CWND_{CA,avg} = 38$ packets for the average congestion window in congestion avoidance phase.

The client allocation matrix S with the elements s_{ij}, representing the number of clients for the flow from node i to node j, can be calculated finally with Eq. (4.7) and (4.6) by dividing the observed throughput tp_{ij} by the average client download rate $r_{server,ij}$ for each flow and rounding the resulting numbers :

$$\begin{aligned} s_{ij} &= round\left(\frac{tp_{ij}}{r_{server,ij}} \cdot \left(1 - \frac{r_{client,ji}}{r_{server,ij}}\right)\right) \\ &= round\left(\frac{tp_{ij} \cdot \left(n_{RTT} \cdot rtt_{ij} + t_{off}\right)}{v+o} \cdot \left(1 - \frac{n_{Req} \cdot s_{Req} + o}{v+o}\right)\right) \\ &= round\left(\frac{tp_{ij} \cdot \left(n_{RTT} \cdot rtt_{ij} + t_{off}\right) \cdot \left(v - n_{Req} \cdot s_{Req}\right)}{v+o}\right). \end{aligned} \tag{4.14}$$

The factor $1 - r_{client,ji}/r_{server,ij}$ accounts for the throughput generated by the clients due to acknowledgements and HTTP requests. The reverse direction of this traffic flow is considered by swapping the indices indicating the flow direction from node i to node j. The factor is smaller than one since the download rate $r_{server,ij}$ is usually significantly larger than the upload rate $r_{client,ji}$ (cf. (4.6) and (4.7)). Please note that it is necessary to convert some units in (4.14) and other equations in order to get consistent results (e.g. factor $8/10^6$ from Bytes to Mbit for v, o and s_{Req}).

4.1.2 Bandwith Limitation on Client Side

The bandwidth limitation on the client side can be taken into account by

- adapting the formula for the maximum number of packets that can be transmitted in one RTT and
- adding the propagation delay of the client access lines to the RTT and
- changing the average throughput in congestion-avoidance to the download link capacity.

The maximum number of packets that can be transmitted in slow-start in Eq. (4.8) has to be modified in order to incorporate the download link capacity $C_{client,down}$ as a limiting

factor, because increasing the window does not lead to an increase of the sending rate anymore when the link capacity is reached:

$$n_{P,CWNDmax} = \min\left\{\left\lfloor\frac{C_{client,down}}{MTU}\right\rfloor, \left\lfloor\frac{CWND_{max}}{MSS}\right\rfloor\right\}. \tag{4.15}$$

The average RTT can be calculated in this case by adding the transmission time of a minimum sized packet ($h = 40$ Bytes for TCP/IP) in the upload direction and of a maximum sized packet (MTU) in the download direction to the sum of the propagation delays in the core network. The transmission time is negligible for e.g. 100 Mbit/s Ethernet access lines but it could have a significant impact on the RTT for e.g. very slow modem connections.

The full capacity of the download link can be reached in congestion-avoidance whenever there is enough buffer in the router of the Internet access provider. The download link can still be filled completely when the buffer contents bridges the time when a reduction of the congestion window is triggered on the server side due to a packet loss. The required router buffer capacity in packets per connection is the product of the minimum RTT and the download capacity divided by the MTU, see Eq. (4.16). The maximum value that the congestion window can reach in this case is twice the bandwidth delay product. The window is halved upon packet loss when it reaches this limit. This means that the rate falls down to exactly the download capacity. Therefore, TCP can completely fill a link of capacity $C_{client,down}$ if (4.16) is fulfilled under the condition that there is no other source for a packet loss than the limited buffer under consideration.

$$B \geq \frac{RTT_{min} \cdot C_{client,down}}{MTU}. \tag{4.16}$$

Table 4.1 lists Internet access technologies with typical values for the minimum RTT and the derived buffer capacities according to Eq. (4.16). The minimum RTT value is composed of a constant value representing a typical RTT between the ISP and the selected webserver and the propagation delay from the client to the ISP, which depends on the access technology. Several measurements with the tool traceroute have shown that only very few connections exceeded a minimum round-trip delay of 150 ms between ISP and webserver. A connection between a German ISP and `www.linux.org` in the Canada for example has an RTT of approximately 140 ms although it is routed over 24 hops. Longer delays can be experienced e.g. at connections from Germany to webservers in Taiwan (e.g. `www.asus.com.tw` with approximately 320 ms and 26 hops). However, values larger than 150 ms for the minimum RTT between ISP and webserver are not the standard case, but rather the rare exception. Therefore 150 ms was selected as typical minimum RTT. A special case must be considered for the RTT of DSL connections: interleaving is used here if the wire quality is not good enough, leading to a larger propagation delay.

Table 4.1: Required buffer capacity per connection at ISP Router for download throughput at full link capacity.

$C_{client,down}$ [bit/s]	MTU [Bytes]	RTT_{min} [ms]	B [Packets]
Ethernet:$10 \cdot 10^6$	1500	151	126
DSL: $2 \cdot 10^6/192 \cdot 10^3$	1500	180	30
DSL: $768 \cdot 10^3/128 \cdot 10^3$	1500	180	12
ISDN: $128 \cdot 10^3$	1500	180	2
ISDN: $64 \cdot 10^3$	1500	180	1
ISDN: $64 \cdot 10^3$	576	180	3
Modem: $56 \cdot 10^3/33 \cdot 10^3$	576	300	4

It can be expected that the router of an Internet Service Provider (ISP) has enough buffer capacity to store at least four packets per connection. Therefore, it can be noticed that clients with ISDN and modem connections can always achieve the full throughput of the download capacity. DSL and Ethernet connections on the other hand are unlikely to be filled completely, since the required buffer per connection is rather large and the server might also not be able to send data with such a high rate, especially when loaded with many other connections.

It can be summarised that Eq. (4.12) should be changed if the download capacity is smaller or equal to 128 kbit/s with $n_{P,CWNDmax}$ from Eq. (4.15) to

$$n_{RTT,CA} = \left\lceil \frac{\max\left\{0,\, n_p - n_{P,SS,max}\right\}}{n_{P,CWNDmax}} \right\rceil . \tag{4.17}$$

A DSL connection with 128 kbit/s upload and 768 kbit/s download capacity on the other hand represents no significant limitation for an average HTTP connection with a download volume of 60 KB: the required number of connections s_{ij} for such a DSL connection is only 2.8 % larger as compared to no bandwith limitation at the client side (for $t_{off} = 5$ s). The slightly larger RTT is responsible for the larger number of required connections in this case. However, for a large download volume and/or a very small off-time the average (and peak) throughput increases such that the download capacity can be reached.

4.1.3 Iterations Beyond the First Order Approximation

After the first simulation was driven with the client allocation described by the equations above, the deviation of the simulation results from the given traffic matrix can be further

reduced with iterations (cf. Algorithm 1, p. 38). It is assumed in the following that the deviation between the throughput matrix measured in the first simulation and the target throughput matrix is reasonably small. A linearisation of the non-linear relation between number of clients and throughput is performed here.

The achieved throughput per flow $tp_{ij,k}$ and the average client throughput per flow $r_{src,ij,k}$ need to be measured for every iteration step. The measurement of the achieved throughput can be used to calculate the difference between the prescribed and the observed traffic intensities:

$$\Delta tp_{ij,k} = tp_{ij} - tp_{ij,k}. \tag{4.18}$$

The difference matrix $\Delta tp_{ij,k}$ and the average client throughput per flow $r_{server,ij,k}$ are used to calculate the update for the client allocation $\Delta s_{ij,k}$

$$\Delta s_{ij,k} = \left\lceil \frac{\Delta tp_{ij,k}}{r_{server,ij,k}} \right\rceil. \tag{4.19}$$

The new allocation of sources can be calculated as shown in Eq. (4.20) with a constant $c < 1$ to prevent strong oscillation:

$$s_{ij,k+1} = s_{ij,k} - c \cdot \Delta s_{ij,k}. \tag{4.20}$$

The ceiling function in Eq. (4.19) introduces a non-linearity, that could lead to convergence problems together with the approximation by linearisation. Furthermore, it is important that the steady state of the simulation for each step k is reached to ensure convergence. It is shown in Sec. 6.2.4 that $c = 0.6$ is a good compromise between oscillation and convergence speed.

4.2 Parameters Influencing the Hurst Parameter

Realistic traffic models have to incorporate the generation of self-similar traffic since the self-similarity was measured in several studies in local area networks and in the Internet [FGW98, FGHW99, LTWW93, CB97, PKC97]. The parameters influencing the degree of self-similarity, the Hurst parameter, are discussed in this section in order to gain control over it.

A first approximation is described here, in which it is assumed that the major contribution to the self-similarity are the power-tail distributed file sizes. The Hurst parameter depends on the shape parameter α of the power-tailed file size distribution [PKC96a, CB97]. The shape parameter α determines the slope of the power-tail (cf.

Section 2.3.2). The aggregated traffic from an infinite number of on-off sources with power-tail distributed on and/or off times follows $H = (3-\alpha)/2$ [TWS97]. The assumption that this relation holds approximately also for a limited number of TCP connections is verified in Sec. 6.4.2. Therefore, we assume that

$$H \approx (3-\alpha)/2. \tag{4.21}$$

This is, however, not the only effect that influences the Hurst parameter. It has been shown in [GCM01] that the loss probability p_L has an impact on the Hurst parameter of TCP traffic:

$$H = (3-\log_2(\frac{1}{2p_L}))/2. \tag{4.22}$$

The region where the loss rate has an impact on the Hurst parameter can be calculated from Eq. (4.22) by simple transformations. The result (4.23) shows that this effect is only important for very high loss probabilities of more than 12.5 %. All the simulations are performed at significantly lower loss rates and therefore this effect can be neglected in this work.

$$p_L \geq 2^{-(\alpha+1)} \geq 0.125 \tag{4.23}$$

Riedi et al. [RW00] have discovered that the Hurst parameter follows Eq. (4.24) if the distribution of the rates of the connections on a link is a power-tail distribution with shape parameter β (α is the shape parameter of the file size distribution).

$$H = \frac{\beta-\alpha+1}{\beta} \tag{4.24}$$

In the case of $\beta = 2$, i.e. finite variance of the connection rate distribution, Eq. (4.24) reduces to Eq. (2.12). Note that the Hurst parameter is smaller for $\beta < 2$ as compared to the case of finite variance for a constant shape parameter α. Therefore, the power-tail of the rate distribution tends to cancel out the effect of the power-tailed file size distribution.

The traffic load has also an impact on the Hurst parameter. Packet inter-arrival times and counting process show self-similarity and long-range dependence for small utilisation values. It has been shown that the long-range dependence fades out and the distribution of packet inter-arrival times tends to an exponential distribution when the load increases [CCLS01]. However, the simulation studies performed in this work also identify a case where the Hurst parameter increases with the traffic load. An extensive discussion about this phenomenon is presented in Sec. 6.4.3.

4.3 Parameters Influencing the Coefficient of Variation

The coefficient of variation $cv = \sigma/\mu$ is studied in this work for the inter-arrival times of packets entering a queueing system. The value of variance and mean of Poisson traffic are one and therefore also $cv = 1$. Values larger than one characterise a higher variability and values smaller than one a lower variability as compared to Poisson traffic.

The average traffic load has an impact on the value of *cv*: the room for variability is reduced when the traffic load approaches the link capacity. The number of connections responsible for a fixed amount of traffic also have an impact on *cv*, since the statistical multiplexing of a large number of connections leads to a smoothing effect, where the variance is reduced when the number of connections is increased. This effect is not so pronounced for self-similar traffic, but nevertheless, it is also present in this case (cf. Sec. 2.3.2 and 2.4).

Other important factors for the value of *cv* are the network layers 1 – 3: whether or not the IP packets are segmented at the sender and reassembled at the receiver affects the variance of the traffic as well as any additional delay introduced by these network layers, e.g. caused by collisions on a broadcast medium.

In ATM networks for example, one IP packet of size 1500 Bytes is segmented and transmitted in 32 ATM cells of fixed size (53 Bytes with 48 Bytes payload). Each packet on IP layer leads to a burst of ATM packets which has an influence on the variance of the traffic. Therefore, it can be expected that the value controlling the maximum packet size in TCP, the Maximum Segment Size (MSS), and the distribution of the actual packet sizes have an impact on *cv* as well.

The average download volume has also an impact on *cv*: TCP operates with a small congestion window. That is, only few packets are sent per RTT, resulting in significantly lower correlation and variance as compared to large download volumes with large congestion window sizes.

Among all discussed parameters, the link load is no free parameter of the simulation. But the MSS/MTU distribution and the exact average download volume of the users in the considered network might not be known. Therefore, the MTU distribution could be used to tune the value of *cv*, if it does not match the measured values in the real network (see Section 6.4.1).

Chapter 5

Simulation Methodology

This chapter covers the simulation methodology developed and followed in this work. Configuration and implementation aspects are presented in this chapter as well as the optimisation of the simulator regarding simulation speed and the memory requirements. The requirements on the simulation environment leading to the selection of Ptolemy are discussed in Sec. 5.1. Further, the basic principle of discrete event simulation is described in Sec. 5.2.

The tools developed in this work for an automatic configuration that easily scales up to several ten thousands of clients are presented in Sec. 5.3. The implementation and optimisation of the simulation models is discussed in Sec. 5.4 and 5.5. The optimised simulation model structure requires less memory and achieves a higher simulation speed. The three network simulation models used in this work are described in Sec. 5.6.

5.1 Simulator Requirements

The requirements on the simulator leading to the selection of Ptolemy are briefly discussed in this section. The simulator had to fulfil the following requirements at the time when the decision had to be taken:

- Graphical User Interface (GUI)
- Object oriented programming language
- Object oriented design of simulation entities
- Easy extensibility of existing code, preferably open source code
- Fast simulation

- Concept for automatic scaling of the number of sources
- Protocols to be simulated: HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), Transmission Control Protocol (TCP), Internet Protocol (IP), ATM Adaptation Layer 5 (AAL5) and Asynchronous Transfer Mode (ATM).

The GUI was required in order to allow students to use existing blocks by just connecting and configuring them without the need to program. An object oriented programming language should ease the programming and code maintenance in a group of PhD students. Another important feature of object oriented programming, the inheritance, was a very important feature for the simulation of communication networks: a Random Early Drop (RED) queue could be derived from a generic queue. Packet classes as well as TCP variants can be implemented easier using the concept of inheritance with generic prototypes. The demand for easy extension of existing code and preferably open source code can be justified by the experience of earlier projects: it was many times helpful to look at the source code of the simulator to find a solution for a problem. Further, to understand the performance bottlenecks of the simulator and to find solutions to avoid the performance bottlenecks is easier with open source software.

Since the target was the simulation of large networks, it is clear that a fast simulation is one of the very important reasons. This was the reason against simulators implemented in Java. The concept for automatic scaling of the number of sources is one principle that was already implemented in Ptolemy with the so called “Higher Order Functions” (HOF) . This principle allows to connect a source to such a HOF block and to specify the number of replications of the source. The re-wiring and configuration can be avoided and the number of sources is very variable. The protocols mentioned above were needed in the project called ERNANI [BSK01].

All requirements except of the very last one (set of protocols required) were fulfilled by Ptolemy classic [PCB]. Ptolemy classic is written in C++ and distributed with source code and precompiled binaries for many platforms by the university of California at Berkeley together with extensive documentation [Ber97a, Ber97b, Ber97c]. The other candidates, mainly ns-2 [UCB], did provide more protocols than Ptolemy. But ns-2 had weaknesses in the other requirements like a missing GUI and a missing concept for scaling the number of clients, at the time where the decision had to be taken. Therefore Ptolemy was selected as the base framework for the simulations. The TCP/IP stack was implemented and verified against ns-2.

5.2 Discrete Event Simulation

Discrete event simulation is a major simulation principle used for the simulation of communication networks. Discrete event simulation can be used to simulate asynchronous,

non-periodic data transmission and is therefore well suited for simulation of communication networks. The principle is based on the existence of a global scheduler with an attached global event list. An event consists of a time stamp, the data to be transmitted to the receiver block and a reference to the receiver block. Actors can generate and consume events. The global event list is sorted according to the time stamps of the events. The event with the smallest time stamp is selected by the scheduler and the corresponding receiver block is being executed by the scheduler. The time stamp has a high floating point resolution. The time between two consecutive events can be arbitrarily large or small. Also simultaneous events with exactly the same time-stamps are possible. The simulation time does not necessarily advance in fixed time units as it is the case in other models of simulation used for e.g. digital signal processing.

A schematic of the scheduler and three blocks is sketched in Fig. 5.1: the global event queue holds sorted events with a time stamp and destination block ID. The next event is to be processed at time 2.1, destined for block C. A feedback event of block A follows at time 3.2. The last event is scheduled for time 7.9 at block B. Whenever one of the three blocks generate a new event it is stored in the event queue in a way that the sorted order is conserved. The simulated time advances to the value with the smallest time stamp in the global event queue when the execution of the block has been terminated.

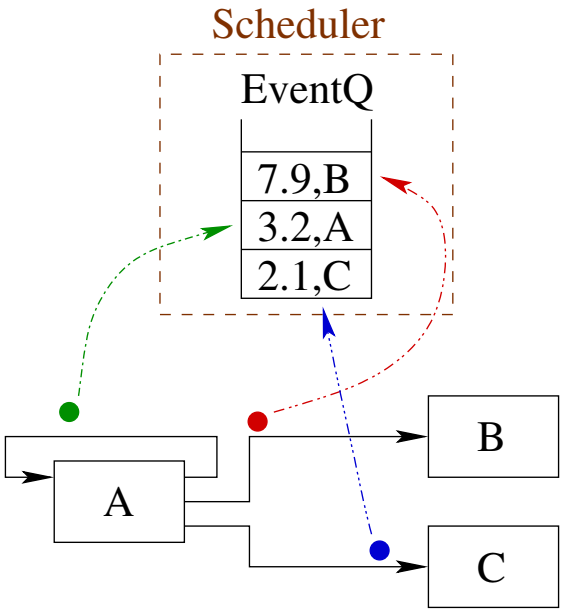


Figure 5.1: Discrete event simulation principle.

5.3 Tools for Automatic Configuration

A method for the automatic configuration is needed in order to implement and verify the solution for the problem of finding the appropriate number of clients for all flows, as described in Section 2.2. Perl (Practical Extraction and Report Language, `www.perl.org`) was selected for configuration, parsing and statistical evaluation of the results since it is fast (compiled at run time) and very well suited for parsing text files.

The following parameters need to be configured for the client module representing all clients attached to one node. The parameters are subject to changes whenever the number of clients is changing:

- number of clients for all flows
- client IP address
- webserver (destination) IP address
- enable or disable a feature only for a certain percentage of clients, e.g. Explicit Congestion Notification (ECN) for TCP
- enable or disable the aggregated UDP traffic model for a fraction of the traffic volume for each flow.

The architecture sketched in a Unified Modelling Language (UML) sequence diagram in Fig. 5.2 was designed to form a solution for the automatic configuration. The Perl script *generate-config.pl* loads the simulation topology described in a Perl script *simConfig.pl* (actual file names for the topologies are: *bottleneckConfig.pl*, *parkingLotConfig.pl* and *bwinconfig.pl*). The Perl script *configTools.pl* implements a library of methods for the configuration and statistical evaluation of the results. The configuration script *generate-config.pl* writes the text file *configbwin.txt* and returns the control to Ptolemy. The text file *configbwin.txt* is read and parsed by the C++ template class *Ptet6Config* (see Section 5.3.1) that loads all the data into static memory so that it is accessible in all blocks of the simulation. The main simulation run starts, after finishing the described setup part. The simulation data is written to text files within the simulation time and at the end of the simulation time.

The configuration data is organised in matrices, a string key distinguishes different matrices. The three data types integer, floating point and string are supported. The row index of the matrices corresponds to a node in the network and the column index is linked to one instance at the corresponding node. The values stored in the matrices can be e.g. IP addresses (string), delay and rate values (floating point) or the number of clients at one node (integer). The file *configbwin.txt* is organised as follows (see Fig. 5.3): each line starts with the data type (int/double/string), the key follows after a blank. The matrix

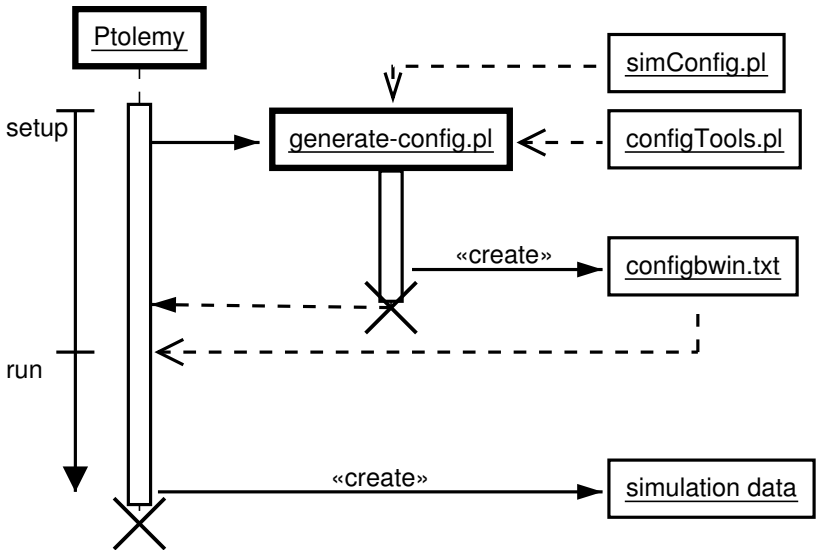


Figure 5.2: UML sequence diagram of configuration setup and simulation.

row and column number can be appended to the key, separated by dots (i.e. “delay.1.0”). The row and column numbers start from zero. The matrix reduces to a scalar if row and column number are omitted. A complete matrix row can be specified on one line with blank separated values if the column value is omitted. The matrix can contain rows with variable lengths.

The process of parsing the simulation results, one of the major reasons why Perl was selected, is sketched in the UML sequence diagram in Fig. 5.4. The script *get-results.pl* loads the description of the simulation topology from *simConfig.pl* (see description for *generate-config.pl* above) and the library *configTools.pl*. Furthermore, it reads the text file *configbwin.txt* and the simulation output data stored in text files. It writes summaries to text files and some Matlab scripts (*.m-files*). Matlab executes the generated scripts in order to produce figures in the format Joint Picture Expert Group (JPEG), Encapsulated PostScript (EPS) and the matlab figure format FIG. Matlab returns the control to *get-results.pl* which generates a LaTeX file (*.tex*). LaTeX is called, which reads the *.tex* file and the EPS figures and creates PostScript (PS) and Portable Document Format (PDF) documents (with help of the tools *dvips* and *ps2pdf*). The JPEG figures are included in the HTML overview pages generated by the script *get-results.pl*.

5.3.1 C++ Classes for Configuration

The C++ template classes used for storing the configuration data are described and visualised with UML diagrams in this section. The overview is presented in Fig. 5.5: the

```
# ATTENTION: File automatically generated by
# 'generate-config.pl'
#
string configScript bwinconfig.pl
#
int numSpecifiedSources 475
int MSS 1460
int MTU 1500
#
int sourceDistr.00 0   2   1
int sourceDistr.01 2   0   3
int sourceDistr.02 1   5   0
#
double throughputMat.00 0.0   7.2   5.5
double throughputMat.01 5.8   0.0  11.1
double throughputMat.02 4.6  15.8   0.0
#
string srcAddr.0 1.2.0.1 1.2.0.2 1.2.0.3 1.2.0.4
string srcAddr.1 2.1.0.1 2.1.0.2
string srcAddr.2 3.1.0.1 3.1.0.2 3.2.0.1
```

Figure 5.3: Example for configbwin.txt.

template classes *Ptet6Config*, *GenericMatMap*, *GenericMat* and *GenericVec* are derived by the template class *Ptet6BaseClass*, a class that provides an interface for test methods as well as warning and aborting the simulation with an error message. The template class *Ptet6Config* has three members of the template class *GenericMatMap*, one for each type integer, double and string which hold the configuration data. The template class *GenericMatMap* has one or more members of the template class *GenericMat*. The template class *GenericMat* holds one or more members or the template class *GenericVec*.

A detailed UML diagram of the template class *GenericVec* is shown in Fig. 5.6. The class owns a private member *vector*<*T*> from the Standard Template Library (STL) and three public methods for copying the contents of the array to Ptolemy's array states for the types integer, double and string.

The template class *GenericMat* owns three private members (cf. Fig. 5.7): one STL *vector* of type *GenericVec*<*T*>, an optional scaling factor stored in a double and a boolean indicating whether or not to invert the value. The last two members can be used to specify the values in a format more appropriate for the user and to use the scaling factor and inversion $(1/X)$ to transform it to the units required by the simulator. Example: a minimum packet inter-arrival time is specified instead of a maximum link capacity; the capacity can be calculated with a scaling factor and inversion as $C = const/IAT$. There are three

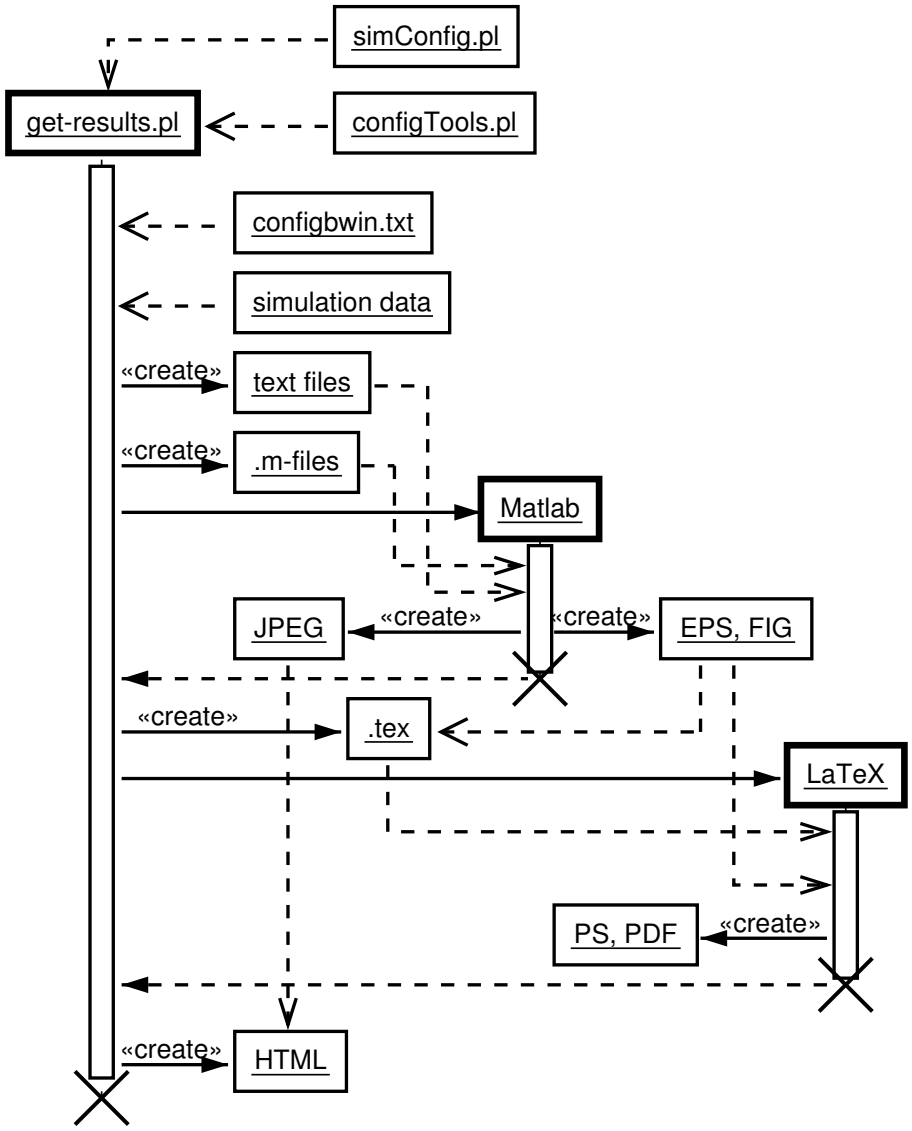


Figure 5.4: UML sequence diagram of parsing the simulation results.

methods implemented called *copyCol()* for copying matrix columns into Ptolemy state arrays, in addition to the copy methods of *GenericVec*.

The template class *GenericMatMap* holds an STL map that connects string keys with objects of type *GenericMat* as private members, as shown in Fig. 5.8. Several matrices, accessible via the string key, can be saved in an object of this class. The public method *keyExists()* can be used to verify the existence of a matrix with a specific key. The public method *getMat()* returns a pointer to the matrix associated with the key and thus gives full access to the matrix contents.

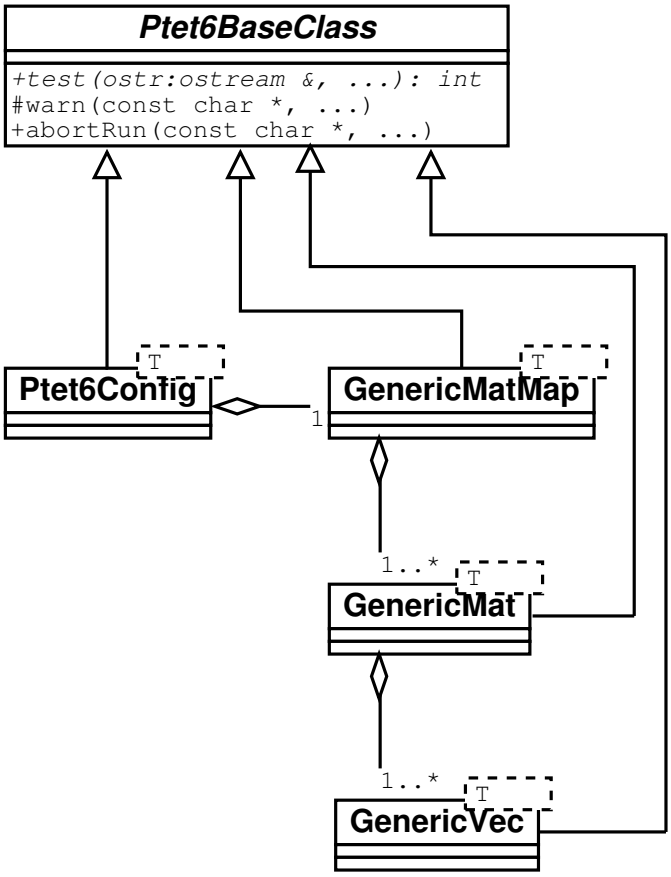


Figure 5.5: UML diagram class overview.

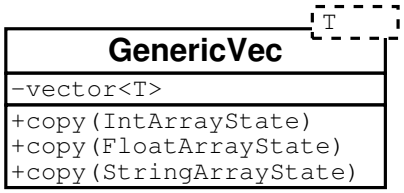


Figure 5.6: UML diagram of the class *GenericVec*.

The template class *Ptet6Config* has one private and static member of the template class *GenericMatMap* (see Fig. 5.9). The member is static to ensure that the data exists only one times in memory, even if many interface classes of type *Ptet6Config* are allocated. The public methods *readConfig()* and *parseKeyAndIndex()* are used to parse the file *configbwin.txt*, generated by the Perl script as described above. Three objects of the template class *Ptet6Config* are used in the simulation, one for each of the types integer, floating point and string.

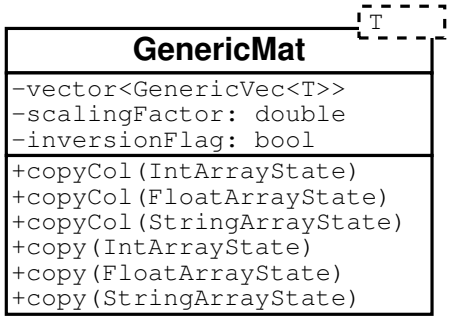


Figure 5.7: UML diagram of the class *GenericMat*.

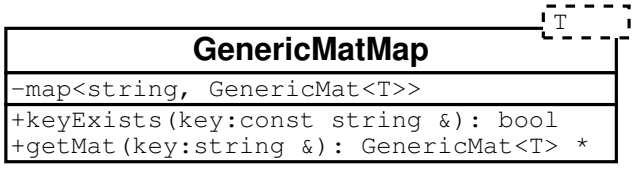


Figure 5.8: UML diagram of class *GenericMatMap*.

5.3.2 Ptolemy Base Star

Ptolemy is structured in analogy to the objects in the solar system: the basic block is called star, stars and galaxies can form a galaxy and the universe is composed of stars and galaxies. A universe is a simulation unit that can be executed.

The UML class diagram for *PtolemyBaseStar* is depicted in Fig. 5.10. The class members are present in all inherited stars:

- *starId*: a string identifier that is prepended in each line written to the log file
- *logDir*: the destination directory for the log files
- *verbose*: the verbosity level of debug output, values in the range $[0, 3]$

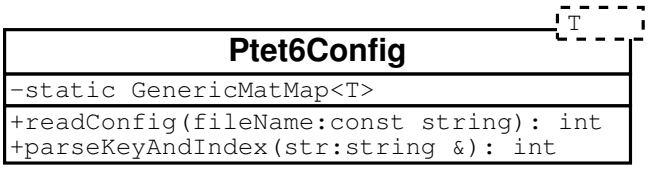


Figure 5.9: UML diagram of the class *Ptet6Config*.

- *usePtet6Config*: a boolean indicating whether or not to use the configuration stored in Ptet6Config
- *Ptet6ConfigDefaults*: a string array state for specification of which state should be filled with which values from *Ptet6Config*; a pair of state name and key plus the matrix row and column index.
- *intCfg*, *dblCfg* and *strCfg*: static members of the template class *Ptet6Config<T>* for the three types int, double and string to allow easy access within each derived star.

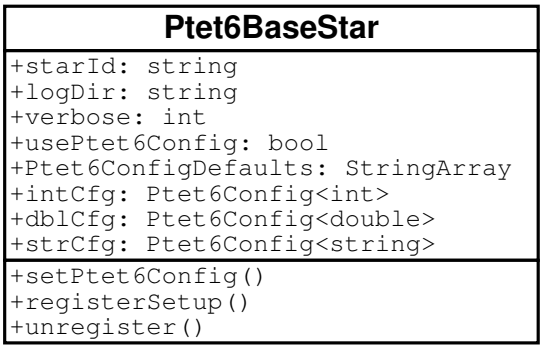


Figure 5.10: UML diagram of the star *Ptet6BaseStar*.

The star *Ptet6BaseStar* was developed to manage the following issues for all inherited stars:

- Opening and closing of files for writing log messages: The methods *registerSetup()* and *unregister()* allocate one file (Ptolemy ostream, *pt_ofstream*) per class at run time in order to save file handles. The same file handle is used by all instances of the same class. This is used to save file handles which are limited by the operating system. The Linux kernel typically allows up to 1000 files to be opened per process. An easy access for writing to this file is provided by the macro *STARLOG(X)*, including the automatic prepending of the *starId* in each line that identifies each class instance.
- User defined automatic configuration: the method *setPtet6Config()* evaluates the contents of the string array state *Ptet6ConfigDefaults*. The state contains value pairs: the first value is the state name of which the value should be replaced by a value from the global configuration data *Ptet6Config<T>*. The second value specifies the key and (optional) row and column index of the data (cf. Fig. 5.3).

5.4 Implementation of Simulation Models

The structure of the simulation scenarios implemented in this work is described in this section. One simple model, the so called bottleneck scenario, is depicted in Fig. 5.11; it servers as demonstration object to explain the architecture and functionality in the following. A more detailed discussion of the network aspects of this model can be found in Section 5.6.1. The scenario consists of the star *Ptet6Config*, which mainly implements an actor for the class *Ptet6Config* described in Section 5.3.1, and two different galaxies: the *coreNode* (labels S1 – S6) and the *clientNode* (labels R1 and R2).

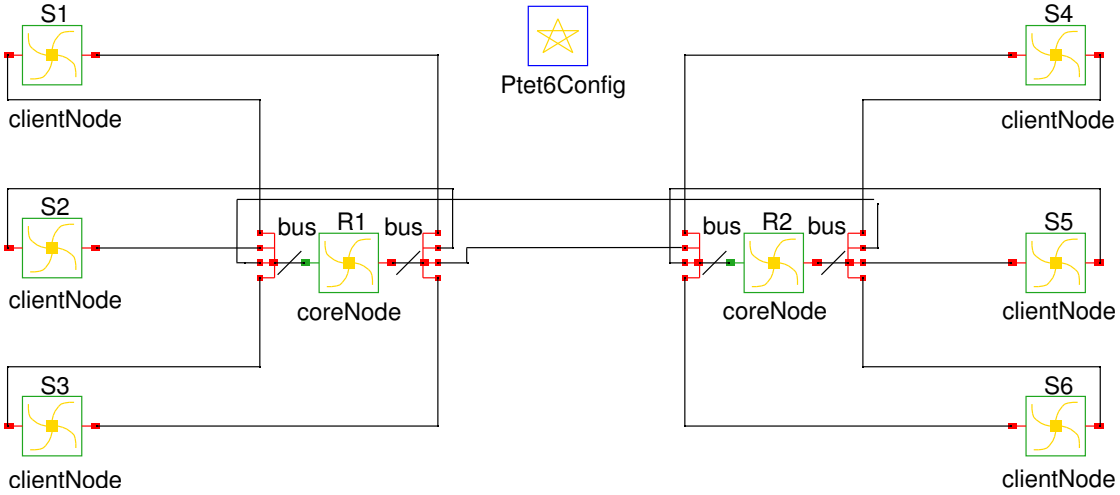


Figure 5.11: Universe *bottleneck*.

The core node implements a router with an output queueing system as displayed in Fig. 5.12. The output queueing system is implemented via the galaxy *routerPort* and Ptolemy's comfortable scaling facility: the Higher Order Function (HOF) star called *MapGr* reads the number of replacement blocks (for the connected block *routerPort*) from the width of the bus connecting the HOF star to the router, such that adding an output port to the router can be done by simply changing the value of the bus width marker.

The galaxy *routerPort* contains a queue, a server and a delay as shown in Fig. 5.13. The queue implements the First In First Out (FIFO) discipline and measures the probability density function of the queue arrival occupancy via a histogram and is therefore called *HistFifoQueue*. The Inter-Arrival Times (IAT) can be also measured by the queue. The so called *StatServer* measures the statistics of the Inter-Departure Times (IDT). The throughput is measured via a counting process which counts the number of bytes for a fixed time frame. The *Delay* star models the propagation delay between this port and the

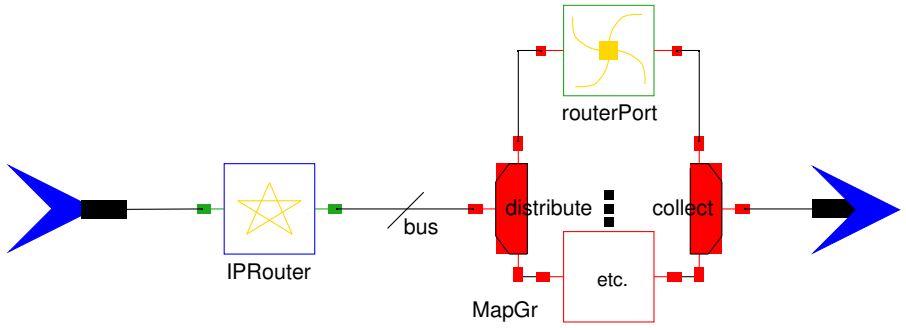


Figure 5.12: Galaxy *coreNode*.

next node. The output of the *StatServer* is fed back to the *demand* input of the queue. This implements the feedback that informs the queue that the server is free to serve the next packet.

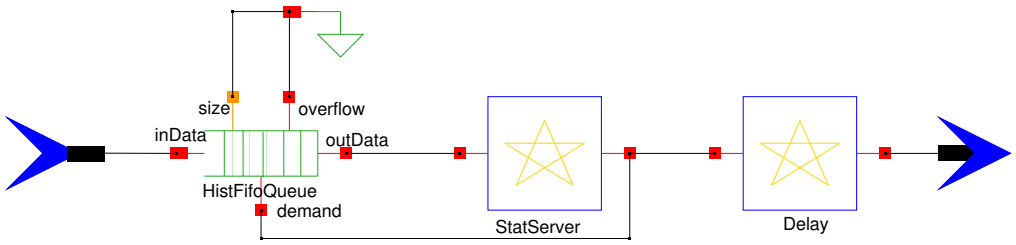


Figure 5.13: Galaxy *routerPort*.

A node equipped with network clients, the so called *clientNode*, is shown in Fig. 5.14. The difference between *coreNode* and *clientNode* is that the latter is equipped with the *client* galaxy and a HOF star *MapGr* connected to the router via *BusSplit* and *BusMerge* stars. The number of client galaxies can be controlled by the bus widths of the connection wires.

The *client* galaxy in Fig. 5.15 consists of the two stars *HTTPSource* and *TCPSocket*, which implement HTTP, TCP, and IP layer for the client. The HTTP source actually acts as client or server for a single connection, a state value determines the actual behaviour. The IP layer is also implemented in the star *TCPSocket*, since it requires only a marginal extension of the functionality but is more efficient as opposed to an implementation in a separate star, see Section 5.5.

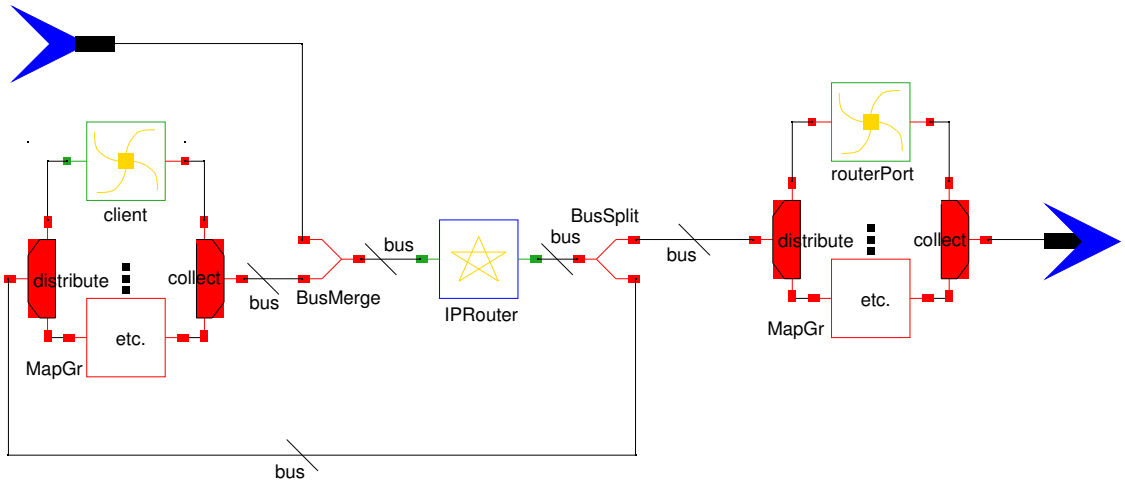


Figure 5.14: Galaxy *clientNode*.

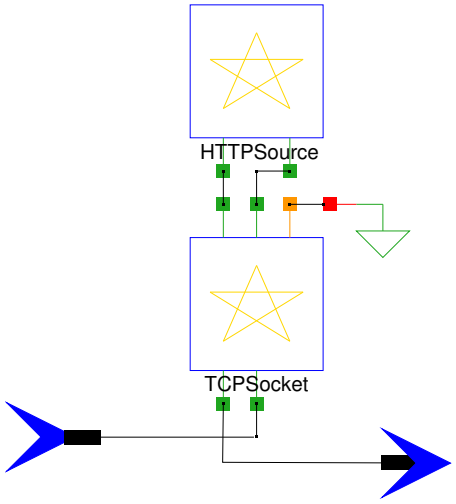


Figure 5.15: Galaxy *client*.

5.5 Optimisation of Simulation Speed and Memory Requirements

The simulation experiments with the node architecture described in Section 5.4 have shown that memory consumption and simulation time is a severe issue: a simulation with 24430 sources requires about 2660 MB of memory and is approx. 9400 times slower than real-time. A new architecture for the *clientNode* was developed which is presented in this section in order to overcome the abovementioned performance problems. The

clientNode (cf. Fig. 5.14) was replaced by the so called *clientServerNode* as displayed in Fig. 5.16. The galaxy webservers and the star *MultiHttpTcpClient* implement the functionality that was implemented before in the HOF star *MapGr* and the *client* galaxy. The star *HTTPSource* in the old design acts as client and server, depending on a state value. The functionality of server and client is implemented separately now. The server and queue stars are used as rate limiters for the input of the clients and webservers. The implementation of HTTP and TCP was completely moved into separate classes such that the star *MultiHttpTcpClient* can allocate a vector with an arbitrary amount of HTTP/TCP clients.

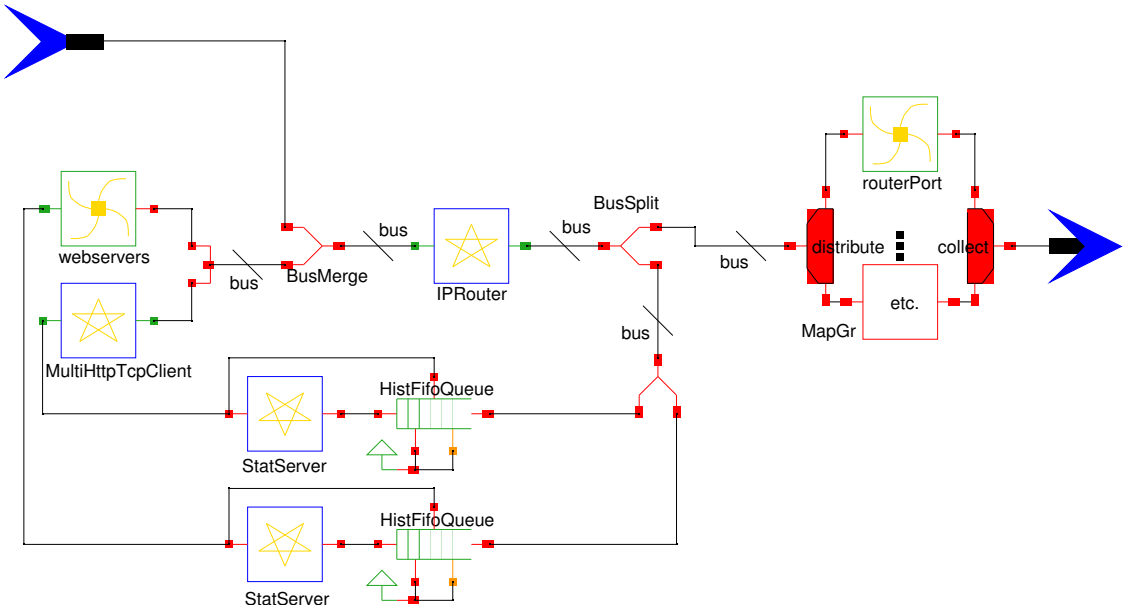


Figure 5.16: Galaxy *clientServerNode*.

The galaxy *webservers* is presented in Fig. 5.17. It contains a *ConnectionDistributor* that distributes the incoming IP packets with a modulo operation on the senders IP address to the *HTTPServer*. This type of load-balancing ensures that all packets of one connection are forwarded to the same server. However, it can not be guaranteed that the load is distributed uniformly over all webservers in the pool. The number of HTTP servers can be adjusted via the HOF star *MapGr*. The HTTP server is able to handle an arbitrary amount of TCP connections with a vector of TCP socket classes. The output bandwidth of the server pool is limited by the queue and the server. Each webserver has a limited capacity of 100 Mbit/s and the complete pool is connected via a 1 Gbit/s line to the router. A separate line with 1 Gbit/s connects all clients to the router.

The savings gained with the new design are summarised in Table 5.1. The values in the columns “Speed” contain the speed factor, the factor that the simulation is slower than

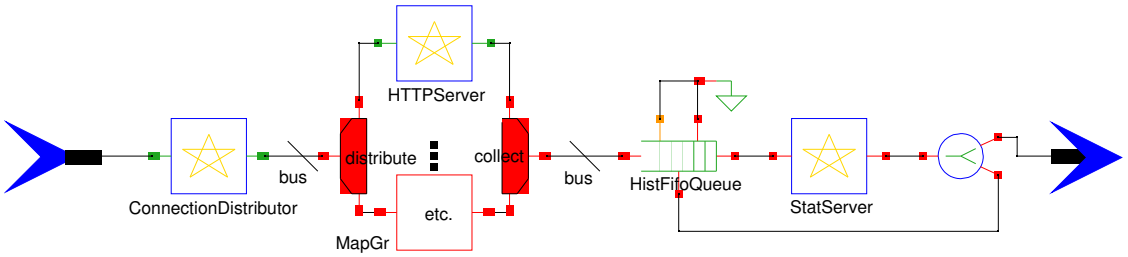


Figure 5.17: Galaxy *webservers*.

real time. A gain of up to 436 % in the required memory and up to 278 % in simulation speed could be gained with the new design. Furthermore, the new design allows simulations with a larger number of clients and thus a more realistic simulation of large networks with the same amount of available memory.

Table 5.1: Comparison of simulation speed and memory requirements of original and optimised architecture.

# Sources	RAM1	RAM2	Speed1	Speed2	RAM Gain	Speed Gain
682	73	33	181.5	101.4	221 %	179 %
2107	205	83	424.0	153.5	247 %	278 %
24430	2660	610	9400.0	4700	436 %	200 %

It can be concluded from this experience that it is advisable to avoid using the HOF stars for thousands of galaxies/stars in Ptolemy since this goes along with a large memory overhead. Furthermore, stars with a small functionality should be avoided in order to prevent generating too many events for the scheduler, especially if these events could be saved and replaced by a direct function call. The functionality should be rather implemented in C++ classes such that the star acts as a handler for several classes like different protocols or several instances of the same protocol.

A further increase in simulation speed can be gained by inspecting the implementation of the TCP timers. Three options are possible:

- one timer per connection, timer is active even when TCP connection is closed
- one timer for all connections handled by the same star instance
- one timer per connection, only active when TCP connection is open.

All three possibilities are associated with advantages and disadvantages. The first variant is the easiest for implementation but the overhead is very large, especially when the off-time is much larger than the on-time, as it is the case for the WWW user model. In this case the number of timer events while the connection is closed can reach (or even exceed) the same order of magnitude as the number of necessary timer events plus the number of events for sending and receiving packets. In other words: it is very inefficient when the off-time is large.

The second variant is also simple to implement but it could lead to significant errors when the packet loss rate is higher: all TCP connections that have encountered a time-out in the same time-slot (defined by the granularity of the TCP RTT timer, e.g. 100 ms used here) would start the retransmission at the same time. This could lead again to multiple losses and therefore to a behaviour that is completely different from reality, where the clients have uncorrelated timers. However, if the buffer capacity is large enough or if the network utilisation is low this option can be used to speed-up the simulation. The speed increase of the second method versus the first method amounts to a factor of 10 for 24430 clients.

The third variant is more complicated to implement since it is necessary to know the state of the connection when restarting a timer. However, this strategy does not introduce errors with respect to reality and the simulation speed is only reduced by approx. 10 % with respect to the second method.

The second method was used in all simulations since the third method was not stable when the simulations had to be scheduled. In most cases the loss probability was lower than 0.1 % so that it can be assumed that the error introduced by a common timer for all TCP connections is very small. The results of some simulations, where the result was unexpected, have been validated by a simulation with individual timers (e.g. high Hurst parameter values for high link utilisation in Sec. 6.4.3).

5.6 Network Simulation Models

Three simulation scenarios are used in this work: an advanced version of the well-known bottleneck scenario (Sec. 5.6.1), a so called parking-lot model with four main links and different RTTs and hop-counts for the flows (Sec. 5.6.2) and the B-WiN model, a large network model that is representative of an existing backbone (Sec. 5.6.3, see also the technical report [BSK01]). The three scenarios can be characterised as follows:

- The 6 flows in the bottleneck have three different round-trip times but the same hop-count and share all the same bi-directional link. The target traffic matrix is uniform, 11 Mbit/s per flow. The total throughput is 66 Mbit/s, the maximum link utilisation is 66 %.

- The 16 flows in the parking-lot scenario have different round-trip times, different hop-counts and do not share all the same link. The target traffic matrix is uniform, 8.5 Mbit/s per flow. The total throughput is 136 Mbit/s, the maximum link utilisation is 68 %.
- The 110 flows in the B-WiN have different round-trip times, hop-counts and different routes. The target traffic matrix is completely non-uniform, the values are ranging from 0.44 Mbit/s to 126.78 Mbit/s. The total throughput is 1.44 Gbit/s, the maximum link utilisation is 98.4 %.

The sketch shown in Fig. 5.18 shows other aspects of the same scenario shown already as schematic of the Ptolemy universe in Fig. 5.11. The representation with a sketch for the network scenarios is used in the following because it allows an easier overview and presents more information describing the network properties than the Ptolemy plots.

The connection of subnets to the core network is modelled as follows: every node is equipped with two internal links with a capacity of 1 Gbit/s and propagation delay of 0.1 ms to which the local clients and webservers are connected, respectively. Each webserver has a throughput limitation of 100 Mbit/s.

5.6.1 Bottleneck Scenario

All flows share the same link between two routers R1 and R2 in the bottleneck scenario shown in Fig. 5.18. Different link delays are configured so that the flows experience different minimum RTTs: the capacity of the bottleneck link (“R1<->R2”) was set to 50 Mbit/s and the propagation delay to 8 ms. These values are denoted as “$50/8$” in Fig. 5.18. The client nodes S1, S2 and S3 are connected via links of capacity 100 Mbit/s and propagation delay 1 ms, 5 ms and 10 ms, respectively. The installed flows are depicted as dashed lines. The target traffic matrix is set to 11 Mbit/s for each flow resulting in an average utilisation of 66 % of the bottleneck link in both directions. Each clientServerNode is equipped with two webserver modules.

5.6.2 Parking-Lot Scenario

The parking-lot scenario visualised in Fig. 5.19 has the advantage of being already complex enough to show the effect of different RTTs, hop counts and loss rates but still being simple enough to provide fast simulations and an easy interpretation of the results. All links between the source nodes S1 – S12 and the router nodes R1 – R5 are configured with a rate of 1 Gbit/s and a propagation delay of 1 ms so that there is no bottleneck there. The routers are connected via links with 50 Mbit/s. The links from R1 to R2 and

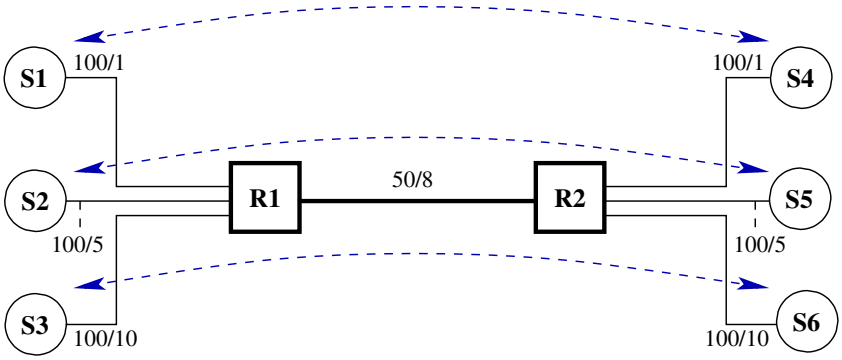


Figure 5.18: Topology of the bottleneck simulation model (annotation C/D: link capacity C in Mbit/s and propagation delay D in ms).

R4 to R5 have a propagation delay of 8 ms, the remaining two links are configured with a propagation delay of 24 ms. 2 webservers are allocated per clientServerNode.

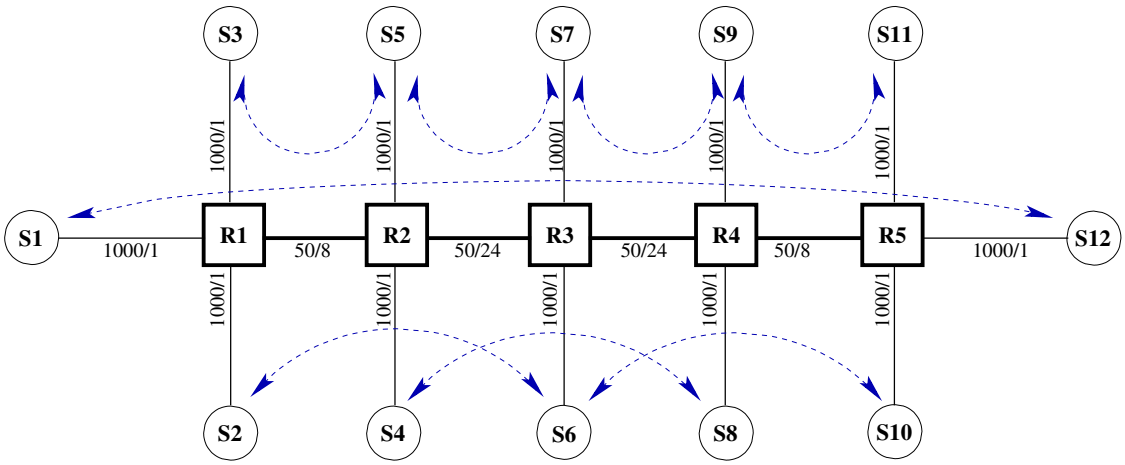


Figure 5.19: Topology of the parking lot simulation model (annotation C/D: link capacity C in Mbit/s and propagation delay D in ms).

The specified traffic matrix for the parking-lot scenario is again very simple: all flows should achieve a throughput of 8.5 Mbit/s which results in a maximum average link load of 68 % (between R2 and R4). Although this traffic matrix is simple, it is nevertheless not trivial to find the corresponding number of clients for all flows: due to TCP's sensitivity to different round-trip times and packet loss probability values, the six-hop connections can not achieve the same throughput as the shorter connections, so that relatively more sources are required for longer flows.

5.6.3 B-WiN Scenario

The B-WiN scenario is very complex (cf. Fig. 5.20) and the simulation speed is significantly lower than for the bottleneck and parking-lot scenario. Moreover, it is sometimes hard to interpret the results because it is not easy to get an overview over 110 competing flows in the meshed network. The B-WiN is a model (of an early state) of the network that connects all German universities and research institutes. A complete traffic matrix was measured by the provider “Deutsches ForschungsNetz” (DFN).

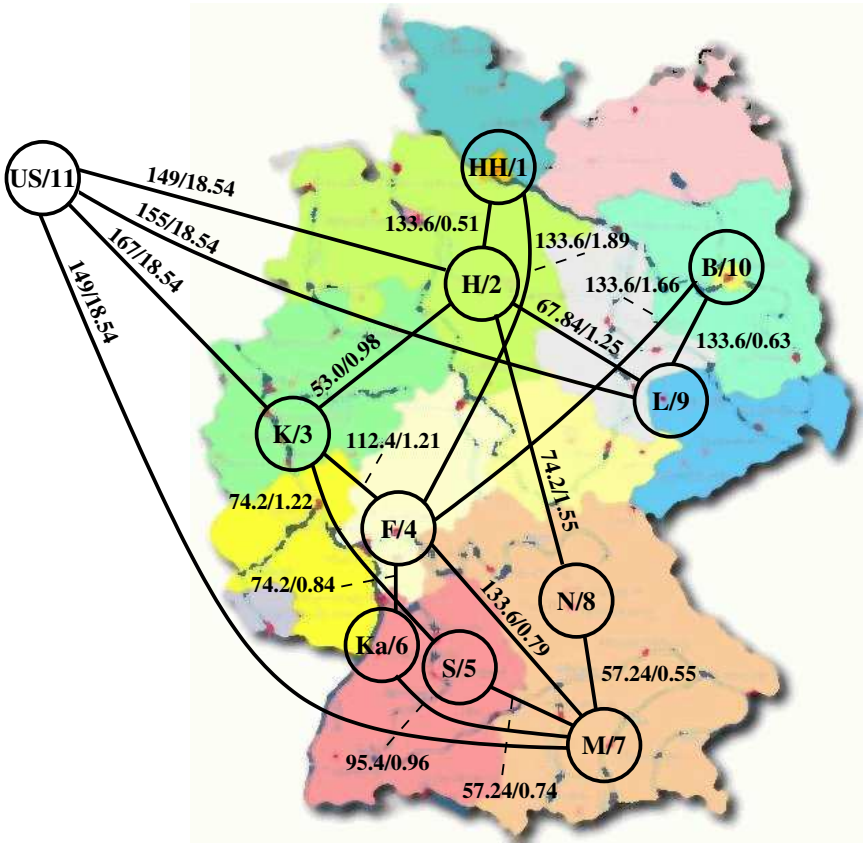


Figure 5.20: Topology of the B-WiN simulation model (annotation C/D: link capacity C in Mbit/s and propagation delay D in ms).

The results of simulations with this model can be regarded as a test case to evaluate real network behaviour. The model consists of 11 nodes connected via 18 bi-directional links with a total capacity of 3.9 Gbit/s and a total measured throughput of 2.17 Gbit/s (1.44 Gbit/s routed over several links). The link capacities range from 53 Mbit/s to 167 Mbit/s and the link delays from 0.5 ms to 18.5 ms. 20 webservers are allocated at node “US”, 10 web servers at node “F” and 5 web servers at all remaining nodes.

Flows from all nodes to their respective neighbour nodes exist, which results in a total of 110 flows. The routing was optimised in order to keep the traffic load below 70 % on all links except for those connecting the USA (node “US/11”) with the German part of the network. These links were allowed to have a load over 90 %; the maximum link load was measured on link “US->K” with $\rho = 98.4\,\%$. The provider realised that the utilisation could not be kept below 70 % on those links: whenever the capacity was increased by the provider, an increase of the traffic volume was observed in the next months, driving the utilisation soon again close to the capacity of the links.

The measured traffic matrix (average throughput per flow over one month in Mbit/s) is depicted in Fig. 5.21. The range of values is very large, the minimum value is 0.44 Mbit/s and the maximum value is 126.78 Mbit/s. Further, the flows with the largest throughput are the flows from node “US” to all other nodes and the flows originating at node “F”. The majority of the traffic are downloads from the USA and from other servers located outside the B-WiN, reachable via the cross-interchange point in Frankfurt (node “F”).

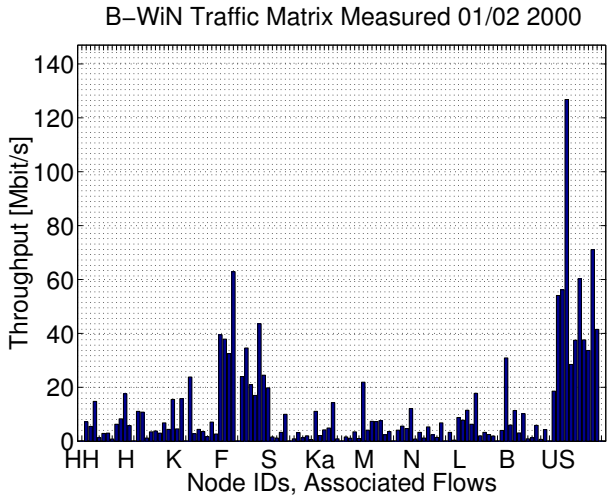


Figure 5.21: Measured traffic matrix of the B-WiN, Jan/Feb 2000.

Chapter 6

Simulation Results

The simulation results are collected and discussed in this chapter. All common simulation parameters are listed in Sec. 6.1. The performance evaluation of the algorithm for the allocation of clients used for matching the average traffic load is presented in Sec. 6.2. The convergence speed of the iterations used to increase the accuracy is discussed in Sec. 6.2.4. The assessment of the matching of higher order moments of the traffic statistics to measurements, the coefficient of variation and the Hurst parameter, is presented in Sec. 6.4. One special case of high Hurst parameter values at high link loads is discussed in Sec. 6.4.3.

The evaluation of reducing the simulation complexity by reducing the number of clients is presented in Sec. 6.5. The simulation study shows the change in the measured values for the average traffic load, the loss probability, the coefficient of variation, the Hurst parameter and the end-to-end delays in dependence on the number of clients. The potential of the optimisation results are discussed and the conclusions are presented.

6.1 Common Simulation Parameters

The common simulation parameters of all simulations are listed in Table 6.1; deviations from the parameter values in the table are mentioned explicitly in the respective section, if any. The simulation time was set to 700 s and the measurements were re-initialised at time $t = 100$ s to eliminate the measurements of the transition phase, before the simulation reaches a more or less steady state. Therefore, the effective simulation time was 10 minutes (700 s minus 100 s initialisation phase).

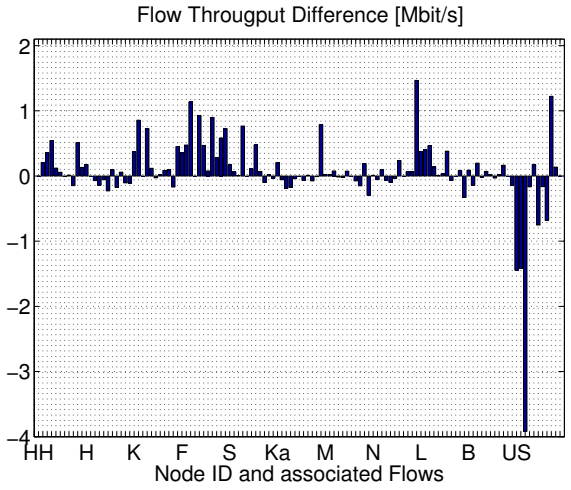
Table 6.1: Common default parameters of all simulations.

HTTP	Revision	1.1
	Persistent connections	yes
	Parallel connections	no
	Request packet size distribution	const.
	Request packet size, average	360 Bytes
	Object number distribution	geometric
	Object number, average	6
	Objects size distribution	TPT
	Object size, average	10 KB
	Object size, shape α	1.5
	Object size, truncation level T	20
	Object size, min	10 Bytes
	Object size, max	100 MB
	Off-time distribution	neg. exponential
	Off-time, average	40 s
TCP	Revision	NewReno
	MSS	1460 Bytes
	CWND_MAX	65535 Bytes
	Slow-start threshold init	65535 Bytes
	Receive window	65535 Bytes
	Slow timer granularity	100 ms
IP	MTU	1500 Bytes
	Router buffer	2500 Packets
General	Sample interval, counting process	2.5 ms
	Simulation time	700 s
	Reset time for measurements	100 s
	Maximum IP Throughput per connection	10 Mbit/s
	Maximum IP Throughput all clients per node	1 Gbit/s
	Maximum IP Throughput all webservers per node	1 Gbit/s

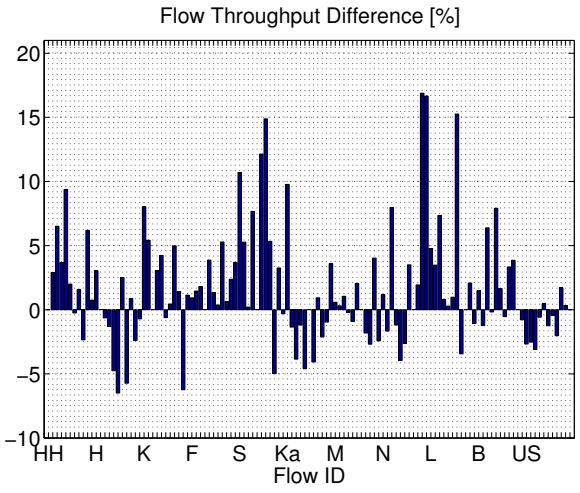
6.2 Generating Prescribed Traffic Intensities with HTTP/TCP Sources

The performance of the algorithm for allocation of clients introduced in Sec. 4.1 is discussed in this section. The deviation of the traffic from the target traffic matrix for the B-WiN scenario is shown in Fig. 6.1. The absolute deviation $\Delta = tp_{ij,given} - tp_{ij,measured}$

and the relative deviation $\delta = \left(tp_{ij,given} - tp_{ij,measured}\right)/tp_{ij,given}$ to the target traffic matrix $tp_{ij,given}$ (cf. Fig. 5.21), are shown in Fig. 6.1 (a) and (b), respectively. The absolute deviation range is $[-3.92, 1.46]$ Mbit/s and the relative deviation range is $[-6.49, 16.89]$ %.



(a)



(b)

Figure 6.1: (a) Absolute deviation Δ and (b) relative deviation δ from the given traffic matrix, B-WiN, $v = 60$ KB, $t_{off} = 40$ s and $\alpha = 1.5$.

Comparing Fig. 6.1 (a) and (b) it becomes obvious that the flows with a large absolute error (node “US”) have a small relative error and the flows with a large relative error have a small absolute error. This is induced by the large range of the values in the target traffic

matrix: it is very likely that a flow with a very small target throughput of e.g. 0.4 Mbit/s, where the traffic will be generated by few connections, has a large relative error. A time-out of a single connection could provoke already a large change in the throughput of this flow. However, this error has a very small contribution to the overall network traffic and can thus be neglected. If the target throughput of a flow is very large on the other hand, e.g. 126.8 Mbit/s, it is more likely to achieve a large absolute error. The relative error in this flow is rather small since many connections contribute to the throughput and it is normalised by a large value.

The absolute and relative deviation are evidently not very good parameters to judge the quality of the traffic match. Therefore, three error measures are introduced:

- the sum of the differences normalised to the sum of the throughput matrix values Δ_{sum} [%]:
$$\Delta_{sum} = 100 \cdot \sum_{i,j} \left(tp_{ij,sim} - tp_{ij,given} \right) / \sum_{i,j} tp_{ij,given}, \tag{6.1}$$
- the sum of the absolute differences normalised to the sum of the throughput matrix values Δ_{sum} [%]:
$$\Delta_{sum,abs} = 100 \cdot \sum_{i,j} \left| tp_{ij,sim} - tp_{ij,given} \right| / \sum_{i,j} tp_{ij,given}, \tag{6.2}$$
- and the maximum absolute error in percent of the given throughput $\Delta_{max,abs}$:
$$\Delta_{max,abs} = \max_{i,j} \left\{ tp_{ij,sim} - tp_{ij,given} \right\} / tp_{ij,given}. \tag{6.3}$$

The measure Δ_{sum} is an indication whether the total number of clients matches the total given throughput. Two cases must be considered, if Δ_{sum} is close to zero:

- the deviation between prescribed and measured throughput matrix could be close to zero or
- positive and negative errors add up to a value close to zero.

The total traffic volume and therefore the total number of clients is in both cases close to the optimal value, but the distribution of the clients over the flows might not be close to the optimum in the second case.

The measure $\Delta_{sum,abs}$ characterises the overall deviation between prescribed and simulated throughput matrix. The parameter $\Delta_{max,abs}$ shows the largest absolute error in the whole network. These two error measures $\Delta_{sum,abs}$ and $\Delta_{max,abs}$ have the potential to fully characterise the quality of the solution.

The performance evaluation is carried out for all combinations of the parameters download volume $v = \{60, 100, 150, 200\}$ KB and off-time $t_{off} = \{10, 20, 40\}$ s for several seeds of the random number generator. The parameter pair $v = 60$ KB and $t_{off} = 40$ s is of major importance here, since it matches approximately a measured user model [CL99]. However, it can be expected that the average download volume v is increasing over time, according to the increase of the Internet access bandwidth of the users. It can be expected that the error is larger for a large download volume: TCP operates more in congestion-avoidance phase than in slow-start phase in this case, but it is still not likely that it reaches steady state. Therefore, a larger deviation from the estimated throughput can be expected.

The influence of the off-time t_{off} on the accuracy is two-fold: the sum of on- and off-time in Eq. (4.1) is dominated by the off-time and thus larger values of the off-time should lead to a smaller impact of estimation errors of the on-time. On the other hand, smaller values of the off-time go hand in hand with more on-off cycles that can be measured in the same simulation time. Therefore, the simulation converges faster for small off-time values. However, it is not clear which of the two effects is dominating and under which condition.

It is assumed in the following that the average values of simulation results with different seeds follow approximately a normal distribution. The assumption is reasonable since the simulation results from different runs (with different seeds) are uncorrelated. Further, the difference of independent average values is likely to follow a normal distribution.

The distributions of the resulting measurements are visualised with boxplots (cf. Fig. 6.2): the box contains 50 % of the measurement values, the average is depicted with a '+' symbol and the horizontal line in the box represents the median value. The number of samples is annotated to the right of the box. The *confidence intervals of the average value* are represented by dashed horizontal lines following the equation

$$ci_{avg,\pm} = m \pm t_{95} \cdot \frac{s}{\sqrt{N}}, \tag{6.4}$$

where t_{95} is the student-t distribution value for N degrees of freedom, N is the number of samples, m represents the estimated average value

$$m = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i, \tag{6.5}$$

and s is the unbiased estimate of the standard deviation

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - m)^2}. \tag{6.6}$$

The upper and lower horizontal lines ("whiskers"), which are connected with vertical lines to the box, represent the 95 % *confidence interval for the value range* of the measurement values (width of the distribution). That is, 95 % of all measured values are expected to fall in this range:

$$ci_{\pm} = m \pm t_{95} \cdot s. \tag{6.7}$$

The minimum and maximum values are marked with a '+' if the corresponding value is outside the confidence interval for the value range $ci_{\pm}$.

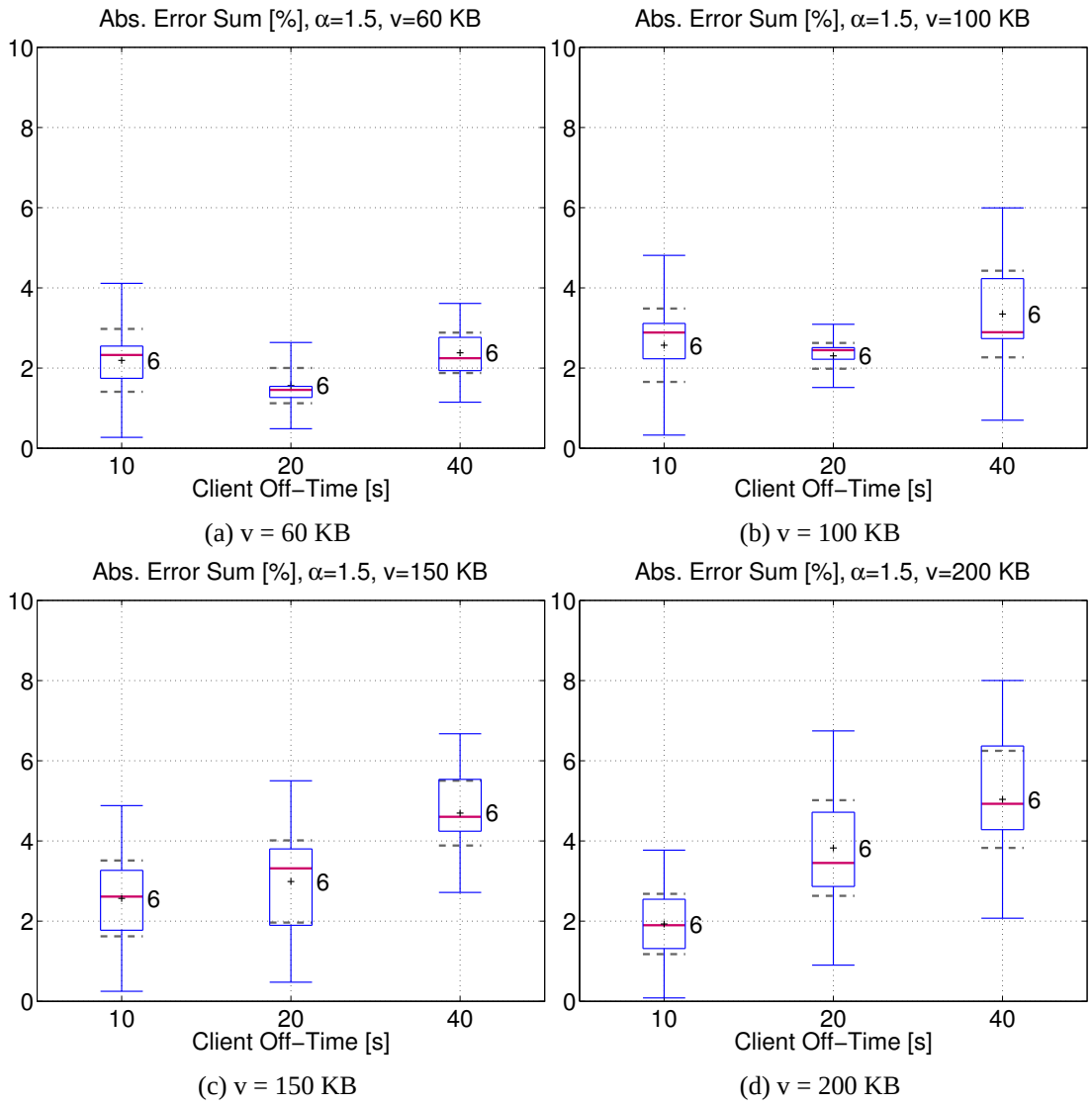
All parameters are kept constant when the evaluation is performed for different seeds. Also the allocation of clients is the same for all seeds with the same parameter set. Therefore, the performance of the client allocation algorithm (described in Sec. 4.1) should be judged by the average and median value. The width of the confidence interval is an indication of the quality of the random number generator and of the convergence of the simulation, but it is not an evidence for the accuracy of the estimated number of clients. The average and median value are an indication of a potential bias caused by an over- or underestimated number of clients and are thus meaningful quality measures.

The tolerable error bounds need to be discussed: very tight error bounds are always welcome. However, a simulation study for performance evaluation of design alternatives for a network provider would take some months to be completed and the average throughput changes every month. Even if the simulations for the final decision in the study would take the latest data, e.g. the average throughput matrix of the previous month, it can be expected that the current month's throughput differs by some percent and therefore the simulation is always behind the times. Reasonable error bounds for the considered problem under the circumstances described above are $\Delta_{sum,abs} \leq 5$ % and $\Delta_{max,abs} \leq \pm 10$ % for the average value of all seeds.

6.2.1 Bottleneck Scenario

The boxplots of the absolute overall error measure $\Delta_{sum,abs}$ (cf. Eq. (6.2)) are visualised in Fig. 6.2. The average absolute error is in all cases below 5 % and in most cases even below 3 % indicating that the bias of the estimated number of clients for all flows is very small in the bottleneck scenario. The smallest error occurs for $v = 60$ KB in Fig. 6.2 (a); the error increases with larger download volumes (cf. Fig. 6.2 (b)-(d)) as expected.

The two oppositional effects determining the accuracy depending on the off-time – the larger number of samples for small off-times on one hand and the higher convergence speed for large off-times on the other hand – play an important role for $v = \{150, 200\}$ KB (Fig. 6.2 (c) and (d)): the effect of the convergence speed dominates the accuracy and therefore the error is larger for large values of t_{off}. The two effects are balanced for small download volumes $v = \{60, 100\}$ KB, no clear tendency is visible. The average error is also higher for $t_{off} = 40$ s and $v = \{150, 200\}$ KB than in the other



(a) v = 60 KB

(b) v = 100 KB

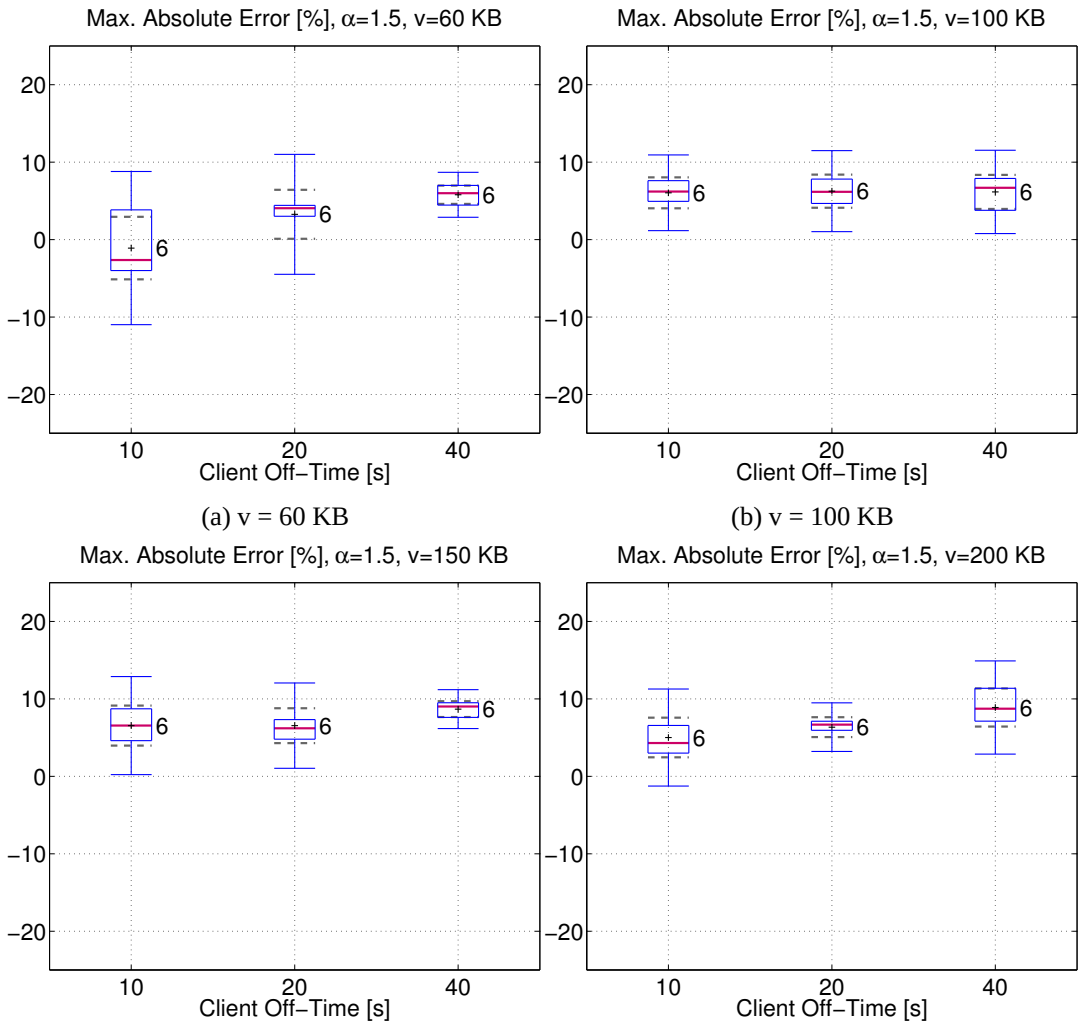
(c) v = 150 KB

(d) v = 200 KB

Figure 6.2: Sum of absolute errors in percent of throughput sum, Bottleneck, $\alpha = 1.5$.

cases, which seems to indicate a bias in the estimation leading to a slightly larger error for large off-time values and large download values.

The boxplots of the maximum absolute error $\Delta_{max,abs}$ (cf. Eq. (6.3)) are depicted in Fig. 6.3. The average throughput of a single connection is in most cases slightly underestimated – most error values are positive values with the exception of $v = 60$ KB and $t_{off} = \{10, 20\}$. However, the average of the maximum error is in all cases smaller than ± 10 % and therefore it can be concluded that the accuracy requirements are met by the estimated number of connections for the bottleneck scenario.



(a) v = 60 KB

(b) v = 100 KB

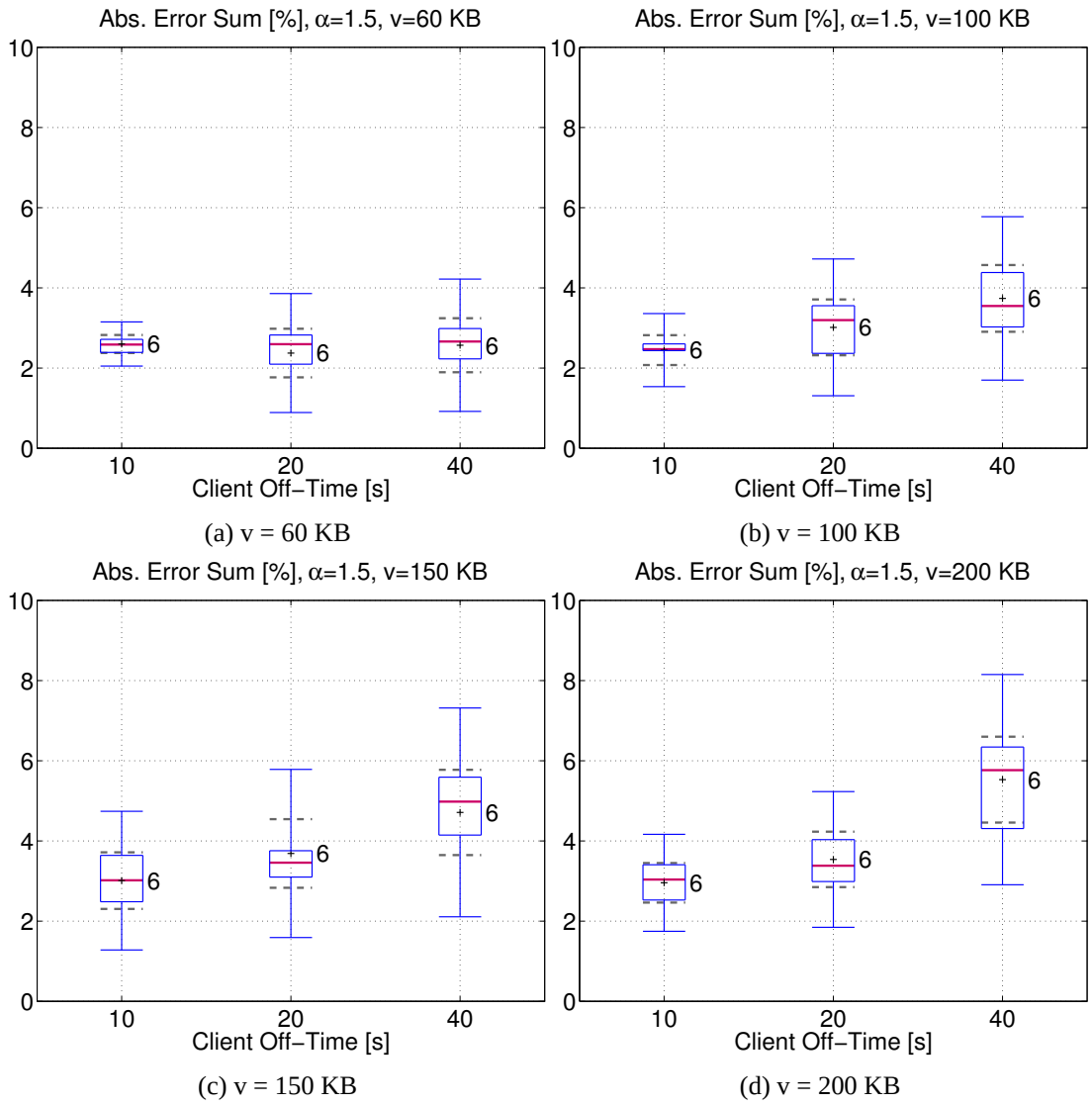
(c) v = 150 KB

(d) v = 200 KB

Figure 6.3: Maximum deviation from prescribed throughput matrix, Bottleneck, $\alpha = 1.5$.

6.2.2 Parking-Lot Scenario

The quality of the traffic match is very similar for the parking-lot scenario shown in Fig. 6.4 as compared to the bottleneck scenario. The average of $\Delta_{sum,abs}$ is approximately constant for $v = 60$ KB, but the error measure increases for larger values of v as well as for increasing values of t_{off} in qualitatively the same manner as already discussed for the bottleneck. Also the absolute values of the average $\Delta_{sum,abs}$ are very similar, there is only one exception: the average value of $\Delta_{sum,abs}$ reaches 6 % for $v = 200$ KB and $t_{off} = 40$ s and therefore exceeds the error threshold $\Delta_{sum,abs} \leq 5$ %.

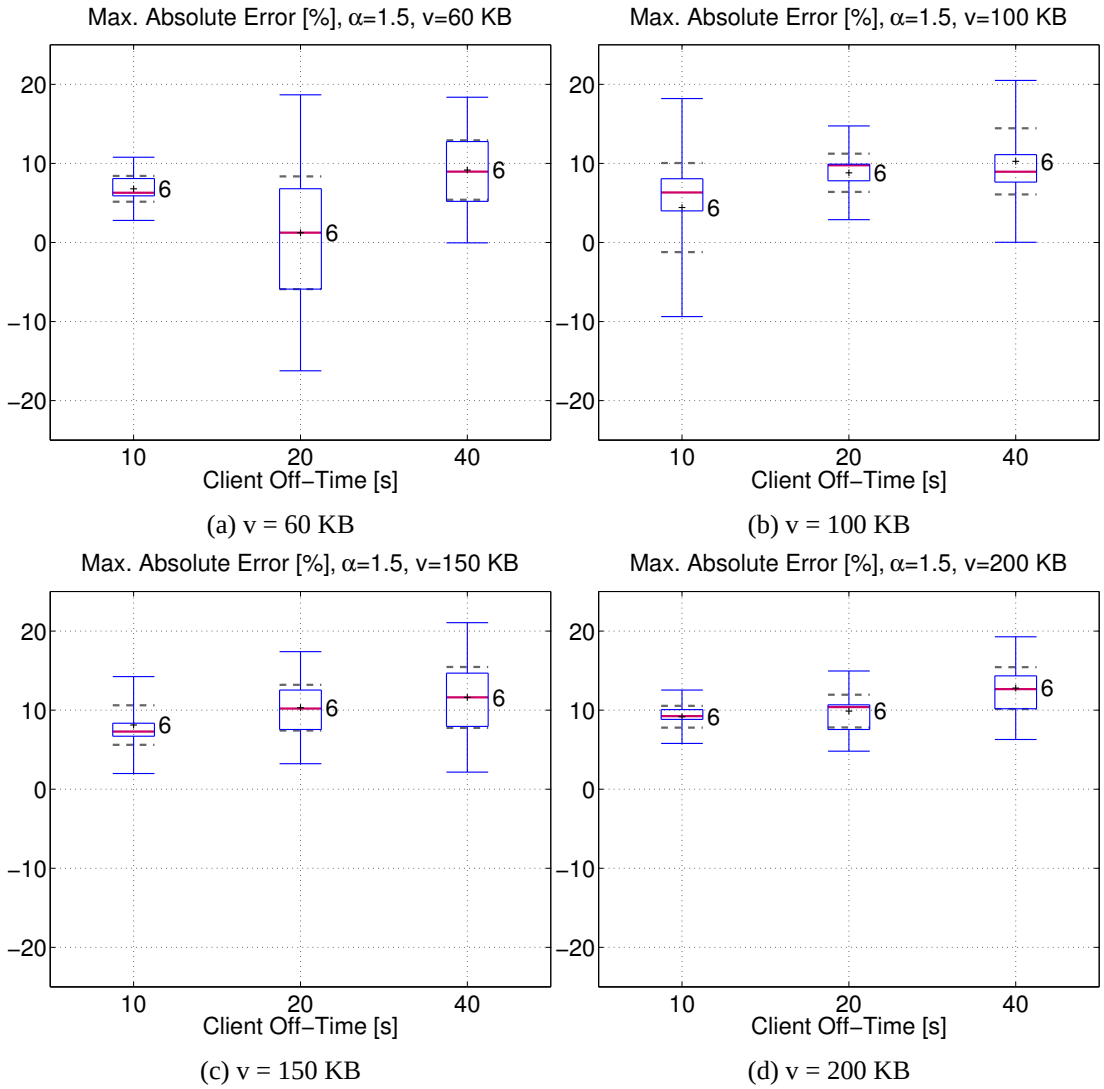


(a) v = 60 KB (b) v = 100 KB

(c) v = 150 KB (d) v = 200 KB

Figure 6.4: Sum of absolute errors in percent of throughput sum, Parking-Lot, $\alpha = 1.5$.

The average of the maximum absolute error of the traffic match in Fig. 6.5 is in general larger for the parking-lot as for the bottleneck. Many values are very close to the 10 % threshold and the threshold is exceeded in two cases ($v = \{150, 200\}$ KB and $t_{off} = 40$ s). The average of the maximum error is always positive as in most cases the bottleneck, which is indicating that the average throughput per connections is slightly under-estimated and thus too many clients are allocated to produce the prescribed throughput.



(a) v = 60 KB

(b) v = 100 KB

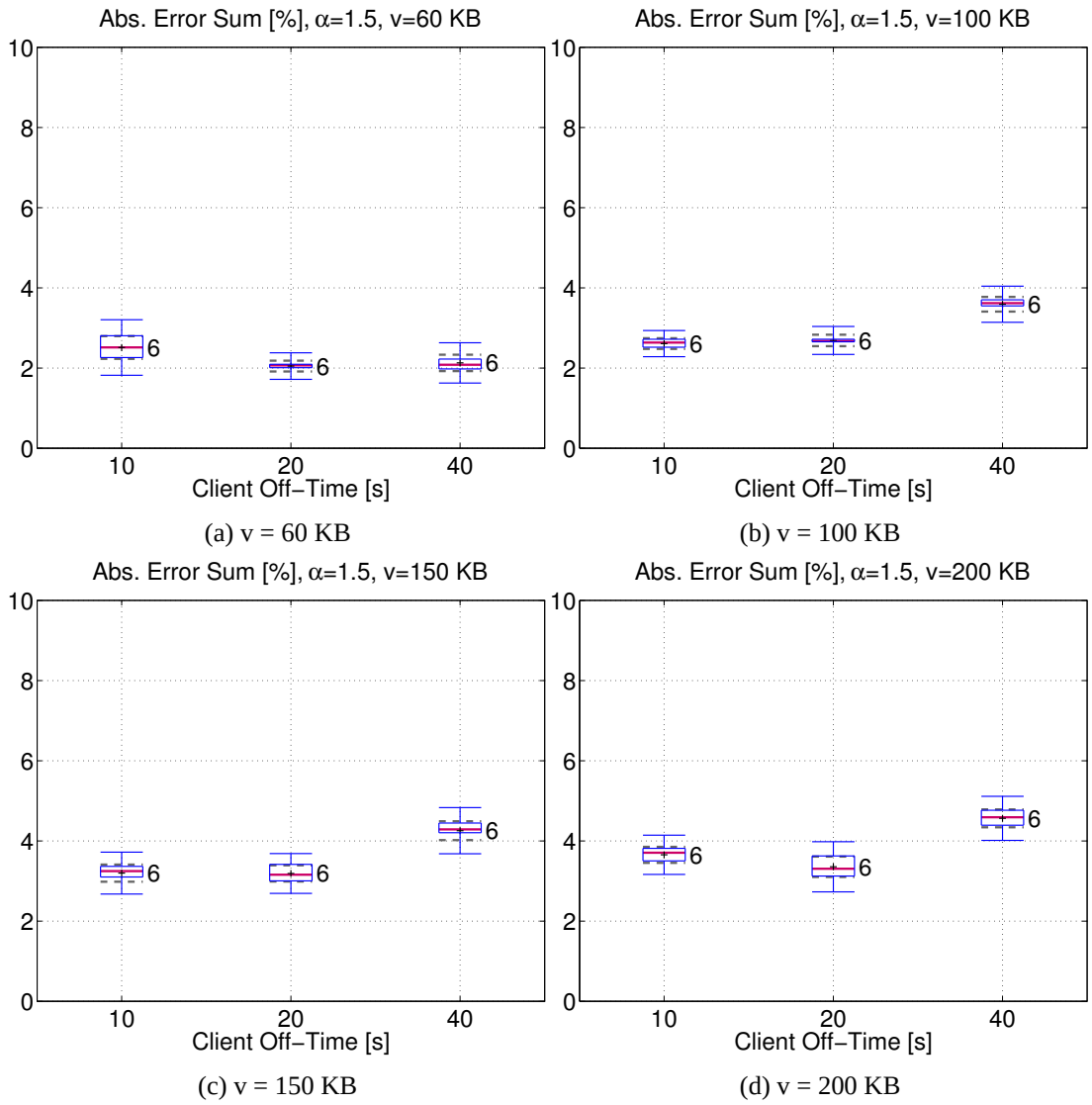
(c) v = 150 KB

(d) v = 200 KB

Figure 6.5: Maximum deviation from prescribed throughput matrix, Parking-Lot, $\alpha = 1.5$.

6.2.3 B-WiN Scenario

The results from the B-WiN scenario simulations shown in Fig. 6.6 reveal smaller average values for $\Delta_{sum,abs}$ as compared to the results of bottleneck and parking-lot scenario. Also the confidence interval for the average and the box width are smaller and in all but one case even the confidence interval for the value range does not exceed 5 %. This indicates that the overall traffic match is significantly better than for the bottleneck and parking-lot scenario.



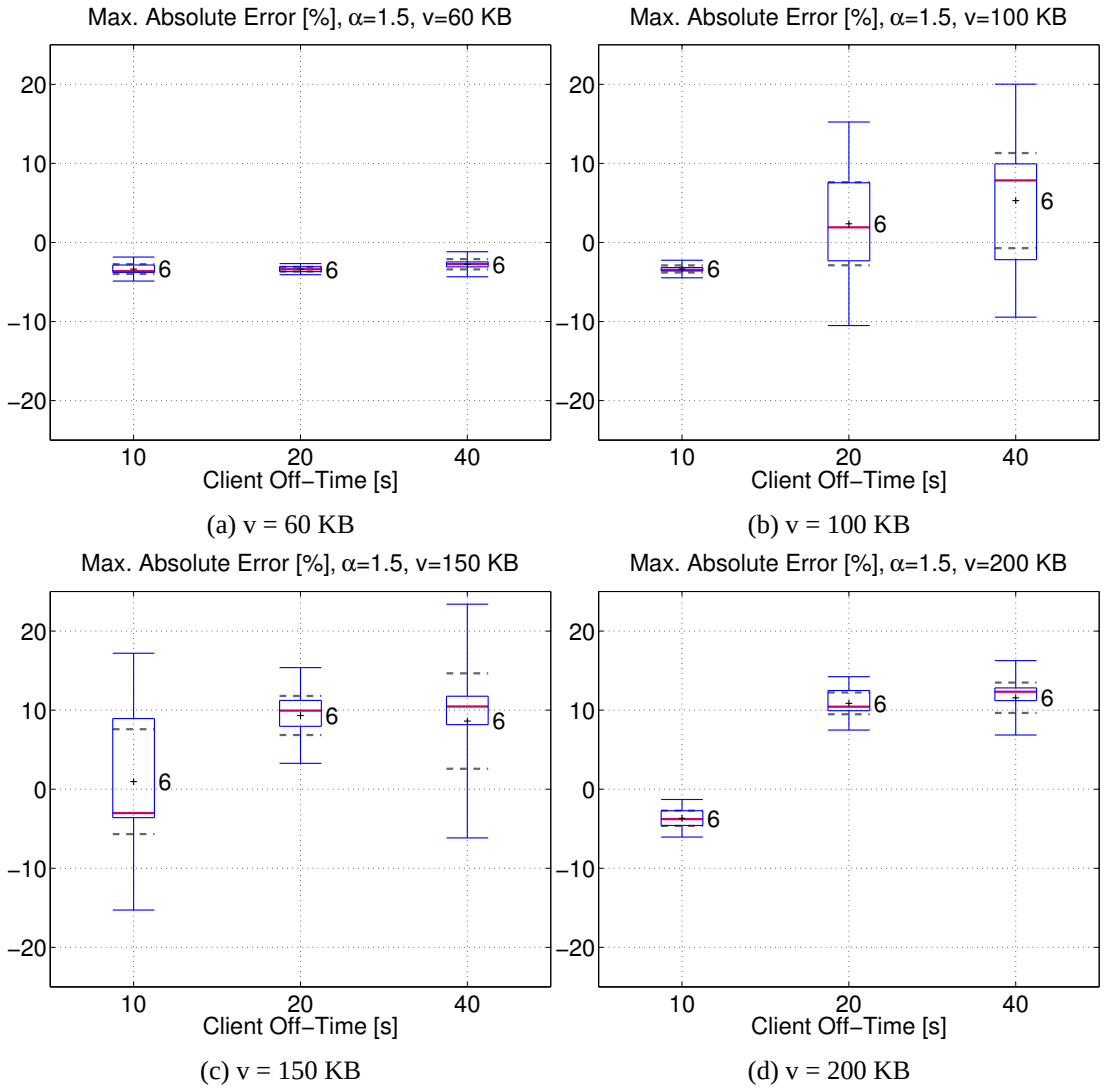
(a) v = 60 KB (b) v = 100 KB

(c) v = 150 KB (d) v = 200 KB

Figure 6.6: Sum of absolute errors in percent of throughput sum, B-WiN, $\alpha = 1.5$.

The average of the error shown in Fig. 6.7 is also lower than in the parking-lot scenario (cf. Fig 6.5). The average value is very small and negative for $v = 60$ KB indicating a small over-estimation of the average throughput in this case. Also the confidence intervals are very small in this case. The average of $\Delta_{max,abs}$ and the value range increases for larger download volumes v. The average exceeds the threshold of 10 % in three cases for $t_{off} = 40$ s, $v = \{150, 200\}$ KB and for $t_{off} = 20$ s and $v = 200$ KB.

The error values are in most cases smaller than for the bottleneck and parking-lot scenario. The opposite was expected, since the complexity of the B-WiN scenario is sig-



(a) v = 60 KB

(b) v = 100 KB

(c) v = 150 KB

(d) v = 200 KB

Figure 6.7: Maximum deviation from prescribed throughput matrix, B-WiN, $\alpha = 1.5$.

nificantly higher than those of the two other scenarios. The reason for this result is that the relatively large relative errors of the flows with a small target throughput have only small influence on the error measures $\Delta_{sum,abs}$ and $\Delta_{max,abs}$. Further, the number of flows is very large in the B-WiN such that the error of a single flow has only a small impact on the overall error.

Another reason for this behaviour could be that the total number of samples drawn from the TPT distribution is much smaller for bottleneck and parking-lot scenario as compared to the B-WiN, as discussed in Sec. 2.4. Therefore, the reason for the unexpected result

could be that the random numbers for the file sizes did not reach the same convergence state for the different scenarios.

6.2.4 Convergence of Iterations

Iterations can be used according to the algorithm described in Section 4.1 in order to increase the accuracy of matching the prescribed traffic matrix. The performance of the iteration – in terms of convergence to the specified values – is shown in this section.

The accuracy after several iterations is limited by the variability of the results of each simulation, as illustrated in the previous sections: the resulting deviation from the target throughput differs to some extend for different seeds but for the same allocation of clients (the height of the confidence intervals). Therefore, if a simulation in iteration step k results in a smaller throughput as the target value for one flow, the update for next iteration $k+1$ will increase the number of clients for this flow. This reaction is wrong, if the smaller throughput was not a result of an error in the estimation of the required number of clients, but caused by the large variability of the results. The result is that the error is oscillating after a certain number of iterations around zero. The oscillation amplitude is determined by the variability of the results with the same seed. The weighting constant c in Eq. (4.20) smoothes the oscillation at the expense of slower convergence.

The convergence behaviour is evaluated for three different settings:

- The parameters responsible for the largest errors in Sec. 6.2 (download volume $v = 200$ KB and $t_{off} = 40$ s) are used to show how far the error can be reduced with iterations. The dependency of the convergence on the shape parameter α, the weighting constant c and the simulation time is shown in Fig. 6.8.
- The convergence behaviour for realistic parameter settings $v = 60$ KB, $t_{off} = 40$ s and two different values for the buffer capacity is depicted in Fig. 6.9.
- One case with a “very bad starting situation” is considered as the worst case: a small buffer capacity of only 20 packets and an off-time of $t_{off} = 20$ s is used for the client allocation algorithm but the clients were configured for an off-time of $t_{off} = 40$ s. This setting is used to show the convergence behaviour when the initial client allocation is far away from the optimum (see Fig. 6.10). This case is selected to visualise the convergence in a case, where all estimations match very badly, much worse than ever experienced in all simulation studies performed by the author.

The convergence is visualised with four plots where the abscissa represents the step count k of the iteration. The total number of clients (a), the total throughput difference

Δ_{sum} (b), the total absolute throughput difference $\Delta_{sum,abs}$ (c) and the maximum absolute error $\Delta_{max,abs}$ (d) are shown in Fig. 6.8.

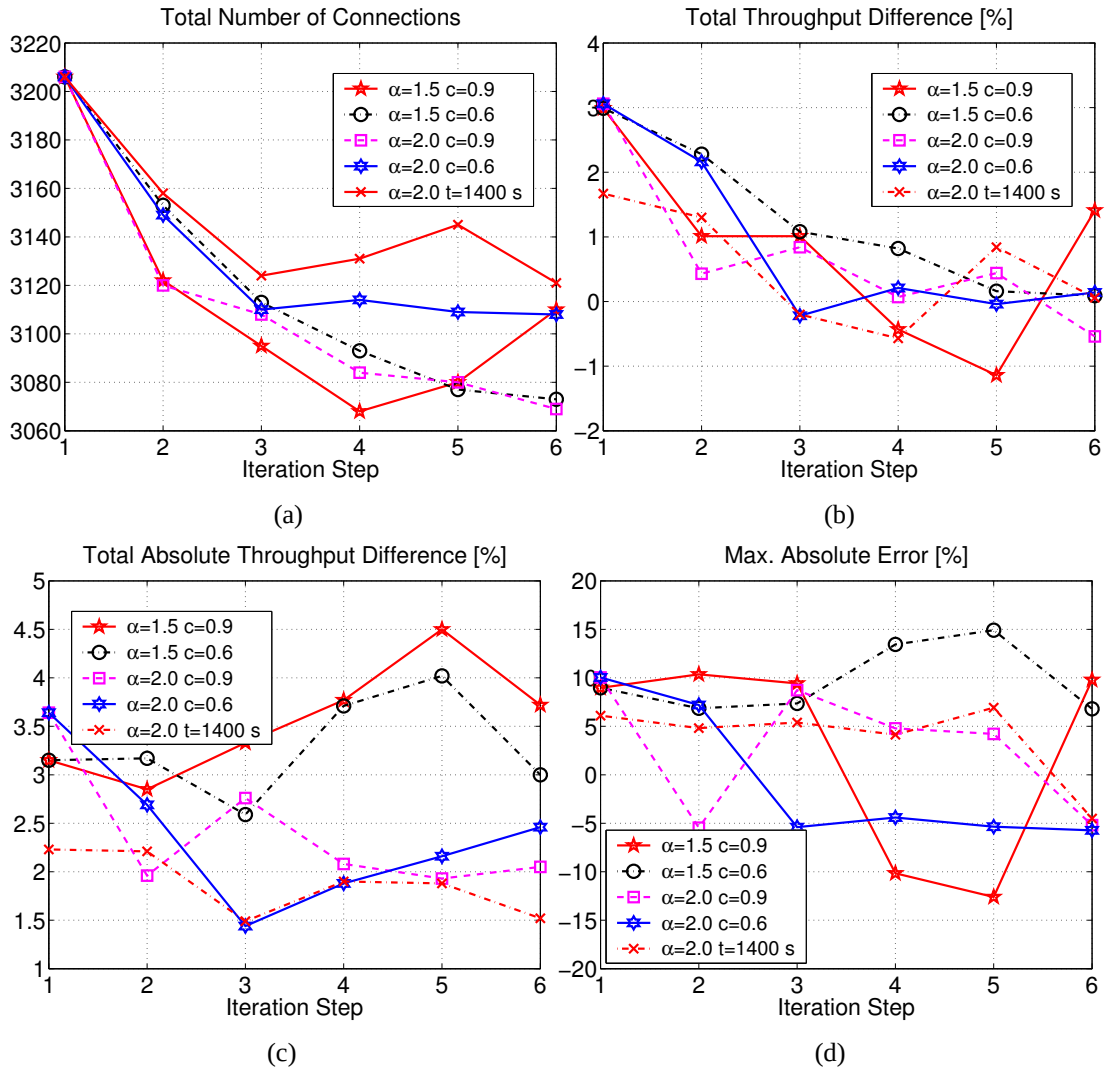


Figure 6.8: Convergence of iterations, parking-lot, buffer capacity $B = 200$ packets, $v = 200$ KB, $t_{off} = 40$ s.

The total throughput difference Δ_{sum} is in all cases positive for the first step of the iteration, declines in the following steps and oscillates around zero (cf. Fig. 6.8 (b)). The oscillation amplitude depends on the shape parameter α: the traffic is only short-range dependent for $\alpha = 2$ and long-range dependent for $\alpha = 1.5$. The smaller the value of α, the more oscillations can be expected for a constant simulation time (cf. Sec. 2.4, [CL97]). The curves with $c = 0.9$ are converging faster but with a stronger tendency to

oscillations as compared to the curves with $c = 0.6$. The iterations with a simulation time of 1400 s (twice the amount as the other simulations, performed with $c = 0.9$) indicates that a longer simulation time results in more stable values, as expected.

The behaviour of the absolute error $\Delta_{sum,abs}$ in Fig. 6.8 (c) displays that the two curves with $\alpha = 1.5$ are so instable that the oscillation leads to states with a larger error than the initial state. The absolute error value is still tolerable but the iterations are of no use in this case. The iterations can improve the results for $\alpha = 2.0$ and the long simulation with $c = 0.9$. The simulations with $c = 0.6$ show again the best convergence behaviour. The maximum absolute error $\Delta_{max,abs}$ depicted in Fig. 6.8 (d) gives one more evidence that the iteration converges only for $\alpha = 2$ to smaller error values and that $c = 0.6$ and long simulation times can avoid turbulent oscillations.

The evaluation with realistic parameter settings ($v = 60$ KB, $t_{off} = 40$ s) and for a buffer capacity of $B = 200$ packets, depicted in Fig. 6.9, reveal that the oscillations have a significantly smaller amplitude than for $v = 200$ KB in Fig. 6.8. The simulations with $c = 0.6$ also converge slower and with less pronounced oscillations. The simulations with a smaller buffer capacity of $B = 20$ packets experience a packet loss probability of approx. 2.3 %. The packet loss probability is not estimated and taken into account by the algorithm for allocation of clients (see Sec. 6.2). Nevertheless, already the second iteration step yields satisfying results.

The convergence behaviour for the “bad starting point” is shown in Fig. 6.10. The simulations with correct starting point and buffer capacity $B = 20$ packets shown already in Fig. 6.9 are plotted here again as reference. The curve denoted with ’*’ in the legend corresponds to the artificially bad starting point.

The curves with bad starting point experience an initial error of approx. 50 % as could be expected by specifying an off-time of 20 s instead of 40 s. However, the iterations converge fast towards an error of zero and the simulation reaches a region with tolerable error values in the third step for $c = 0.9$ and in the fifth step for $c = 0.6$.

The results of the iterations can be summarised as follows: the iterations are converging successfully and fast if the variability of the traffic is low. This can be achieved by setting $\alpha = 2$ for the iterations. Smaller values for α can be used after the iterations, once the operating point has been reached. The average download volume v also has an influence on the variability, as already mentioned above: the variability is much lower for $v = 60$ KB as compared to $v = 200$ KB. This was reflected by a smaller oscillation amplitude during the iterations. Furthermore, long simulation runs improve the convergence.

Therefore, the following strategy for using iterations to increase the accuracy of the traffic match can be derived from the measurements:

- The first simulation should be used evaluate whether or not iterations should be used at all to improve the match of the prescribed traffic.

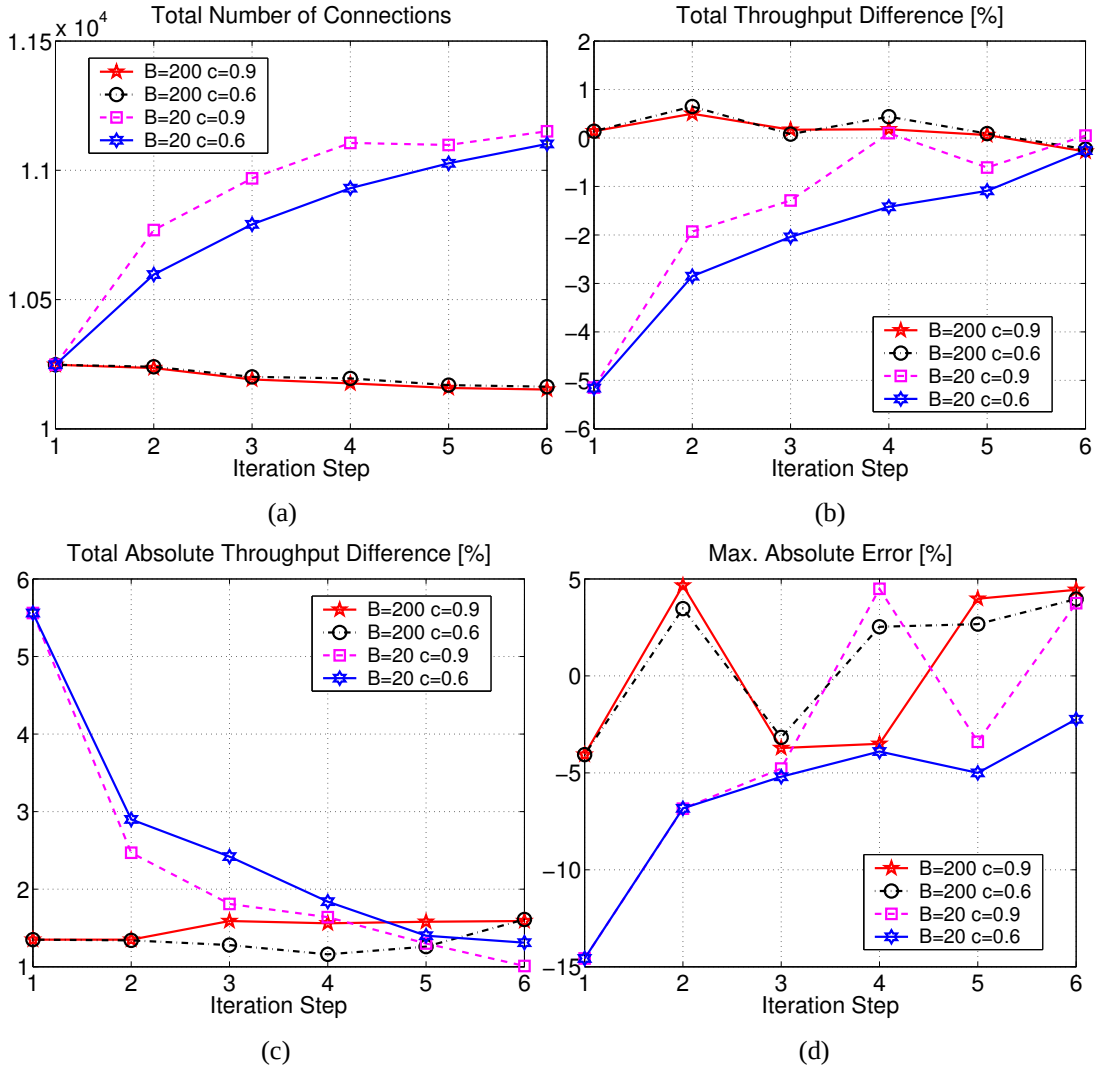


Figure 6.9: Convergence of iterations, parking-lot, $v = 60$ KB, $t_{off} = 40$ s.

- The iterations should be performed with a reduced variability of the power-tail distributed random variables (e.g. $\alpha = 2$) to increase convergence and reduce oscillations.

- The weighting constant should be set to a value close to one (e.g. $c = 0.9$) for large deviations and c should be reduced when reaching smaller error values to avoid strong oscillations.

- Increasing the simulation time reduces the number of iterations required to reach a given error threshold. However, the simulation has to terminate within a reasonable time; a compromise has the be found here.

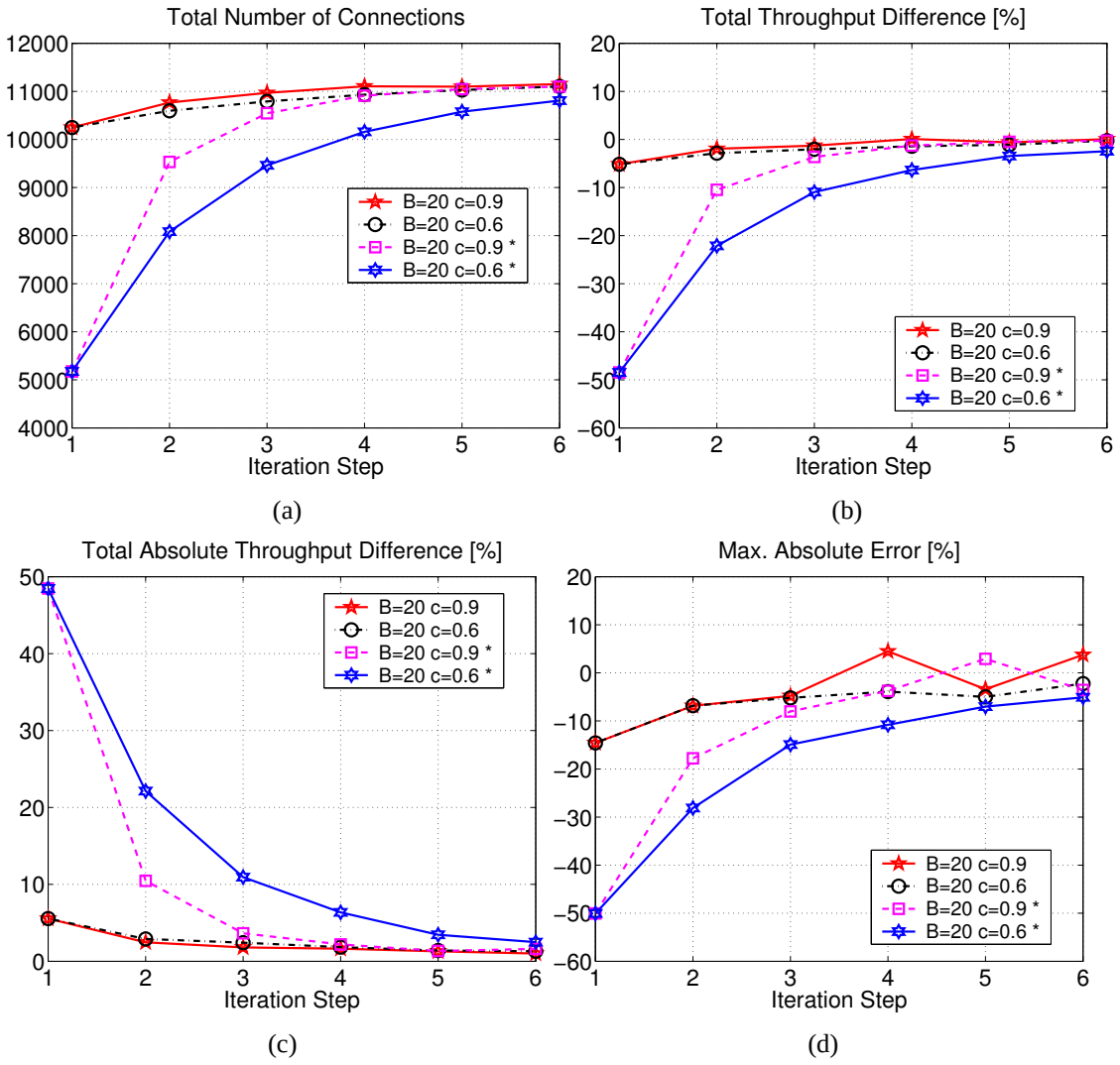


Figure 6.10: Convergence of iterations, parking-lot, $v = 60$ KB, $t_{off} = 40$ s, '*' denotes the simulations with a "bad starting point".

- The number of clients for all flows $c_{ij,k}$ of the last iteration conforming to certain error bounds can be used to drive simulations with full variability of all random variables, e.g. $\alpha < 2$ (see also Sec. 4.1.3).

However, in most situations the estimations of the algorithm for allocation of clients are accurate enough so that the first simulation shows already sufficiently small errors; no iterations are required in this case.

6.3 Comparison with the TCP-modified Engset Model

The simulation results are compared here with the TCP-modified Engset model [HLN97] described in Sec. 3.4. Moreover, the estimation results based on the algorithm for allocation of clients presented in Sec. 4.1 are given and discussed with the simulation results and the results of the TCP-modified Engset model. The performance comparison is presented for the average download rate of a single client given in Eq. (3.15) for the algorithm for allocation of clients and in Eq. (4.6) for the TCP-modified Engset model. The algorithm for allocation of clients is referred to as algorithm and the TCP-modified Engset model is called Engset model in this section for convenience.

The Engset model provides solutions for a single bottleneck model with homogeneous round-trip times and fair bandwidth sharing between the connections. The solutions for the Engset model are provided for the weighted average of the round-trip times since the round-trip times are not homogeneous in the considered scenarios:

$$RTT = \frac{\sum_{i=1}^{n} N_i \cdot RTT_i}{\sum_{i=1}^{n} N_i}. \tag{6.8}$$

The number of flows is denoted by n, the number of clients in flow i by N_i and the minimum round-trip time in flow i by RTT_i (sum of propagation delays). Further, some selected links are treated as if they were isolated from the rest of the scenario for the B-WiN in order to analyse to which degree the results from Engset model can be used in this way to predict the performance in a more complex network. The value r_{sim} is the weighted average of the average download rate of all clients with a connection routed over the considered link

$$r_{sim} = \frac{\sum_{i=1}^{n} N_i \cdot r_i}{\sum_{i=1}^{n} N_i}, \tag{6.9}$$

with the average download rate of each client r_i in flow i (average over all seeds).

The comparison is performed for the bottleneck and for the B-WiN scenario (four selected links) for the parameter sets $t_{off} = \{10, 40\}$ s, $v = \{60, 200\}$ KB and the four possible combinations. The deviation from the average throughput r_{sim} measured in the simulation is presented as relative deviation

$$\delta_{Eng} = 100 \cdot (r_{Eng} - r_{sim})/r_{sim} \tag{6.10}$$

for the Engset model and

$$\delta_{alg} = 100 \cdot (r_{alg} - r_{sim})/r_{sim} \tag{6.11}$$

for the algorithm presented in Sec. 4.1.

The results for the bottleneck scenario are given in Tab. 6.2. The relative deviation between the simulation results and the algorithm δ_{alg} is in the range $[-2, 1.81]$ %. The relative deviation between the simulation results and the Engset model δ_{Eng} is in the range $[-4.74, 0.30]$ %. Further, the deviation is in three of four cases larger for the Engset model than for the algorithm.

Table 6.2: Average throughput per client r_{sim} [kbit/s], bottleneck scenario, $c =$ 10 Mbit/s, $MTU = 1500$ Bytes, $B = 2500$, link capacity C [Mbit/s], RTT [ms], avg. page volume v [KB], t_{off} [s], target link load ρ_{targ} [%].

Link	C	ρ_{targ}	RTT	v	t_{off}	N	r_{sim}	δ_{alg}	δ_{Eng}
R1->R2	50	66	40.57	60	40	2484	12.58	-2.00	-4.74
R1->R2	50	66	40.57	60	10	635	48.43	-0.28	-1.42
R1->R2	50	66	40.57	200	40	774	41.58	-1.77	-4.23
R1->R2	50	66	40.57	200	10	198	156.69	1.81	0.30

The results for four selected links of the B-WiN scenario are given in Tab. 6.3. The relative deviations are in the range $\delta_{alg} \in [-2.72, 4.39]$ and $\delta_{Eng} \in [-5.82, 2.42]$. Moreover, the maximum deviation of the Engset are larger than those of the algorithm. Further, the Engset model provides better solutions for small off-time values than for large off-time values and the quality of the estimations seems to be independent of the volume. However, the algorithm shows opposite behaviour: the deviation is larger for small off-time values and for large volume values.

It can be concluded that TCP-modified Engset model and the algorithm for allocation of clients provide estimations of the average throughput of a single HTTP/TCP client which are quite close to the simulation results. However, the algorithm for allocation of clients yields better estimations. Further, a link with flows experiencing different RTTs can be modelled by the TCP-modified Engset model with the weighted average values for the RTT, as presented above. However, the model can only describe the situation of the bottleneck and the average of all clients. It is not possible to get information about the different characteristics of the flows sharing the bottleneck from the Engset model.

6.4 Matching Traffic Statistics

This section deals with the problem of matching the higher moments of the traffic statistics to measured values. Two traffic parameters are under focus in this section: the coefficient of variation of packet inter-arrival times and the Hurst parameter of a byte counting process which are of major importance for communication networks (see also Sec. 4.2

Table 6.3: Average throughput per client r_{sim} [kbit/s], B-WiN scenario, $c =$ 10 Mbit/s, $MTU = 1500$ Bytes, $B = 2500$, link capacity C [Mbit/s], RTT [ms], avg. page volume v [KB], t_{off} [s], target link load ρ_{targ} [%].

Link	C	ρ_{targ}	RTT	v	t_{off}	N	r_{sim}	δ_{alg}	δ_{Eng}
K->US	167	30.3	39.05	60	40	4105	12.31	0.16	-2.60
K->US	167	30.3	39.05	60	10	833	47.02	2.81	1.62
K->US	167	30.3	39.05	200	40	1117	40.78	0.20	-2.31
K->US	167	30.3	39.05	200	10	287	153.76	3.88	2.42
F->Ka	74.2	67.4	4.23	60	40	3857	12.69	-2.21	-5.59
F->Ka	74.2	67.4	4.23	60	10	967	48.97	1.12	-2.47
F->Ka	74.2	67.4	4.23	200	40	1186	42.29	-2.72	-5.82
F->Ka	74.2	67.4	4.23	200	10	298	158.81	3.36	-0.94
US->M	155	81.6	39.78	60	40	9941	12.36	-0.24	-3.07
US->M	155	81.6	39.78	60	10	2536	48.28	0.08	-1.10
US->M	155	81.6	39.78	200	40	3041	41.59	-1.78	-4.23
US->M	155	81.6	39.78	200	10	778	156.79	1.80	0.31
US->K	167	98.4	39.07	60	40	13029	12.05	2.32	-0.58
US->K	167	98.4	39.07	60	10	3324	46.97	2.92	1.32
US->K	167	98.4	39.07	200	40	3972	40.04	2.05	-0.80
US->K	167	98.4	39.07	200	10	1015	152.99	4.39	1.86

and 4.3). The variability of the traffic can be characterised by those two values. The simulation based measurements of the coefficient of variation are discussed in Section 6.4.1 and the Hurst parameter measurements are illustrated in Section 6.4.2.

Please note that the simulation results with $\alpha \leq 1.3$ show a larger variance, since the number of samples is not sufficient to guarantee a two digit accuracy on a mean estimation, as discussed already in Sec. 2.4. However, it might still be useful to detect the tendency of the parameters under consideration for very small values of the shape parameter α.

6.4.1 Coefficient of Variation

The coefficient of variation is defined as the ratio of standard deviation and mean value: $cv = \sigma/\mu$. The coefficient of variation of the packet inter-arrival time was measured at the input of the queues at the output ports of the routers. It is a measure of the burstiness of the traffic on the link. Traffic with negative exponentially distributed inter-arrival

times (i.e. Poisson traffic) has a coefficient of variation of one – the standard deviation and the mean value have the same value. Values larger than one indicate a larger traffic variability, which requires also larger buffers as compared to buffers dimensioned for Poisson traffic. The coefficient of variation is approximately constant for the three off-time values $t_{off} = \{10, 20, 40\}$. Therefore, the dependency on the off-time is not shown and discussed here.

The boxplots from the inter-arrival times of selected queues of the bottleneck scenario are shown in Fig. 6.11 for $t_{off} = 40$ s and $\alpha = \{1.1, 1.25, 1.5, 1.75, 2\}$. The queues corresponding to the measurements shown in Fig 6.11 (a)–(c) carry traffic with increasing RTT (cf. Sec. 5.6.1) but with a low utilisation of approximately 11 %. The coefficient of variation is in the range $[1.4, 1.7]$ in this case, indicating a significantly larger variability as compared to Poisson traffic. The core link carries the aggregate traffic and has an average utilisation of approx. 68.5 %; the coefficient of variation of approx. 1.15 is much lower in the core, see Fig. 6.11 (d). The coefficient of variation is in all cases approximately constand for $\alpha \geq 1.5$ and approaches slightly higher values for lower values of α, i.e. higher degree of self-similarity.

The influence of the link load and the RTT on the coefficient of variation can be summarised by comparing the four figures:

- Larger RTT values lead to smaller values of the coefficient of variation – TCP reacts slower and the traffic is therefore less bursty.
- The coefficient of variation decreases with increasing utilisation – the variability is limited when the instantaneous throughput reaches the link capacity frequently (cf. Fig. 6.11 (a) and (d)). The coefficient of variation reaches in the case of Fig. 6.11 (d) approximately the value one, indicating that the traffic has approximately the same variability as Poisson traffic. The correlation degree and structure is not measured by the coefficient of variation. Hence, the traffic might still be somewhat different from Poisson traffic when looking at e.g. buffer requirements.

The measurements corresponding to the links from R2 to S4, S5 and S6 are shown in Fig. 6.12.The three flows from S1 to S4, S2 to S5 and S3 to S6 are all routed over the bottleneck link “R1->R2”. Therefore, the traffic flow in Fig. 6.11 (a) and Fig. 6.12 (a) is the same, except that it has passed the bottleneck link in the latter case. The same holds for Fig. 6.11 (b) and Fig. 6.12 (b) and for Fig. 6.11 (c) and Fig. 6.12 (c). One interesting effect becomes visible when comparing the measurements in Fig. 6.11 with those in Fig. 6.12: the astonishing fact is that the coefficient of variation is very similar before and after passing the bottleneck, although the coefficient of variation is much smaller in the bottleneck. It seems as if the multiplexing in the router does not have an impact on the coefficient of variation, which is surprising.

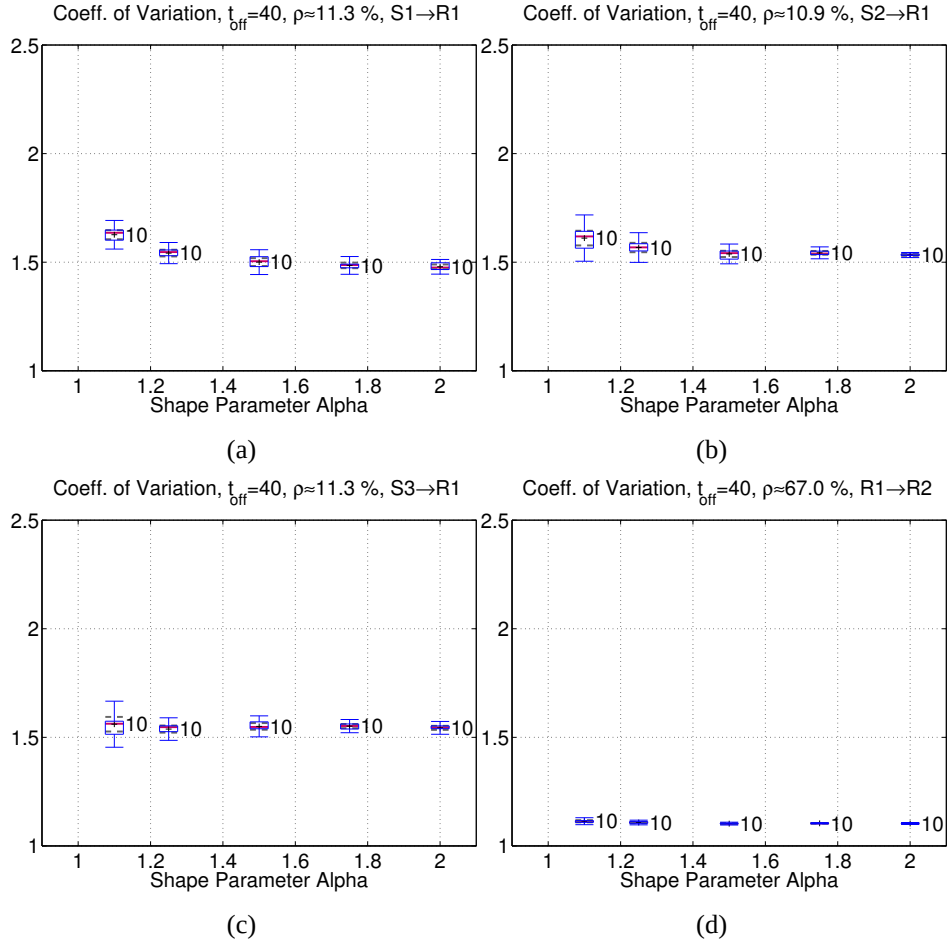


Figure 6.11: Coefficient of variation of IATs at queue input, bottleneck.

At least two different cases can lead to this effect: First, if the utilisation would be so small that the buffer is not used at all, then it could be expected, that the independently generated processes do not correlate in the bottleneck. Hence, the coefficient of variation would be the same before and after the bottleneck. However, an average buffer size of more than 15 packets was measured which indicates that this is not the case here.

Second, it is clear that mean value μ and standard deviation σ are affected by the multiplexing in the router. However, it can be always expected that the mean value is the same after demultiplexing if very few losses occur in the router buffers, as it is the case here. Due to the fact that the coefficient of variation is the same after demultiplexing, it can be deduced that also the standard deviation must be the same before multiplexing and after demultiplexing. Therefore, the effect of multiplexing and demultiplexing on the standard deviations discussed in the following.

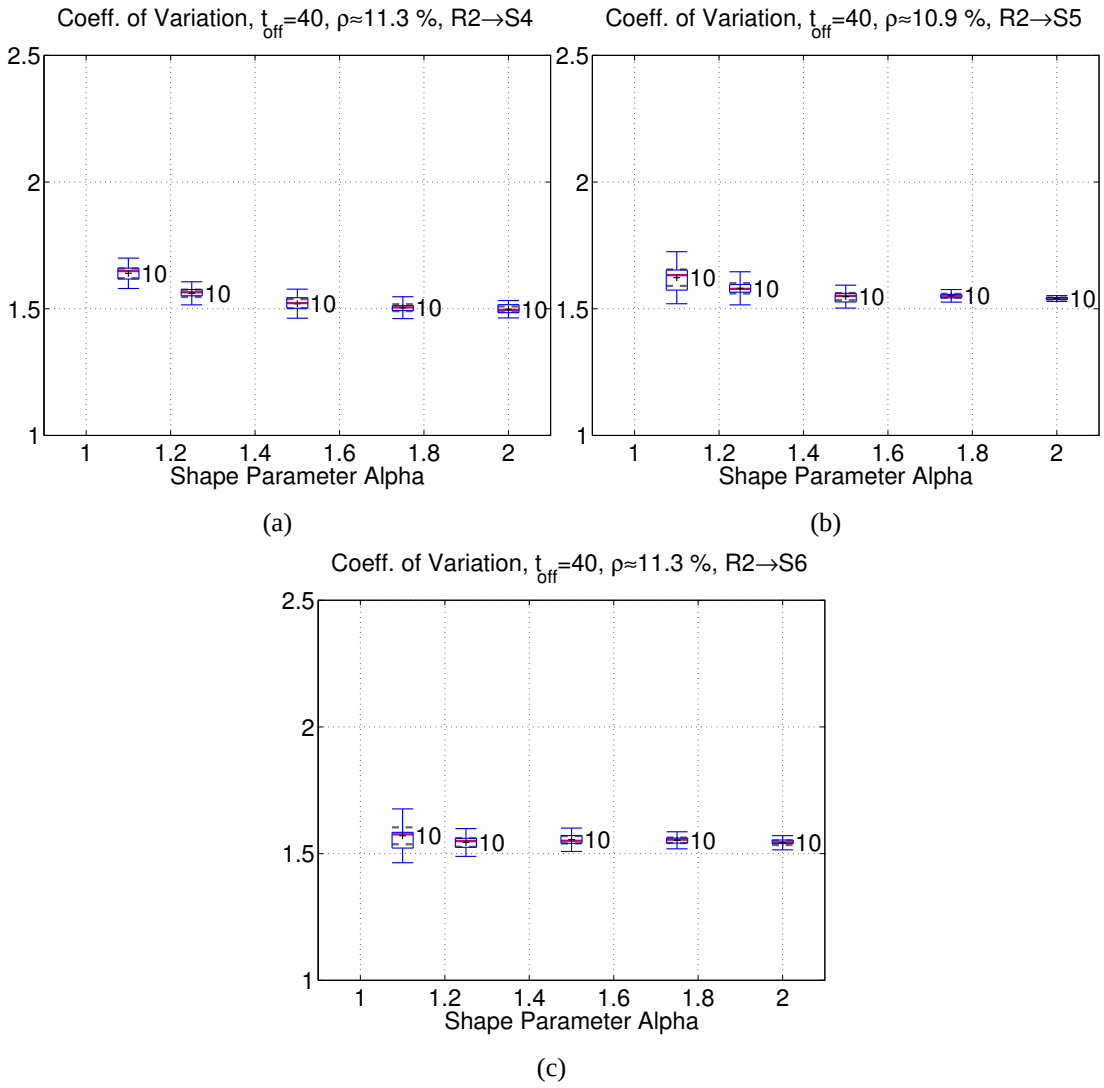


Figure 6.12: Coefficient of variation of IATs at queue input, bottleneck.

The packet stream of each flow i forms a stochastic process with random variables T_i, $i = 1, \ldots, n$. The random variables T_i consist of the packet inter-arrival time between two successive packets of all packets with the same source and destination. The multiplexing can be analytically modelled by the addition of the random variables. The mean value of the sum of random variables is given by [LG89]:

$$\mu_s = \sum_{i=1}^{n} \mu_i. \tag{6.12}$$

The variance of the sum of random variables is [LG89]

$$\sigma_s^2 = \sum_{i=1}^{n} \sigma_i^2 + 2 \cdot \sum_{i \neq j} \mathrm{COV}\left[T_i, T_j\right]. \tag{6.13}$$

The covariance of statistically independent random variables is zero. Furthermore, the covariance in Eq. (6.13) is responsible if the demultiplexed streams would have a different coefficient of variation as compared to before multiplexing. However, if the streams are not completely independent, but if the covariance values are very small compared to the variances, then it can be expected that the influence of multiplexing and demultiplexing on the coefficient of variation is small. For the case that the covariances are very small (or zero) compared to the variances, the coefficient of variation of the multiplexed process can be approximated by

$$cv_s = \frac{\sigma_s}{\mu_s} \tag{6.14}$$

$$\approx \frac{\sqrt{\sum_{i=1}^{n} \sigma_i^2}}{\sum_{i=1}^{n} \mu_i}. \tag{6.15}$$

For the case that all flows have the same mean value μ and the same variance σ the resulting coefficient of variation of the multiplexed stream (for small covariances) reduces with n according to the Generalised Central Limit Theorem (GCLT) presented in Sec. 2.4 to

$$cv_s \approx \frac{1}{n^{1-1/\beta}} \cdot \frac{\sigma}{\mu}. \tag{6.16}$$

The parameter β represents the shape parameter of the heavy-tailed inter-arrival time processes before multiplexing.

The inter-arrival times were saved for one bottleneck simulation in order to evaluate whether or not the covariance values are small compared to the variances for $\alpha = 1.5$, $v = 60$ KB and $t_{off} = 40$ s. A special case with homogeneous round-trip times was selected for this case because otherwise the values of cv_i and α_i differed significantly due to the different round-trip times (cf. Tab. 6.4). Each of the three flows per direction contains more than $1.5 \cdot 10^6$ values. The measurements of mean value μ_i, standard-deviation σ_i, coefficient of variation cv_i and shape parameter α_i are are shown in Tab. 6.4 for the three flows and the multiplexed flow. The covariance values where in all cases smaller than $3.68 \cdot 10^{-9}$ for all three flows, more than two orders of magnitude smaller than the variance values. Therefore, the streams can be treated as approximately independent,

which is the reason why the values of *cv* before multiplexing and after multiplexing are approximately the same.

Table 6.4: Stochastic parameters of inter-arrival times, bottleneck, $\alpha = 1.5$, $v =$ 60 KB and $t_{off} = 40$ s.

	S1->R1	S2->R1	S3->R1	R1->R2
μ_i	$4.589 \cdot 10^{-4}$	$4.493 \cdot 10^{-4}$	$4.402 \cdot 10^{-4}$	$1.498 \cdot 10^{-4}$
σ_i	$6.965 \cdot 10^{-4}$	$6.912 \cdot 10^{-4}$	$6.783 \cdot 10^{-4}$	$1.536 \cdot 10^{-4}$
cv_i	1.518	1.538	1.541	1.025
α_i	1.73	1.62	1.56	1.71

The shape parameter values α_i are unfortunately slightly different for all three flows. However, small shape parameter values are dominating with the heavier tail of the distribution. Further, inserting the values in Eq. (6.16) with the the average of cv_i yields

$$cv_s = \frac{1}{3^{1-1/1.56}} \cdot 1.532 = 1.032,$$

which matches closesly the value of 1.025 measured for the multiplexed stream on the link "R1->R2".

Therefore, it can be concluded that this multiplexing effect can be explained with the GCLT and the fact that the covariance values are more than two orders of magnitude smaller than the variances: the small variance values are responsible for the fact that the coefficient of variation is approx. the same before multiplexing and after demultiplexing and the GCLT describes the coefficient of variation of the multiplexed traffic.

The coefficient of variation of selected locations of the parking-lot scenario is shown in Fig. 6.13. The access links are even more over-provisioned as compared to the bottleneck case – the average link utilisation in Fig. 6.13 (a)–(c) is approximately 1.5 % and consequently the values achieved for the coefficient of variation are larger than in the bottleneck. The coefficient of variation covers the range $[1.15, 2.15]$ for the parking-lot. The core link is similarly loaded as in the bottleneck case ($\rho \approx 70$ %) and therefore coefficient of variation is also very close to one, see Fig. 6.13 (d).

The measurements of the coefficient of variation for the B-WiN scenario are depicted in Fig. 6.14. The biggest value of the coefficient of variation of the IATs was measured at the queue of the link from node K to node F (cf. Fig. 6.14 (a)). The link carries traffic flows routed within the German part of the network, the flows experience a small RTT. The link was utilised to approximately 27 % and the coefficient of variation was approx. 1.75. The boxplots of the measurements at the link with the highest utilisation of

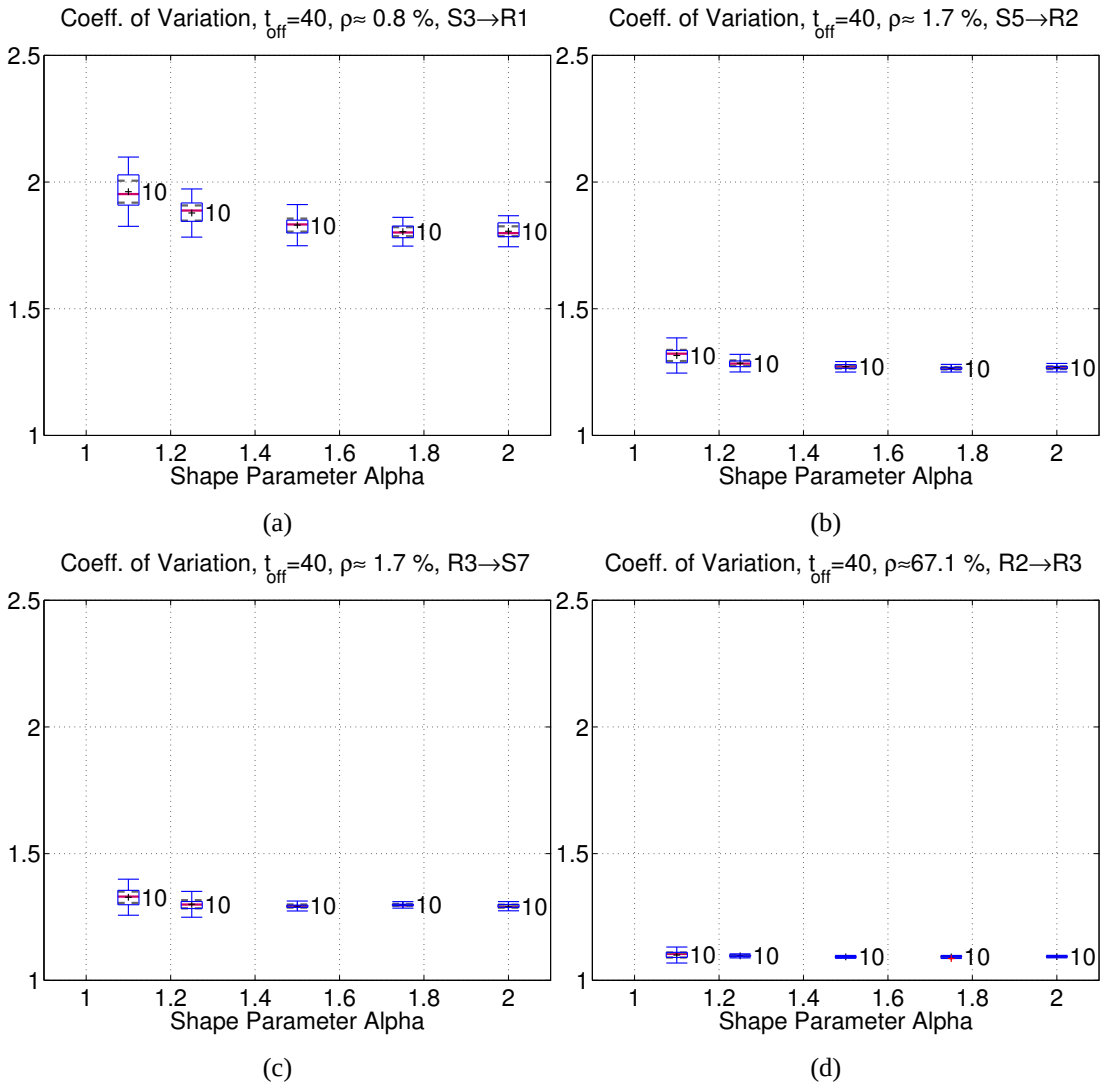


Figure 6.13: Coefficient of variation of IATs at queue input, parking-lot.

approx. 97 % (Fig. 6.14 (b), “US->K”) indicate a coefficient of variation of about 1.27 and a very small range of the values in contrast to Fig. 6.14 (a).

The coefficient of variation of two links originating from node M with destination node S (within Germany, small RTT values) and US (USA, large RTT values) is shown in Fig. 6.14 (c) and (d), respectively. The link utilisation is approximately 64 % in the first case and 34 % in the second case. Among the two competing effects on the coefficient of variation (utilisation and RTT, s.a.) the RTT has the stronger impact in this case: the average and the range of the coefficient of variation is larger in Fig. 6.14 (c) than in Fig. 6.14 (d).

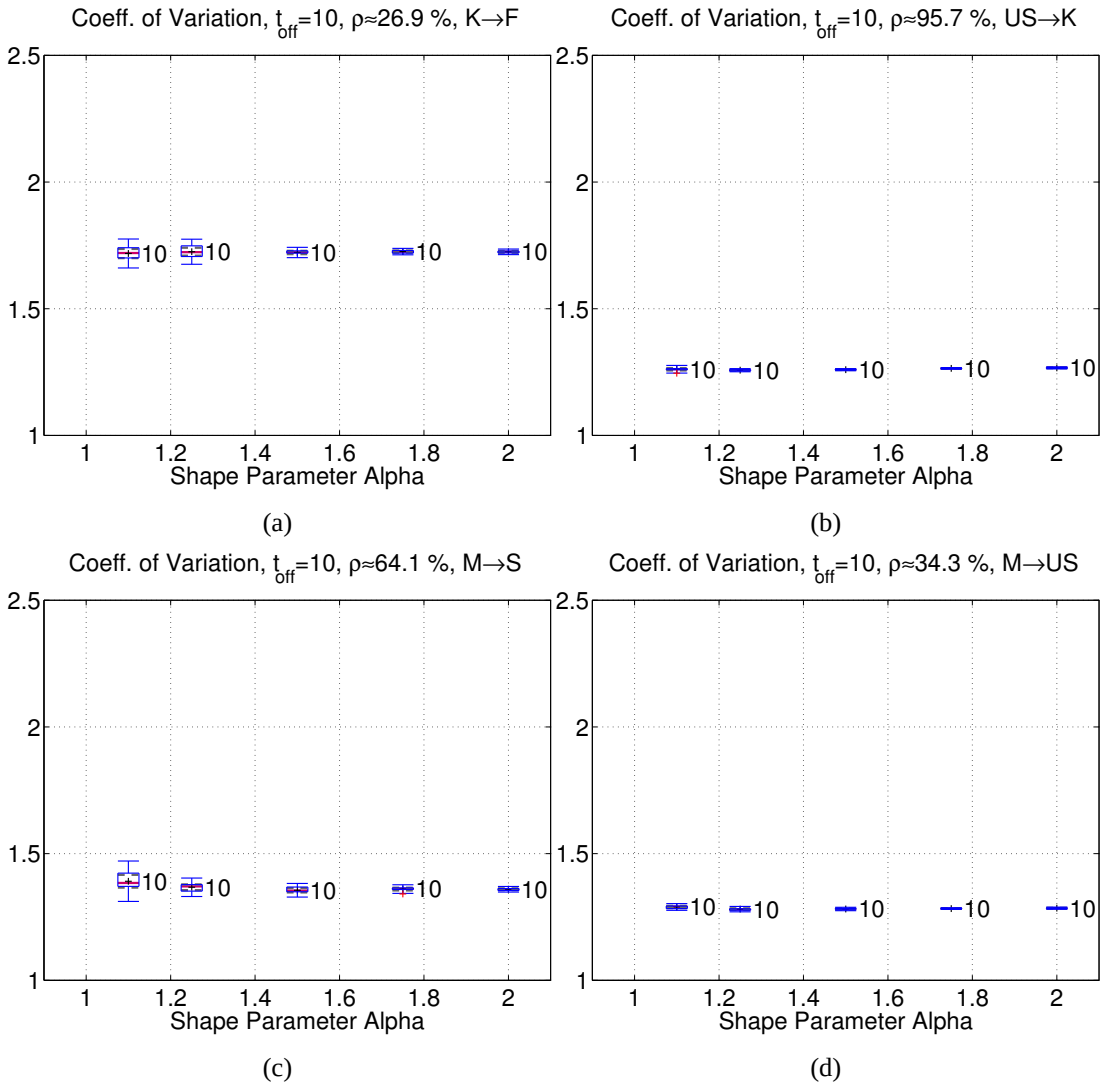


Figure 6.14: Coefficient of variation of IATs at queue input, B-WiN.

The influence of the Maximum Segment Size (MSS) value of TCP is shown in Fig. 6.15: the coefficient of variation is much larger for small values of the MSS if the link load is small, see Fig. 6.15 (a). However, the influence of the MSS on the bottleneck link in Fig. 6.15 (b) is negligible; the higher link load seems to dominate strongly such that the effect of the MSS is hardly visible in the bottleneck.

The influence of the average download volume on *cv* is shown in Fig. 6.16: a download volume of $v = 200$ KB (Fig. 6.16 (a)) results in a significantly higher coefficient of variation as compared to a download volume of $v = 60$ KB (Fig. 6.16 (b)). The reason is that the correlation and burstiness introduced by TCP are weak for very short connections

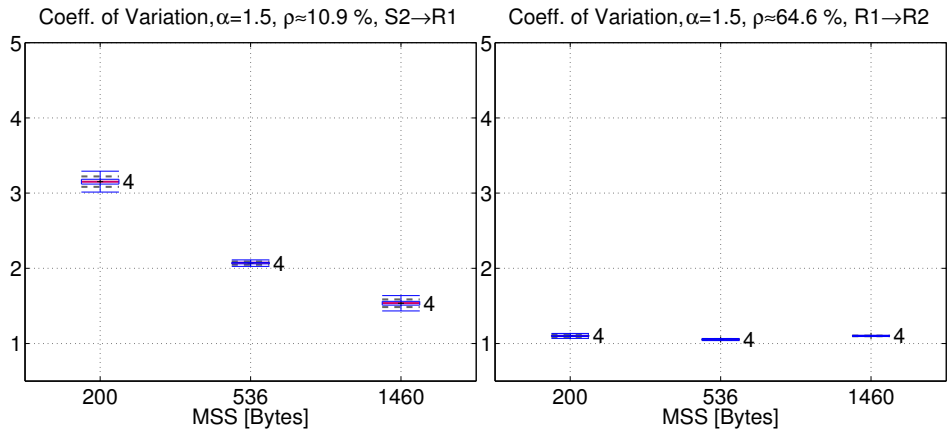


Figure 6.15: Coefficient of variation of IATs for different values of MSS, bottleneck.

due to a small congestion window. But the correlation is stronger for longer connection durations with a large congestion window. Therefore, the coefficient of variation increases with the connection duration (or download volume).

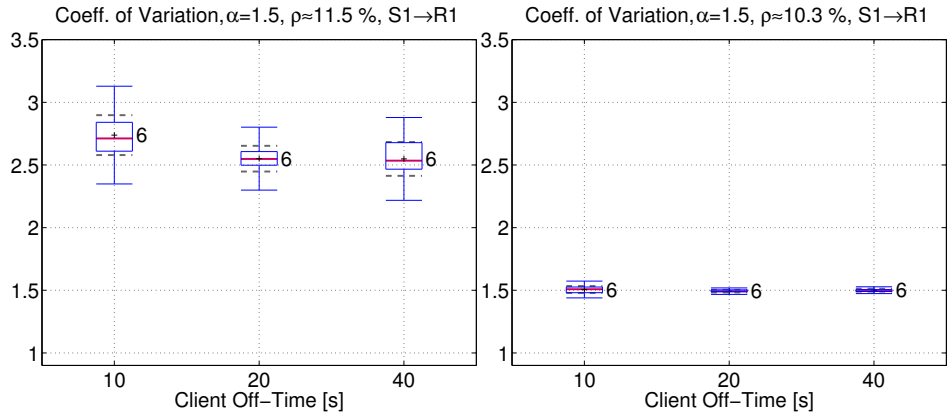
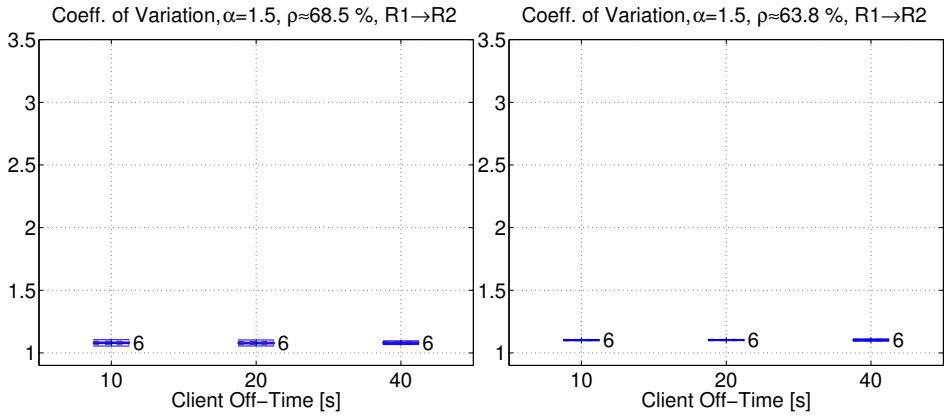


Figure 6.16: Coefficient of variation of IATs for (a): $v = 200$ KB and (b): $v = 60$ KB, bottleneck with $\alpha = 1.5$, S1->R1.

However, the effect on the measurements on the bottleneck link in Fig. 6.17 indicate that the effect is only marginal here: the values for $v = 200$ KB in Fig. 6.17 (a) and for $v = 60$ KB in Fig. 6.17 (b) are approximately the same. The same tendency was observed also for different MSS values. The reason is that the coefficient of variation is dominated

by the link utilisation; parameters like the average download volume and the MSS have only an impact on *cv* for small link loads.



(a) (b)

Figure 6.17: Coefficient of variation of IATs for (a): $v = 200$ KB and (b): $v = 60$ KB, bottleneck with $\alpha = 1.5$, R1->R2.

Recent measurements of inter-arrival times conducted in [IPL02, NZI00] show very similar coefficients of variation as compared to our simulation results. The measurements on a 100 Mbit/s link in year 2000 [NZI00] with an average load of approx. 7 % showing a coefficient of variation of $cv \approx 1.26$. The second measurement [IPL02] was performed in year 2002 on an OC-48 link (2.4 Gbit/s) with an average utilisation of approx. 36 % and the result was $cv \approx 1.01$. The range of the measurements with the B-WiN simulations was $[1.175, 1.752]$ for an average download volume of 60 KB, depending on the selected link and the link utilisation. This confirms that the simulations match the reality quite well and that there is no special need to tune this parameter. Nevertheless, the parameter could be further tuned by e.g. changing the average download volume or the MSS, as shown above.

6.4.2 Hurst Parameter

The Hurst parameter measurements are based on simulations with 10 different seeds for each value of the shape parameter α of the HTTP object size distribution. The average HTTP download volume was set to 60 KB and the average off-time was set to $t_{off} = 10$ s for this experiment. Qualitatively the same behaviour was also observed for an off-time of 40 s. However, the larger memory and simulation time requirements did not allow to drive simulations with the same number of seeds for $t_{off} = 40$ s. Estimated Hurst parameters are ignored if the Abry-Veitch Hurst parameter estimator (described

in Appendix A.1) reports a bad quality of fit. Therefore, the number of samples for the boxplots is not always 10 in this section.

Mainly three different cases could be identified evaluating the plots for all 36 links of the B-WiN:

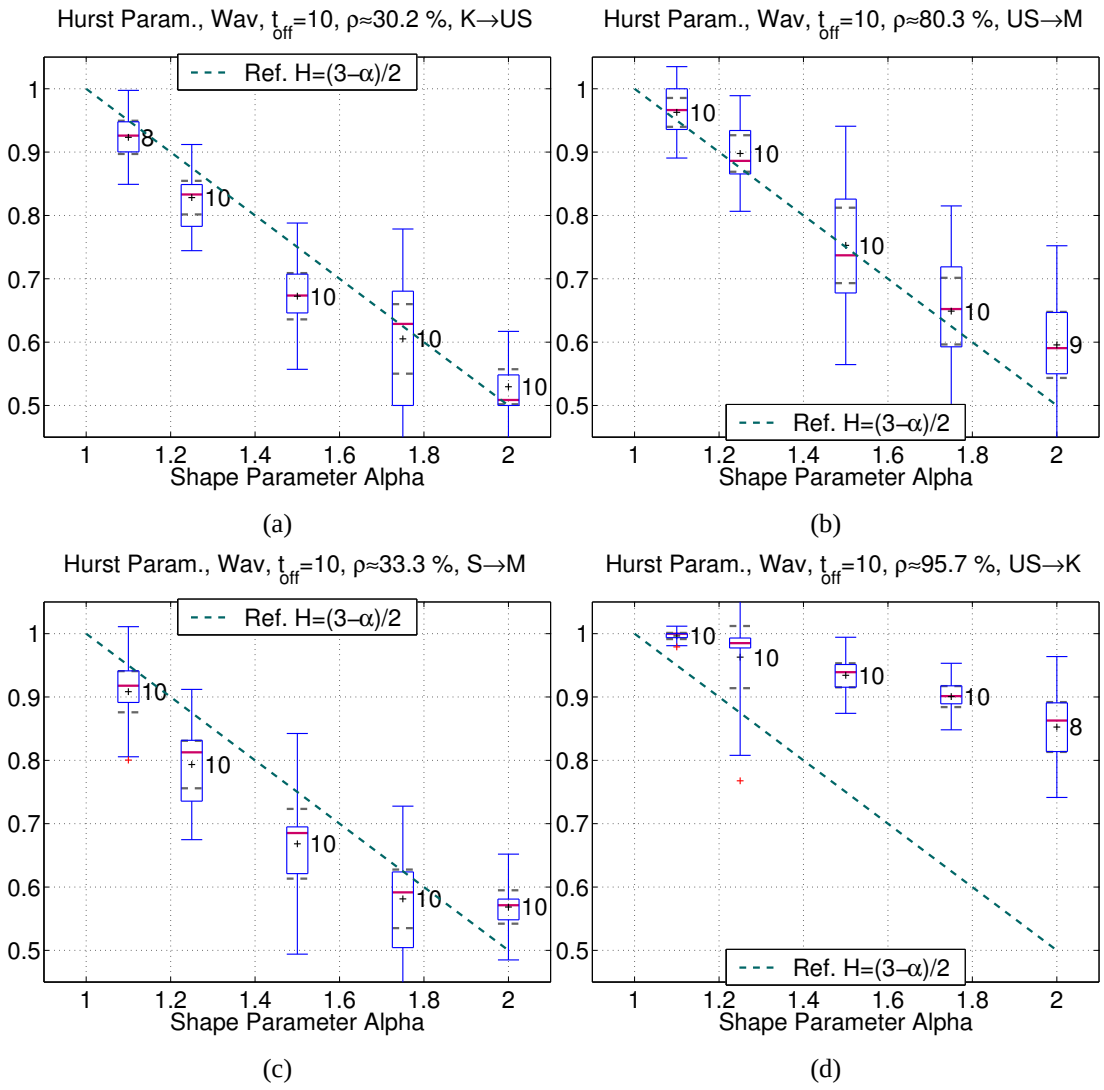


Figure 6.18: Hurst parameter H versus shape parameter α of the HTTP object size distribution, B-WiN.

- The Hurst parameter follows approximately $H = (3 - \alpha)/2$ as shown in Fig. 6.18 (a) and (b) (cf. Sec. 4.2).

- The Hurst parameter shows a large variability and the values are sometimes substantially lower than expected from $H = (3-\alpha)/2$ (cf. Fig. 6.18 (c)).
- The Hurst parameter remains high for large values of α, even for $\alpha = 2$, as apparent in Fig. 6.18 (d).

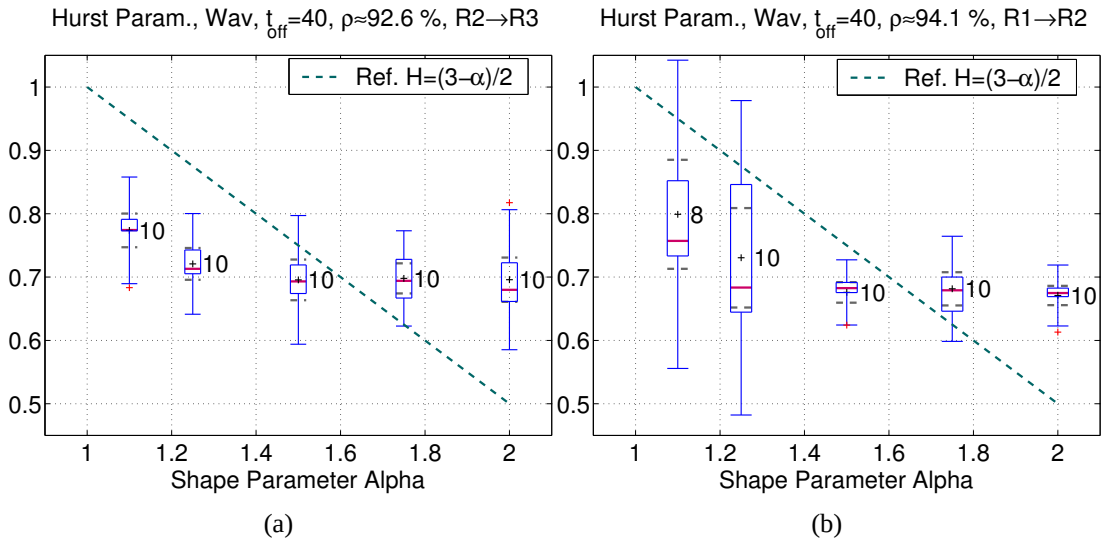
The average Hurst parameter values show a behaviour as expected in Fig. 6.18 (a) and (b). Obviously the traffic within Germany, that experiences very small RTTs, adds some variability to the Hurst parameter values (cf. Fig. 6.18 (c)). The last case, high Hurst parameter values even for $\alpha = 2$ in Fig. 6.18 (d), is discussed detailed in the following.

6.4.3 Hurst Parameter for High Link Load

The Hurst parameter estimations for the B-WiN show an interesting behaviour for one link with a very high traffic load of $\rho \approx 97$ %, see Fig. 6.18 (d): the estimated values for the Hurst parameter are generally larger than those for the low load case; especially $\alpha = 2$ still results in a high Hurst parameter of $H \approx 0.84$ (cf. Fig. 6.20 (b)). This result was completely different from what was expected. The traffic was expected to tend to Poisson behaviour when the load increases [CCLS01]. Furthermore, the process requires some variability to establish self-similarity: the property "slowly decaying variance" of self-similar processes in Eq. (2.11) can only be fulfilled when the variance of the process is sufficiently large. However, a completely filled link, where the packets follow each other without any gap, would have a variance of zero. Therefore, it was expected that the Hurst parameter is very small already for link loads close to 100 %.

However, the measurements on link "US->K" shown Fig. 6.18 (d) with a traffic load of $\rho \approx 97$ % show a high Hurst parameter for a very high link load even for $\alpha = 2$. Qualitatively the same phenomenon – high Hurst parameter values for high link utilisation – was observed for a larger number of connections (with an off-time of 40 s). A test with different buffer sizes revealed that the phenomenon is also not related to a specific buffer size. A simulation with individual timers for each TCP connection showed the same behaviour (see discussion about TCP timers in Sec. 5.5), therefore the simplified timer handling is also not responsible for the phenomenon.

Simulations with the parking-lot scenario and high link load support the findings, although the indicated Hurst parameter values are around $H \approx 0.7$, smaller than in the B-WiN, see Fig 6.19 (a). However, the Hurst parameter was also significantly larger, as if a lower target load was selected. A similar qualitative behaviour of the Hurst parameter is visible for the bottleneck scenario in Fig. 6.19 (b): the Hurst parameter stays at $H \approx 0.67$ for $\alpha = 2$. However, the increase of the Hurst parameter for smaller values of α is different in bottleneck and parking-lot as compared to the B-WiN (compare Fig. 6.19 (a) and (b) with 6.18 (d)).



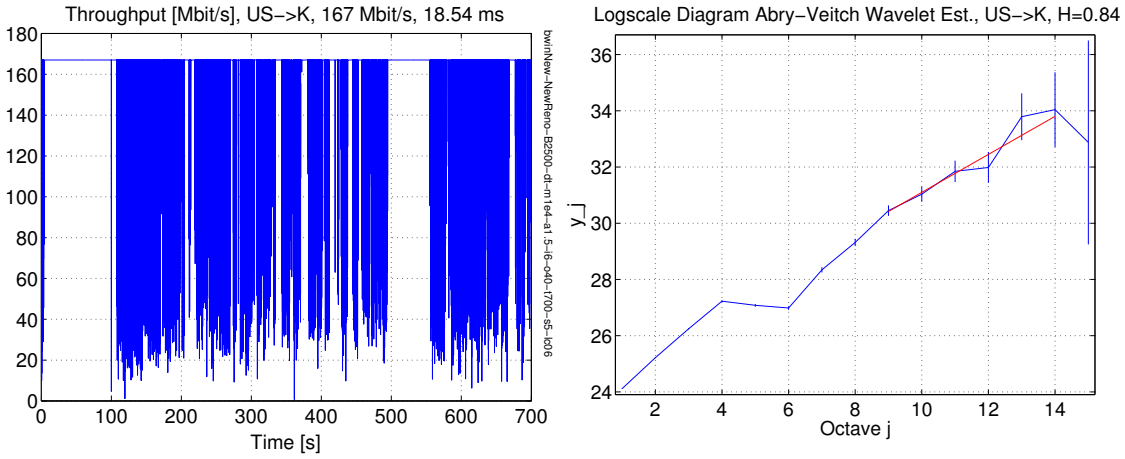
(a) (b)

Figure 6.19: Hurst parameter as a function of the shape parameter α, (a): parking-lot, R2->R3, (b): bottleneck, R1->R2.

The presence of the phenomenon for different scenarios and various settings indicate that the phenomenon is not related to a special setting or bug in the simulation. It shows that this is a new phenomenon that was not discovered before, to the best knowledge of the author. An intuitive understanding of the phenomenon of high Hurst parameter values for high utilisation (and high values of α) is developed in the following.

The counting process of the B-WiN scenario (bytes counted per 2.5 ms, scaled to Mbit/s) for $\alpha = 2$ is shown in Fig. 6.20 (a). The link utilisation is high, but there are also some very deep "dips" which are responsible for the variability and long-range dependence of the process that becomes obvious in high Hurst parameter values, as visible in the logscale diagram in Fig. 6.20 (b). The slope of the logscale diagram is used by the Abry-Veitch Hurst parameter estimator to estimate the Hurst parameter. The ordinate represents the energy of the wavelet coefficients (second moment). The small time scales (or octaves) $1 \leq j \leq 7$ represent the short-term behaviour and the values $j \geq 8$ represent the long-range scaling behaviour. See Appendix A.1 for a more detailed explanation of the Abry-Veitch estimator and the logscale diagram.

The dips are caused by TCP time-outs. Especially for very short-lived TCP connections, like it is the case for WWW traffic, it is likely that a time-out occurs when there are not enough samples for a good RTT estimation such that default time-out values are used. Those values are very large, 6 s and 12 s for the first and the second RTT, respectively (cf. Sec. 3.2.4). Another possibility for large time-out values is when consecutive time-outs occur. The time-out duration is doubled for each time-out in this case. The time-outs introduce long-range dependence into the process: the time-out duration of several sec-



(a) (b)

Figure 6.20: (a): Counting process, (b): logscale diagram Abry-Veitch wavelet estimator, B-WiN, US->K.

onds is very large compared to packet transmission times and RTT values. Further, the connection duration is also extended by the time-out. The fact that it is likely that several connections are suffering from time-outs at the same time, due to buffer overflow, introduces further correlation that is persisting over a longer period and thus supports the long-range dependence.

To further support the understanding, an artificial sequence is produced that shows similar features. The so called Cantor set [Sta98, p. 181 ff] is a famous construct appearing in many books on chaos and fractals. It forms a purely deterministic process with implicit self-similarity. It can be constructed with the recursive Algorithm 2 (see also Appendix A.2 for the Perl code). The algorithm of the Cantor set is described here with the special interpretation as a byte counting process. The average traffic load of such a counting process is very low, since many count values are set to zero. However, the inverse process is defined as $v_{inv,count} = \max(v_{count}) - v_{count}$ has a very high traffic load and can be compared with the simulation data.

The resulting Cantor set counting process for a simulation time of 600 s and a counting window of 2.5 ms for a link with a capacity of 167 Mbit/s created with Algorithm 2 is shown in Fig. 6.21. The threshold for the termination was set to 30 because a very similar average load as in the B-WiN setting was reached with this value. The smallest sub-intervals, where the values are constant, contain therefore at least 30 values. These intervals contain the peaks that are translated into dips by the inversion. The time duration of the dips is therefore $30 \cdot 2.5\,\text{ms} = 75$ ms. The length of the vector and the threshold value should be a multiple of three in order to avoid strange artefacts.

1. Start with a vector v of length l, v represents the counting process. Set all counting values to the maximum byte-count (link capacity multiplied by time unit used for counting window).
2. Set all values in the interval $[l \cdot 1/3, l \cdot 2/3]$ to zero, where l is the maximum index of the vector.
3. Divide the intervals remaining from the previous steps into three sub-intervals of the same size and set all values in the middle interval to zero.
4. Repeat step 3 recursively until the remaining sub-intervals have a size smaller or equal to a certain threshold. The threshold Thr determines the finest time scales for which the scale invariance holds (cf. Fig. 6.22). Setting the threshold to one results in scale invariance on all scales of the process.
5. Build the "inverse process" by setting all values to the maximum byte-count minus the previously calculated value: $v_{inv,count} = \max(v_{count}) - v_{count}$.

Algorithm 2: Creation of a Cantor set.

The counting process of the Cantor set in Fig. 6.21 looks only a little similar to the measured counting process in the B-WiN Fig. 6.20, since the Cantor set has a completely deterministic and regular shape. Nevertheless, the Hurst parameter estimation with the wavelet estimator (see Appendix A.1) in Fig. 6.22 (a) shows a Hurst parameter of $H = 0.79$ for $\alpha = 2$ and looks similar to Fig. 6.20 (b), especially the range $j \geq 8$.

The shape of the curve in Fig. 6.22 (a) with a threshold value of 30 depends for small octaves $j = 1-5$ on the stopping criterion of the algorithm. That is, setting the threshold to a smaller value results in a more consistent slope of the curve also for small octaves (see Fig. 6.22 (b) with a threshold value of 3). The average traffic load on the other hand increases for a smaller threshold value ($\rho = 99.15$ % for a threshold value of 3).

The dips are limiting the maximum throughput on the network providers links and thus have an impact on the network providers income. If there are some long-lived connections, e.g. FTP downloads, which are not modelled in this work, these connections could fill up the dips as well.

A new graph type is introduced in the following to improve the understanding of the relation between Hurst parameter and link load: the correlation between link load and Hurst parameter value is shown in a plot, where the x-axis represents the link load in percent, the y-axis represents the Hurst parameter value and a diamond is drawn for each measured value pair.

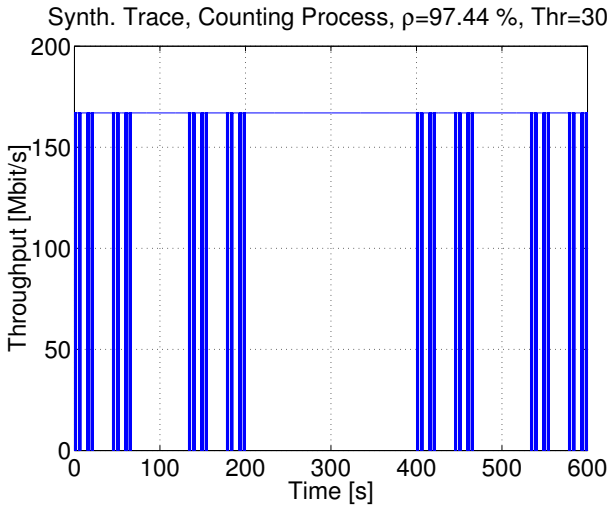
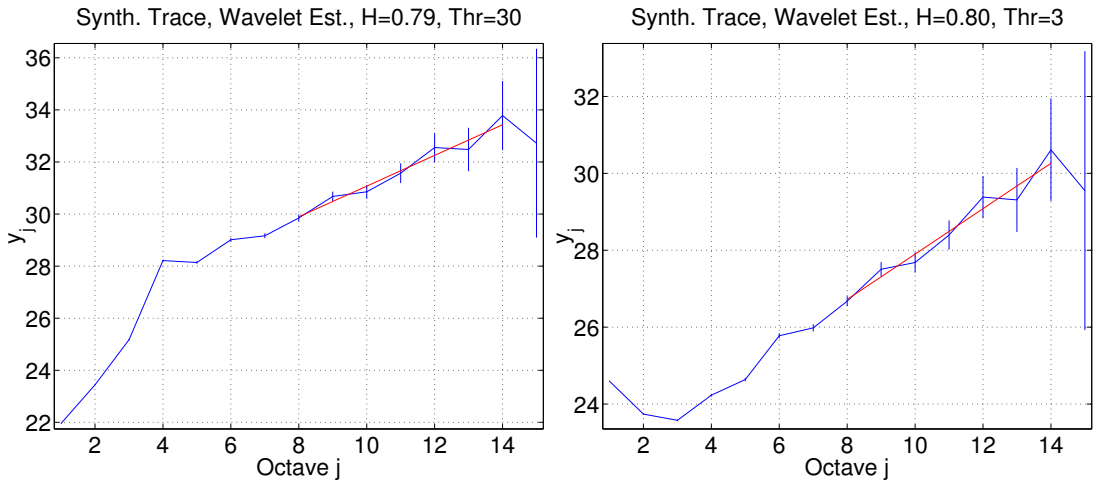


Figure 6.21: Counting process derived from Cantor set, average load $\rho = 97.44$ %, threshold $Thr = 30$.

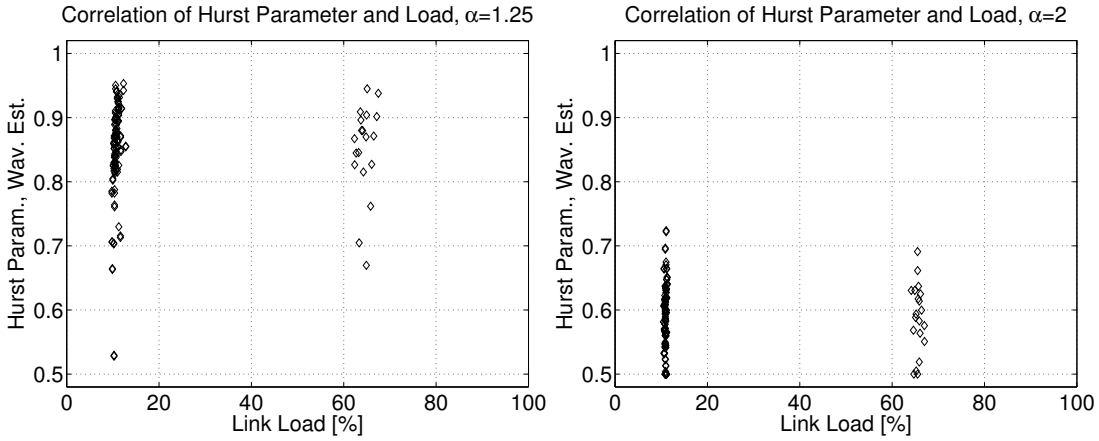


(a) (b)

Figure 6.22: Wavelet estimation of Cantor set, (a): threshold $Thr = 30$, (b): threshold $Thr = 3$.

The correlation plot for the bottleneck with max. load of $\rho \approx 66$ % is shown for $\alpha = 1.25$ and $\alpha = 2$ in Fig. 6.23 (a) and (b), respectively. The theoretical Hurst parameter values are $H = 0.875$ for $\alpha = 1.25$ and $H = 0.5$ for $\alpha = 2$ (cf. Eq. (2.12)) for the case that the file size distribution is the only source of self-similarity. The figures show two clouds, one cloud around $\rho \approx 11$ % for the links between the source node and the second cloud around $\rho \approx 66$ % for the measurements at the core link. The Hurst parameter values are

around 0.875 for $\alpha = 1.25$, as expected. The Hurst values for $\alpha = 2$ are mostly larger than 0.5; the effect that TCP introduces some self-similarity becomes visible.



(a) (b)

Figure 6.23: Correlation between Hurst parameter and link load, bottleneck.

The measurements are significantly different for the bottleneck under high load ($\rho \approx$ 95 %), as shown in Fig. 6.24: the spreading of the values is significantly larger for $\alpha = 1.25$ (Fig. 6.24 (a)) than for Fig. 6.23 (a). Further, the Hurst parameter values are significantly larger for $\alpha = 2$ in Fig. 6.24 (b) as compared to Fig. 6.23 (b). Moreover, high Hurst values are measured for the core as well as for the lightly loaded links. The reason for this is, that the self-similarity, introduced by high load at the bottleneck, is transported from TCP to the other links carrying the same traffic flows as well [VKMV00]. That is, the stochastic properties of the traffic inserted at the core propagate through the network, such that the complete path is affected. The correlation plots for the parking-lot do not show any new behaviour, they are not shown here.

The correlation plots between link load and Hurst parameter are shown in Fig. 6.25 for the B-WiN scenario. The large number of links with different link loads between 25 % and 98 % are represented by clouds with a larger horizontal spreading, as compared to the bottleneck case. The majority of Hurst values for link loads below 90 % are consistent with the theoretical values (s.a.). However, the Hurst parameter values for link load values above 90 % are significantly larger than for lower link load values. Further, the correlation plot for $\alpha = 2$ in Fig. 6.25 indicates that the very high Hurst parameters are correlated with high link load values.

A comparison with UDP traffic is shown in the following in order to assess the role of TCP for the Hurst parameter values at high link loads. The UDP traffic was generated by an aggregated source model called SupFRP [RN96, Hug97] with $\alpha = 1.25$ and $\alpha \approx 2$

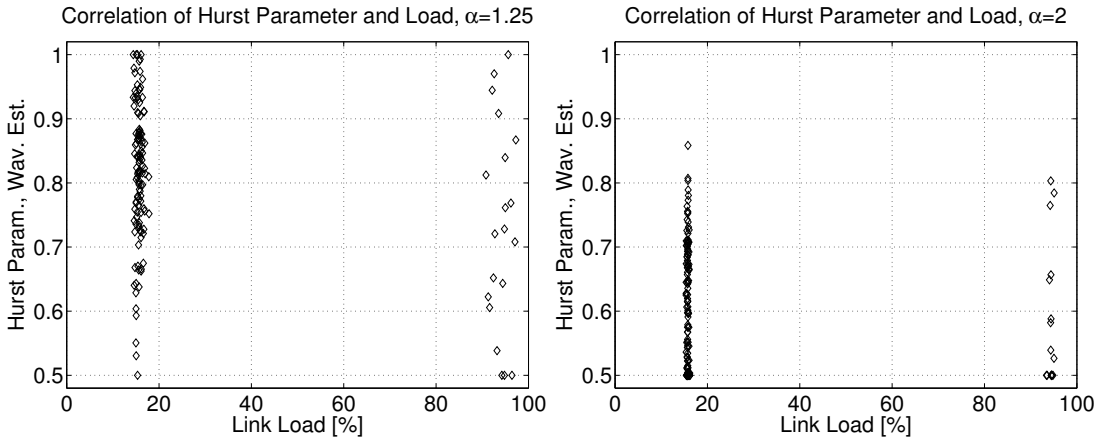


Figure 6.24: Correlation between Hurst parameter and link load, bottleneck, high load.

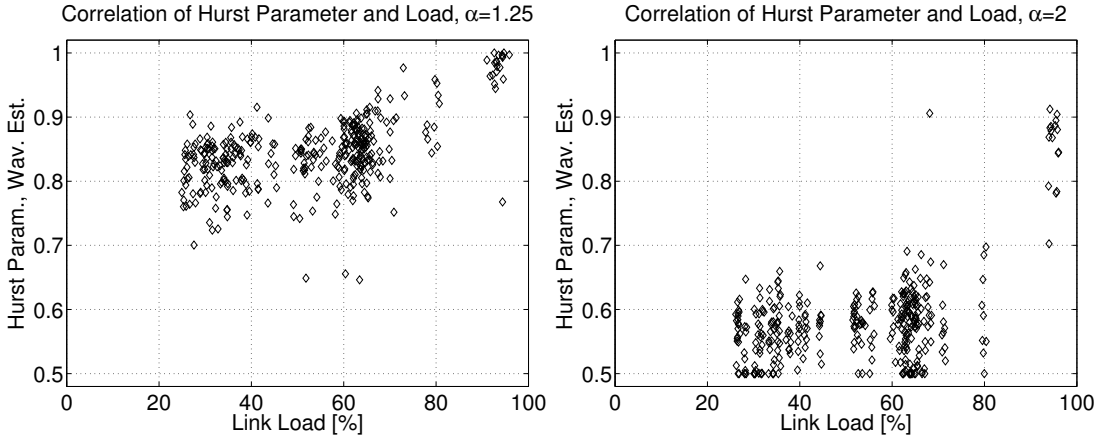


Figure 6.25: Correlation between Hurst parameter and link load, B-WiN.

($\alpha = 2$ is not possible with the SupFRP). The correlation plot in Fig. 6.26 reveals that the high Hurst parameter values for high link loads are caused by TCP: the Hurst parameter values are approximately independent of the link load for both values of α. Therefore, the correlation induced by TCP is responsible for the high Hurst parameter values for high link loads.

It can be concluded that the coexistence of very high link load and self-similarity with a high Hurst parameter is possible. The majority of the estimated Hurst parameter values

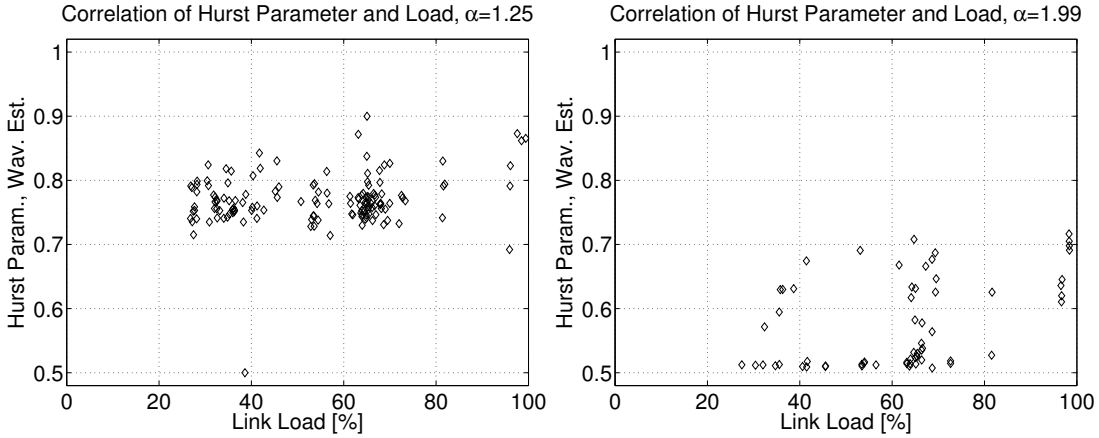


Figure 6.26: Correlation between Hurst parameter and link load with UDP traffic, B-WiN.

is smaller than 0.65 for $\alpha = 2$ for all three simulation scenarios for low link load values. The estimated Hurst parameter reaches values up to 0.92 for high link load values with $\alpha = 2$. The long-range dependence of the traffic, visible in high Hurst parameter values, is embedded in the dips and not in the peaks as usually reported.

However, the relevance of this form of self-similarity for the Quality of Service (QoS) is rather low: the dips do not cause any additional delay or packet loss, the dips only limit the maximum throughput on the link. Therefore, it can be concluded that it is not enough to do a Hurst estimation and use this value as the degree of long-range correlation for QoS issues, since this special type of long-range correlation has no impact on the QoS. Moreover, the source of the high Hurst parameter value must be known, when the impact of self-similarity is judged.

6.5 Reducing the Simulation Complexity

The impact of reducing the complexity of the simulations by reducing the number of clients is evaluated here. The average traffic load is kept approximately constant by increasing the activity of the clients via a reduction of the off-time. Efficiency is gained for the simulations by scaling down the number of clients, as discussed in Sec. 2.1. Mainly two parameters are affected: the required memory and the simulation speed. Simulations are performed for each of the combinations of off-time $t_{off} = \{1, 2, 5, 10, 20, 30, 40\}$ and $\alpha = \{1.3, 1.5, 2\}$ with several different seeds of the random number generator.

A formula for the average number of active connections is derived in Sec. 6.5.1. The efficiency of the simulations in terms of memory requirements and simulation speed is discussed in Sec. 6.5.2. Further, the changes associated with the reduction of the number of clients are evaluated based on measurements of the following parameters: the average traffic load (Sec. 6.5.3), the coefficient of variation of the inter-arrival times (Sec. 6.5.5), the Hurst parameter (Sec. 6.5.6) and the average end-to-end delay (Sec. 6.5.7).

The discussion presented in the following focuses on the measurements of selected links or flows, showing characteristic behaviour for other measurements not shown here. Most figures show measurements for $\alpha = 1.5$ because this leads to a degree of self-similarity of $H \approx 0.75$ which is a realistic value [LTWW93]. Figures for $\alpha = 1.3$ or $\alpha = 2.0$ are only shown when they show unexpected or very interesting results.

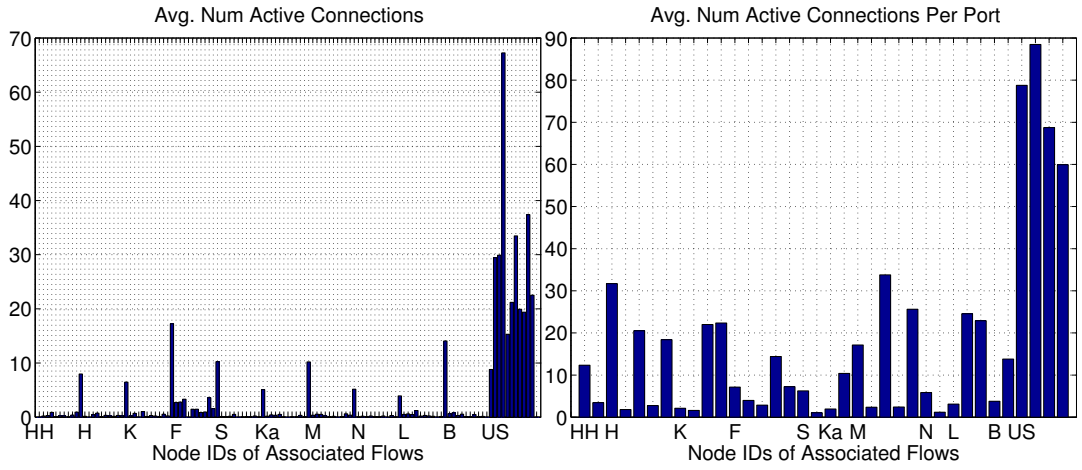
6.5.1 Average Number of Active Connections

The average number of active connections can be calculated by multiplying the number of connections (cf. Eq. (4.14)) with the activity ratio (on-time divided by on- plus off-time, see Eq. (6.17)). The average number of active connections is approximately independent of the off-time and therefore remains constant when the off-time is reduced:

$$\begin{aligned}
\overline{s_{ij,active}} &= s_{ij} \cdot \frac{t_{on,ij}}{t_{on,ij} + t_{off}} \\
&= round\left(\frac{tp_{ij} \cdot \left(t_{on,ij} + t_{off}\right)}{v+o} \cdot \left(1 - \frac{n_{Req} \cdot s_{Req} + o}{v+o}\right)\right) \cdot \frac{t_{on,ij}}{t_{on,ij} + t_{off}} \\
&\approx \frac{tp_{ij} \cdot t_{on,ij}}{v+o} \cdot \left(1 - \frac{n_{Req} \cdot s_{Req} + o}{v+o}\right).
\end{aligned} \tag{6.17}$$

The average number of active connections is shown in Fig. 6.27 (a) for all flows and in Fig. 6.27 (b) for all links in the B-WiN scenario. The graphs visualise that the average number of active connections is also very non-uniform as could already be expected from the non-uniform traffic matrix. Most active connections produce dataflow from node “US” to all other nodes. The average number of active connections of several links within Germany is smaller than 10. This is one reason why the statistical traffic parameters cover a broad range of values for all links in the B-WiN, like e.g. the coefficient of variation of packet inter-arrival times.

The effect of rounding on the average number of active connections is visualised in Fig. 6.28 for different off-time values. The result of rounding is an oscillation around the solution without rounding and the impact is very large for a small target throughput value (0.44 Mbit/s in Fig. 6.28 (a)). The oscillation amplitude is negligible for $t_{off} \geq 20$ s and increases for small values of the off-time.



(a) (b)

Figure 6.27: Average number of active connections (a) for all flows and (b) for all links for the B-WiN scenario, $v = 60$ KB.

The oscillation is still apparent for a target throughput of 10 Mbit/s in Fig. 6.28 (b) or 126.78 Mbit/s in Fig. 6.28 (c), but the impact is much lower since the offset (average value) is much higher. The two extreme cases of 0.44 Mbit/s and 126.78 Mbit/s are the minimum and maximum throughput values measured in the B-WiN, respectively. However, the oscillation amplitude is always the same, as can be seen in Fig. 6.28 (d) for 126.78 Mbit/s. But the constant oscillation amplitude becomes negligible when the average value is large enough. Therefore, the selected range for the ordinate in Fig. 6.28 (a)–(c) is appropriate for judging the effect of the oscillations on the accuracy.

The rounding is a result of the fact that only an integer number of client models can be connected to the network. Therefore, the accuracy of reaching the target throughput value depends on the following parameters:

- The rounding leads to errors if the number of clients s_{ij} in one flow is small, e.g. $s_{ij} = 10$ results in a maximum rounding error of ± 5 %.
- The rounding results in errors in the average number of active connections if the target throughput and the off-time are small. The error can be significant for very small target throughput values (cf. Fig. 6.28 (a)).
- The variability of the traffic is likely to be higher if the average number of simultaneously active connections is small. On the other hand, solutions with on average more than e.g. $\overline{s_{ij,active}} \geq 5$ active connections per link are expected to produce more stable results (see Fig. 6.28 (b)). However, the average number of active connections in on several links of the B-WiN smaller than 5, cf. Fig. 6.27 (b).

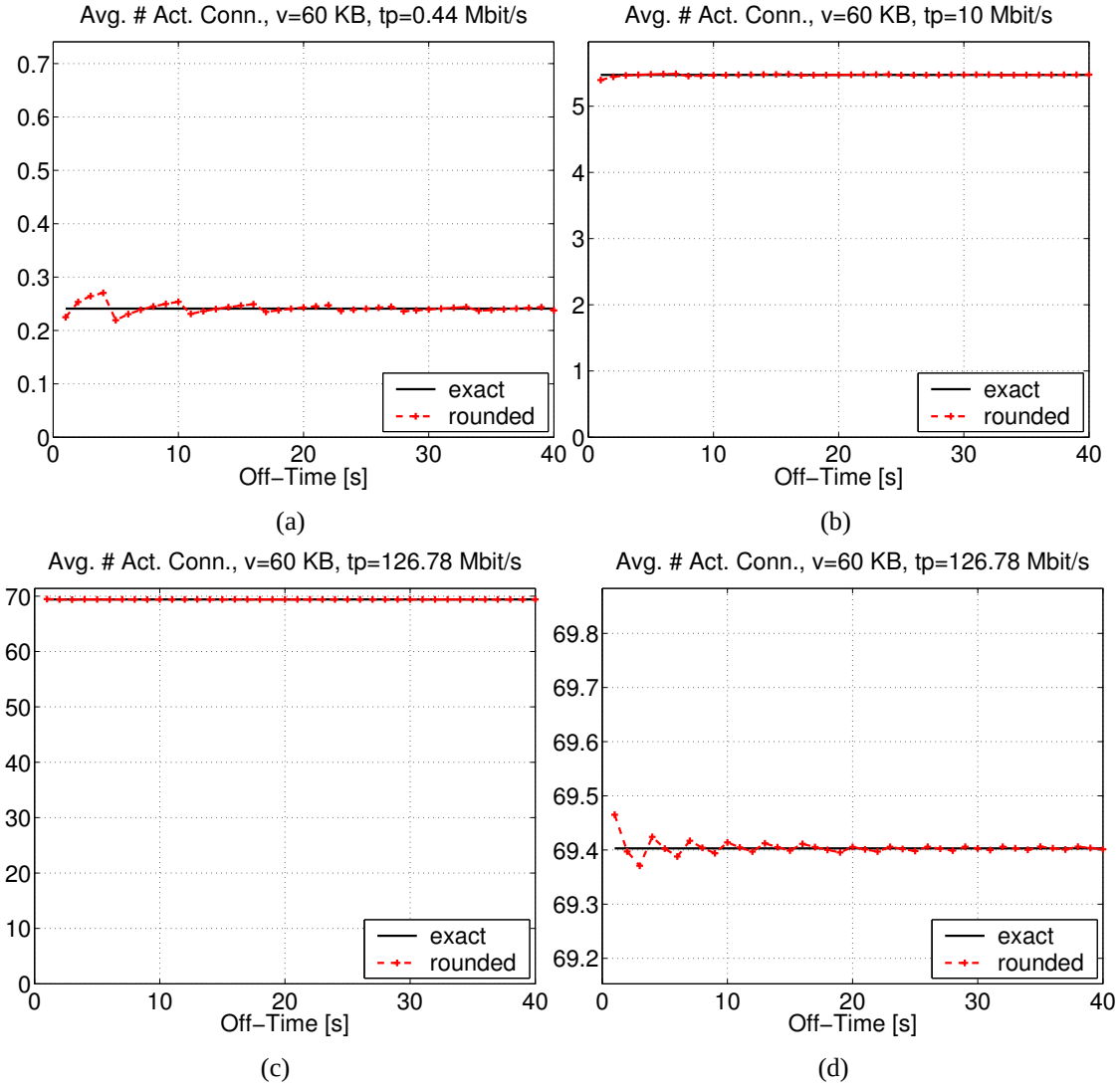


Figure 6.28: Average number of active connections for different target throughput values and $v = 60$ KB.

A threshold can be defined as a conclusion of the discussion above for an accurate modelling of all flows: the limiting case for reducing the number of clients by reducing the off-time is reached, when the flow with the smallest number of clients reaches the threshold of $s_{ij} \geq 10$. The rounding leads to a significant error for the number of clients s_{ij} for this flow if the condition is not fulfilled. Since the average number of active connections is approximately constant (neglecting from the rounding artefacts discussed above), it can be expected that the first moments of the measured parameters are also approximately constant.

The threshold of $s_{ij} \leq 10$ clients is reached for one of the 110 flows of the B-WiN at $t_{off} = 10$ s. Further, already 7 flows have less than 10 clients for $t_{off} = 5$ s. However, it is not likely that one of 110 flows changes the overall behaviour, but it can be expected that for $t_{off} \leq 5$ s the network behaviour starts to change since more than 6 % of the flows are below the threshold in this case.

A direct conclusion from the discussion above is that the accuracy depends strongly on the smallest values of tp_{ij} (see Eq. (4.14)) and therefore a more uniform traffic matrix would allow a larger reduction of the number of clients than the completely non-uniform throughput matrix of the B-WiN (see Fig. 5.21). Therefore, it is clear that also the accuracy of the simulation results shown here are bound by the given traffic matrix and the rounding. The deviations are not necessarily an indication of inaccuracy of the algorithm for allocation of clients in Sec. 4.1.1.

The basis for consistent measurements with a reduced number of clients is that the average traffic load is constant, independent of the number of clients, as discussed in Sec. 6.2. Otherwise, it can not be expected to see consistent measurements of e.g. the average end-to-end delay for a reduced set of clients. The quality of the match depends on the accuracy of the estimation of the on-time $t_{on,ij}$. The relation between on- and off-time is also very important, because the sum determines the accuracy of the estimated throughput in Eq. (4.1): an estimation error in the on-time results only in a small error in the estimated throughput if the off-time is very large and vice versa.

6.5.2 Simulation Efficiency

The relationship between off-time, total number of clients and the measured memory required by the simulator is displayed in Tab. 6.5. The memory savings for smaller off-times are of major importance since the current 32-bit computer systems do not allow processes to use a larger address space than 3 GB. The theoretical limit of the address space of 32-bit systems is 4 GB, but Linux and Windows reserve 1 GB for the kernel [LML03, MSM03].

The empirical upper bound on the required memory for the simulations in Tab. 6.5 was found to be a linear relation Eq. (6.18). The simulation core allocates about 10 MB and each client requires approximately additional 20 KB:

$$M_{req}\,[MB] \leq N_{clients} \cdot 0.02 + 10. \tag{6.18}$$

The maximum gain in simulation speed is smaller than 50 % (in this implementation), which is a rather small profit compared to the reduction of the memory usage. This can be explained as follows: keeping the traffic load approximately constant means that the number of generated events in the simulator is also approximately constant. Event-driven

Table 6.5: Total number of clients and measured memory requirements as a function of the off-time.

Off-time [s]	# Clients	RAM [MB]
40	108139	2090
30	81212	1591
20	54279	1027
10	27346	536
5	13893	278
2	5819	123
1	3119	73

simulators spend a significant amount of time for scheduling and sorting the generated events. Therefore, the simulation speed remains approximately constant, too. It is likely that the observed increase in speed is a result of smaller operating system overhead (memory allocation and deallocation) and more cache hits in the CPU caches for the processes with lower memory utilisation.

6.5.3 Average Link Load

The average traffic load measured in the simulations with decreasing client population is compared with the target traffic load in this section (see also Sec. 5.6.3). The target load value is calculated as the sum of all measured flow throughput values from Fig. 5.21 that are routed over the corresponding link, divided by the link capacity. Although the estimation of the required number of clients based on Eq. (4.1) tries to keep the load constant at the measured values (Fig. 5.21), some deviations from the ideal behaviour can be observed.

The error in matching the link load consists of the sum of the errors of all flows traversing the link under consideration. Therefore, it depends on the sign of the error values and their distribution whether the error for the link is larger or smaller than the error per flow. However, it can be expected that the error is large for the case of very high link utilisation (e.g. link “US->K” with $\rho \approx 98.4$ %), since the throughput of the flows measured in the simulation is in this case always smaller than the estimated throughput (sum of error values with the same sign).

Some characteristic measurements of the average link load for $\alpha = 1.5$ are depicted in Fig. 6.29. The first three graphs (a)–(c) show a good match with the target load (dashed line) for $t_{off} \geq 5$ s with less than 5 % error. The link “US->K” has a very high target load of $\rho = 98.4$ %, the measured link load is shown in Fig. 6.29 (d). It is harder to give

a good estimate on the number of clients required to produce this traffic load, since the throughput depends on parameters that are highly non-linear with respect to very high traffic loads. The error is nevertheless smaller than 5 % for $t_{off} \geq 5$ s, but the target load is never reached, as opposed to the other three cases.

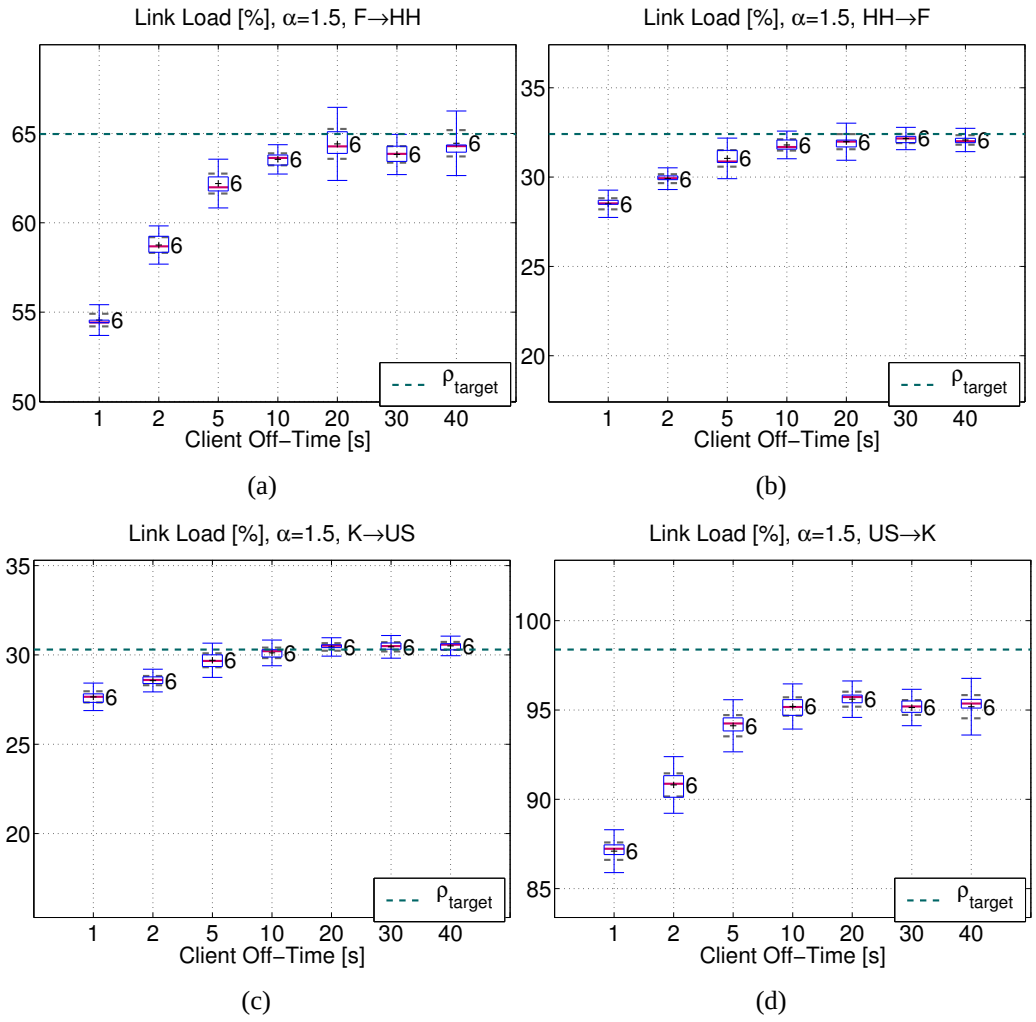


Figure 6.29: Average traffic load in dependence on off-time, $\alpha = 1.5$.

All four curves have in common that the average link load decreases with decreasing values of the off-time after a threshold of approximately $t_{off} = 5$ s is reached. This effect is more pronounced when the target load is high (Fig. 6.29 (a) and (d)). The reason for this behaviour is that the throughput degradation is larger with fewer connections when a packet loss or a time-out occurs. The rounding effects discussed in Sec. 6.5.1 are not responsible for this error: the average number of active connections is larger than 15 for all flows originating at node “US” (cf. Fig. 6.27).

The measurements in Fig. 6.30 at link “US->M” and “US->L” with a target load of 81.6 % and 72.2 %, respectively, show a significantly better result: the average traffic load is noticeably closer to the target value as compared to Fig. 6.29 (d). This is an indication that the problem is not present for normal load situations; it is only related to overload situations with very high link load values ($\rho > 90$ %).

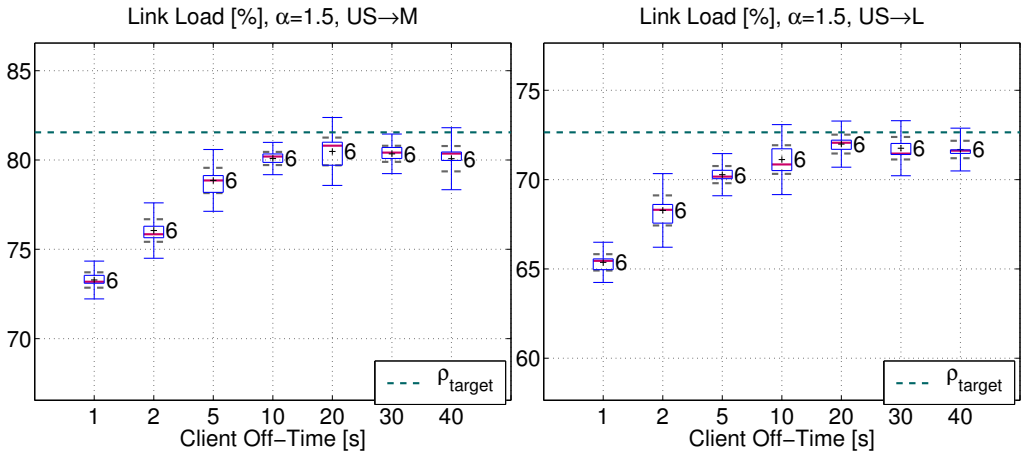


Figure 6.30: Average traffic load in dependence on off-time, $\alpha = 1.5$.

The measurements depicted in Fig. 6.31 indicate that the value of $\alpha = 1.3$ is associated with larger variations in the measurements due to the stronger tail in the power-tail distribution as compared to $\alpha = 1.5$. The deviations from the target load are larger in this case. Nevertheless, the average values differ also in this case by less than 5 % from the target values for $t_{off} \geq 5$ s.

The traffic is less bursty and correlated for $\alpha = 2$, leading to a smaller variance in the measurements and also to slightly smaller deviations from the target utilisation, as can be seen comparing Fig. 6.32 with Fig. 6.29. However, the measurements for $t_{off} = \{1, 2\}$ still deviate significantly from the target throughput.

The question whether the problem of the throughput degradation at link “US->K” for small off-time values could be solved with the TCP-modified Engset model (see Sec. 3.4) is discussed in the following. Unfortunately, this model provides no solution for the number of required clients for a given throughput (inverse problem). Therefore, the number of clients used for simulation is used as input for TCP-modified Engset model and the resulting link load estimation is given in Tab. 6.6.

The comparison with the link load results of the simulation ρ_{sim} reveals that the TCP-modified Engset model points into the wrong direction: the estimated link load ρ_{Eng} is

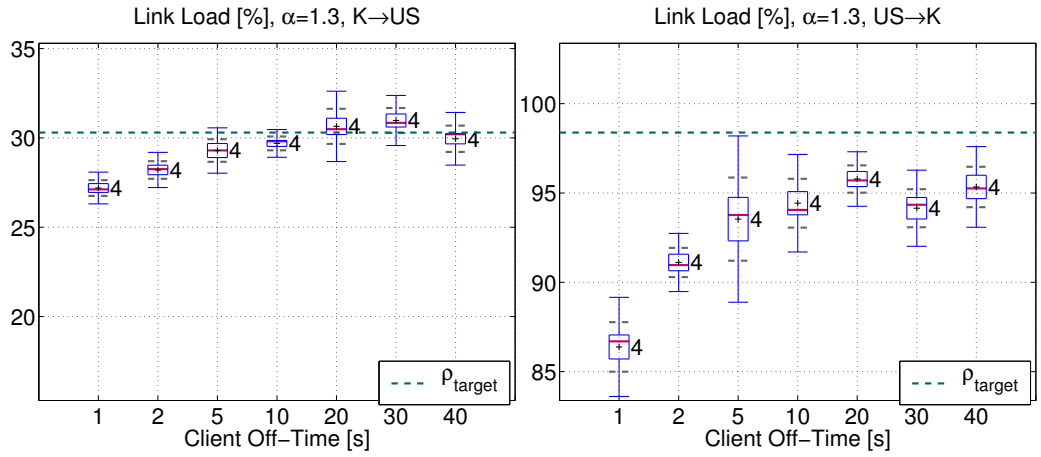


Figure 6.31: Average traffic load in dependence on off-time, $\alpha = 1.3$.

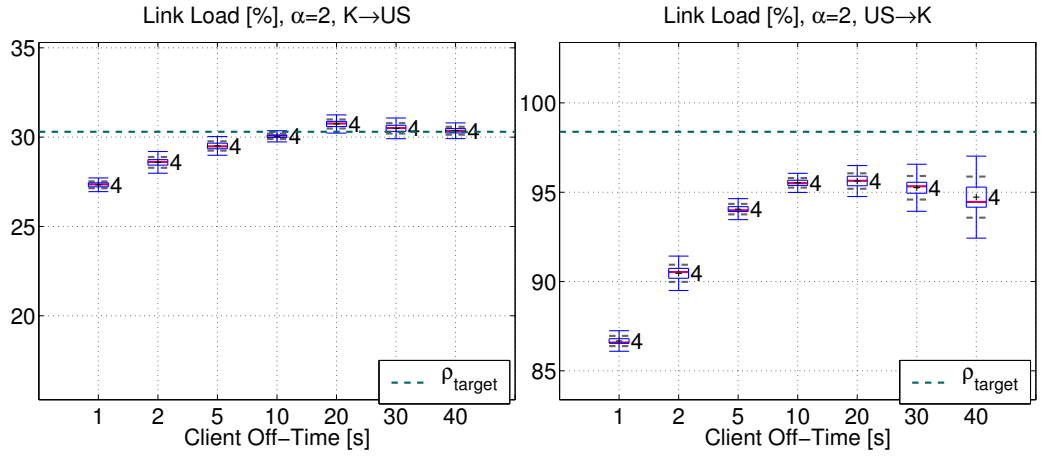


Figure 6.32: Average traffic load in dependence on off-time, $\alpha = 2$.

increasing for smaller off-time values with the given number of clients, while the simulation results show that the average load ρ_{sim} is decreasing in fact. For the case of $t_{off} = 1$ s, the TCP-modified Engset model estimates that only $N = 378$ clients would be required for a link load of $\rho = 98.4$ % even though the simulation shows that $N = 412$ clients can produce a load of only $\rho_{sim} = 87.1$ %.

Therefore, it can be concluded that the solutions of the TCP-modified Engset model can not solve the problem of estimating the correct number of clients for very high link load values. Further, the algorithm for allocation of clients presented in this work provides

Table 6.6: Average link load, simulation result ρ_{sim} and TCP-modified Engset model ρ_{Eng} in dependence on the number of clients N, link "US->K", $v = 60$ KB, $\alpha = 1.5$, $c = 10$ Mbit/s, $C = 167$ Mbit/s, $\rho_{target} = 98.4$ % and $RTT_{min} = 39.07$ ms.

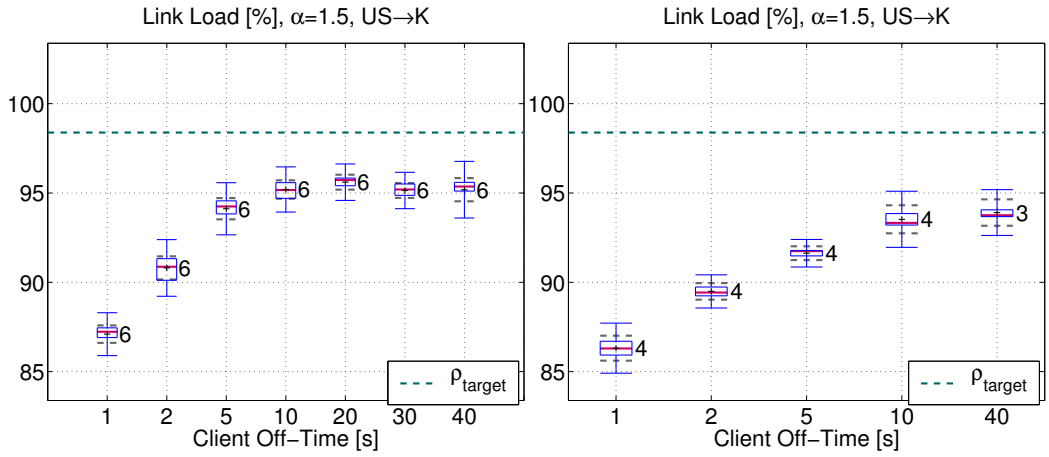
t_{off}	N	ρ_{sim}	ρ_{Eng}
40	13029	95.19	93.44
20	6558	95.61	93.87
10	3324	95.20	94.73
5	1706	94.12	96.22
2	735	90.80	98.97
1	412	87.10	99.95

better solutions than the TCP-modified Engset model. The reason for this result is that the algorithm for allocation of clients considers the slow-start behaviour of TCP more accurately.

The influence of the buffer capacity, the simulation time and the utilisation of the ceiling function instead of rounding in Eq. (4.14) for determining the number of clients is visualised for link "US->K" in Fig. 6.33. The original behaviour is shown again in Fig. 6.33 (a). The Figures 6.33 (b)–(d) show simulation results for a reduced set of off-times.

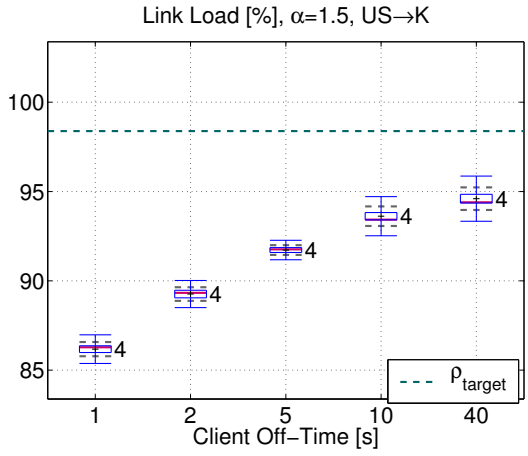
The comparison of 6.33 (a) and (b) shows the expected behaviour: the link load decreases with a smaller buffer capacity due to a higher loss probability. The results from longer simulation runs in 6.33 (c) show slightly higher link load values than for the shorter simulations. This is an indication that the simulation was already considerably close to steady state for $t = 700$ s, since the differences are quite small. Using the ceiling function instead of rounding to determine the number of clients results also in slightly higher link utilisation values, as can be observed in Fig. 6.33 (d). However, the figures show that the problem is not caused by too short simulations or the problem of rounding. It is rather a problem related to the very high target link load.

It can be concluded that the average traffic load is approximately constant for $t_{off} \geq 5$ s with only few exceptions, where the target link load is higher than 90 %. The utilisation matches almost the measurements in the real world network. Thus, a reduction of the number of modelled clients by reducing the off-time from 40 s to 5 s is feasible in order to reduce memory requirements and simulation time. The result is a reduction of the total number of clients and of the memory requirements by a factor of approximately 8; the simulation speed increases by approx. 33 % at the same time.

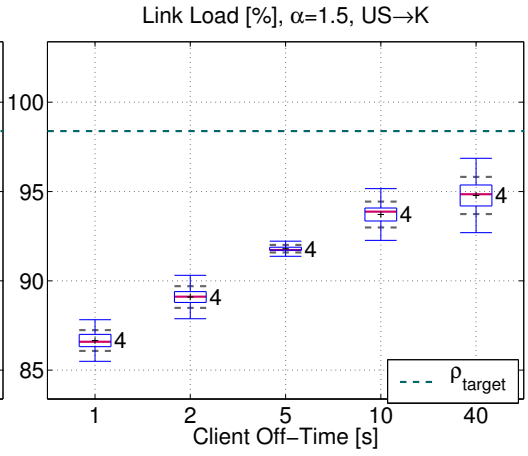


(a) $t = 700$ s, $B = 2500$, round()

(b) $t = 700$ s, $B = 500$, round()



(c) $t = 1100$ s, $B = 500$, round()



(d) $t = 1100$ s, $B = 500$, ceil()

Figure 6.33: Average traffic load on link “US->K” for different simulation time values t, buffer capacities B; (a)–(c): rounding and (d): ceiling function used for determining the number of clients.

6.5.4 Packet Loss Probability

The packet loss probability with a buffer capacity of 2500 packets was in all cases below 0.1 %, even for the links with high utilisation over 95 %. Therefore, no figures are shown for this parameter, since the differences are marginal with respect to the region of packet loss probability where the throughput of TCP connections is affected significantly.

A set of simulations with varying buffer capacity was carried out in order to evaluate the sensitivity of the TCP throughput of short-lived connections to packet losses, as shown in Fig. 6.34. The absolute error measure $\Delta_{sum,abs}$ in Fig. 6.34 (a) and the maximum ab-

solute error $\Delta_{max,abs}$ in Fig. 6.34 (b) are approximately constant if the buffer capacity is at least 500 packets. The average value of the maximum loss probability for different seeds reaches 0.66 % for a capacity of 250 packets. The error exceeds the thresholds of $\Delta_{sum,abs} \leq 5$ % and $\Delta_{max,abs} \leq 10$ % only for a buffer capacity of $B = 100$ packets which is associated with a loss probability of approx. 1.25 %. Therefore, it can be concluded that the algorithm for allocation of clients reaches a sufficient accuracy up to a packet loss probability of at least 0.66 %.

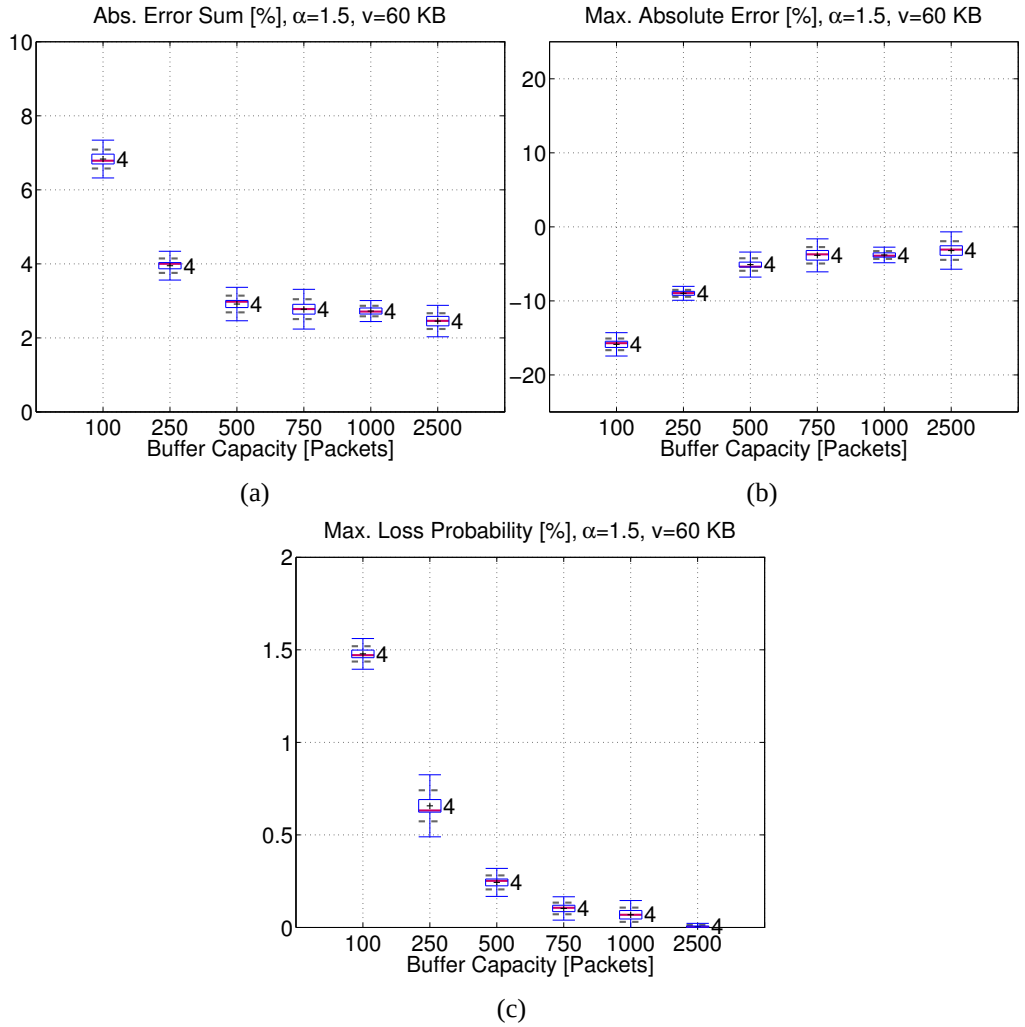


Figure 6.34: Error measures and packet loss probability in dependence on the buffer capacity, $\alpha = 1.5$, $t_{off} = 10$ s.

A subset of the simulations with variable off-times was performed also for a buffer capacity of $B = 500$ packets, as shown in Fig. 6.35. The error thresholds of $\Delta_{sum,abs} \leq 5$ %

and $\Delta_{max,abs} \leq 10$ % are also not exceeded for $t_{off} \geq 5$ s, even though the loss probability does depend on the off-time, or better on the number of clients in the network, see Fig. 6.35 (c).

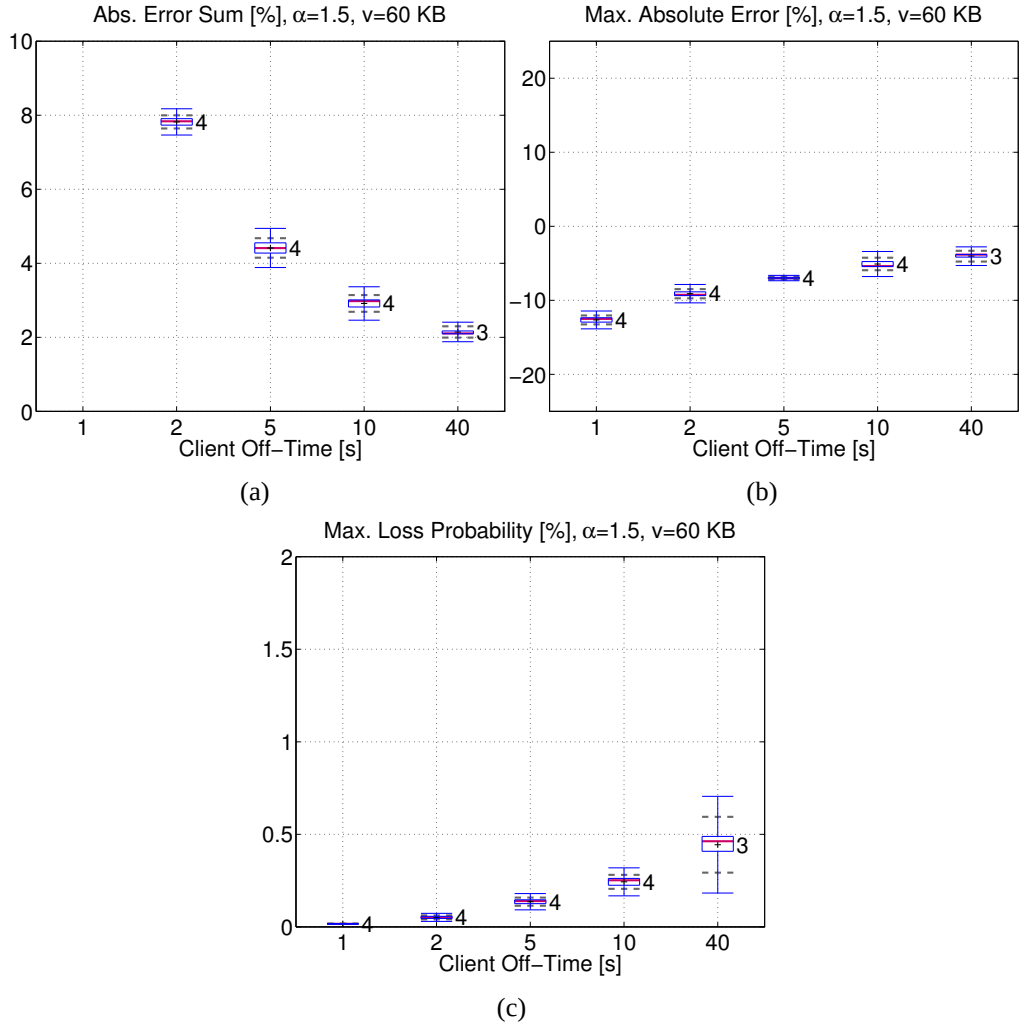


Figure 6.35: Error measures and packet loss probability in dependence on the off-time, buffer capacity 500 packets, $\alpha = 1.5$.

The reason for the dependency of the packet loss probability is given as follows: the loss probability in a system with temporal overload depends on the burstiness of the traffic. The burstiness, however, is no parameter associated with average values. The average number of simultaneously active connections is the same for all off-time values, see Sec. 6.5.1. But the losses are likely to occur when the number of active connections is temporarily significantly larger than the average. Packet losses are related to the variance and tail probabilities in the number of active connections, which are surely not constant

when reducing the total number of clients. Therefore, it can unfortunately not be expected that the loss probability of a simulation with a reduced set of clients is consistent with the original set of clients.

Quantitative results are added in the following to the qualitative reasoning to explain the effect in more detail. Suppose that the number of active connections follow a Bernoulli process: each client draws a random number and the value determines whether or not the client is active in this slot. This is a simplified model that does not model the real situation closely. However, the same effect can be shown qualitatively with this simple model. The average number of active clients is

$$\mu = N \cdot p, \tag{6.19}$$

with the total number of clients N and the probability that the client becomes active p. Further, the variance of the number of active connections is

$$\sigma^2 = N \cdot p \cdot (1-p). \tag{6.20}$$

The coefficient of variation of the number of active clients can be specified with Eq. (6.19) and (6.20) as

$$cv = \frac{\sigma}{\mu} = \frac{\sqrt{N \cdot p \cdot (1-p)}}{N \cdot p}. \tag{6.21}$$

The fact that the average number of active connections is constant for all considered cases (cf. Sec. 6.5.1) leads with $p = \mu/N$ and some transformations to

$$cv = \sqrt{\frac{N-\mu}{N \cdot \mu}}. \tag{6.22}$$

The values of *cv* are listed for the link “US->K” (link with highest load and loss probability) in Tab. 6.7. The last column contains the deviation of the coefficient of variation compared to the value associated with $t_{off} = 40$ s in percent. It is obvious, that the coefficient of variation of the number of active clients decreases with N. The tendency is correct, but the change in the values is rather small, compared to the change of the loss probability in Fig. 6.35. However, the same evaluation with the TCP-modified Engset model provides worse estimates (not shown here) and the qualitative behaviour is captured correctly with the Bernoulli model.

However, if the simulation scenario is too complex to use a realistic set of clients, errors due to approximations have to be accepted. The strategy behind the reduction of clients used in this work leads to consistent average values for all measured parameters, except of the loss probability. Unfortunately the loss probability does not depend on those average values but rather on higher moments, which are not captured correctly by the reduced set of clients. However, also aggregated traffic models can not promise more consistent loss probability values that match measurements in real networks, as shown in Sec. 2.1.

Table 6.7: Coefficient of variation *cv* of number of active clients Bernoulli process in dependence on the number of clients N, avg. number of active clients $\mu = 88.48$, δ: deviation from *cv* at $t_{off} = 40$ s in percent, link "US->K".

t_{off} [s]	N	*cv*	δ [%]
40	13029	0.1059	–
20	6558	0.1056	0.34
10	3324	0.1049	1.00
5	1706	0.1035	2.30
2	735	0.0997	5.89
1	412	0.0942	11.08

6.5.5 Coefficient of Variation

The coefficient of variation is defined as $cv = \sigma/\mu$, with standard deviation σ and mean value μ. The coefficient of variation is a measure of the relative dispersion. It is used here to evaluate the variability of the packet inter-arrival times. The inter-arrival times are measured at the input of the queueing module that is used to limit the bandwidth of the links. The queueing module is located at every outgoing port of the router.

The behaviour of the coefficient of variation for different off-times (number of clients) is shown in Fig. 6.36 for $\alpha = 1.5$. All four graphs have in common that the values show a very small variance, which is reflected by boxplots with a very small height. Furthermore, the coefficient of variation is in all cases significantly larger than one, indicating a larger variability as compared to Poisson traffic. The first two graphs, Fig. 6.36 (a) and (b), represent measurements corresponding to two links where the coefficient of variation is approximately constant, independent of the off-time. Also the absolute value is approximately the same for those two links, although the utilisation is $\rho = 95.6$ % in Fig. 6.36 (a) and $\rho = 38.6$ % in Fig. 6.36 (b). Figure 6.36 (c) on the other hand has a similar traffic load as compared to Fig. 6.36 (b) but the coefficient of variation is larger and is only approx. constant for $t_{off} \geq 5$ s.

This phenomenon can be explained as follows: Fig. 6.36 (b) corresponds to the link from node N to node M. On the backward path, from node M to node N, the utilisation is higher ($\rho \approx 63.9$) and some flows from node US are also routed over the backward link. Therefore, the traffic on link "N->M" is shaped by the other flows on the backward path, as opposed to the case in Fig. 6.36 (c). The first three figures show approx. constant graphs, or graphs where the value of *cv* decreases with smaller off-time values. Figure 6.36 (d) shows a slightly increasing coefficient of variation with decreasing off-time. However, the differences are very small and not significant.

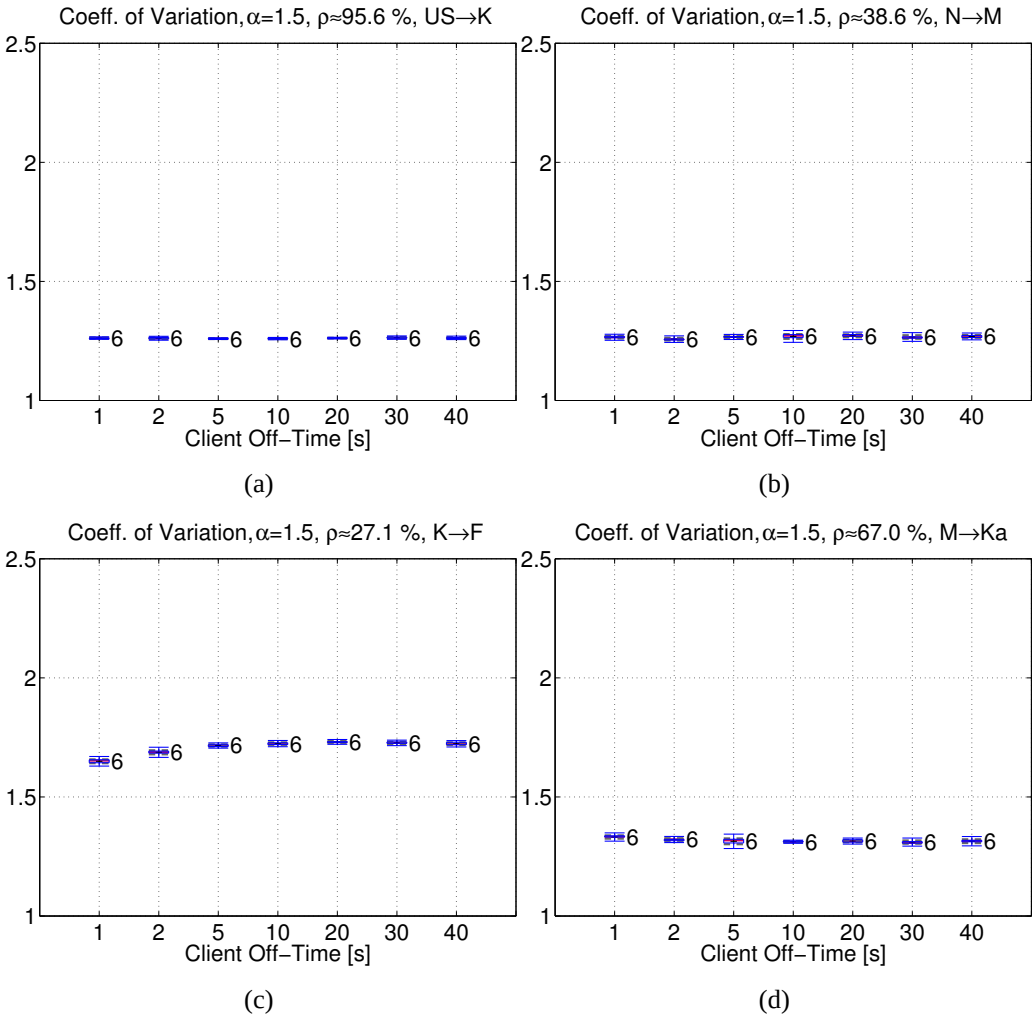


Figure 6.36: Coefficient of Variation over off-time, $\alpha = 1.5$.

It can be concluded that the coefficient of variation of the inter-arrival times is nearly not affected by changing the number of clients in the system, the changes are very small (less than 5 % for all off-time values) and the absolute values are in ranges confirmed by recent measurements (see Sec. 6.4.1).

6.5.6 Hurst Parameter

The Hurst parameter values estimated from the byte counting process are subject to large variations for different seeds, as can be observed for $\alpha = 1.5$ in Fig. 6.37. The Hurst parameter of the traffic on link “F->HH” depicted in Fig. 6.37 (a) shows the expected

behaviour: the average value is slightly lower than 0.75, which is the theoretical value (cf. Sec. 4.2). The average value is changing with the off-time but the change is not large with respect to the estimation accuracy.

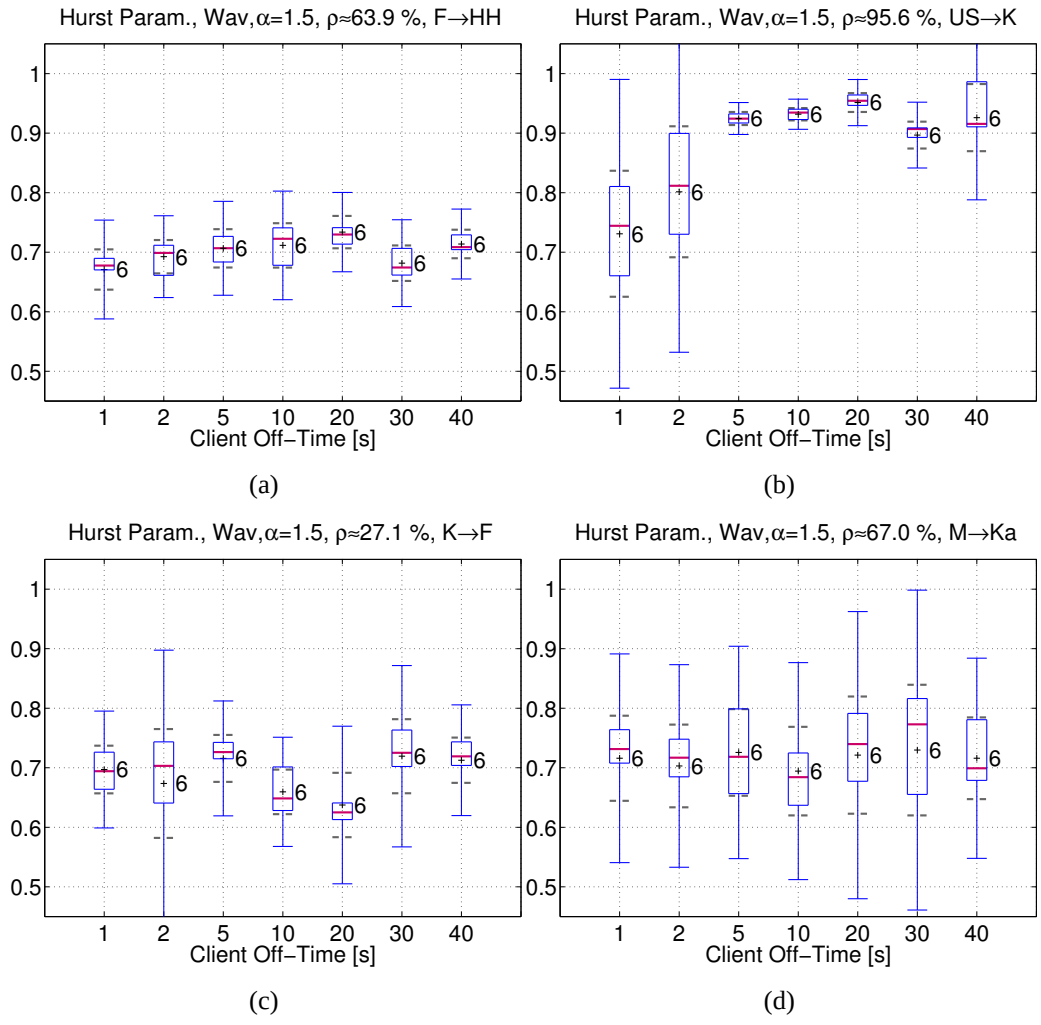


Figure 6.37: Hurst parameter over off-time, $\alpha = 1.5$.

The Hurst parameter values on link “US->K” shown in Fig. 6.37 (b) have a more or less consistent, but higher average value for $t_{off} \geq 5$ s (see discussion about high Hurst parameter at high link utilisation in Sec. 6.4.3). The inconsistency for $t_{off} < 5$ s could be caused by the fact that also the average link load is not constant in this range (cf. Sec. 6.5.3).

Figure 6.37 (c) and (d) indicate a high variability of the average Hurst parameter for different seeds (large confidence intervals) and for different off-time values. This be-

haviour is independent of the utilisation as shown with Fig. 6.37 (c) with $\rho \approx 27.1$ % and (d) with $\rho \approx 67$ %. However, there is no evidence that this behaviour is correlated with the off-time, it seems to be rather a sign of an intrinsic high variability.

6.5.7 Average End-To-End Delay

The average end-to-end delay is a very important measure since it has a major impact on the user perceived Quality of Service (QoS). The average end-to-end delay of the data packets was measured and averaged over all clients in the same flow, i.e. clients with same source and destination node (not over all connections that share one link, as the other measures before). The end-to-end delay was measured on different layers, the lowest layer (IP) measurements are shown here. The average end-to-end delay depends strongly on the average queue size and therefore also strongly on the average link load. It can be expected that the average end-to-end delay is not constant in cases where the link load is not constant for different off-times (cf. Sec. 6.5.3).

The average end-to-end delay of flow “K->US” is approximately constant for all off-time values, as can be seen in Fig. 6.38 (a). The utilisation $\rho \approx 30.3$ % denoted in the title of the graph refers to the maximum utilisation on all links that the packets follow on the route from source “K” node to destination node “US”. The measurements on many other flows show very similar characteristic.

However, the measurements in the opposite direction with a maximum utilisation of 95.6 % (flow “US->K”, Fig. 6.38 (b)) show a strong dependency on off-time: the average end-to-end delay is only approximately constant for $t_{off} \geq 20$ s here. A very similar curve is shown in Fig. 6.38 (c) for the flow “US->HH” with maximum utilisation of $\rho \approx 93.4$ %. The same characteristic as in Fig. 6.38 (b) and (c) was observed for the flows “US->H”, “US->N” and “US->S”.

A completely different characteristic was measured at flow “US->M” in Fig. 6.38 (d) with maximum utilisation of 78.7 %: the end-to-end delay is nearly independent of the off-time. The same characteristic was observed for the flows “US->B”, “US->L” and “US->Ka”.

This phenomenon can be explained as follows: the two classes formed by the measurements are characterised by the fact, that they are all routed over two links “US->K” and “US->H”. Both links have a target load larger than 95 % and the matching with the target load was not very accurate (cf. Sec. 6.5.3). That is, the load was not constant for different off-times; the dependency of the link load on the off-time was significant already for $t_{off} \geq 5$ s, see Sec. 6.5.3. Therefore, the decreasing average end-to-end delay for small off-times is a direct result from the deviations from the ideal behaviour of the link load on these two links. The second set of flows (“US->M”, “US->B”, “US->L” and

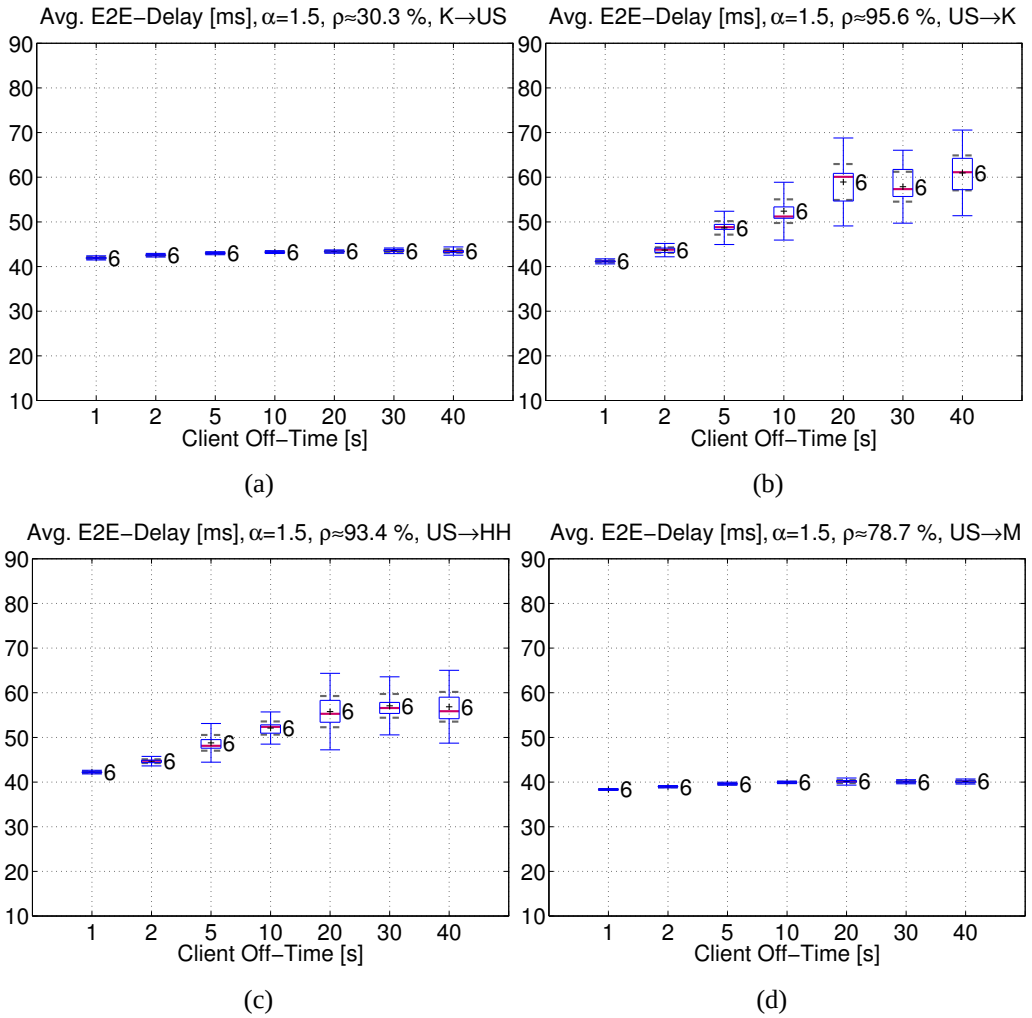


Figure 6.38: Average end-to-end delay over off-time, $\alpha = 1.5$.

"US->Ka") experiencing average end-to-end delays nearly independent of the off-time are routed over links with smaller target link load.

The measurements of some flows show an unexpected characteristic as visualised in Fig. 6.39: the average end-to-end delay decreases for large values of the off-time and therefore for larger number of clients. However, the difference is rather small as compared to e.g. Fig. 6.38. Furthermore, it is not critical when the end-to-end delay is slightly larger than in reality, since this leads to a conservative decision regarding QoS dimensioning issues.

It can be summarised that the average end-to-end delay is approximately constant for $t_{off} \geq 5$ s with the exception of some flows routed over two links with a target utilisation

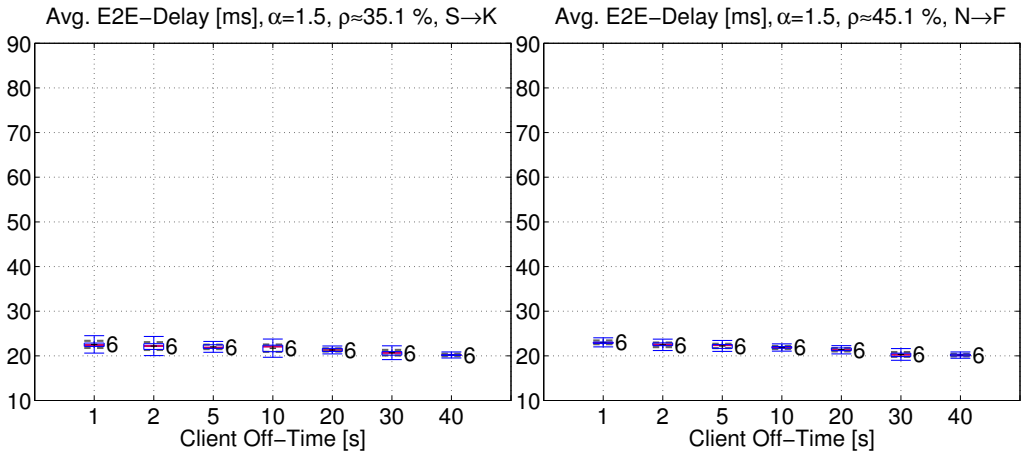


Figure 6.39: Average end-to-end delay over off-time, $\alpha = 1.5$.

over 95 %. It could be shown that this exception was caused by a problem in reaching a constant link load for the whole range of off-times, which is only apparent for very high link loads, as in this case.

6.5.8 Potential of the Complexity Reduction

The successful complexity reduction of the simulation allows to simulate even larger network models. The B-WiN model was scaled up in order to evaluate a simulation close to the physical limits of current 32 bit computer systems. All link capacities were set to 1.5 Gbit/s resulting in a total capacity of 54 Gbit/s. A uniform target traffic matrix with 140 Mbit/s was used for all flows (traffic from all nodes to all neighbour nodes). An off-time of $t_{off} = 5$ s was used for a total of $146,846$ client modules. The total average throughput was 25.96 Gbit/s (15.4 Gbit/s from all clients/webservers routed over several hops).

The resulting throughput of all flows is depicted in Fig. 6.40. The target of 140 Mbit/s was approximately reached. However, this simulation really touched the limits: the simulation required 2853 MB RAM and the simulation of 20 s real time lasted for 15.7 days and nights on a Pentium © III with 1.4 GHz. This is about 107 times slower than a B-WiN simulation with the standard settings for $t_{off} = 40$ s with $108,000$ clients (and a total throughput of only 2.17 Gbit/s); the simulation of 700 s real time took 4.3 days and nights in this case (on the same computer). Further, the simulation of the B-WiN with standard settings and $t_{off} = 5$ s was already finished after 3.2 days and nights. These numbers make clear that the simulation of network models of this size is really a com-

plex task. However, the price development allows to build economically priced Linux clusters capable of performing such complex simulation tasks.

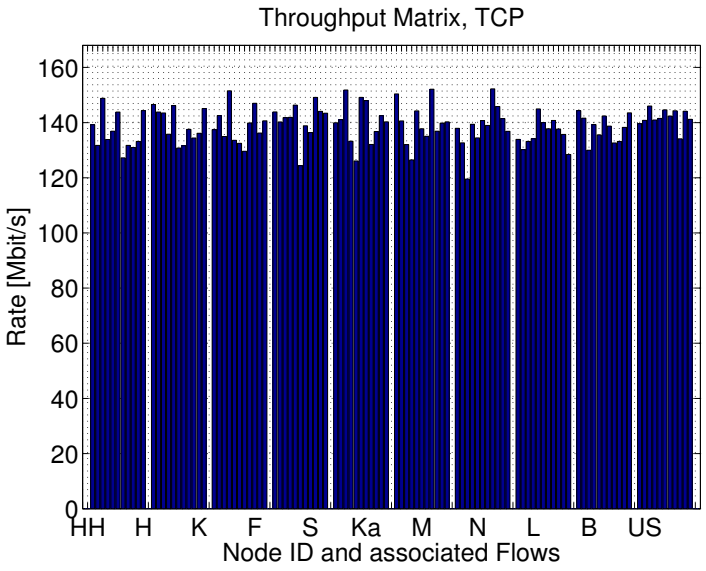


Figure 6.40: Average throughput of all flows, B-WiN with higher link capacities and uniform target traffic matrix.

6.6 Conclusion

It can be concluded that the algorithm for allocation of clients presented in Sec. 4.1 successfully estimates the required number of clients for the given throughput matrix and network scenario. The maximum error for all flows in the B-WiN is smaller than 4 % for realistic parameter settings ($t_{off} = 40$ s and $v = 60$ KB). The deviation from the target increases for smaller off-time values and larger download volumes. However, the error is in most cases still acceptable.

It has been verified with an extensive simulation study that a reduction of the number of clients by a factor of 8 was possible without significant changes of the average values of the link load, coefficient of variation, Hurst parameter and end-to-end delay.

One issue was identified: a link utilisation very close to 100 % causes problems for estimating the required number of clients, since the end-to-end delay and loss probability, and therefore also the TCP throughput, follow a non-linear behaviour in this region. The result was, that the number of clients could be only reduced by a factor of 4 in this case instead of 8 with the same error bounds.

However, link loads close to 100 % are problematic due to several reasons: the congestion avoidance mechanism of TCP reduces the throughput when faced with packet losses or time-outs. Therefore, it can not easily be identified whether the high load situation is the result of e.g. an offered traffic of approx. 100 % of the link capacity or an overload situation of 180 % of the link capacity (see Sec. 2.1, p. 6 for the definition of the degree of overload for TCP traffic). Considering this fact, it becomes clear that such an overload situation results in more degrees of freedom for the solution. Further, it is not trivial to determine the degree of overload from other measurements like the average queue size – which were not available for the B-WiN.

Average link utilisation values of more than 90 % represent a bad operating point, since they go hand in hand with bad QoS (delay and packet loss) for the user. Nevertheless, if such a situation is existing, as in the B-WiN, a solution must be found, and therefore it might be necessary to simulate this current overload situation of the network. This is possible with the given solution. However, the errors are larger in this case and it could be that the degree of overload is not captured correctly, since the corresponding measurements might be unknown, as in the case of the B-WiN.

As a result of the successful reduction of the number of clients an equivalent of 1.2 million clients can be simulated (for $t_{off} = 5$ s) instead of $150,000$ (for $t_{off} = 40$ s) for a memory limitation of 3 GB. The average throughput of a source with a small RTT in the B-WiN scenario is approximately 13.3 kbit/s for $t_{off} = 40$ s and approximately 99.1 kbit/s for $t_{off} = 5$ s. The total average throughput of a simulation with $t_{off} = 5$ s can therefore reach up to 15.4 Gbit/s as compared to a maximum of 2 Gbit/s for $t_{off} = 40$ s (cf. Sec. 6.5.8).

Chapter 7

Conclusions

The main focus of this work is the realistic simulation of communication networks. The optimisation of an existing network poses a new problem for the simulation: a realistic simulation of the current state of the network is required for the qualitative and quantitative assessment of new alternatives to the status quo. Moreover, measurements of the network provider need to be taken into account when setting up the simulation. The simulation of existing networks is an inverse problem: the network description (number of nodes, connectivity, capacities, routing) and some measurements (traffic matrix, etc.) are given from the network provider. However, the number of clients, their behaviour and distribution over the network are in most cases unknown. Therefore, the number of clients has to be derived from the network description and traffic measurements.

A methodology was developed in this work that provides a solution for the inverse problem. This methodology allows for realistic simulations of existing networks. The average traffic intensity is controlled by the iterative algorithm for allocation of clients, presented in Sec. 4.1. The iterations convergence fast but the resulting error oscillates around zero, as shown in Sec. 6.2.4. However, it has been shown that the algorithm's estimation accuracy is in most cases sufficient, so that no further iterations beyond the first simulation are required. It has been shown that the proposed algorithm yields more accurate estimations than the TCP-modified Engset model [HLN97] (cf. Sec. 3.4). Furthermore, a methodology is provided in this work, describing how higher moments of the traffic – Hurst parameter and coefficient of variation – can be matched to corresponding measurements in the real network (cf. Sec. 4.2 and 4.3).

Realistic network simulations have a high complexity, the required memory and the simulation speed are a major issue in this area. Therefore, the reduction of the complexity is treated in this work. Two alternatives for the complexity reduction have been discussed: the usage of aggregated traffic models and the reduction of the number of clients by increasing the activity of the single client. It has been shown in Sec. 2.1 that aggregated

traffic models can not be used to solve the performance problem: one of the most important features of web traffic – the reactivity of TCP – is not captured by aggregated traffic models.

Reducing the number of clients is a feasible solution for the complexity reduction, as shown in Sec. 6.5: the activity of each client can be increased by reducing the off-time so that less clients are needed to generate the same average throughput. This strategy leaves the on-time unchanged. Therefore, the complete dynamic behaviour of the client (distributions, protocols, etc.) remains the same. Furthermore, it has been shown that the average number of active connections remains constant, independent of the reduction.

An extensive simulation study was performed to validate the correctness of this approach for reducing the simulation complexity. To asses the applicability of this solution, critical network parameters are estimated as a function of the number of clients. The parameters considered are: average link load, loss probability, coefficient of variation, Hurst parameter and average end-to-end delay. It has been shown in Sec. 6.5 that the measurements are approximately constant for a wide range of off-time values. The number of clients could be reduced by a factor of $4-8$ (off-time 10 s or 5 s instead of 40 s) without having significant impact on the measured parameters. The required accuracy, the complexity and the maximum link utilisation of the network scenario are the factors determining whether a factor of 4 or 8 can be gained. The simulation speed was increased by up to 33 % when using the reduced client set ($t_{off} = 5$ s versus $t_{off} = 40$ s).

The results reveal the potential for even larger simulation models than the ones presented in this work: a maximum of $150,000$ clients can be allocated with the fixed memory limit of 3 GB of current 32-bit systems. The average throughput per client in the B-WiN scenario is approx. 13.3 kbit/s for an off-time of 40 s. This results in a total throughput of approx. 1.44 Gbit/s for the B-WiN with approx. $108,000$ clients requiring 2090 MB RAM. The potential of the complexity reduction by a factor of 8 is that a maximum of $150,000$ clients (for 3 GB RAM) can generate a traffic of up to 15.4 Gbit/s with an off-time of $t_{off} = 5$ s, an equivalent of $1.2 \cdot 10^6$ clients with $t_{off} = 40$ s, as shown in Sec. 6.5.8.

It can be concluded that this work represents a major step in research towards a realistic modelling and simulation of complex simulation models of existing networks. The solutions for matching average traffic load, coefficient of variation and Hurst parameter to measured values in the real network are promising and can be used as basis for further studies. The successful increase of the simulation efficiency gained by decreasing the client population represents one step towards the realistic simulation of current and future multi-Gbit networks.

Appendix A

Tools for Estimation and Generation of Self-Similar Traffic

One of the most powerful Hurst parameter estimators, the Abry-Veitch estimator, is described in Sec. A.1. The Perl source code for generating a synthetic self-similar series following a recursive scheme called Cantor set is presented in Sec. A.2.

A.1 Abry-Veitch Wavelet Hurst Parameter Estimator

The Abry-Veitch Hurst parameter estimator [VA99, AV98, AFTV00] is based on the discrete wavelet transform. It has been shown that this estimator outperforms most other Hurst parameter estimators in the aforementioned publications. The non-parametric Abry-Veitch estimator has a very good performance in terms of computational complexity and accuracy of the results.

The coefficients of the discrete wavelet transform $d_X(j,k)$ are used for the estimation of the Hurst parameter. The variable j represents the scaling index and k the time index. The frequency can be calculated from the scaling index with $f = 2^{-j} \cdot \nu_0$. The base frequency ν_0 depends on the so called mother-wavelet, that is selected for the wavelet transform. The mother wavelet is the core of the wavelet transform. The properties of the transform can be adapted to the requirements by selecting the appropriate mother wavelet. The Daubechies wavelets [Dau88] are used as mother wavelet here since they represent a very good compromise between time domain and frequency domain resolution for the wavelet transform. The time can be calculated from the index k with $t = 2^j \cdot k$.

The second moment of the wavelet coefficients $E\{d_X(j,k)^2\}$ is used to estimate the Hurst parameter H

$$E\{d_X(j,k)^2\} \sim 2^{j\alpha}, \qquad j \to +\infty, \tag{A.1}$$

with

$$\alpha = 2H - 1. \tag{A.2}$$

The second moment of the wavelet coefficients (the energy) is plotted on a double-logarithmic plot versus the scaling index j. A slope estimation of the part forming a straight line of the double logarithmic plot can be used to estimate the Hurst parameter with Eq. (A.2), cf. Fig. A.1. The vertical lines represent 95 % confidence intervals for the estimated second moment. Small scales j represent the short-time behaviour of the process (left hand side of Fig. A.1) and large scales j represent the long-term behaviour (right hand side of Fig. A.1).

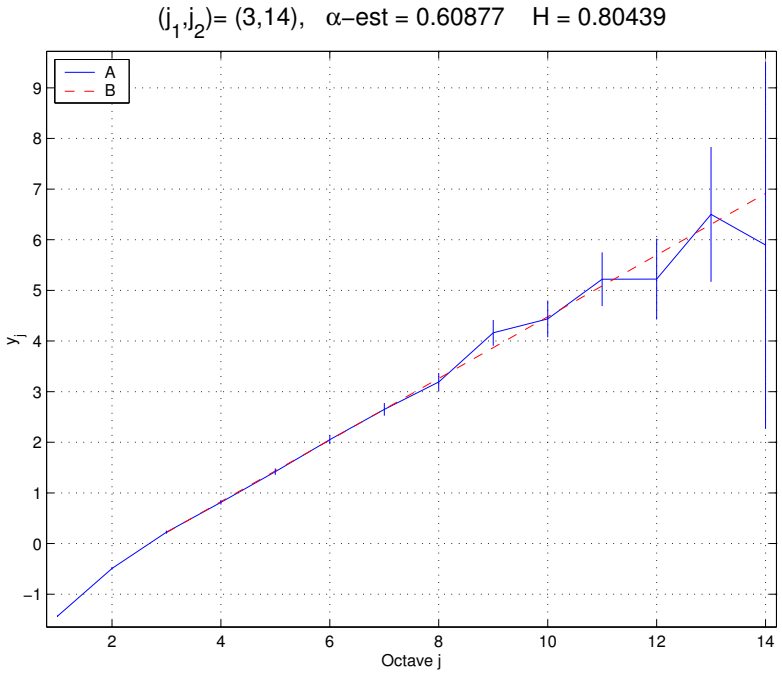


Figure A.1: Wavelet estimation of a Fractional Gaussian Noise (FGN) series, length 131072, $H = 0.8$. A: estimated second moment, B: approximated straight line for range $3 \leq j \leq 14$.

The plot of synthetic traffic with pure self-similar characteristic (like FGN in this case) exhibits an approximately linear behaviour over all scales j. Real measurements experience a constant slope usually only over a limited amount of scales. The larger scales are important when judging the Hurst parameter, the degree of long-range dependence. However, the accuracy of the estimation decreases for large values of j, as can be observer by the increasing size of the confidence intervals with increasing j in Fig. A.1. Therefore, the region used for estimating the Hurst parameter should be chosen as follows: the smallest scales should be skipped unless they have already consistent slope with the larger scales. Some few of the largest scales should be skipped, since the large confidence interval results in a weak estimation of the slope. Therefore the middle and large scales should be used for estimation of Hurst parameter (approx. rule of thumb for the range: from $\lceil j_{max}/3 \rceil$ to $j_{max} - 2$).

A.2 Source Code for Generating Cantor Set

The source code of the Perl script *syntheticTraceSelfSimHighLoad.pl* is listed in the following.

```
#!/usr/bin/perl -w

use strict;

use Getopt::Std;
&getopts('t:');
use vars qw/
    $opt_t
    /;

# vector cv with counting process, 2.5 ms, counting window, 600 seconds total
# number of samples = 600 / 2.5e-3 = 240000
my @cv = ();
my $cw = 2.5e-3;      # counting window 2.5 ms
my $st = 600;         # 600 seconds simulation time
my $ns = $st / $cw;   # number of samples = simulation time / counting window
my $lc = 167e6;       # link capacity in Mbit/s

# maximum value of counting process, capacity [Mbit/s] / 8 * count-window:
my $maxCount = $lc / 8 * $cw;

# initialize vector:
for (my $i = 0; $i < $ns; $i++) {
    $cv[$i] = $maxCount;
}

# produce a synthetic self-similar trace similar to a cantor set
```

```
# (refer to book "High-Speed Networks, TCP/IP and ATM Design Principles"
# William Stallings, Prentice Hall, 1998, ISBN 0-13-525965-7, p. 181 ff.)

my $ord = 3;
my $minInd = 0;
my $maxInd = $ns - 1;
my $recursions = 0;
my $thresh = $opt_t || 30;

&buildCantorSet(\@cv, $ord, $ns, $minInd, $maxInd, $thresh);

# build "inverse counting-process":
for (my $i = 0; $i <= $#cv; $i++) {
    $cv[$i] = $maxCount - $cv[$i];
}

# calculate load:
my $valSum = 0;
for my $val (@cv) {
    $valSum += $val;
}
my $load = 100 * $valSum / ($maxCount * $ns);

my $outfile = "syntrace" . $thresh;
open(OUT, "> $outfile") or die "\n Could not open $outfile for writing!\n";
for my $val (@cv) {
    print OUT $val, "\n";
}
close OUT;

print STDERR "\n wrote vec to file: $outfile";
printf STDERR "\n link capacity: %g Mbps, maxCount: %g, %d samples"
    , $lc / 1e6, $maxCount, $ns;
print STDERR "\n recursions $recursions, threshold: $thresh, load: $load %";

my $of = "syntrace" . $thresh . ".m";
&printMatlabScript($of);
print STDERR "\n wrote Matlab script: $of\n";

system "matlab -nodesktop -nosplash < $of; gzip -9f $outfile";

exit 0;

# --------------------

sub buildCantorSet() {
    my ($vec, $ord, $ns, $minInd, $maxInd, $thresh) = @_;

    if (($maxInd - $minInd) <= $thresh) { # finished, nothing to do anymore
return;
    }
```

```
    # zero out middle third:
    my $seg = int(($maxInd - $minInd) / $ord);
    for (my $i = $minInd + $seg; $i < ($maxInd - $seg); $i++) {
$cv[$i] = 0;
    }
    $recursions++;

    &buildCantorSet($vec, $ord, $ns, $minInd, $minInd + $seg, $thresh);
    &buildCantorSet($vec, $ord, $ns, $maxInd - $seg, $maxInd, $thresh);
}

sub printMatlabScript() {
    my ($of) = @_;

    my $loadString = sprintf("%5.2f", $load);
    open(OUT, "> $of") or die "\n Could not open $of for writing!\n";
    print OUT <<EOM

a = load('$outfile');
x = [0:$st/$ns:$st-$st/$ns];
plot(x, a / $maxCount * $lc / 1e6);
ylabel('Throughput [Mbit/s]');
xlabel('Time [s]');
title('Synth. Trace, Counting Process,  \\rho=$loadString %, Thr=$thresh');
mfont('helvetica', 21, 1);
grid on;
saveas(gcf, 'syntrace-cantor-set-self-sim-highload-seq-$thresh.fig', 'fig');
print(gcf, '-depsc2', 'syntrace-cantor-set-self-sim-highload-seq-$thresh.eps');

clf;
[Hwav, alphaest, cfCest, cfest, Cest, Qwav, j1opt] = mywavest(a);
titleString = ['Synth. Trace, Wavelet Est., H=', ...
       sprintf('%3.2f', Hwav), ', Thr=$thresh'];
title(titleString);
mfont('helvetica', 21, 1);
saveas(gcf, 'syntrace-cantor-set-self-sim-highload-$thresh.fig', 'fig');
print(gcf, '-depsc2', 'syntrace-cantor-set-self-sim-highload-$thresh.eps');

EOM
    ;
    close OUT;

}

# end ...
```

Index

Bibliography

[ABK02] D. Abendroth, K. Below, and U. Killat. The interaction between tcp and traffic shapers - clever alternatives to the leaky bucket. In *IEEE GLOBECOM 2002*, Taipei, Taiwan, November 2002. Online available: http://www.tu-harburg.de/et6/papers/documents/Abendroth_Dirk/GLOBECOM_2002.pdf.

[AFTV00] P. Abry, P. Flandrin, M. S. Taqqu, and D. Veitch. Self-similarity and long-range dependence through the wavelet lens, January 2000. Online available: http://citeseer.nj.nec.com/394097.html.

[AV98] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998. Online available: http://citeseer.nj.nec.com/abry98wavelet.html.

[Ber97a] Berkeley, University of California. *The Almagest - Volume I, Ptolemy 0.7 Users Manual*, 1997. http://ptolemy.eecs.berkeley.edu/ptolemyclassic/almagest/user.htm.

[Ber97b] Berkeley, University of California. *The Almagest - Volume II, Ptolemy 0.7 Programmers Manual*, 1997. http://ptolemy.eecs.berkeley.edu/ptolemyclassic/almagest/prog.htm.

[Ber97c] Berkeley, University of California. *The Almagest - Volume III, Ptolemy 0.7 Kernel Manual*, 1997. http://ptolemy.eecs.berkeley.edu/ptolemyclassic/almagest/kern.htm.

[BK02a] K. Below and U. Killat. Allocation of HTTP/TCP sources for traffic generation in large network simulations. In *2nd Polish-German Teletraffic Symposium (PGTS 2002)*, pages 175–182, Gdansk, Poland, September 2002. Online available: http://www.tu-harburg.de/et6/papers/documents/Below_Kai/below-PGTS2002.pdf.

[BK02b] K. Below and U. Killat. Internet traffic generation for large simulations scenarios with a target traffic matrix. In *Advances in Communications and Software Technologies (WSEAS ICOSMO 2002)*, pages

146–151, Skiathos, Greece, September 2002. WSEAS Press. Online available: http://www.tu-harburg.de/et6/papers/documents/Below_Kai/below-ICOSMO2002-c.pdf. ISBN: 960-8052-71-8.

[BK02c] K. Below and U. Killat. On the configuration of simulations of large network models with HTTP/TCP sources. In P. Tran-Gia and J. Roberts, editors, *15th ITC Specialist Seminar, Internet Traffic Engineering and Traffic Management*, pages 143–149, Würzburg, July 2002. Online available: http://www.tu-harburg.de/et6/papers/documents/Below_Kai/below-ITC-SPEC-Seminar2002-c.pdf.

[BK03a] K. Below and U. Killat. Generating prescribed traffic with HTTP/TCP sources for large simulation models. In K.-P. Fähnrich, editor, *Kommunikation in Verteilten Systemen (KiVS)*, pages 283–294, Leipzig, Germany, February 2003. Online available: http://www.tu-harburg.de/et6/papers/documents/Below_Kai/below-KiVS2003-c.pdf.

[BK03b] K. Below and U. Killat. Reducing the complexity of realistic large scale internet simulations. In *IEEE Global Communications Conference (GLOBECOM)*, San Francisco, USA, December 2003.

[BK04] K. Below and U. Killat. On the reduction of the complexity of realistic large scale network simulations. *To appear in: AEÜ, International Journal of Electronics and Communications*, 58(2), 2004.

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol — HTTP/1.0, May 1996. Online available: ftp://ftp.internic.net/rfc/rfc1945.txt, ftp://ftp.math.utah.edu/pub/rfc/rfc1945.txt. Status: INFORMATIONAL.

[BSK01] K. Below, C. Schwill, and U. Killat. Erhöhung des Nutzungsgrades eines ATM Netzes für den Wissenschaftsbereich (ERNANI). Technical report, Dept. Communication Networks, Technical University Hamburg-Harburg, September 2001. http://www.tu-harburg.de/et6/papers/documents/Below_Kai/Abschlussbericht-ERNANI.pdf.

[CB97] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997. Online available: http://www.cs.bu.edu/faculty/crovella/paper-archive/self-sim/journal-version.ps.

[CCLS01] Jin Cao, S. William Cleveland, Dong Lin, and Don X. Sun. Internet traffic tends to Poisson and independent as the load increases. Techni-

cal report, Bell Labs, Murray, Hill, NJ, 2001. Online available: http://citeseer.nj.nec.com/426819.html.

[CL97] Mark E. Crovella and Peter Lipsky. Long-lasting transient conditions in simulations with heavy-tailed workloads. In *Proceedings of the 1997 Winter Simulation conference (WSC97)*. Boston University, University of Conneticut, 1997. Online available: http://cs-www.bu.edu/faculty/crovella/paper-archive/wsc97.ps.

[CL99] Hyoung-Kee Choi and John O. Limb. A behavioral model of web traffic. In *Proceedings of the Seventh Annual International Conference on Network Protocols, Toronto, Canada*, November 1999. Online available: http://www.cc.gatech.edu/computing/Telecomm/people/Phd/hkchoi/paper/icnp99.pdf.

[CSA98] N. Cardwell, S. Savage, and T. Anderson. Modeling the performance of short TCP connections. Technical report, Washington University, Computer Science Department, November 1998. Online available: http://citeseer.nj.nec.com/cardwell98modeling.html.

[CTB98] Mark E. Crovella, Murad S. Taqqu, and Azer Bestavros. Heavy-tailed propability distributions in the World Wide Web. In Robert J. Adler, Raisa E. Feldmann, and Murad S. Taqqu, editors, *A Practical Guide yo Heavy Tails: Statistical Techniques and Applications*, pages 3–25, 1998. Online available: http://math.bu.edu/people/murad/pub/www4-posted.ps.

[Dau88] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41:909–996, 1988.

[Fel71] William Feller. *An introduction to probability theory and its applications*, volume 2. Wiley & Sons, 2nd edition, 1971.

[FGHW99] Anja Feldmann, Anne C. Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *ACM/SIGCOMM 1999*. AT & T, ISI/USC, 1999. Online available: http://www.research.att.com/~anja/feldmann/papers/sigcomm99_trace_sim.ps.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol — HTTP/1.1, January 1997. Online available: ftp://ftp.internic.net/rfc/rfc2068.txt, ftp://ftp.math.utah.edu/pub/rfc/rfc2068.txt. Status: PROPOSED STANDARD.

[FGW98] A. Feldmann, A.C. Gilbert, and W. Willinger. Data networks as cascades: Investigating the multifractal nature of internet WAN traffic. *ACM SIGCOMM*, pages 42–55, 1998. Online available: `http://www.research.att.com/~anja/feldmann/papers/sigcomm98_cascade.ps.gz`.

[FP01] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001. Online available: `http://www.icir.org/floyd/papers/simulate_2001.pdf`.

[GCM01] Liang Guo, Mark Crovella, and Ibrahim Matta. How does TCP generate pseudo-self-similarity? In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2001), Cincinnati, Ohio*, August 2001. Online available: `http://citeseer.nj.nec.com/guo01how.html`.

[GGS97] Helmut Gogl, Michael Greiner, and Hans-Peter Schwefel. Model calibration. Technical report, Technische Universität München, Institut für Informatik, December 1997. Online available: `http://citeseer.nj.nec.com/gogl97model.html`.

[GJL95] Michael Greiner, Manfred Jobmann, and Lester Lipsky. The importance of power-tail distributions for telecommunication traffic models. Technical report, Technische Universität München, Institut für Informatik, June 1995. Online available: `http://wwwbib.informatik.tu-muenchen.de/infberichte/1995/TUM-I9521.ps.gz`.

[HLN97] D.P. Heyman, T.V. Lakshman, and A.L. Neidhardt. A new method for analysing feedback-based protocols with applications to engineering web traffic over the internet. In *ACM SIGMETRICS, Performance Evaluation Review*, volume 25 of *1*, pages 24–38, 1997. Online available: `http://networks.ecse.rpi.edu/~rsatish/papers/heyman-sigmetrics97.pdf.gz`.

[Hug97] Hughes Research Laboratories, Department of Electrical and Computer Engineering. *Point Process Approaches for Modeling and Analysis of Self-Similar Traffic: Part II-Applications*, March 1997. Online available: `http://citeseer.nj.nec.com/ryu97point.html`.

[HWJL96] Tsang Danny H.K., Wales Kin Fai Wong, Ming Jiang, and Eric Y.S. Liu. A fast switch algorithm for ABR traffic to achieve max-min fairness. *Lecture Notes in Computer Science*, 1044:161–172, February 1996. Online available: `http://www.ee.ust.hk/~ustatm/papers/waizs96.ps`.

[IPL02] Abilene-I data set, IPLS-KSCY-20020814-102000-1, August 2002. http://pma.nlanr.net/Traces/long/ipls1.html.

[KB02] U. Killat and K. Below. Modellierung der Leistungsfähigkeit von TCP/IP in ausgedehnten Breitbandnetzen. In *25. Europäische Congressmesse ONLINE 2002*, volume II, page C230, Düsseldorf, January 2002. ONLINE 2002. ISBN 3-89077-232-3.

[KP87] Phil Karn and Craig Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM/SIGCOMM*, pages 2–7, August 1987. Online available: http://people.qualcomm.com/karn/papers/rtt.ps.gz.

[LG89] Alberto Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, 1989. ISBN: 0-201-12906-X.

[LML03] The linux kernel archives, kernel 2.4, include/asm-i386/page.h, PAGE_OFFSET, January 2003. http://www.kernel.org/pub/linux/kernel/v2.4/.

[LMS00] T.V. Lakshman, U. Madhow, and Bernhard Suter. TCP/IP performance with random loss and bidirectional congestion. *IEEE/ACM Transactions on Networking*, 8(5):541–555, October 2000. Online available: http://citeseer.nj.nec.com/madhow00tcpip.html.

[LTWW93] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In Deepinder P. Sidhu, editor, *ACM SIGCOMM*, pages 183–193, San Francisco, California, 1993. Online available: http://citeseer.nj.nec.com/leland93selfsimilar.html.

[Mor00] Robert Morris. Scalable TCP congestion control. In *IEEE INFOCOM 2000, Tel Aviv*, pages 1176–1183, March 2000. Online available: http://www.pdos.lcs.mit.edu/~rtm/papers/tp.pdf.

[MSM97] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997. Online available: http://citeseer.nj.nec.com/mathis97macroscopic.html.

[MSM03] Microsoft corporation, memory support and windows operating systems, January 2003. http://www.microsoft.com/whdc/hwdev/platform/server/PAE/PAEmem.mspx.

[NZI00] NZIX-II trace archive, NZIX-20000710-120000, July 2000. http://pma.nlanr.net/Traces/long/nzix2.html.

[PCB] Ptolemy classic. University of California at Berkeley. http://ptolemy.eecs.berkeley.edu/ptolemyclassic/body.htm.

[PF95] Varn Paxson and Sally Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3), July 1995. Online available: http://citeseer.nj.nec.com/32808.html.

[PKC96a] Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. Technical report, Boston University, July 1996. Online available: http://www.cs.bu.edu/faculty/crovella/paper-archive/files-protocols/TR-96-016.ps.

[PKC96b] Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proc. IEEE International Conference on Network Protocols*, pages 171–180, October 1996. Online available: http://citeseer.nj.nec.com/park96relationship.html.

[PKC97] Kihong Park, Gitae Kim, and Mark Crovella. On the effect of traffic self-similarity on network performance. In *Proceedings of the 1997 SPIE International Conference on Performance and Control of Network Systems*. Boston University, Purdue University, November 1997. Online available: http://cs-www.bu.edu/faculty/crovella/paper-archive/spie97.ps.

[RL96] Bo K. Ryu and Steven B. Lowen. Point process approaches to the modeling and analysis of self-similar traffic- partI: Model construction. *IEEE INFOCOM*, March 1996. Online available: http://www.wins.hrl.com/people/ryu/papers/infocomm96.htm.

[RL98] Bo Ryu and Steven B. Lowen. Point process models for self-similar network traffic, with applications. *Stochastic Models (M. Neuts, Ed.)*, 14(3):735–761, 1998. Online available: http://www.wins.hrl.com/people/ryu/papers/stomod98.htm.

[RN96] Bo K. Ryu and Mahesan Nandikesan. Real-time generation of fractal ATM traffic: Model, algorithm, and implementation. Technical report, CTR Technical Report, CU/CTR/TR 440-96-06, Center for Telecommunications Research, Columbia University, 1996. Online available: http://comet.columbia.edu/~mahesan/papers/fractal.pdf.

[RW00] R. H. Riedi and W. Willinger. *Self-similar Network Traffic and Performance Evaluation*, chapter Toward an Improved Understanding of Net-

work Traffic Dynamics, pages 507–530. Wiley, 2000. Draft available at http://www-ece.rice.edu/~riedi/Publ/rw99.pdf.

[Sta98] William Stallings. *High-Speed Networks, TCP/IP and ATM Design Principles*. Prentice Hall, 1998. ISBN 0-13-525965-7.

[SW01] Richard Stevens and Gary R. Wright. *TCP/IP illustrated*, volume 1 of *professional computing series*. Addison-Wesley, 19th edition, 2001. ISBN: 0-201-63346-9.

[Tan02] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 4th edition, August 2002. ISBN: 0-130-66102-3.

[TWS97] Murad S. Taqqu, Walter Wilinger, and Robert Sherman. Proof of a fundamental result in self-similar traffic modeling. *Computer Communication Review*, 1997. Online available: http://www.winlab.rutgers.edu/s3/reu/bgreading/TaqquWillinger_ccr97.ps.

[UCB] UCB/USC/LBNL/VINT. Network simulator ns (version 2). Avaliable from http://www.isi.edu/nsnam/ns/.

[VA99] Darryl Veitch and Patrice Abry. A wavelet based joint estimator of the parameters of long-range dependence. *IEEE Transactions on Information Theory*, 45(3):878–897, 1999. Online available: http://citeseer.nj.nec.com/179889.html.

[VFJ+99] B. Vandalore, S. Fahmy, R. Jain, R. Goyal, and M. Goyal. Overload based explicit rate switch schemes with MCR guarantees. In *ICCCN '99*. Ohio State University, October 1999. Online available: ftp://ftp.netlab.ohio-state.edu/pub/jain/papers/ic3n99_bv.ps.

[VKJ+98a] B. Vandalore, S. Kalyanaraman, R. Jain, R. Goyal, and S. Fahmy. Simulation study of World Wide Web traffic over the ATM ABR service. In *Conference on Performance and Control of Network Systems II*, volume 3530, pages 415–422. Ohio State University, November 1998. Online available: ftp://ftp.netlab.ohio-state.edu/pub/jain/papers/websp_bv.ps.

[VKJ+98b] B. Vandalore, S. Kalyanaraman, R. Jain, R. Goyal, and S. Fahmy. Worst case buffer requirements for TCP over ABR. In *Proceedings SICON'98, Singapore*. Ohio State University, June 1998. Online available: ftp://ftp.netlab.ohio-state.edu/pub/jain/papers/wrstc_bv.ps.

[VKMV00] A. Veres, Zs. Kenesi, S Molnár, and G. Vattay. On the propagation of long-range dependence in the internet. In *ACM SIGCOMM*, August 2000. Online available: http://citeseer.nj.nec.com/423564.html.

[WTRW97] Walter Willinger, Murad S. Taqqu, Sherman Robert, and Daniel V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997. Online available: `http://math.bu.edu/people/murad/pub/source-printed-version-posted.ps`.

Curriculum Vitae

27.03.1970	Born in Hamburg, Germany
1976 – 1980	Primary school in Hamburg
1980 – 1989	Rudolf-Steiner school in Hamburg, high school graduation
1989 – 1990	Community service in establishment for handicapped people
10.1991	Started studies of engineering in electronics and telecommunication at the Technical University Hamburg-Harburg (TUHH)
10. – 12.1994	Internship at Cochlear in Sydney, Australia
10.1998	Graduated as Dipl.-Ing. at TUHH
10.1998 – 07.2003	PhD student in the department Communication Networks of the TUHH
From 08.2003	System Engineer at Airbus Hamburg