Budapest University of Technology and Economics

Modeling TCP Dynamics and Engineering Service Differentiation in TCP/IP Networks

András Veres

Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics

Ph. D. Dissertation

Advisors:

Edit Halász, Ph.D.
High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics, Hungary

Andrew T. Campbell, Ph.D.
Department of Electrical Engineering
Columbia University, New York, USA

Sándor Molnár, Ph.D.
High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics, Hungary

Budapest, Hungary
2004

Contents

Chapter 1

Introduction

In recent years, the Internet and its TCP/IP protocol suite have had unprecedented success and impact on the way we interact and communicate. The Internet has become the single global and most successful network used by millions of users around the world. So why has the Internet been such a phenomenal success? The likely answer to this question is that its design is based on a set of enduring principles, that is, simplicity, soft state protocols, scalability, distributed architectures, and the end-to-end principle [SRC84, Cla88].

Simplicity is a property that makes the TCP/IP protocol so flexible and suitable to enable the building of thousands of applications on top of it. Simplicity proves to be more important than more optimal resource usage that could have been possibly achieved by a more sophisticated, but less flexible protocol suite. Soft-state protocols and distributed algorithms are important principles to achieve network robustness, not just against unreliable protocol implementations or hardware failures, but importantly also robustness against unforeseen but otherwise allowable network usage. For example, a new “killer application” may cause the system to collapse if it overloads a weak point in the architecture. The principle of scalability was one of the main requirements behind every standard of the Internet Engineering Task Force (IETF) mandating that all solutions have to scale up to even higher data rates and larger number of hosts than anyone would anticipate for the future.

The end-to-end principle is intimately related to all of the above Internet goals. According to the end-to-end principle most of the intelligence should be in the end hosts and only the necessary minimum intelligence should be implemented in the interconnecting routers. Thus the routers on a connection’s path more or less just dumbly forward the packets toward the destination without making any further consideration about the applications running in the end hosts, or their traffic demands. In turn, the network hosts at the edges have to cooperate in a distributed manner to avoid network congestion collapse. This is achieved by the TCP congestion control mechanism introduced by Van Jacobson in [Jac88]. Every network host and every TCP connection autonomously estimate the network load, continuously adjusting their own traffic rate to avoid network congestion. This simple, robust distributed algorithm proves to be sufficient to keep the network congestion at an acceptable level, in a wide range of scenarios, from low to very high link speeds and traffic load levels. Though several improvements have been added to the original TCP algorithm, such as fast retransmit, fast recovery, or selective acknowledgements, the basic principles remained untouched (i.e., that flow control is performed at the edges). Although these design principles have obvious advantages, they do not come without costs. As a result, it is significantly more complicated to understand what is happening in the network at any point of time compared to other networking architectures and protocols. Consequently, it is significantly more difficult to design and operate TCP/IP networks efficiently.

1.1 Motivation of the Research

Currently, because of the lack of precise analytical methods, network engineers have to rely on heuristics and overprovisioning when they want to design and manage TCP/IP networks. These methods are sufficient in backbone networks, because the economy of scale allows providers to operate high capacity backbone links even at low utilization levels. However, the cost of network capacity in access networks is expensive. Consequently, access networks are usually operated at much higher levels of utilization, leading to the frequent, serious degradation of service quality. Future advances of Differentiated Services (DiffServ) [BBCD98] will demand more predictable and robust network performance than traditional best-effort service even in backbone networks. The necessary level of robustness cannot be achieved by heuristics in a cost effective way any more.

Precise analytic *network performance models*, in contrast to pure heuristics, would help network operators to more deeply understand the state and performance of the network. Network performance models establish mathematically proven relationship among:

1. models of network elements (e.g., packet classification, scheduling mechanisms);
2. models of network protocols (e.g., Ethernet, TCP);
3. traffic demand (e.g., WWW, voice traffic); and
4. performance metrics (e.g., delay, loss).

Application of precise performance models would enable network operators to engineer their networks for higher and more robust grades of services. There has been extensive research on all of the items listed above, from traffic models to quality of service metrics. State of the art in performance modeling, however, does not allow us to achieve the necessary precision yet, and heuristic network engineering is still the most widely applied method.

The reason why performance models are not precise is because there are a number of unresolved problems when the modeling components listed above are applied in a common performance management framework. The central argument of this thesis is that these components have strong impact on each other, and to develop precise performance models, we have to consider how network protocols, mechanisms, and traffic demands interact. Our objective is to analyze the interaction of modeling components and to develop practical performance models based on the new insights gained from the research.

In this thesis, we classify two different performance-modeling frameworks for wired and wireless TCP/IP networks. The specifics of the radio channel require special wireless control protocols (e.g., wireless Medium Access Control, mobility protocols) suited for the wireless channel with different properties than wired scheduling algorithms. In addition, the statistical properties of traffic in wireless networks may be different than in wired networks due to host mobility and the time varying nature of the radio medium. On the other hand, both wired and wireless networks share almost the same higher layer TCP/IP protocols and applications, thus, most of the results from TCP/IP traffic modeling can be applied to both wired and wireless scenarios.

During the research presented in this thesis, we found that conventional analytic techniques, which analyze only one aspect of performance models at a time, are not sufficient. The analysis of the problem requires new modeling assumptions, measurement techniques, and sometimes even new mathematical tools to work with the new models. Although it is the TCP/IP design principles that can be regarded as being responsible for the failure of conventional methods, without these principles the Internet would have probably never reached its current growth. Thus it is desirable to assume that it is not the principles that have to be drastically modified to improve the Internet, but the methods of performance modeling, network management, and design that have to better suit these principles.

The thesis is organized around the above arguments. In the first part of the thesis, we present research results on the interaction between performance modeling components. We discuss the previously accepted assumptions that now have to be reconsidered, and demonstrate that new modeling techniques and assumptions lead to a better understanding of network mechanisms. Based on these new insights gained from the first two chapters, the second part of thesis presents performance models and performance management methods for wired and wireless IP networks.

1.2 New Insights in TCP/IP Traffic Modeling

In what follows, we summarize the key previous work in the field of TCP/IP traffic modeling, introduce our motivation for research, and our contributions to this area.

Our research methodology is to apply mathematical modeling techniques in the analysis of TCP/IP traffic related problems to establish a sound basis for handling more practical problems that arise during the operation and design of TCP/IP networks. During our research we placed an emphasis on strengthening the relationship between mathematical models and real network mechanisms.

Mathematical models are always “distilled” versions of real-life mechanisms that focus on the important aspects of reality and disregard the less important aspects. Probably the most serious mistake a researcher can make is not to take into account an important property present in real life when developing a mathematical model. In this thesis, we take special care to ensure that both the assumptions used in the models and the conclusions derived from the models are compared with measurements and simulations from realistic scenarios whenever possible.

1.2.1 Previous Work on TCP/IP Traffic Models

The basis of using conventional queuing networks and Markov models for the analysis of TCP/IP networks was first brought into question when it was discovered that traditional, relatively easy-to-use, and well understood, short memory models have serious pitfalls [LTWW93, PaFl95, PaFl97]. Beginning in the early 90s, research on Internet traffic indicated that TCP/IP traffic has fractal nature, in particular, statistical self-similarity, long-range dependence and multifractals [LTWW93, BSTW95, CrBe96, CTB96, TTW97, FGW98]. The consequences of these findings are far reaching, for example, it is shown that in the presence of long range correlations, the performance of a network may be significantly worse than the results obtained using "classical" network theory [Nor94].

Initial work in the field of fractal traffic models has changed many previously held assumptions of classic network theory [PaFl95]. One very basic assumption that remained intact is that the network can be modeled separately from the traffic sources. When solving a problem, a researcher would first decompose the model into a model describing networking mechanisms (e.g., buffering, scheduling and routing), and a separate model describing the traffic passing through these network elements. Then, one would calculate the performance of the traffic flow, (e.g., delay variation, or packet loss probability). Recent research demonstrates that this approach does not hold forcing this "last" assumption to be dropped. When modeling network performance, network and source models cannot be separated [ArKa99, GCM00] since network and source mechanisms are interconnected by adaptive mechanisms. TCP plays an important role here because it is the most important adaptive protocol in the current Internet. A new research area has been dynamically developing, which focuses on so-called macroscopic TCP models [MSMO97, PFTK98]. The macroscopic models establish connections between TCP throughput and end-to-end path properties (e.g., packet loss and delay).

The macroscopic models analyze the connections between source mechanisms and network properties, however, the assumptions on end-to-end path properties as well as the performance outputs of the models lack the detailedness of packet dynamics found either in conventional or fractal traffic models. The motivation of our research in this area is to "bridge" the research gap between fractal models and the research in TCP models and to develop traffic models that take into consideration both properties of TCP/IP traffic, namely, the complex correlation structure of Internet traffic and TCP congestion control dynamics.

1.2.2 Contributions to TCP/IP Traffic Modeling

In this thesis, we analyze two important properties of the TCP protocol that have impact on TCP/IP traffic dynamics. Chapter 2 introduces a novel approach to model the competition between multiple TCP connections sharing a common bottleneck buffer. Chapter 3 analyzes the adaptation property of TCP congestion control. We discovered that both competition and adaptation of the TCP protocol, contrary to common wisdom, have an impact on a wide range of timescales of Internet traffic dynamics. They even contribute to the wide scale self-similarity observed in the Internet in at least two ways: competition can lead to generation [C2, W3] and adaptation causes propagation [C3, W2, J2] of self-similarity. To provide strong foundations for our arguments, we demonstrate our findings by simulations, real Internet measurements, as well as mathematical analysis.

Modeling Competition for Resources among TCP Flows

TCP flows continuously intertwine in the Internet competing with each other for service capacity and buffer space in bottleneck routers. Competition is controlled by the window based flow and congestion control algorithms implemented in end-hosts. In Chapter 2, we explore the dynamics of this competition, that is, how to describe the way several TCP flows share a common resource as time evolves.

Previous work on TCP modeling is based on stochastic modeling techniques. We found that if we approach the problem using deterministic modeling techniques, we are able to explain several real-life phenomena that cannot be understood using stochastic models. Our main contribution, in Chapter 2, is that we demonstrate that the end-to-end congestion control used by the TCP protocol, while competing for networking resources, generates deterministic chaos. An important message of our work is that random traffic behavior is not exclusively due to "random" effects, but also due to complex chaotic behavior of TCP.

Chaotic systems, although completely deterministic, produce time-series seemingly indistinguishable from stochastic processes; this is why the most obvious approach is to use stochastic models. On the other hand, chaotic systems have unique properties and they are able to produce a diversity of phenomena. In Chapter 2, the most important of these chaotic properties and phenomena are demonstrated and analyzed:

- fractal attractors of a system consisting of competing persistent TCP flows;
- extreme sensitivity to initial conditions;
- phase transitions between chaotic and non-chaotic states; and
- for certain parameters TCP dynamics produce self-similar traffic.

We introduce a method to visualize the attractor of a system consisting of two TCP connections based on the monitoring of the congestion window variable of the TCPs. The chaotic and non-chaotic regimes are distinguishable by using this visualization technique. We measure the dimension of the attractors in several simulation configurations, and demonstrate that while in the chaotic regime, the system's measured attractor has fractal dimension. On the other hand, when the system is the periodic regime, the attractor has an integer dimension.

We introduce a method to measure the Lyapunov exponent of a network configuration using simulation. The Lyapunov exponent indicates the sensitivity of the system to initial conditions or external effects. We demonstrate that for a network configuration the exponent is positive, which means that small perturbations grow exponentially in time.

We show that competing TCP connections can go through phase transitions from simple periodic to seemingly random, chaotic, and finally self-similar behavior. The regime a system is in depends on the system parameters (e.g., buffer space, service rate, and number of competing TCPs). The transition can happen intermittently as indicated by the experiments.

By showing that TCP/IP networks can be considered as being chaotic, it is now possible to use the models and tools developed in other fields of science where chaos has been used before successfully.

Modeling Adaptation of TCP Congestion Control

Competition and adaptation are related aspects of the TCP congestion control mechanism. Previous work, as well as our results in Chapter 2, demonstrate that self-similarity can emerge in the network for several reasons. In Chapter 3, our goal is to investigate how network path properties, especially self-similarity, impact the end-to-end dynamics of TCP traffic.

The main contribution of Chapter 3 is that we demonstrate and analyze that TCP approximates a linear system, efficiently adapting to any stochastic background traffic process it encounters in a bottleneck buffer and propagates the correlation structure of the background traffic process toward the end-hosts. If the background process is self-similar, TCP inherits and propagates this self-similarity with the same degree of self-similarity characterized by the Hurst exponent. We demonstrate the presence of self-similarity in a number of wide-area TCP measurements in the Internet. The significance of this result is that the performance of TCP-based applications as seen by the end-user can be affected by long-range correlations originating from a distant point in the network. Long-range correlations may have serious impact on networking and application protocols, consequently the end-user perceived network performance might be seriously degraded due to this effect.

Our research results presented in Chapter 3 analyze some aspects of this propagation effect using simulations and mathematical analysis. We found that the propagation property of TCP is independent of how self-similarity is generated: it propagates any kind of self-similarity, even if it is not generated by TCP itself. If a TCP connection passes a bottleneck buffer with long-range dependent traffic load, it will inherit the correlation structure of that process and carry it towards both end-systems. Our analysis reveals that the mechanism of propagation is independent of the TCP version. This propagation effect takes place above a characteristic timescale that depends on the end-to-end path properties. We derive a simple estimation of the characteristic timescale based on previous results from macroscopic TCP models.

We prove analytically that, in certain cases, TCP may even strengthen self-similarity in the network by propagating the largest Hurst exponent encountered on the end-to-end path. The significance of propagation is that traffic fluctuations in different parts of the network can be closely related, thus, when analyzing and trying to improve the performance of the network we have to take a wider, end-to-end perspective. This is especially important at possible bottlenecks (e.g., access networks or traffic exchange points).

1.3 Provisioning Differentiated Services in Wired and Wireless Networks

Chapters 2 and 3 argue that source and network mechanisms are strongly related because of the end-to-end congestion control mechanism of TCP. As a consequence, a wide range of packet dynamics can be observed in the Internet, characterized by complex correlation structures and strange attractors. One of the first questions a network engineer asks is: *‘How do we design and manage the network based on this knowledge?’*

In the second part of the thesis, we apply the results from Chapters 2 and 3 in a wider context of performance models and performance management. Since performance management is particularly important if service quality is to be maintained, we discuss the problem of performance management in Differentiated Services networks [BBCD98]. DiffServ represents the core QoS architecture for the future Internet. It is envisioned that DiffServ will be able to support several levels of services, from the traditional best-effort to real-time services. DiffServ offers certain quality of service assurances while still maintaining architectural scalability.

1.3.1 Previous Work on Provisioning DiffServ

Previous work in the field of provisioning quality of service in data networks has several limitations. Although there have been a number of publications in this field, most methods used in practice are still heuristic. There are no analytic proofs of the performance of these heuristics, which is limiting because it is also not known how they would perform if the traffic mix changes. This is why our goal is to develop methods that have an analytic basis.

Analytic methods developed before are not well suited to Differentiated Services networks because of the following main reasons:

- Methods that are based on deterministic bounds are too conservative and waste resources, or methods that rely on precise and complex flow descriptors or homogeneity of applications, which are not feasible because of the apparent diversity of Internet applications. [PaGa93, PaGa94, WKLZ96, LeB98]
- Some methods assume certain statistical properties, such as short-range dependence or Markovian properties [KWC93, KoMi98, Kel96, KWC93], which are not realistic because of the wide-range presence of long-range dependence in the Internet. This is also underlined by our research on TCP dynamics as discussed earlier.
- Some methods are suited for ATM networks, where the guarantees are very strict, e.g., losses below 10^{-12}. In DiffServ networks the guarantees do not have to be so strict. We can categorize methods based on Large Deviation Theory (LDT) in this group (see e.g., [CLTR97]).

There are also requirements that originate from the DiffServ architecture:

- Detailed per-flow measurements are not feasible in DiffServ due to scalability reasons.
- Complex packet schedulers, which require non-scalable per-flow states in the scheduling hardware are also not feasible.
- The guarantees should not be just asymptotically accurate, for example if the number of flows and buffer sizes are very large or packet loss probability is very small.

There has not been sufficient research into how to address all the above problems for DiffServ networks within the same performance management framework. These problems arise in both wired and wireless DiffServ networks. Wireless networks have other properties due to mobility and the specifics of packet scheduling on the radio channel. There have been several suggestions how to support service guarantees in wireless networks before. Previous work on wireless performance management has the following additional problems:

- Some of the proposals require central scheduling nodes, which has the disadvantage that the central entity has to be able to know not only where mobile hosts are and when they want to send packets, but the central node has to cooperate with other central entities operating in the same radio channel.

- Some proposed solutions do not require central control, but would restrict the traffic flows to sending packets in predefined, order (e.g., periodic patterns).

Although the problem of central scheduling can be solved using complex, centralized control algorithms and sophisticated protocols, we would obviously sacrifice flexibility. In addition, the solution would not be suitable for ad-hoc wireless networks where central entities are non-existent.

The problem with the second proposal is that traffic patterns are usually difficult to predict because of the diversity of applications and protocols. From the research in Chapters 2 and 3 we know that TCP traffic patterns depend on both end-to-end mechanisms operating in end-hosts and the network. The traffic statistics of a wide-area TCP connection, for example, can show both short-range of long-range dependence depending on the end-to-end path. Due to the propagation effect discussed in Chapter 3, the packet arrival dynamics may depend on the dynamics of the bottleneck of the end-to-end TCP path. The competition between TCP flows in a bottleneck (possibly at the wireless hop itself) can also give rise to a diversity of packet arrival patterns, as shown in Chapter 2. Therefore, the solution has to be robust in respect to the properties of the traffic flows over the wireless channel.

1.3.2 Contributions to Provisioning Differentiated Services

In Chapters 4 and 5, we introduce methods to provision Differentiated Services in wired and wireless networks, respectively. We discuss the specifics of DiffServ and the requirements and practical limitations of implementation. The proposed methods follow the main design principles of TCP/IP networks and are based on the insights from TCP/IP traffic modeling discussed in Chapters 2 and 3.

Performance Management for Wired DiffServ Networks

Our aim is to approach the above problem on analytic ground and develop practical methods that can be implemented in real networks. Our analytic approach is within the context of effective bandwidth theory introduced in [GAN91, Kel96, GiKe97]. We first analyze what assumptions have to be considered to establish robust and precise mathematical models for the calculation of the effective bandwidth while avoiding the problems found in previous work.

In our analysis, we first derive general mathematical bounds for DiffServ service classes, given that we are not constrained by the limitations of networking implementations, so we have instant and full knowledge about the most important statistical properties of traffic flows. This first step allows us to prove several important general theorems that describe important relationships between traffic statistics, network mechanisms and network performance. Following this we derive several other theorems that take into consideration these constraints. The benefit of this methodology is that the results can be later extended as these networking constraints change (e.g., as more precise measurement techniques, improved traffic shapers/policers and resource reservation protocols are developed). These methods can be used for network dimensioning or can be implemented to perform flow admission control in bandwidth brokers.

We develop effective bandwidth formulae for throughput sensitive Assured Forwarding (AF) [HBWW99] and delay sensitive Expedited Forwarding (EF) [DCBB02] classes. For AF classes, we develop methods to estimate the probability of link saturation. The analytic methods developed for EF classes estimate delay and loss values on a per-class level. These methods are based on the following assumptions:

- per-class average load measurements;
- simple DiffServ policers at the edges;
- arbitrary correlation structure; and
- arbitrary traffic rate distribution.

We further improve the precision of resource estimation by adding a little extra complexity in the measurement/classification process:

1. per-class average load and rate variance measurement; and
2. per-group measurements within a class based on stateless grouping.

We analyze the tradeoff between the requirements of scalability involved in traffic flow classification and resource efficiency. This analysis also evaluates the price we pay for aggregate traffic handling compared to per-flow processing. We found that most of the achievable statistical multiplexing gain can be utilized with only slight increase in architectural complexity, which supports the end-to-end principle from an analytic perspective.

Performance Management for Wireless DiffServ Networks

Chapter 4 introduces resource management methods for wired TCP/IP networks. Wireless packet networks require other methods because of the peculiarities of the radio environment. These peculiarities include relatively large bit error ratios due to interference and radio propagation effects, shared channel, scarcity of radio bandwidth, and host mobility. All of these characteristics have impact on traffic management. In Chapter 4, we assume that the scheduler on a link has a well-known service capacity C, which has to be managed to serve several traffic classes. In a wireless environment, the channel capacity available to a node is not constant, and the wireless Medium Access Control has to take into account the effect of shared radio channel, interference, collisions, overlapping cells, and it has to be aware of the packets waiting in the queues at other nodes as well.

Because of the above challenges, our goal is to develop methods not just for DiffServ resource management, but introduce a complete traffic control "suite", which incorporates methods for robust, flexible, distributed DiffServ packet scheduling, resource estimation, and traffic control algorithms. A more complete solution improves the compatibility between the IP layer and the lower radio specific data-link and physical layers, improving resource efficiency, which is desirable due to the scarcity of radio bandwidth.

The arguments and results presented in Chapter 5 are valid for a broad class of shared channel wireless data technologies. To be able to demonstrate the validity of the arguments, we choose a particular wireless technology, the IEEE 802.11b standard, since currently IEEE 802.11b is the most popular wireless LAN standard in use [IEEE802.11]. The solutions presented in this chapter are extensions of this standard, and current implementations could be easily modified.

Our basic research methodology and objectives are the same as in Chapter 4: (1) understand the protocols and the wireless environment using mathematical models; (2) analyze the model to develop methods to improve network performance, while making sure the methods follow the main design principles of the Internet; and (3) evaluate the proposed solutions with measurements and simulations.

The distributed wireless DiffServ solution has to take into account the impact of the wireless channel as well as the impact of TCP/IP dynamics. However, during the modeling phase, because of the complexity of the problem, we decided that only the impact of the radio channel of the above two aspects are modeled in detail analytically and we use a simplified traffic model to keep the model mathematically tractable. Because of this limitation the model is not sufficiently precise to be applied for traffic management directly. Nevertheless, the analytic model can be used to evaluate the parameter-space of the wireless Medium Access Control (MAC) protocol qualitatively. The results provide clues of how to improve the network performance, and how to modify the original best-effort MAC protocol to offer better than best effort services.

Since efficient resource management cannot be based on this analytic model, it is not possible to apply the same methodology of resource management developed for the wired infrastructure as discussed in Chapter 4. We solved this problem with the Virtual MAC (VMAC) algorithm, which is a pragmatic solution for precise resource estimation: it is not completely analytic yet it is not completely heuristic. The VMAC algorithm takes as an input passive monitoring data of real traffic patterns overheard on the channel and applies an accurate model of the real MAC algorithm on it. As a result, we are able to take into account both the complexity of traffic arrivals (e.g., correlation structure propagated from a distant bottleneck) and the complexity of the distributed wireless MAC algorithm. This approach allows us to precisely estimate the available resources and the QoS of DiffServ classes on the channel.

To evaluate the efficiency of the above ideas, we implemented and evaluated the VMAC algorithm using both real-life measurements and simulations. We demonstrate that a stable state can be achieved by using these distributed algorithms throughout the wireless network consisting of a large number of wireless hosts.

Chapter 2

Chaotic Nature of TCP Congestion Control

Traffic models used to model current Internet traffic can be categorized into two major groups: *link/source* and *network level models*. Link level models fit statistical models to measurements of traffic on network links or traffic sources for example a WWW server. Recently, a major contribution to this area concerned the exploration of fractal and long-range dependent property of traffic, namely that the second order statistics of traffic volumes observed at different scales does not change. This result revolutionized performance modeling and questioned previous models based on Markovian behavior (see Paxson and Floyd [PaFl95]). We mention two major publications in this area: Leland, Taqqu, Willinger and Wilson demonstrated through rigorous tests that Ethernet traffic is self-similar [LTWW93], Crovella and Bestavros proved how WWW as the major contributor to current Internet traffic can cause long-range dependence and self-similarity [CrBe96]. Chaotic-maps appeared as efficient methods to generate packet traffic on the link/source level, see for example the work by Erramilli and Singh [ErSi90] and the same authors with Pruthi [ESP94].

The drawback of link/source level models is that they disregard one of the major properties of today's Internet, namely that the majority (80-90%) of traffic is generated and controlled by the TCP protocol, which is adaptive in nature. The consequence of adaptivity is that the source behavior cannot be disconnected from the network configuration (e.g., routing, scheduling, buffer management). Traffic statistics change if the network configuration changes, so a link/source model is valid only for the configuration (and all other circumstances) that is present at the time of model fitting. A recent paper by Arvidsson and Karlsson [ArKa99] demonstrates that adaptive simulated traffic behaves significantly differently in the buffers than would be indicated by link/source models.

This problem motivates *network level models*, which attempt to form a unified model taking into account the cooperation of all source and network mechanisms. Due to the complexity of this problem the models published in this area are still in their early phase of development. Mathis, Semske, Mahdavi and Ott [MSMO97] published an analytic model concerning the macroscopic behavior of TCP, while Padhye, Firoiu, Towsley and Kurose [PFTK98] also model the impact of the timeout mechanism on TCP throughput. The drawback of these models is that they assume that TCP congestion control always behaves in a nice periodic and predictable fashion, and that apparent randomness in TCP traffic is due to stochastic effects exclusively. This is, however, in contradiction with the measurements and simulations of TCP traffic.

In this chapter, we bridge the two modeling approaches by modeling the network level behavior of aggregate TCP flows while reproducing the complexity found in link/source level models. The key finding presented in this chapter is that cooperating TCP congestion control processes together form a deterministic chaotic system, which is able to produce periodic and non-periodic, predictable and non-predictable, short-range dependent and also self-similar behavior. We demonstrate some of the key properties of chaotic systems present in the Internet.

A system is called chaotic if it satisfies the following conditions [Ott93, DiMu95]:

- nonlinearity;
- determinism;
- order in disorder;
- sensitivity to initial conditions, or the "butterfly effect"; and
- unpredictability.

Nonlinearity means that the system is controlled through nonlinear functions. In case of TCP, nonlinear functions are used for round-trip time (RTT) measurements, slow start and congestion avoidance.

Determinism means that the system's future is fully described by the past. This is also true as TCP works in a self-clocking manner: no randomness is used and every event (e.g., sending of a packet or time-out) is completely determined by the past.

We are not going to discuss these two properties because they are obviously characteristic of TCP. However, the other conditions need more insight and proof. In this chapter, we discuss these other properties.

The current network level models for TCP/IP traffic assume periodic and stable behavior. Such behavior is demonstrated by simulations in Section 2.2; however, in Section 2.3 and Section 2.4, we prove that this behavior is not universal by giving examples for more complex periodic, and finally, non-periodic patterns.

In Section 2.3, we introduce the notion of *attractors* – hidden multidimensional trajectories of the TCP process, and give a method to efficiently visualize them, thus, making it possible to examine the hidden order in an otherwise seemingly random process.

A system satisfying the *butterfly effect* is extremely sensitive to small changes in the initial parameters or minute perturbations of the system. This property is the major trademark of chaotic systems. In Section 2.5, we demonstrate this property of TCP and quantify the sensitivity of the system by measuring the *Lyapunov exponent* of the system's trajectory.

Although deterministic, the chaotic nature of TCP has inhibited correct network level models in this area so far. This property results in a qualitative difference compared to previous macroscopic models. In Section 2.6, we show that TCP can generate traffic that shows scaling behavior spanning several timescales. This finding throws new light on self-similar traffic modeling, explaining self-similarity with deterministic chaotic mechanisms and not with higher layer stochastic mechanism [CrBe96].

Finally, in Section 2.7, we demonstrate the existence of some of the above phenomena in a real network. We show that by modifying certain configuration parameters, the system changes from a stable, periodic to a seemingly random, non-periodic behavior.

By showing that TCP congestion control leads to deterministic chaos, we can introduce a new set of analytical tools to be applied in TCP/IP traffic modeling. Some of the chaos tools have been developed in other fields of science, for example, physics, meteorology, biology or medicine. These analytic tools offer new insights into TCP/IP traffic dynamics. For example, by using chaos tools, we can revisit and explain certain phenomena observed in TCP/IP networks, now in a unified manner (e.g., phase effects, duality of periodicity and randomness).

2.1 TCP Congestion Control

The TCP protocol [RFC793][Jac88] provides reliable data delivery between two computers using data acknowledgements and retransmissions. TCP uses the so-called window based congestion control to control its data transfer rate over the interconnecting networks. The window controls the amount of data that can be outstanding unacknowledged in the network at any time. So if the window is x bytes, then at most x bytes can be delivered maximum during the end-to-end round-trip time between the two computers.

TCP does not have explicit information about the optimal congestion window it should use, so it uses implicit information instead by means of detecting packet losses and estimating the round-trip delay. TCP assumes that packet losses are indications of network congestion. When a packet loss happens, TCP drops its window (and thus reduces its speed). In between losses, TCP gradually increases its sending rate. If the congestion window is small, TCP increases fast; it increases by one packet after it received an Ack of each sent packet (slow-start phase). After it has reached the so-called slow-start threshold, it increases by one packet every round-trip time (congestion avoidance phase).

There are several optimizations of this basic algorithm, like fast retransmit, fast recovery, or selective acknowledgements, just to name the most important ones [RFC1072][RFC2001][RFC2018].

2.2 Macroscopic Models' Assumptions: Periodicity and Order

First we take a simple configuration containing just two greedy TCP connections sharing a single link. This configuration helps us to explain the graphical methods used later in the chapter. We used the *ns-2* [NS] simulator [1] and the TCP Tahoe version for all simulation experiments (later in Section 2.7 we present real network measurements as well). The link parameters are: link rate $C = 0.2$ Mbps, delay $d = 10$ ms, buffer size $B = 20$ packets. The receiver window is set to a very large value so that the congestion window (*cwnd*) is the limitation. We used the Tahoe version of TCP.

[1] Exact verson of the simulation platform was ns-2.22

The two TCPs are started simultaneously. After a short transient (6 cycles of $cwnd$) the two TCPs settle down into a periodic pattern, one of them always a little ahead of the other, but both follow the same pattern of slow start, congestion avoidance, packet loss and backoff. This is clearly visible in Figure 2.1. Another way of displaying the system evolution is to use

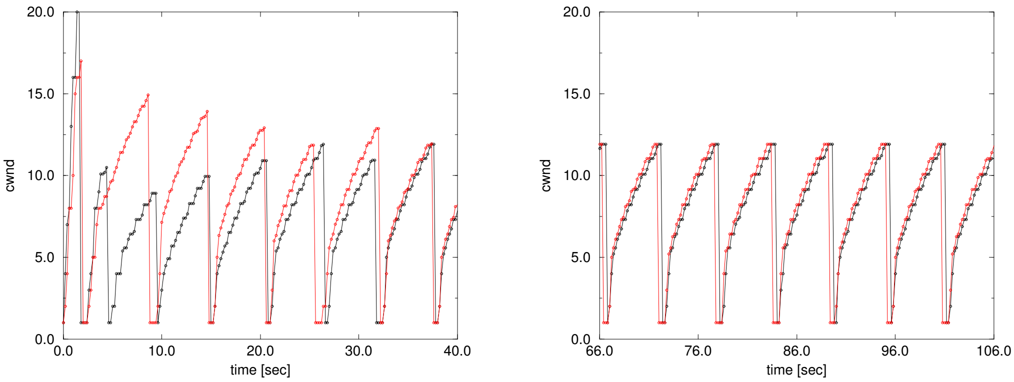


Figure 2.1: The congestion window processes of two competing TCP sources: a) with transient part b) transient removed.

a spatio-temporal graph where the window size of a TCP is displayed as a shaded strip: the larger the window, the brighter the shade (we borrowed the idea from [BLD95]). This method is used in Figure 2.2, the two TCPs are displayed on top of each other: the first TCP is displayed in the first row, the row below corresponds to the second TCP. The periodic pattern appears as synchronized "waves", the back-off times colored as black strips are always close to each other for the two TCPs.

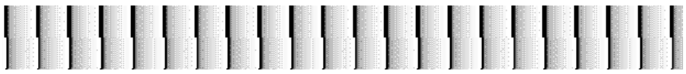


Figure 2.2: Spatio-temporal evolution of the congestion window processes.

One can observe the system's evolution not only as a function of time, but also by drawing the trajectory of the system as it moves in the phase space. The phase space is a multi-dimensional space where each dimension represents a system variable, thus each point in the phase space represents a unique state of the system. If we plot the evolution of the system in this space, then – as the system is completely deterministic – if the system gets back to a previous point, it will continue that path again, creating a closed loop. If the system is periodic, then the corresponding trajectory will be a loop and *vice versa*, if the system evolution can be represented by a loop in the phase space, the system is periodic. This method is thus very appealing to examine the periodicity of a multi-TCP system.

Even in this 2-TCP system the number of state variables that completely describe the system is very large (e.g., the whereabouts of previously sent packets, internal variables of the sending and receiving TCPs), it is not possible to draw them on a single piece of paper but it is possible to properly choose a section of the phase space. We chose the TCP congestion window (*cwnd*) size because it has a close relation with the sending rate of TCP. We have logged the *cwnd* values every 10 ms for each TCP. Figure 2.3 shows this plot for the previously examined simple configuration. The transient part is removed from the right graph. As can be seen from the figure, the process gets into a periodic loop, although the period is fairly large. Interestingly, this loop is very stable, which means that it does not matter how we disturb the system (e.g., drop a packet randomly) or choose the initial conditions (e.g., we start the second TCP a few seconds later) eventually the system will return to the same pattern. The graph also reveals that the two TCPs are synchronized, they move along a “staircase”; that is, they increase their *cwnd* one after the other, finally at the top loss is detected and both TCPs decrease their *cwnd*s to 1 packet at which time the period starts over again.

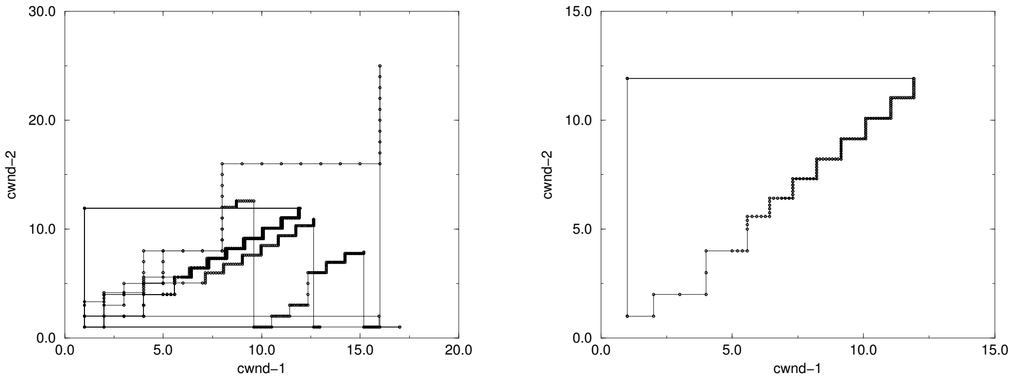


Figure 2.3: The congestion window processes of two competing TCP sources. a) with transient part b) transient removed. (The connecting lines are not data points, they show the movements between points.)

This behavior is unsurprising and is assumed widely when calculating macroscopic performance values. Unfortunately, this nice behavior is not universal as will be shown later in this chapter.

2.3 Complex Periods and Attractors

Can this simple system produce different behavior? Surprisingly, yes. If we change the system parameters: link rate $C = 0.5$ Mbps, delay $d = 10$ ms, buffer size $B = 4$ packets, we get a period again, but it is more complex, see Figure 2.4 and Figure 2.5. There is still an underlying regular beat, but on a larger timescale there is an alternating pattern: one gains speed over the other for a given time and then the other takes over.

By changing the parameters further, we can make this simple system change from a simple regular beat to a very complex pattern; here our form of graphical representation reaches its limit. The problem is that the chosen set of system variables (i.e., the *cwnd*s) is just a subset of the

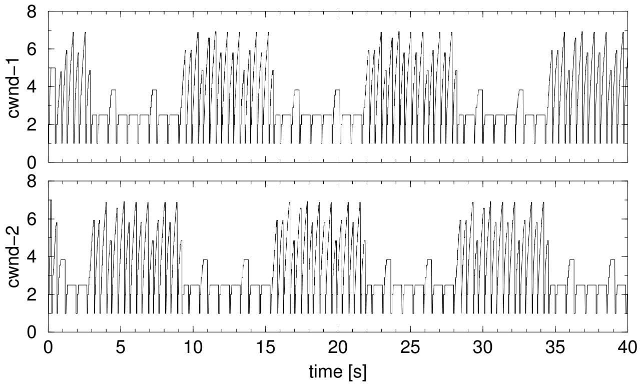


Figure 2.4: The congestion window processes of two competing TCP sources. $C = 0.5$ Mbps, $d = 10$ ms, $B = 4$ packets.

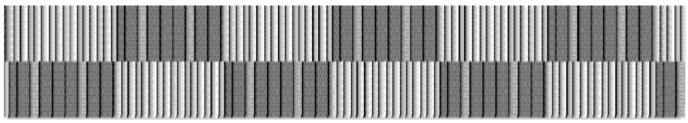


Figure 2.5: Spatio-temporal graph: $C = 0.5$ Mbps, $d = 10$ ms, $B = 4$ packets (the same configuration was used as for Figure 2.4).

complete set of system variables and the value of *cwnd* are not continuous. This means that it is not possible to efficiently visualize periods for complex behavior, because only a limited number of points will be touched, thus meeting and then diverging trajectories are indistinguishable. Another problem is that the congestion window process at a certain time instant does not reveal the underlying state of the system in a comprehensive manner.

In [PCFS80] the authors propose to use the time shifted past values $[x_t, x_{t-\delta t}, x_{t-2\delta t}, ...]$ of an easily measurable quantity for complex systems to equivalently reconstruct the underlying multidimensional trajectories if there is no access to the state variables. The choice of δt can be nearly arbitrary in a wide range. The result is a multidimensional vector that is projected to the 2D plane simply by averaging the values $\hat{X} = 1/n(x_t + x_{t-\delta t} + ...)$. The method described above is used for the *cwnd* values:

$$x[i] = \frac{1}{n} \sum_{j=1}^{n} \text{cwnd}_x[i-j] \tag{2.1}$$

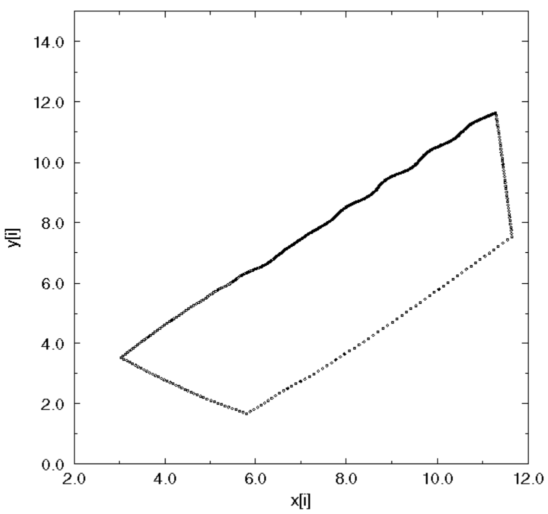
$$y[i] = \frac{1}{n} \sum_{j=1}^{n} \text{cwnd}_y[i-j] \tag{2.2}$$

Here x and y denote the two TCPs. n controls the scale over which the congestion windows are averaged, the larger the value is, the more hidden dimensions can be reconstructed. The method has two other benefits:

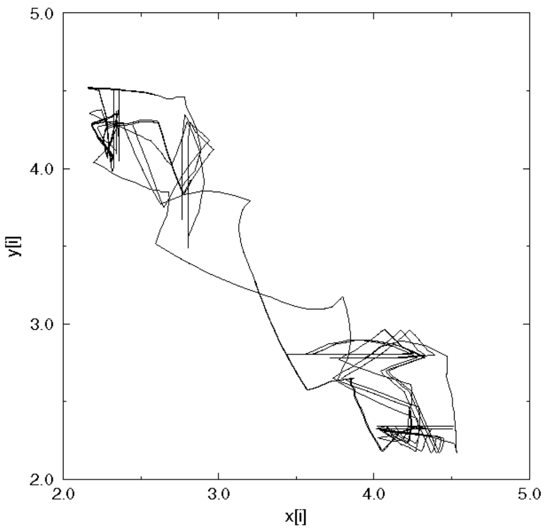
- The number of possible points on the (x, y) plane is increased from W^2 where W is the number of possible *cwnd* values to $(nW - n)^2$ (if *cwnd* is counted in packets).
- Consecutive results $x[i]$ and $x[i+1]$ are placed close to each other, actually no further than $(2 * W - 2)/n$. Thus using this construction the generated graph can be made as smooth as required.

A nice property of the graph is that it preserves the periodicity property: periodic trajectories are displayed as closed loops in (x, y). (If we choose n equal to the period, then we get a single data point - a degenerate loop.)

Figure 2.6 shows the periodic trajectories of the simple (staircase) and the more complex (alternating) periodic systems discussed so far. The alternating behavior can be easily observed on the right graph. Both systems are represented as closed loop, which is a sign of periodicity, but the complexity of the two loops is significantly different. The "staircase" system has a nice simple loop, while the "alternating" system has a more complex but more or less symmetric trajectory. Although being very complex, both trajectories prove to be very stable: disturbing the system, (e.g., changing the relative starting times of TCPs thus giving gain to one of them or perturbing the congestion control by artificially changing the size of the *cwnd*), does not destroy the trajectory. After a short detour both systems revert back to the same regular pattern. Such trajectories are called *attractors*. An attractor is a set of points to which nearby trajectories are attracted to. A more formal definition of an attractor can be found in [PJS92].



(a) $C = 0.2$ Mbps, $d = 10$ ms, $B = 20$ packets, time shifts $n = 100$

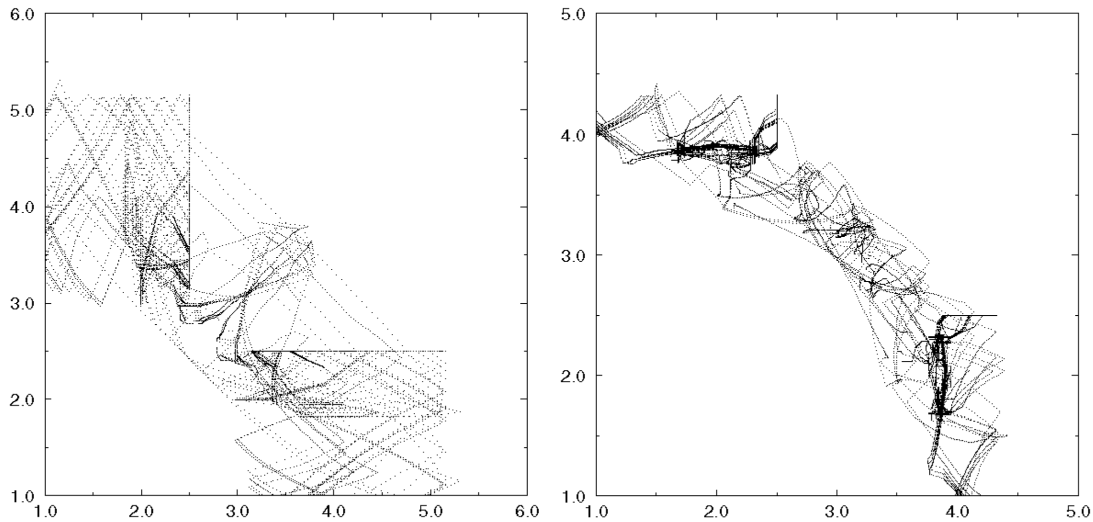


(b) $C = 0.5$ Mbps, $d = 10$ ms, $B = 4$ packets, time shifts $n = 90$

Figure 2.6: Periodic attractors of two competing TCP sources.

2.4 Strange Attractors

For certain parameter sets (i.e., the number of competing TCPs, service rates, buffer sizes or transmission delays) the system exhibits simple behavior, and for other sets, very complex behavior. It is not difficult to find parameters, where the system seems to never repeat itself. Such a parameter set is $C = 0.1$ Mbps, $d = 10$ ms, $B = 4$ packets. Of course as the variables of the system are discrete, the trajectories will always be periodic, but the size of the period is extremely large. The attractor of this system is displayed in Figure 2.7. The structure of the trajectory is very fine and the result represents four hours of simulation.



(a) Time shifts $n = 100$

(b) Time shifts $n = 300$

Figure 2.7: Strange attractor. $C = 0.1$ Mbps, $d = 10$ ms, $B = 4$ packets.

A very interesting experiment can be made to show the fine structure of the attractors presented previously, namely we show that the projection of the attractor has fractal dimension. When one considers a simple drawing on a two-dimensional plane (e.g., a loop), this can easily be measured to determine its length. This is not the case for fractal structures whose length depends on the size of the unit used for the measurement [PJS92]. The dimension of such an object is represented by a non-integer value. We can, for example, measure the attractor's box-counting dimension D, which is done in the following way: choose a grid on the plane of size s, then count the number of boxes which have a part of the object in it. Denote this number as $N(s)$. Then change s to cover several scales (e.g., half it each step). Finally, plot the values $log(N(s))$ versus $log(1/s)$. If the object is fractal, then $log(N(s)) \sim Dlog(1/s)$, where D is a non-integer value. In practice one should fit a least-squares regression to the logarithmically spaced sample points s_i and calculate its slope. As the graph is made up of a finite number of points, the final dimension is 0 so the last few values are dropped, and also the first few values of s_i are omitted because they are in the scale of the object itself. In between this lies the area where the fractal property holds. Figure 2.8 shows the calculated fractal dimension of the "staircase" and the "nonperiodic" trajectories. The fitted regression follows the plot through 4-5 magnitudes, which supports the significance of the measurements.

The simple periodic "staircase" system has an attractor with dimension $D \approx 1$ which equals a simple 1 dimensional line and so it is not fractal. In the case of the "nonperiodic" system we can calculate $D \approx 1.61$, which is significantly different from 1, but below 2 - the attractor is a fractal. Attractors with fractal properties are called *strange attractors*. Furthermore, if a system

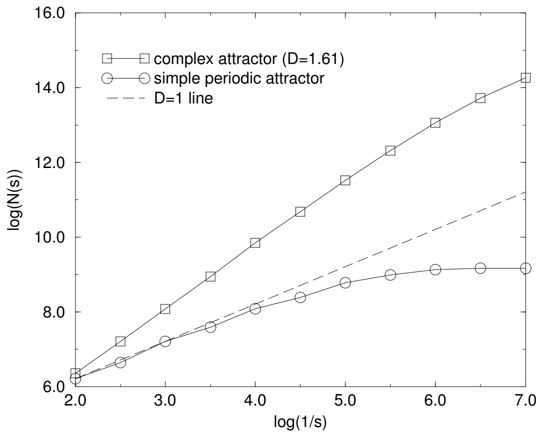


Figure 2.8: Box-counting dimension of the simple periodic "staircase" and "nonperiodic" trajectories (time shifts $n = 300$ was used for the "nonperiodic" and $n = 100$ for the "staircase" trajectory).

shows sensitivity to the initial conditions, then the corresponding attractor is called a *strange chaotic attractor*.

Note that there exist chaotic systems with non-fractal attractors and strange attractors of non-chaotic systems.

2.5 Sensitivity to Initial Conditions

In this section, we demonstrate that in certain cases TCP congestion control is prone to large sensitivity to initial conditions, which means that very small perturbations in the system may cause that the trajectory departs from the original, unperturbed, system's trajectory within a very short time. The distance can grow to the range of the signal itself. This is one of the major properties of chaotic systems.

In the following experiment we increase the number of simultaneous TCP sessions to 30. ($C = 1$ Mbps, $d = 15$ ms, $B = 60$ packets). First, we let the system evolve for a while, then at $t = 50$ s we artificially increased the congestion window of one of the TCPs with one packet. Then we plotted the spatio-temporal graph of both systems, see the original system in Figure 2.9 (top) and the perturbed system in Figure 2.9 (middle). The length of the plot is 100s so the perturbation is done right at the middle of the plot. If we compare the two systems, first the differences are invisible, but a few seconds later the two systems look completely different. To make this more visible, we plotted the difference of the two systems in a way that each dot was colored according to the distance defined as $d(i,t) = |w^{orig}(i,t) - w^{pert}(i,t)|$, where i and t is the id. of the TCP and the time respectively, $w^{orig}(i,t)$ is the *cwnd* of ith TCP in the original system at time t and $w^{pert}(i,t)$ is the same for the perturbed system. See Figure 2.9 (bottom). The first part is white, which means that the two systems are identical, then a few dim dots appear and a few seconds later the difference looks like the original plots themselves.

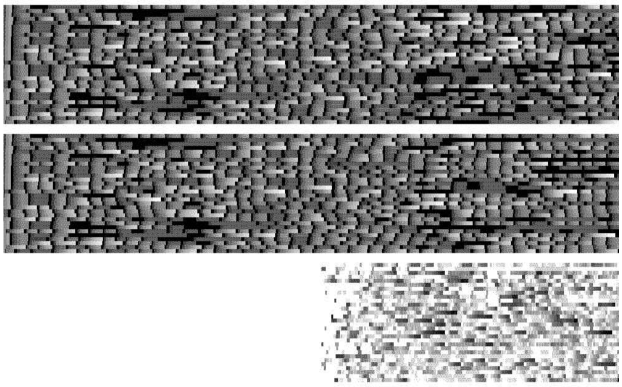


Figure 2.9: Spatio-temporal graph of the original system (top). Spatio-temporal graph of the perturbed system (middle). Difference between the two systems (bottom).

To quantify how fast this divergence happens, we define the distance between the two systems at time t as the Euclidean distance in the *cwnd* space:

$$E(t) = \sqrt{\sum_{i=1}^{N} (w^{orig}(i,t) - w^{pert}(i,t))^2}. \tag{2.3}$$

See Figure 2.10.

The rate at which the systems diverge after a small perturbation of the ith TCP ϵ_i at time t_0 can be described by the so called *Lyapunov exponent*, which we approximate by measuring the time Δt it takes for the two systems to reach a given distance $E(t_0 + \Delta t) > \hat{E}$, then:

$$\lambda(t_0, i) \approx \frac{1}{\Delta t} \ln |\frac{E(t_0 + \Delta t)}{\epsilon_i}| \tag{2.4}$$

For the experiment we chose $\hat{E} = 10$ and $\epsilon_i = 1$. The motivation to calculate an *exponent* is that trajectories diverge at an exponential rate. However, as the two systems cannot get arbitrarily far from each other (as the phase space is limited) only the increasing part of Figure 2.10 should be considered, this explains the choice of $\hat{E} = 10$.

The Lyapunov exponent is the rate at which the two systems diverge from each other every time unit. This value of course depends on *which* TCP we perturb and *when* the interference is

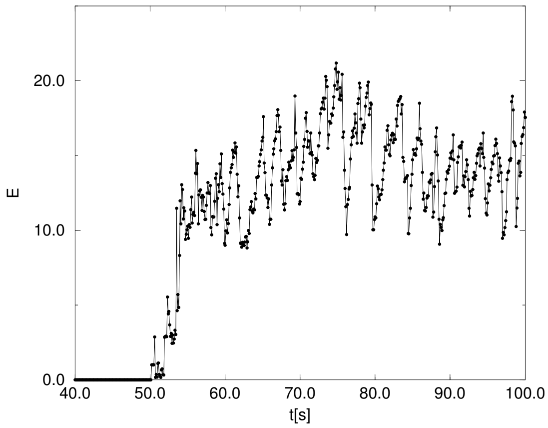


Figure 2.10: Divergence of the original and the perturbed systems.

done. In other words, to which direction in phase space we push the system and at what part of the phase space the system is at the time of perturbation. There are cases when even an otherwise sensitive system is not affected by a small interference. A negative exponent characterizes this case. We note that our approximation can be used for positive exponents only. There is no generally accepted definition when an attractor is called chaotic, but we can say that if sensitive points ($\lambda > 0$) are dense on the trajectory then the attractor is chaotic.

To arrive at a more general numerical result for a given system we calculate the exponent at many different points of the trajectory (t_0) and for all 30 TCPs. Then for each t_0 we choose the most sensitive direction where the largest λ is measured and average these values over time to get the average maximum exponent of the trajectory:

$$\lambda = E\left[\max_i \lambda(t_0, i)\right] \tag{2.5}$$

See Figure 2.11. In the experiment we got $\lambda \approx 1.11$, which means that after a perturbation the difference between the two systems increase at an average rate of $e^{\lambda} \approx 3.03$ every second.

2.6 Testing for Self-Similarity of the Time Series

In this section, we show that a system of competing TCPs with certain parameters generates second-order self-similar traffic over several timescales ranging from a few round-trip times to hundreds of seconds. A number of statistical tests were performed to search for scaling behavior: absolute values method, wavelet analysis, periodogram and R/S method [TTW95].

Not all configurations produce self-similarity, we have observed that it is mainly the B/N ratio that controls the behavior of the system. As the ratio decreases (i.e., the ratio of the 'pipe' for one TCP flow becomes smaller), the system goes through a phase transition from periodic to chaotic behavior, and for certain parameters it produces a self-similar time-series.

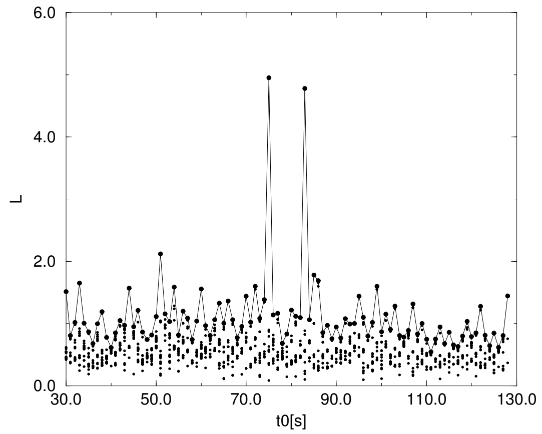


Figure 2.11: Lyapunov exponents at different points of the trajectory and for all 30 TCPs, maximum exponents along the trajectory are connected with a line (average maximum $\lambda \approx 1.11$).

The simulation setup is as follows: $C = 1$ Mbps, $d = 15$ ms, $B = 20$ packets and $N = 40$ TCPs. Note that the buffer can store less packets than the number of active flows. This results in a packet loss ratio of around 16%, creating a strong bottleneck on the path. It is argued in [Mor97] that such small pipes substantially contribute to the performance of the current Internet.

During simulation the amount of bytes sent by all TCPs is logged individually every 0.1 s. The packet trace represents a few hours duration and consists of approximately 1.6 million data packets, allowing us to perform tests on sufficiently large timescales. Figure 2.12 shows the trace of one-TCP microflow filtered out from the trace.

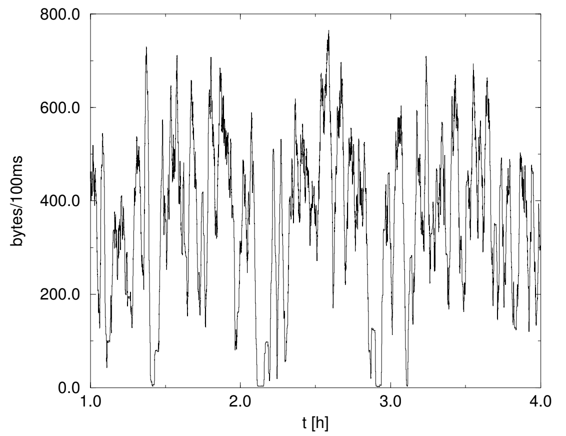
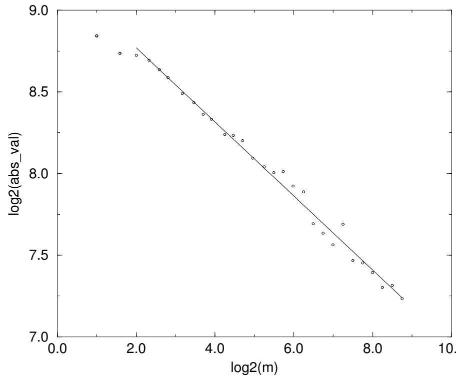
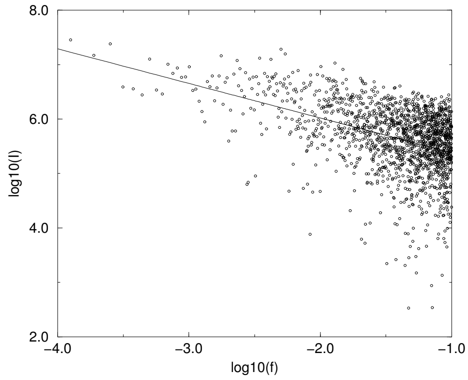


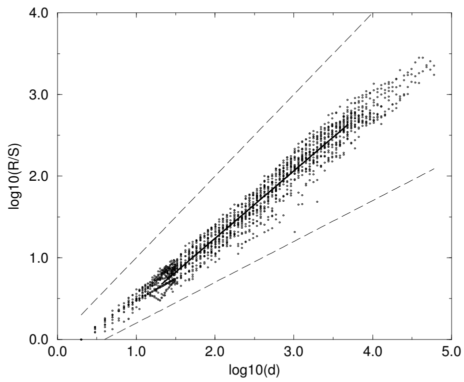
Figure 2.12: Time series of sent bytes by a one-TCP microflow (running average of $\tau = 100$s.)



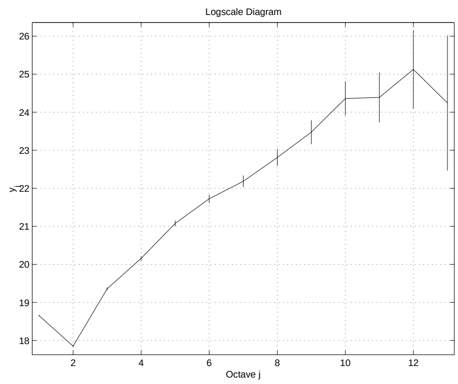
(a) Absolute values method $H = 0.79$.



(b) Periodogram method $H = 0.815$.



(c) R/S method $H = 0.813$.



(d) Wavelet method $H = 0.787$, $(0.754, 0.819)$.

Figure 2.13: LRD tests for one-TCP microflow.

Consider a weakly stationary stochastic process X, with constant mean, finite variance and autocorrelation function $\rho(k)$. Let $X^{(m)}(k) = 1/m \sum_{i=(k-1)m+1}^{km} X(i)$ denote the m aggregated series of X. The process X is called *exactly self-similar* if for all m it satisfies $X =_d m^{1-H} X^{(m)}$. X is said to be *asymptotically self-similar* if this property holds as $m \to \infty$. Furthermore, X is *second-order self-similar* if $m^{1-H} X^{(m)}$ has the same variance and autocorrelation as X. If this holds asymptotically, then X is *asymptotically second-order self-similar* [LTWW93].

The tests were conducted over the time series of the amount of bytes sent by a one-TCP microflow and on the aggregate quantity sent by all TCPs as well. The results support that traffic of one-TCP microflow is consistent with asymptotic second-order self-similarity with $H > 0.5$. On the other hand, aggregate traffic of all TCPs is short-range dependent with $H \approx 0.5$.

The first test is based on the behavior of the expectation of the absolute values of the series

$X^{(m)}$ [TTW97, TTW95]

$$\mu^{(m)} = \frac{1}{N/m} \sum_{k=1}^{N/m} \Big| X^{(m)}(k) - \frac{1}{N} \sum_{i=1}^{N} X(i) \Big|. \tag{2.6}$$

If X is self-similar, then on a log-log plot the absolute values increase linearly with m, with a slope of $H - 1$. Figure 2.13(a) shows the result for the absolute values method ($H = 0.79$).

The periodogram method approximates the spectral density of the process with the expression

$$I(\lambda) = \frac{1}{2\pi N} \Big| \sum_{j=1}^{N} X_j e^{ij\lambda} \Big|^2$$

For a LRD series the spectrum is $I(\lambda) \sim |\lambda|^{1-2H}$ at the origin. Figure 2.13(b) shows the result of the periodogram method for one-TCP microflow, resulting in an approximation of $H = 0.815$.

The rescaled adjusted range statistics (R/S) [TTW95] and the wavelet analysis method [AbVe98] are not described here, only the results are depicted in Figure 2.13(c) (R/S: $H = 0.813$) and Figure 2.13(d) (Wavelet: $H = 0.787$, with 95% quantiles at $[0.754, 0.819]$).

All methods resulted in a Hurst exponent of $H \approx 0.8$.

Aggregate traffic entering the bottleneck buffer shows a significantly different behavior. See Figure 2.14 for the result of the R/S method performed on aggregate traffic. The results of other methods are not shown here. However, all tests support that the Hurst exponent of aggregate traffic is around $H \approx 0.5$ (i.e., it is short range dependent).

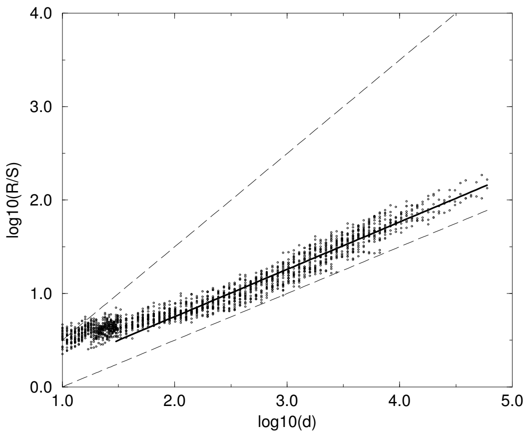


Figure 2.14: R/S method for aggregate traffic. $H = 0.51$

The aggregate effect of multiple congestion control algorithms smoothes the aggregate rate passing through the bottleneck buffer. Nevertheless, individual TCP flows, within the aggregate, still become long-range dependent, as shown by the Hurst parameter of individual TCP flows.

How can one then measure $H > 1/2$ or long-range dependence for aggregate network traffic? There are two possible explanations:

- The effect of TCP congestion control is mixed by higher layer protocols with heavy tailed properties (e.g., WWW file sizes, waiting times) [CrBe96]; or
- Individual long-range dependent TCP flows exit the bottleneck buffer and enter other non-bottleneck buffers. There they mix with other flows coming from other buffers.

Assume that we have a number of LRD processes X_is with autocovariance function given in the form

$$\gamma_i(t) = a_i t^{-\delta_i} \tag{2.7}$$

If $\delta_i = 2 - 2H \leq 1$ (or $H \geq 1/2$), this process is long-range dependent, on the other hand, if $\delta_i > 1$, it is short-range dependent. Let us assume that if we multiplex such streams coming from different bottleneck buffers in a non-bottleneck buffer then they remain independent. The autocovariance function of multiplexed traffic is then

$$\gamma(t) = \sum_i \gamma_i(t) = \sum_i a_i t^{-\delta_i} \sim t^{-\min \delta_i} \quad \text{as} \quad t \to \infty \tag{2.8}$$

In other words, long-range dependent traffic remains long-range dependent in a non-bottleneck buffer, and the largest exponent characterizes it. This way TCP aggregates can show long-range dependence. This effect does not happen in the bottleneck buffer because there the assumption of independence does not hold.

2.7 Periodic-Chaotic Transitions in Real Networks

In this section, we show that periodic and non-periodic behaviors are present not just in simulation models but also in real networks. Three different configurations are analyzed, demonstrating stable periodic behavior, sensitive behavior when the system is on the edge of chaos, and non-periodic chaotic behavior.

For the real network tests we used four computers, all of them running the Linux RedHat 6.2 operating system. As shown in Figure 2.15, the four hosts shared a single, 10 Mbps Ethernet segment. Logically the Ethernet segment is divided to two IP subnets, hosts A and B are in subnet 1 and host C in subnet 2. Host D is configured to act as a router between the two subnets. We implemented and configured a precise shaping buffer on the interface of the router connecting to subnet 2. The speed and buffer size is configurable and the accuracy of the packet processing is within 1 ms.

During the experiments, host C downloads very long files from hosts A and B at the same time. As a result, the two parallel TCP connections have to share a single low speed link represented by the shaper. Similarly, as in the previous sections, we use the evolution of congestion windows to analyze the system. However, because it is difficult to access the *cwnd* variable in a real operating system without modifying the kernel code, we estimate its value by the number of outstanding and unacknowledged packets at the point of the interface of the sending host, in this experiment in hosts A and B, respectively.

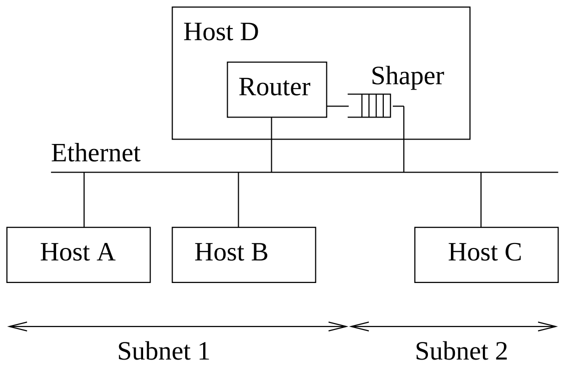


Figure 2.15: Test network configuration.

2.7.1 Stable Periodic Regime

In the first configuration, the shaping speed is set to 5 packets per second, or 60 kbps (MTU is 1500 bytes) and the buffer size to hold 10 packets. A few seconds after the TCP transactions is started, the system settles down to a periodic pattern. To demonstrate the stability of the pattern, we disturb the system by starting a third short TCP connection, which downloads approximately 120 kbytes. The evolution of the congestion windows are shown in Figure 2.16. The top two graphs show the $cwnd$ of the long TCP connections, while the third graph shows the $cwnd$ of the perturbing short TCP. It can be seen that after the perturbation has ended, the system returns to the same periodic pattern within a short time.

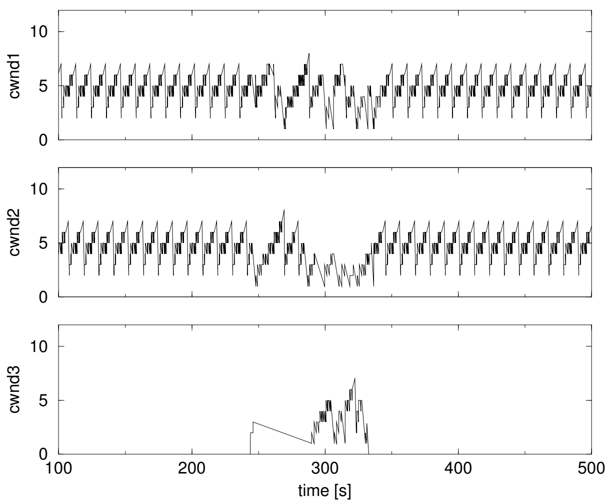


Figure 2.16: Stable periodic system. After a short perturbation caused by a third TCP connection (bottom), the periodic pattern returns. Shaping speed 60 kbps, buffer size 10 packets.

The periodicity and the effect of the perturbation on the system can be demonstrated by displaying the trajectory of the system in the state space. The left graph of Figure 2.17 shows the trajectory before the perturbation. It can be seen that the periodic behavior is represented by a closed loop. On the right hand side, the trajectory of the system is shown right after the perturbation is started. The effect of the perturbation is represented by a short detour off the closed loop, but the system returns to the same pattern as in the left graph. A negative Lyapunov exponent characterizes such a system.

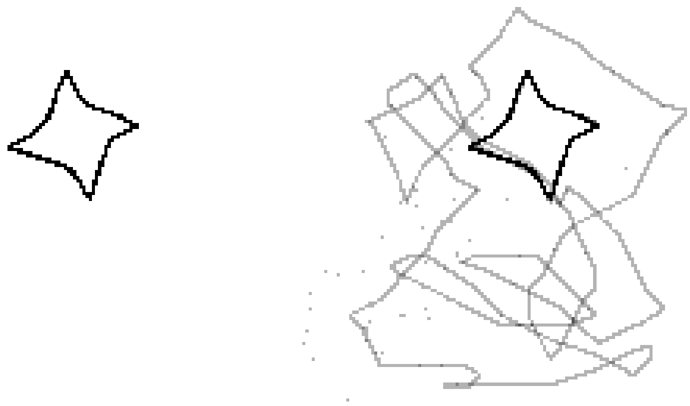


Figure 2.17: Trajectories of a stable system. Averaging window $w = 6\ s$. a) Periodic behavior is represented by a closed loop. b) After a perturbation the system returns to the loop.

2.7.2 On the Edge of Chaos

For certain parameters the system becomes more sensitive, and stable periodic trajectories become less stable. When the system is in this regime, it may switch states without any obvious external perturbation.

During the experiment the shaping speed is set to 20 packets per second or 240 kbps and the buffer size to 20 packets. Figure 2.18 shows that initially the two-TCP system settles down in a periodic state, but after a few minutes, they leave the periodic pattern and enter a non-periodic state. The two TCPs settle down again into a periodic pattern, but only after approximately 5 minutes. The new periodic state proves to be unstable, and it is maintained for a few minutes only.

We call this behavior "the edge of chaos" when the system switches between periodic and non-periodic behavior intermittently. The cause of these state changes can be attributed to small perturbations in the operating systems, the inaccuracy of the shaper and perhaps infrequent collisions on the Ethernet segment.

Using the technique introduced in Section 2.3 to reveal attractors, we get Figure 2.19a. The non-periodic behavior completely hides periodic trajectories; so we modified the plotting program to highlight those parts of the trajectory that are more frequently visited. Although random effects push the system away from its attractors, this technique can still help spotting orderly behavior. The result is shown in Figure 2.19b, two sharply distinguishable closed loops indicate

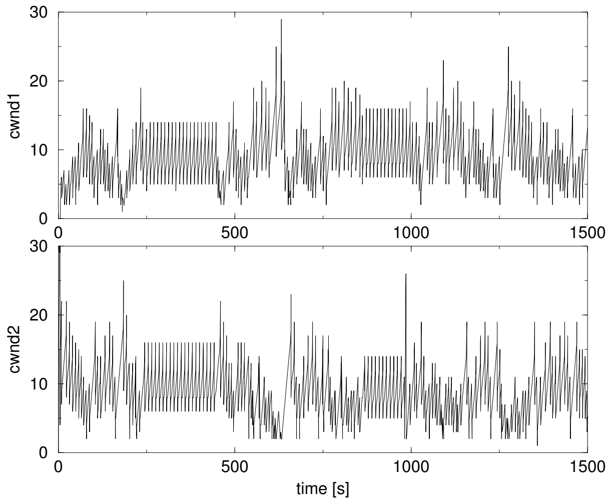


Figure 2.18: Intermittent behavior on the edge of chaos. Shaping speed 240 kbps, buffer size 20 packets.

two periodic attractors.

The observed periodic behavior indicates that the system still has negative Lyapunov exponents, but the attraction is weak, and small perturbations can push the system away from the attractor and even to the *basin of attraction* of another nearby periodic attractor. The *basin of attraction* is the set of initial points in phase space that are drawn to the attractor [Ott93].

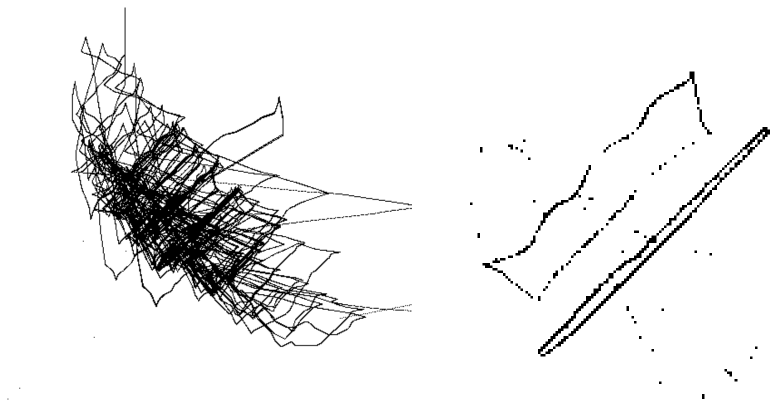


Figure 2.19: Trajectory of the intermittent system. Averaging window $w = 3\ s$. a) Original trajectory. b) Filtering reveals periodic trajectories (magnified).

2.7.3 Chaotic Regime

For certain parameters the system does not show any seemingly ordered behavior, as similarly demonstrated by simulation in Section 2.4. Such systems have positive Lyapunov exponents and non-periodic attractors. Because of extreme sensitivity it is almost impossible to reveal any order in the trajectory of a real-life system prone to uncontrollable, random perturbations. An additional challenge is that the techniques we use are limited to the observation of the congestion window.

During the experiment the shaping speed remains at 20 packets per second or 240 kbps, as in the previous experiment, but the buffer size is reduced to 10 packets. The test results consisting of $cwnd$ logs shows no obvious periodic or any other orderly behavior. After applying the filtering technique on a 3600 s long logfile using different settings, we could not observe any periodicity either. Although non-periodic, we could still observe certain non-obvious, hidden orderly behavior. First, the logfile is split into three consecutive parts, 1200 s long each, then the trajectories were plotted using the highlighting technique. The resulting graphs of the three time periods are shown in Figure 2.20. The three figures show certain resemblance to each other (e.g., similar loops appear on the trajectories 1200 s apart). We have to underline that these similarities do not prove that the system returns to the same state as before, but obviously certain parts of the state space are more preferred by the system than others.

Figure 2.20: Trajectories of a two-TCP system, using the highlighting technique. Averaging window $w = 0.3\ s$. a) 0-1200 s, b) 1200-2400 s, c) 2400-3600 s.

2.7.4 Self-Similar Regime

The traffic traces obtained from the previous configuration, although chaotic and seemingly random, do not pass the tests of self-similarity. We found that, exactly as the simulation in Section 2.6 suggested it, self-similarity arises when the number of TCPs increases in relation to the size of the end-to-end pipe.

We found that if the total number of parallel TCPs is increased to four (i.e., both hosts A and B start two parallel TCPs) then the traffic of individual hosts and also the traffic of individual TCP microflows become self-similar with $H > 0.5$. Figure 2.21 shows the R/S test for the traffic generated by host A. The base time of the measurement was 1 s. The estimated Hurst exponent obtained from the test was $H = 0.83$.

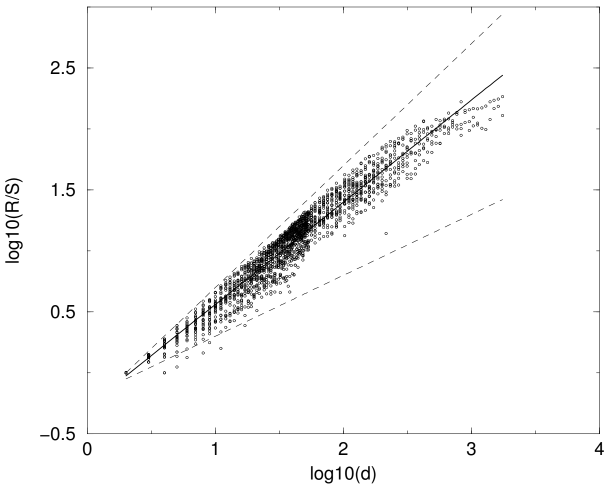


Figure 2.21: R/S test of a one hour long traffic trace generated by host A. Base time 1 s. Estimated $H = 0.83$.

The recent paper by Guo et al. [GCM00] analyses a similar scenario using a Markov chain based model and arrives at a similar conclusion. The authors derive that self-similarity can arise in TCP traffic if the blocking probability reaches a certain value. Their arguments are consistent with our observations, since small buffer/TCP ratios induce higher loss rates. The work by Figueiredo et al. [FLMT00] further develops the model to include the congestion avoidance phase and not just the exponential backoff phase of TCP congestion control.

The significant difference between their methodology and ours is that both papers approach the same problem from a stochastic perspective. Our work, although it takes a significantly different, deterministic modeling approach, actually supports the validity of the approach discussed in [GCM00] and [FLMT00]. The complex, chaotic nature of TCP blurs the borderline between deterministic and stochastic techniques in traffic modeling. On the one hand, it makes it possible to use either stochastic or deterministic modeling techniques in the analysis of a system, but it calls for extra care to be taken before drawing conclusions.

2.8 Conclusions

In this chapter, we demonstrated how TCP can produce for certain parameters both simple and very complex behavior. We have shown that for certain parameters TCP behaves chaotically and that the main properties of chaos are present in TCP. We demonstrated that TCP congestion control could create self-similar traffic with Hurst exponents showing both short-range and long-range dependence depending on system parameters. This property is more fundamental than the second order self-similarity property reported before in the literature because it is the property of a low level deterministic system (TCP) itself, regardless of the applications running on top of

TCP.

We also demonstrated some of the above behavior in an experimental network. For different configurations we observed stable periodic, and chaotic regimes, and also the transition phase between them. The conventional paradigm that traffic sources are treated separately from the network has major drawbacks; that is, it is not possible to disregard the network status when we create traffic models. Another important implication is that the randomness observed in TCP/IP networks originates from chaotic deterministic mechanisms and not just stochastic effects.

Deterministic chaotic models offer new tools for modeling TCP/IP traffic. Some of these techniques are applied and demonstrated in this chapter; however, many more can be applied from other fields of science where chaos theory has been used. We hope that the results of this chapter contribute to a better understanding of TCP/IP traffic modeling.

There are many open questions and problems not covered in this chapter. For example, how to unite stochastic models of higher layer mechanisms (e.g., heavy tails) with the chaotic model of TCP. One of the most promising future directions is to reuse chaos control techniques already applied successfully to control lasers or heart beats. Chaos control techniques, which rely on the butterfly effect and partial reconstruction of the system attractor, operate by minute perturbations of the system. If these methods are applied to TCP/IP networks, possibly, the system could be effectively controlled with minimum intervention, consequently, we speculate, better throughput, fairness or delay could be achieved.

The next chapter analyzes another aspect of TCP congestion control, namely end-to-end path adaptation. Together with competition, which was discussed in the chapter, adaptation is the other major feature of the TCP protocol that significantly impacts the traffic dynamics in TCP/IP networks.

Chapter 3

TCP's Role in the Propagation of Self-Similarity in the Internet

In Chapter 2 we discussed a new, chaotic modeling approach to TCP modeling. We found that TCP dynamics can cause not only seemingly random traffic fluctuations, but, for certain parameters it can generate statistically self-similar time series. This finding complements previous research, which explained the emergence of self-similarity with stochastic reasons, for example heavy tailed distributions in higher layers of the TCP/IP protocol stack [CrBe96] [CTB96] [PaFl95] [TWS97] [WTSW97].

In this chapter we discuss another role of TCP in the wide-scale emergence of self-similarity in the Internet. We show that TCP, apart from generation, can also propagate self-similarity between distant areas in the Internet. This means that even if there are no reasons for self-similarity at a certain point of the network, for example, heavy-tailed file sizes or chaotic competition, it is still possible that traffic fluctuations become self-similar due to the propagation effect. We found that TCP propagates any kind of self-similarity, regardless of how it is generated. As a matter of fact, it is shown that TCP propagates other kinds of correlation structures as well, not just long-range dependence. In this chapter we also analyse the impact of self-similarity on end-to-end TCP dynamics, how the end-user perceived rate fluctuations depend on the end-to-end path properties, and what are limitations of propagation.

TCP uses an end-to-end congestion control algorithm to continuously adapt its rate to perceived network conditions. If network conditions are governed by large timescale fluctuations, then TCP will "sense" this and react accordingly. Our work demonstrates that TCP adapts to traffic rate fluctuations on several timescales efficiently. Moreover, we show that TCP can be modeled as a linear system above a characteristic timescale of a few round-trip times, which implies that the correlation structure of a background traffic stream is reproduced faithfully by an adaptive TCP flow. In particular, it is shown that *TCP can inherit self-similarity from a self-similar background traffic stream*. Since TCP has an end-to-end control, while adapting to these fluctuations, it *propagates self-similarity* encountered on its path all along from the source to the destination host.

We also demonstrate that if a TCP stream is multiplexed with another one, it can pass on self-similar scaling to the other TCP stream, depending on network conditions. In our model the network is regarded as a mesh of end-to-end adaptive streams. Intertwined TCP streams can *spread self-similarity* throughout the network contributing to global scaling. By analyzing the effects from a network point of view we argue that, on one hand, TCP plays an important role in balancing and propagating global scaling and on the other hand, it keeps local scaling intact where it is already strong. Our work complements results reported in [FGHW99]. The main purpose of this chapter is to analyze the basic mechanisms behind these phenomena.

The chapter is organized as follows. A wide-area TCP measurement is analyzed showing self-similar scaling for the traffic of a single long TCP connection, and a possible explanation is presented based on a few simple assumptions in Section 3.1. Section 3.2 investigates how TCP adapts to fluctuations on different timescales, and it is shown that TCP in a bottleneck buffer can be modeled as a linear system above a characteristic timescale of a few round-trip times. In Section 3.3 we investigate how an aggregate of TCP sessions with durations of heavy-tailed and light-tailed distributions propagates self-similarity of a background traffic stream. Finally, in Section 3.4, we present results about the spreading of self-similarity in the network case when TCP has to pass multiple hops and compete for resources with other TCP streams.

3.1 Adaptivity of TCP: a Possible Cause of Widespread Self-Similarity

In this section, we demonstrate how TCP adaptation leads to the propagation of LRD, using wide area network measurements. First, the test methodology is explained using a single measurement, and then the test results of several other WAN TCP measurements are presented. Finally, we introduce a simple analytic model of TCP adaptation.

3.1.1 Scaling Analysis of Wide Area TCP Measurements

During the first experiment a large file was downloaded (a traffic trace file from the Internet Traffic Archive) from an FTP server (*ita.ee.lbl.gov*) to a client host 15 hops away in Hungary (*serv1.ericsson.co.hu*), passing several backbone providers and a trans-Atlantic link. At the client side there was no other traffic present. The client was directly connected to an ISP by a 128 kbps leased line, which was the narrowest link on the path. All packets were captured at the client side with the *tcpdump* utility. The total amount of bytes received was 50 Mbyte and it was logged with a resolution of 50 ms during the file transfer for 6900 s. The average throughput, which takes into account the retransmissions and the TCP/IP overhead, was about 58 kbps (i.e., some congestion was experienced in the network). The average round-trip delay between the server and the client was 208 ms. From the packet trace we concluded that the version of the TCP was Reno.

Tests were performed for the presence of self-similarity. Here we present three tests: the first and second ones are based on the scaling of the absolute moments (also called absolute mean and variance-time plots [TTW95]), and the third one is a wavelet-based analysis [AbVe98]. All three tests are illustrated in Figure 3.1. The result of the tests suggests asymptotic self-similarity with Hurst parameter around 0.72.

We performed a wavelet based test developed by Veitch [VeAb99] to test if the scaling exponents are constant throughout the trace. The test also spots other types of non-stationarity.

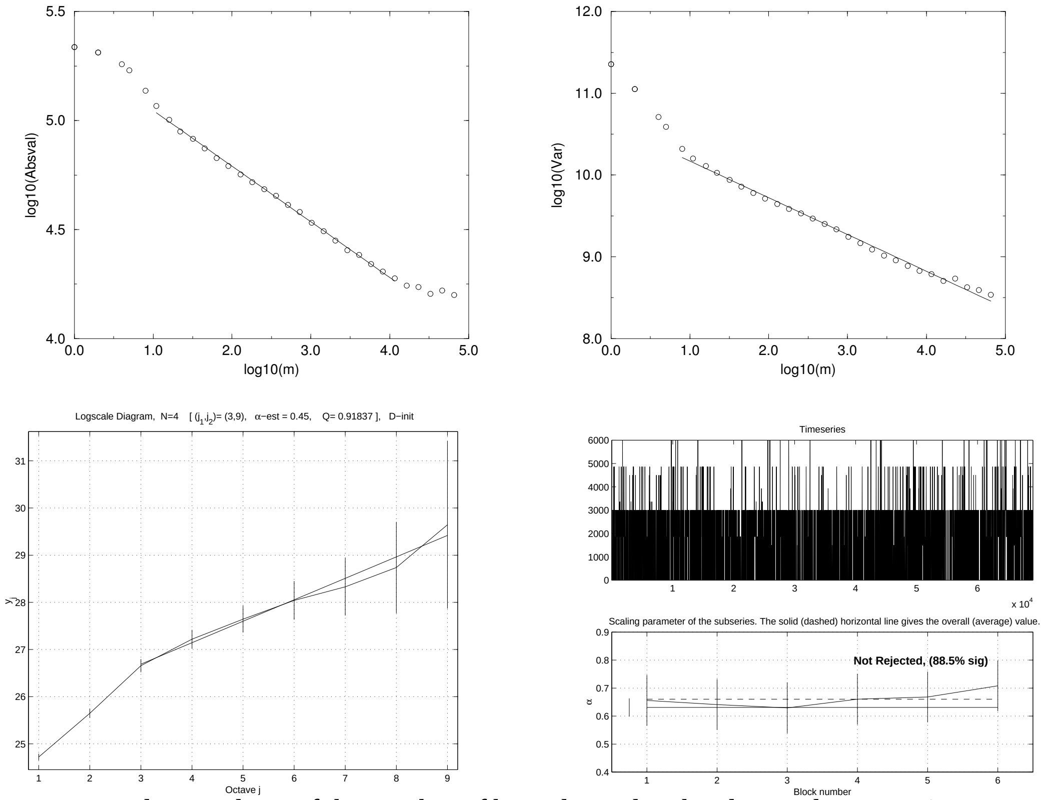


Figure 3.1: Scaling analysis of the number of bytes logged at the client side every 50 ms. Parameter m denotes the scale of aggregation. a) Absolute mean method $H \approx 0.76$. b) Variance-time plot $H \approx 0.77$. c) Wavelet analysis $H \approx 0.72$ [0.68, 0.77]. d) Wavelet based analysis of stationarity.

First the trace is cut into several non-overlapping sections. Then the sample mean and variance and also a wavelet based scaling analysis are performed for each period. Finally, the resulting statistics are compared with each other and against the overall values. If the values are not within the confidence intervals, the trace is not considered to be stationary. Based on results it is reasonable to accept the hypothesis that the time series is stationary and it has a constant scaling parameter H.

During the experiment, there was only one connection active on the link, so explanations based on the superposition of heavy-tailed On/Off processes or chaotic behavior [C2] are not applicable *locally*. However, the investigated TCP connection traversed several backbone links where, due to the large traffic aggregations, self-similarity could arise either because of heavy-tails or chaotic competition. Presumably, whatever the reason for self-similarity was, the TCP connection adapted to the background traffic stream at the bottleneck link, and the effect of the adaptation was that self-similarity was propagated to the measurement point.

We repeated the measurement between different hosts in the Internet, covering very different networking situations. Because the scaling region has to cover at least two orders of magnitude

server	bin size	duration	avg. rate	VT-H	Wavelet-H
fr3	100 ms	1200 s	471 kbps	0.77	0.874 [0.808, 0.940]
uk	100 ms	1800 s	115 kbps	0.66	0.704 [0.621, 0.786]
au	100 ms	3346 s	566 kbps	0.8	0.94 [0.854, 1.038]
leasjan28	10 ms	3591 s	131 kbps	0.8	0.95 [0.875, 1.019]
leasjan26	100 ms	1000 s	154 kbps	0.82	0.855 [0.782, 0.929]
hu	50 ms	6900 s	58 kbps	0.77	0.72 [0.68, 0.77]
cablejan	10 ms	3601 s	474 kbps	0.85	0.831 [0.759, 0.903]
us3	10 ms	495 s	1.4 Mbps	0.53	0.562 [0.458, 0.666]
usFeb3	10 ms	1868 s	552 kbps	0.5	0.511 [0.466, 0.556]
usRog	10 ms	2115 s	98 kbps	0.89	0.889 [0.848, 0.931]

Table 3.1: Summary of WAN measurements

for the scaling analysis, the measurements have to be long enough, at least several hundred seconds long. During such relatively long measurements, daily trends or other types of non-stationarity may disturb the tests. Stationarity analysis was performed to filter out and disregard such traces. As shown in [FLMT00] and [GCM00], if the packet loss probability exceeds a certain value, it may also cause TCP traffic to become LRD eeven if background traffic is SRD and packet losses happen independently. Although this phenomenon may be very important in the emergence of LRD in TCP traffic in highly congested links, since we are interested in TCP adaptation, we also disregarded traces with excessively long timeouts.

The basic statistics of the traces that passed the above requirements are shown in Table 3.1. The first group of measurements consists of "trans-Atlantic" measurements between hosts in Europe and Columbia University, New York. In this group we have traces *fr3* (France), *uk* (UK), *au* (Australia). The next three measurements are also trans-Atlantic measurements, but in these cases the end-hosts were connected via relatively small speed leased lines, in the case of *hu* the speed of the leased line was 128 kbps, while *leasejan26* and *leasejan28* were done on a 256 kbps leased line of a small home-ISP, all in Budapest, Hungary. The next measurement, *cablejan*, was performed between Columbia University and a host connecting with a cable modem to a public ISP, also in New York. The last group consists of high-speed "backbone" measurements, all within the US: *us3*, *usFeb3*, *usRog*, respectively. All these measurements were done between Columbia University and WWW or FTP servers connecting to the Internet via high-speed links.

LRD was present mostly in traces where the test TCP connection possibly passed one or more bottlenecks, (i.e., in the "trans-Atlantic", "cable modem" and "leased line" traces). In case of backbone measurements, LRD was not present in most of the traces, however, during peak hours we could measure $H > 0.5$ as well. The following question arises: if the trace does not show LRD, does this mean that TCP did not mix with LRD traffic at all? The answer is possibly not. In almost all the cases when LRD was not present, we could find that the advertised window and not the free capacity on the path limited the TCP rate. Another possible reason is that the access line limits the TCP rate, in which case the resulting traffic flow is smooth. The effect of paths with large bandwidth delay products and small windows on the propagation of LRD is further analyzed in Section 3.3.3.

3.1.2 Simple Analytic Model

In this section, we introduce a simple analytic model supporting the argument that TCP can adapt and inherit LRD from background traffic. Later in this chapter, we analyze the limitations of this simple model and we discuss how to refine it.

All relevant components of the simplified network model are depicted in Figure 3.2. A single greedy TCP connection sends data between host A and host B. The path of the connection consists of three parts: a network cloud before and after router R and a bottleneck buffer in router R, where the connection has to share the service capacity and the buffer space with a self-similar background traffic flow. Self-similarity of the background traffic can be induced, for example, by large aggregations of infinite variance On/Off streams as suggested in [CrBe96]. In the analytic model it is assumed that TCP can adapt ideally to a background traffic stream

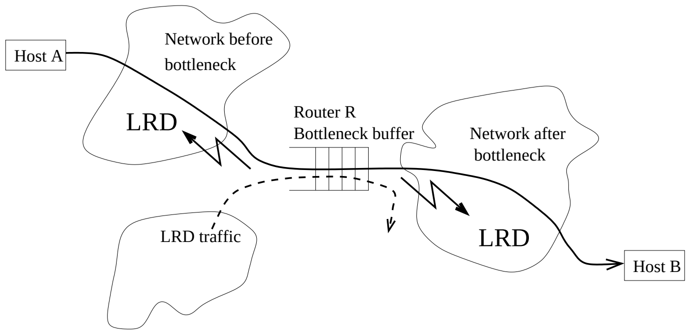


Figure 3.2: Network model

in a bottleneck buffer. Under "ideal adaptivity" we mean that the TCP connection is able to consume all remaining capacity unused by the background traffic stream. It is also assumed that the TCP connection does not have any effect on the background traffic. The generality of this assumption covers several practical cases, for example, if the background flow is a large aggregate consisting of a large number of connections. The limits of these assumptions are analyzed later in the chapter.

Denote the background traffic rate by $B(t)$, $0 \leq B(t) \leq C$, where C is the service rate of the bottleneck buffer in bit per seconds. If TCP congestion control is "ideal" and its effect on the background traffic is neglected, then the TCP connection will utilize all unused service in the bottleneck. The rate of the "ideal" TCP flow is denoted by $A(t)$:

$$A(t) = C - B(t).$$

The resulting process is simply a shifted and inverted version of $B(t)$, which implies that the correlation structure of processes $A(t)$ and $B(t)$ are the same. In other words, TCP "inherits" the statistical properties of the background process. In particular, let us model the background traffic rate as Fractional Gaussian Noise (FGN):

$$B(t) = m + \sqrt{a} N_H(t) \tag{3.1}$$

where m is the average rate in bit per seconds [bps], a is the variance, and $N_H(t)$ is a normalized FGN process with Hurst parameter H. Note that FGN is a discrete time process, so the rate at time t is approximated by the amount of bytes sent during sufficiently small constant duration time periods. Based on the arguments above, the adapting TCP will also be an FGN with the same statistical self-similarity exponent H. As TCP congestion control works end-to-end, the same traffic rate can be measured along the path *before* and *after* router R as well. This implies that TCP propagates self-similarity or LRD to parts of the network where otherwise it would not be present.

The result above is based on a simple scenario using a few assumptions, such as ideal TCP adaptivity, single bottleneck, and assuming that the TCP flow does not modify the background traffic characteristics. However, if the implications of this simple scenario are valid in real TCP/IP networks, the consequences for traffic engineering are far reaching. Regarding this, we are going to address the following important questions:

1. What are the limitations of TCP adaptation, i.e., how "ideal" is TCP congestion control when propagating self-similarity or other statistical properties?

2. A single long-living connection was used in the simple network model and in the measurements. Can self-similarity be propagated by short duration TCP connections?

3. We assumed that the background LRD traffic flow used is non-adaptive. Is self-similarity still propagated if the background traffic flow is an aggregate of adaptive flows?

4. We considered a single bottleneck on the TCP path. On the other hand, in some cases TCP connections may traverse multiple bottleneck routers and buffers multiplexing with multiple self-similar inputs. What are the characteristics of the end-to-end TCP flow in this case?

5. Is self-similarity propagated between adaptive connections, i.e., can self-similarity be inherited from one TCP to another one that has no direct contact with the source of self-similarity?

3.2 TCP as a Linear System

In the previous section it was assumed that TCP congestion control is "ideal", which, as a matter of course, cannot be the case in real networks. The consequence of self-similarity is that fluctuations are not limited to a certain timescale. When analyzing how "real" TCPs propagate self-similarity, the adaptation of TCP to fluctuations on several timescales should be investigated. In this section, it is shown that TCP in a bottleneck buffer can be modeled as a linear system, i.e., TCP takes over the correlation structure of the background traffic through a linear function.

TCP is an adaptive mechanism that tries to utilize all free resources on its path. Adaptation is performed as a complex control loop called the congestion control algorithm. Of course, full adaptation is not possible, as the network does not provide prompt and explicit information about the amount of free resources. TCP itself must test the path continuously by increasing its sending rate gradually until congestion is detected, signaled by a packet loss, and then it adjusts its internal state variables accordingly. Using this algorithm, TCP congestion control is able to roughly estimate the optimal load in a few round trip times. Since congestion control was introduced in the Internet [Jac88], it has proved its efficiency in keeping network-wide congestion under control in a wide range of traffic scenarios.

In this section, we analyze the adaptivity of TCP, and conclude that a simple network configuration, which consists of a single bottleneck buffer shared by a “generator” flow and a “response” TCP flow can be well modeled as a linear system above a characteristic timescale. The cut-off timescale depends on the path properties of the connection. The linear system transforms certain statistical properties, e.g., autocovariance, between the “generator” stream and the “response” traffic stream through a transform function, which is characteristic of the network configuration.

3.2.1 Measuring the Adaptivity of TCP on Several Timescales

In the first analysis a single, long, greedy TCP stream is mixed with random background traffic streams. See Figure 3.3 for the configuration. The background streams are constructed in a way, such that they fluctuate on a limited, narrow timescale.

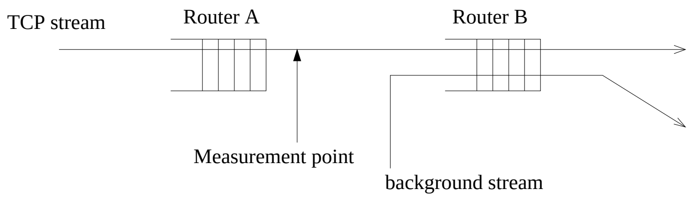


Figure 3.3: Simulation model for the test of TCP adaptivity to a self-similar background traffic stream. The two buffers are identical: service rates $C_1 = C_2 = 1$ Mbps, propagation delays $d_1 = d_2 = 5$ ms, buffer sizes $B_1 = B_2 = 40$ packets.

To limit the timescale under investigation, the background traffic approximates a constant amplitude sine wave of a given frequency f: $A_{background}(f,t) = a\sin(2\pi f t + \alpha) + m$ where α is a uniformly distributed random variable between $[0, 2\pi]$. The process $A_{background}(f,t)$ is a stationary ergodic stochastic process with correlation $R(\tau) = a^2/2 \cdot \cos(2\pi f \tau)$. The power spectrum of this process consists of a single frequency component at f. In the simulation the background process had to be approximated by a packet stream (packet size of 1000 bytes), with the result that the spectrum is not an impulse but a narrow spike, see Figure 3.4.

If TCP is able to adapt to the fluctuations of the background traffic flow, the same frequency f should appear as a significant spike in the power spectrum of the TCP traffic rate process as well. The ratio of the amplitudes of this frequency component in the spectra is a measure of the success of TCP adaptation on this timescale. Denote the *measure of adaptivity* at frequency f by $D(f)$

$$D(f) = S_{tcp}(f)/S_{background}(f) \tag{3.2}$$

where $S_{background}(f)$ is the spectral density of the background traffic rate process at frequency f and $S_{tcp}(f)$ is the spectral density of the adapting TCP rate process at the same frequency.

Figure 3.4a depicts an experiment with a background signal of $f = 0.01[1/s]$. The top part of the figure shows the spectrum of the background traffic approximating a sine wave of frequency f. The bottom part is the measured spectrum of the TCP response. The spectrum of the response has a significant spike at f, but it also contains a few smaller spikes at higher frequencies caused by the congestion control.

Conducting the experiment for a wide range of frequencies f, it is possible to plot the adaptivity curve of TCP. Figure 3.4b shows the result for several versions of TCP. Note that the shape of the function only slightly depends on the TCP version. It can be seen that TCP adapts well to frequencies below $f_0 \approx 0.15[1/s]$, but it cannot adapt efficiently to fluctuations on higher frequencies in this configuration.

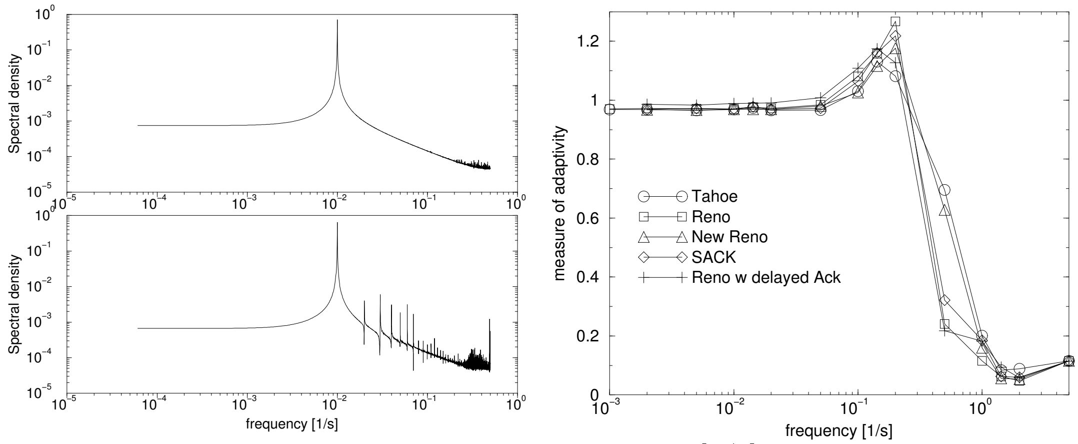


Figure 3.4: a) Frequency response to a sine wave of $f = 0.01[1/s]$ (top: background sine wave, bottom: TCP response). In this configuration the measure of adaptivity is $D(0.01) \approx 1$. b) Measure of adaptivity $D(f)$ as a function of the frequency for several TCP variants.

At f_0 a resonance effect can be observed, at this frequency TCP is more aggressive, and gains even higher throughput than what is left unused by the non-adaptive background flow. This frequency is equal to the dominant frequency of the TCP congestion window process when there is no background traffic present (idle frequency), see Figure 3.5. In [MSMO97] a macroscopic model for TCP connections is published. It is derived that if every p^{th} packet is lost for a TCP connection, then the congestion window process traverses a periodic sawtooth and the length of the period is $T = RTT * W/2$, where RTT is the round-trip time of the path in seconds and W is the maximum window size in packets. In our case we can approximate $RTT = B/C + d$, where B is the buffer size in packets, C is the service rate in packets per second, and d is the total round-trip propagation delay in seconds. The maximum window size is $W = B + Cd$, which is the maximum number of packets in the pipe (buffer and link). This gives an estimate of $T = 6.81$ s and $f_{cycle} = 1/T = 0.15$ [1/s]. The result agrees with the measured resonance frequency f_0, and confirms our argument that the resonance effect observed in the measure of adaptivity function $D(f)$ is due to the TCP window cycles (see Figure 3.4b).

The characteristic timescale of the TCP window cycles ranges in relatively wide ranges in real networks, and the relation of $T \approx RTT * W/2$ can be used for an approximation. For example, if the round-trip time, which in the previous simulation was approximately 0.33 s, is

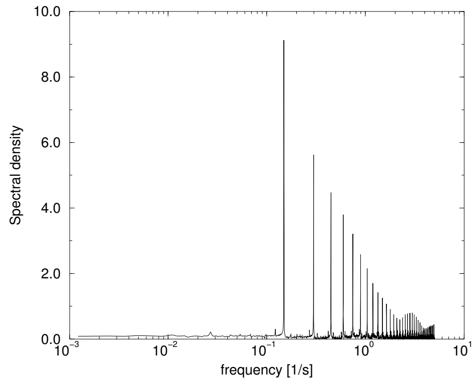


Figure 3.5: Spectrum of TCP congestion window process when no background traffic is present.

rather in the range of a few tens of milliseconds, the cut-off timescale drops below 1 s. Even below this timescale TCP adapts to fluctuations, though the effectiveness is limited, as shown by the transmission curve; f_0 approximately separates traffic dynamics to “local” and “global” scales, above f_0 it is the background process which shapes the spectrum, below f_0 the spectrum is a result of TCP control dynamics and external stochastic processes has less impact on it.

In the next section, we analyze the case when the background traffic stream is more complex and contains fluctuations on several timescales.

3.2.2 Tests for Linearity

In real networks background traffic is not limited to a single timescale. In the following, we analyze the case when several frequencies are present and test whether TCP is able to adapt to fluctuations on these timescales or not. The motivation is to prove that TCP can adapt to fluctuations on several timescales independently of each other, more precisely, we want to show that TCP control forms a linear system in this configuration.

By linear system we mean that if the background traffic rate is given by $B(t)$, and the adapting TCP traffic rate $A(t)$ is expressed using a function Ψ, then $A = C - \Psi(B)$, where Ψ is a linear function of B, i.e., $\Psi(a_1B_1 + a_2B_2) = a_1\Psi(B_1) + a_2\Psi(B_2)$. In case of ideal adaptivity, Ψ takes the simple form of $\Psi(x) = x$, and the TCP rate is obtained simply as $A(t) = C - B(t)$, see Section 3.1. If the background traffic is a superposition of streams $B_i(t), \quad i = 1 \dots N$

$$B(t) = \sum_{i=1}^{N} B_i(t)$$

then the rate of TCP is given by

$$X(t) = C - \Psi(B(t)) = C - \sum_{i=1}^{N} \Psi(B_i(t)).$$

This construction provides us with a simple test on linearity: we investigate the response to the superposition of several $B_i(t)$ streams and investigate the spectrum of the response. Figure 3.6a shows the spectral density of the background and the TCP response when the background is a composition of 10 random phase sine waves equidistantly spaced in a logarithmic scale (the non-zero widths of the spikes are due to the fact that the background mix only approximates sine waves with varying packet spacing). It can be observed that TCP was able to adapt to all frequency components in the mix below $f = 1$.

To test whether TCP really adapts to fluctuations independently, a wide range of traffic mixes were simulated consisting of two frequencies f_1 and f_2. A large number of simulations were performed, covering a whole plane with the two frequencies, in the range of $[0.05, 500][1/s]$. Then, the adaptivity measure for *one* of the frequencies ($D(f_1)$) was calculated. If the system is linear, the measure of adaptivity function at frequency f_1 should be independent of the other frequency f_2. The results of the simulations support our conclusions, see Figure 3.6b.

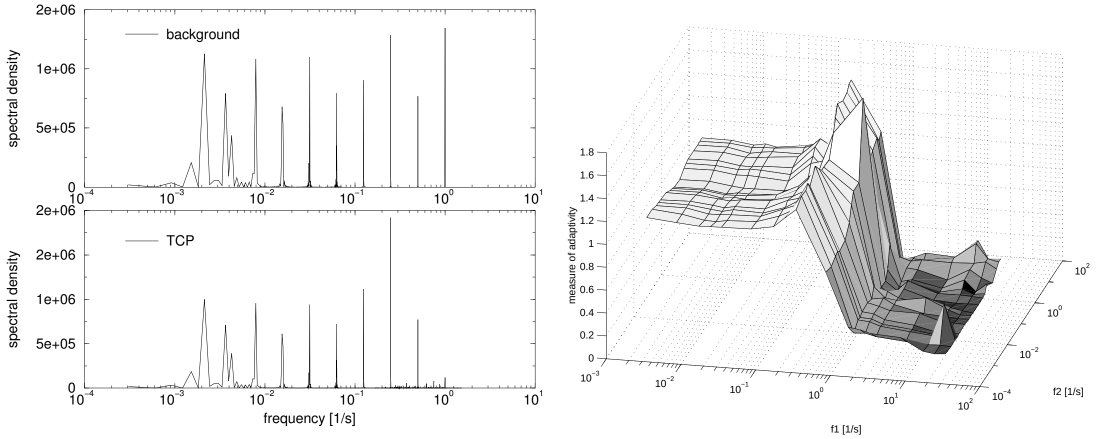


Figure 3.6: a) TCP frequency response to the superposition of 10 random phase sine waves. top) background traffic, bottom) TCP response. b) Measure of TCP adaptivity $D(f_1)$ when the background process is composed of two frequencies f_1 and f_2.

3.2.3 Response to White Noise

In the previous analysis the background processes were limited to superposition of sine wave processes. In real networks background traffic streams cannot be modeled by just a few frequency components, it is more appropriate to model background traffic streams as "noises".

Two types of special noises are most relevant in traffic modeling: the White Noise (WN) process and the Fractional Gaussian Noise (FGN) process. The White Noise process is the appropriate signal for analyzing the frequency response of a system and the Fractional Gaussian Noise process frequently appears as the limit process of traffic aggregations [TWS97].

If TCP is a linear system, then it should transform the correlation of any complex stochastic process (e.g., WN or FGN) through the same transform function. In this section, the response of TCP to a WN process is analyzed. WN is a special noise as it has constant spectral density. If TCP is linear, then it should respond with the characteristic curve obtained previously. The result is depicted in Figure 3.7. The similarity of the curve to our previous test-signal based test supports the linearity argument. In addition, the constant flat range, which starts at a characteristic timescale and spans several timescales upwards, provides us with information about the timescale limitation of TCP adaptivity. Note that this mechanism behaves like a low-pass filter.

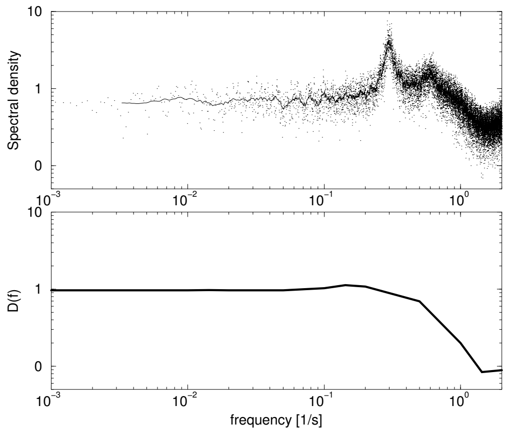


Figure 3.7: a) TCP's frequency response to white noise, spectral density (dots) and its smoothed version (line). b) Measure of adaptivity $D(f)$, see also Figure 3.4b

3.3 TCP Adaptation to Self-Similar Background Traffic

Once we have investigated the linearity of TCP and have shown that the transform function is flat below a characteristic frequency, it is quite obvious to expect that TCP, while adapting to signals of complex frequency content, reproduces the same spectral density as the original signal above a timescale, which depends on the path properties (round-trip time, size of the pipe, etc.). If, for example, TCP traverses a link where the traffic shows self-similarity, it will adapt to it with a spectral response equal to the spectrum of the self-similar traffic (asymptotically). As TCP's control algorithm works end-to-end, this property is "propagated" all along the TCP connection path.

In the experiment we simulated a single TCP sharing the bottleneck buffer with a synthetic FGN traffic flow of $H = 0.8$. Figure 3.8 shows the power spectrums of both the TCP and FGN traces at an aggregation level of 10ms. As suggested in the previous section, TCP shows the same spectrum as FGN asymptotically, there is a difference only around and below the cut-off timescale. Consequently, the TCP traffic flow is also asymptotically second-order self-similar with the same scaling parameter as the FGN process ($H = 0.8$).

3.3.1 Can Adaptive SRD Traffic Propagate Self-Similarity?

So far we have analyzed cases when long greedy TCP sessions were mixed with background traffic. It has been shown that the distribution of file sizes in Web traffic is heavy-tailed [CTB96]. This increases the probability of the occurrence of such long TCP connections. Nevertheless, it

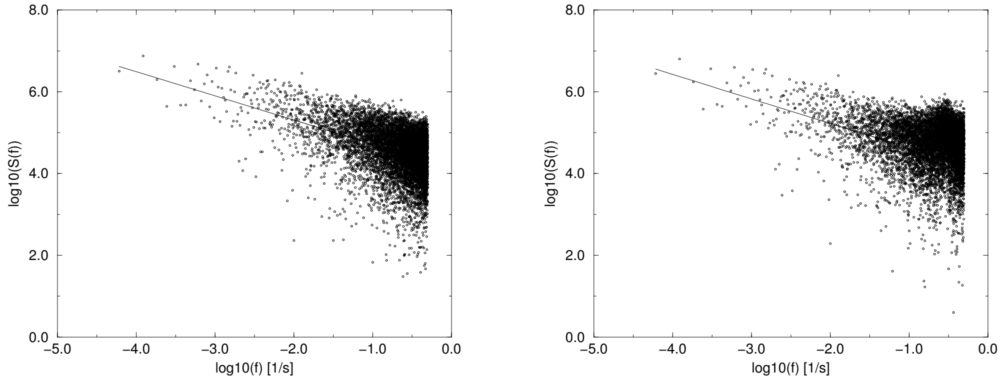


Figure 3.8: a) Power spectrum of background traffic $H = 0.8$. b) Power spectrum of TCP traffic adapting to the FGN, estimated $H = 0.8$.

is investigated whether short duration TCPs (durations with light-tailed distributions) have the same adaptivity property to LRD traffic or not. A positive answer increases the generality of our argument. Based on previous work [TWS97] we would expect that if On and Off durations are light-tailed, the aggregate traffic is short-range dependent (SRD). This section demonstrates that TCP streams have LRD properties in spite of the short-range dependent result suggested by the On/Off model.

During the simulation we established k parallel sessions. Within each session TCP connections were generated independently and the durations of TCP connections were exponentially distributed (with mean T_{On}) followed by exponentially distributed silent periods (T_{Off}). The simulation was started from the equilibrium state of the process. (See Figure 3.9.) Let's denote

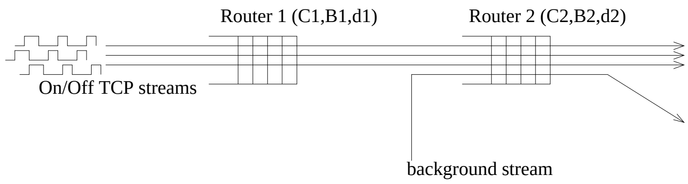


Figure 3.9: Simulation model of SRD driven TCP traffic multiplexed with self-similar background traffic (FGN with $H = 0.8$). $C_1 = C_2 = 1$ Mbps, $d_1 = d_2 = 5$ ms, $B_1 = B_2 = 40$ packets. $k = 10$ parallel sessions with exponentially distributed On and Off periods with means $T_{ON} = T_{OFF} = 10s$.

the number of active TCPs at time t by $N(t)$, $0 \leq N(t) \leq k$. With this construction $N(t)$ is a stationary Markov process and it is short-range dependent. See the self-similarity tests for $N(t)$ in Figure 3.10a ($H \approx 0.5$).

On the other hand, if these sessions are mixed with LRD background traffic, the aggregate TCP traffic, i.e., the amount of bytes transmitted by all TCPs, is LRD (Figure 3.10a). The reason is that the superposition of short duration TCPs can efficiently adapt to a background

LRD process just like one long duration TCP connection.

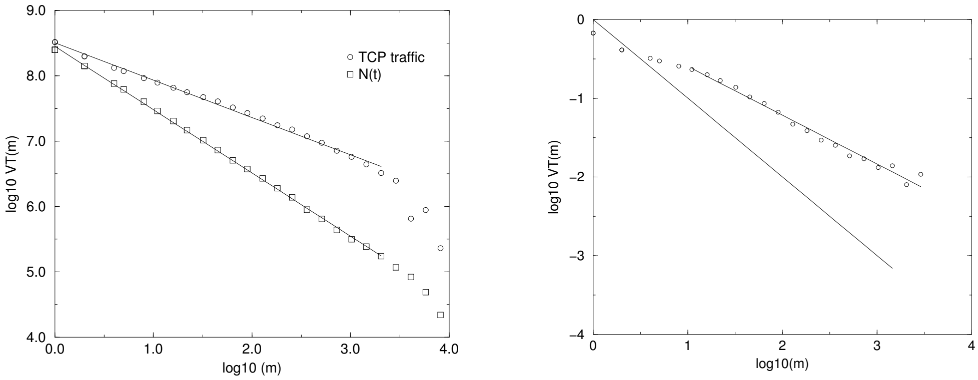


Figure 3.10: a) Variance-time test for the simulated On/Off process $N(t)$ ($H \approx 0.5$) and the simulated aggregate TCP traffic ($H \approx 0.72$). b) Variance-time plot of traffic generated by equal size short file transfers from *serv1.ericsson.co.hu* to *locke.comet.columbia.edu*, logging resolution 100 ms, $H \approx 0.7$.

A real network measurement also supports our argument. Short files (90 kbyte) were downloaded using the *wget* utility from *serv1.ericsson.co.hu* to *locke.comet.columbia.edu* (round-trip time $RTT \approx 180$ ms, average download rate $r \approx 160$ kbps, SACK TCP)[1]. Whenever the download ended, a new download was initiated for the same file. The experiment lasted for an hour, and the file was downloaded about 800 times. The traffic was captured with *tcpdump* at the client host. The Variance-Time plot shows that the traffic rate dynamics was self-similar, in spite of the short file-sizes, see Figure 3.10b. As a new download does not use any memory from a previous TCP connection, long-range correlations can be explained only by the long-memory dynamics of the network. In case of smaller files, TCP's capability to adapt to changing network conditions decreases. Although 90 kbyte is larger than the current average file size in the Internet, it has to be emphasized that a subset of connections is enough to propagate self-similarity. Furthermore, if HTTP 1.1 replaces HTTP 1.0, persistent TCP connections will be able to adapt better to traffic fluctuations, eventually improving the propagation effect; similarly, if a TCP implementation preserves some state from a previous connection, the propagation effect is improved.

3.3.2 Discussion on SRD TCP Streams

For simplicity, first assume that there is only one session with On/Off TCP connections multiplexed with LRD traffic. In this case $N(t)$ takes the values 0 or 1 for exponentially distributed durations. Assuming ideal adaptivity, when the session is active (a TCP is active) it can grab all capacity left unused by the background LRD traffic. Then the traffic rate during the *active* periods of the On/Off session can be expressed by $A(t) = F(t)$ where $F(t)$ is the free capacity (bit rate) left by the self-similar background traffic, $F(t)$ is an FGN process, see Section 3.1.

[1]Note that the access speed at the *serv1.ericsson.co.hu* side was increased to 256 kbps during this measurement.

During *inactive* periods $A(t) = 0$. Thus the traffic rate of the TCP controlled On/Off session for all t can be written in explicit form as

$$A(t) = N(t)F(t) \tag{3.3}$$

Assuming that the sessions are independent of the background process ($N(t)$ and $F(t)$ are independent), the autocovariance of $A(t)$: $\gamma_A(\tau) = \text{cov}(A(t), A(t+\tau))$ is

$$\gamma_A(\tau) = E\left[\left(N(t)F(t) - m_N m_F\right)\left(N(t+\tau)F(t+\tau) - m_N m_F\right)\right] \tag{3.4}$$

where $m_N = E[N(t)]$ and $m_F = E[F(t)]$. Factorizing:

$$\gamma_A(\tau) = E[N(t)N(t+\tau)]E[F(t)F(t+\tau)] - m_N^2 m_F^2 \tag{3.5}$$

The left hand side of the product is

$$E[N(t)N(t+\tau)] = E[(N(t) - m_N + m_N)(N(t+\tau) - m_N + m_N)] = \gamma_N(\tau) + m_N^2 \tag{3.6}$$

The same holds for $F(t)$, and so the covariance can be written as

$$\gamma_A(\tau) = (\gamma_N(\tau) + m_N^2)(\gamma_F(\tau) + m_F^2) - m_N^2 m_F^2 \tag{3.7}$$

Finally,

$$\gamma_A(\tau) = \gamma_N(\tau)\gamma_F(\tau) + m_F^2\gamma_N(\tau) + m_N^2\gamma_F(\tau) \tag{3.8}$$

If $F(t)$ is LRD, its autocovariance decays asymptotically as $\gamma_F(\tau) \sim \tau^{-\beta_F}$ as $\tau \to \infty$, where $0 \leq \beta_F < 1$. On the other hand, if $N(t)$ is SRD, its autocovariance decays asymptotically faster than $\tau^{-\beta_N}$ where $\beta_N \geq 1$.

Consequently, the covariance of $A(t)$ decays asymptotically at the lower rate, in this case at the rate of the background LRD process since $\beta_F < \beta_N$:

$$\gamma_A(\tau) \sim \tau^{-\beta_F} \quad \text{as} \quad \tau \to \infty \tag{3.9}$$

If the On/Off process is LRD as well, e.g., the On and/or Off times are heavy-tailed, then asymptotically the larger Hurst exponent is measured on the path. In practice, the border of the scaling region depends on the actual shape of the covariances and the means m_A and m_F.

If there are more than one On/Off streams sharing the bottleneck buffer with a self-similar background traffic stream, $N(t)$ takes higher values than 1 as well. However, for the adaptivity of the aggregate it is sufficient to have at least one active connection as it was shown in Section 3.3.1. The aggregate traffic of multiple On/Off streams adapting to a background stream may be approximated by

$$A_{\text{aggr}}(t) = \Theta[N(t)]F(t) \tag{3.10}$$

where $\Theta(.)$ is the Heaviside-function, ($\Theta(x) = 1$ if $x > 0$ and 0 otherwise). $\Theta[N(t)]$ itself is also an On/Off process. If the On/Off processes are independent and they are exponentially distributed, then $N(t)$ forms a Markov process ($\Theta[N(t)]$ is the indicator process for the empty state of this Markov chain) and it is SRD.

The conclusion of this section is that if the end-to-end service uses TCP connections, then the traffic generated by the service is also adaptive, and in this case the adaptivity of the end-to-end service is sufficient to "propagate" LRD to other parts of the network. Moreover, if $N(t)$ is LRD, then the larger Hurst exponent $\max(H_N, H_F)$ is propagated.

3.3.3 TCP Connections Limited by the Receiver Window

In Section 3.1.1 it was found that most connections between hosts directly connected to the US backbone do not show LRD in the tests. All of these measurements have in common that that the bandwidth delay product on the connection's path is large, and the TCPs traffic rate is limited not by network congestion, but the advertised window of the receiver. Even in this case, it is most likely that the connection shares buffers with LRD background traffic, but since TCP is not able to utilize the full bandwidth, the adaptation effect between background traffic flow and the TCP connection is weak.

We performed a series of simulations that reproduce the above backbone scenario. We modified the previous simulation model: the link speed was increased to 4 Mbps and the round-trip link propagation delay to 100 ms. The buffer size remained 40 packets as in the previous simulations. The background traffic flow was synthetic FGN with $H = 0.8$, average 0.5 Mbps and variance also 0.5 Mbps. With these settings the average link utilization was just 12.5 %, similarly to an overprovisioned backbone link.

We performed several simulations with a single TCP passing a buffer with FGN background load. In consecutive simulations the TCP receiver had different window sizes between 10 to 45 packets. Each simulation lasted for 7200 s, long enough to perform LRD tests. We expect that when the window is small, TCP is limited by the advertised window and TCP is not able to utilize the maximum available throughput. In this case the impact of background traffic on TCP rate is weak. As the window increases, the TCP rate also increases, finally, after some point it flattens out as the network load becomes the limit for the TCP throughput, see Figure 3.11a. From the end-user perspective, this is the desired operating point of the system.

After each simulation run, we performed the LRD tests. Figure 3.11b shows the results of the Variance-Time tests. It can be seen that in all cases, asymptotically all traces have the same slope as the VT plot of the background traffic. The difference is in the timescale where LRD scaling starts, as the window increases, scaling starts at smaller and smaller timescales.

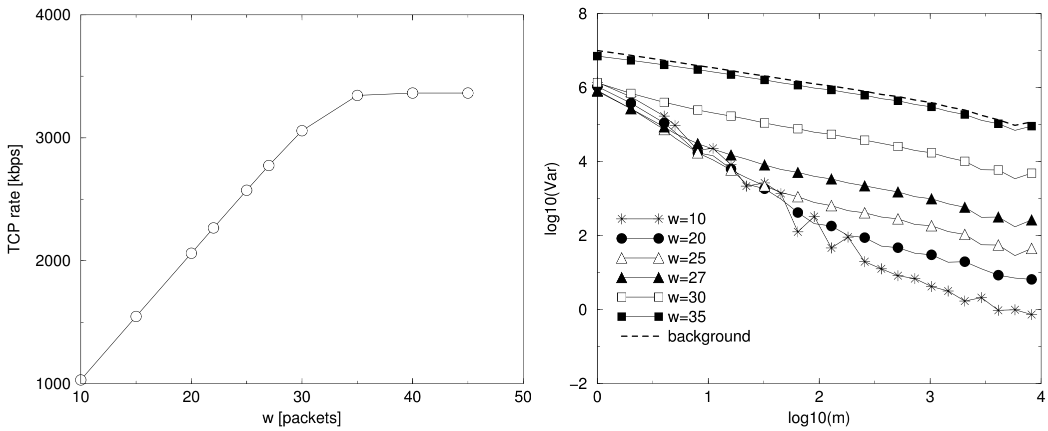


Figure 3.11: a) TCP throughput vs. receiver window size in packets. b) Variance-time plot of the number of bytes recorded during m for several windows sizes. The topmost graph is the VT plot of the FGN trace.

Today, there are still many operating systems and applications that do not support large windows. In case of high-speed access speeds, the price is sub-optimal TCP performance since the window limits the maximum throughput. As high speed access technologies (e.g., DSL or cable modem) become more widespread, it is likely that operating systems and applications will demand larger windows, moving the cut-off timescale of propagation closer to the estimated analytic value derived in Section 3.2.1.

In this sense, propagation is a desired effect. Although larger windows cause increased variances in all timescales, see Figure 3.3.3b, the end-user perceived end-to-end throughput also increases significantly.

3.4 Spreading of Self-Similarity in the Network

Previously we analyzed the case when a TCP connection shares a single bottleneck buffer with LRD background traffic, and it was only this bottleneck that affected the rate of TCP. In this section the network case is discussed.

Two aspects are analyzed. The first one deals with the case when the path of an adaptive connection passes through several buffers with self-similar inputs. These buffers are candidates to become bottlenecks occasionally during the lifetime of the connection. The second one investigates whether self-similarity can spread from one adaptive connection to the other causing widespread self-similarity in a network area.

The presented results are intended to highlight the basic mechanisms, so the investigated scenarios are simplified for the ease of discussion.

3.4.1 Discussion of the Multiple Link Case

A wide area TCP connection usually spans 10-15 routers along its path, out of which there are usually several backbone routers with high level of aggregated traffic, see Figure 3.12. A TCP

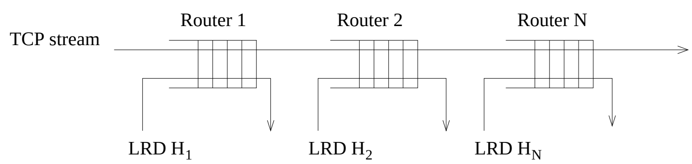


Figure 3.12: A TCP connection traversing multiple hops with independent background LRD (H_i) inputs.

connection has to adapt to the whole path. The capacity of the end-to-end path, at time t, depends on which buffer is the bottleneck at this time. Because of traffic fluctuations, the location of the bottleneck moves randomly from one router to the other.

Assuming ideal end-to-end adaptivity, the rate of the adaptive TCP connection is equal to the free capacity of the bottleneck link at time t:

$$A(t) = \min_{i \in N} F_i(t) \tag{3.11}$$

where N is the number of links and $F_i(t)$ denotes the free capacity of the i^{th} link on the path.

For simplicity, assume that the crossing background LRD streams on the links are independent and the link at time t is either empty: $F_i(t) = 1$, or full: $F_i(t) = 0$. With this simplification the rate of the adaptive connection can be written as

$$A(t) = \prod_{i=1}^{N} F_i(t) \tag{3.12}$$

In the previous section it was shown that the product of independent LRD processes is also LRD and it is asymptotically characterized by the largest exponent:

$$\gamma_A(\tau) = \tau^{-\min_i \beta_i} \quad \text{as} \quad \tau \to \infty \tag{3.13}$$

Thus, in the multiple link case it is the largest Hurst exponent among the background LRD streams on the links that characterizes the TCP connection.

For a numerical example using more complex processes, four FGN background samples were generated with equal mean rates, but with different Hurst exponents, to model $F_i(t)$, $i = 1 \ldots 4$, see Figure 3.13. The end-to-end process $A(t)$, which is the minimum of the FGN processes, is asymptotically second-order self-similar, and it has the same Hurst exponent as the largest Hurst exponent among the $F_i(t)$ processes, i.e., the result is the same as in the simple full/empty case.

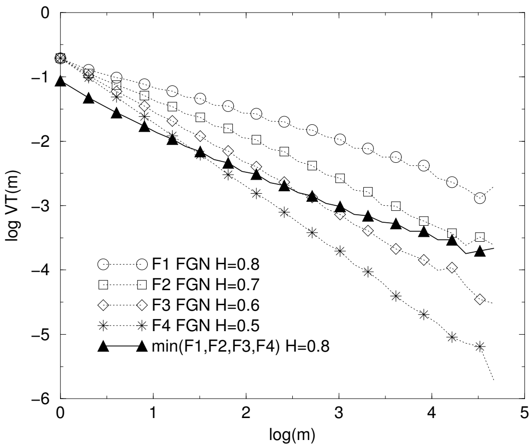


Figure 3.13: Variance-Time plots of F_i FGN processes of $H = 0.8, 0.7, 0.6, 0.5$ respectively (identical mean rates and variance), and the end-to-end process $A(t) = \min_{i \in N} F_i(t)$. The end-to-end path is characterized by $H \approx 0.8$ asymptotically.

Another possible interpretation of (3.11) is that we consider the $F_i(t)$ not as rate processes, but as indicator processes of congestion. From the end user perspective it is important to analyze whether the network is able to support the expected service level requirements, for example, whether the file transfer rate degrades below an acceptable level or not. Let $F_i(t)$ be the indicator process of link i indicating whether the link is congested and it cannot support the expected service rate for the connection ($F_i(t) = 0$), or is not congested ($F_i(t) = 1$). Thus, if the background congestion indicator processes are LRD, then it is the largest Hurst exponent that characterizes the end-to-end service characteristics of the investigated TCP connection.

3.4.2 Spreading of Self-Similarity among Adaptive Connections in Multiple Steps

So far, in all cases analyzed, adaptive traffic was in direct contact with self-similar background traffic. In this section, it is investigated whether self-similarity caused by adaptation can be passed on to adaptive traffic streams that have *no direct contact with the source of self-similarity*. A few simple conditions are given as well. Assuming that our argument is valid, self-similarity can spread out from a localized area, consequently, strong self-similarity is balanced throughout a wider area of the network.

A simple network scenario is used for the investigation. An adaptive traffic stream (*direct stream*) shares a link with self-similar FGN ($H = 0.8$) background traffic. The *direct stream* is mixed with another adaptive stream on a second link, which itself has no direct connection with the FGN traffic (indirect stream), see Figure 3.14. Two other streams thus affect the data rate of

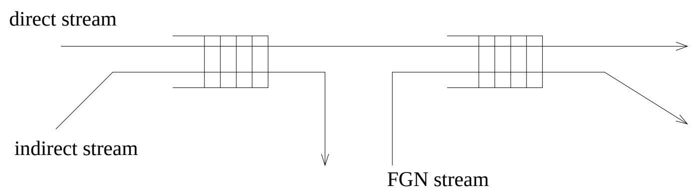


Figure 3.14: Network model for the investigation of self-similarity spreading.

the direct stream, and also the two adaptive streams have an effect on one another. We are going to investigate the statistical properties of both the direct and the indirect streams.

Assume ideal adaptivity and max-min fairness among the adaptive streams. Also assume that the service rates of both links are equal (C). If the background stream was inactive, the bottleneck would be the first buffer and the adaptive streams would share simply half the service rate, both sending at a rate of $C/2$.

$$A_{dir}(t) = A_{indir}(t) = C/2 \tag{3.14}$$

In the presence of the FGN stream flowing through the second buffer, the rates can still remain $C/2$, unless it is the second buffer which becomes the bottleneck, i.e., when the capacity left unused by the FGN stream is $C - A_{FGN}(t) < C/2$. In this case the direct stream can use at most $A_{dir} = C - A_{FGN}(t)$, so the indirect stream can grab all remaining service capacity in the first buffer $A_{indir} = C - A_{dir} = A_{FGN}$. In short:

$$A_{dir}(t) = \min(C/2, C - A_{FGN}(t)) \tag{3.15}$$
$$A_{indir}(t) = \max(C/2, A_{FGN}(t)) \tag{3.16}$$

Calculation of the autocovariance of A_{dir} and A_{indir} is difficult because of the *min* and *max* operators. We consider two simple, extreme cases. In the first case, the rate of the background LRD stream is always greater than $C/2$, simplifying the expressions to $A_{dir}(t) = C - A_{FGN}(t)$ and $A_{indir}(t) = A_{FGN}(t)$, i.e., *spreading of self-similarity is ideal*. In the second extreme case the rate of background process is always smaller than $C/2$, leading to $A_{dir}(t) = A_{indir}(t) = C/2$, i.e., *self-similarity disappears* from both adaptive streams. Simulations have verified these results as well.

The investigated scenario demonstrates the simplest mechanism of how adaptive connections may have effect on each other. We simulated a more complex scenario, where the synthetic *FGN* stream is replaced by an aggregate stream of randomly generated short TCP file transfers. The distribution of the file sizes is heavy-tailed. The direct and indirect TCP streams are also replaced by aggregates, but the file sizes within these aggregates are light-tailed.

The streams consist of $n_{heavy-tailed} = n_{dir} = n_{indir} = 100$ sessions. The file size distributions are Pareto distributions with the following parameters: the average file size is 40 Kbytes for all streams, the average waiting time between files is 20 sec. The shape parameters are $a_{heavy-tailed} = 1.1$ and $a_{dir} = a_{indir} = 3$ for both the file size and the waiting time distributions. With these parameters only one stream has heavy-tails ($a_{heavy-tailed} < 2$).

The results of the simulation experiment are shown in Figure 3.15. As suggested in [CrBe96]

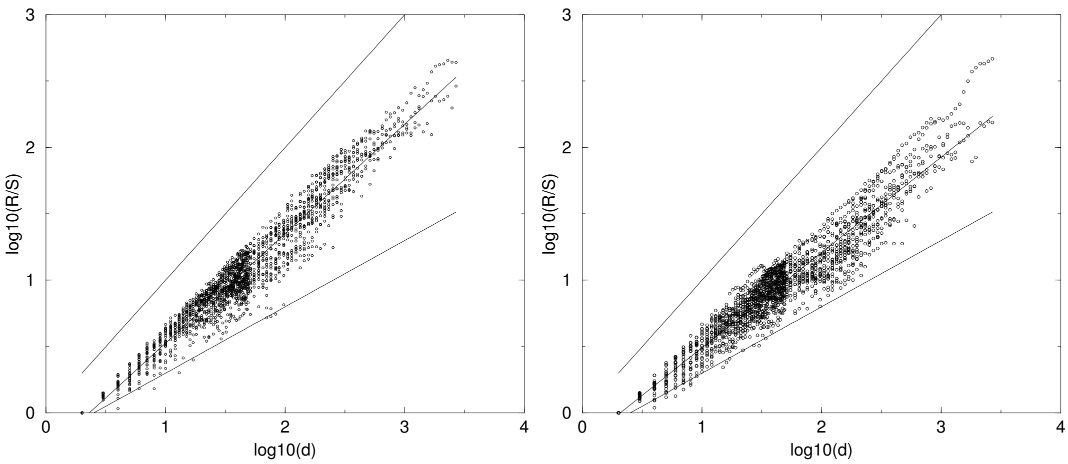


Figure 3.15: a) R/S plot of *heavy-tailed* stream $H = 0.82$. b) R/S plot of *indirect* stream $H = 0.71$

the traffic stream consisting of heavy-tailed file downloads is LRD ($H \approx 0.82$). Furthermore, the indirect traffic stream, although it was created using light-tailed distributions is LRD as well ($H \approx 0.71$). The cause is that long-range dependent fluctuations are propagated via the indirect stream.

Performing the previous experiment using different parameters, we have found that depending on the traffic mix, the spreading between indirect and direct streams can be strong but it can be weak as well. In certain cases, spreading to an indirect stream does not happen at all, just like in the simple analytic example assuming ideal TCP flows and max-min fairness. The exact requirements for spreading are subjects for further study.

3.5 Conclusions

It was demonstrated how a TCP connection, when mixed with self-similar traffic in a bottleneck buffer, takes on its statistical second-order self-similarity, propagating scaling phenomena to other parts of the network. It is suggested that the adaptation of TCP to a background traffic stream can be modeled by a linear system and the validity of our approach is analyzed. It was shown that TCP inherits self-similarity when it is mixed with self-similar background traffic in a bottleneck buffer through the transform function of the linear system. This property was demonstrated for both short and long duration TCP connections.

We also investigated TCP behavior in a networking environment. It was found that if congestion periods are long-range dependent in several hops on a connection's path, the largest Hurst exponent characterizes the end-to-end connection. It was also demonstrated that TCP flows, in certain scenarios, can pass on self-similarity to each other in multiple hops.

As thousands of parallel TCP connections continuously intertwine the Internet, the mechanisms described in this and the previous chapter can provide us with deeper insights why significant and strong self-similarity is a general and widespread phenomenon in current data networks.

The next chapters, Chapter 4 and 5, discuss the problems of provisioning Differentiated Services in the Internet. We discuss the special requirements and limitations of the Differentiated Services architecture [BBCD98], and analyze the implications of our results in Chapter 2 and 3 to the performance modeling of wired and wireless data networks.

Chapter 4

Resource Management for Differentiated Services Networks

There is an increasing demand for Quality of Service (QoS) in the Internet to enable a wide variety of new services. There have been a number of attempts to introduce QoS into the Internet, but most of them have suffered from scalability limitations inhibiting global deployment, and lack of robustness to maintain performance guarantees. Differentiated Services (DiffServ) [BBCD98] address the limitations of Integrated Services [BCS94, Shen95] by resolving the scalability problems. However, DiffServ leaves certain elements of the desired end-to-end QoS architecture unspecified.

In essence DiffServ specifies local packet handling rules, called Per Hop Behaviors (PHBs). The idea is that end-to-end services can be built up using these PHBs. For these services to work, appropriate resource management is necessary. It is essential to ensure, for example, that resources assigned to a class are sufficient to serve the traffic entering the class at the desired QoS.

Resource management can be approached from two directions. First, dimensioning of class and link resources is a preventive approach where the task is to select the correct link speed, and equally importantly, to partition the link resources to several DiffServ classes. For this process longer term traffic measurements and subscriber Service Level Agreement parameters can be used. The second approach is more dynamic, and operates by controlling the amount of traffic in the classes. This can be accomplished by signaling [Bra97] and admission control, using bandwidth brokers or by subscription control, depending on the network operator.

In this chapter, we introduce a set of measurement based resource estimation methods for DiffServ networks based on the effective bandwidth concept first discussed in [GAN91, Kel96, GiKe97]. The methods are suitable for both traffic engineering and admission control purposes and are suitable to be implemented in routers, bandwidth brokers or in off-line traffic engineering tools. The novelty of our work is that the resource estimation methods presented in this chapter are tailored to meet the characteristics of the DiffServ environment:

- traffic descriptors are coarse;
- only aggregate measurements are feasible;

- scheduling redistributes service among classes;
- guarantees are “soft”; and
- traffic flows have diverse, complex statistical properties.

Hosts or edge devices that are connected to a DiffServ network can only provide minimal and coarse information about their traffic. Coarse information means that the statistics do not describe the traffic precisely, for example, in case of a token bucket policer the policer parameters would be set very conservatively. The reason for this is because of the extreme diversity of Internet applications. It is not possible to develop accurate, yet general and flexible traffic descriptors suitable to all applications. We assume that in the future only very simple traffic descriptors will be used, (e.g., token bucket descriptors, access speed), which are easy to use and easily understood by the users. However, such simple descriptors are inherently inaccurate. We argue that the estimation methods have to be aware of this property, and they should use measurements of actual traffic load together with the traffic descriptors to improve the tightness of estimations [SaSh91, CKT96, JDSZ97, CLTR97].

In practice, because of scalability reasons, only aggregate measurements are feasible. This precludes most measurement based admission control algorithms suited for ATM networks, where per flow measurements are usually assumed. In our proposed methods, only scalable, aggregate measurements of large numbers of flows are performed. These flow aggregates are either entire queues, or large flow sets within a queue, classified based on packet header information, e.g., the DiffServ field (TOS field) or protocol information (TCP, UDP).

Since traffic descriptors are expected to be coarse, due to highly varying rates and uncertainty of demand, traffic flows usually do not utilize the guaranteed resources to the full extent. For better overall network performance resources are not rigidly allocated to different queues in DiffServ architectures. Unused bandwidth by a class is dynamically available to other classes on a packet-by-packet basis. For example, in case of Static Priority Queuing (SPQ), when there are no packets waiting in a queue, the scheduler will serve a packet from a lower priority queue. This dynamic allocation of resources is a crucial property for the economy of network services in the future Internet. Our analytic models take this property into account by estimating the QoS in every queue of the SPQ DiffServ architecture.

Many Internet applications can operate in the absence of QoS guarantees. It is expected that future better than best-effort applications will tolerate limited variations of the QoS, thus looser QoS guarantees may be sufficient. This makes probabilistic approaches more desirable, as these result in better resource efficiency than worst-case deterministic solutions [PaGa93, WKLZ96, LeB98]. On the other hand, previous work using probabilistic bounds usually assumed asymptotic models, which means that the derived probability bounds are asymptotically accurate for very small loss probabilities, for example, 10^{-12}. In reality, most real-time applications do not require such strong guarantees, for example, voice service can efficiently operate even if the loss ratio is as high as $1-5\%$ (depends on coding) [JaCh81]. Easing stringent QoS requirements enables the exploitation of more statistical multiplexing gain. In this chapter, we develop bounds that are precise not only asymptotically, but also in the QoS ranges expected in DiffServ networks.

The statistics of TCP/IP applications are diverse and the correlation structure of Internet traffic is complex, this impacts the assumptions we have to make in analytic models. The resource estimation methods should be robust with respect to these properties to maintain the flexibility and robustness of the Internet. Several methods assume that traffic flows follow certain models, e.g., Markov Modulated Processes [KWC93, KoMi98, Kel96], which assumption is too restrictive knowing the diversity of applications. Other methods are more general, but assume that traffic flows suit certain classes of statistical processes, for example, methods based on the assumption that traffic rate fluctuations are short-range dependent [CLTR97, KWC93]. These methods also have limitations due to the wide-scale presence of long-range dependence in the Internet, see for example, Chapters 2 and 3. We developed methods that do not assume that traffic flows fit any specific distribution or correlation structure, yet they are not heuristic but analytically precise.

The chapter is organized as follows. An overview of the DiffServ management framework can be found in Section 4.1. Specific effective bandwidth formulae are developed for Expedited Forwarding and Assured Forwarding classes to provide bandwidth and delay guarantees in Sections 4.2 and 4.3, respectively. We propose several solutions that utilize different measurements and require different implementation complexity. In Section 4.4 we show how to perform resource management of a practical DiffServ router implementation based on Static Priority scheduling. Finally, in Section 4.5 we present some concluding remarks.

4.1 DiffServ Traffic Management Framework

The Differentiated Services standardization effort of the IETF does not specify how traffic management is performed. The guarantees supported by DiffServ PHBs are relative rather than absolute, and do not eliminate the need of appropriate traffic management tools that estimate, anticipate and help maintain acceptable levels of network performance. To perform these tasks efficiently, detailed information is needed about the traffic demands in the network. The most important information is listed below:

- Traffic specifications of end-to-end or edge-to-edge flows, containing access speeds, parameters of traffic policers.
- QoS guarantees required by flows, (e.g., description of DiffServ classes AF, EF, or BE).
- Routes of the flows across the administrative domain.
- Aggregate load measurements of DiffServ classes on a link-by-link basis.

There have been considerable amount of work to obtain this data in TCP/IP networks. With some limitations, most of this data are already used in the Network Operation Centers of major network operators for network management [FGLR00, FGLR00-2]. However, not all the required information is available in real-time, since it would require specific dynamic resource reservation protocols, such as RSVP [Bra97], which have not been widely deployed. Although there has been proposals to introduce dynamic DiffServ specific protocols that would solve this problem, it is still an open issue whether the IETF will initiate standardization in this area, or proprietary protocols will emerge instead.

Today, traffic management has to rely on network "snapshots", which are updated on coarser timescales (e.g., hours). The time delay makes it impossible to perform real-time traffic regulation (e.g., admission control), since measured data would be outdated. We argue, however, that longer timescale resource management can still be efficiently performed using the currently available data mining techniques. Longer timescale resource management includes traffic engineering, which is done on an hourly basis, and network design, which is a longer timescale task, and usually involves upgrades of links and routers.

4.2 Effective Bandwidth Bounds for Assured Classes

Assured classes are intended to provide guaranteed throughput. Throughput guarantees are achieved by keeping the total input rate of the queue serving the assured class below the service rate of the queue. In practice, throughput is not interpreted as an instantaneous value but it is defined as the average amount of bits per second transmitted over a given time period. This softer definition cannot capture the packet arrival dynamics below a certain timescale. This implies that even when the input rate is below the queue service rate, packets may accumulate temporarily in the buffer. However, long lasting congestion cannot form. Therefore, when discussing assured classes, we ignore buffer fluctuations and consider only the rate process.

In this section, we assume that all packets belonging to an assured class share a common queue with fixed service rate C. The flows entering this queue are individually policed/shaped to a maximum rate at the ingress points. The total load of the queue is measured periodically. Flows are assumed to be independent as shaping/policing is done at the edges and packet loss is small. Flows are also assumed to be weakly stationary, (i.e. their mean and variance do not change in time). In the following, we derive three simple effective bandwidth bounds based on the above assumptions.

4.2.1 Effective Bandwidth Basics

The theoretical basis of our work is the concept of *effective bandwidth* developed by Kelly [Kel96]. In the following, we briefly present previous results and notations (see also [C1]).

Definition 1 *The* ***effective bandwidth*** *of a traffic flow aggregate is BW if the rate process of the aggregate traffic exceeds this value with probability less than ϵ. Denote the rate of the individual flows by random variables $X_k,\ k = 1 \ldots N$:*

$$\Pr\left(\sum_{k=1}^{N} X_k \geq BW\right) \leq \epsilon. \tag{4.1}$$

The objective of resource management is to maintain the above probability in the network by regulating the amount of admitted traffic, by allocating enough resources to accommodate the demand, or both. In either cases, it is important to precisely estimate the effective bandwidth BW.

First, we introduce several important inequalities in relation to the effective bandwidth. For a positive valued random variable ξ and a constant a, the Markov inequality says that $\Pr(\xi > a) \leq \mathbb{E}[\xi/a]$. If we apply this inequality to (4.1), we get an upper bound for the saturation probability:

$$\Pr\left(\sum_{k=1}^{N} X_k \geq BW\right) \leq \mathbb{E}\left[\frac{\sum_{k=1}^{N} X_k}{BW}\right] \tag{4.2}$$

or, for any $s > 0$, as the exponential function is monotonous:

$$\Pr\left(\sum_{k=1}^{N} X_k \geq BW\right) = \Pr\left(e^{s\sum_{k=1}^{N} X_k} \geq e^{sBW}\right) \leq \mathbb{E}\left[\frac{e^{s\sum_{k=1}^{N} X_k}}{e^{sBW}}\right] \tag{4.3}$$

If X_k are independent, the expectation of the product is equal to the product of expectations:

$$\mathbb{E}\left[\frac{e^{s\sum_{k=1}^{N} X_k}}{e^{sBW}}\right] = \frac{\prod_{k=1}^{N} \mathbb{E}\left[e^{sX_k}\right]}{e^{sBW}} \tag{4.4}$$

Have the logarithm of (4.4), and put in into (4.2):

$$\ln \Pr\left(\sum_{k=1}^{N} X_k \geq BW\right) \leq \sum_{k=1}^{N} \ln \mathbb{E}\left[e^{sX_k}\right] - sBW. \tag{4.5}$$

where $\mathbb{E}\left[e^{sX_k}\right]$ is called the moment generating function of X_k. This inequality holds for any value of $s > 0$, (so for the minimum of the right side as well). This bound is a form of the Chernoff bound.

The right side of (4.5) can be used as the desired value for ϵ in (4.1):

$$\ln \epsilon = \sum_{k=1}^{N} \ln \mathbb{E}\left[e^{sX_k}\right] - sBW \tag{4.6}$$

From the above, we can introduce a parametric form of the effective bandwidth $BW(s)$, first published by Kelly et al.:

Theorem 1 ([Kel96]) *Denote the rates of flows as X_k, $k = 1 \ldots N$. If X_k are independent, and their moment generating function $\mathbb{E}\left[e^{sX_k}\right]$ exists, the parametric effective bandwidth of the flow aggregate is*

$$BW(s) = \frac{1}{s} \sum_{k=1}^{N} \ln \mathbb{E}\left[e^{sX_k}\right] + \frac{\gamma}{s}. \tag{4.7}$$

where $\gamma = -\ln(\epsilon)$ and $s > 0$.

If we have a link of capacity C and we want to find out whether the traffic mix “fits” into the link, we have to compare C with $BW(s)$. If we find any value of $s > 0$ where $BW(s) \leq C$, the bound will hold; i.e., the following test has to be evaluated:

$$\min_{s>0} BW(s) \leq C \tag{4.8}$$

Unfortunately, we need full knowledge of the distributions of all flows to calculate $\mathbb{E}\left[e^{sX_k}\right]$ and so $BW(s)$, which is not feasible. However, if every X_k are bounded by a peak rate h_k (which, for example, can be the access speed) and we know (estimate, or measure) the average rate m_k of the flows, $BW(s)$ can be bounded as well. First we apply that $e^{sx} \leq 1 + x\left(e^{sh} - 1\right)/h$ for all $s > 0$ and $0 \leq x \leq h$:

$$\mathbb{E}\left[e^{sX_k}\right] \leq 1 + \frac{e^{sh} - 1}{h}\mathbb{E}\left[X_k\right] \tag{4.9}$$

If we combine (4.7) and (4.9), we can bound $BW(s) \leq BW^O(s)$. This leads to the following theorem:

Theorem 2 ([Kel96]) *If flows X_k, $k = 1 \ldots k$ are bounded by peak rates $X_k \leq h_k$ and we know the mean rates $m_k = \mathbb{E}\left[X_k\right]$, the effective bandwidth of the aggregate can be estimated by $BW^O(s)$:*

$$BW^O(s) = \frac{1}{s}\ln\prod_{k=1}^{N}\left(1 + \frac{e^{sh_k} - 1}{h_k}m_k\right) + \frac{\gamma}{s}. \tag{4.10}$$

where $\gamma = -\ln(\epsilon)$ and $s > 0$.

This effective bandwidth formula contains only the mean and peak rates of the flows. It can be shown that $BW(s) = BW^O(s)$ if sources are on/off type (i.e. X_k can only be 0 or h_k). In this respect on/off sources have the worst-case effective bandwidth.

4.2.2 A Tight Bound Based on the Aggregate Load Measurement of a DiffServ Queue

The effective bandwidth expression BW^O cannot be used directly in DiffServ as it would require the knowledge of the average rate of every flow individually (m_k). Per-flow measurements would require per-flow packet classification in the routers, which is not scalable because of the large number of flows. In a DiffServ router, packets are only classified according to their PHBs, which usually means all flows entering the same queue.

If we can measure the load of a whole queue only, we need to modify (4.10) to contain only the aggregate mean. For this, we give an upper, conservative bound on the product part of 4.10. First, we decompose it to two products:

$$\prod_{k=1}^{N}\left(1 + \frac{e^{sh_k} - 1}{h_k}m_k\right) = \prod_{k=1}^{N}\left(\frac{e^{sh_k} - 1}{h_k}\right)\prod_{k=1}^{N}\left(m_k + \frac{h_k}{e^{sh_k} - 1}\right) \tag{4.11}$$

As the geometric mean is smaller than the algebraic:

$$\sqrt[N]{\prod_{k=1}^{N}\left(m_k + \frac{h_k}{e^{sh_k} - 1}\right)} \leq \frac{1}{N}\sum_{k=1}^{N}\left(m_k + \frac{h_k}{e^{sh_k} - 1}\right) \tag{4.12}$$

If we insert 4.12 into 4.10, we get a practical upper bound $BW(s) \leq BW^A(s)$:

Theorem 3 *If we know the peak rates h_k of flows $k = 1 \dots N$, and the mean rate of the aggregate M, the effective bandwidth can be expressed as*

$$BW^A(s) = \frac{N}{s} \ln \left(\frac{M + \sum_{k=1}^{N} \frac{h_k}{e^{sh_k}-1}}{N} \right) - \frac{1}{s} \sum_{k=1}^{N} \ln \left(\frac{h_k}{e^{sh_k} - 1} \right) + \frac{\gamma}{s} \tag{4.13}$$

Note, that with this approximation the bound is still holding, but we have a sum of rates instead of a product of m_k. The aggregate average rate M can be easily measured in a router interface. The only per-flow information needed is the peak rates of flows, which are usually available from the network management databases.

Before we can use the derived effective bandwidth $BW^A(s)$ in practice, we have to choose an appropriate value for s. The definition of effective bandwidth tells us that the saturation probability ϵ is not exceeded if there is an $s > 0$ for which $BW^A(s) < C$. That is, it is sufficient to find a single value for s, for which the bound holds to ensure the QoS guarantee. However, performing numeric optimization every time we recalculate $BW^A(s)$ may be computationally too expensive. Therefore, we derive a closed form solution that approximates the minimum value $BW^A = \min_s BW^A(s)$, (see the derivation in Appendix A.1):

$$s_{opt} = \sqrt{\frac{8\gamma}{\widehat{H} - (2M - H)^2/N}} \qquad \text{where } H = \sum_{k=1}^{N} h_k \text{ and } \widehat{H} = \sum_{k=1}^{N} h_k^2 \tag{4.14}$$

The resulting s_{opt} is inserted into $BW^A(s)$.

The derived bound has an interesting relation to previously published work. In [Flo96] the Hoeffding bound is suggested for admission control purposes. The Hoeffding bound, similarly to ours, also uses the peak rates and the aggregate mean:

$$BW^H = M + \sqrt{\gamma \widehat{H}/2} \tag{4.15}$$

It can be shown that if we approximate (4.10) with a unit slope tangent in m_k, we get the Hoeffding bound, this connection was derived in [GiKe97]. On the other hand, it can be shown that the *optimal slope tangent*, which leads to the tightest bound, leads to BW^A (proof can be found in [C1]). Consequently, our bound, BW^A, is always tighter than the Hoeffding bound (see Figure 4.3). The gain of using our improved bound over the Hoeffding bound is significant when the traffic is biased towards either high or low mean-to-peak ratios, which are the typical cases in networking applications.

4.2.3 Improving the Bounds by Measuring the Aggregate Rate Variance

In Section 4.2.2, the effective bandwidth estimation is calculated using the measured aggregate mean and the admitted peak rates. Theoretically, the more information we have about the distribution of flow rates (X_k), the tighter bound can be given. In this section, we give a closed form effective bandwidth formula with an extra parameter, which is the measured variance of the aggregate traffic rate.

The choice for variance is not ad hoc:

1. The variance, similarly to the mean rate, sums up, so an expression with the sum of the variances of the flows can be replaced with the variance of the rate of the aggregate traffic. The aggregate variance can be easily measured in the routers.

2. The previous bounds containing only the mean and the peak rates had to assume worst case variance. It can be shown that on/off sources have the worst possible variance for a given peak and mean rate. However, a large part of real-life traffic will not be on/off, so a measurement on the variance is expected to lead to a tighter bound.

In the literature, several bounds have been proposed that use the variance. In [BrSi99], a bound is derived for the variance from token bucket parameters. We argue that the token bucket parameters are too loose, and measurement of the variance can lead to significantly tighter bounds. The Central Limit Theorem (CLT) is used to estimate the effective bandwidth, in [GAN91], using the formula

$$BW^N = M + \sqrt{(2\gamma - \ln 2\pi)S} \tag{4.16}$$

where $S = \sum_{k=1}^{N} \sigma_k^2$ and σ_k^2 is the variance of flow k. The weakness of this formula is that it assumes large number of similar flows and the approximation is only valid around the mean value of the distribution. The precision of the bound for more realistic cases has not been evaluated.

The variance based effective bandwidth expression presented in this section gives an upper bound even when the assumptions of the CLT do not hold, e.g., in case of very diverse traffic mix or relatively small number of flows. In addition, it uses the measurement of the aggregate variance, and not the loose descriptors of the traffic policer.

To include the variance, we have to estimate the moment generating function of X_k in a different way. First rewrite the moment generating function of X_k as:

$$\mathbb{E}\left[e^{sX_k}\right] = \mathbb{E}\left[e^{s(X_k - m_k)}\right] e^{sm_k} \tag{4.17}$$

where $e^{s(X_k - m_k)}$ can be bounded by [1]

$$e^{s(X_k - m_k)} \leq 1 + s(X_k - m_k) + \frac{e^{sh_k} - sh_k - 1}{h_k^2}(X_k - m_k)^2 \tag{4.18}$$

for all $h_k \geq m_k, X_k \geq 0$. See Figure 4.1 for an example of the estimation. Note that with this parabolic approximation, we have a product of $(X_k - m_k)^2$, so the expectation in (4.17) is:

$$\mathbb{E}\left[e^{s(X_k - m_k)}\right] = 1 + \frac{e^{sh_k} - sh_k - 1}{h_k^2}\sigma_k^2 \tag{4.19}$$

[1] This overestimation is not the best second order one as can be seen in Figure 4.1. We could use

$$1 + s(X_k - m_k) + \frac{e^{s(h_k - m_k)} - s(h_k - m_k) - 1}{(h_k - m_k)^2} \cdot (X_k - m_k)^2$$

instead, but the resulting formulas require per flow measurements. This tighter parabola is also shown in Figure 4.1.

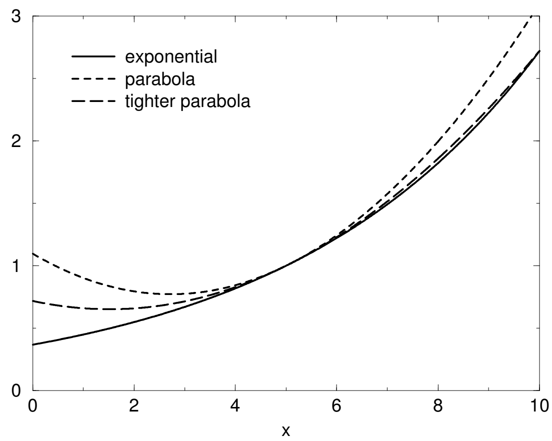


Figure 4.1: Example for bounding of $e^{s(x-m)}$ with parabola, see Eqn. (4.18).

As we intended, this approximation of the moment generating function contains per-flow variances (σ_k) and mean rates (m_k) only. When this modified moment generating function is inserted into (4.7), we get

$$BW(s) \le \frac{1}{s}\sum_{k=1}^{N} \ln\left(1 + \frac{e^{sh_k} - sh_k - 1}{h_k^2}\sigma_k^2\right) + M + \frac{\gamma}{s} \tag{4.20}$$

Finally, we apply the geometric-algebraic inequality, similarly as in the previous section. The product becomes a sum in the expectation, so only the easily measurable aggregate variance S and mean rate M remains, which concludes the proof of the following theorem:

Theorem 4 *If we know the peak rates h_k of flows $k = 1 \dots N$, the mean rate of the aggregate M, and the aggregate variance S, the effective bandwidth takes the form*

$$BW^V(s) = \frac{1}{s}\sum_{k=1}^{N} \ln\left[\frac{S + \sum_{j=1}^{N} \frac{h_j^2}{e^{sh_j} - sh_j - 1}}{N} \cdot \frac{e^{sh_k} - sh_k - 1}{h_k^2}\right] + M + \frac{\gamma}{s} \tag{4.21}$$

The approximation for the optimal s is obtained similarly as in the case of $BW^A(s)$:

$$s_{opt}^V = \sqrt{\frac{18\gamma}{9S + \widehat{H} - H^2/N}}. \tag{4.22}$$

To evaluate the tightness of an effective bandwidth formula, we consider it to be made up of two components: $BW = M + \Omega$, the aggregate mean rate M and an overhead term Ω, which accounts for the burstiness of the traffic. Effective bandwidth formula can be compared by comparing their overhead terms, for the same guarantee ϵ. In case of the Hoeffding bound

BW^H, the overhead depends on the peak rates, but in case of the CLT based BW^N and our BW^V bounds, it also depends on the variance.

Figure 4.2 shows an example of how the variance S influences the overhead. One can observe that the closed form approximation using s^V_{opt} is not very good for small variances, but the numeric optimization stays close to the value obtained from BW^N. The figure shows that $BW^V > BW^N$ which is because BW^V is always an upper bound unlike BW^N, which is not as robust.

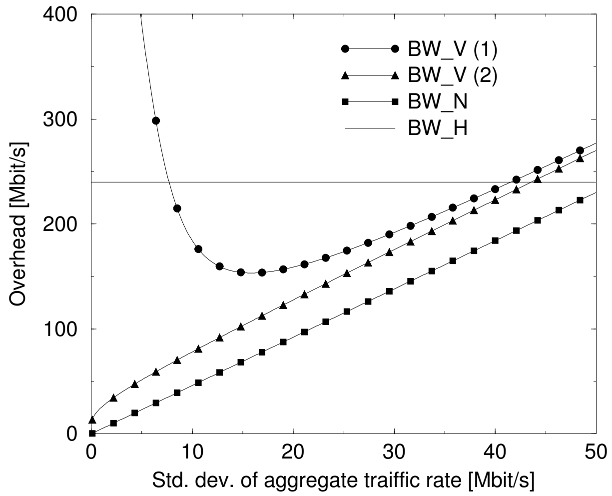


Figure 4.2: The overhead calculated using BW^H, BW^N and BW^V (1:closed form, 2:numerical optimization) as the function of $\sqrt{S}$. 100 sources, $h_k = 10$ Mbit/s, $\epsilon = 10^{-5}$.

In a summary, measuring the variance can significantly reduce the required amount of allocated resources. Nevertheless, the closed form, simple approximation does not perform well in case of small variances. In these cases, we propose the use of simple numeric approximation, or use a modified, artificially inflated value of the variance, where the closed form solution is minimal. The resulting bound is still robust and significantly tighter than the Hoeffding bound, regardless of the composition of the traffic mix.

4.2.4 An Improved Bound using Measurement Groups

Within a DiffServ queue, a large number of flows are multiplexed with diverse statistical properties. Simply measuring aggregates hides these differences. We speculate if we know more about the composition of the aggregate, more precise bounds can be given.

There are two extremes concerning how detailed the measurements can be. The first means that we only measure the total aggregate rate of all flows. Measurements of the total aggregate provide little information about the composition of the traffic mix. On the other extreme we would need measurements separately for every flow. This would require complex, non-scalable router implementation.

In this section, we investigate an alternative method that lies in between these two "extremes". Flows within the same DiffServ class are partitioned into a fixed number of groups. Flows are not further differentiated within a group, so only the statistics of whole groups are measured. We argue, that this type of grouping can be implemented in a scalable, stateless manner.

We expect that the more groups we make, the tighter bound can be given. The presented numeric experiment suggests that a small number of groups are sufficient to approximate the optimal, (but non-feasible), per flow measurement case. The result supports the DiffServ principle of handling only traffic aggregates from a theoretical aspect.

Effective Bandwidth Based on Measurement Groups

Let's group the N flows in a DiffServ class into G sets (groups). G can range from 1 (in this case we have aggregate measurement only) to the total number of flows (in this case per flow measurements are done).

Theorem 5 *Group flows $k = 1 \dots N$ into G groups: $A_i,\ i = 1..G$. Let $n_i = |A_i|$ denote the number of flows in group i. If we know thhe peak rates of flows h_k, and we know the average rate of the groups $M_i = \sum_{k \in A_i} m_k$, the effective bandwidth expression for the grouping case is*

$$BW^G(s) = \frac{1}{s}\sum_{i=1}^{G} n_i \ln\left(\frac{M_i + \sum_{k\in A_i}\frac{h_k}{e^{sh_k}-1}}{n_i}\right) - \frac{1}{s}\sum_{k=1}^{N}\ln\left(\frac{h_k}{e^{sh_k}-1}\right) + \frac{\gamma}{s}. \tag{4.23}$$

where $H_i = \sum_{k \in A_i} h_k$.

Proof of Theorem 5 First, rearrange the terms in the product in (4.11):

$$\prod_{k=1}^{N}\left(m_k + \frac{h_k}{e^{sh_k}-1}\right) = \prod_{i=1}^{G}\left(\sqrt[n_i]{\prod_{k\in A_i}\left(m_k + \frac{h_k}{e^{sh_k}-1}\right)}\right)^{n_i} \tag{4.24}$$

Applying the algebraic-geometric inequality among the groups we get

$$\prod_{i=1}^{G}\left(\sqrt[n_i]{\prod_{k\in A_i}\left(m_k + \frac{h_k}{e^{sh_k}-1}\right)}\right)^{n_i} \le \prod_{i=1}^{G}\left(\frac{1}{n_i}\sum_{k\in A_i}\left(m_k + \frac{h_k}{e^{sh_k}-1}\right)\right)^{n_i} \tag{4.25}$$

The proof concludes if we insert (4.25) into (4.10) of Theorem 2. ■

A derivation of a closed form approximation for the optimal s can be found in Appendix A.1:

$$s_{opt}^{G} = \sqrt{\frac{8\gamma}{\widehat{H} - \sum_{i=1}^{G}\left(2M_i - H_i\right)^2/n_i}} \tag{4.26}$$

We performed a test to see the magnitude of the gain achieved by grouping. Two randomized set of flows were created with similar mean and peak rates within a group. Type-1 and type-2 flows have 100 kbit/s and 1 Mbit/s average peak rates and average mean-to-peak ratios of 0.5 and 0.1 respectively. A variety of traffic mixes were generated, such that the average rate of the total traffic mix is 100 Mbit/s. The effective bandwidth of the random sets was calculated using several formulas. In Figure 4.3a the overhead ($BW - M$) is plotted for BW^H (Hoeffding bound), BW^A and the grouping based expression BW^G. The *x* axis shows the composition of the traffic mix, indicating the percentage of type-1 flows. In the grouping case, flows were sorted into two groups according to their type.

Figure 4.3b shows the ratio of the overheads of BW^A and BW^G. It can be seen that the required overhead suggested by BW^A can be 40% higher than required by BW^G. Without group measurements this bandwidth is wasted. We also note that in this case the overhead suggested by the optimal, but also architecturally expensive, per-flow measurement based bound (BW^O) is better than the grouping bound by less than 2%. This result suggests that more detailed measurements would hardly improve the tightness of the estimation.

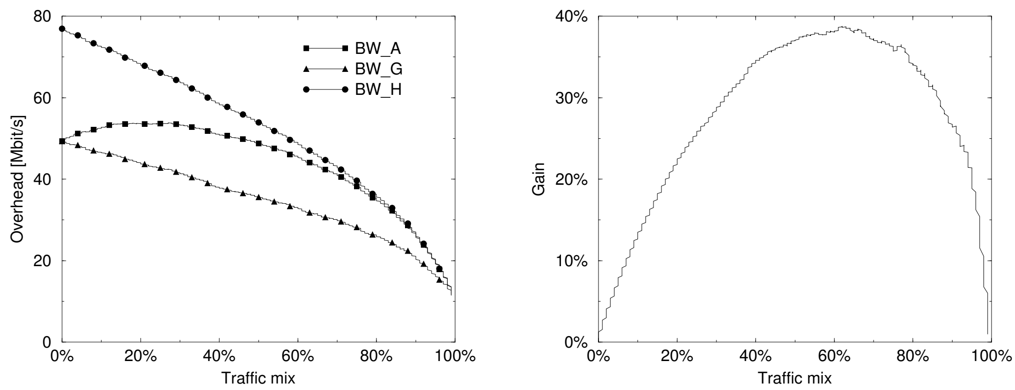


Figure 4.3: Evaluation of several bounds for a varying traffic mixes consisting of two types of sources. The traffic mix changes from consisting only type-1 (0%) to only type-2 sources (100%). a) Overheads of the bounds BW^H, BW^G and BW^A with the appropriate closed form s_{opt} expression. b) Ratio of the overheads obtained using BW^A and BW^G.

Grouping Strategies

The tightness of (4.23) depends on the choices of the number of groups (G) and how flows are classified into groups (A_i). The effective bandwidth expression (4.23) can be optimized for both. For the classification problem a simple rule of thumb exists: if we select statistically “similar” flows into the same group, the geometric mean will be closer to the algebraic in (4.25), resulting in tighter bandwidth estimation.

The term similar denotes flows with $V(k) = m_k + h_k/(e^{sh_k} - 1)$ values close to each other

($V(k)$ is the inner term in the product in (4.24)). The problem is nontrivial for several reasons: s is an unresolved parameter, m_k is not known, since it would need per flow classification. We argue that it is still possible to group flows without complex classification algorithms and without keeping any state in the routers.

The idea originates from the fact that any grouping is better than no grouping; that is, grouping does not have to be optimal. Significant gain can be achieved by simply "guessing" what group a packet should belong to. For example, grouping can be based on any header field in the packets, or locally available information, including, but not limited to:

- DS code point (TOS field);
- IP protocol and port;
- higher layer application header fields (e.g., RTP);
- lower layer protocol fields (e.g., ATM VCI/VPI, MPLS label);
- SLA parameters, traffic contract;
- access type and speed; or
- any other known parameter that affects the traffic of the flow.

We argue that all these fields can be candidates for efficient grouping, since these fields usually separate different types of applications or users, very likely having different statistics. In Figure 4.4 we plot data measured in a company dial-in modem pool, gathered during a four week long period. The two histograms show the distribution of measured average rates of TCP and UDP flows. From the measurement it is evident that even such rough grouping (i.e., based on protocol type) leads to significant difference between the statistics of the two groups. The reason for different statistics lies in the fact that different applications use different protocols. In this case, the majority of UDP traffic is streaming media, while TCP is dominated by WWW applications.

The effect of increasing the number of measurement groups is shown in Figure 4.5. For the experiment a random set of flows is generated (see Figure 4.5a), it contains flows with highly varying statistics (power-law distribution for the peak rates and uniform distribution for the mean-to-peak ratios). The number of groups G is increased to see how much gain is achieved with different number of groups. For a certain G the flows are sorted by their $V(k)$ values and are evenly divided into G groups such that N/G flows fall into each group. Figure 4.5b shows the ratio of overheads of BW^G and BW^A as a function of the number of groups G.

The figure suggests that it is sufficient to use only a small number of groups, and any further increase in the number of groups (e.g. per-flow measurements) does not give significantly higher gain. In practice, this means that the grouping algorithms can be easily implemented in a real network, and little extra complexity required compared to the purely aggregate based methods.

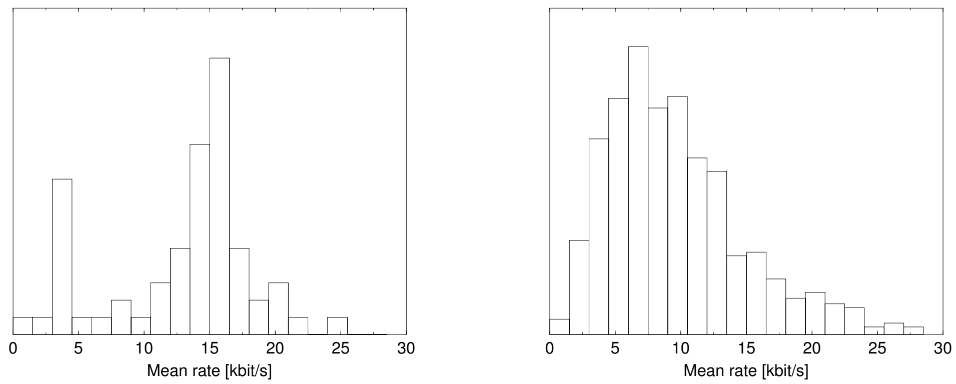


Figure 4.4: Histogram of measured mean rates of modem flows containing a) UDP and b) TCP packets.

4.3 Effective Bandwidth for Delay Sensitive Classes

In the previous sections, we investigated a "bufferless" case. The bufferless approach is suitable to provide throughput guarantees but it is not able to limit small timescale fluctuations that cause buffer build-ups influencing the end-to-end delay. In DiffServ networks, it is envisioned that delay sensitive classes will share separate queues and resources. In these classes, resource dimensioning and admission control have to take delay sensitivity into account.

In this section, we analyze a queue shared by delay sensitive sources. In the DiffServ context, the queue can implement the Expedited Forwarding PHB. The sources are policed with token bucket policers. Similarly to the previous section, the presented bounds rely only on aggregate queue measurements.

4.3.1 Effective Load of a Traffic Aggregate

Define $X_k[t]$, $k = 1 \dots N$ as the number of bits sent by flow k into the network during a time interval of length t. The average rate of flow k is $m_k = \mathbb{E}X_k[t]/t$. The objective is to derive a probabilistic bound for the total number of bits arriving to the queue during a time interval of t:

Definition 2 *Let $B(t)$ be the **effective load** of a traffic aggregate, if it satisfies*

$$\Pr\left(\sum_{k=1}^{N} X_k[t] \geq B(t)\right) \leq \epsilon \tag{4.27}$$

As in the case of the effective bandwidth bounds, our objective is to give simple, closed form expressions. If flow k is policed by a token bucket policer, the amount of traffic is bounded by an envelope for any t [PaGa93]:

$$X_k[t] \leq \sigma_k + \rho_k t \tag{4.28}$$

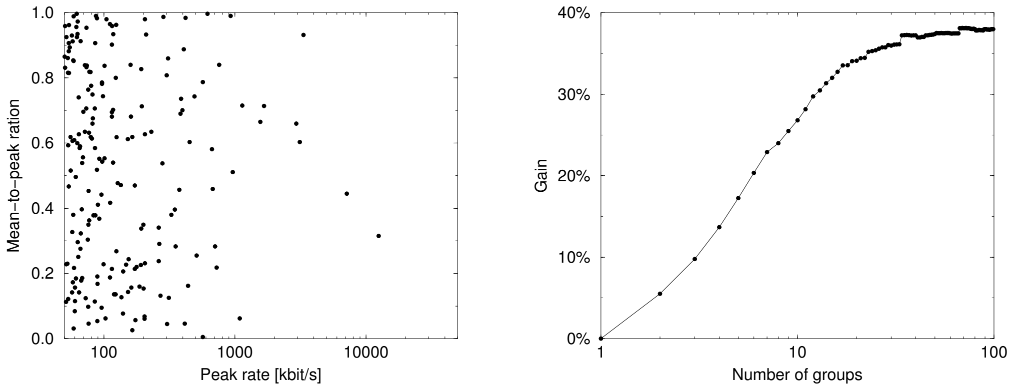


Figure 4.5: a) Random flow set. b) The gain in the overheads compared to the no-grouping case (BW^A). Note that the *x* axis is logarithmic.

or, if the peak rate is policed as well:

$$X_k[t] \leq \min(h_k t, \sigma_k + \rho_k t) \tag{4.29}$$

where ρ_k denotes the token rate and σ_k is the bucket size.

Since $X_k[t]$ are positive bounded random variables, the bounds derived in Section 4.2 can be easily applied to them, but instead of the symbol "X", we have to write "$X_k[t]$". For example, the Hoeffding bound takes the following form:

$$B^H(t) = Mt + \sqrt{\frac{\gamma}{2}\sum_{k=1}^{N} \min(h_k t, \sigma_k + \rho_k t)^2}. \tag{4.30}$$

$B^A(t)$, $B^V(t)$ and $B^G(t)$ can be similarly derived from BW^A, BW^V and BW^G.

As a result, we have a set of bounds that limit the amount of traffic entering a buffer. In the following sections, Section 4.3.2 and 4.3.3, we establish relations between these bounds and the maximum queuing delay in a First In First Out (FIFO) queue. The results can be used to estimate the required resources for a traffic mix, or to limit the traffic mix by admission control in a delay sensitive DiffServ queue.

4.3.2 Bounding Delay by Limiting the Length of the Busy Period

One very simple way to control the delay within a queue is to limit the length of the busy period. The busy period is defined as the time period during which the server is continuously busy. Theorem 6 provides us with a simple test to check the delay guarantees.

Theorem 6 *The delay in a FIFO queue of service rate C is probabilistically bounded by d_{max} if:*

$$B(d_{max}) \leq Cd_{max} \tag{4.31}$$

where $B(t)$ is the effective load of the traffic aggregate.

Proof of Theorem 6 There are two statements in relation to the length of the busy period:

1. If all busy periods are shorter than d, the queuing delay can not be larger than d either, since the queue goes empty within d.

2. If the length of a busy period is longer than d, more than Cd bits arrived during the first d seconds of the busy period.

Thus, if $B(d_{dmax}) \leq Cd_{max}$, the lengths of all busy periods are probabilistically limited by d_{max}. Consequently, the probability that the delay stays within the desired limit is also probabilistically limited by d. ■

Note that this probability does not equal the probability of individual packet delays exceeding the delay limit, since once the delay limit is violated, several packets may suffer large delays. Nevertheless, if this violation probability is kept at sufficiently small level, the packet delay will be also effectively bounded. This is why we call our guarantees "soft".

Theorem 6 can be directly used for admission control or dimensioning. In place of $B(t)$, any of the probabilistic bounds, $B^H(t)$, $B^A(t)$, $B^V(t)$, or $B^G(t)$, can be inserted. The choice of which $B(t)$ is used may depend on implementation preferences. For example, for $B^V(t)$, the variance of the number of bits during time intervals of length d_{max} has to be measured in addition to the average rate.

In Figure 4.6, the number of admitted flows are plotted as a function of the delay guarantee. $B^A(t)$ and $B^H(t)$ are compared for the cases when policing is done for (h, σ, ρ) or just for (σ, ρ). It can be observed that under a certain delay limit (in this case $t = \sigma/(h - \rho) = 8$ ms) the peak rate is a stronger limit than the token bucket and the maximum number of flows is considerably higher than if only (σ, ρ) is policed for. If the delay requirement is above this delay limit, the maximum amount of admissible traffic increases gradually, improving the statistical multiplexing gain. This implies that for DiffServ classes where the per-hop delay tolerance is small, peak rate policing can significantly improve the efficiency of resource estimation.

4.3.3 Bounding Delay by Using the Queue Occupancy Curve

In this section, several bounds based on the probabilistic occupancy curve of the queue are presented. The resulting bounds are tighter, but also more complicated than the ones presented in Section 4.3.2.

Definition 3 *The* ***probabilistic queue occupancy curve*** *$\Delta O(t)$ is defined as a probabilistic upper bound on the queue length $Q(t)$ during a busy period t seconds after the start of a busy period:*

$$\Pr\left(Q(t) \geq \Delta O(t)\right) \leq \epsilon \tag{4.32}$$

where t is measured from the beginning of a busy period.

Lemma 1 *The buffer occupancy curve is related to the effective load:*

$$\Delta O(t) = B(t) - Ct \tag{4.33}$$

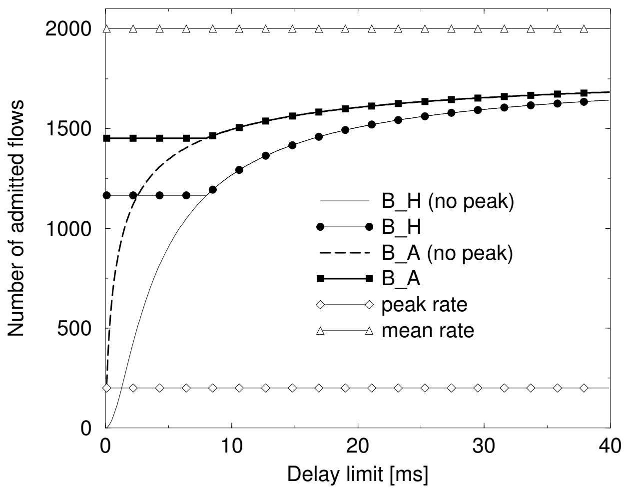


Figure 4.6: Number of admitted flows in the function of d_{max}. The flows are identical with parameters: $h_k = 5$, $m_k = 0.5$, $\rho_k = 1$ Mbit/s and $\sigma_k = 32$ kbit. $\epsilon = 10^{-5}$, $C = 1$ Gbit/s.

The proof of this lemma can be found in Appendix A.2.

Theorem 7 *The maximum delay in the queue is bounded by d_{max} with probability ϵ if:*

$$\max_{t \geq 0} \Delta O(t) \leq C d_{max} \tag{4.34}$$

Proof of Theorem 7 Let t_0 be the value where $\Delta O(t)$ takes its maximum: $\Delta O(t_0) = \max_{t \geq 0} \Delta O(t)$. Relation (4.34) implies that

$$\Pr\left(Q(t_0) \geq C d_{max}\right) \leq \Pr\left(Q(t_0) \geq \Delta O(t_0)\right) \tag{4.35}$$

If we apply Definition 3, we get

$$\Pr\left(Q(t_0) \geq C d_{max}\right) \leq \epsilon \tag{4.36}$$

Since $Q(t_0) \geq Q(t)$ for $\forall t > 0$,

$$\Pr\left(Q(t) \geq C d_{max}\right) \leq \Pr\left(Q(t_0) \geq C d_{max}\right) \leq \epsilon \tag{4.37}$$

That is, the queue length $Q(t)$ is bounded probabilistically by $C d_{max}$. In case of work-conserving FIFO buffer of service rate C, this amount of work is served within d_{max}. ■

Theorem 7 gives a bound on the maximum delay if we know the queue occupancy curve, which can be derived from the effective load of the aggregate using Theorem 1.

In Figure 4.7, several occupancy curves are plotted for several $B(t)$ bounds for a given traffic mix. The maximum of these curves shows the delay that can be guaranteed. Significant difference can be observed between the bounds. For example, $B^A(t)$ always outperforms the Hoeffding bound based formula. Also, token bucket policing combined with peak rate policing (h, σ, ρ) allows tighter guarantees compared to simple token bucket policing (σ, ρ), because it restricts the occupancy curves near $t = 0$.

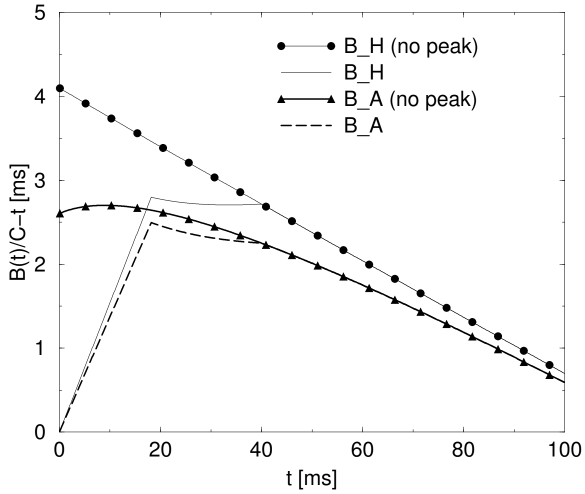


Figure 4.7: $B(t)/C - t$ as the function of t. The traffic mix contains two kinds of sources 2300 and 200 from each type respectively: $h_{12} = [1, 7]$, $m_{12} = [0.3, 1]$, $\rho_{12} = [0.5, 1.5]$ Mbps, $\sigma_{12} = [20, 100]$ kbit, $\epsilon = 10^{-5}$, $C = 1$ Gbps

4.3.4 Practical Delay Bounds Based on the Queue Occupancy Curve

In this section, we derive practical closed-form bounds. The derived bounds have closed forms and are easy to implement. For the ease of discussion, we chose the Hoeffding based effective load formula $B^H(t)$, which has the shortest form, although it is not the tightest.

Delay Bounds for (σ, ρ) Policing

Theorem 8 *Assume that flows $k = 1 \dots N$ are policed by token bucket policers (σ_k, ρ_k). Delay d_{max} is guaranteed in the queue, with probability ϵ, if both relations (4.38) and (4.39) hold.*

$$M + \sqrt{\frac{\gamma}{2} \sum_{k=1}^{N} \rho_k^2} < C. \tag{4.38}$$

$$\sqrt{\frac{\gamma}{2} \sum_{k=1}^{N} \sigma_k^2} < C d_{max} \tag{4.39}$$

Proof of Theorem 8 If flows are policed by a simple token bucket (σ, ρ) and we choose the $B^H(t)$ bound, the occupancy curve takes the following form:

$$\Delta O^H(t) = Mt - Ct + \sqrt{\frac{\gamma}{2}\sum_{k=1}^{N}(\sigma_k + \rho_k t)^2} \tag{4.40}$$

This follows from (4.30) and Theorem 1.

It can be proven that $\Delta O^H(t)$ is a convex function of t. If

$$\lim_{t\to\infty} \Delta O^H(t)' < 0 \tag{4.41}$$

it implies that $\Delta O^H(t)$ decreases monotonously. This is equivalent to checking if

$$M + \sqrt{\frac{\gamma}{2}\sum_{k=1}^{N}\rho_k^2} < C. \tag{4.42}$$

If $\Delta O^H(t)$ decreases monotonously, it takes its maximum at $t \to 0$.

$$\max_t \Delta O^H(t) = \lim_{t\to 0} \Delta O^H(t) = \sqrt{\frac{\gamma}{2}\sum_{k=1}^{N}\sigma_k^2} \tag{4.43}$$

Theorem 7 states that if this expression is less than Cd_{max}, the delay is guaranteed. This concludes the proof. ■

The tests in Theorem 8 are very simple to perform either for flow admission control or offline network dimensioning. Note that (4.38) is similar to the bufferless bound BW^H, with ρ_k as peak rates. It can be regarded as a test for long-term stability. The second test (4.39) is needed to ensure that even during short-term bursts, the delay is guaranteed. If the policer does not allow any burstiness (i.e., $\sigma_k = 0$), in that case (4.38) is equivalent to BW^H indeed.

Delay Bounds for (h, σ, ρ) Policing

If flows are policed for a short-term peak rate as well (h, σ, ρ), a tighter closed from bound can be given compared to the one in the previous section.

Theorem 9 *Assume that flows $k = 0 \ldots N$ are policed by (h_k, σ_k, ρ_k) policers. The delay in the queue is guaranteed to be less than d_{max} with probability ϵ, if the following conditions hold:*

$$M + \sqrt{\frac{\gamma}{2}\sum_{k=1}^{N}\rho_k^2} \leq C \quad \textit{and} \tag{4.44}$$

$$\Delta O^H(t_k) < Cd_{max} \tag{4.45}$$

for all $k = 0 \ldots N$, where $t_k = \sigma_k/(h_k - \rho_k)$, and $t_0 = 0$, and where

$$\Delta O^H(t) = Mt - Ct + \sqrt{\frac{\gamma}{2}\sum_{k=1}^{N}\min{(h_k t, \sigma_k + \rho_k t)^2}}. \tag{4.46}$$

Proof of Theorem 9 It can be proved that (4.46) is a concatenation of convex sections (see Figure 4.7 for an example). The concatenation points or *breakpoints* occur where for any flow $h_k t = \sigma_k + \rho_k t$. The first breakpoint t_0 is defined to be 0.

Between two breakpoints $\Delta O(t)$ is convex, which implies that $\Delta O^H(t)$ has its maximum at one of the breakpoints t_k. It is sufficient to check the relation in Theorem 7 at these points to ensure the delay guarantee. Regarding the global maximum of $\Delta O(t)$, two cases are possible:

1. No delay can be guaranteed as $\Delta O(t)$ diverges to infinity as $t \to \infty$. To test if this is the case (4.38) can be used because after the last breakpoint (4.46) takes the same form as (4.40).

2. $\Delta O(t)$ takes its maximum at one of the breakpoints. If the maximum value is below Cd_{max}, the delay can be guaranteed. ■

For example, in case of the B^H curve of Figure 4.7, the second case holds and the maximum is taken at the first breakpoint. The maximum value and so the guaranteed delay is less than 3 ms in this case.

When applying the bounds for admission control, checking the value of $\Delta O(t)$ in several breakpoints may be computationally expensive. However, it can be expected that many flows will have similar descriptors (service classes), in which case, they share breakpoints, reducing the number of points to be tested to just a few.

The same algorithm can be used if flows are described by multiple token bucket descriptors. (Only the number of breakpoints increases.) Multilevel descriptors characterize flows better and lead to more efficient admission control [ZhKn94]. Flows with descriptors of different detail e.g., $(h), (\sigma, \rho), (h, \sigma, \rho)$ or as complex as $(h, \sigma_1, \rho_1, \sigma_2, \rho_2, \ldots)$ can be mixed in the same queue using the same formula.

To compare the various methods for delay sensitive classes, we generated a random traffic mix, and plotted the number of flows from that traffic mix that fit into the link. Figure 4.8a shows the bucket sizes and token rates of the flow mix (the average of m_k was 500 kbit/s). For tight delay requirements significantly more flows could have been admitted from the same flow set if they were policed for (h, σ, ρ) compared to (σ, ρ), see Figure 4.8b. This observation underlines the importance of peak rate policing. Another observation is that the occupancy curve based method significantly outperforms the method based on the busy period limitation. Finally, the resource efficiency of both probabilistic methods is much higher than the deterministic method of Parekh et al. [PaGa93, PaGa94]. Due to the statistical multiplexing of sources, the worst-case scenario assumed by deterministic bounds is too conservative. This is the reason for the significantly better performance of the statistical methods observed in Figure 4.8b.

4.4 Supporting Multiple DiffServ Classes using Static Priority Scheduler

In this section, a complex DiffServ router is discussed that has multiple queues in every interface for several service classes served with SPQ scheduling. The task of resource management is ensuring the delay and loss guarantees in each DiffServ queue.

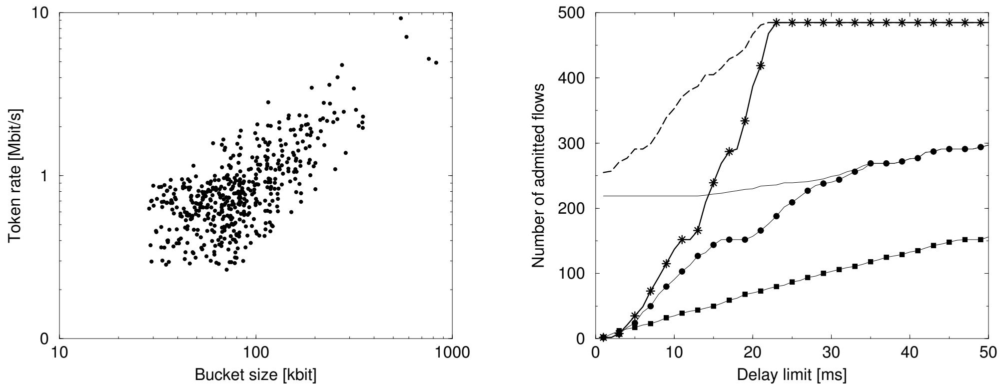


Figure 4.8: Comparison of different methods for delay sensitive class. a) Random traffic mix composition. b) The number of admitted flows from the traffic mix. Busy period limitation: (solid line) (h, σ, ρ) policing, (circles) (σ, ρ) policing. Occupancy curve limitation: (dotted line) (h, σ, ρ) policing, (stars) (σ, ρ) policing. Deterministic bound by Parekh and Gallager: (squares).

Assume we have L priority levels with one queue at each level, served in a fixed priority order. Let $\mathcal{X}_{a,b,c,\ldots}$ denote the set of flows in queues $a, b, c, \ldots$. The first R (high priority) queues provide different guaranteed delay d_k ($d_i \leq d_j, i \leq j$) with probability ϵ_k. The queues $R + 1 \ldots L$ provide assured service with saturation probabilities ϵ_k ($\epsilon_i \leq \epsilon_j, R < i \leq j$). See Figure 4.9.

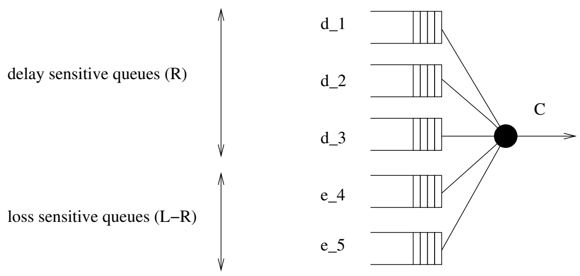


Figure 4.9: Guaranteeing multiple service classes with SPQ scheduling.

4.4.1 Bounds for Assured Forwarding Queues

Theorem 10 *Denote $BW^{\epsilon}[A]$ as the effective bandwidth of a flow set A with probability ϵ. The saturation guarantee ϵ_k is ensured in the assured forwarding queue k, if*

$$BW^{\epsilon_k}[\mathcal{X}_{1\ldots k}] \leq C \tag{4.47}$$

where any previously derived form of the effective bandwidth BW can be used.

Proof of Theorem 10 Priority scheduling ensures that the traffic of queue k can only be affected by traffic in k or higher priority queues, but packets in lower queues have no effect (we neglect the impact of a possible, single, lower priority packet already in service). Therefore, to check the guarantees of queue k, only the traffic of queues $j \leq k$ has to be taken into account.

To guarantee the saturation probability ϵ_k in the assured queue k ($R < k \leq L$) we want to ensure that

$$\Pr\left(\sum_{i \in \mathcal{X}_{1\ldots k}} X_i \geq C\right) \leq \epsilon_k \tag{4.48}$$

Theorem 10 follows from the definition of the effective bandwidth. ■

Using the above calculation, statistical multiplexing is exploited not only within a queue, but among queues as well, since

$$BW^{\epsilon}[\mathcal{X}_{1\ldots k}] \leq \sum_{j=1}^{k} BW^{\epsilon}[\mathcal{X}_j] \tag{4.49}$$

That is, the amount of resources required for a set of queues served by SPQ scheduling is less than the sum of the effective bandwidths of individual traffic aggregates in separate queues.

4.4.2 Bounds for Delay Sensitive Queues

Delay sensitive queues are the first R queues in the system (Figure 4.9). As in the case of assured queues, when analyzing the delay of a queue, only the traffic in higher queues has to be taken into account. To estimate the delay in a queue, we have to estimate how much service remains from higher queues. For this, we can rely on per-queue traffic measurements. The following theorem establishes a relation between the effective loads of queues and the queue occupancy.

Theorem 11 *The delay limit d_k, in queue k, is exceeded with probability less than $2\epsilon_k$ if, for $\forall t \geq 0$:*

$$B^{\epsilon_k}[\mathcal{X}_{1\ldots k}](t) - Ct \leq Cd_k - B^{\epsilon_k}[\mathcal{X}_{1\ldots k-1}](d_k) \tag{4.50}$$

where $B^{\epsilon}[\mathcal{A}](t)$ is the effective load of flow set A and violation probability ϵ.

In Theorem 11, $B^{\epsilon_k}[\mathcal{X}_{1\ldots k}](t)$ is the effective load of all queues through $1\ldots k$, but with the guarantee of the lowest priority queue among them ϵ_k. The following Lemmas are required to prove the theorem. (The proofs of the Lemmas can be found in Appendix A.3).

Lemma 2 *t seconds after the start of the busy period of queue k, the queue occupancy of queue k is bounded probabilistically by*

$$\Pr\left(Q_k(t) \geq B^{\epsilon_k}[\mathcal{X}_{1\ldots k}](t) - Ct\right) \leq \epsilon_k \tag{4.51}$$

Lemma 3 *The amount of service queue k receives during time t is bounded by*

$$\Pr\left(U_k(t) \le Ct - B^{\epsilon_k}[\mathcal{X}_{1\dots k-1}](t)\right) \le \epsilon_k \tag{4.52}$$

Lemma 4 *If X and Y are two independent random variables, and $\Pr(X \ge c) \le \epsilon$ and $\Pr(Y \le c) \le \epsilon$,*

$$\Pr\left(X \le Y\right) \le 2\epsilon \tag{4.53}$$

Proof of Theorem 11 If for $\forall t > 0$

$$B^{\epsilon_k}[\mathcal{X}_{1\dots k}](t) - Ct \le Cd_k - B^{\epsilon_k}[\mathcal{X}_{1\dots k-1}](d_k) \tag{4.54}$$

holds, Lemma 2, Lemma 3 and Lemma 4 imply that

$$\Pr\left(Q(t) \ge U_k(d_k)\right) \le 2\epsilon_k \tag{4.55}$$

for $\forall t > 0$. The expression limits the queue occupancy below a certain value $U_k(d_k)$. If the queue occupancy is less than the amount that can be served from this queue during time d_k, the queuing delay does not exceed d_k only with a small probability $2\epsilon_k$. ■

4.4.3 Analysis of a Simple DiffServ Implementation

In this section, we apply the theorems developed for multiple SP queues. For simplicity, we investigate a DiffServ router using just three queues for three traffic classes: the highest priority queue serves EF, the second AF and the lowest priority queue serves BE traffic. The three queues are served by a SPQ scheduler, at a rate of C.

Based on Theorem 11, the delay in the delay sensitive EF queue is guaranteed if

$$B^{\epsilon_1}[\mathcal{X}_1](t) - Ct \le Cd_k \tag{4.56}$$

This has the same form as the single queue case, so Theorem 8 can be applied here to check the delay guarantees of the EF queue. Both relations must hold for the guarantee:

$$M_1 + \sqrt{\frac{\gamma_1}{2}\sum_{k=1}^{N_1}\rho_k^2} < C \quad \text{and} \quad \sqrt{\frac{\gamma_1}{2}\sum_{k=1}^{N_1}\sigma_k^2} < Cd_{max} \tag{4.57}$$

where $\gamma_1 = -ln\epsilon_1$, ρ_k, σ_k, $k = 1\dots N_1$ are the token bucket parameters of flows in the EF queue, and M_1 is the average load of the EF queue.

The rate guarantee of the AF queue can be checked using Theorem 10, which takes the following simple form:

$$BW^{\epsilon_2}[\mathcal{X}_{1,2}] \le C \tag{4.58}$$

Using the effective bandwidth expression BW^A from Theorem 3 we get

$$BW^A(s) = \frac{1}{s}\sum_{k=1}^{N}\ln\left[\frac{M + \sum_{j=1}^{N}\frac{h_j}{e^{sh_j}-1}}{N} \cdot \frac{e^{sh_k}-1}{h_k}\right] + \frac{\gamma_2}{s} \le C \tag{4.59}$$

and

$$s_{opt} = \sqrt{\frac{8\gamma_2}{\widehat{H} - (2M - H)^2/N}} \qquad \text{where } H = \sum_{k=1}^{N} h_k \text{ and } \widehat{H} = \sum_{k=1}^{N} h_k^2 \tag{4.60}$$

where $\gamma_2 = -ln\epsilon_2$, $N = N_1 + N_2$ the total number of flows in both AF and EF queues, and $M = M_1 + M_2$, the total average rate the two higher priority queues. The peak rate of the flows in the AF queue h_k is either an admitted value, or simply the access line speed. The peak rate of the flows in the EF queue can be approximated by their token rates $h_k \approx \rho_k$ if we intend to guarantee the AF rate over a longer time period, i.e., over more than of $T \approx \sigma/\rho$, which is still a reasonable timescale for the AF class.

4.5 Conclusions

In this chapter, we introduced a set of resource estimation methods for Differentiated Services networks. These methods can be used for dimensioning, traffic engineering or admission control. The specialties of DiffServ were taken into account, namely aggregate traffic handling, simple scheduling and traffic conditioning at the edges. We derived several effective bandwidth expressions for assured and delay sensitive classes. These methods utilize simple measurements of the aggregate traffic in DiffServ queues, such as mean rate or variance. We showed that the efficiency of bounds utilizing per flow measurements could be approximated by using a small number of measurement groups. Methods were given for delay sensitive traffic classes, utilizing token bucket, peak rate or more complex descriptors and queue measurements. It was shown, that peak rate policing considerably improves resource efficiency. Finally, a practical example for a SPQ based DiffServ router implementation was discussed.

The proposed methods can be used in the core or the access part of a DiffServ network. The assumptions on traffic flow statistics are general to any DiffServ enabled network, however, the assumptions on the properties of lower layers limits the application of the methods to wired networks. The most important such property is that the managed links are highly reliable and their service rate can be regarded as being constant. In the case of wireless networks, these two properties cannot be usually guaranteed. The next chapter discusses the wireless environment: the challenges of the radio channel, the open issues in wireless DiffServ provisioning, and our proposed solutions.

Chapter 5

Supporting Service Differentiation in Wireless Packet Networks

In the past several years the Internet has started to penetrate the wireless world with the result that the emphasis in wireless communication will be the support of TCP/IP based applications, in contrast to the current circuit switched voice. It is envisioned that TCP/IP will be the glue for all applications in future mobile environments, many of them requiring better than best-effort services. Wireless access may be considered just another hop in the communication path. Therefore, it is desirable that the architecture supporting quality assurances follows the same principles in the wireless network as in the wireline Internet assuring compatibility between the wireless and wireline parts.

There are two principal approaches to support better than best-effort services for Internet based services in a future wireless network. The first approach starts from the conventional circuit switched paradigm and extends it with datagram services. These systems are characterized by strict control over both the wireline and wireless resources, motivated by the argument that such control, with its complex and sophisticated mechanisms and protocols, is necessary to maintain good quality in the wireless environment [MoPa92][GPRS][NLB99].

Another increasingly popular approach is based on an important Internet design principle that mandates that only minimal control and signaling is viable, since only simple mechanisms can accommodate the diversity of applications in the Internet, let alone unforeseen future wireless applications. A good example for such a wireless technology is the IEEE 802.11 standard [IEEE802.11], which in itself does not guarantee anything other than best-effort service for mobile hosts using the Distributed Coordination Function (DCF). However, the IEEE 802.11 DCF enables the fast installation of simple wireless access networks, with minimum management and maintenance costs, and with virtually no requirement for cell planning. Similar distributed algorithms are analyzed and compared in [Bhar98] [BDSZ94].

In the case of ad-hoc wireless networks, there is no notion of a central entity. The dynamic nature of ad-hoc networks makes it very difficult to dynamically assign a central controller, and maintain connection, reservation and scheduling states, not to mention the difficulty of handling overlapping coverage areas, in which case, nearby mobile hosts need to discover and negotiate resources. Instead of introducing complex layer two signaling, distributed algorithms attempt to solve these problems in a more straightforward, although possibly less radio efficient manner.

In this chapter, we propose a set of algorithms that form a fully distributed wireless differentiated services network based on:

- a distributed, differentiated services capable MAC;
- a distributed radio resource monitoring mechanism;
- service quality estimation; and
- distributed traffic and admission control.

Each of these components performs a well-defined task and can be implemented in a fully distributed manner, without the need of a central host. While our framework is generally applicable to distributed wireless access schemes, we design, implement and evaluate our framework within the context of existing wireless technology. Service differentiation is based on the IEEE 802.11 DCF. Supporting better than best-effort service over such a shared wireless channel using distributed control algorithms presents a number of challenges, however.

The first challenge relates to the difficulty in providing service differentiation at the distributed wireless MAC layer. The impact of packet collisions, hidden terminals, fading and interference suggests that such a radio environment lends itself more to soft service assurances rather than deterministic ones. In this work, we take that lead from this observation and attempt to quantify the level of assurance and service differentiation that can be delivered to Internet applications. This means that under such a regime quality measures can only be probabilistically guaranteed where relative quality differentiation can be assured for different service classes.

Providing differentiated services in this manner requires that the radio MAC supports some degree of separation between different types of services. We propose a modified IEEE 802.11 radio MAC algorithm for mobile hosts and base stations. The proposed MAC ensures that all packets sent by a mobile host are differentiated and, more importantly, that differentiation is effective among packets sent by other mobile hosts as well.

Providing service differentiation solely at the radio interface is insufficient to enable predictable behavior for individual traffic types, however. This leads to our next challenge. Network cells may overlap significantly and service differentiation has to be maintained across cells. The probabilistic assurances offered by such a wireless differentiated services MAC itself cannot ensure that traffic levels experienced by a mobile host are not only relatively better, but kept within some absolute limit for acceptable application quality. We address this challenge by proposing a distributed solution without the need for any central control over multiple cells. In particular, we propose a distributed traffic control algorithm, which maintains the traffic load such that the relative assurances offered by a differentiated services MAC also meet the absolute limits required by the applications using better than best-effort service.

In response to these challenges, we develop the *Virtual MAC (VMAC)* and *Virtual Source (VS)* algorithms that monitor the capability of the radio channel and passively estimate whether the channel can support new service demands (e.g., delay and loss) taking into account both local conditions and interference caused by external effects or overlapping cells. The Virtual MAC channel monitoring capability is capable of collecting information about all transmissions in the proximity of a mobile host. Mobile hosts utilize this information to estimate the quality experienced by other mobiles. The difficulty of estimation in this environment is that there is little relationship between the monitored channel load and the delay or loss statistics of the channel. The Virtual MAC and Virtual Source algorithms, which, based on the information provided by a passive channel monitor, can efficiently estimate the necessary quality metrics for different traffic classes. These "virtual algorithms" are passive and do not load the channel, avoiding further increases of load in potentially congested wireless networks.

Based on the service quality estimations obtained from the virtual monitoring algorithms, mobile hosts and base stations determine whether a new session with a particular service level requirement should be admitted or not. In this chapter, we simplify traffic control and propose an admission control solution that simply accepts or rejects real-time sessions. Admission is granted if the average delay estimated by the Virtual Source algorithm in the last time period falls within a certain delay limit. We show that if all nodes use passive monitoring and base their admission decisions accordingly, a globally stable state can be maintained even in multicell environments.

In this chapter, we present the design, implementation and evaluation of a framework for wireless differentiated services within the context of an IEEE 802.11 network. The principles that underpin our distributed approach are based on minimal control and signaling. While our implementation is evaluated within the context of IEEE 802.11 the algorithms that support service differentiation, radio monitoring and admission control are more generally applicable. We recognize that such an approach can only deliver softer assurances in comparison to more tightly coupled control systems. We argue, however, that distributed control is more scalable (i.e., provides minimum coupling between architectural components), extensible (i.e., one component can be replaced or improved without the need to change other system components) and flexible (i.e., in accommodating new and diverse needs of applications).

The structure of the chapter is as follows. Section 5.1 presents the related work on wireless service differentiation and distributed control. Section 5.2 discusses and analyses the achievable service differentiation using a distributed approach. We analyze the delay experienced by a mobile host implementing the IEEE 802.11 Distributed Coordination Function and derive a closed form formula. We then extend the Distributed Coordination Function with the capability to tune and set the backoff mechanisms to provide service differentiation for delay sensitive and best-effort traffic based on the results from the analysis. In Section 5.3, we introduce the Virtual MAC, which estimates key MAC level statistics related to service quality such as delay, delay variation, packet collision and packet loss. We show the efficiency of the Virtual MAC algorithm through simulation, and in Section 5.4, we implement and evaluate the Virtual MAC in an experimental differentiated services wireless testbed. In Section 5.5, we present the Virtual Source algorithm, which utilizes the Virtual MAC to estimate application level service quality. The Virtual Source allows application parameters to be tuned in response to dynamic channel conditions based on "virtual delay curves". In Section 5.6, we demonstrate through simulation that when these distributed virtual algorithms are applied to the admission control of the radio channel. Finally, we present some concluding remarks and discuss future work.

5.1 Related Work

Effective wireless MAC protocols must find a good balance between the added complexity of offering service guarantees for multiple service classes, most efficient use of available resources and the ability to react promptly to failed transmissions [AMMP99]. A number of MACs intended for third generation protocols are also analyzed in [AMMP99], some of which offer probabilistic guarantees. In general, these MAC protocols and wireless algorithms rely on centralized control.

In [LBS97] [NLB99] a numbers of wireless scheduling algorithms are analyzed, several of which approximate optimal fluid fair scheduling even in the presence of location-dependent error burst. However, these mechanisms rely on centralized control and the polling of backlogged mobile hosts. These algorithms are analyzed using short memory models (e.g., CBR, Poisson and MMPP), which have been shown to be inefficient when modeling real TCP/IP traffic [PaFl95].

A distributed architecture to support weighted rate differentiation among flows is introduced in [NKSB99]. This proposal works in an end-to-end manner, where the end hosts adjust their rate using the Additive Increase Multiplicative Decrease (AIMD) algorithm. Instead of using packet loss, the AIMD actions are based on the observed end-to-end packet separation, which is treated as a sign of congestion. The algorithm works over low bandwidth links assumes that sources are greedy.

The IEEE 802.11 Point Coordination Function (PCF) is intended to support real-time services by using a centralized polling mechanism. This mechanism is not supported by most current wireless cards, however. In addition, cooperation between PCF and DCF modes leads to poor performance [VeZa95].

We argue that distributed control for supporting real-time services is more efficient and flexible than centralized control in the case of highly bursty traffic. We argue, that the basic IEEE 802.11 DCF standard, which is not capable of supporting better than best-effort services, can in fact be extended to support service differentiation. The DCF mechanism of IEEE 802.11 has been investigated in numerous papers.

In [SoKr96] a distributed solution for the support of real-time sources over IEEE 802.11 is discussed, which modifies the MAC to send short transmissions to gain priority for real-time service. It is shown that this approach is capable of offering bounded delay. One disadvantage of the design [SoKr96] is that it is optimized to meet the service needs of isochronous traffic sources, which can be a significant limitation for applications with variable data rates.

The fairness of distributed control is investigated in [NKGB98]. A theoretic analysis of the protocol can be found in [CCG98]. Analysis and protocol enhancements for the DCF are presented in [BFO96] [WSFW97] [WWEW95]. Shared medium access in case of multicell environments is analyzed using simulation in [ChLe99].

5.2 Distributed DiffServ Enabled Wireless Mac

Providing differentiated services in a mobile environment requires that the radio MAC supports some degree of separation between different types of services. This separation is based on the DiffServ field in IP packets [BBCD98]. A “DiffServ enabled MAC” has to ensure that available radio resources are shared among active users, while at the same time ensuring that different traffic types receive service in a differentiated manner. The ideal radio MAC is adaptive and robust to both internal and external dynamics; that is, it offers effective protection for the differentiated traffic classes against traffic fluctuations in lower classes. The MAC should also be robust to changes in the external environment, for example, growth of traffic in a cell must have a predictable and limited effect on the delay and loss experienced by all service classes in neighboring cells.

We argue that decentralized and adaptive mechanisms can more efficiently solve these problems in comparison to centralized ones. Distributed control of the radio resources may result in more productive use of radio resources. Distributing control of the radio resources allows mobile hosts within the same class to compete for radio resources and achieve acceptable fairness, while at the same time offering differentiated access to different service classes.

We propose a simple modification of the IEEE 802.11 radio MAC algorithm that runs in mobile hosts and base stations. The proposed MAC ensures not only that packets sent by the host itself are differentiated, but more importantly, that differentiation is effective among packets sent by other mobile hosts as well. Furthermore, IEEE 802.11 network cells may overlap significantly where service differentiation has to be maintained across cells. We show how this can be achieved in a distributed manner without any central control over multiple cells.

5.2.1 IEEE 802.11 MAC Distributed Coordination Function Protocol

The IEEE 802.11 MAC DCF protocol is a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. In the DCF mode, a station must sense the medium before initiating the transmission of a packet. If the medium is sensed as being idle for a time interval greater than a Distributed Inter Frame Space (DIFS) then the mobile host transmits the packet. Otherwise, transmission is deferred and a backoff process is entered. Specifically, the station computes a random value in the range of 0 to the so-called Contention Window (CW). A backoff time interval is computed using this random value: $T_{backoff} = Rand(0, CW) * T_{slot}$, where T_{slot} is the slot time [IEEE802.11]. This backoff interval is then used to initialize the backoff timer. This timer is decreased only when the medium is idle. The timer is frozen when another station is detected as transmitting. Each time the medium becomes idle for a period longer than DIFS, the backoff timer is periodically decremented once every slot-time.

As soon as the backoff timer expires, the mobile host accesses the medium. A collision occurs when two or more mobile hosts start transmission simultaneously in the same slot. An acknowledgement is used to notify the sending station that the transmitted frame has been successfully received. If an acknowledgement is not received, the station assumes that the frame was not received successfully and schedules a retransmission, reentering the backoff process. To reduce the probability of collisions, after each unsuccessful transmission attempt, the Contention Window is doubled until a predefined maximum (CW_{max}) is reached. After a successful or unsuccessful frame transmission, if the station still has frames queued for transmission, it must execute the new backoff process.

To deal with the hidden terminal problem the MAC protocol can use a short Request To Send (RTS) – Clear To Send (CTS) negotiation before sending a data packet. This reduces the collision probability for data packets but increases the protocol overhead.

5.2.2 Delay Analysis of the Distributed Coordination Function

Previous work has analyzed the IEEE 802.11 DCF mode from several different perspectives, including fairness, throughput and the effect of hidden terminals. We are interested, however, in analyzing the kind of delay guarantees that can be achieved using DCF. Furthermore, we would like to determine how sensitive these guarantees are to certain channel conditions and MAC parameterization, (e.g., channel utilization, average packet size, contention window sizes). We derive a closed form formula for the delay of the packets originated from a single traffic flow on a channel occupied with background traffic. We use this analysis to guide the configuration of our modified DCF MAC.

Denote the mobile host sending the traffic flow under investigation as the "tagged host", and all other packets generated by other mobile hosts as the background traffic. Assume that each packet in the background has a transmission time L, which duration includes the time needed for RTS/CTS/ACK transmissions as well. Assume that the time between the last bit of a background packet and the first bit of the next background packet is exponentially distributed with average $1/\lambda$. Also assume the tagged traffic only occupies a small portion of the total channel utilization, (i.e., its effect on the background traffic is negligible).

The average channel utilization U can be approximated as,

$$U = \frac{L}{L + 1/\lambda} \tag{5.1}$$

$$\lambda = \frac{U}{L - L \cdot U} \tag{5.2}$$

When a tagged packet arrives, the mobile host senses the channel and sends the packet if the channel appears to be idle. If the channel is busy or a collision occurs, the MAC invokes the backoff procedure and delays the transmission, otherwise, the tagged packet is sent. Assume each tagged packet has a transmission time m, and $m < L$. Denote d' as the average delay conditional on the backoff procedure. The average delay d of the tagged packet can be approximated as,

$$d = U \cdot d' + (1 - U) \cdot m \tag{5.3}$$

Denote d'_i as the total deferred time during the i-th backoff period. According to the 802.11 protocol, the backoff timer is only decreased when the channel is idle. Denote b_i as the random deferred time chosen by the DCF algorithm during the i-th backoff, where b_i is a uniformly distributed random variable in the interval $[0,\, CW_i]$ times the length of a backoff time slot T_{slot}. During the i-th backoff period, a number of background packets k_i are sent. Because the idle time between two background packets is exponentially distributed, k_i is a Poisson random variable with average λb_i. In the first backoff, the delay also includes the residual background packet length L' , which causes the backoff in the first place. In the subsequent backoffs caused by collisions, the delays include the length of the colliding background packet L.

Adding all the above together, the i-th deferred time d'_i can be written as,

$$d'_i = \begin{cases} L' + k_i L + b_i \text{ for } i = 1 \\ k_i L + b_i + L \text{ for } i > 1 \end{cases} \tag{5.4}$$

The probability of collision after a backoff, denoted as p, can be estimated as the probability that a transmission attempt of a background packet starts exactly in the same time slot as chosen by the tagged host, otherwise the tagged station would sense the packet and could avoid collision.

$$p \approx \lambda T_{slot} \tag{5.5}$$

The average value of the total accumulated deferred time, $d' = \mathrm{E}[\sum d_i]$, takes into account occasional retransmissions and consecutive backoffs, and can be estimated as,

$$\begin{aligned} d' &= \sum_{i=1}^{\infty} \mathrm{E}\Big[\sum_{j=1}^{i} d'_j \mid \text{i backoffs}\Big](1-p)p^{i-1} + m \qquad (5.6) \\ &= \sum_{i=2}^{\infty} \Bigg(\sum_{j=2}^{i} \mathrm{E}\Big[(k_j+1)L + b_j \mid \text{i backoffs}\Big] + \\ &\quad \mathrm{E}\Big[L' + k_1 L + b_1 \mid \text{i backoffs}\Big]\Bigg)(1-p)p^{i-1} + \\ &\quad \mathrm{E}\Big[L' + k_1 L + b_1 \mid \text{1 backoff}\Big](1-p) + m \qquad (5.7) \end{aligned}$$

The contention window CW ranges from $2^{W_{min}}$ to $2^{W_{max}}$. In the j-th backoff period the backoff time b_j is chosen randomly in the range of $[0, 2^{W_{min}+j-1}]T_{slot}$ until we reach the maximum backoff time, when it is chosen from $[0, 2^{W_{max}-1}]T_{slot}$. The average backoff time is thus,

$$\mathrm{E}[b_j] = \begin{cases} T_{slot} \cdot 2^{W_{min}+j-2} \text{ for } 1 \le j \le W_{max} - W_{min} + 1 \\ T_{slot} \cdot 2^{W_{max}-1} \text{ for } j > W_{max} - W_{min} + 1 \end{cases} \tag{5.8}$$

In the j-th backoff period, the randomly chosen backoff time is b_j. Given this choice, the average number of background packets that arrive before the backoff timer expires is $\mathrm{E}[k_j \mid b_j] = \lambda b_j$. The average number of packets is thus $\mathrm{E}[k_j] = \lambda \mathrm{E}[b_j]$. The average residual packet time is $E[L'] = L/2$.

Let $u = W_{min} - 1$, $v = W_{max} - W_{min}$. Given these notations, the final, closed form result is,

$$d' = 2^u \cdot T_{slot} \cdot (L\lambda + 1) \left(\frac{1-(2p)^{v+1}}{1-2p} + 2^v \frac{p^{v+1}}{1-p} \right) + \frac{L}{1-p} - \frac{L}{2} + m \tag{5.9}$$

Put d' in equation (5.3), we have the estimated average delay d as a function of the channel utilization U.

Figure 5.1 shows a comparison between the analysis and the measured delay from our differentiated services wireless testbed. We compared the measured average delay of a tagged host at increasing levels of background traffic loads. The tagged session generates 120 byte long packets every 0.02 seconds, the length of background packets is 1500 bytes and the channel rate is 11 Mbps. The background traffic rate is gradually increased to the saturation point in small incremental steps. At every step the average delay of tagged packets is calculated. The result shows that the estimated average delay closely matches the measured delay. Section 5.4 presents detailed description of our wireless testbed and the configuration for these results.

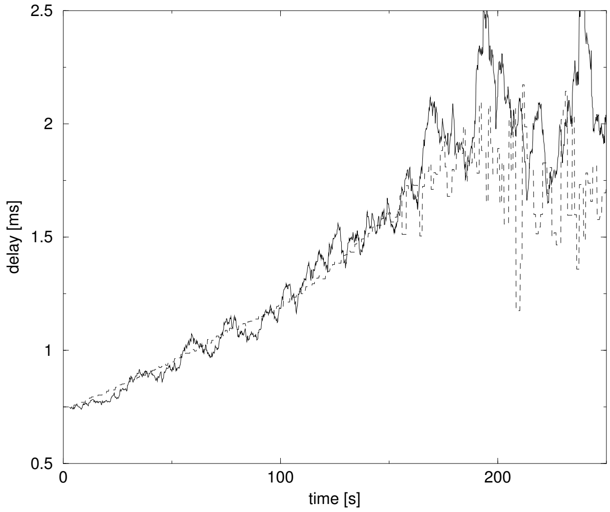


Figure 5.1: The comparison of the analysis and measured results. Packet delays of single session vs. increasing background load (running average) for a channel rate of 11 Mbps.

5.2.3 Discussion on Backoff Timers and Service Differentiation

We have previous described how initial values used by the backoff procedure are determined using the CW parameter, which increases exponentially towards an upper bound as the backoff procedure is reset for a given transmission. In other words, the more transmission attempts for a given packet, the larger the CW, and so the longer the time between transmission attempts.

Backoff times are set to a random value in the range $[0, CW] * T_{slot}$. After a collision, a new backoff time is chosen but with an increased CW value. After every successful transmission, CW is reset to an initial value CW_{min}. We propose to support at least two service classes, high priority (i.e., premium service) and best effort. Setting different CW_{min} values for each service class means that for two or more packets entering a backoff procedure at the same time, but with different CW_{min} values, the packet with the smaller value of CW is more likely to be transmitted first. Even if collisions occur, all MACs increase CW at the same rate and it is likely that the CW of high priority packets remain lower than that of low priority packets. with the result of experiencing smaller average delays. Intuitively, even during highly congested periods, all classes have increased delays but still in a differentiated manner.

By decreasing the maximum CW limit, CW_{max}, for a service class, the maximum backoff time can be limited during congestion. This limits the range of congestion control, thus we trade lower delay for increased collision probability, and eventually larger packet loss ratio. Nevertheless, we argue that for better than best-effort services it is preferable to drop a packet than to delay it excessively.

The analysis in the previous section can be used to address the issue of how backoff values impact the average MAC delay for different levels of channel loads. Only W_{min} and W_{max} values have to be modified in the equations accordingly. Figure 5.2 shows the estimated average delays for increasing levels of utilization for several choices of $CW_{min} = 2^{W_{min}} = 8, 16, 32$

and 64.

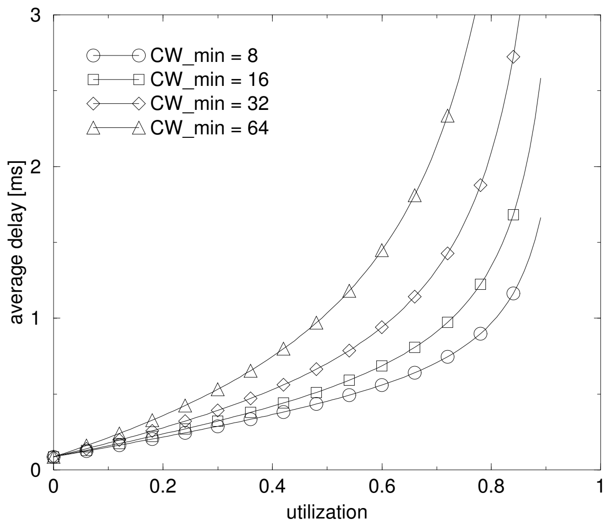


Figure 5.2: Estimated average delays for different values of CW_{min} and increasing level of channel utilization, while $CW_{max} = 1024$ is kept constant. Channel rate is 11 Mbps.

The analysis shows that by setting different values of CW differentiated levels of service can be achieved. We note, however, that the results of the analysis should be treated as qualitative results only since some of the assumptions made in the model are too simple when one considers highly bursty traffic scenarios. In the next section, we simulate realistic traffic mixes for TCP and UDP sources, and explore the achievable service differentiation using these simple means of control.

5.2.4 Evaluation of the Modified MAC to support Service Differentiation using Simulation

Initially, the degree of separation between high priority and best-effort traffic for different values of CW_{min} and CW_{max} is investigated for a fixed traffic mix consisting of delay sensitive voice sources and best-effort TCP transmissions. We used network simulation for the evaluation of the proposed mechanisms. For simulation, we use the ns-2 network simulator developed by the VINT Project [NS] with the wireless extension produced by the MONARCH Group [CMU].

The traffic mix we consider consists of 5 hosts sending high priority voice traffic and 10 mobile hosts starting best effort greedy TCP connections. Voice traffic was modeled using an on/off source with exponentially distributed on and off periods of 300ms average each. Traffic was generated during the on periods at a rate of 32kbps with a packet size of 160 bytes, thus the inter-packet time was 40ms. During all simulations the channel rate was 2 Mbps.

We ran a set of tests for this traffic mix with varying values of CW_{min} for both traffic classes. For high priority traffic, the CW_{min} values varied between [8,32], the CW_{min} for best effort traffic varied between [32,128]. A value of 32 is the proposed by the standard [IEEE802.11], which applies to the case when only best effort traffic class is supported. We chose this value to be the delimiter between the two traffic classes under test. The values chosen for high priority traffic range below this value. A value of 8 is proposed by the standard as an absolute minimum. By using the values above 32 for the best effort traffic class, the ranges do not overlap, and for all combinations it is assured that $CW_{min}^{highprio} \leq CW_{min}^{lowprio}$. Based on the intuitive discussion in the previous section, the maximum contention window for the high-priority class was lowered to $CW_{max}^{highprio} = 64$, while the upper limit for the low priority class was set to the recommended value of $CW_{max}^{lowprio} = 1024$.

In both intervals 5 values were chosen to cover each range of CW_{min} values. Simulations were performed for all 5x5 combinations covering the whole plane. Packet delays were logged for both high and low priority traffic classes.

Figure 5.3 shows the summary of the simulation results. The x-axis corresponds to the CW_{min} of the best-effort packets. It can be observed that increasing this value results in larger delays for best-effort traffic and somewhat decreasing delays for real-time traffic. The delay for real-time sources is more significantly affected by their CW_{min} values (see dashed lines), while the delays of best-effort packets are not affected greatly by the value chosen for real-time sources (straight lines). For all combinations (apart from the trivial case where $CW_{min}^{highprio} = CW_{min}^{lowprio} = 32$) the streams in different traffic classes experienced differentiated delay. The experiment supports the argument that the delay differentiation can be increased by increasing the gap between $CW_{min}^{lowprio}$ and $CW_{min}^{highprio}$, i.e., decreasing $CW_{min}^{highprio}$ and increasing $CW_{min}^{lowprio}$.

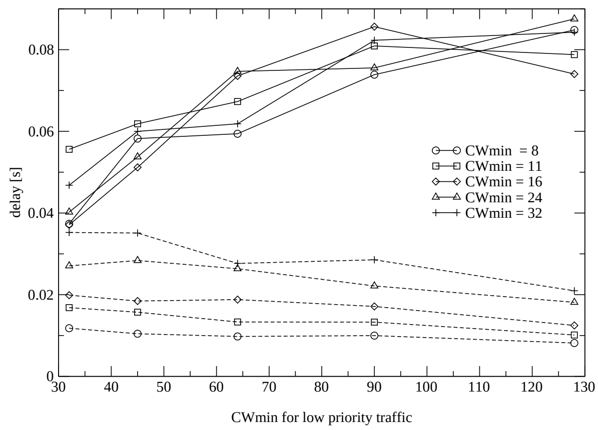


Figure 5.3: Average delay experienced by voice and TCP flows for varying values of CW_{min}. The x axis is the CW_{min} for best-effort traffic. Different symbols represent different CW_{min} for high-priority traffic. Channel rate 2 Mbps.

The previous test demonstrated that effective service separation is possible by appropriately adjusting the backoff times through the contention window limits. However, it is still an open question whether this separation can be effectively maintained across a wide range of traffic loads for moderate to high congestion. In the next test, the robustness of service separation is investigated by simulating increasing levels of traffic up to the level of channel saturation.

During simulation, the channel load is increased by adding a new voice, video (64kbps constant rate source) and TCP session periodically every 5 seconds. The voice and video sources use CW_{min} and CW_{max} values of 16 and 64, while the TCP traffic uses 128 and 1024, respectively. Figure 5.4 shows the delay throughout the simulation for the three traffic types. It can be observed that the delay increases for all service types but the delay separation is efficiently maintained from low load up until the channel is saturated.

For best-effort traffic the achievable throughput is of more importance than delay. Figure 5.5 shows that the modified MAC enables the best-effort adaptive TCP traffic to utilize any free capacity unused by high priority sources. It can be observed that even at the saturation point, the TCP traffic is not completely starved. This is due to the statistical and non-deterministic nature of service separation.

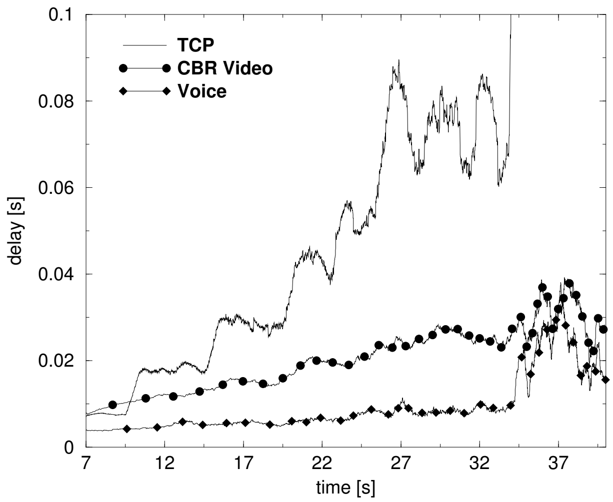


Figure 5.4: Average delay experienced by gradually increasing the number of TCP and real time sources over time. Channel rate 2 Mbps.

The modified MAC provides good service differentiation in terms of throughput and delay over a wide range of high priority and best effort traffic mixes. We investigate more dynamic traffic scenarios in Section 5.6.

5.3 Estimation of Available Resources using a Virtual MAC Algorithm

Many aspects of the wireless channel preclude exact control of resources (e.g., channel fading or interference). Furthermore, the lack of cell planning and shared resources in the access network

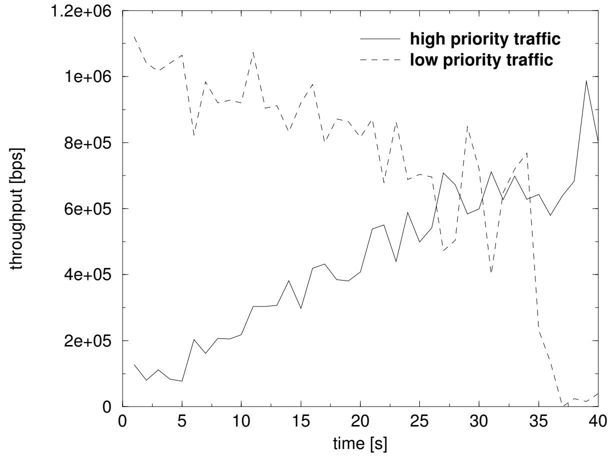


Figure 5.5: Aggregate throughput of high priority and best effort traffic classes: number of TCP and real time sources increase over time. Channel rate is 2 Mbps.

may result in densely packed base stations severely degrading the available capacity, as perceived by neighboring base stations. The MAC described in the previous section ensures effective service differentiation even in the case of overlapping cells and high traffic loads. However, to support real-time services it is not sufficient to ensure that high priority traffic gets better service than best effort, as in most cases, applications require absolute and not relative service quality, (e.g. for voice or video). If a mobile host realizes that the channel is not able to meet its delay and loss requirements, it can either refrain from loading the channel or reduce application traffic demands, (e.g., by increasing compression). In order to make this decision the host has to rely on accurate estimations of the achievable QoS of the radio channel.

The difficulty of this problem is that measuring simple channel properties, such as channel utilization is not sufficient to estimate the loss and delay statistics of a new session. The reason for this is that the actual QoS depends on a number of factors, (e.g., the actual arrival pattern of packets or the ratio of hidden terminals). The analytic models published in the literature usually focus on one of these aspects, and make a number of assumptions about the other aspects. Furthermore, the traffic models used are usually simplistic for real traffic scenarios, (e.g., assuming only long, greedy sessions). Even if the analytic models were more accurate, and could take into account the relevant modeling issues, parameterising them would be an extremely difficult task.

To overcome the problem of channel modeling we take a more pragmatic approach: instead of modeling the interaction of MAC, the radio channel and background traffic load, we introduce a Virtual MAC (VMAC), which emulates the behavior of the MAC performance. We argue that the algorithm is accurate, can be easily implemented, and scales to high data rates. To prove these claims, we implemented the virtual MAC algorithm on a mobile host accessing a 11 Mbps wireless LAN. The efficiency of the algorithm and the implementation are discussed in the section below.

5.3.1 Operation of the Virtual MAC Algorithm

The VMAC algorithm operates in parallel to the real MAC in the mobile host but the VMAC does not handle real packets; rather, it handles "virtual packets". Scheduling of these virtual packets on the radio channel is performed in the same way as for real packets, which means channel testing and random back-off is performed, as necessary. The difference arises when the Virtual MAC decides to send a virtual packet. Unlike the case of real packets, no packet is transmitted at this point. Rather, the Virtual MAC algorithm estimates the probability of collision if the virtual packet was "really" sent. To make the algorithm conservative, a collision is "detected" whenever any other mobile station chooses the same slot for transmission, (i.e., the channel is occupied by any station within the same slot time). In this case, the Virtual MAC enters a back-off procedure, as a real MAC would do after a collision had occurred.

For a real MAC, collision detection is realized using a timer, which expires if neither a CTS in response to an RTS nor an ACK in reply to a data packet arrives in time, depending on the operation. If no CTS or ACK has been received before this timer expires then the real MAC assumes that a collision has occurred and the packet must be retransmitted. At this point a real MAC would begin a backoff procedure. The Virtual MAC does not detect collisions in this manner. Rather, it decides that a collision would have happened if a transmission occurs in the timeslot determined by its congestion avoidance algorithm. In other words, the Virtual MAC detects "virtual collisions" immediately and not through using a timer. Thus, the VMAC enters the backoff procedure after a delay equal to that of an RTS timer in a real MAC.

If no collision occurs, the MAC delay is estimated by the total defer time accumulated plus the time to fully acknowledge the packet (e.g., if RTS/CTS is enabled it is $d = t_{defer} + t_{RTS} + t_{CTS} + t_{packet} + t_{ACK} + 3t_{SIFS} + 3\tau$ where τ is an estimate of the maximum propagation delay). An example of the operation of the Virtual MAC is illustrated in Figure 5.6.

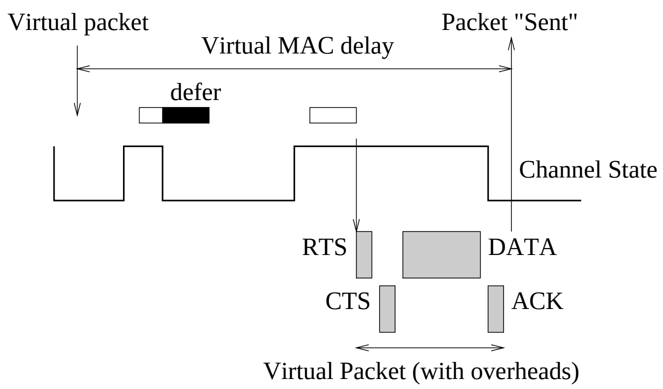


Figure 5.6: An example of the operation of the Virtual MAC algorithm. The channel state indicates an idle (state is high) or busy (low) channel. A virtual packet arrives during a busy period and the deferred timer is decremented during a short idle period, and virtual transmission happens during the next idle period, when the deferred timer expires.

The virtual MAC emulates not only backoff and collision resolution aspects of the real MAC but all other aspects of a real MAC are also emulated. For example, packet loss is signaled by the VMAC if the maximum number of "retries" is reached. It also adjusts the contention window by doubling the window until it reaches CW_{max}, when it stops increasing it further. If a transmission is successful then the contention window is reset.

After every successful or failed "transmission", the VMAC waits for the next virtual packet to process. If, for example, the packets arrive at the VMAC at a rate of 20 ms and with size of 80 bytes, the output of the VMAC algorithm will closely match the delays experienced by a real constant rate encoded voice application.

One of the key advantages of the VMAC algorithm over analytic models is that it does not produce just a small set of performance measures, (i.e., estimates of first order statistics). Rather, it produces a time series that can be identically analyzed to a time series produced by a real test. Consequently, there is no limit on using higher order statistics, which makes it possible to apply more sophisticated analyses and traffic control methods. For example, not only the nth moments of the delay can be calculated but also percentiles, burstiness, traffic envelopes, number of errored seconds, etc., which are more closely related to user perceived quality measures.

The VMAC can be applied to estimate the performance of either best effort or better than best effort traffic by changing the MAC mechanism to match the changes discussed for service differentiation in Section 5.2. These estimates can be used for a variety of traffic control algorithms. In the proposed architecture, we use the VMAC algorithm to estimate the QoS of better than best effort traffic, and base the admission decision on that estimate.

5.3.2 Evaluation of the Virtual MAC Algorithm

Fig 5.7 shows results from a simulation test of the efficiency of the Virtual MAC algorithm. The figure shows the simulated and the VMAC estimated delays experienced by a new real-time voice source for an increasing number of homogeneous voice sources. The estimation is precise over the whole range of traffic loads, most importantly in the saturation region. Thus, it is suitable for evaluating the admissible capacity of the channel for real time traffic.

Figure 5.8 shows the results for a more complex simulation test where voice traffic is mixed with an increasing number of "Web sources". The Web sources are modeled by short TCP file transfers where the file sizes are drawn from a Pareto distribution with mean file size of 10kbps and shape parameter 1.2. The length of the silent period between two downloads is also Pareto with the same shape parameter and mean delay of 10s. This creates a highly bursty background data traffic load with multiple time-scale fluctuations [LTWW93] [CrBe96] [TWS97]. The TCP load is sufficient to saturate the channel by itself.

The figure shows two scenarios. In the first, the voice source is not prioritized over the data sources. In the second, the MAC algorithm is modified, as discussed in the previous section for the voice source. The results show that the VMAC algorithm efficiently estimates the delay. In both cases, the estimation is conservative and the mean delay is about 1-2 ms greater than the result obtained by simulation. Another important observation is that priority for voice provides significantly smaller and smoother delay and delay variation values in the case of highly bursty data traffic. Without modifying the MAC for voice, the voice packets have to compete with data packets, which, since the data traffic is much burstier, does not only increase the voice packets' delay but also increases the delay variance, as shown in Figure 5.8.

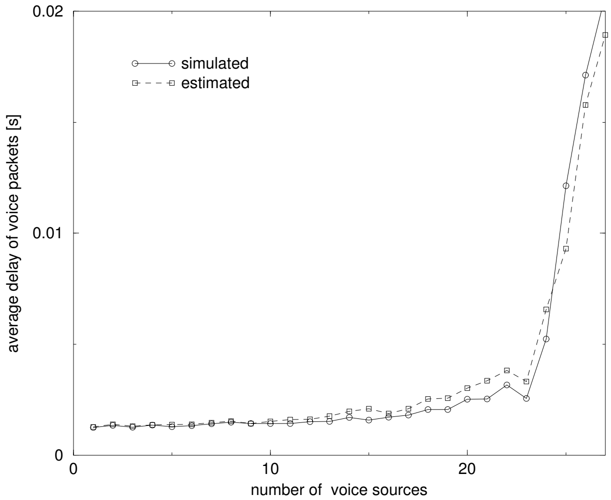


Figure 5.7: Virtual and simulated average MAC delay of a new voice source versus the number of active voice sources.

5.4 Implementation of the Virtual MAC Algorithm in a Wireless Testbed

Since the wireless DiffServ MAC can only offer soft and relative differentiation, it is important that the mobile hosts have accurate estimations of the channel. There are several issues concerning the VMAC that can only be satisfactorily evaluated in a real wireless network with real applications. In what follows, we describe a wireless differentiated services testbed and its VMAC implementation. In addition, we compare the estimates given by the VMAC and the performance perceived by real applications.

The VMAC was implemented on a Linux machine with a modification to the wireless card's device driver. We used 11 Mbps Lucent and Aironet PCMCIA cards in the experiments. These cards, with the modified drivers, are capable of capturing all "overheard" layer two transmissions,(e.g., CTS, RTS, ACK packets, even with CRC errors). Packets were timestamped with approximately microsecond precision. This traffic trace was used as input for the VMAC algorithm. In a commercial implementation, the VMAC could be placed into the firmware of the wireless card and would operate in real-time.

The testbed generates traffic mixes of TCP and UDP flows, with different levels of offered load, as illustrated in Figure 5.9. The wireless testbed consists of 6 hosts with 11 Mbps IEEE 802.11 PCMCIA cards. All mobile hosts were configured to operate in DCF ad-hoc mode. Three of the mobile hosts (indicated as TCP hosts) were used to generate random TCP traffic. The TCP hosts transferred random length files independently of each other using TCP. The average file size was 50 kbytes. Between file-transfers each host waited a random duration before the next transfer was started. Adjusting the average idle time modified the load on the channel. The UDP host generates packets every 20 ms at a data rate of 32 kbps, resulting in a voice-like traffic

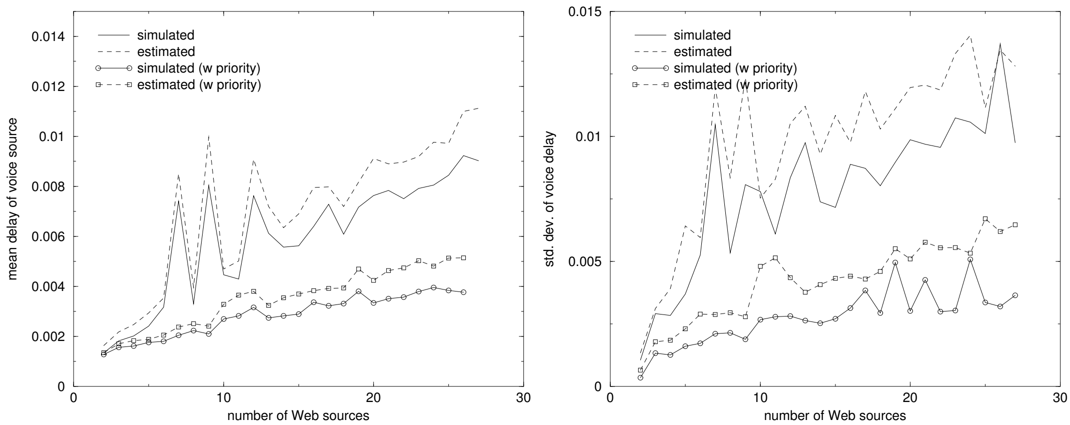


Figure 5.8: Average delay (a) and delay variation (b) of a new voice source obtained by simulation and from the virtual MAC algorithm, versus number of Web sources with and without priority for voice traffic. Channel rate is 2 Mbps.

stream.

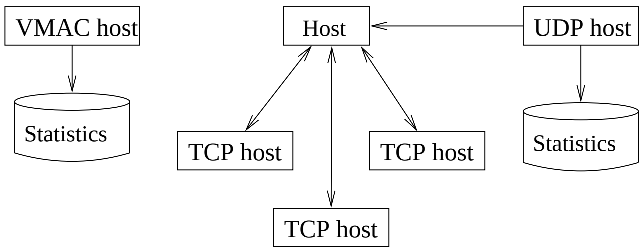


Figure 5.9: Testbed configuration.

Because the cards do not support APIs to change the contention window limits, all sources use the same backoff algorithm, using factory set default values. Therefore, we were not able to evaluate the previously proposed DiffServ MAC but we could still evaluate the accuracy of the VMAC algorithm.

The UDP host logged the delays of the wireless MAC. This was achieved by modifying the wireless card network driver to capture all packet processing events together with an accurate timestamp at a resolution of approximately one microsecond. The resulting log file consists of packet arrivals to the MAC, packet sizes, MAC deferred delays and indications of successful or unsuccessful delivery.

The fifth host acted as a traffic monitor and executed the VMAC algorithm (indicated in the figure as the VMAC host). The VMAC host logged a similar file as the UDP host but this file consisted of estimated delays provided by the VMAC algorithm.

During the experiment the channel utilization gradually increased up to its saturation point by decreasing the average idle time from 10 to 0 sec. Figure 5.10(a) shows the physical level channel utilization vs. time. The maximum channel utilization reached was approximately 70%.

The measured UDP delay statistics and the estimated delay statistics from the VMAC algorithm are shown in Figure 5.10(b). It can be observed that the VMAC implementation could estimate the measured delay with excellent precision during the entire experiment for all channel loads. Thus, mobile hosts running passive monitors and VMACs can rely on precise quality feedback for traffic control purposes.

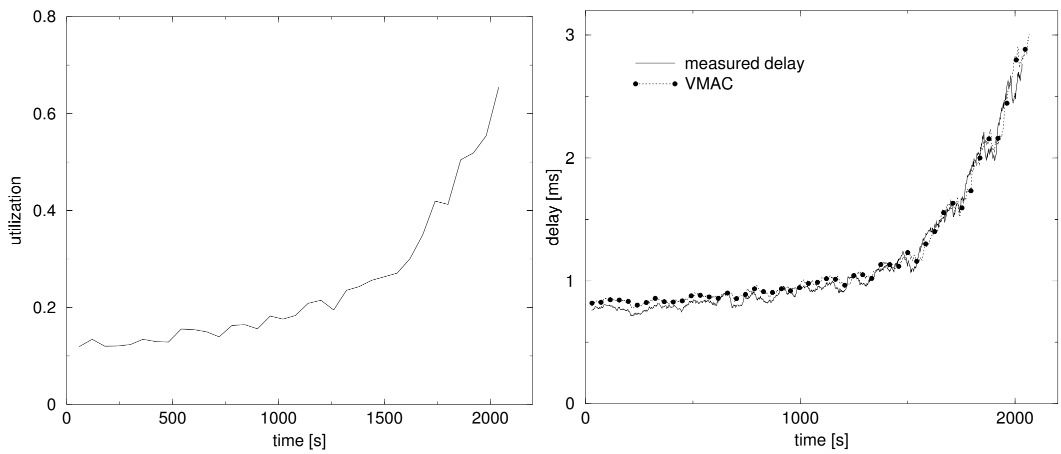


Figure 5.10: a) Measured channel utilization with increasing TCP traffic. b) Average delay of UDP traffic. Channel rate 11 Mbps.

5.5 Estimation of Application Level QoS using a Virtual Source Algorithm

The VMAC measures virtual packet delays, packet losses and collisions at the MAC level. The delay experienced by an application can be very different than the delay provided by the MAC. The reason for this is that application level data usually has to be packetized, encoded and placed into an interface queue before the MAC layer receives it.

Also, even the estimated MAC layer delay depends not only on the channel but also the arrival pattern of packets at the MAC. This effect can be due to the correlation structure of the traffic load on the channel. These factors are taken into consideration by the Virtual Source (VS) algorithm. For certain applications, running the VS can provide more precise estimates of the achievable performance. In addition, the VS makes it possible to tune certain application level QOS parameters.

The VS algorithm consists of a Virtual Application, interface queue and Virtual MAC. The Virtual Application generates virtual packets, as a real application would do (e.g., generating virtual voice packets at a constant rate). Packets are time-stamped and placed in a virtual buffer.

When the virtual packet is finished processing in the VMAC, the total delay is calculated comparing the actual time to the timestamp stored in the packet.

Although the VS gives a more useful estimation for an application, the VS is not as generally applicable as the VMAC, since it requires that the application traffic is well known in advance. Nevertheless, we believe that there are a number of important applications that fit into this category such as constant bit rate encoded voice or video. If the application traffic is not easy to emulate then traffic management falls back to the estimations provided by the VMAC, which is independent of application type.

5.5.1 Virtual Delay Curves

The application delay depends on several factors. Certain factors depend on the application (e.g., packet size, packet rate), others depend on the load of the channel. The VS algorithm monitors the channel continuously and estimates the application performance taking into account these factors. Thus, the VS algorithm can be used to find the optimal parameters for the best application performance. Intuitively, at the same data bit rate, the application delay can be reduced by increasing the packet rate, since it reduces the packetization delay. In contrast, higher packet rates load the radio channel more. Higher rates cause more collisions, increasing the average contention window. This eventually leads to larger MAC delays. In addition, higher packet rates mean smaller data packets, which results in larger protocol overhead, (i.e., larger load on the radio channel). Thus, even at the same application bit rate, there is a tradeoff between packetization delay and MAC delay.

Denote the function $d(p_{rate})$ as the virtual delay curve of an application, where p_{rate} is the packet inter-arrival time of the application, e.g., $p_{rate} = 0.02$ packets per second for voice, but the data bit rate is kept constant, i.e., $p_{size} * p_{rate} = const$ (where p_{size} is the size of the application level packet). The virtual delay curve at p_{rate} gives the average delay of virtual packets if the VS algorithm generates packets at the rate of p_{rate}. The mobile host or the base station runs VS algorithms with several p_{rate} values in parallel. Delay curve can be constructed from the virtual packet delays obtained from the VS algorithm. Similarly, we can define the virtual delay variance curve $v(p_{rate})$ which calculates the virtual delay variances, respectively. Based on the delay curve, a mobile host or base station can choose the optimal packet rate and packet size so that an application experiences minimum delay and delay variance.

Figure 5.11 shows the virtual delay and variance curves for a virtual voice source at several background traffic loads. It can be observed that in the case of low background traffic ($N = 20$ "Web" sessions) the delay curve increases monotonically, which means that the best end-to-end delay can be achieved if the packet rate is high and the application sends small packets. The estimation of delay variance appears to be constant. As the background load increases, the MAC delay increases, and the optimum is not at the highest rate but at about 20 ms. The delay variance also decreases as the inter-packet times increase.

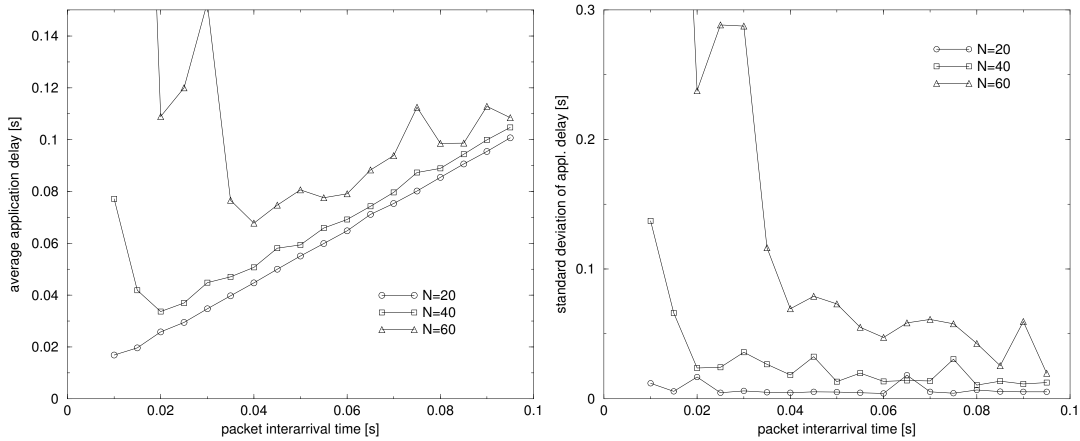


Figure 5.11: Virtual delay and delay variance curves at several radio channel loads, N denotes the number of background Web sessions. Web traffic uses $CW_{min} = 31$ and the virtual algorithm uses $CW_{min} = 8$. Channel rate 2 Mbps.

5.6 Distributed Admission Control Algorithm in a Multicell Environment

A mobile host can use the VS and the VMAC estimates before actually starting transmission. Because the virtual algorithms do not require high processing capacity and do not load the channel, they may run continuously and not only when a service request arrives. In other words, the virtual algorithms are designed to continuously keep track of the health of the channel.

This estimate can be used to apply traffic control to maintain the congestion of the channel at a low level and the relative performance guarantees provided by the DiffServ MAC at absolute levels. There are numerous ways to utilize these estimates from the VMAC and VS algorithms. For example, elastic, best-effort traffic can be policed or shaped in response to estimation of congestion. Premium, delay sensitive sessions are usually not elastic, thus admission control is more appropriate to control them. In this section, we apply the latter type of traffic control. However, we note that adding some sort of control for best effort traffic may further improve the quality assurances. Every mobile host keeps track of the state of the channel using either VMAC or VS. The admission control algorithm compares the results of the VS and VMAC with the service requirements and admits or rejects a new session accordingly. For admission we only use the average delay estimation over the last few seconds. The admission algorithm runs in every mobile host and is performed in a fully distributed and autonomous manner.

Because the radio channel properties may be different at the receiving and transmitting mobile hosts, it is preferable that both hosts execute the VS and VMAC algorithms to ensure that the service quality will be met for a new session. This can be executed during session setup. Admission is granted if both virtual algorithms at the mobile hosts admit the new request.

In this section, we investigate this concept through simulation of a complex configuration with random topology and random traffic. The aim is to test how the modified MAC and VMAC algorithm perform in the presence of a highly dynamic real-time and non real-time traffic mix when the radio channel is dynamically shared among traffic streams between mobile hosts and base stations.

Ten base stations were placed randomly on a 400m by 400m rectangular area with their coverage areas significantly overlapping. One hundred mobile hosts were placed randomly in the coverage area. The mobile hosts were stationary during the test. Every mobile host was associated with the nearest base station. Half of the mobile hosts randomly generated Web sessions and the other half randomly generated voice traffic. The length of the voice sessions and the inter-arrival times between connection requests were exponentially distributed. The average session length was 30s. Upon completion of a session, a mobile host attempted a new call after an average waiting period of 10s.

Independent Virtual Source algorithms running in all base stations continuously monitored the radio channel. Admission control was applied to delay sensitive voice sessions. When the estimated delay exceeds 10 ms, new voice sessions were rejected from service. If accepted, the voice packets use the modified MAC algorithm with $cw_{min} = 32$ slots and $cw_{max} = 64$ slots, while the Web sessions use values 64 and 1024, respectively. There was no admission control applied to Web traffic.

Figure 5.12a shows the total TCP and voice traffic rates in the entire coverage. After an initial startup, the aggregate voice rate settles around a stable throughput, while the TCP traffic shows high levels of burstiness.

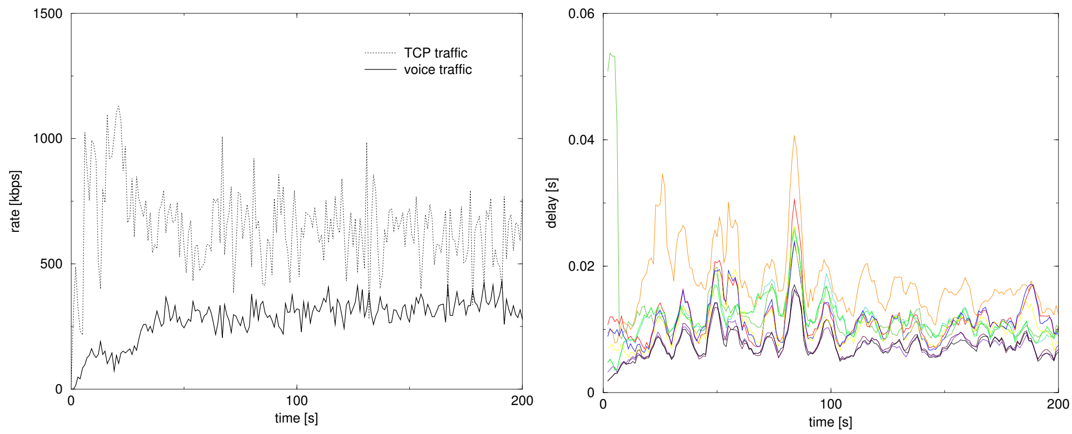


Figure 5.12: a) Aggregate rates of TCP and voice traffic in the entire service area. b) Estimated delays by VS algorithms running at base stations.

Figure 5.12b shows the delay estimations by the VS algorithms running in base stations. It can be observed that the delay estimation is kept below the admission target most of the time for most base stations. However, the estimated delay is significantly different at a few base stations, where, the estimated delay reaches 10 ms for long durations. These base stations did not accept voice traffic during these periods. On the other hand, other base stations were continuously in the accept state. This was due to the overlapping of cells and the shared radio channel.

Figure 5.13 shows the empirical distribution of voice packet delays from accepted sessions. The low delays experienced indicate that the overall channel state is efficiently controlled by the distributed monitoring and admission control algorithm, even in the presence of highly dynamic TCP traffic.

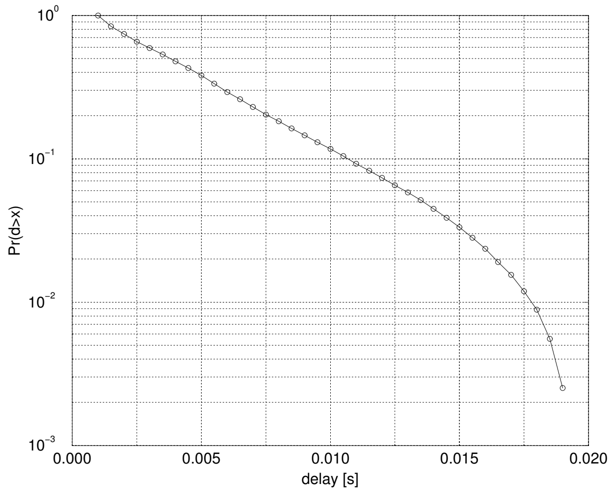


Figure 5.13: Delay distribution of voice packets.

5.7 Conclusions

This chapter has shown how service differentiation can be provided in a mobile access network in a fully distributed manner with minimal control. By manipulating the contention window limits of the IEEE 802.11 DCF mode it is possible to provide service differentiation at the radio MAC layer. The proposed MAC provides good delay and throughput separation for best effort and high priority traffic for a range of traffic mixes and channel loads.

We have proposed two passive radio channel monitoring algorithms. By emulating MAC (Virtual MAC) and application (Virtual Source) mechanisms, these algorithms can estimate the achievable level of service without actually loading the channel. We evaluated the efficiency of the Virtual MAC algorithm using simulation and implementation in an experimental differentiated services wireless testbed. The notion of virtual delay curves has been introduced in relation to the virtual algorithms. Delay curves enable an application to tune its traffic parameters to match the dynamic characteristics of the radio channel in an efficient manner.

We have demonstrated through simulation that thee modified MAC together with distributed admission control algorithm can maintain a globally stable state in a micro-cellular environment even if cell areas overlap and the radio channel is shared.

Recently, there is a new proposed standard by the IEEE called 802.11e [MCMK02], which introduces a modification to the DCF mode called Enhanced DCF (EDCF). In EDCF, packets in lower priority must wait a longer waiting time before they can transmit on the channel. Our work predated and influenced the new proposed standard, which is very similar to our proposal.

Chapter 6

Summary

The objective of this thesis is to discuss, analyze, and improve several aspects of TCP/IP networks in the context of performance modeling and resource management in wired and wireless Differentiated Services networks.

The central argument of this thesis is that to develop precise performance models, we have to understand the interaction between network, application and end-host protocols and mechanisms. Only by collectively studying these issues in the context of an Internet extended by wireless services can we fully grasp the great challenges facing the evolving nature of the global Internet infrastructure.

In the first part of this thesis, we argued that the modeling of TCP/IP networks requires new analytical tools and modeling techniques, because, as we discussed, some of the major assumptions of conventional traffic theory cannot be applied to the Internet. We argued that the reason is because network mechanisms and end-host protocols are strongly coupled, mainly due to the end-to-end congestion control used by the TCP protocol. In the second part of the thesis we developed DiffServ performance management methods based on the insights from Chapters 2 and 3 for wired and wireless networks. For wired DiffServ networks we developed resource estimation methods based on the effective bandwidth concept in Chapter 4. In Chapter 5 we introduced a complete DiffServ solution 'suite' using exclusively distributed algorithms for wireless IP networks.

In what follows, we summarize the main contributions of our work.

6.1 Chaotic Modeling of TCP Congestion Control

Chapter 2 analyzes the dynamics of competing TCP connections in bottleneck buffers. Through simulations and network measurements, we demonstrated that TCP competition for network resources could be modeled as a chaotic system. We demonstrated the major properties of chaotic systems in TCP. We applied analytical methods developed in chaos theory to quantify the properties of TCP as a chaotic system. In particular, we measured the fractal attractor dimension and the Lyapunov exponent, which is a measure of the sensitivity to initial conditions. We demonstrated that TCP competition could generate self-similar traffic, contributing to traffic self-similarity observed in the Internet.

The chaotic modeling technique allows us to understand TCP dynamics in a unified modeling framework by explaining previously separately modeled phenomena, such as phase effects, synchronization and apparent randomness. We also demonstrated how we could use the tools developed in chaos related research in other fields of science, for example, meteorology, biology and physics.

This work represents the first analysis and modeling of TCP as a chaotic system.

A possible future direction of research is to make use of the sensitivity property of the chaotic system, and develop chaos control methods as buffer management to improve network performance using minute interactions.

6.2 TCP's Role in the Propagation of Self-similarity in the Internet

Chapter 3 analyzed the adaptation property of TCP congestion control. Through a number of wide area Internet measurements we showed that TCP propagates self-similarity encountered on its path to other parts of the network where it self-similarity would not arise otherwise. We presented a simple analytic model to support our arguments, which model approximates TCP congestion control as a linear system. Linearity was demonstrated by simulations, showing that TCP propagates the correlation structure of any stochastic background process above a characteristic timescale. The characteristic timescale was approximated by an analytic model. It was shown that TCP inherits self-similarity when it is mixed with self-similar background traffic in a bottleneck buffer through the transform function of the linear system. This property was demonstrated for both short and long duration TCP connections.

The proposed mechanisms are basic "building blocks" in a future wide-area traffic model, and in real-life it is always their combined effect that we can observe. The network measurements that we presented are intended to highlight the basic mechanisms in simplified network scenarios, when it can be assured that only the network conditions and TCP's response to network conditions are the cause of the investigated phenomena.

The work presented in Chapter 3 shows for the first time that TCP propagates self-similarity.

Chapters 2 and 3 analyzed two aspects of TCCP congestion control. The two aspects are strongly related since they are two sides of essentially the same mechanism. The adaptation property can propagate packet dynamics created by chaotic competition, but it is still an open question how to model propagation and competition in the same modeling framework.

6.3 Resource Management for Differentiated Services Networks

Chapter 4 introduced a set of resource estimation methods for wired Differentiated Services networks. We derived several analytic formulae for the estimation of required resources to provision loss and delay sensitive service classes. The properties of a DiffServ framework were taken into account, namely, aggregate traffic handling, simple scheduling and traffic conditioning at the edges. The models behind the proposed methods are based on the assumptions of the statistical properties of TCP/IP traffic flows analyzed in Chapters 2 and 3.

The proposed methods can be used for either admission control or network dimensioning.

We built on previous research results, and analyzed their weaknesses and strengths from the perspective of practical application in real networks.

A possible future direction of the research discussed in this chapter is to analyze the robustness of these methods in a real network or a realistic testbed. We have analyzed a simple DiffServ router implementation using SPQ schedulers. We have started research to investigate how to apply the theorems for other types of class based scheduling methods, for example, Weighted Fair Queuing (WFQ) or Weighted Round Robin (WRR).

6.4 Providing Differentiated Services in Wireless Packet Networks

We developed a set of distributed algorithms to offer service differentiation in wireless packet networks. The methods are general to be used in a wide range of wireless technologies, however, to be able to demonstrate the feasibility of the framework, we have applied them as an extension of the IEEE 802.11 wireless LAN standard.

The contribution of our research is that the proposed solution is fully distributed and robust against traffic fluctuations, host mobility or uncontrolled interference. We have demonstrated through simulation that the modified MAC together with a distributed admission control algorithm can maintain a globally stable state in a micro-cellular environment even if cell areas overlap and the radio channel is shared.

We have implemented passive radio channel monitoring and the Virtual MAC algorithm in a IEEE 802.11 testbed. Our future research is to integrate the DiffServ solution with Cellular IP [Val99], which provides a solution for mobility management. We also plan to implement the full suite of algorithms in a testbed, which will include the modified MAC and virtual control algorithms and will provide support for service level agreements with fast handoff.

The work presented in this chapter presents the first attempt to engineer service differentiation in wireless IP networks based on IEEE 802.11.

Appendix A

Proofs for Effective Bandwidth Bounds

A.1 Finding an Approximate Value for s

To obtain an approximation of the optimal s the same derivation can be used for BW^A and BW^G. For BW^A consider all flows as one group.

$$BW^G(s) = \frac{1}{s}\sum_{i=1}^{G} n_i \ln\left(\frac{M_i + \sum_{k\in A_i}\frac{h_k}{e^{sh_k}-1}}{n_i}\right) - \frac{1}{s}\sum_{k=1}^{N}\ln\left(\frac{h_k}{e^{sh_k}-1}\right) + \frac{\gamma}{s}. \tag{A.1}$$

The objective is to approximate this equation with a closed form solution for the optimal s. The key idea is to arrive to a polynomial of BW in s, that can be optimized in a closed form.

Step 1. In the first step, we approximate the first sum:

$$F_1 = \frac{1}{s}\sum_{i=1}^{G} n_i \ln\left(\frac{M_i + \sum_{k\in A_i}\frac{h_k}{e^{sh_k}-1}}{n_i}\right) \tag{A.2}$$

For $h_k/(e^{sh_k}-1)$ we take the series expansion with respect to variable s about $s=0$ up to order two.

$$\frac{h_k}{e^{sh_k}-1} = \frac{1}{s} - \frac{1}{2}h_k + \frac{1}{12}h_k^2 s - O(s^3) \tag{A.3}$$

Substituting the first 3 elements to the first sum we get

$$F_1 \approx \frac{1}{s}\sum_{i=1}^{G} n_i \ln\left(\frac{M_i}{n_i} + \frac{1}{s} - \frac{1}{2n_i}H_i + \frac{1}{12n_i}\widehat{H}_i s\right) \tag{A.4}$$

using the notation $H_i = \sum_{k\in A_i} h_k$ and $\widehat{H}_i = \sum_{k\in A_i} h_k^2$. Taking the series of the logarithm this time about $s=0$ up to order two, we get:

$$\ln\left(\frac{M_i}{n_i} + \frac{1}{s} - \frac{1}{2n_i}H_i + \frac{1}{12n_i}\widehat{H}_i s\right) \approx -\ln(s) + \frac{2M_i - H_i}{2n_i}s + \left(\frac{\widehat{H}_i}{12n_i} - \frac{(2M_i - H_i)^2}{8n_i^2}\right)s^2 + O(s^3) \tag{A.5}$$

Place the approximation in F_1:

$$F_1 \approx \frac{1}{s}\left(-N\ln(s) + \left(M - \frac{H}{2}\right)s + \left(\frac{\widehat{H}}{12} - \sum_{i=1}^{G}\frac{(2M_i - H_i)^2}{8n_i}\right)s^2\right) \qquad \text{(A.6)}$$

where $M = \sum_{i=1}^{N} M_i$, $H = \sum_{i=1}^{N} H_i$ and $\widehat{H} = \sum_{i=1}^{N} \widehat{H}_i$.

Step 2. In step 2, we approximate the second sum:

$$F_2 = \frac{1}{s}\sum_{k=1}^{N}\ln\left(\frac{h_k}{e^{sh_k}-1}\right) \qquad \text{(A.7)}$$

Take the series expansion of the logarithm with respect to variable s about $s = 0$ up to order two:

$$\ln\left(\frac{h_k}{e^{sh_k}-1}\right) \approx -\ln(s) - \frac{h_k}{2}s - \frac{h_k^2}{24}s^2 + O(s^3) \qquad \text{(A.8)}$$

With this approximation:

$$F_2 \approx \frac{1}{s}\left(-N\ln(s) - \frac{H}{2}s - \frac{\widehat{H}}{24}s^2\right) \qquad \text{(A.9)}$$

Step 3. Substituting the two approximations $F1$ and $F2$ back to BW^G we get:

$$BW^G(s) \approx M + \left(\frac{\widehat{H}}{8} - \sum_{i=1}^{G}\frac{(2M_i - H_i)^2}{8n_i}\right)s + \frac{\gamma}{s}. \qquad \text{(A.10)}$$

Step 4. Our objective was fulfilled, the resulting approximation is second order with respect to s. The single minimum is at

$$s_{opt} \approx \sqrt{\frac{8\gamma}{\widehat{H} - \sum_{i=1}^{G}\frac{(2M_i - H_i)^2}{n_i}}}$$

A.2 Effective Bandwidth for Delay Sensitive Classes

Proof of Lemma 1 We want to prove that the buffer occupancy curve is related to the effective load:

$$\Delta O(t) = B(t) - Ct \qquad \text{(A.11)}$$

During a busy period, after t seconds from the beginning of the busy period, the buffer occupancy equals the total traffic entering the queue minus the amount of traffic serviced from the queue:

$$Q(t) = \sum_{k=1}^{N} X_k[t] - Ct \qquad \text{(A.12)}$$

If we insert (A.12) into Definition 3 we get

$$\Pr\left(\sum_{k=1}^{N} X_k[t] - Ct \geq \Delta O(t)\right) \leq \epsilon \tag{A.13}$$

Now apply Definition 2 about the effective load. This concludes the proof. ■

A.3 Delay Bounds for Multiple Queues

Proof of Lemma 2 The amount of service unused by all higher priority queues $l < k$ during time interval of length t is:

$$U_k(t) = \left(Ct - \sum_{i \in \mathcal{X}_{1\ldots k-1}} X_i(t)\right)^+ \tag{A.14}$$

Thus t seconds after the beginning of a busy period of queue k, the buffer occupancy is

$$Q_k(t) = \sum_{i \in \mathcal{X}_k} X_i(t) - U_k(t) \leq \sum_{i \in \mathcal{X}_{1\ldots k}} X_i(t) - Ct \tag{A.15}$$

Applying the effective load bound for the set $\mathcal{X}_{1\ldots k}$ with probability ϵ_k:

$$\Pr\left(\sum_{i \in \mathcal{X}_{1\ldots k}} X_i(t) - Ct \geq B^{\epsilon_k}[\mathcal{X}_{1\ldots k}](t) - Ct\right) \leq \epsilon_k \tag{A.16}$$

This concludes the proof of Lemma 2. ■

Proof of Lemma 3 The amount of unused service(A.14), is

$$U_k(t) \geq Ct - \sum_{i \in \mathcal{X}_{1\ldots k-1}} X_i(t) \tag{A.17}$$

From the definition of the effective load of set $\mathcal{X}_{1\ldots k-1}$:

$$\Pr\left(Ct - \sum_{i \in \mathcal{X}_{1\ldots k-1}} X_i(t) \leq Ct - B^{\epsilon_k}[\mathcal{X}_{1\ldots k-1}](t)\right) \leq \epsilon_k \tag{A.18}$$

which concludes the proof. ■

Proof of Lemma 3 The probability of $\Pr(X \leq Y)$ can be expressed as

$$\Pr(X \leq Y) = \int_0^\infty \Pr(X = x) \Pr(Y \geq x) dx \tag{A.19}$$

$$\Pr(X \leq Y) = \int_0^c \Pr(X = x) \Pr(Y \geq x) dx + \int_c^\infty \Pr(X = x) \Pr(Y \geq x) dx \tag{A.20}$$

If $\Pr(Y \leq c) \leq \epsilon$, the first term

$$\int_0^c \Pr(X = x) \Pr(Y \geq x) dx \leq \epsilon \int_0^c \Pr(X = x) dx \leq \epsilon \tag{A.21}$$

and, if $\Pr(X \geq c) \leq \epsilon$, the second term

$$\int_c^\infty \Pr(X = x) \Pr(Y \geq x) dx \leq \int_c^\infty \Pr(X = x) dx \leq \epsilon \tag{A.22}$$

This concludes the proof. ■

Bibliography

[AbVe98] P. Abry and D. Veitch, “Wavelet Analysis of Long-Range-Dependent Traffic”, *IEEE/ACM Transactions on Networking*, vol. 44, No.1., January 1998.

[ArKa99] A. Arvidsson and P. Karlsson, “On Traffic Models for TCP/IP”, *ITC-16*, Edinburgh, UK, June 1999

[AMMP99] I. F. Akyldiz, J. McNair, L. C. Martorell, R. Puigjaner, Y. Yesha, “Medium Access Control Protocols for Multimedia Traffic in Wireless Networks”, IEEE Network, vol 13 No 4, pp. 39-47, July 1999

[ABLM98] D. Avnir, O. Beham, D. Lidar, and O. Malcai, “Is the geometry of nature fractal?”, Science, 279:39. 1998.

[BBCD98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services”, *RFC 2475*, December 1998

[BCS94] R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, *RFC 1633*, June 1994.

[BDSZ94] V. Bharghavan, A. Demers, S. Shenker, L. Zhang, “MACAW: A Media Access Protocol for Wireless LANs”, *Proceedings of ACM SIGCOMM*, London, UK, Sept. 1994

[Ber99] Y. Bernet *et al*, “A Conceptual Model for Diffserv Routers”, *Internet draft work in progress* Jun. 1999

[BFO96] G. Bianchi, L. Fratta, M. Oliveri, “Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs”, *Proceedings of PIMRC*, Taipei, Taiwan, Oct. 1996

[Bhar98] V. Bharghavan, “Performance Analysis of a Medium Access Protocol for Wireless Packet Networks”, *IEEE Performance and Dependability Symposium*, Raleigh, NC, Aug. 1998

[BLD95] Y. Braiman, J. F. Lindner and W. L. Ditto, “Taming Spatiotemporal Chaos with Disorder”, *Nature*, Vol 378, 30. Nov. 1995

[Bra97] B. Braden B. Ed., *et. al.*, “Resource Reservation Protocol (RSVP) - Version 1 Functional Specification”, RFC 2205, Sept. 1997

[BrEs00] L. Breslau, D. Estrin, et al, “Advances in Network Simulation”, *IEEE Computer Magazine*, May 2000

[BrSi99] F. Brichet, A. Simonian, ”Conservative models for measurement-based admission control”, *ITC 16*, Edinburgh, June 1999

[BRSV96] F. Brichet, J. Roberts, A. Simonian, and D. Veitch, “Heavy traffic analysis of a storage model with long range dependent on/off sources”, *Queuing Systems*, 23:197–215, 1996.

[BSTW95] J. Beran, R. Sherman, M. S. Taqqu, and W. Willinger, “Long-range dependence in variable-bit-rate video traffic” *IEEE Transactions on Communications*, 43(2/3/4):1566–1579, Febuary/March/April 1995.

[CCG98] F. Cali, M. Conti, E. Gregori, “IEEE 802.11 Wireless LAN: Capacity Analysis and Protocol Enhancement”, *Proceedings of IEEE INFOCOM*, Apr. 1998.

[ChLe99] C. C. Chow, V. C. M. Leung, “Performance of IEEE 802.11 Medium Access Control Protocol over a Wireless Local Area Network with Distributed Radio Bridges”, *Proceedings of WCNC*, New Orleans, LA, Sep. 1999

[CKT96] C. Casetti, J. Kurose, D. Towsley, “A New Algorithm for Measurement-based Admission Control in Integrated Services Packet Networks”, *Protocols for High Speed Networks’96*, INRIA, Sophia Antipolis, Oct. 1996

[Cla88] D. Clark, “The Design Philosophy of the DARPA Internet Protocols”, *In Proc. SIGCOMM*, 106-114, Palo Alto, CA, Sept 1988.

[CLTR97] S.Crosby, I.Leslie, J.T.Lewis, R.Russell, F.Toomey and B.McGurk, “Statistical Properties of a Near-Optimal Measurement-based CAC algorithm”, *IEEE ATM’97*, Lisbon, June 1997

[CMU] CMU MONARCH Project, *www.monarch.cs.cmu.edu*

[CrBe96] M. E. Crovella and A. Bestavros, “Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes”, *in Proc. ACM SIGMETRICS*, pp. 160-169, May 1996.

[CTB96] M. E. Crovella, M. S. Taqqu, and A. Bestavros, “Heavy-tailed probability distributions in the world wide web” *Preprint: A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, 1996.

[DCBB02] B. Davie, A. Charny, J.C.R. Bennett, K. Benson et al, “An Expedited Forwarding PHB (Per-Hop Behavior)”, *RFC 3246*, March 2002

[DiMu95] W. Ditto and T. Munakata, “Principles and Applications of Chaotic Systems”,*Comm. of the ACM*, Nov. 1995/Vol. 38, No.11

[ENW96] A. Erramilli, O. Narayan, and W. Willinger, “Experimental queuing analysis with long-range dependent packet traffic”, *IEEE/ACM Transactions on Networking*, 4(2):209–223, April 1996.

[ErSi90] A. Erramilli and R. P. Singh, ”Application of Deterministic Chaotic Maps to Model Packet Traffic in Broadband Networks”, *Proc. 7th ITC Specialist Seminar*, Morristown, NJ, 8.1.1-8.1.3, 1990

[ESP94] A. Erramilli, E. P. Singh and P. Pruthi, “Chaotic Maps As Models of Packet Traffic”, *Proc. of the 14th ITC*, 329-338, June, 1994

[GPRS] ETSI, “Digital Cellular Telecommunications System General Packet Radio Services: Service Description EN 301 344 v6.3.2”, July 1999

[FGHW99] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger, “Dynamics of IP traffic: A study of the role of variability and the impact of control”, *In Proceedings of SIGCOMM*, Cambridge, MA, USA, August 1999.

[FGLR00] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, “NetScope: Traffic engineering for IP networks”, *IEEE Network Magazine, special issue on Internet traffic engineering*, March/April 2000.

[FGLR00-2] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, “Deriving Traffic Demands for Operational IP Networks: Methodology and Experience”, *Proceedings ACM SIGCOMM*, August/September 2000.

[FGW98] A. Feldmann, A. C. Gilbert, and W. Willinger, “Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic”, *ACM Computer Communication Review*, 28:42–55, September 1998.

[FLMT00] D. R. Figueiredo, B. Liu, V. Misra and D. Towsley , “On the Autocorrelation Structure of TCP Traffic”, UMass CMPSCI Technical Report TR 00-55, 2000

[Flo96] S. Floyd, “Comments on Measurement-based Admissions Control for Controlled-Load Service”, *Technical report*, July 1996, available at ftp://ftp.ee.lbl.gov/papers/admit.ps.Z

[GAN91] R. Guerin, H. Ahmadi, M. Naghshineh, “Equivalent capacity and its application to bandwidth allocation in high-spped networks”, *IEEE JSAC*, 9(7):968-981, Sept. 1991

[GaWi94] M. Garrett and W. Willinger, “Analysis, modeling and generation of self-similar VBR video traffic”, *In Proc. ACM SIGCOMM*, pages 269–280, 1994.

[GCM00] L. Guo, M. Crovella and I. Matta, “TCP Congestion Control and Heavy Tails”, Technical Report BUCS-TR-2000-017, Boston University, 2000

[GiKe97] R. J. Gibbens, F. P. Kelly, “Measurement-Based Connection Admission Control”, *ITC 15*, Jun. 1997

[HBWW99] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, “Assured Forwarding PHB Group”, *RFC 2597*, December 1999

[IEEE802.11] IEEE, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, *IEEE Standard 802.11*, June 1999

[Jac88] V. Jacobson, “Congestion avoidance and control”, *In Proceedings of ACM SIGCOMM*, pages 314–329, Stanford, USA, 1988.

[JaCh81] N. Jayant, S. W. Christensen, “Effect of packet losses in waveform coded speech and improvement due to an add-even sample-interpolation procedure”, *IEEE Transactions on Communications*, vol. 29, pp. 101-109, Feb. 1981.

[JDSZ97] S. Jamin, P. Danzig, J. Shenker, L. Zhang, “A Measurement-Based Admission Control Algorithm for Integrated Service Packet Networks”, *IEEE/ACM Trans. on Networking*, 5(1):56-69. Feb. 1997

[Kel96] F. P. Kelly, “Notes on Effective Bandwidths”, In F. P. Kelly, S. Zachary and I. B. Ziedins “Stochastic Networks: Theory and Applications, Royal Statistical Society”, *Lecture Note Series 4*, p 141-168, Oxford Univ. Press, 1996.

[Kni96] E.W.Knightly, “H-BIND: A new approach to providing statistical performance guarantees to VBR traffic”, *IEEE Infocom’96*, San Francisco, 1996

[KoMi98] K. Kontovasilis, N. Mitrou, “Effective Bandwidths for a Class of Non Markovian Fluid Sources”, *ACM SIGCOMM’98*, Cannes, Sept. 1997

[KWC93] G. Kesidis, J. Walrand, C. Chang, “Effective Bandwidths for Multiclass Markov Fluids and Other ATM Sources”, *IEEE/ACM Trans. Networking*, 1(4):424-428, Aug. 1993

[LBS97] S. Lu, V. Bharghavan, R. Srikant, “Fair Scheduling in Wireless Packet Networks”, *Proceedings of ACM SIGCOMM*, Cannes, France, 1997

[LeB98] J. Y. Le Boudec, “Application of Network Calculus To Guaranteed Service Networks”, *IEEE Trans. on Information theory*,(44)3, May 1998

[LTWW93] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson, “On the Self-Similar Nature of Ethernet Traffic”,*ACM SIGCOMM ’93*, San Francisco, CA, USA, Sep. 1993

[LTWW94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, “On the self-similar nature of Ethernet traffic (extended version)”, *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[MCMK02] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz and L. Stibor, “IEEE 802.11e Wireless LAN for Quality of Service”, *in Proc. European Wireless (EW’2002)*, Florence, Italy, February 2002.

[MDV99] S. Molnár, T. D. Dang, and A. Vidács, “Heavy tailedness, long-range dependence and self-similarity in data traffic”, *In Proc. 7th International Conference on Telecommunication Systems Modeling and Analysis*, Nashville, Tennessee, USA, March 1999.

[MoPa92] M. Mouly, M. B. Pautet, “The GSM System for Mobile Communications”, *Telecom Pub*, June 1992

[Mor97] R. Morris, “TCP Behavior woth Many Flows”, *IEEE International Conference on Network Protocols*, October 1997.

[MoVi97] S. Molnár and A. Vidács, “On modeling and shaping self-similar ATM traffic”, *In Proc. 15th Int. Teletraffic Congress*, Washington D.C., 1997.

[MSMO97] M. Mathis, J. Semske, J. Mahdavi and T. Ott, “The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm”, *Computer Communication Review*, 27(3), July 1997

[NKGB98] T. Nandagopal, T.-E. Kim, X. Gao, V. Bharghavan, “Achieving MAC Layer Fairness in Wireless Packet Networks”, *Proceedings of MOBICOM*, Boston, MA, Aug. 2000

[NKSB99] T. Nandagopal, T. Kim, P. Sinha and V. Bharghavan, “Service Differentiation Through End-to-end Rate Control in Low Bandwidth Wireless Packet Networks.” *IEEE International Workshop on Mobile Multimedia Communications*, San Diego, CA. November 1999.

[NLB99] T. Nandagopal, S. Lu, V. Bharghavan, “A Unified Architecture for the Design and Evaluation of Wireless Fair Queueing Algorithms”, *Proceedings of MOBICOM*, Seattle, WA, Aug. 1999

[Nor94] I. Norros, “A storage model with self-similar input”, *Queuing Systems*, 16:387–396, 1994.

[NS] NS software and documentation is available at the following site: http://www-mash.CS.Berkeley.EDU/ns

[Ott93] E. Ott, “Chaos in Dynamical Systems”, *Cambridge University Press* (July 1993)

[PaFl95] V. Paxson and S. Floyd, “Wide Area Traffic: The Failure of Poisson Modeling”, *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226-244, June 1995.

[PaFl97] V. Paxson, S. Floyd, “Why We Don’t Know How To Simulate The Internet”, *Winter Simulation Conference*, Dec. 1997

[PaGa93] A.K. Parekh and R.G.Gallager, “A generalized processor sharing approach to flow control in integrated services networks – the single node case”, *IEEE/ACM Trans. on Networking*, 1(3):334-357, 1993

[PaGa94] A.K. Parekh and R.G.Gallager, “A generalized processor sharing approach to flow control in integrated services networks – the multiple node case”, *IEEE/ACM Trans. on Networking*, 2(2):137-150, 1994.

[PCFS80] N. H. Packard, J. P. Crutchfield, J. D. Farmer and R. S. Shaw, “Geometry from a Time Series”, *Phys. Rev. Lett.*, 45 (1980) 712-716

[PFTK98] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation”, *Proccedings of SIGCOMM’98,* Vancouver, CA, September 1998

[PJS92] H. O. Peitgen, H. Jurgens and D. Saupe, “Chaos and Fractals - New Frontiers of Science”, *Springer-Verlag*, NY, 1992

[PKC96] K. Park, G. Kim, and M. Crovella, “On the relationship between file sizes, transport protocols, and self-similar network traffic”, *In Proceedings of the International Conference on Network Protocols*, pages 171–180, October 1996.

[PKC97] K. Park, G. Kim, and M. Crovella, “On the effect of traffic self-similarity on network performance”, *In Proceedings of the SPIE International Conference on Performance and Control of Network Systems*, pages 296–310, November 1997.

[RFC793] Postel, J., ”Transmission Control Protocol - DARPA Internet Program Protocol Specification”, RFC 793, DARPA, September 1981

[QiKn99] J. Qiu and E. Knightly, “Inter-Class Resource Sharing using Statistical Service Envelopes,” *in Proceedings of IEEE INFOCOM ’99*, New York, March 1999.

[SaSh91] H. Saito, K. Shiomoto, “Dynamic call admission control in ATM networks”, *IEEE JSAC*, 9(7):982-989, Sep. 1991.

[Shen95] S. Shenker, “Fundamental Design Issues for the Future Internet”, *Journal on Selected Areas in Communications*, Sept. 1995.

[SoKr96] J. L. Sobrinho, A. S. Krishnakumar, “Real-Time Traffic over the IEEE 802.11 Medium Access Control Layer”, *Bell Labs Technical Journal*, Autumn 1996

[SRC84] J.H. Saltzer, D.P. Reed and D.D. Clark, “End-to-end Arguments in System Design”, *ACM Transactions on Computer Systems*, Nov 1984, p. 277-288.

[RFC2001] W. Stevens, “ TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, *RFC2001*

[RFC2018] M. Mathis et al., “TCP Selective Acknowledgement Options”, *RFC2018*

[RFC1072] V. Jacobson, R. Braden, “TCP Extensions for Long-Delay Paths”, *RFC1072*

[TsGe97] B. Tsybakov and N. D. Georganas, “On self-similar traffic in ATM queues: Definitions, overflow probability bound, and cell delay distribution”, *IEEE/ACM Transactions on Networking*, 5(3):397–409, June 1997.

[TTW95] M. S. Taqqu, V. Teverovsky and W. Willinger, “Estimators for long-range dependence: an empirical study”, *Fractals*, Vol 3, No. 4. pp. 785-788, 1995.

[TTW97] M. S. Taqqu, V. Teverovsky and W. Willinger, “Is Network Traffic Self-Similar or Multifractal?”, *Fractals*, 5 (1997) 63-73

[TWS97] M. S. Taqqu, W. Willinger and R. Sherman, “Proof of a Fundamental Result in Self-Similar Traffic Modeling”, *Computer Communication Review*, 27 (1997) 5-23.

[Val99] A. Valko, “Cellular IP – A New Approach to Internet Host Mobility”, *Computer Communication Review*, Vol. 29. pp. 50-65, Jan. 1999

[VeAb99] D. Veitch and P. Abry, “A statistical test for the time consistancy of scaling exponents”, *Preprint*, 1999.

[VeZa95] M. A. Visser, M. E. Zarki, “Voice and Data Transmission over an 802.11 Wireless Network”, *Proceedings of PIMRC*, Toronto, Canada, Sept. 1995

[WKLZ96] D. Wrege, E. Knightly, J. Liebeherr, and H. Zhang, “Deterministic Delay Bounds for VBR Video in Packet-Switching Networks: Fundamental Limits and Practical Trade-offs,” *IEEE/ACM Trans. on Networking*, 4(3):352-362, June 1996.

[WSFW97] J. Weinmiller, M. Schlager, A. Festag, A. Wolisz, “Performance Study of Access Control in Wireless LANs - IEEE 802.11 DFWMAC and ETSI RES 10 HIPERLAN”, *Mobile Networks and Applications*, Vol. 2, pp. 55-67, 1997

[WTE96] W. Willinger, M. S. Taqqu, and A. Erramilli, “A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks”, *In Stochastic networks: Theory and applications, Ed:F. P. Kelly, S. Zachary and I. Ziedins*, Clarendon Press, Oxford, 1996.

[WTSW97] W. Willinger, M. Taqqu, R. Sherman, and D. 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, February 1997.

[WWEW95] J. Weinmiller, H. Woesner, J. P. Ebert, A. Wolisz, “Analyzing and Improving the 802.11-MAC Protocol for Wireless LANs”, *Proceedings of MASCOT*, San Jose, CA, Feb. 1995

[ZhKn94] H. Zhang, E. Knightly, “Providing end-to-end statistical performance guarantee with bounding interval dependent stochastic models”, *ACM SIGMETRICS’94*, 1994

[ZLKT97] Z.L. Zhang, Z. Liu, J. Kurose and D. Towsley, “Call Admission Control Schemes under the Generalized Processor Sharing Scheduling Discipline”, *Telecommunication Systems*, 7(1-3):125-152, July 1997

[ZTK95] Z.L. Zhang, D. Towsley and J. Kurose, “Statistical Analysis of Generalized Processor Sharing Scheduling Discipline”, *IEEE JSAC*, 13(6):1071-1080, Aug. 1995

[YaSi92] O. Yaron and M. Sidhi, “Performance and Stability of Communication Networks via Robust Exponential Bounds”, *IEEE Trans. on Networking*, 1(3):372-385, Jun. 1992

[YaSi94] O. Yaron and M. Sidhi, “Generalized Processor Sharing Networks with Exponentially Bounded Burstiness Arrivals”, *Journal of High Speed Networks*, 3:375-387, 1994

Publications

Journal Publications

[J1] **A. Veres**, A.T. Campbell, M. Barry and L-H. Sun, “Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control”, *IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on Mobility and Resource Manegement in Next-Generation Wireless Systems*, Vol. 19, No. 10, pp. 2094-2104, October 2001.

[J2] **A. Veres**, Zs. Kenesi, S. Molnár, G. Vattay, “TCP’s Role in the Propagation of Self-Similarity in the Internet”, *Computer Communications, Special Issue on Performance Evaluation of IP Networks and Services, Elsevier Science*, Volume 26, Issue 8, pp. 899-913, May 2003.

[J3] G-S. Ahn, A.T. Campbell, **A. Veres** and L-H. Sun, “Supporting Service Differentiation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)”, *IEEE Transactions on Mobile Computing, Special Issue on Mobile Ad-Hoc Networks.*, Vol. 1, No. 3, pp 192-207, July-September 2002

[J4] M. Boda, G. Vattay, **A. Veres**, “Kaotikus viselkedés az Interneten”, *Természet Világa*, Vol. 134. , Issue 1, January 2003.

Conference Publications

[C1] Z. R. Turányi, **A. Veres**, A. Olah, “A Family of Measurement-Based Admission Control Algorithms”, *In Proc. IFIP Performance of Information and Communication Systems PICS’98* Lund, June 1998

[C2] **A. Veres** and M. Boda, “The chaotic nature of TCP congestion control”, *In Proc. IEEE INFOCOM*, Tel Aviv, Israel, April 2000.

[C3] **A. Veres**, Zs. Kenesi, S. Molnár, G. Vattay, “On the Propagation of Long-Range Dependence in the Internet”, *In Proc. ACM SIGCOMM Computer Communication Review 30, No 4, 243-254 (2000)*, Stockholm, Sweden, Sep. 2000.

[C4] **A. Veres**, Z. Turányi, “On the Tradeoff between Efficiency and Scalability of Measurement-Based Admission Control”, *In Proc. International Conference on the QoS and Performance of Next Generation Networking (P&QNet)*, Nagoya, Japan, Nov. 2000

[C5] M. Barry, A. T. Campbell, **A. Veres**, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks", *In Proc. IEEE INFOCOM*, Anchorage, Alaska, April 2001

[C6] Gahng-Seop Ahn, Andrew T. Campbell, **A. Veres** and Li-Hsiang Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks", *In Proc. IEEE INFOCOM*, New York, June 2002

[C7] A. Fekete, G. Vattay and **A. Veres**, "Improving the 1/sqrt law for single and parallel TCP flows", *In Proc. International Teletraffic Congress ITC-17*, Salvador da Bahia, Brazil, December, 2001

[C8] P. Benkõ and **A. Veres**, "A Passive Method for Estimating End-to-End TCP Packet Loss", *In Proc. IEEE GLOBECOM*, Taipei, Taiwan, November, 2002

[C9] P. Benkõ, G. Malicskó and **A. Veres**, ""A Large-scale, Passive Analysis of End-to-End TCP Performance over GPRS ", accepted to IEEE INFOCOM 2004

Workshops

[W1] **A. Veres**, Z. Turányi, "On the Tradeoff between Efficiency and Scalability of Measurement-Based Admission Control", *IFIP WG6.3 Workshop*, Crete, Greece, Aug 1999

[W2] **A. Veres**, "On TCPs Contribution to Wide-Scale Self-Similarity", *Qualitiy of Service in Networks and Distributed Systems, Dagstuhl Seminar*, Germany, May 2000.

[W3] **A. Veres**, M. Boda, "On the Impact of Short Files and Random Losses on Chaotic TCP Systems", *In Proc. IFIP ATM & IP 2000 Workshop*, Ilkley, UK, July 2000.

[W4] **A. Veres**, "Service Differentiation in Wireless Packet Networks", *IEEE Computer Communications Workshop (CCW 2000)*, Captiva Island, Florida, October 2000

Standardization and Patent Publications

[P1] **A. Veres**, J. Barta, "Soft handover method for mobile access network using packet transmission, involves effecting buffer management to nodes handling data packets marked according to channel priority ", *patent application*, Publication No GB-2367719A, date April 10, 2002

[P2] **A. Veres**, A. Fekete, "Scalable real time quality of service monitoring and analysis of service dependent subscriber satisfaction in IP networks", *patent application*, Publication No US-0706759 date November 7, 2000

[P3] Z. Turányi, **A. Veres**, "Administration control method for switching node in integrated packet-switch network", *patent granted*, No US 6,614,790 B1, date September 2, 2003

[P4] F. Hellstrand, **A. Veres**, “Simulation of TCP/IP Router Traffic over ATM using GFR and VBR.3”, *ATM Forum contribution,* ATM98-0087, February 1998

[P5] G-S. Ahn, A. T. Campbell, **A. Veres**, L. Sun, “SWAN”, *MANET Working Group Internet Draft*, Work in Progress, draft-ahn-swan-manet-00.txt, http://www.ietf.org/internet-drafts/draft-ahn-swan-manet-00.txt, October 2002

[P6] Sz. Malomsoky, **A. Veres**, I. Szabó, T. Borsos, “Advanced performance management of mobile data networks”, *patent application*, Application No PCT/SE03/01517, date September 30, 2003