POLITECNICO DI TORINO

SCUOLA DI DOTTORATO
Dottorato in Ingegneria Elettronica e delle Comunicazioni – XVIII Ciclo

Tesi di Dottorato

On Internet Traffic Measurements, Characterization and Modelling

Luca Muscariello

Tutore
Prof. Fabio Neri

Marzo 2006

Table of contents

List of figures

List of tables

Prof. Andrea **Bianco**	Politecnico di Torino	Commissione esaminatrice
Prof. Maurice **Gagnaire**	ENST Paris	Commissione esaminatrice
Prof. Udo R. **Krieger**	Otto-Friedrich University Bamberg	Esaminatore
Dr. Marco G. **Mellia**	Politecnico di Torino	Correlatore
Prof. Ivo **Montrosset**	Politecnico di Torino	Coordinatore del Dottorato
Dr. James W. **Roberts**	France Télécom R&D	Correlatore
Prof. Matteo **Sereno**	Università di Torino	Commissione esaminatrice
Prof. Tier **Van Do**	Budapest University of Technology	Esaminatore

Acknowledgments

Desidero ringraziare tutte le persone che mi sono state vicine in questi tre anni di dottorato a Torino, in particolar modo i miei genitori Vittorio e Rosalba che hanno costantemente incentivato il mio entusiasmo per lo studio e la ricerca e quindi appoggiato questa mia scelta senza riserve. Ringrazio i miei fratelli Armando e Stefano, coi quali ho condiviso la quotidianità torinese. Dovrei ringraziare Giovanna per i suoi numerosi consigli e lo spirito critico, spesso costruttivo; tuttavia la ringrazierò sopratutto per aver sopportato il mio pessimo carattere. Ringrazio Fabio e Dario che hanno reso molto più colorata la grigia Torino.

I would like to thank the entire Telecommunication Network Group at the Electronic departement of Politecnico di Torino led by Prof. Marco Ajmone Marsan and Prof. Fabio Neri. In particular Prof. Marco Ajmone Marsan for his generous support and research collaboration in the TANGO project (Traffic models and Algorithm for New Generation IP Networks Optimization) as a CNIT collaborator (National Inter-University Consortium for Telecommunications) in the research unit in Torino. I would also thank Prof. Fabio Neri for have being my thesis advisor and for the hospitality and genereous support in the LIPAR Lab (Laboratory in Internet network Protocols and ARchitectures). Heartfelt thanks go to Marco Mellia for his substantial and constructive support to my research always based on effective collaboration.

I want to thank Michela Meo for his encouragement to participate to the COST279 meetings (Analysis and Design of Advanced Multiservice Networks supporting Mobility, Multimedia and Internetworking) that have been a valuable research forum of discussion that stimulated much of my research. Therefore I'd like to thank Prof. Udo Krieger for his constructive discussion from the first COST279 summer school during the summer 2002 in Darmstadt, to the EURO-NGI JRA5.1 activities (Design and Engineering of the Next Generation Internet) meetings. I also thank Prof. Krieger to have accepted to be in the evaluation committee of my PhD. thesis.

I thank the research group led by Dr. Ilkka Norros at the Technical Research Center of Finland (VTT) where I have been working with Jorma Kilpi and Pasi Lassila (from Helsinki University of Technology). I thank them for their warm hospitality in a so cold country, kiitos. I also thank CNR (National Research Council) for the financial support

of my visit in Finland.

I thank Prof. Andrea Bianco, Prof. Matteo Sereno, Prof. Maurice Gagniere and Prof. Tien Van Do to have accepted to be in the evaluation committee of my Ph.D. thesis.

Je ne pourrais jamais remercier assez Jim Roberts pour m'avoir accueilli dans son équipe de recherche chez France Télécom R&D et pour son stupéfiant enthousiasme qui stimule les échanges d'idées et l'engagement à trouver des solutions à problèmes réeles. Je le remercie aussi pour avoir accepté d'être un des rapporteur de ma thèse. Je tiens à addresser mes remerciements à toute l'équipe pour son hospitalité et l'agreable collaboration surtout à Abdesselem Kortebi et Sara Oueslati.

Je n'oublie pas de remercier mon professeur de français et ami Matthieu Lafon, tout les copains de l'alliançe française et ce petit marché que l'on trouve toutes les dimanche matin dans Boulevard Edgar Quinet, à Paris.

Abstract

This thesis is about Internet traffic characterization and its intrinsic properties that strongly influence the performance of the present Internet architecture. The thesis is composed of two main parts. The first is a brief survey of known statistical properties of data traffic with, in addition, some original results on traffic at flow and session layer. It is also shown how complex statistical properties can be approximated by simpler Markovian models through a newly developed packet model and its single server queuing performance. The second part is composed of some examples to show that traffic theory is an effective tool to evaluate the performance of the Internet. In particular scalability aspects of fair queuing have been considered under random traffic using flow and packet traffic models. The performance evaluation of fair-queuing to realize statistical bandwidth sharing has led to define an alternative Internet architecture that keeps that actual best effort and end-to-end paradigms and that empowers its capability to provide QoS in a statistical sense in opposition to previous proposition as diffserv/intserv. All results have been compared with real traffic measurements gathered from different IP networks in Europe and USA collected at both access and backbone links.

Introduction

The focus of this thesis is on the study of Internet traffic, by means of characterization of real measurements, and also on modelling to evaluate the performance of specific architectures in the Internet, e.g. per-flow fair queueing.

The thesis work has been done primarily at Politecnico di Torino with main focus on Internet traffic characterization at flow and session level, whose contents come from the published material in [GMMR04, RMM04, BMM+05a, BMM+05b, MMM05].

All these studies have been carried out under the research supervision of Dr. Marco Mellia that coordinates the measurement activities of the Telecommunication Network Group at the EE Department, led by Prof. Marco Ajmone Marsan. The measurement activity comprises the hard job of setting up and testing the hardware equipments collecting and storing raw packet data with high accuracy from different network links.

Data collection is followed by the post-processing needed to obtain flow level statistics which are the starting point of most of the analysis presented in Chapters 2 and 3. Flow level statistics have been obtained using a tool called Tstat (`http://tstal.tlc.polito.it`) developed at the Electronic Department of our institute and maintained primarily by Dr. Marco Mellia.

The thesis is organized in six chapters. Chapter 1 is a brief description of the measurement points that were the source of our traffic data. Chapter 2 is devoted to the study of flow and session level traffic. More precisely, Section 2.2 studies time correlation properties of different aggregates of traffic sharing the same network link, and we show how this can be used to explain time correlation at packet level also. By contrast section 2.3 focus on a very particular kind of traffic that is made every anomalous TCP packet. The term anomalous is used in a un-proper way just to tag all packets that have been measured as out of sequence or duplicated at the monitor point. We have studied the time correlation properties of such traffic in order to model the network feedback that drives every TCP flow. Section 2.4 investigates the relationship between TCP flow size, duration and mean rate making use of the statistical framework of copula. This study has been object of an internship in the labs of the Technical Research center of Finland, VTT in the research group led by Dr. Ilkka Norros.

Chapter 2 concludes with a study devoted to find empirical evidence that web-user sessions are open according to a Poisson process. Web-sessions are not defined from

a protocol point of view, nevertheless a heuristic approach is developed in Section 2.5. Statistical analysis on web-sessions is described in 2.9.1.

The thesis continues in Chapter 3 whose contents can also be found in [MMM+04b, MMM+04a]. This chapter describe a measurement based traffic model that has a hierarchical structure that is able to induce time correlation mimicking the traffic generation of a network source in session, flow and packets.

Chapters 4,5 report also the results of an internship at the research labs in France Télécom (Issy-Les-Moulineaux) under the supervision of Dr. James Roberts. The internship focused on scalability of fair-queueing algorithms under random traffic and its relations to traffic characteristics and this can be found in the published papers [KMOR04, KMOR05a, KMOR05b].

Chapter 1

Traffic Measurements

In this chapter we list all measurements that have been used in this thesis, collected at different Internet network links with different capacities in different portions of the networks, from the edge to the backbone, in Europe and in the USA. We also describe the measurement tools that have been used to obtain our statistics at different levels, from the packet to the flow traces.

Data collection is a complex task that might also be expensive because of the high performance of the network adapters needed at high speed links. The complexity of packet collection increases with link capacity and this results in high precision hardware equipments like DAG cards (`http://www.endace.com/products.htm`).

We have used two different approaches to deal with network data collection: on the shelf hardware, at the edge of the network on relatively low rate links (up to 100Mbps), and dedicated hardware like DAG cards on the backbone (up to 10Gbps)

On the shelf hardware has shown to be feasible below 100Mbps, while above, dedicated cards came to be necessary: packets are lost at the measurement point and packet timestamping is no longer accurate.

1.1 Measurements Points Description

The main source of our traffic measurements has been the unique external IP link that connects the switched-LAN of our institution to the Internet. We had access to this link from 2000 to 2005, during which capacity has been upgraded many times to higher rates: 4Mbps during 2000, 16Mbps during 2001, 28Mbps from 2002 to 2004, 48Mbps during 2005. We have used measurements collected during all the aforementioned time periods throughout the present thesis. Our campus network is built upon a large 100 Mbps Ethernet LAN, which collects traffic from more than 7,000 hosts, hence it is representative of a highly loaded Internet edge link. It is mainly composed of clients, but some servers are also regularly accessed from the outside network.

Other two traces measured at the edge of the network have been analyzed in two different studies done during two different international collaboration: France Télécom R&D in Issy-Les-Moulineaux and in VTT (Technical research center of Finland) in Helsinki. These two data traces are not publicly available because of privacy concerns. The first data set is an OC3 link measurement concentrating the traffic of several thousand ADSL users of the France Télécom network and it represents 5 minutes of data recorded on August 25 2003; The second one has been measured from a GPRS/UMTS cell that aggregates traffic of mobile users of the Finnish operator Elisa. ADSL and GPRS/UMTS accesses are rate limited compared to the university campus of Politecnico di Torino, therefore they present a traffic mix which is completely different from other available data.

As far as the backbone concerns, traffic traces have been collected from two different networks: Abilene (`http://abilene.internet2.edu/`) and GARR (`http://www.garr.it/`). Abilene is an Internet2 high-performance backbone network built to develop and test new Internet services and applications as virtual laboratories and digital libraries. It has been created by the Internet2 community and it serves 220 universities, corporate, and affiliate member institutions in USA. The network is primarily a OC-192c backbone employing optical technologies and high performance routers. The National Laboratory for Applied Network Research (NLANR `http://www.nlanr.net/`) makes publicly available packet data measurements gathered from the Abilene backbone network. This network serves users with very high access rates and they are not usually bottle-necked because of network congestion, as we will see later, but more probably because of congested servers or applications. We have considered two traces among all the available measurements on the web-site. The AbileneI trace is the first OC48c Packet-over-SONET data set published by the NLANR MNA team. Data consist of a pair of two hour contiguous bidirectional packet header traces collected at the Indianapolis router node (IPLS). The links instrumented are the ones eastbound and westbound, towards Cleveland (CLEV) and Kansas City (KSCY) (`http://pma.nlanr.net/Traces/long/ipls1.html`). AbileneIII, the second trace, was collected on June 1st, 2004 at the OC192c Packet-over-SONET link from Indianapolis (IPLS) Abilene router node towards Kansas City (KSCY). The OC192MON hardware used in this data collection deployed precision timestamping and global CDMA/GPS time synchronization (`http://pma.nlanr.net/Special/ipls3.html`) . The map of the Abilene backbone is drawn in figure 1.1

Another backbone link comes from the GARR network which supplies connectivity to all Italian Academic and Scientific Institutions and it takes part into the European research network consortium GEANT. Users are very etherogeneous and a backbone link merges traffic from low to high access rates. We have measured on the the backbone in different time periods, from 2004 to 2005 with increasing frequency. The map of the GARR backbone is shown in figure 1.2.

Trace	Year	Network area
Polito01	2001	Edge
Polito02	2002	Edge
Polito03	2003	Edge
Polito04	2004	Edge
Polito05	2005	Edge
ADSL-FT	2003	Edge
Elisa	2004	Edge
GARR	2005	Backbone
AbileneIII	2004	Backbone
AbileneI	2003	Backbone

Table 1.1. Summary table of the available network measurements.

Figure 1.1. Map of the Abilene backbone network.

1.2 Measurement Tools

Traffic measurements have been analyzed at the flow level. The definition of flow is clear whether TCP is considered, while for any other kind of Internet traffic it is even questionable to say that flow can be measured in the present Internet architecture. TCP flows are also bidirectional, hence easy to measure at the edge. However, at the backbone, packets flowing into the upstream and downstream channels could not be easily collected at the same measurement point because Internet routing may be not symmetric. The

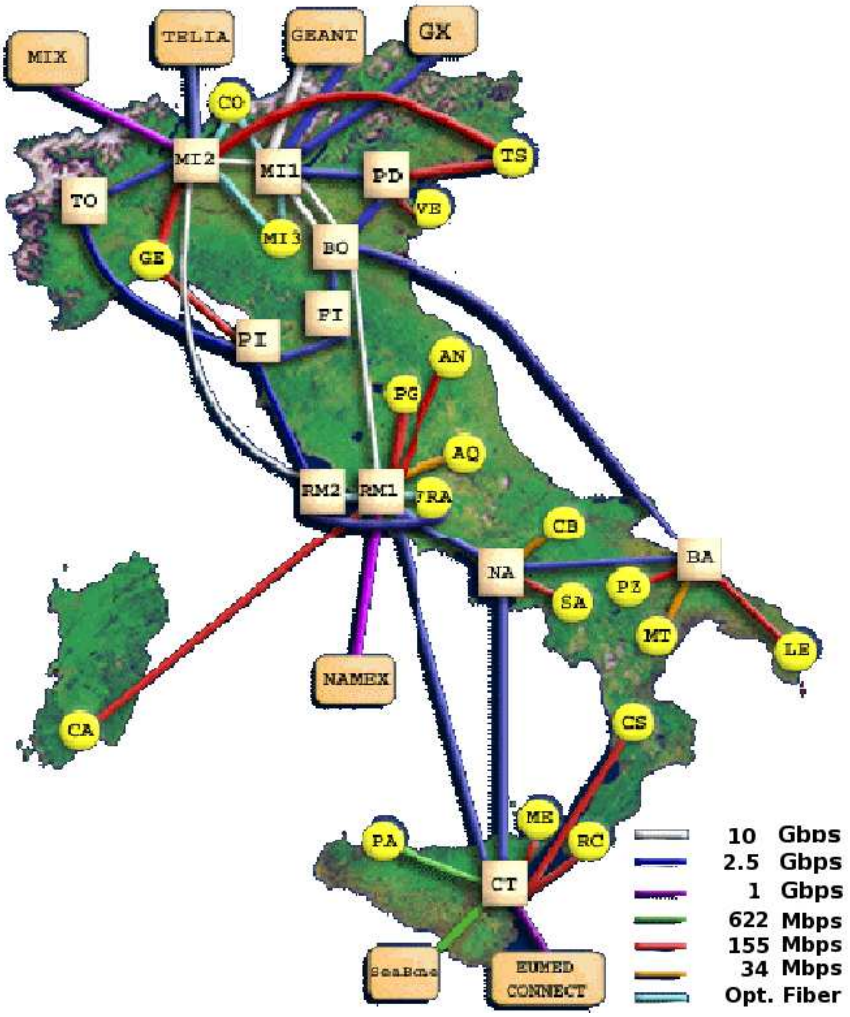


Figure 1.2. Map of the GARR backbone network.

Abilene backbone traces, for example, have only about 50% symmetric traffic , thus an alternative definition of flow would be appropriate. In the present thesis we have usually considered TCP to define flows exept in the chapters 4,5 where flows are better defined in a single link direction. A flow in this context is a flight of datagrams, localized in time (packets are spaced by no more than a certain interval) and space (packets in question are observed at a particular interface) and having the same unique identifier. The identifier is deduced from header fields including IP addresses and user-specified fields like the IPv6 flow label or IPv4 port numbers. The expectation is that users define flows to correspond to a particular instance of some applications such as a video stream or a document transfer. This definition is much more general and could fit to a wider class of Internet traffic.

This second definition of flow can always be measured very easily while strict TCP definition imposes to monitor data packets and acknowledgments also. We have used Tstat (`http://tstat.tlc.polito.it`), a tool developed at Politecnico di Torino, to collect almost any kind of information related to the status of TCP connections. The edge of the network is the natural region to monitor TCP flows where routing is necessarily symmetric.

We have run Tstat on all Polito traces in table 1.1, on Elisa and GARR. The latter is a rare case of symmetric backbone link that permitted us to collect the statistics of 100% of TCP traffic.

Chapter 2

Internet Traffic Characteristics

The seminal paper by Leland et al. [LTWW94] led the teletraffic community to deal with data traffic measurements, especially in the Internet, such that, nowadays, most of the recent literature on the performance evaluation of the Internet, very often, takes into account many aspects related to the statistical nature of Internet traffic (see the mathematical appendix). Nevertheless even if traffic characterization helped understand the complexity of the performance evaluation of computer networks it did not help much to solve most of the arisen problems that are subject of research still now.

In the next sections we briefly present the main characteristics of Internet traffic at different levels of abstraction, i.e. packets, flows, and sessions. The topic has been deeply investigated in the literature and here we report a summary of it with, in addition, some original results that have been the subject of this thesis.

Packet traffic is a stochastic process, because IP datagrams arrive randomly according to the coordination between the user's behaviour and network protocols. The packet arrival process has been found to be highly complex from a statistical point of view and [LTWW94, PF95, BSTW95, TG97, CB97] are some of the most important papers that pointed out the self-similar nature of data traffic. In this monograph we focus on flow traffic after have chosen a proper definition of flow in an IP network.

2.1 Flow Level Traffi c

Traffic can be thought at a different level of abstraction than that of the streams of packets that actually cross the network and it can be viewed as a flow process. We use two different definitions of flows, a unidirectional and a bidirectional definition that we chose regarding our objective; for example, we would prefer a bidirectional flow whether we focus on TCP traffic. Nevertheless the present architecture is already carrying a mixture of traffic of different nature, that is usually classified as streaming and elastic. This traffic mix has no bidirectional definition because of streaming flows (that have no opening and closing

protocol procedure), and also peer to peer flows that are a kind of multi-point-to-point elastic data transactions, therefore a uni-directional flow definition is more convenient for studying the performance of a network interface.

This justifies our second flow definition, further in this monograph in the chapter 4,5 that present results on the scalability of fair queueing. In this chapter we consider TCP traffic afterwards a bidirectional flow definition is suitable. Note that it is not always possible to measure bidirectional flows mainly because of asymmetric routing which is dominant in the Internet. Hence such measurements are available only at the network's edge or in very special backbone links where routing is symmetric for very specific architectural reasons.

In this chapter a flow is a measured valid TCP flow from a passive measurement and we say that a flow is valid if the three-way handshake is detected at the measurement point. The TCP opening transaction is the exange of three packets: the SYN packet sent by the client, the SYN-ACK from the server and a third packet that acknowledges the opening of both directions.

2.2 TCP Arrivals Properties

Instead of considering the packet level trace, we performed a live collection of data directly at the TCP flow level, using `Tstat` [MCC02, Mel05], a tool able to keep track of single TCP flows by looking at both data and acknowledgment segments. The flow level trace was then post-processed by `DiaNa` [Ros05], a novel tool which allowed to easily derive several simple as well as very complex measurement indexes in a very efficient way. Both tools are under development and made available to the research community as open source.

To gauge the impact of elephants and mice on the TCP flow arrival process, we follow an approach similar to [ENW96, HVA02], that creates a number of artificial scenarios, deriving each of them from the original trace into a number of sub-traces, mimicking the splitting/aggregation process that traffic aggregates experience following different paths inside the network. We then study the statistical properties of the flow arrival process of different sub-traces, showing that the LRD (see mathematical appendix) tends to vanish on traffic aggregates composed mostly of TCP-mice.

2.2.1 Preliminary Definitions

When performing trace analysis, it is possible to focus on different level of aggregations. Considering the Internet architecture, it is common to devise three different levels, i.e., IP-packet, TCP/UDP flow, and user session level. While the definition of the first two layers is commonly accepted, the user session one is more fuzzy, as it derives from the user behavior. In this paper we therefore decided to follow a different approach in the

definition of aggregates. In particular, to mimic the splitting/aggregation process that data experience following different paths in the network, we define four level of aggregation, sketched in Fig. 2.1: IP packets, TCP flows, Traffic Relations (TR), and Traffic Aggregates (TA). Being interested into the flow arrival process, we neglect the packet level, and also the UDP traffic, because of its connectionless nature, and because it consists of a small portion of the current Internet traffic.

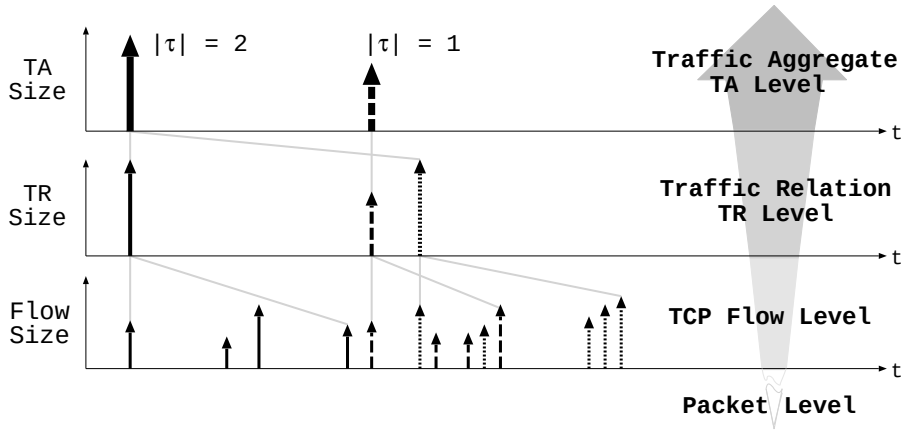


Figure 2.1. Aggregation Level From Packet Level to TA Level

Let us first introduce the formal definition of the different aggregation levels considered in this paper, as well as the related notation:

- **TCP flow Level**
  A single TCP connection[1] is constituted, as usual, by several packets exchanged between the same client c (i.e., the host that performed the *active* open) and the same server s (i.e., the host that performed the *passive* open), having besides the same (source,destination) TCP ports pair. We consider only successfully opened TCP flows –i.e., whose three-way-handshake was successful– and we consider the flow arrival time as the observation time of the first client SYN segment. We denote with $F_i(c,s)$ the byte-wise size of the i-th TCP flow tracked during the observation interval; the flow size considers the amount of bytes flowing from the server d toward the client s, which is usually the most relevant part of the connection.

- **Traffic Relation Level**
  A Traffic Relation (TR) $\mathcal{T}(c,s)$ aggregates all the $|\mathcal{T}(c,s)|$ TCP flows, 'having c as source and s as destination; we indicate its size, expressed in bytes, with $T_R(c,s) = \Sigma_{i=1}^{|\mathcal{T}(c,s)|} F_i(c,s)$. The intuition behind this aggregation criterion is that all the packets

[1] In this paper we use the term "flow" and "connection" interchangeably.

within TCP flows belonging to the same TR usually follow the same path(s) in the network, yielding then the same statistical properties on links along the path(s).

- **Traffic Aggregate Level**
  Considering that several TRs can cross the same links along their paths, we define a higher level of aggregation, which we call Traffic Aggregate (TA). The J-th TA is stated by τ_J and its byte-wise size is $T_A(J) = \Sigma_{(c,s)\in\{\tau_J\}} T_R(c,s)$; the number of traffic relations belonging to a traffic aggregate is indicated with $|\tau_J|$.

In the following, rather than using the byte-wise size of flows, TRs and TAs, we refer to their *weight*, that is, their size expressed in bytes normalized over the total amount of bytes observed in the whole trace $W = \Sigma_i \Sigma_{(c,s)} F_i(c,s)$; thus

$\hat{w}_i(c,s) = F_i(c,s)/W$	TCP flow Level
$\hat{w}_{cs} = T_R(c,s)/W$	Traffic Relation Level
$\hat{w}_J = T_A(J)/W$	Traffic Aggregate Level

We used `Tstat` to perform the live analysis of the incoming and outgoing packets sniffed on the router Politecnico di Torino WAN link, obtaining several statistics at both packet and flow levels. Moreover, `Tstat` dumped a flow-level trace, which has then been split into several traces, each of which refers to a period of time equal to an entire day of real traffic. Besides, since our campus network is mainly populated by clients, we consider in this analysis only the flows originated by clients internal to our Institution LAN. Each trace has been then separately analyzed, preliminary eliminating the non-stationary time-interval (i.e., night/day effect) and considering then a busy-period from 8:00 to 18:00. Given that the qualitative results observed on several traces do not change, in this paper we present results derived from a single working day trace, whose properties are briefly reported in Tab. 2.1. During the 10 hours of observation, 2,380 clients contacted about 36,000 different servers, generating more than 172,000 TRs, for a total of more then 2.19 million TCP flows, or 71.76 million packets, or nearly 80 GBytes of data.

In the considered mixture of traffic, TCP protocol represents the 94% of the total packets, which allows us to neglect the influence of the other protocols (e.g., UDP); considering the application services, TCP connections are mainly constituted by HTTP flows, representing the 86% of the total services and more than half of the totally exchanged bytes. More in detail, there are 10664 different identified TCP server ports, 99 of which are well-known: these account for 95% of the flows and for the 57% of the traffic volume.

Considering the different traffic aggregation levels previously defined, Fig. 2.2 shows examples of the flow arrival time sequence. Each vertical line represents a single TCP flow, which started at the corresponding time instant of the x-axis, and whose weight $\hat{w}_i(c,s)$ is reported on the y-axis. The upper plot shows trace τ_A, whose $\hat{w}_A = 1$, and represents the largest possible TA, built considering all connections among all the possible source-destination pairs. Trace sub-portions τ_B and τ_C, while being constituted by a rather

Table 2.1. Trace Informations

Internal Clients	2,380	Flows Number	$2.19 \cdot 10^6$
External Servers	35,988	Packets Number	$71.76 \cdot 10^6$
Traffic Relations	172,574	Total Trace Size	79.9 GB

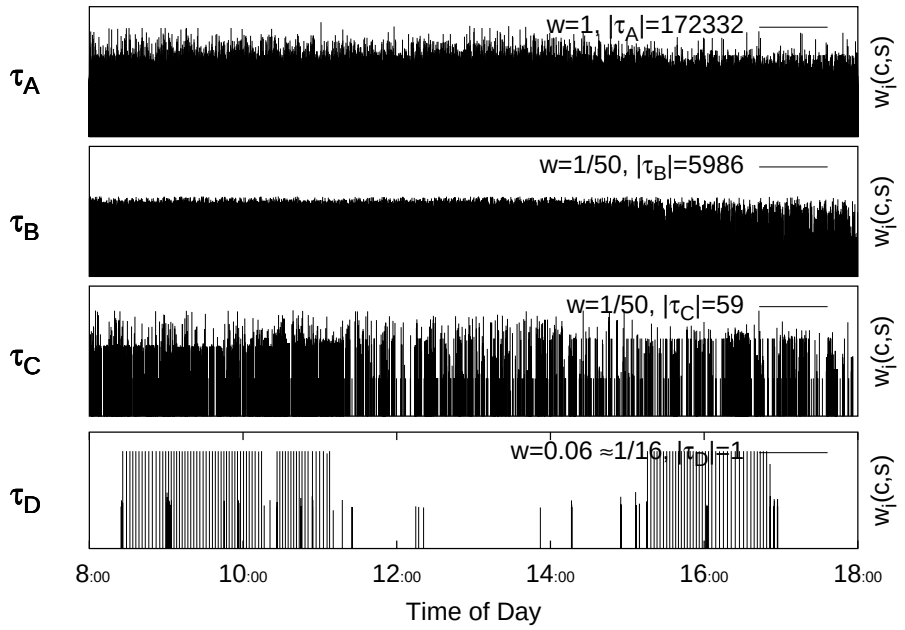


Figure 2.2. Flow Size and Arrival Times for Different TAs

different number of TRs ($|\tau_B| = 5986$ and $|\tau_C| = 59$), have indeed the same weight $\hat{w}_B = \hat{w}_C = 1/50$. Observing Fig. 2.2, it can be gathered that τ_B aggregates a larger number of flows than τ_C; furthermore, weight of τ_B flows is smaller (i.e., TCP flows tend to be "mice"), while τ_C is build by a much smaller number of heavier (i.e., "elephants") TCP flows. This intuition will be confirmed by the data analysis presented in Sec. 2.2.4. Finally, TCP flows shown in τ_D constitute a unique traffic relation; this TR is built by a small number of TCP flows, whose weight is very large, so that they amount to 1/16 of the total traffic.

To give the reader more details on the statistical properties of the different TRs, we show in Fig. 2.3 using a log/log plot. Indeed, the almost linear shape of the pdf shows that it exhibits a heavy tail, which could be at the basis of the LRD properties in the TCP-flow arrival process. The analogy with the heavy-tailed pdf of the flow-size, which induces LRD properties to the packet level, is straightforward.

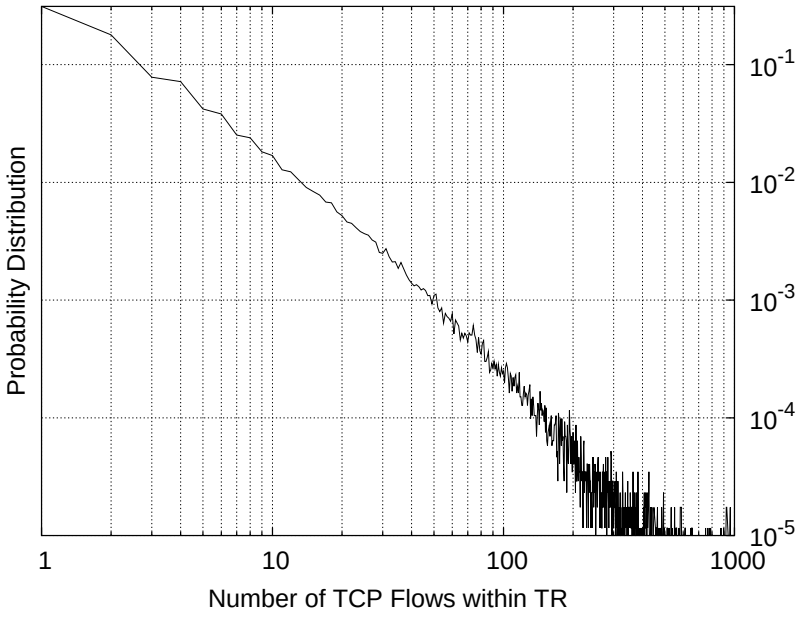


Figure 2.3. TR Flow Number Distribution

2.2.2 Properties of the Aggregation Criterion

We designed the aggregation criterion in order to satisfy some properties that help the analysis and interpretation of the results.

The key-point is that the original trace is split into K different TA, such that i) each TA has, byte-wise, the same amount of traffic, i.e., the K-th portion of the total traffic and ii) each TA aggregates together one or more TRs. Indeed, we considered TR aggregation a natural choice, since it preserve the characteristics of packet within TCP flows following the same network path, having therefore similar properties.

Being possible to find more than one solution to the previous problem, the splitting algorithm we implemented packs the largest TRs in the first TAs; besides, in virtue of the byte-wise traffic constraint, subsequently generated aggregates are constituted by an increasing number of smaller TRs. Therefore the TAs, while composed by several TRs related to heterogeneous network paths, are ordered by the number $|\tau_J|$ of TRs constituting them. To formalize the problem, and to introduce the notation that will be used also to present the results, let us define:

- Class K: the number of TA in which we split the trace;
- Slot J: a specific TA of class K, namely $\tau_J(K)$, $J \in [1,K]$;
- Weight $\hat{w}_J(K)$: the weight of slot J of class K;

- Target Weight $\hat{w}(K) = 1/K$: the ideal portion of the traffic that should be present in each TA at class K.

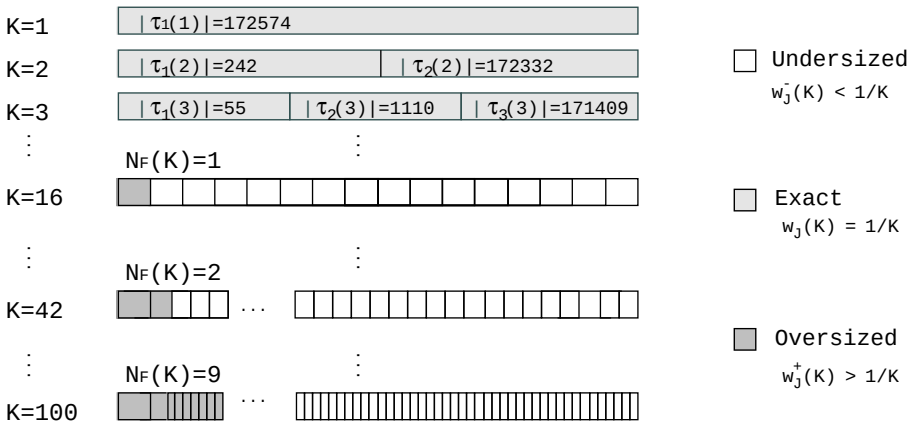


Figure 2.4. Trace Partitioning: Algorithmic Behavior

Fig. 2.4 sketches the splitting procedure. When considering class $K = 1$ we have a single TA of weight $\hat{w}(K) = 1$, derived by aggregating all the TRs, which corresponds to the original trace. Considering $K = 2$, we have two TAs, namely $\tau_1(2)$ and $\tau_2(2)$; the former is build by 242 TRs, which account to $\hat{w}(K) = 1/2$ of relative traffic, while the latter contains all the remaining TRs. This procedure can be repeated for increasing values of K, until the weight of a single traffic relation becomes larger than the target weight. Being impossible to split a single TR into smaller aggregates, we are forced to consider TAs having a weight $\hat{w}_J^+(K) > \hat{w}(K)$.

The weight $\hat{w}(K)$ has therefore to be interpreted as an *ideal* target, in the sense that it is possible that one or more TRs will have a weight larger than $\hat{w}(K)$, as the number of slots grows. In such cases, there will be a $N(K)$ number of *fixed* slots, i.e., TAs constituted by a single TR, of weight $\hat{w}_i^+(K) > 1/K$; the remaining weight will be distributed over the $K - N(K)$ non-fixed slots; therefore the definition of $\hat{w}_J(K)$ is:

$$\hat{w}_J(K) = \begin{cases} \hat{w}_J^+(K) > \dfrac{1}{K}, & 0<J\leq N_F(K) \\ \dfrac{1 - \sum_{i=1}^{N_F(K)} \hat{w}_i^+(K)}{K - N_F(K)} < \dfrac{1}{K}, & N_F(K)<J\leq K \end{cases}$$

In the dataset considered in this paper, for example, the TR τ_D shown in Fig. 2.2 is the largest of the whole trace, having $\hat{w}_D = 0.062 > 1/16$. Therefore, from class $K = 16$ on, the slot J=1 will be always occupied by this aggregate, i.e., $\tau_1(K) = \tau_D, \forall K \geq 16$, as evidenced in Fig. 2.4.

2.2.3 Trace Partitioning Model and Algorithm

More formally, the problem can be conduced to a well known optimization problem $P\|C_{max}$ of job scheduling over identical parallel machines [GKZ93], which is known to be strongly NP-hard. Traffic relations TR are the jobs that have to be scheduled on a fixed number K of machines (i.e., TA) minimizing the maximum completion time (i.e., the TA weight).

The previously introduced ideal target $\hat{w}_K = 1/K$ is the optimum solution in the case of *preemptive* scheduling. Since we preserve the TR identities, preemption is not allowed; however, it is straightforward that minimizing the maximum deviation of the completion time from $w(K)$ is equivalent to the objective function that minimize the maximum completion time.

We define $\underline{R}$ as the $1 \times |\tau_1(1)|$ vector of the jobs length (i.e., TR weights $\hat{w}_i(c,s)$) and $\underline{A}$ as the $1 \times K$ vector of the machine completion times (i.e., TA weights $\hat{w}_J$). Stating with $\underline{M}$ the mapping matrix (i.e., $\underline{M}_{ij} = 1$ means that the i-th job is assigned to the j-th machine), and stating with $\underline{M}_i$ its i-th column, we have:

$$
\begin{aligned}
&min\ p \\
&p \geq z_i, \quad \forall i \in [1,K] \subset \mathbb{N} \\
&\text{s.t.} \begin{cases} \sum_j \underline{M}_{ij} = 1, & \forall i \\ z_i \geq \underline{M}_i \cdot \underline{R} - A_i, & \forall i \\ z_i \geq A_i - \underline{M}_i \cdot \underline{R}, & \forall i \end{cases}
\end{aligned}
$$

The greedy adopted solution, which has the advantage over, e.g., an LPT[GKZ93] solution of the clustering properties earlier discussed, implies the preliminary byte-wise sorting of the traffic relations, and three simple rules:

- allow a machine load to exceed $1/K$ if the machine has no previously scheduled job;
- keep scheduling the biggest unscheduled job into the same machine while the load is still below $1/K$;
- remove the scheduled job from the unscheduled jobs list as soon as job has been scheduled.

A representation of the algorithm output applied to the TA creation problem is provided in Fig. 2.5; the x-axis represents the number of TR within TA and the y-axis represents the weight of each generated TA, i.e, $\hat{w}_J(K)$. For the ease of the read, results for classes $K \in \{10,50,100\}$ are highlighted, whereas neither classes $K < 3$ nor fixed slots are reported in the picture. We notice that, given a class K, the number $|\tau_J|$ of TRs inside the J-th slot increases as J increases. For example, the last slot (the one on the rightmost part of the plot) exhibits always a number of TRs larger than 100,000. On the

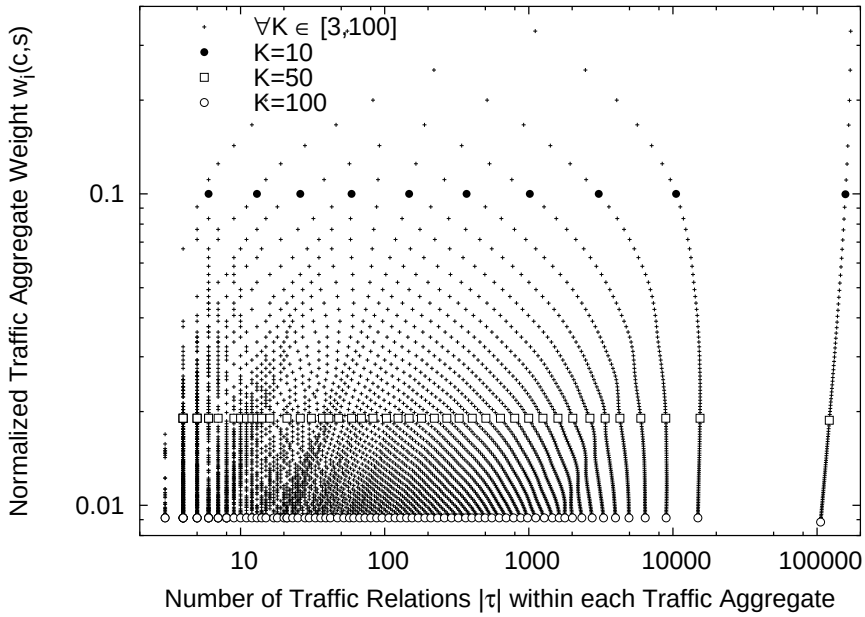


Figure 2.5. Trace Partitioning: Samples for Different Aggregated Classes K

contrary, looking at the first classes, we observe that the number of TRs tends to decrease for increasing K, showing the "packing" enforced by the selected algorithm.

In this section we investigate the most interesting properties of the artificially built traffic aggregates; the analysis will be conduced in terms of, mainly, aggregated size and TCP flow inter-arrival times, coupling their study with the knowledge of the underlying layers – i.e., traffic relation and flow levels.

To help the presentation and discussion of the measurement results, we first propose a visual representation, alternative to the one of Fig. 2.5, of the dataset obtained by applying the splitting algorithm. Indeed, the aggregation process induces a non-linear sampling of both the number $|\tau_J(K)|$ of TRs within each TA and the TA weight $\hat{w}_J(K)$, complicating the interpretation of the results. However, the same qualitative information can be immediately gathered if we plot data as a function of the class K and slot J indexes, using besides different gray-scale intensities to represent the measured quantity we are interested in. As a first example of the new representation, Fig. 2.6 depicts the number $|\tau_J(K)|$ of the traffic relations mapped into each traffic aggregate $\tau_J(K)$. Looking at the plot and choosing a particular class K, every point of the vertical line represents therefore the number of TRs within each of the K possible TA – obtained by partitioning the trace into the K TAs having approximatively a $1/K$ weight. Given the partitioning algorithm used, it is straight-forward to understand the reason why the higher is the slot considered, the larger is the number of TRs within the same TA, as the gray gradient

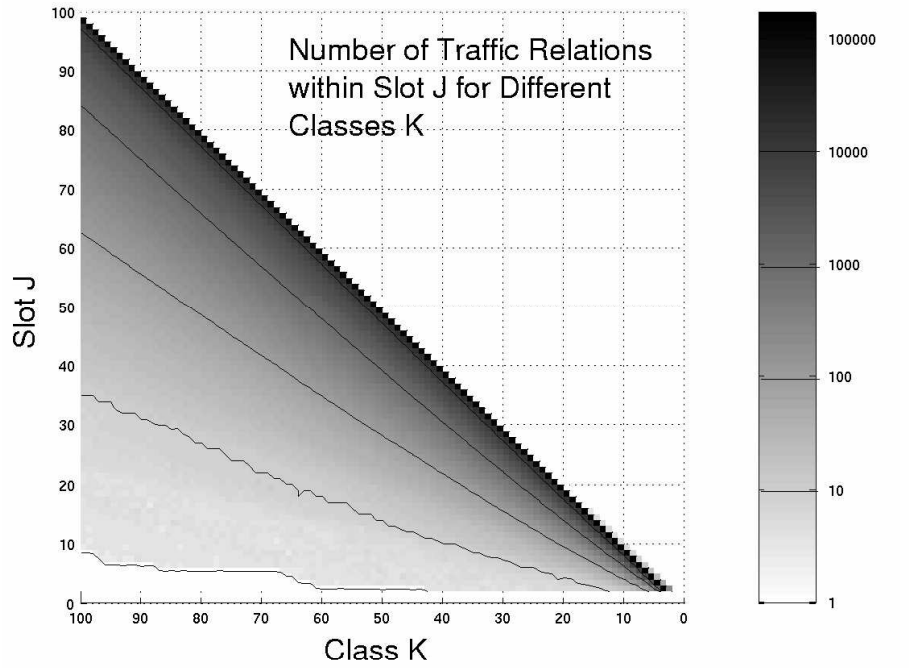


Figure 2.6. Number of Traffic Relations $|\tau_J(K)|$ within each Traffic Aggregate

clearly shows. To better appreciate this partitioning effect, contour lines are shown for $|\tau_J(K)| \in \{1{,}10{,}100{,}1000{,}10000\}$ as reference values; it can be gathered that the bottom white-colored zone of the plot is constituted by *fixed* slots (i.e., $|\tau| = 1$), whereas the slot $J = K$ has always $|\tau_K(K)| > 100{,}000, \forall K$, as already observed from Fig. 2.5. This further confirms the validity of our simple heuristic in providing the previously described aggregation properties.

2.2.4 Traffi c Aggregate Byte-wise Properties

Having showed that the number of TR within the generated TA spread smoothly over a very wide range for any TA weight, we now investigate if these effects are reflected also by the number of TCP flows within each TA, $\Sigma_{(c,s)\in\tau_J(K)}|\mathcal{T}(c,s)|$, shown in Fig. 2.7. Quite surprisingly, we observe that also the number of TCP connections within TA shows an almost similar spreading behavior: the larger the number of TRs within a TA, the larger the number of TCP flows within the same TA. Indeed, the smoothed upper part of the plot (i.e., roughly, slots $J > K/2$) is represent by TAs with a large number of TCP flows, larger than about 1,000; instead, TAs composed by few TRs contain a much smaller number of TCP connections: that is, the number of TCP flows keeps relatively small, while not as regular as in the previous case. Indeed, it must be pointed out that there are exception to

this trend, as shown by the "darker" diagonal lines – e.g., those ending in slot $J = 25$ or $J = 54$, considering class $K = 100$. Probably, within these TAs there is one (or possibly more) TR which is built by a large number of short TCP flows.

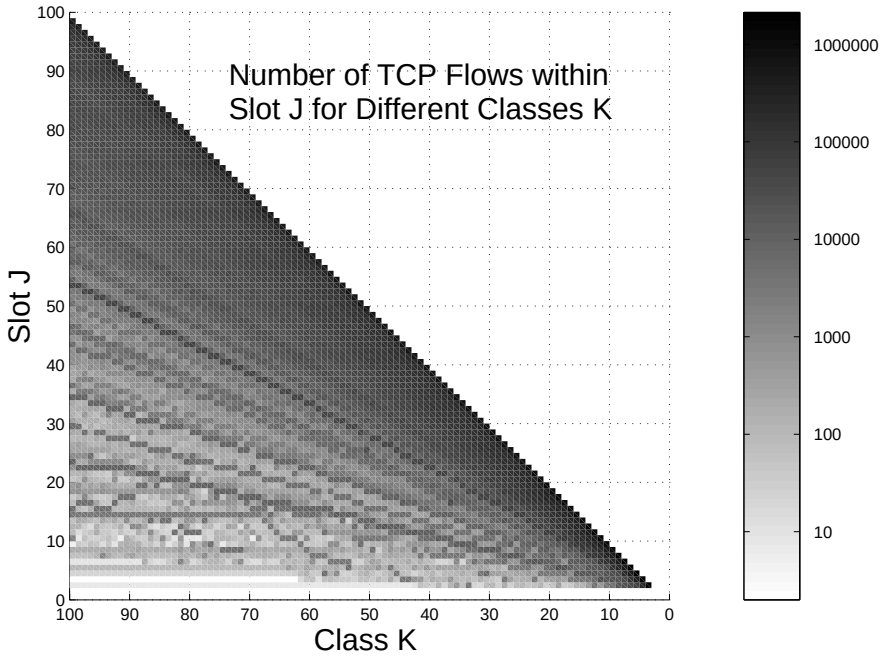


Figure 2.7. Number of TCP Flows within each Traffic Aggregate

Coupling this result with the byte-wise constraint imposed on TAs within the same class, we can state that the bottom region is constituted by a small number of flows accounting for the same traffic volume generated by a huge number of lighter flows. This intuition is also confirmed by Fig. 2.8, which shows the mean TCP flow size in the different aggregates; the higher is the slot J considered, the larger are both the TRs and TCP flows number, the shorter are the TCP flows.

While this might be surprising at first, the intuition behind this clustering is that the largest TRs are built by heavier TCP-connection than the smaller TRs, i.e., TCP-elephants play a big role also in defining the TR weight. Therefore the splitting algorithm, packing together larger TRs, tends also to pack together TCP-elephants.

The TCP flow size variance, not shown to avoid cluttering the figures, further confirms the clustering of TCP elephant flows in the bottom side of the plot. To better highlight this trend, we adopted a threshold criterion: the mean values of the TCP flow size are compared against fixed quantization thresholds, which allows to better appreciate the mean flow size distribution in TAs. The results for threshold values set to 100,250,500,1000 kB

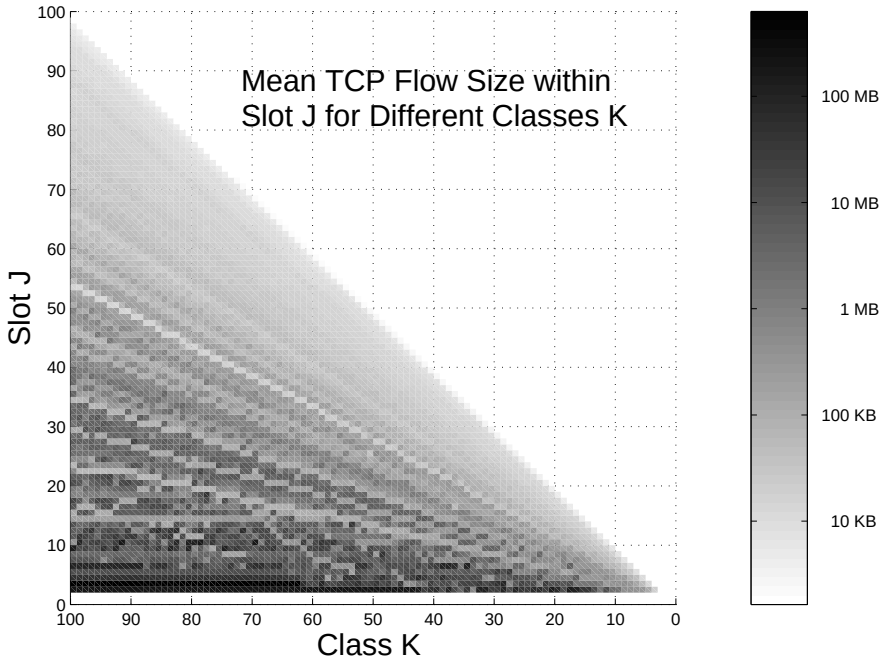


Figure 2.8. Mean Size of TCP Flows within each Traffic Aggregate

are shown in Fig. 2.9, where higher threshold values correspond to darker colors. The resulting plot underlines that the mean flow size grows toward TA constituted by a smaller number of larger TR, which are in their turn, constituted by a small number of TCP-elephants. Similarly, the higher-order slots are mostly constituted by mice. However, the gained result do not exclude the presence of TCP-mice in the bottom TAs, nor the presence of TCP-elephants in the top TAs; therefore, to further ensure that the clustering property of TCP-elephants in the first slots holds, we investigated how the TCP flow size distributions in the different aggregates vary as a function of the TA number K. We thus plot in Fig. 2.10 the empirical flow size distribution for each TAs, showing only the results of class $K = 3$ for the ease of the read. As expected, i) TCP-elephants are evidently concentrates in lower slots, as shown by the heavier tail of the distribution; ii) the TCP-elephants presence decreases as the slot increases, i.e., when moving toward higher slots. The complementary cumulative distribution function $P\{X > x\}$, shown in the inset of Fig. 2.10, clearly confirms this trend.

This finally confirms that the adopted aggregation criterion induces a division of TCP-elephants and TCP-mice into different TAs: TRs containing the highest number of TCP-elephants tends to be packed in the first TA. Therefore in the following, we use for convenience the terms TA-mice and TA-elephants to indicate TA constituted (mostly) by

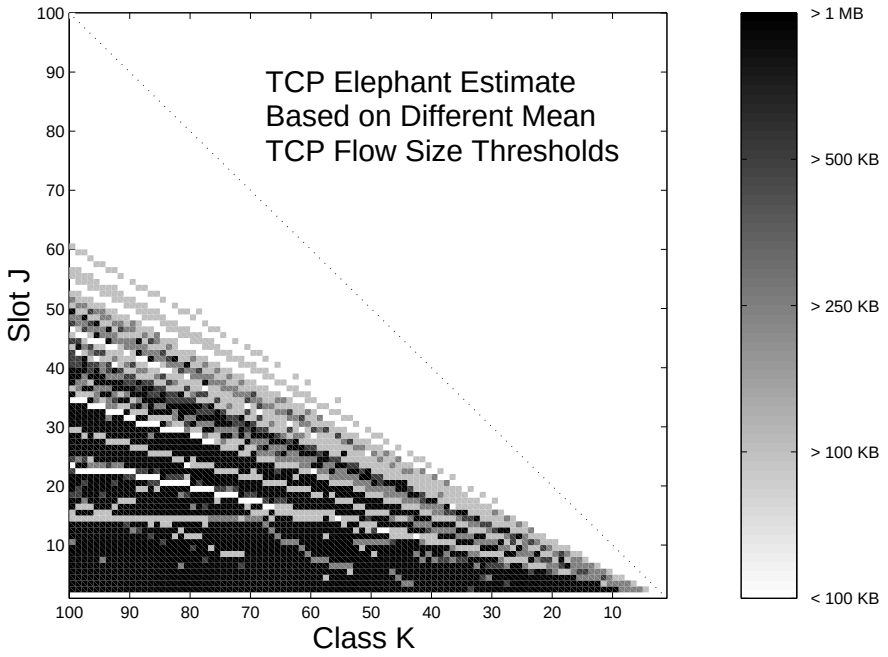


Figure 2.9. Elephant TCP Flows within each Traffic Aggregate

TCP-mice and TCP-elephants flows respectively.

2.2.5 Inspecting TCP Inter-arrival Time Properties within TAs

The result gained in the previous section has clearly important consequences when studying the TCP flow arrival process of different aggregates.

Let us first consider the mean inter-arrival time of TCP flows within each TA, shown in Fig. 2.12. Intuitively, TA-mice have a large number of both TR and TCP flows, and therefore TCP flow mean inter-arrival time is fairly small (less than 100ms). This is no longer true for TA-elephants, since a smaller number of flows has to carry the same amount of data over the same temporal window. Therefore, the mean inter-arrival time is much larger (up to hours).

We recognize, however, that a possible problem might arise, affecting the statistical quality of the results: the high inter-arrival time may be due to non-stationarity in the TA-elephants traffic, where TCP flows may be separated by long silence gap. This effect becomes more visible as long as the class index K and so the aggregation level $|\tau_J(K)|$ decreases. We try to underline whether the presented results are affected by this problem in the remaining part of the analysis.

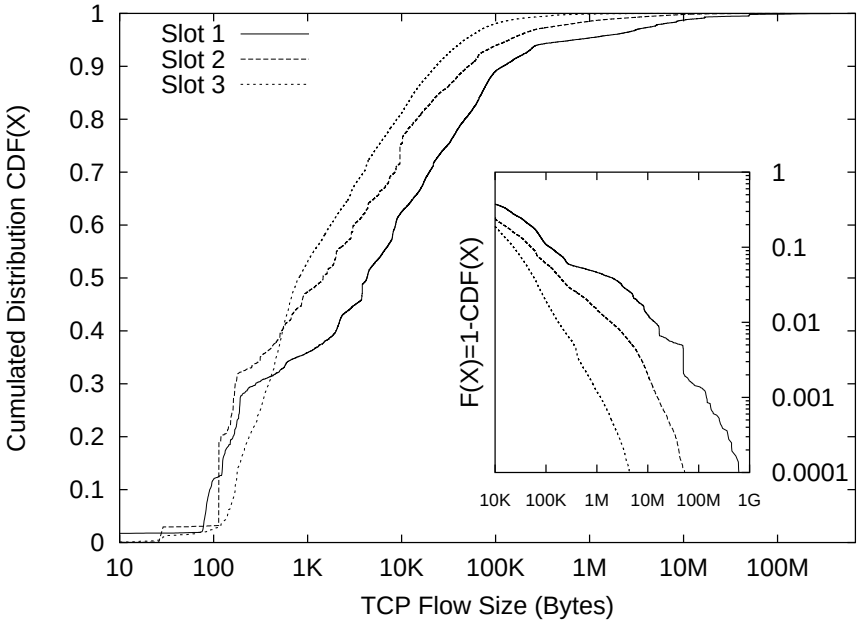


Figure 2.10. TCP Flows Size Distribution of TAs (Class $K = 100$)

The previous assertion is also confirmed observing Fig. 2.12, which shows, within each TA, the plot of TCP flow inter-arrival time variance. It can be noticed that the TCP flow inter-arrival time variance is several orders of magnitude smaller considering TA-mice with respect to the TA-elephants.

Let us now consider the Hurst parameter (see mathematical appendix) $h(J,K)$ measured considering the inter-arrival time of TCP flows within TAs. For each TA, we performed the calculation of $h(J,K)$ using the wavelet-based approach developed in [AFTV00] (see also the mathematical appendix). We adopted the tools developed there and usually referred to as the *AV* estimator. Other approaches can be pursued to analyze traffic traces, but the wavelet framework has emerged as one of the best estimators, as it offers a very versatile environment, as well as fast and efficient algorithms.

The results are shown in Fig. 2.13, which clearly shows that the Hurst parameter tends to decrease for increasing slot, i.e., the TA-mice show $h(J,K)$ smaller than TA-elephants. Recalling the problem of the stationarity of the dataset obtained by the splitting algorithm, not all the $h(J,K)$ are significant; in particular, the one whose confidence interval is too large or the series is not stationary enough to give correct estimations were discarded, and not reported in the plot. Still, the increase in the Hurst parameter is visible for TA-elephants.

To better show this property, Fig. 2.14 presents detailed plots of $h(J,K)$ for $K \in \{10,50,100\}$ (on the top-left, top-right, bottom-left plots respectively) and for all the slots

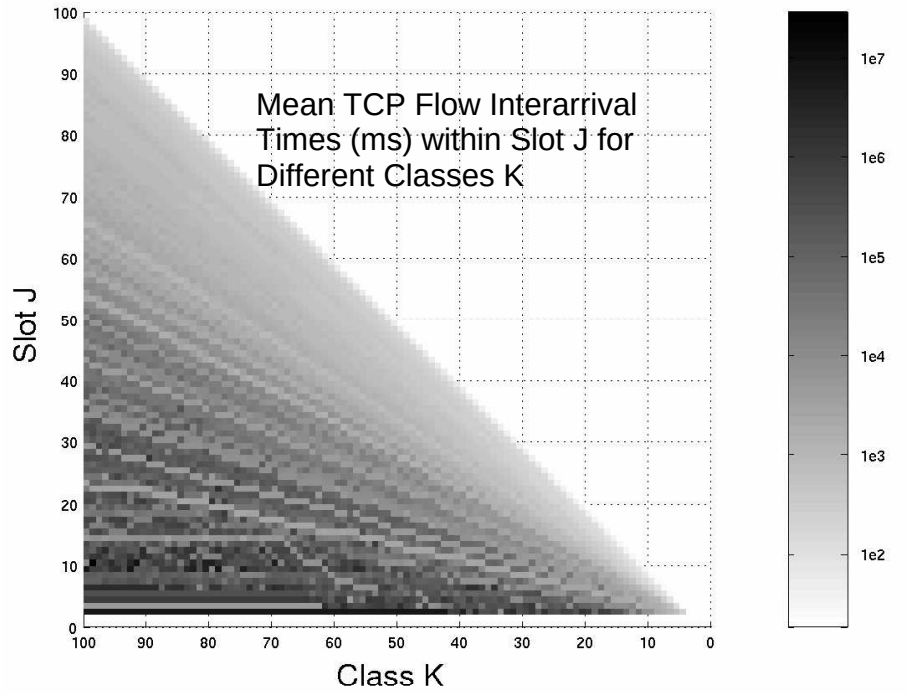


Figure 2.11. Inter-arrival Time Mean of TCP Flows within TAs

J of each class. It can be observed that the Hurst parameter always tends to decrease when considering the TA-mice slots, while it becomes unreliable for TA with few TRs, as testified by the larger confidence intervals. This is particularly visible when considering the $K = 100$ class. In the bottom-right plot, finally, we report a detail of the decaying feature of the $h(J,K)$ value for large J. In the x-axis, the J/K value is used, so that to allow a direct comparison among the three different classes. Notice that the last class, the one composed by many, small TRs which are aggregation of small TCP flows, always exhibits the same Hurst parameter.

Therefore, the most important observation, trustable due to good confidence interval, is that we are authorized to say that TA-mice behavior is driven by TCP-mice. Similarly, TA-elephants are driven by TCP-elephants. Moreover, the inter-arrival process dynamic in TA-elephants and TA-mice are completely different in nature because light TRs tend to contain a relatively small amount of data carried over many small TCP flows, which do not clearly exhibit LRD properties. On the contrary, TCP-elephants seem to introduce a more clear LRD effects in the inter-arrival time of flows within TA-elephants.

A possible justification of this effect might reside in the different behavior the users have when generating connections: indeed, when considering that TCP-mice are typically of Web browsing, the correlation generated by Web sessions tends to vanish when a large

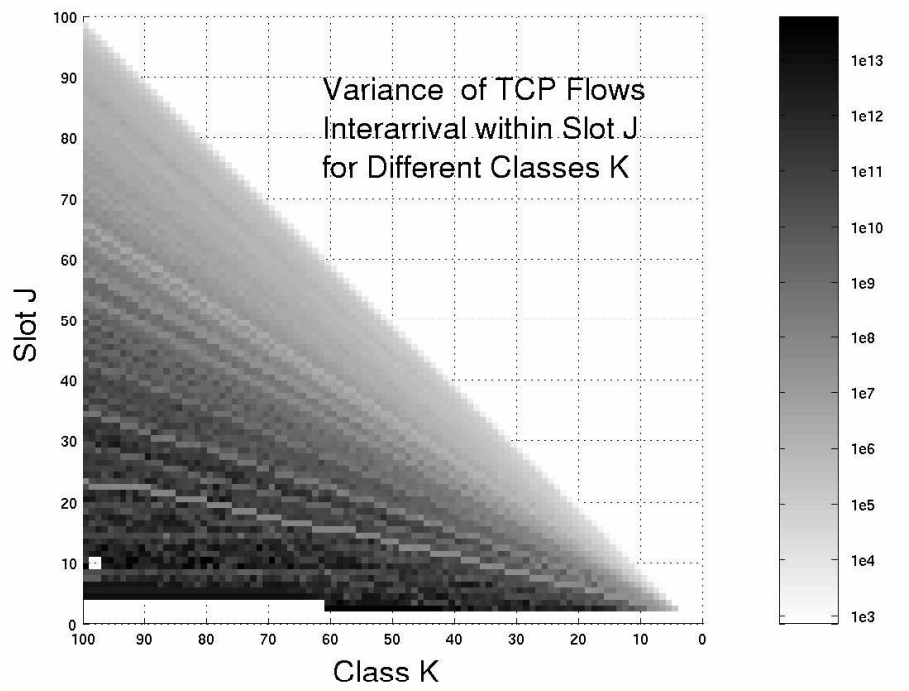


Figure 2.12. Inter-arrival Time Variance of TCP Flows within TAs

number of (small) TRs are aggregated together. On the other side, the TCP-elephants, which are rare but not negligible, seem to be generated with a higher degree of correlation so that i) TRs are larger, ii) when aggregating them, the number of users is still small.

For example, consider the different behavior of a user which is downloading large files, e.g., MP3s; one can suppose that the user starts downloading the first file, then – immediately after having finished the download– he starts downloading a second one, and so on until, e.g., the entire LP has been downloaded. The effect on the TCP flow arrival time is similar to a ON-OFF source behavior, whose ON period is heavy-tailed and vaguely equal to the file download period, which turns out to follow an heavy tailed distribution; besides, we do not care about the OFF period. This is clearly one of the possible causes of LRD properties at the flow level: see, for an example, the τ_D aggregate in Fig. 2.2.

Besides, consider again the heavy-tailed distribution of the TCP flow number within TRs, shown earlier in Fig. 2.3. If we further consider TRs as a superset of web sessions, we gather the same result stated in [FGWK98], that is, the $M/G/\infty$ effect with infinite variance service time distribution.

Finally, TAs tends to aggregate several TRs, separating short term correlated connections (TA-mice) from low-rate ON-OFF connections with infinite variance ON time

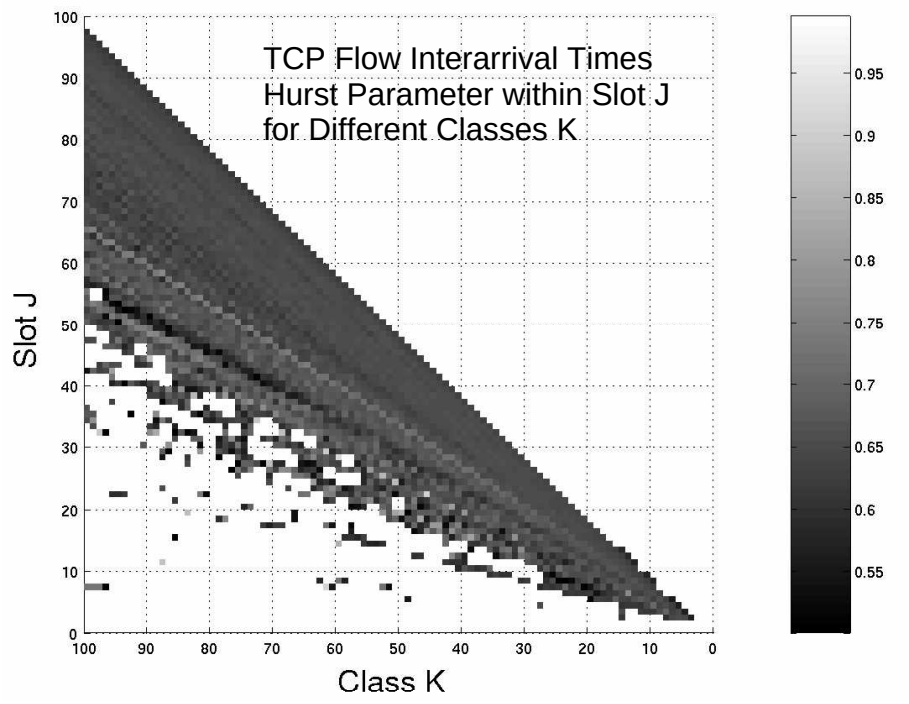


Figure 2.13. Inter-arrival Time Hurst Parameter of TCP Flows within TAs

(TA-elephants). This clearly recall to the well know phenomena described in [WTSW97]; that is, the superposition of ON-OFF sources (as well as "packet trains") exhibiting the Noah effect (infinite variance in the ON or OFF period) produces the so called "Joseph effect": the resulting aggregated process exhibits self-similarity and in particular LRD properties.

In this paper we have studied the TCP flow arrival process, starting from the aggregated measurement taken from our campus network; specifically, we performed a live collection of data directly at the TCP flow level, neglecting therefore the underlying IP-packet level. Trace were both obtained and post-processed through software tools developed by our research group, publicly available to the community as open sources.

We focused our attention beyond the TCP level, defining two layered high-level traffic entities. At a first level beyond TCP, we identified traffic relations, which are constituted by TCP flows with the same path properties. At the highest level, we considered traffic relation aggregates having homogeneous byte-wise weight; most important, the followed approach enabled us to divide the whole traffic into aggregates mainly made of either TCP-elephants or TCP-mice.

This permitted to gain some interesting insights on the TCP flow arrival process. First, we have observed, as already known, that long range dependence at TCP level can be

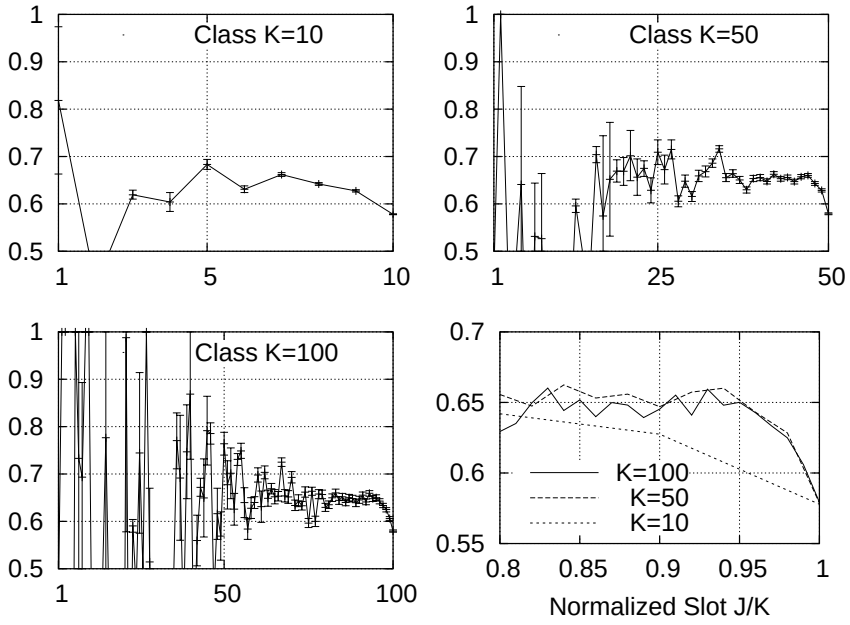


Figure 2.14. Inter-arrival Time Hurst Parameter of TCP Flows within TAs

caused from the fact that the number of flows within a traffic aggregate is heavy-tailed. In addition, the traffic aggregate properties allowed us to see that TCP-elephants aggregates behave like ON-OFF sources characterized by an heavy-tailed activity period. Besides, we were able to observe that LRD at TCP level vanishes for TCP-mice aggregates: this strongly suggests that even ON-OFF behavior is responsible of the LRD at TCP level.

2.3 Measuring TCP Anomalies

Starting from [JID+03], where a simple but efficient classification algorithm for out-of-sequence TCP segments is presented, our work aims at identifying and analyzing a larger subset of phenomena, including, e.g., network duplicates, unneeded retransmissions and flow control mechanisms triggered or suffered by TCP flows. The proposed classification technique has been applied to a set of real traces collected at different measurement points in the network. Results on both the network core and at the network edge show that the proposed classification allows to inspect a plethora of interesting phenomena: e.g., the limited impact of the daily load variation on the occurrence of anomalous events, the surprisingly large amount of network reordering, the correlation among different phenomena, to name a few.

2.3.1 Methodology

The methodology adopted in this paper furthers the approach followed in [JID+03], where authors presented a simple classification algorithm of out-of-sequence TCP packets and applied it to analyze the Sprint backbone traffic; the algorithm was able to discriminate among out-of-sequence segments due to i) necessary or unnecessary packet retransmissions by the TCP sender, ii) network duplicates, or iii) network reordering. Similarly, rather than using active probe traffic, we adopt a passive measurement technique and, building over the same idea, we complete the classification rule to include other event types. Besides, we not only distinguish among the other possible *kinds* of out-of-sequence or duplicate packets, but we also focus on the *cause* that actually triggered the segment retransmission by the sender.

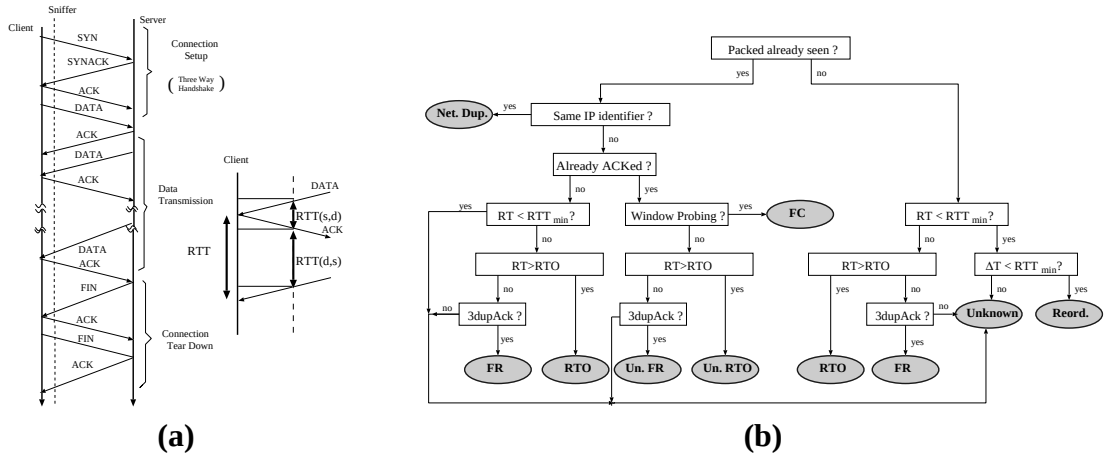


Figure 2.15. Evolution of a TCP flow and the *RTT* Estimation Process **(a)** and Anomalous Events Classification Flow-Chart **(b)**

As previously mentioned, we record packets in both directions of a TCP flow: thus, both data segments and ACKs are analyzed; moreover, both the IP and TCP layers are exposed, so that the TCP sender status can be tracked. Figure 2.15a) sketches the evolution of a TCP flow: connection setup, data transfer, and tear-down phases are highlighted. It is worth to point out that the measurement point (or sniffer) can be located anywhere in the path between the client and the server; furthermore, as shown in the picture, the sniffer can be close to the client when measurements are taken at the network edge (as it happens, e.g., on our campus LAN), whereas this bias vanishes when the sniffer is deployed on the core of the network (as it happens considering backbone traces). Since we rely on the observation of both DATA and ACK segments to track TCP flow evolution, a problem may arise in case asymmetric routing forces DATA and ACK segment to follow

two different path. In such cases, we ignore the "half" flows.

A TCP flow starts when the first SYN from the client is observed, and ends after either the tear-down sequence (the FIN/ACK or RST messages), or when no segment is observed for an amount of time larger then a given threshold[2]. By tracking both flow directions, the sniffer correlates the sequence number of TCP data segments to the ACK number of backward receiver acknowledgments and classifies the segments as:

- *In-sequence*: if the sender initial sequence number of the current segment corresponds to the expected one;
- *Duplicate*: if the data carried by the segment have already been observed before;
- *Out-of-sequence*: if the sender initial sequence number is not the expected one, and the data carried by the segment have never been observed before.

The last two entries are symptomatic of some *anomaly* occurred during the data transfer: to discriminate such anomalous events further, we devise a fine-grained heuristic classification and implement it in Tstat [Mel05], which we then use to analyze the collected data. The decision process followed to classify such anomalous events makes use of the following variables:

- $RTT_{\min}$: the Minimum *RTT* estimated since the flow started;
- *RT*: the Recovery Time is the time elapsed between the time the current anomalous segment has been observed and the time the segment with the *largest* sequence number has been received;
- ΔT: the Inverted-packet gap is the difference between the observation time of the current anomalous segment and of the previously received segment;
- *RTO*: the sender Retransmission Timer value, estimated according to [PA00];
- $DupAck$: the number of duplicate ACKs observed on the reverse path.

2.3.2 Anomalies Classification Heuristic

Following the flow diagram of Figure 2.15b, we describe the classification heuristic. Given an anomalous segment, the process initially checks if the segment has already been seen by comparing its TCP sequence number with those carried by segments observed so far: thus, the current segment can be classified as either duplicate or out-of-sequence. In the first case, following the left branch of the decision process, the IP identifier field

[2]The tear-down sequence of flows under analysis is possibly *never* observed: to avoid memory starvation we use a 15-minutes timer, which is sufficiently large [IDG+01] to avoid discarding on-going flows.

of the current packet is compared with the same field of the original packet: in case of equality, then the anomalous packet is classified as **Network Duplicate** (Net. Dup.). Network duplicates may stem from malfunctioning apparatuses, mis-configured networks (e.g., Ethernet LANs with diameter larger than the collision domain size), or, finally, by unnecessary retransmissions at the link layer (e.g., when a MAC layer ACK is lost in a Wireless LAN). Differently from [JID+03], we classify as network duplicate all packets with the same IP identifier *regardless* of ΔT: indeed, there is no reason to exclude that a network duplicate may be observed at any time, and there is no relation between the *RTT* and the time a network can produce some duplicate packets.

When the IP identifiers are different, the TCP sender may have performed a retransmission. If all the bytes carried by the segment have already been acknowledged, then the segment has successfully reached the receiver, and therefore this is an unneeded retransmission, which has to be discriminated further. Indeed, the *window probing* TCP flow control mechanism performs "false" retransmissions: this is done to force the immediate transmission of an ACK so as to probe if the receiver window *RWND*, which was announced to be zero on a previous ACK, is now larger than zero. Therefore we classify as retransmission due to **Flow Control** (FC) the segments for which the following three conditions hold: i) the sequence number is equal to the expected sequence number decreased by one, ii) the segment payload size is of zero length, and iii) the last announced *RWND* in the reverse ACK flow was equal to zero. This is a new possible cause of unneeded retransmissions which was previously neglected in [JID+03].

Otherwise, the unnecessary retransmission, could have been triggered because either a Retransmission Timer (*RTO*) has fired, or the fast retransmit mechanism has been triggered (i.e., when three or more duplicate ACKs have been received for the segment preceding the retransmitted one). We identify three situations: i) if the recovery time is larger than the retransmission timer ($RT > RTO$), the segment is classified as an **Unneeded Retransmission by *RTO*** (Un. *RTO*); ii) if 3 duplicate ACKs have been observed, the segment is classified as an **Unneeded Retransmission by Fast Retransmit** (Un. FR); iii) otherwise, if none of the previous conditions holds, we do not know how to classify this segment and we are forced to label it as **Unknown** (Unk.). Unneeded retransmissions may be due to a misbehaving source, a wrong estimation of the *RTO* at the sender side, or, finally, to ACK loss on the reverse path; however, distinguishing among these causes is impossible by means of passive measurements.

Let us now consider the case of segments that have already been seen but have not been ACKed yet: this is possibly the case of a retransmission following a packet loss. Indeed, given the recovery mechanism adopted by TCP, a retransmission can occur only after at least a *RTT*, since duplicate ACKs have to traverse the reverse path and trigger the Fast Retransmit mechanism. When the recovery time is smaller than $RTT_{\min}$, then the anomalous segment can only be classified as **Unknown**[3]; otherwise, it is possible

[3]In [JID+03] authors use the *average RTT*; however, being each *RTT* possibly different than the average

to distinguish between **Retransmission by Fast Retransmit** (FR) and **Retransmission by *RTO*** (*RTO*) adopting the same criteria previously used for unneeded retransmissions. Retransmissions of already observed segments may be due to data segments loss on the path from the measurement point to the receiver, and to ACK segments delayed or lost before the measurement point.

Eventually, let us consider the right branch of the decision process, which refers to out-of-sequence anomalous segments. In this case, the classification criterion is simpler: indeed, out-of-sequence can be caused either by the retransmission of lost segments, or by network reordering. Since retransmissions can only occur if the recovery time *RT* is larger than $RTT_{\min}$, by double checking the number of observed duplicate ACKs and by comparing the recovery time with the estimated *RTO*, we can distinguish retransmissions triggered by *RTO*, by FR or we can classify the segment as an unknown anomaly. On the contrary, if *RT* is smaller than $RTT_{\min}$, then a **Network Reordering** (Reord) is identified if the inverted-packet gap ΔT is smaller than $RTT_{\min}$. Network reordering can be due to either load balancing on parallel paths, or to route changes, or to parallel switching architectures which do not ensure in-sequence delivery of packets [BPS99].

2.3.3 $RTT_{\min}$ and *RTO* Estimation

The algorithm described so far needs to estimate some thresholds from the packet trace itself, whose values may not be very accurate, or even valid, when classifying the anomalous event. Indeed, all the measurements related to the *RTT* estimation are particularly critical, since they are used to determine the $RTT_{\min}$ and the *RTO* estimates. *RTT* measurement is updated during the flow evolution according to the moving average estimator standardized in [PA00]: given a new measurement m of the *RTT*, we update the estimate of the average *RTT* by mean of a low pass filter $E[RTT]_{new} = (1-\alpha)E[RTT]_{old} + \alpha\ m$ where $\alpha \in (0,1)$ is equal to $1/8$.

Since the measurement point is co-located neither at the transmitter nor at the receiver, the measure of *RTT* is not directly available. Therefore, to get an estimate of the *RTT* values, we build over the original proposal of [JID+03]. In particular, denote by $RTT(s,d)$ the *Half path RTT sample*, which represents the delay at the measurement point between an observed data segment flowing from source s to destination d and the corresponding ACK on the reverse path, and denote by $RTT(d,s)$ the delay between the ACK and the following segment, as shown in Figure 2.15a.

From the measurement of $RTT(s,d)$ and $RTT(d,s)$ it is possible to derive an estimate of the total round trip time as $RTT = RTT(s,d)+RTT(d,s)$; given the linearity of the expectation operator $E[\cdot]$, the estimation of the average round trip time $E[RTT] = E[RTT(s,d)] +$

RTT, and in particular *smaller*, we believe that the use of the average *RTT* forces a uselessly larger amount of unknown.

$E[RTT(d,s)]$ is unbiased. Furthermore, the *RTT* standard deviation, std(*RTT*), can be estimated following the same approach, given that in our scenario $RTT(s,d)$ and $RTT(d,s)$ are independent measurements. Finally, given $E[RTT]$ and std(*RTT*), it is possible to estimate both the sender retransmission timer, as in [PA00], and the minimum *RTT*:

$$\begin{aligned} RTO &= \max(1, E[RTT] + 4\text{std}(RTT)) \\ RTT_{\min} &= \min(RTT(s,d)) + \min(RTT(d,s)) \end{aligned}$$

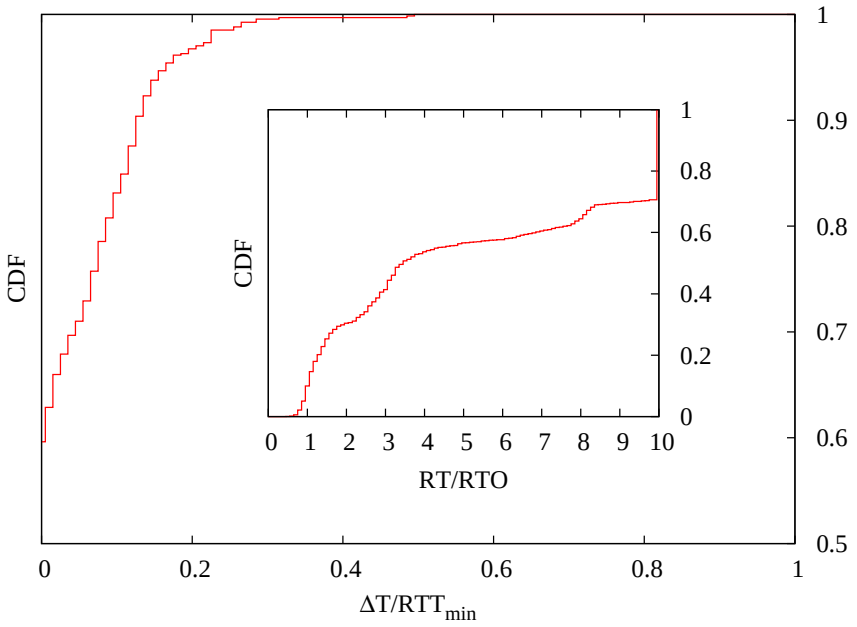


Figure 2.16. CDFs of the ratio between the inverted packet gap ΔT over the estimated $RTT_{\min}$ (outset) and of the ratio between the recovery time *RT* over the actual *RTO* estimation (inset).

In general, since $RTT_{\min} \leq \min(RTT)$, the latter equation gives a conservative estimate of the real minimum *RTT*, leading to a conservative algorithm that increases the number of anomalies classified as unknown rather than risking some mis-classifications.

Besides, we lead a set of measurements to inspect the $RTT_{\min}$ and *RTO* values, in order to double-check the impact of the estimation so far described. The former is involved on the classification of the network reordering anomalies that may occur when identifying two out-of-sequence segments separated by a time gap smaller than $RTT_{\min}$. The outer plot of Figure 2.16 reports the typical[4] Cumulative Distribution Function (CDF) of the ratio between the inverted packet gap ΔT and the value of the $RTT_{\min}$, when considering anomalous events classified as network reordering only. It is easy to gather that ΔT

[4]Since the behavior is qualitatively similar across all the analyzed traces, we report the results referring to a single dataset, namely GARR'05 traces.

is much smaller than the $RTT_{\min}$, which suggests that i) the initial choice of $RTT_{\min}$ is appropriate, and that ii) the conservative estimation of $RTT_{\min}$ does not affect the classification. The inset of Figure 2.16 reports the CDF of the ratio between the actual Recovery Time *RT* and the corresponding estimation of the *RTO* relative to retransmissions by *RTO* events only. The CDF confirms that $RT > RTO$ holds, which is a clear indication that the estimation of the *RTO* is not critical. Moreover, it can be noted that about 50% of the cases have a recovery time which is more than 5 times larger than the estimated *RTO*. This apparently counterintuitive result is due to the *RTO* back-off mechanism implemented in TCP (but not considered by the heuristic), which doubles the *RTO* value at every retransmission of the same segment. Not considering the back-off mechanism during the classification leads to a robust and conservative approach.

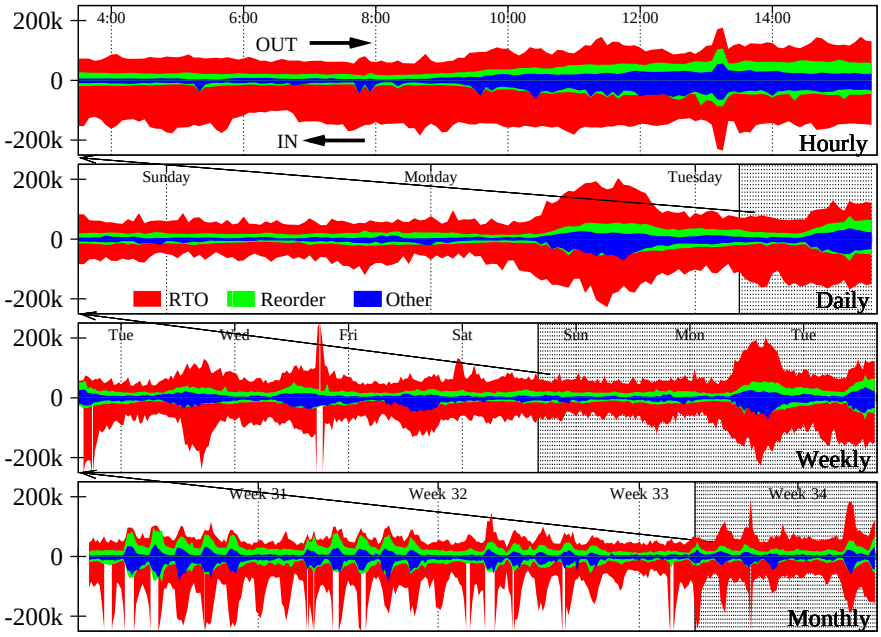


Figure 2.17. Amount of incoming (bottom y-axis) and outgoing (top y-axis) anomalies at different timescales.

2.3.4 Measurement Results

In this section we present results obtained through traffic traces collected from different networks. The first one is a packet-level trace gathered from the Abilene Internet backbone, publicly available on the NLANR Web site [NLA05]. The trace, which is 4 hours long, has been collected on June 1st, 2004 at the OC192c Packet-over-SONET link from Internet2's Indianapolis node toward Kansas City. The second measurement point is located on the GARR backbone network [top05], the nation-wide ISP for research and

Figure 2.18. Anomalies percentage normalized over the total traffic (top) and anomalies breakdown (bottom).

educational centers, that we permanently monitor using Tstat [Mel05]. The monitored link is a OC48 Packet-over-SONET from Milano-1 to Milano-2 nodes, and statistics reported in the following are relative to the month of August 2005. The third measurement point is located at the sole egress router of Politecnico campus LAN, which behaves thus as an Internet stub. It is a OC4 AAL5 ATM link from Politecnico to the GARR POP in Torino. Traces show different *routing symmetry*: while the POLITO and GARR traces reflect almost 100% of traffic symmetry, only 46% of ABILENE traffic is symmetric; this is of particular relevance, since Tstat relies on the observation of both DATA and ACK segments to track TCP flow evolution.

All the gathered flow-level and aggregated statistics, related to anomalous traffic as well as to many other measurements that we do not report here, are publicly available on the Tstat Web page [Mel05]. In the following, we *arbitrarily* use "In" and "Out" tags to discriminate between the traffic directions of GARR and Abilene traces: since the measurement point is located in the backbone, neither an inner nor an outer network regions actually exists.

2.3.5 Statistical Analysis of Backbone Traffi c

Figure 2.17 depicts the time evolution of the volume of anomalous segments classified by the proposed heuristic applied to the GARR traffic; in order to make plots readable, only

the main causes of anomalies, i.e., *RTO* and network reordering, are explicitly reported, whereas the other kinds of anomalies are aggregated. Measurements are evaluated with different granularities over different timescales: each point corresponds to a 5 minute window in the hourly plot, a 30 minute window for both daily and weekly plots and a 2 hour interval in the monthly plot. Both link directions are displayed in a single plot: positive values refer to the "Out" direction, while negative to "In". The plot of the entire month clearly shows a night-and-day trend, which is mainly driven by the link load. The trend is less evident at finer time scales, allowing to identify stationary periods during which perform advanced statistical characterization. The same monthly dataset is used in Figure 2.18, which reports in top plot the volume of anomalies normalized over the total link traffic, and in bottom plot the anomaly breakdown, i.e., the amount of anomalies of each class normalized over the total number of anomalous events. Labels report, for each kind of anomaly, the average occurrence over the whole period obtained aggregating both traffic directions. The amount of *RTO* accounts for the main part of anomalous events, being almost 70%. This is not surprising, since *RTO* events correspond to the most likely reaction of TCP to segment losses: indeed fast retransmit is not very frequent, since only long TCP flows, which represent a small portion of the total traffic, can actually use this feature [MMC05]. Packet reordering is also present in a significant amount. The average total amount of anomalous segments is about 5% considering Out traffic direction, and about 8% considering In direction, with peaks ranging up to 20%.

Besides the precise partitioning of such events, it is interesting to notice that the periodical trend exhibited by the *absolute* amount of anomalies observed early in the monthly plot of Figure 2.17, almost completely vanishes when considering the *normalized* amount shown in Figure 2.18. Thus, while the total amount depends on the traffic load, the percentage of anomalies seems quite independent on the load. This is an interesting finding that clashes with the usual assumption that the probability of anomalies and segment losses increases with load. Very similar results were also obtained for the Abilene trace, confirming that not only the percentage of anomalies over the total amount of traffic, but also the anomaly breakdown remain steadily the same even when the amount of anomalies, over the link load, exhibits large fluctuations. A possible intuition that may be the cause of such counterintuitive result may be due to the greedy nature of TCP, which pushes the instantaneous offered load to 1, therefore producing packet losses. Indeed, even if the average offered load is much smaller during off-peak periods, congestion may arise on bottleneck links (i.e., possibly at the network access) when there are active TCP flows.

Another interesting statistics we can look into is the correlation structure of the time series counting the number of anomalies per time unit. Though this could be presented in many different ways, we decided to use the Hurst parameter H as a very compact index to identify the presence of Long Range Dependence (LRD) in the series. The estimation of H as been carried over whenever it has been possible to analyze stationary time-windows. In particular, this holds true for the two backbone links when considering few hours period, while this was not verified on campus LAN traces, due to much smaller number of

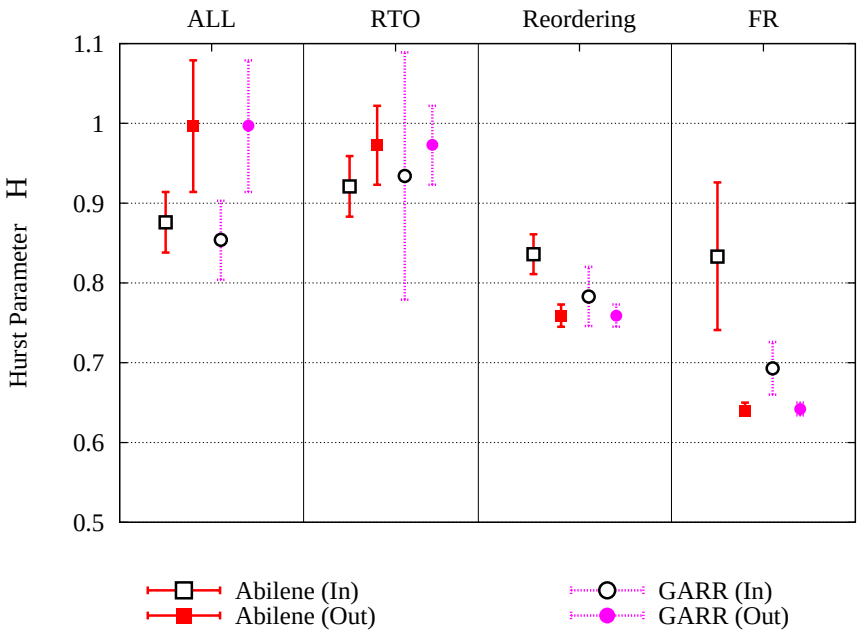


Figure 2.19. Hurst parameter estimate for different anomalies and traces.

events. The estimate of the Hurst parameter H, measured through the wavelet based estimator [VA98], is depicted in Figure 2.19 for different kinds of anomalies, measurement points and traffic directions. Roughly speaking, $H = 0.5$ hints to uncorrelated processes, while increasing values of $H \in [0.5,1]$ are symptomatic of increasing correlation in the process, with values around $H = 0.7$ usually considered typical of LRD. In all cases, H is considerably higher than 0.5, meaning that these series are LRD. moreover, H for Reordering and FR series is smaller than for the *RTO* series, for all traces and traffic directions. Eventually, we stress that by further measuring H in different subsets of the original trace and by using different time units (from 10 ms to a few seconds), we gathered sufficient informations to confirm the presence of LRD in the anomalous traffic.

2.3.6 Analysis of a Single Elephant Connection

Other interesting observations can be drawn by focusing on a specific network path, where it can be assumed that network setup is homogeneous: in particular, we considered the longest TCP flows that has been measured on our campus LAN. The longest "elephant" that we ever observed corresponds to a large file download from a host in Canada which lasted about ten hours (from 9:00am to 8:00pm of Wednesday the 5th of May). The flow throughput varied from 60 KBytes/s up to 160 KBytes/s, while the measured *RTT* varied in the $[300,550]$ ms range. Figure 2.20 plots a set of time series regarding the flow anomalies, where each series corresponds to the occurrence of events of one of the anomalies

identified by the proposed heuristic. First of all, we point out that, while each time series is non-stationary, the aggregated series is stationary. Then, observe that the largest number of anomalies refer to *RTO* and FR, and this holds for the whole flow lifetime. It is worth noticing that typically, while the number of *RTO* increases, the number of FR decreases (and vice-versa): high numbers of *RTO* probably correspond to high levels of congestion, driving the TCP congestion control to shrink the sender congestion window, so that the chance of receiving three dup-ACKs is small. Similar behaviors were observed for other very long elephants not shown here for the sake of brevity.

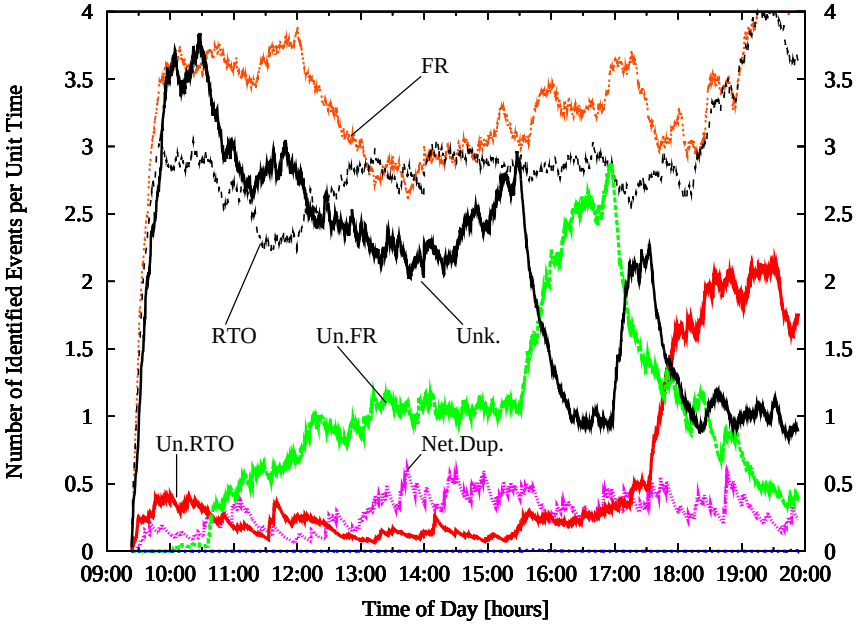


Figure 2.20. Time plot of anomalous events for the selected elephant flow.

No network reordering has been identified during the whole flow lifetime. In our experiments we observed that some flows do not suffer from network reordering at all, while others present several network reordering events: this behavior suggests network reordering to be path dependent, and a similar reasoning applies to network duplicate events.

Some of the anomalies could not be classified other than Unknown. Investigating further the reasons of such misidentification, we found that their majority is due to segments whose recovery time was smaller than the estimated *RTO* (and therefore could not classified as retransmission due to *RTO*firing), but larger than the $RTT_{\min}$ (thus forbidding to identify them as network reordering) and, eventually, the number of duplicate ACKs was smaller than 3 (therefore forbidding to identify them as retransmission due to Fast Retransmit). We suppose that either the sender was triggering FR with a number of Duplicate ACKs smaller than 3, or that the *RTO* value estimated by the sender was smaller

than the *RTO* estimation at the measurement point. Notice again that, as we already mentioned, our classification strictly follows the TCP standards, avoiding mis-classifications, at the price of the increase of unidentified events.

We now focus on the fluctuation of the rate and the statistical properties of the counting process of i) the received packets per time unit and ii) the anomalous events per time unit, when the time unit is 100 m long. We analyze the scaling behavior of the data through the Multi Resolution Analysis (MRA) [VA01] with the code provided in [Vei]; according to MRA, both these processes are stationary: moreover, from our measurements, stationarity usually holds for long flows in general. Figure 2.21 depicts the log-scale diagram for both the considered time series. The log-scale diagram can be seen as a complex way to evaluate the power spectra, but it has good properties once dealing with LRD random signals and in particular it delivers very good estimates of scaling exponents [VA98]. The diagram reports the estimate of the correlation of the considered process: an horizontal line is symptomatic of an uncorrelated process, while linear slopes for large scales have to be read as signs of high correlation. Therefore, from Figure 2.21 we observe that the anomalous events that TCP handles are *not* correlated. On the contrary, as already observed on traffic aggregates [VA98], we found the traffic generated by the elephant flow to be *highly* correlated for the whole flow lifetime. The slope of this curve leads to an estimation of the parameter H about 0.74; the time scale of the *RTT* is visible as the "bump" around 400 ms in the left-hand side of the curve, which represents an intrinsic time periodicity for the flow dynamic related to the RTT.

This is not surprising, since the *RTT* has also been recognized as a natural time cutoff scale i) below which traffic is highly irregular and uncorrelated and ii) above which is long-term correlated. These results confirm similar findings in the literature: as already observed in [FLMT02] and [JD05], TCP exhibits an ON/OFF kind of behavior which is responsible for the burstiness over a limited range of time scales: in other words, TCP generates bursts of traffic clocked with the *RTT*.

2.3.7 Discussion on Anomalies

In this paper, we have defined, identified and studied the TCP anomalies, i.e., segments received either out-of-sequence or duplicate. We have extended and refined the heuristic originally proposed in [JID+03], and applied it the real traffic traces collected at both network edges and core links. By studying the statistical properties of these phenomena, we can state few guidelines to understand the stochastic properties of TCP traffic. First, we observed that, though the absolute amount of anomalies is highly dependent on the link load, both its relative amount over the total traffic and the anomaly breakdown are almost independent from the current load. This is a quite interesting finding that clashes with the usual assumption that the probability of anomalies and segment losses increases with load. As a possible justification to this, we point out the greedy nature of TCP congestion control algorithm, that pushes the instantaneous offered traffic to link capacity,

even though the average link utilization is much smaller, thus causing anomalies to arise.

Second, we have shown that aggregated anomalies at any Internet link are correlated, as clearly shown by the estimation of the Hurst parameter H. Nevertheless, when considering single flows, we observed that the anomalies arrival process is on the contrary uncorrelated, even though the general packet arrival process generated by the same flow is known to be highly correlated, as we experimentally verified.

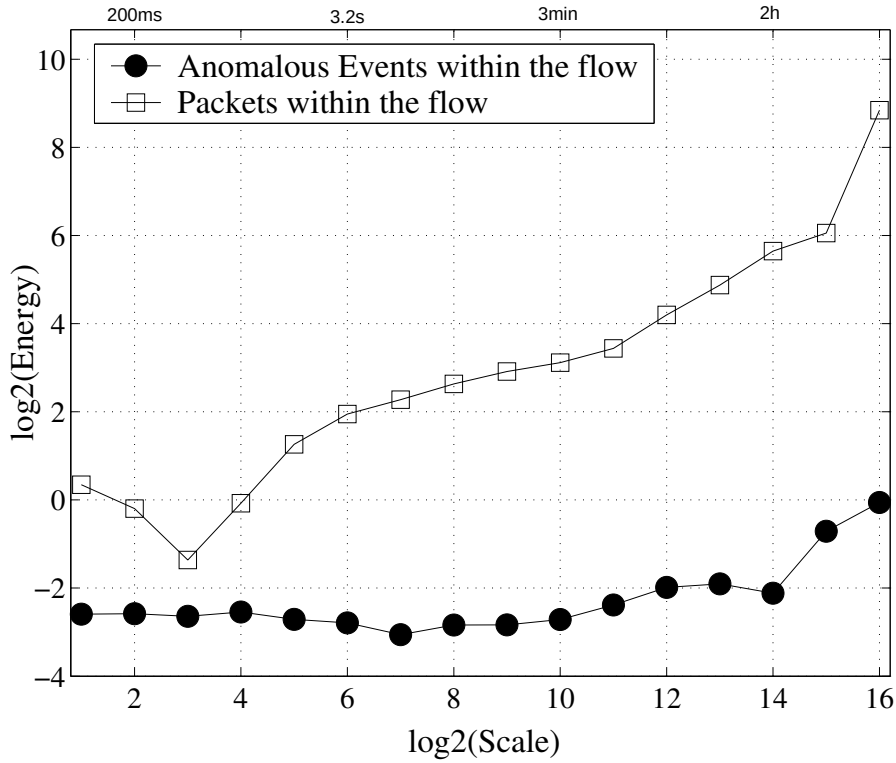


Figure 2.21. Energy plots for the time series of number of segments and number of out of orders within the selected elephant flow.

The correlation exhibited by the aggregated anomalies is intuitively tied to the duration of their causes (i.e., congestion periods, path reordering, etc.) which may last for large timescales. When considering a single flow, the sub-sampling process induced by packet arrivals (i.e., the process observed by flow packets crossing a particular node/link/path), and the congestion control mechanisms adopted by TCP tend to Un-correlate the process of anomalous segments observed by the flow itself. Indeed, recalling the large number of very short flows and the small number of long-lived flows, it is intuitive that only a very small fraction of anomalous segment observed over a single link may be due to a single flow. Thus, we can then conclude that, even though there is high correlation in the aggregate process, its impact on single flows is marginal.

2.4 Flow Traffic Dependence

Analysis of flow level traffic involves characterizing the dependencies between flow sizes, their rates and their durations (delays). Classical queuing models for these include processor sharing (PS) models and infinite server models. PS-models can be used to capture the fundamental impact of bandwidth sharing among competing flows. In the simplest PS-model it is assumed that when there are multiple flows present in the system, each flow obtains its fair share of the total capacity. Variants of this basic model include GPS [Coh79] (fair sharing, but state dependent service rates) and DPS [FMI80] (weighted sharing). These are models for a single link. In a network setting, well known idealized bandwidth sharing notions include max-min fairness [BG92b], proportional fairness [KMT98] and the recently introduced balanced fairness [BP03]. Infinite server models (see, e.g., [MM05]), on the other hand, assume that the flows do not share the bandwidths. However, the bandwidth that each flow obtains may be random, but it does not depend on the number of concurrent flows.

A common property in all above idealized models is that the file size distribution is independent of the rate that the flow obtains. Additionally, the size and the duration are heavily positively dependent, and duration and rate heavily negatively dependent. These simple dependencies are not necessarily true in practice. Firstly, because TCP protocol does not behave according to the idealized models. Depending on the situation, TCP approximates either DPS [BT01], when there are TCP flows with very different RTTs, or fair sharing according to GPS or pure PS [LBMK03, FBP$^+$01], depending on whether flows are peak rate limited or not. Also, TCP has internal congestion control mechanisms (e.g., slow start and AIMD) that influence differently whether the file is long (AIMD dominates) or short (slow start dominates). Infinite server models have been also applied for modeling TCP [AAA$^+$03] and measurements of backbone traffic [MM05]. Secondly, there may be other human behavioral factors influencing the dependencies. Consider for example a user with a slow wireless GPRS connection and a user with a high speed DSL link. It is more probable that the user with the DSL connection will download long files rather than the user with a slow connection.

In this study we perform a rigorous statistical analysis of the flow-level dependencies between flow sizes, rates and their durations. We are particularly interested in verifying upper tail dependencies between the variables, for example to verify whether it is indeed users with high rates that download large files (as intuition would suggest). To analyze this type of tail dependence prior studies [LH03, ZBPS02, HCJP$^+$05] have utilized thresholded correlation analyses (only big or long flows were considered). Moreover, the results in the studies are partly conflicting. However, as shown in this paper, thresholding is not a correct way to analyze such dependencies, as the thresholding approach basically interferes with the correlation structure, and hence the obtained correlation results are influenced by the method itself. Thus, thresholded correlation can not be considered as a robust measure of dependency.

Our contribution is the development of a robust method for analyzing tail dependencies. The method is based on using the theory of copulas, see the mathematical appendix. We apply our method to empirical measurements from different live networks (both fixed IP backbone networks and wireless GPRS/UMTS networks). In these traces, the flows will be also influenced by the behavioral aspects as discussed earlier and by the fact that flows themselves are also heterogeneous (not all TCP flows in real networks correspond to file transfers).

2.4.1 Linear correlation coeffi cient is a wrong method

Linear correlation coefficient ρ, or it's sample version, must not be used for two important reasons, when trying to capture the dependence structure between size S and duration D of a flow from a data trace. The first reason is that they both have very clear high variability character in the sense that heavy-tailed distributions typically fit very well for the upper tail. It is questionable whether ρ is defined even though it's sample version is always finite.

One could then try to make a log-log transform of the data since then the high variability would disappear. However, the size of a flow has intrinsically discrete nature: some flow sizes, even large ones, exist very often while other flow sizes occur hardly ever. We can, and do assume that the flow size distribution can be approximated arbitrarily close by a continuous distribution, but this does not remove the discrete character. The second important reason, the ordinary correlation coefficient must not be used, comes from the fact that durations, especially when calculated from accurate time-stamps, are naturally continuous random variables. The range of the linear correlation coefficient, when calculated between discrete and continuous random variables, cannot have full range $[-1,1]$. Thus, the interpretation of ρ would at least be extremely difficult if not impossible and the log-log transform does not alter this fact. Rank based dependence measures Kendall's Tau τ and Spearman's Rho (see mathematical appendix) ρ_S can have full range $[-1,1]$ even when comparing discrete versus continuous RVs and an interpretation in the extremal cases of ± 1: monotone functional but discontinuous dependence. We use mostly τ since it has slightly more interpretation than ρ_S in the nonzero case. However, the computational complexity of τ is essentially worse than of ρ_S. These two measure of dependence are copula properties. Le us deep into the copula framework.

Copulas provide a convenient way to represent the joint distribution of two or more random variables. A copula and the marginal distributions of each variable are sufficient to express the joint distribution. Furthermore conditional distribution can also be expressed from the copula. This clarify how powerful copulas are in many applications.

Note that correlation coefficients are able to summarize the strength of the correlation in a single number but cannot describe how that varies across the distribution. Nevertheless, the copulas can do this because they express all the association between margins and, for example, they can emphasize the relationships in the tails of the distributions.

In the following with focus on copulas of couples of continuous random variables (X,Y) with probability measure $H(x,y)$ but the theory is similar for more than two variables. Let us define the copula (see the appendix for more details).

Definition 1 *A 2-dimensional copula is a function C with domain $[0,1]^2$ such that*

(i) C is grounded and 2-increasing

(ii) C has margins C_1, C_2 which satisfy $C_1(u) = u$, $C_2(u) = u$ $\forall u \in [0,1]$

2.4.2 Thresholding of data is a completely wrong method

Division of data by thresholding, even with 'small' thresholds is a wrong method. Assume, for example, that the sample value of τ for a data is strictly positive and that the data contains such a pair (x_0,y_0) for which $Rank(x_0) = Rank(y_0) = 1$ (the closer to 1 the value of τ is, the more probable such an situation is). Then, all remaining data points are concordant with this particular data point (x_0,y_0) and no data point can be discordant with it. If the data contains n points, remove of the point (x_0,y_0) reduces the number of concordant pairs in the remaining data by $n-1$ but the number discordant data pairs is not changed. Hence, the sample value of τ, which essentially depends on the ratio of concordant versus discordant data pairs, is strictly smaller for the remaining sample. It is clear that similar argument holds for any pair (x,y) such that both $Rank(x)$ and $Rank(y)$ are small. Iteration of such thresholded dropping, where the threshold is in one or another, or in both variables, will finally lead to the situation where the ratio of concordant and discordant data pairs is approximately equal and the sample value of $\tau \approx 0$. This does not tell anything about the dependence structure in the data.

Figure 2.22 shows what happens when data (X,Y) from the Marshall-Olkin Copula (see the appendix for the definition of copula) is dropped according to a threshold which is either in X or in Y coordinate. When the threshold increases, the value of τ ultimately decreases.

An alternative and better choice to thresholding is *conditioning*. One example of the conditioning approach is *Cumulative Tau*, which we denote here as J_τ. It is defined in terms of a copula C as

$$J_\tau(\alpha) = \frac{4}{C(\alpha,\alpha)^2} \int_{[0,\alpha]^2} C(u,v)\, dC(u,v) - 1. \tag{2.1}$$

Recall that $\tau = 4\mathbb{E}C(U,V) - 1$ and $\mathbb{E}C(U,V) = \mathbb{E}(\mathbb{E}[C(U,V)|(U,V) \in [0,\alpha]^2])$. As a function of α, the conditional expectation

$$\mathbb{E}[C(U,V)|(U,V) \in [0,\alpha]^2] = \frac{1}{C(\alpha,\alpha)} \int_{[0,\alpha]^2} C(u,v)\, dC(u,v) \tag{2.2}$$

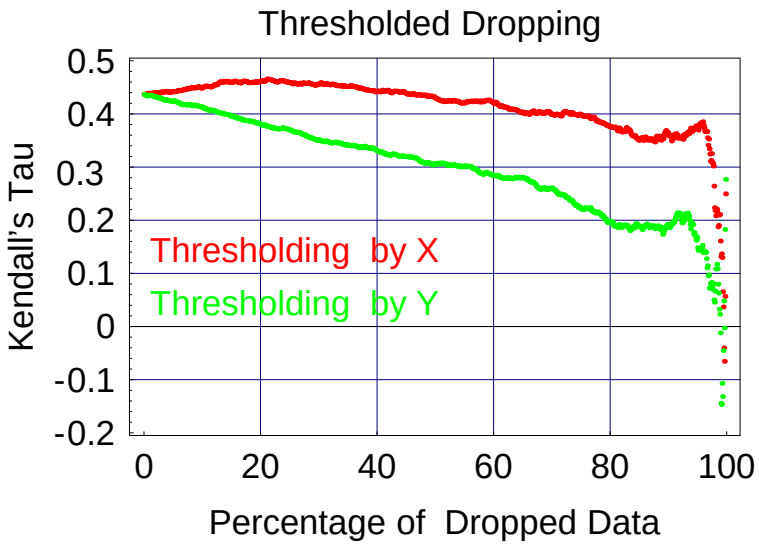


Figure 2.22. Thresholded dropping from the Marshall-Olkin Copula.

is always increasing. Note that this is in-line with what was said about thresholding above. However, when related to $C(\alpha,\alpha) = \mathbb{P}\{X \leq F^{-1}(\alpha), Y \leq G^{-1}(\alpha)\}$, the maximum value of C in the square $[0,\alpha]^2$, it can be increasing or decreasing *depending* on the copula. This brings the second power of $C(\alpha,\alpha)$ in the denominator of (2.1). Thus, the curve $J_\tau(\alpha)$, $0 \leq \alpha \leq 1$ can be increasing or decreasing but it's final value is always $J_\tau(1) = \tau$. If it is increasing, the interpretation is that the ratio of concordant data pairs in the quadrant space $]-\infty,F^{-1}(\alpha)]\times]-\infty,G^{-1}(\alpha)]$ is typically smaller than in the whole data set. If it is decreasing, the interpretation is the opposite. Previous works studied the correlation of flow traffic [HCJP+05, ZBPS02, LH03] removing the flows with a duration or size smaller than a chosen threshold. This choice has already been shown in [HCJP+05] to influence the estimation of the correlation coefficient so much that [HCJP+05] and [ZBPS02] obtained opposite results. For example, since $\tau(X,Y) = 4\mathbb{E}H(X,Y) - 1$, we can write

$$\tau(X,Y) = 4\mathbb{E}\left(\mathbb{E}[H(X,Y)|X > F^{-1}(u)]\right) - 1.$$

However, as a function of the threshold $u \in [0,1]$, the conditional expectation $\mathbb{E}[H(X,Y)|X > F^{-1}(u)]$ is *always decreasing* when $u \uparrow 1$. This implies that calculation of τ for the thresholded data, even with "small" thresholds is a method which, as such, does not tell anything about the dependence structure in the remaining data. Instead, τ is a single number which should be estimated from the whole data and considered as *a parameter* of the copula.

2.4.3 Rank and Copula Transforms

The Rank Transform is the map

$$(x_i,y_i) \mapsto (Rank(x_i),Rank(y_i))\,, \qquad i = 1,\ldots,n. \tag{2.3}$$

Since our data are observations from time series, the original order (x_i,y_i) indexed by $i = 1,\ldots,n$ in (2.3) above denotes the arrival order. We describe here an algorithm to calculate the Rank Transform for a data in the case that there are no ties. It assumes that the data is originally stored as $n \times 2$-matrix and has the following six steps:

1. Add a new 3rd column that will remember the original arrival order.
2. Sort the data according to the first column.
3. Add a new 4th column that will contain ranks of x_i.
4. Sort the data according to the 2nd column.
5. Add a new 5th column that will contain ranks of y_i.
6. Sort the data according to the 3rd column in order to get back to the original order.

The steps can be visualized as follows:

$$\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \vdots & \\ x_n & y_n \end{pmatrix} \overset{Begin}{\longrightarrow} \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 2 \\ x_3 & y_3 & 3 \\ \vdots & & \\ x_n & y_n & n \end{pmatrix} \overset{After2)}{\longrightarrow} \begin{pmatrix} x_{(1)} & \cdot & \cdot & 1 \\ x_{(2)} & \cdot & \cdot & 2 \\ x_{(3)} & \cdot & \cdot & 3 \\ \vdots & & & \\ x_{(n)} & \cdot & \cdot & n \end{pmatrix}$$

$$\overset{After4)}{\longrightarrow} \begin{pmatrix} \cdot & y_{(1)} & \cdot & \cdot & 1 \\ \cdot & y_{(2)} & \cdot & \cdot & 2 \\ \cdot & y_{(3)} & \cdot & \cdot & 3 \\ \vdots & & & & \\ \cdot & y_{(n)} & \cdot & \cdot & n \end{pmatrix} \overset{After6)}{\longrightarrow} \begin{pmatrix} x_1 & y_1 & 1 & Rank(x_1) & Rank(y_1) \\ x_2 & y_2 & 2 & Rank(x_2) & Rank(y_2) \\ x_3 & y_3 & 3 & Rank(x_3) & Rank(y_3) \\ \vdots & & & & \\ x_n & y_n & n & Rank(x_n) & Rank(y_n) \end{pmatrix}$$

Hence, in the end each row of the matrix will contain the Rank Transform image.

The algorithm requires three times adding a line number to each row, which is of complexity $O(n)$, and three times sorting of the matrix, $O(n \log n)$, hence the total complexity is $O(n \log n + n)$. It is fast enough so that the data size n is not a problem.

We emphasize that the algorithm does not work correctly if there are ties in x_i or y_i columns, but it does work correctly when we first add random noise to flow sizes since

sorting according to the size column produces a random order for tied observations while non-tied observations will get the correct ranks.

Let F_n^X and F_n^Y denote the 1-dimensional empirical distributions of the X and Y coordinates. Since

$$Rank(x_i) = \sum_{j=1}^{n} 1_{\{x_j \leq x_i\}} = nF_n^X(x_i), \tag{2.4}$$

from the Rank Transform (2.3) we get directly the Copula Transform

$$(x_i,y_i) \mapsto \left(F_n^X(x_i),F_n^Y(y_i)\right), \qquad i = 1,\ldots,n. \tag{2.5}$$

simply by dividing ranks with n.

2.4.4 The effect of ties must be taken into account correctly

Unless properly taken into account, ties introduce significant bias when estimating properties of an underlying copula. If X_1 and X_2 are two non-identical RVs drawn from the same continuous distribution,then $\mathbb{P}\{X_1 = X_2\} = 0$. If $x_1,\ldots,x_n$ is an accurate sample from the same continuous distribution, then we should have $x_{(1)} < \ldots < x_{(n)}$ and we can choose $Rank(x_{(i)}) = i$. Observations x_i and x_j, $i \neq j$, are *tied* if their values are the same: $x_i = x_j$. If this is the case for data points (x_i,y_i) and (x_j,y_j), $i \neq j$, then this pair is neither concordant nor discordant. A sample formula of τ when no ties are present in the data is denoted by τ_a. When calculating the sample value of τ in the presence of ties a slightly different sample formula called τ_b, which takes ties into account, must be used. If τ_a is used for data with ties, the range of τ_a will be different from $[-1,1]$ making the interpretation again extremely difficult. We show, however, a method that allows us to use τ_a even for data with ties. There are a couple of choices when ranking observations in the presence of ties. The problem, for example, of the 'mid-rank' method is that often it leads to non-integer ranks. We will need to calculate the Rank Transform, which maps pairs (x,y) to corresponding pairs $(Rank(x),Rank(y))$. In this context the integer ranks are superior and it is tempting to just sort the data and give ordinal numbers as ranks, *i.e.*, we would like to have $Rank(x_i) \neq Rank(x_j)$ even when $x_i = x_j$ for $i \neq j$, and that rank values are integers. A little bit surprisingly this is possible, but there is a very serious danger in such an operation which is easily explained by the following elementary example.

Example 2 *Assume we have data* $\{(123,2.4),(123,1.4),(123,3.4)\}$. *Computer algorithms that are used to sort data are deterministic: if they cannot find an unique order by the first*

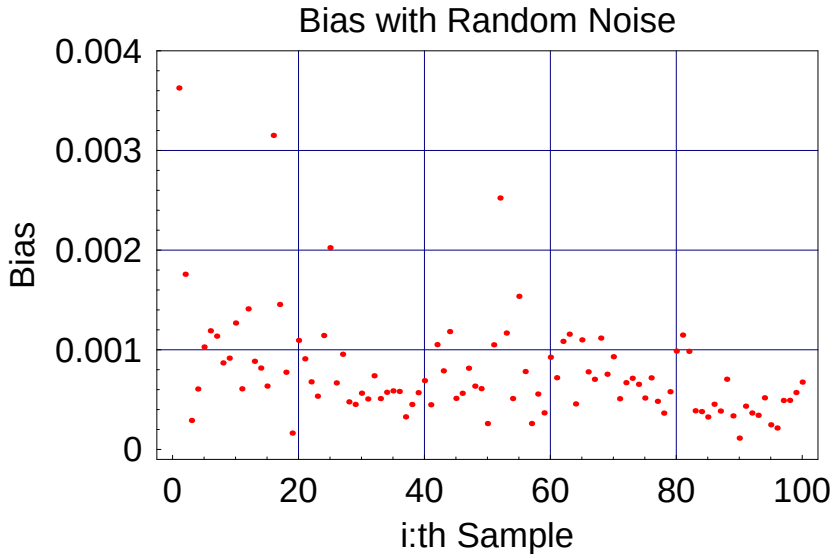


Figure 2.23. Bias $\tau_n^b - \tau_n^a$

variable they will look at the second column and produce something like

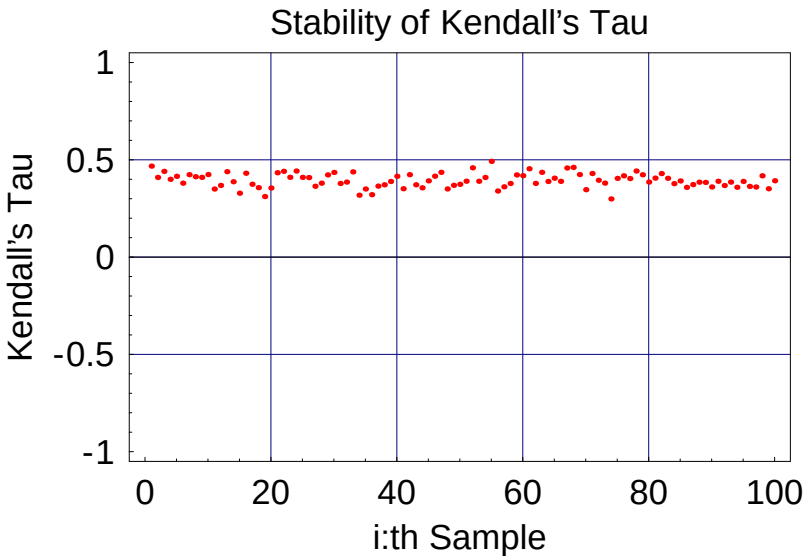
$(123,1.4)$ *:hence the rank of* 123 *of this pair would be* 1
$(123,2.4)$ *:hence the rank of* 123 *of this pair would be* 2
$(123,3.4)$ *:hence the rank of* 123 *of this pair would be* 3

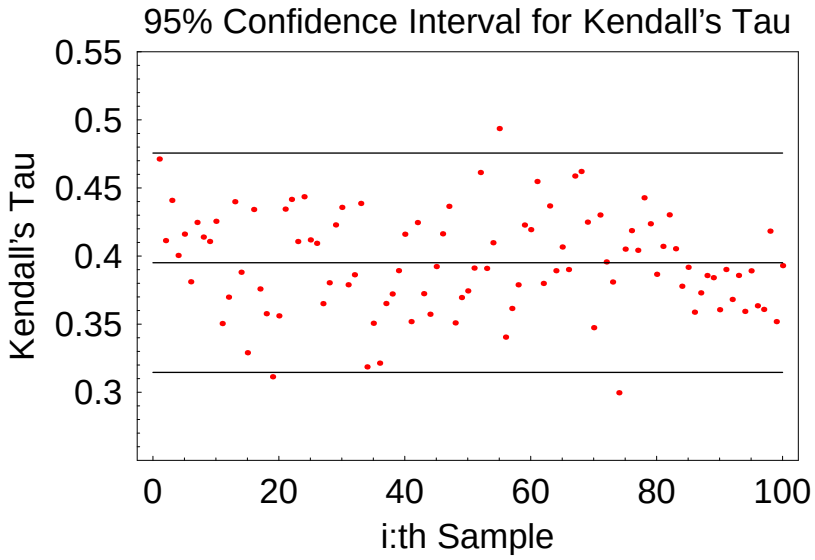
as output. However, if 123*s are ranked in this way we will bring to the data such dependence between* $Rank(x)$ *and* $Rank(y)$ *which is not present in the original data! Values of* y_i *must not affect in any way when we give ranks to* x_i*.*

	$\tau_{S,D}$	$\tau_{S,R}$	$\tau_{D,R}$	$\rho_{\log S,\log D}$	$\rho_{\log S,\log R}$	$\rho_{\log D,\log R}$
Elisa	0.33	0.62	-0.04	0.49	0.77	-0.17
Garr0	0.13	0.36	-0.51	0.22	0.50	-0.74
Garr1	0.06	0.46	-0.48	0.13	0.64	-0.67
Abilene0	0.19	0.40	-0.41	0.50	0.74	-0.21
Abilene1	0.35	0.19	-0.46	0.51	0.63	-0.34

Table 2.2. Estimation of Kendall's tau and Linear correlation coefficient

The method to avoid the above described danger is to put the pairs $\{(123,2.4),(123,1.4),(123,3.4)\}$ in *a random order* which is independent of the values of the other coordinate. Indeed, if tied ranks are replaced by unequal integer ranks, and we





take an average of τ_a over all possible integer ranking orders we get the same result as if we had used τ_b in the beginning. This is simply because the number of ways that pairs with ties are put in concordant or discordant order will be equal and they cancel out each other. This also suggest that if tied observations are ranked in a random order the effect is the same, especially if the number of ties is large.

Due to the discrete nature of flow sizes there are a lot of ties. Indeed, the percentage of non-tied, unique values is less than 10% for all of the data traces. Therefore we chose

to add *a random noise (RN)*, uniformly distributed between $]-0.5,0.5[$ to flow sizes. This random noise does not alter the ranks of non-tied values but has the effect that tied observations will be sorted in a random order. Moreover, as τ_a is computationally slightly less demanding than τ_b the sample value τ_a can be used to evaluate Kendall's Tau of samples. Taking equality in the estimate $Var(\tau_n) \leq \frac{2}{n}(1-\tau^2)$, which is always valid, it allows us to calculate a confidence interval for τ. The variability in the estimation seems to be explained by the Normal distribution.

The stability of τ in the random dropping is the starting point. It also suggests that the opposite direction, namely adding randomly new points to a sample from where the τ is estimated should stabilize rather fast. A natural random order is the flow arrival order.

Tail Dependence

Another measure of dependence is the coefficient of upper tail dependence λ_U, defined as

$$\lambda_U = \lim_{u \uparrow 1} \mathbb{P}\{X > F^{-1}(u) | Y > G^{-1}(u)\}.$$

provided that the limit exists. If $\lambda_U > 0$, then X and Y are asymptotically dependent in the upper tail. Upper tail dependence is a copula property since it can also be evaluated from the copula itself as the limit $\lambda_U = \lim_{u\uparrow 1}[1-2u+C(u,u)]/(1-u)$. Thus, λ_U should also be thought of as *a parameter* of the copula. (Note that $\tau = 0$ does not imply independence and it is possible to have simultaneously $\tau = 0$ and $\lambda_U > 0$.) The value of λ_U can be estimated from the *empirical copula* $C_n(u,v) = H_n(F_n^{-1}(u),G_n^{-1}(v))$.

2.4.5 Empirical Copula

Let $(X_1,Y_1),\ldots,(X_n,Y_n)$ be an i.i.d. sample from a distribution with a bivariate CDF H and with continuous marginal CDFs F and G of X and Y, respectively. Recall that when the marginals are continuous, the copula is unique and is given by $C(u,v) = H\left(F^{-1}(u),G^{-1}(u)\right)$. Recall also that the generalized inverse of an univariate empirical distribution is right-continuous with left-hand limits while univariate empirical distribution is left-continuous with right-hand limits.

There exist at least three choices to define *an empirical copula*. First of all, if we would know F and G exactly then we could first transform the observations into $(F(X_k),G(Y_k)) = (U_k,V_k)$, $k = 1,\ldots,n$, and define

$$C_n^*(u,v) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{U_k \leq u, V_k \leq v\}}. \tag{2.6}$$

However, a full knowledge of F and G is hardly available unless in certain very simple simulation scenarios. Instead of true marginals it is more typical to work with empirical

distributions, but still then there are two choices. The first, more common standard choice which mimics the definition of a copula is

$$C_n(u,v) = H_n\left(F_n^{-1}(u), G_n^{-1}(v)\right) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{X_k \le F_n^{-1}(u), Y_k \le G_n^{-1}(v)\}} \qquad \text{for all } (u,v) \in [0,1]^2 \tag{2.7}$$

and the other choice, which mimics (2.6) is

$$C'_n(u,v) = \frac{1}{n}\sum_{k=1}^{n} 1_{\{F_n(X_k) \le u, G_n(Y_k) \le v\}} \qquad \text{for all } (u,v) \in [0,1]^2. \tag{2.8}$$

These are not equivalent. Indeed, if $0 < u \le 1$, then $\frac{i-1}{n} < u \le \frac{i}{n}$ for some $1 = i, \ldots, n$, and then $F_n^{-1}(u) = X_{(i)}$ but $F_n(X_{(i)}) = \frac{i}{n}$. That is, when $\frac{i-1}{n} < u < \frac{i}{n}$, then $F_n(F_n^{-1}(u)) > u$. However, $F_n(F_n^{-1}(\frac{i}{n})) = \frac{i}{n}$ for all $i = 1, \ldots, n$. Hence, the definitions (2.7) and (2.8) agree on grid points $(\frac{i}{n}, \frac{j}{n})$, $i,j = 1, \ldots, n$ and the difference is small

$$\sup_{(u,v)\in[0,1]^2} |C_n(u,v) - C'_n(u,v)| \le \frac{2}{n}. \tag{2.9}$$

What is more surprising is that $C_n(u,v) = C_n^*(u,v)$ at every point $(u,v) \in [0,1]^2$.

We need not care about the difference of (2.7) and (2.8) since the copula transform (2.3) places our data points exactly to the grid points 2.24.

What is good to know about empirical copulas are convergence results summarized and proven in quite recent paper of Fermanian & al.,[FRW04]. Namely, these convergence results, which describe the limiting Gaussian process of *the empirical copula process* $\sqrt{n}(C_n(u,v) - C(u,v))$, require the existence of continuous partial derivatives of the copula C. Due to the discrete character of the flow sizes this may not be the case in our situation, at least it is not self-evident. (Recall that we are essentially working with a continuous *approximation* of the flow size distribution.) Indeed, we base our approach on the observation that the properties of the empirical copulas for different subsamples of a fixed trace seem to be quite stable and this holds for for all traces considered.

We will take advantage of the well-known facts that the empirical distribution (copula) is an unbiased estimator of the underlying distribution (copula)

$$\mathbb{E}C_n(u,v) = \mathbb{E}C_n^*(u,v) = C(u,v) \qquad \text{for all } (u,v) \in [0,1]^2 \tag{2.10}$$

with variance

$$Var(C_n(u,v)) = \frac{(1 - C(u,v))C(u,v)}{n} \qquad \text{for all } (u,v) \in [0,1]^2. \tag{2.11}$$

It follows from the ordinary CLT that, for large n and at any fixed inner-point $(u,v) \in]0,1[^2$, $C_n(u,v)$ is approximately normally distributed with mean and variance given by

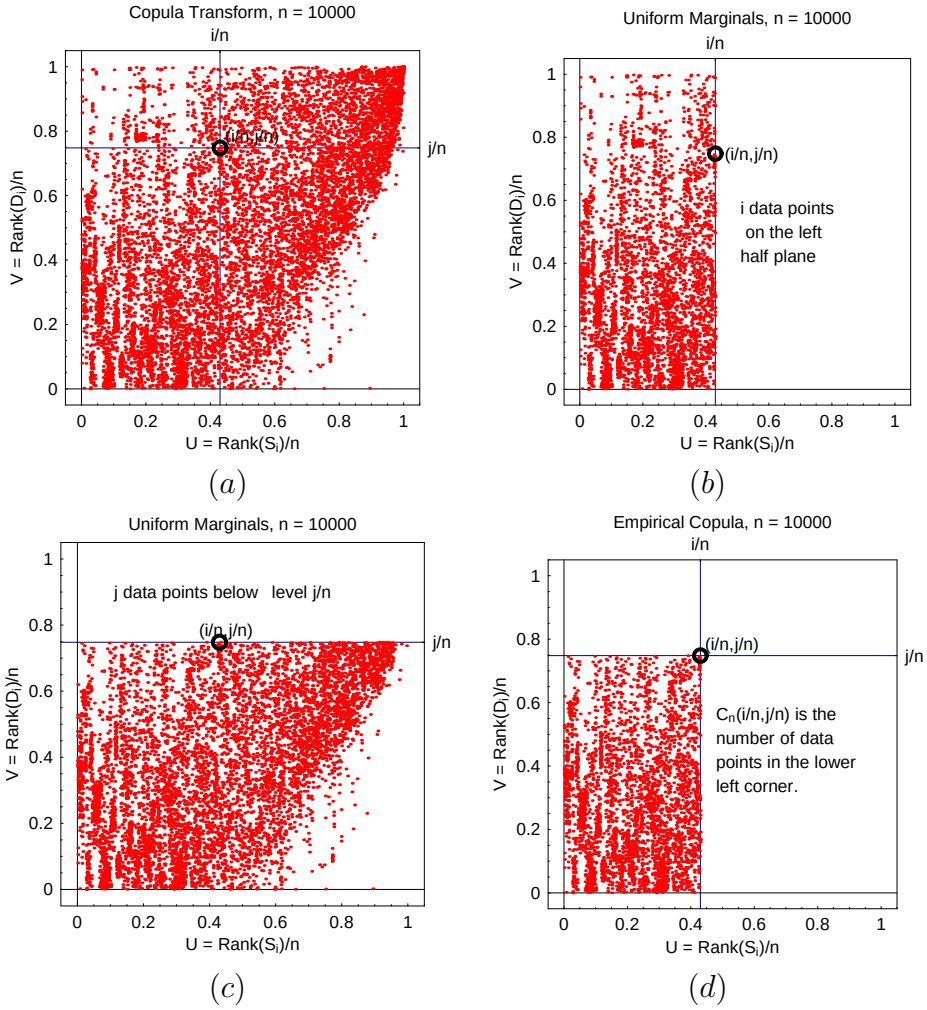


Figure 2.24. Figure (a) is the copula transform of Elisa (S,D)data. Figures (b) and (c) explain why the marginals are uniform. Figure (d) recalls visually that the value of the empirical copula at a grid point, $C_n\left(\frac{i}{n},\frac{j}{n}\right)$ is the number of data points in the lower left corner divided by n.

(2.10) and (2.11). (This result differs from the limiting Gaussian process of the empirical copula process in that n depends on the point (u,v).) Once we first estimate $C(u,u)$, this allows us to calculate a confidence interval (CI) tube around C when we evaluate it through the diagonal $\left(\frac{i}{n},\frac{i}{n}\right)$, see 2.25.

2.4.6 Histogram based approach

Working directly with the empirical copula is computationally meaningful only up to some level. (This level is surprisingly high though.) Histogram based approach reduces the size limits, is faster and smoothes the local variability. We simply divide $[0,1]^2$ into sub-squares and calculate first an approximation of the density and then an approximation of the empirical copula. Sub-squares could have equal or different sizes, for the moment we use equal size bins.

2.4.7 Inference about the upper tail dependence

Recall the definition of λ_U:

$$\lambda_U = \lim_{u \nearrow 1} \frac{1 - 2u + C(u,u)}{1-u}. \tag{2.12}$$

The very first idea is just to replace u by $\frac{i}{n}$ and $C(u,u)$ by $C_n\left(\frac{i}{n},\frac{i}{n}\right)$ above and see what happens when $i \to n$. This means a plot

$$\frac{1 - 2\frac{i}{n} + C_n\left(\frac{i}{n},\frac{i}{n}\right)}{1-\frac{i}{n}} \qquad i = 1,\ldots,n-12. \tag{2.13}$$

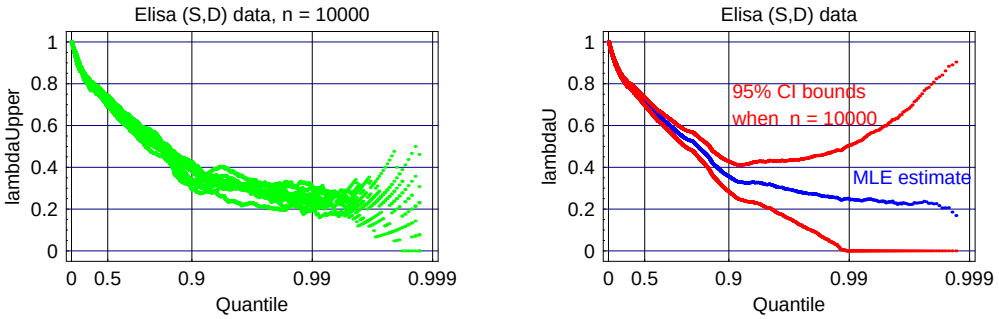
Figure 2.25 (a) shows ten such plots for ten disjoint Elisa (S,D) data samples, each of size $n = 10000$. We have dropped the last 12 points away due to fact that, for example, the value $C_n\left(\frac{n-12}{n},\frac{n-12}{n}\right)$ depends on at least 12 but at most on 24 extremal observations whose S or D ranks were larger that $n-12$.

In figure 2.25 (b) we have first replaced $C_n\left(\frac{i}{n},\frac{i}{n}\right)$ in (2.13) by $\widehat{C}\left(\frac{i}{n},\frac{i}{n}\right)$, where the $\widehat{C}$ is an MLE estimate,

$$\widehat{C}\left(\frac{i}{n},\frac{i}{n}\right) = \frac{1}{10}\sum_{k=1}^{10} C_n^k\left(\frac{i}{n},\frac{i}{n}\right) \tag{2.14}$$

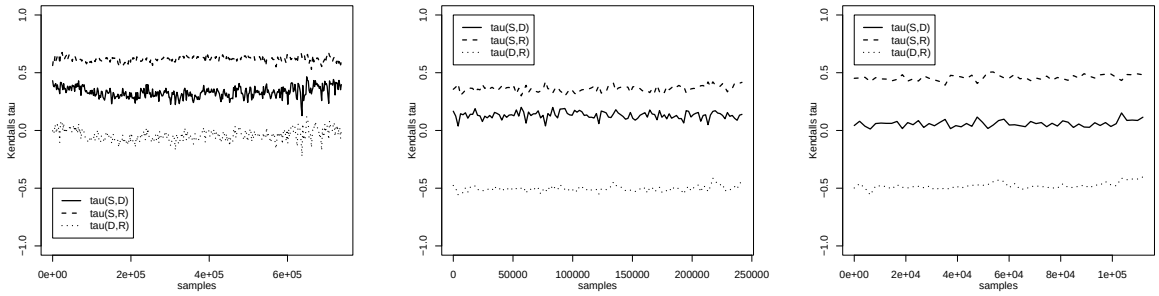
of $C\left(\frac{i}{n},\frac{i}{n}\right)$ and C_n^k, $k = 1,\ldots,10$ are the ten empirical copulas of the Elisa (S,D) data. Using the CI tube approach we can calculate a CI tube first to $\widehat{C}$ at the diagonal which translates directly to CI tube for (2.13).

From any finite data set we cannot infer limiting behavior for certain. While we can be quite certain that for the Elisa (S,D) data the $\lambda_U > 0$, at least the data does not support the case $\lambda_U = 0$, the value of λ_U cannot be estimated with so simple approach.



(a) (b)

Figure 2.25. (a) 10 different realizations of (2.13). (b) MLE estimate of the empirical curve (2.13) together with 95% CI tube.

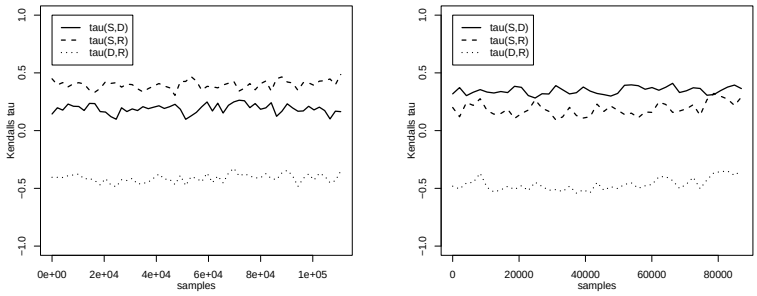


Elisa GARR Direction 0 GARR Direction 1

Figure 2.26. Kendalls tau estimation for Elisa and GARR data

2.4.8 Discussion on the Estimations

The structure of dependence between flow size, duration and rate is strongly related to the measurement point, that is to say that it is influenced by the traffic mix as summarized in



Abilene Direction 0 Abilene Direction 1

Figure 2.27. Kendalls tau estimation for Abilene data

	$\tau_{S,D}$	$\tau_{S,R}$	$\tau_{D,R}$	$\lambda_{S,D}$	$\lambda_{S,R}$	$\lambda_{D,R}$
Elisa	0.33	0.62	-0.04	0.47	0.50	0.18
Garr0	0.13	0.36	-0.51	0.22	0.18	0.02
Garr1	0.06	0.46	-0.48	0.19	0.37	0.10
Abilene0	0.19	0.40	-0.41	0.38	0.58	0.12
Abilene1	0.35	0.19	-0.46	0.20	0.40	0.07

Table 2.3. Summary table of the estimations on all considered data measurements

table 2.3 and in figures 2.26,2.27. The global measure of dependence τ and the coefficient of upper tail dependence λ_U have been measured and tested not to be equal to zero with a 95% significance level. For both coefficients confidence interval are available (as shown in the previous sections) that allow to claim that our estimation are not only unbiased but also accurate.

In this section we have shown that the study of the dependence should be done very carefully and we have chosen the copula framework to deal with dependence instead of simply measure correlation as done previously in the recent literature.

From our estimations we cannot claim to have considered every kind of traffic mix. Nevertheless we have a few evidence to observe.

The dependence between flow size, duration and rate comes in part from the transport protocol, that is also responsible of bandwidth sharing realized by the congestion control algorithm, and in part from the user behaviour. Users with a high access rate download large document in a small time compared to users with lower bandwidth. The structure of the dependence in (S,D,R) is then determined by the proportion of the users with different access rates. The more heterogeneous the traffic mix the more complex is the dependence to model while a homogeneous case can be explained very easily because users tend to behave, regarding the available bandwidth, in a similar way.

A network link like Elisa merges traffic from users that are very similar because they have similar access rate while the AbileneIII link is populated by very high speed users. The GARR network, on the contrary is more complex. It would be desirable to find a copula that could fit to a general scenario given a simple descriptor of the traffic mix. This problem has not been solved in this thesis but this approach looks very promising as an appropriate framework to describe Internet flows dependence.

2.5 Session Level Traffic

Study of telecommunication networks was often based on traffic measurements, out of which traffic models and performance estimates are built. While the measuring and modeling processes proved to be reasonably simple in traditional, circuit-switched telephone networks, they seems to be much harder in packet-switched data networks. In particular, on a data network like the Internet, the layered structure of the TCP/IP protocol suite requires the analysis of traffic at least at the IP (packet), TCP/UDP (flow[5]), and Application/User–behavior (session) layers. While lot of attention has been traditionally dedicated to the characterization of the packet and flow levels (see for example [PF95, CB97, CDJM91, DJ91, DJC+92, Pax94]), few are the studies on the properties at the session/user layer [PF95, CIWA02]. This is due to the difficulties in the definition of "session" itself [WVM98], which depends on the application used: applications such as TELNET or SSH tend to generate a single TCP connection per user session, whereas application layer protocols such as HTTP, IMAP/SMTP and X11 usually generate multiple connections per user session. Also, the generally accepted conjecture that such sessions would follow a Poisson arrival process (see [BPRJ01] for example) might have reduced the interest in the session process analysis.

The paper goals are, first, to correctly identify, and, second, to determine statistical properties of Web user-sessions, i.e., set of TCP connections created by a given user while surfing the Web in a given time frame, by analyzing traces of measured data. We concentrate on the identification of client Web user-sessions generated by a single host, being WWW the most widely used interactive service. We assume that only a single user runs a browser on each host, a reasonable assumption today given the vast majority of PC based hosts. The informal definition of a user session can be obtained by describing a typical behavior of a user running a Web browser: an activity period, when the user browses the Web, alternates with a silent period over which the user is not active on the Internet. This activity period, named session in this paper, may comprise several TCP connections, opened by the same host toward possibly different servers.

The identification of user-session plays an important role both in Internet traffic characterization and in the proper dimensioning of network resources. Unfortunately, the identification of active and silent periods is not trivial. Traditional approaches [PF95, CIWA02] rely on the adoption of a threshold η: TCP connections are aggregated in the same session if the inter-arrival time between two TCP connections is smaller than the threshold value; otherwise, a new session is identified. This approach works well if the threshold value is correctly matched to the average value of connection and session inter-arrival time; however, to know these values in advance is unrealistic in practice. If the threshold value is not correctly matched to user session statistical behavior, threshold based mechanisms are highly error prone in session identification.

[5] In this paper we interchangeably use the term "TCP connection" and "TCP flow".

Clustering techniques [HMS01] are used in many areas to find groups of similar variables/objects, to analyze a large data set, and in particular they allow to partition the data set in "similar subsets", by defining a proper notion of similarity. Typically, several metrics over which a distance measure can be defined are associated to points (named samples in this paper) in the data set; informally, the partitioning process tries to put in the same subset neighboring samples and in different subsets distant samples. The main advantage of using such approach is that there is no need to define a priori any threshold value. Thus, this methodology should be less error prone than simpler threshold based mechanisms.

The contributions of this paper are the following: first, we adapted classical clustering techniques to the described scenario, a non-trivial task that requires a lot of ingenuity to optimize the performance of user session identification algorithms both in terms of speed and precision. Then, we tested the proposed methodology using artificially generated traces to assess the error performance of the proposed technique and to compare it with traditional threshold based mechanisms. Finally, the defined algorithms were run over real traffic traces, to obtain statistical information on user sessions, such as distributions of i) session duration, ii) amount of data carried over a single session, iii) number of connection within a single session. A study of the inter-arrival times of Web user-sessions is also presented, from which it emerges that Web user-sessions tend to be Poisson, but correlation may arise during network/hosts anomalous functioning.

We revise some basics of clustering, whereas in Sec. 2.6.4 we describe the methodology adopted to analyze the measured data set, including the details on the specific use of clustering techniques. To obtain a statistically significant model, we need to ensure that the proposed methodology is able to correctly identify a set of TCP connections belonging to the same user session. Thus, the proposed methodology is tested in Sec. 2.8 against artificial data sets based on a simple statistical model of user behavior. Finally, in Sec. 2.9 we analyze the statistical properties of Web user-sessions as identified by the clustering technique methodology. Sec. 2.10 summarizes the paper.

2.5.1 Clustering Techniques

Progress in digital data acquisition and storage technology has resulted in the growth of huge databases. This has occurred in all areas of human endeavor, from the mundane (such as supermarket transaction data, credit card usage records, telephone call details, government statistics) to the more exotic (such molecular databases). The ability to extract useful knowledge hidden in these data and to act on that knowledge is becoming increasingly important in today's competitive world. The entire process of applying a computer-based methodology, including new techniques, for discovering knowledge from data is called *data mining*.

So, we can define data mining as the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.

The above definition refers to "observational data". Data mining typically deals with data that have already been collected for some purpose other than the data mining analysis (for example, they may have been collected in order to maintain an up-to-date record of all the transactions in a bank). This means that the objectives of the data mining exercise play no role in the data collection strategy. This is one way in which data mining differs from much of statistics, in which data are often collected by using efficient strategies to answer specific questions. For this reason, data mining is often referred to as "secondary" data analysis.

The two primary goals of data mining tend to be prediction and description. Prediction involves using some variables or fields in the data set to predict unknown or future values of other variables of interest. Description, on the other hand, focuses on finding patterns describing the data that can be interpreted by humans. Therefore, it is possible to put data-mining activities into one of two categories:

- *Predictive* data mining, which produces the model of the system described by the given data set, or
- *Descriptive* data mining, with produces new, nontrivial information based on the available data set.

On the predictive end of the spectrum, the goal of data mining is to produce a model which can be used to perform classification, prediction, estimation, or other similar tasks. On the other, the goal of descriptive data mining is to gain an understanding of the analyzed system by uncovering patterns and relationships in large data sets. The relative importance of prediction and description for particular data-mining applications can vary considerably. The goals of prediction and description are achieved by using data-mining techniques for the following primary data-mining tasks:

- *Classification*: discovery of a predictive learning function that classifies a data item into one of several predefined classes.
- *Regression*: discovery of a predictive learning function, which maps a data item to a real-value prediction variable.
- *Clustering*: a common descriptive task in which one seeks to identify a finite set of categories or clusters to describe the data.
- *Summarization*: an additional descriptive task that involves methods for finding a compact description for a set (or subset) of data.
- *Dependency Modeling*: finding a local model that describes significant dependencies between variables or between the values of a feature in a data set or in a part of a data set.

- *Change and Deviation Detection*: discovering the most significant changes in the data set.

2.6 Cluster Analysis

A central problem in data mining is that of automatically summarizing vast amounts of information into simpler, fewer and more comprehensible categories. The most common and well-studied way in which this categorizing is done is by partitioning the data elements into groups called clusters, in such a way that members of the same cluster are as similar as possible, and points from different clusters are as dissimilar as possible. By examining the properties of elements from a common cluster, practitioners hope to discover rules and concepts that allow them to characterize and categorize the data.

Cluster analysis has been widely used in numerous applications, including pattern recognition, data analysis, image processing. By clustering one can identify crowded and sparse regions, and discovers interesting correlation among data attributes.

Classic approaches to clustering include partitional methods such as k-means, hierarchical, agglomerative clustering, Un-supervised Bayes, mode finding or density based approaches. There are also a variety of soft clustering techniques. The main reason there is such a diversity of techniques is that although the clustering problem is easy to conceptualize, it may be quite difficult to solve in specific instances. Moreover the quality of clustering obtained by a given method is very data dependent, and although some methods are uniformly inferior, there is no method that works best over all types of data sets. Indeed, each approach has certain underlying assumptions about the data and thus is most suitable for specific scenarios, and there is no method that is "best" across all situations.

A clustering algorithm should consider the following requirements:

- *Scalability*: Many clustering algorithms work well in small data sets, however a large database can contain millions of objects. Clustering on a sample of a given large data set may lead to biased results.

- *Discovery of clusters with arbitrary shape*: Many clustering algorithms determine clusters based on Euclidean distance measures. Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could have any shape. A good clustering algorithm should be able to discover cluster of any shape.

- *Ability to deal with noisy data*: Most real database contain outliers or missing or erroneous data. A cluster algorithm should be insensitive to such data.

- *Minimal requirements to determine input parameters*: Many clustering algorithms require users to input certain parameters in cluster analysis, for instance the number

of desired final clusters. The clustering results are often quite sensitive to input parameters. Many parameters are hard to find out a priori.

2.6.1 Overview of the clustering methods

Let's consider a metric space X, which we refer to as a sampling space, and a set of samples $A = \{x_1, \ldots, x_N | \ x_i \in \ X\}$. We assume that each sample $x_i \in \ X$, $i = 1, \ldots, N$, is represented by a vector $x_i = \{x_i^1, x_i^2, \ldots, x_i^M\}$. The value M is the number of features of samples, while N is the total number of samples prepared for a clustering process that belongs to the sample domain X. Samples, represented conventionally as multidimensional vectors, have each dimension as a single attribute (feature). These features can be either quantitative or qualitative descriptions of the object. If x_i^j is the component j of a sample x_i, then each component x_i^j, $j = 1, \ldots, M$ is an element of a domain X_j, where X_j could belong to different types of data such as binary ($X_j = \{0,1\}$), integer ($X_j \in \ Z$), real number ($X_j \in \ R$), or a categorical set of symbols.

Quantitative features can be subdivided as

- continuous values (e.g., real numbers),
- discrete values (e.g., binary numbers, or integers),
- interval values (e.g., $X_j = \{x_i^j \geq 0, x_i^j \leq 100\}$).

Qualitative features can be:

- nominal or unordered;
- ordinal.

The above set of samples A has to be clustered into K subsets: we wish to find a partition $\mathcal{C} = \{C_1, \ldots, C_K\}$, such that $\cup_i C_i = A$ and $C_i \cap C_{j \neq i} = \emptyset$ satisfying some properties, with K possibly being unknown a-priori. The subsets in the partition are named *clusters*. They contain "similar" samples, whereas samples associated to different clusters should be "dissimilar". Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering algorithms. This measure must be chosen very carefully because the quality of a clustering process depends on this decision.

The word "similarity" in clustering means that the value of $s(x_i, x_j)$ is large when x_i and x_j are two similar samples; the value of $s(x_i, x_j)$ is small when x_i and x_j are not similar. Moreover, a similarity measure is symmetric, so $s(x_i, x_j) = s(x_j, x_i)$.

Very often a measure of dissimilarity is used instead of a similarity measure. A dissimilarity measure is denoted by $d(x_i, x_j) \forall x_i, x_j \in \ X$. Dissimilarity is frequently called a distance. A distance $d(x_i, x_j)$ is small when x_i and x_j are similar; if x_i and x_j are not similar $d(x_i, x_j)$ is large. We assume that:

- $d(x_i,x_j) \geq 0$;
- $d(x_i,x_j) = d(x_j,x_i)$;
- $d(x_i,x_k) \leq d(x_i,x_j) + d(x_j,x_k) \forall x_i,x_j,x_k \in X$;

The most known distance $d(x_i,x_j) = \sqrt{\sum_{k=1}^{M}(x_i^k - x_j^k)^2}$ is the classical Euclidean metric

In the next pages, we introduce two techniques, known in the data mining and in the descriptive multivariate statistic environments as "partitional" clustering and "hierarchical" clustering.

2.6.2 The partitional approach

This technique is used when the number K of final clusters in known. The procedure starts with an initial configuration comprising K clusters, selected according to some criteria; the cluster configuration is modified through an iterative process until "steady state" is reached. The two most popular representatives are the *k-means* algorithm and the *k-median* algorithm. The k-means algorithm is an iterative algorithm to minimize the least squares error criterion. The cluster C_i is represented by the so called *centroid* $\hat{c}_i$, defined as the mean value of the cluster samples, i.e.,

$$\hat{c}_i^k = \frac{1}{|C_i|} \sum_{x \in C_i} x^k \qquad k = 1, \ldots, n$$

In other words, $\mathcal{R}(C_i) = \hat{c}_i$ is the representative of cluster C_i.

At the beginning, K clusters are created, with cluster centroids positioned according to a given rule in the measure space, e.g., randomly positioned or partitioning the measured space in $K + 1$ equi-spaced areas. Each sample is associated to the closest cluster, according to the distance between the sample and the centroid of each cluster. When all samples are assigned to a cluster, new centroids are computed and the procedure is iterated. This algorithm reaches steady state when either a prefixed number of iterations is reached, or the number of samples which are moved to a different cluster is negligible according to a predefined threshold. Clearly, there is no guarantee that the final solution is always the same when varying the initial state.

We can summarize the steps of the K-means algorithm in the following way:

1. Select an initial partition with K clusters containing randomly chosen samples, and compute the centroids of the clusters;
2. generate a new partition by assigning each sample to the closest cluster center;
3. compute new cluster centers as the centroids of the clusters;

4. repeat steps 2 and 3 until an optimum value of the criterion function is found (or until the cluster membership stabilizes).

The data model underlying the k-means algorithm is a mixture of Gaussians with identical and spherical covariance. Thus, it does best when one makes the correct guess for k and the data is naturally clumped into spherical regions with comparable spreads. If the data actually deviates significantly from this scenario, the cluster quality can be very poor. Still, it remains one of the most popular partitional methods.

The k-median algorithm is very similar except that the cluster representative is constrained to be one of the actual data points rather than the geometric center. This involves extra computations.

2.6.3 The hierarchical approach

Whereas partition-based methods of cluster analysis begin with a specified number of clusters and search through possible allocations of points to clusters to find an allocation that optimizes some clustering score function, hierarchical methods gradually merge points or divide super-clusters. In fact, on this basis we can identify two distinct types of hierarchical methods: the agglomerative (which merge) and the divisive (which divide). The agglomerative are the more important and widely used of the two.

Hierarchical methods of cluster analysis permit a convenient graphical display, in which the entire sequence of merging (or splitting) of clusters is shown. Because of its tree-like nature, such a display is called a dendrogram (Fig. 2.28).

Agglomerative approach

Agglomerative methods are based on measures of distance between clusters. Essentially, given an initial clustering, they merge those two clusters that are nearest, to form a reduced number of clusters. This is repeated, each time merging the two closest clusters, until just one cluster, of all the data points, exists. Usually at the beginning of the procedure, each sample is associated to a different cluster, i.e., $C_i = \{x_i\}$, thus the number of cluster N_c is equal to $N = |A|$. Then, based on a definition of a cluster-to-cluster distance, the nearest clusters are merged in a new cluster. Iterating this step, the procedure ends when all samples belong to the same cluster $C = A = \{x_1, \ldots, x_N\}$, and $N_c = 1$. This procedure defines a merging sequence based on minimum distance between clusters. At each step $i = 1, \ldots, N$, the *quality indicator* function $\gamma^{(i)}$ is evaluated. The set A is finally clustered by selecting the number of clusters $N_c = N - (i - 1)$ such that $\gamma^{(i)} - \gamma^{(i-1)}$ is maximized. Intuitively, the quality indicator function $\gamma^{(i)}$ measures the distance between the two closest cluster at step i. A sharp increase in the value of $\gamma^{(i)}$ is an indication that the merging procedure is merging two clusters which are too far apart, thus suggesting to adopt the previous partition as the best cluster configuration. Fig. 2.28 sketches a

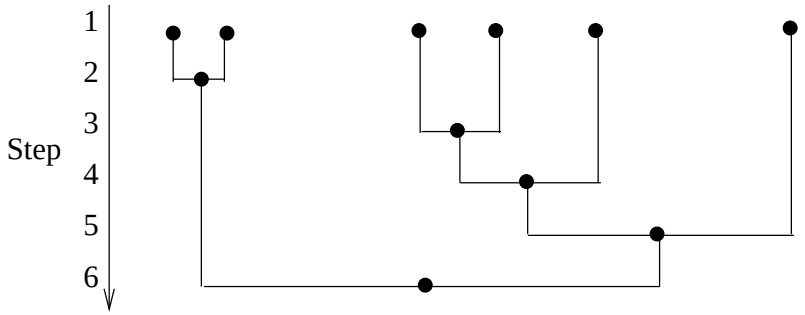


Figure 2.28. Hierarchical clustering scheme

hierarchical agglomerative clustering evolution starting from 6 initial samples, which are merged step by step into clusters.

We can summarize the basic steps of the agglomerative clustering algorithm in the following way:

- Place each sample in its own cluster. Construct the list of inter-cluster distances for all distinct unordered pairs of samples, and sort this list in ascending order;
- Step through the sorted list of distances, connecting into a new cluster the two closest clusters. If all the samples are members of the same cluster, stop. Otherwise, repeat this step.

It is clear how time consuming this approach can be, especially when the data set is very large, given the need of starting with an initial number of clusters equal to $N_c = |A|$, i.e., the number of samples in the data set.

To better understand, we can take a look at the time complexity. In the beginning there are $N_c = |A|$ clusters, and in the end 1; thus there are N iterations of the main loop. In iteration i we have to find the closest pair of clusters among $N - i + 1$ clusters. There are a variety of methods for defining the inter-cluster distance. All of them, however, require in the first iteration that we locate the closest pair of objects. This takes $O(N^2)$ time, unless we have special knowledge about the distance between objects. Thus, the method is typically not feasible for large values of N. Furthermore, interpreting a large dendrogram can be quite difficult. Note that in agglomerative clustering we need distances between individual data objects to begin the clustering, and during clustering we need to be able to compute distances between groups of data points (that is, distances between clusters).

In terms of the general case of distances between sets of objects (that is, clusters) many measures of distance have been proposed. One of the earliest and most important of these is the nearest neighbor or *single link method*. This defines the distance between

two clusters as the distance between the two closest points, one from each cluster;

$$d(C_i,C_j) = \min_{x \in \mathcal{R}(C_i), y \in \mathcal{R}(C_j)} d(x,y) \tag{2.15}$$

where $d(x,y)$ is the distance between objects x and y. The single link method is susceptible (which may be a good or bad thing, depending upon our objectives) to the phenomenon of "chaining" in which long strings of points are assigned to the same cluster (contrast this with the production of compact spherical clusters). This means that the single link method is of limited value for segmentation. It also means that the method is sensitive to small perturbations of the data and to outlying points (which, again, may be good or bad, depending upon what we are trying to do). The single link method also has the property (for which it is unique, no other measure of distance between clusters possesses it) that if two pairs of clusters are equidistant it does not matter which is merged first. The overall result will be the same, regardless of the order of merger.

At the other extreme from single link, furthest neighbor, or *complete link* , takes as the distance between two clusters the distance between the two most distant points, one from each cluster:

$$d(C_i,C_j) = \max_{x \in \mathcal{R}(C_i), y \in \mathcal{R}(C_j)} d(x,y) \tag{2.16}$$

where d(x, y) is again the distance between objects x and y. For vector objects this imposes a tendency for the groups to be of equal size in terms of the volume of space occupied (and not in terms of numbers of points), making this measure particularly appropriate for segmentation problems.

Other important measures, intermediate between single link and complete link, include (for vector objects) the centroid measure (the distance between two clusters is the distance between their centroids), the group average measure (the distance between two clusters is the average of all the distances between pairs of points, one from each cluster), and Ward's measure for vector data (the distance between two clusters is the difference between the total within cluster sum of squares for the two clusters separately, and the within cluster sum of squares resulting from merging the two clusters discussed above). Each such measure has slightly different properties.

Divisive Approach

Divisive methods begin with a single cluster composed of all of the data points, and seek to split this into components. These further components are then split, and the process is taken as far as necessary. Ultimately, of course, the process will end with a partition in which each cluster consists of a single point. Monothetic divisive methods split clusters using one variable at a time. This is a convenient (though restrictive) way to limit the number of possible partitions that must be examined. It has the attraction that the result is easily described by the dendrogram. The split at each node is defined in terms of just a

single variable. In general, divisive methods are more computationally intensive and tend to be less widely used than agglomerative methods.

In this section we briefly describe two main classes of clustering techniques, which will be used in the next sections as key tools. Clustering is a general framework which can be applied to many different areas to infer some sort of similarity among subsets of data, typically on a large size repository. More details on clustering techniques can be found in [HMS01].

2.6.4 Using Clustering Techniques on Measured Data Set

In this section we describe the methodology we developed for the identification of Web user-sessions, including the choices made in the cluster analysis. The description refers to the analysis of data collected through measurement procedures; similarly, the same process was used to analyze artificial traffic, when trying to determine the procedure ability to correctly identify user sessions. We start by giving some details about the dataset of traces that will be analyzed, to define the variables that will be used by the clustering algorithm.

Traffic traces were collected on the Internet access link of Politecnico di Torino, i.e., between the border router of Politecnico and the access router of GARR/B-TEN[top05], the Italian and European Research network. Within the Politecnico campus LAN, there are approximately 7,000 hosts; most of them are clients, but several servers are also regularly accessed from the outside. The backbone of the campus LAN is based on a switched Fast Ethernet infrastructure. It behaves as a stub Internet subnetwork: there is a single point of access to the GARR/B-TEN network and, through it, to the public Internet. The link speed is 28Mbps. A strict regulation of the network facilities is imposed by means of a firewall architecture which blocks (most of) the peer-to-peer traffic. Thus, still today the majority of our Internet traffic is built by Web browsing. Details on the measurements setup and traffic characteristics can be obtained from [MCC02].

Since 2001, several traces have been regularly collected. Among the available data we selected two different time periods:

- OCT.02: from 23/10/2002 to 31/10/2002
- APR.04: from 29/4/2004 to 6/5/2004

Both periods comprise more than a week of continuously traced data. We performed the analysis by considering only the *working* period, i.e., traffic from 8AM to 8PM, Monday to Friday. In Sec.2.9 we mainly report the results relatively to the latter period, being the differences among the two dataset almost negligible. Moreover, the APR.04 dataset includes two working days (the 5th and 6th of May) during which terminals in our campus were attacked and infected by the so called “Sasser.B” worm [tSW04]. The worm infection does not affect our measurement campaign, because the spreading of the worm itself

is not based on the HTTP protocol. Nonetheless, the drawbacks of the network and host malfunctions and subsequent requirements to download worm removal tools and patches for the Operating System have quite a large impact on the properties of the user session, as we will show later in this paper.

Bidirectional packet level traces have been collected using *tcpdump* [MLJ05], and then later processed by *Tstat* [Mel05] to obtain a TCP level trace. Tstat is an open source tool developed at the Politecnico di Torino for the collection and statistical analysis of TCP/IP traffic. In addition to standard and recognized performance figures, Tstat infers TCP connection status from traces. Among all the statistical analysis performed, Tstat produce a TCP level log file, which logs all the TCP connection observed. A TCP flow starts when the first SYN segment from the client is observed, and is terminated either when the tear-down sequence of segments is observed (either the FIN/ACK or RST messages), or when no segment is observed for an amount of time larger then 15 minutes[6]. Only TCP connections whose three-way-handshake was successfully completed are tracked; thus misbehaving connections and activities (e.g., port-scanning) do not affect the dataset. For the purpose of this paper, we used Tstat to track:

- $f_{id} = (C_{IP}, S_{IP}, C_{TCP}, S_{TCP})$: the 4-tuple identifying the flow, i.e., IP addresses and TCP port numbers of client and server (when the IP protocol field is set to TCP).
- t: the flow start time, identified by the time stamp of the first client SYN message.
- t_d: the time instant in which the last segment carrying data is observed (either from the client or form the server).
- t_e: the flow end time, identified by the time instant in which the tear-down procedure is terminated.
- B_c and B_s: the byte-wise amount of data sent from the client and server respectively (excluding retransmissions).

2.6.5 Clustering definition

The first fundamental choice regards the n statistical variables used to define the metric space $X = \mathbb{R}^n$ to be used in the clustering analysis; this implies also to select the metric space that best fits our problem. The typical and easiest approach is to let the clustering algorithm to run over a very large number of statistical variables, typically including the

[6]Given the possibility that a tear-down sequence of a flow under analysis is never observed, a timer is needed to avoid memory starvation. We selected a quite large value for the timer, according to the findings in [IDG+01].

vast majority of available data (in our example, potential statistical variables may be IP source address, TCP destination port, TCP flows starting and ending time, etc).

However, after several trials, we noticed that an accurate pre-filtering of data available in traces both improves algorithmic speed and provides more accurate results. Recalling that we wish to identify a Web user-session, i.e., a group of TCP connections corresponding to the activity period of a user running a Web browser, we used the opening time t of a TCP connection as the only statistical variable for the clustering process; thus $n = 1$, and $X = \mathbb{R}$. We tried to include in the space definition also the ending time of a TCP connection, but we found that this can lead to misleading results, due the presence of very long data transfers which may span over long period of time. Similarly, considering IP destination addresses is not helpful, given that during a user session several servers may be contacted with very different IP addresses (e.g., advertisement pictures may be retrieved from a different server under a completely different administrative domain, or Content Delivery Network service may force the information retrieval from different servers).

Before running the clustering algorithm on this variable, traces are pre-processed according to the following rules. We assume that a "user" is identified by its client IP address C_{IP}, and only connections having TCP server port S_{TCP} equal to 80 (HTTP protocol) are considered to be Web connections. To consider only significant IP (users), we selected among the about 7000 IP addresses that appear in the traces, the most active hosts, i.e., the top 1500 campus LAN IP addresses with respect to the number of generated TCP connections. Each user trace, i.e. a trace containing only data with a given IP source address, is pre-processed according to the following steps: i) data are partitioned day by day, ii) only working hours of working days are considered to obtain a set of statistically homogeneous samples and iii) opening times of two subsequent flows separated by more than half an hour are a priori considered as two independent data sets. Thus, for a given host IP, the set of samples

$$A(\mathrm{IP}) = \{t \mid C_{IP} = \mathrm{IP}, S_{TCP} = 80, t_i - t_{i+1} < 1800s\}$$

represents the opening times of TCP connections within a given time-frame. The intuition behind this is to allow the clustering algorithm to concentrate only on TCP connections created by a single user avoiding to be confused by too long silence periods.

After the metric space definition, we need to select a cluster analysis method. Recall that hierarchical agglomerative cluster analysis methods proceed by creating clusters through a cluster merging procedure: this method is easy to implement, but it does not scale well with the number of samples, which in our case is fairly large (during OCT.02, $N = |A(\mathrm{IP})|$ ranges up to 57000 samples, while during APR.04, $N = |A(\mathrm{IP})|$ ranges up to 26000 samples). On the other hand, partitional methods, which are relatively efficient, require an a priori knowledge of the number of clusters K. Moreover, in partitional methods the placement of the centroid may have a great impact on both the quality of the clustering, and the speed of convergence. To take the advantages and to avoid the drawbacks of both methods, we use a mix of them. Thus, for each $A(\mathrm{IP})$, the following 3-step

algorithm is run to identify sessions relative to a given user:

1. an initial smart clustering is obtained by selecting a number of clusters large enough but smaller than the number of samples of A
2. a hierarchical agglomerative algorithm is used to aggregate the clusters and to obtain a good estimation of the final number of clusters N_c
3. a partitional algorithm is used to obtain a fine definition of the N_c clusters.

Initial clustering selection We use a partitional method with K clusters, where K is large enough, but significantly smaller than the total number of samples (a study of the impact of K is presented in Sec. 2.8). To efficiently position the K centroids at the first step, in our uni-dimensional metric space, we evaluate the distance between any of two adjacent samples t_i,t_{i+1}. According to the distance metric $d(t_i,t_{i+1}) = |t_i - t_{i+1}|$, we take the farthest $(K-1)$ couples and determine K intervals. Let $t_{i,inf}$,$t_{i,sup}$ be the inferior and superior bounds of interval i, the centroid position of each cluster is set to $\hat{c}_i = (t_{i,sup} + t_{i,inf})/2$, and the partitional method is run for up to 1000 iterations: therefore, we define K initial clusters.

At this step, we represent each cluster C with a small subset $\mathcal{R}(C)$ of samples; $|\mathcal{R}(C)| \leq 2$ is enough in our case, since the metric space is $\mathbb{R}$. Possible choices for $\mathcal{R}(C)$ are: (i) the cluster centroid, which gives the name "centroid method" (or K-means) to the procedure; (ii) the g-th and $(100-g)$-th percentiles, which yields the so-called (iii) "single linkage" procedure when $g = 0$.

The hierarchical agglomerative procedure. A hierarchical method is iteratively run using only the representative samples $\{\mathcal{R}(C)\}$ to evaluate the distance between two clusters, and starting with K initial clusters, therefore enormously reducing the number of steps of the hierarchical method. At each iteration, the hierarchical procedure merges the two closest clusters; distances among clusters are recomputed. After K iterations, the process ends.

We need to define the clustering quality indicator function $\gamma^{(i)}$ that allows us to select the best clustering among those determined in the iterative process. Indeed, at each step, we must evaluate the quality of the clustering to decide if the optimal number of clusters has been found. Denote the j-th cluster at step i as $C_j^{(i)}$; at each step, we evaluate the function $\gamma^{(i)}$:

$$\gamma^{(i)} = \frac{d_{min}^{(i)} - \bar{d}_{min}^{(i)}}{\bar{d}_{min}^{(i)}}$$

where

$$d_{min}^{(i)} = \min_{j,k \neq j} d\left(C_j^{(i)},C_k^{(i)}\right), \quad \bar{d}_{min}^{(i)} = \frac{\sum_{l=1}^{i-1} d_{min}^{(l)}}{i-1}$$

and $d\left(C_j^{(i)},C_k^{(i)}\right)$ is defined according to Eq.(2.15)

A sharp increase in the value of $\gamma^{(i)}$ is an indication that the merging procedure is artificially merging two clusters which are too far apart. The optimal number of clusters N_c is:

$$N_c = N - \left(\underset{i}{\text{argmax}} \ \left(\gamma^{(i)} - \gamma^{(i-1)} \right) - 1 \right)$$

which corresponds to the sharpest increase in $\gamma^{(i)}$ and is thus considered the best one for partitioning the sample set. A typical behavior of the evolution of the $\gamma^{(i)}$ function is reported in Fig. 2.29, where the sharpest increase is clearly visible. The plot refers to an artificial trace obtained as described in Sec. 2.8, and shows that for about 1000 steps the aggregation of the two closest clusters is clearly beneficial in terms of clustering quality. Then, the aggregation process joins two clusters which are too far apart, forcing a sudden increase in $d_{min}^{(i)}$ from step $i-1$ to step i and therefore in $\gamma^{(i)}$. When $\gamma^{(i)}$ reaches the maximum, the joining procedure is forcing the artificial aggregation of two distinct clusters. Other errors are induced later in the iterative aggregation process: although clearly visible, they have a minor impact on the quality indicator function.

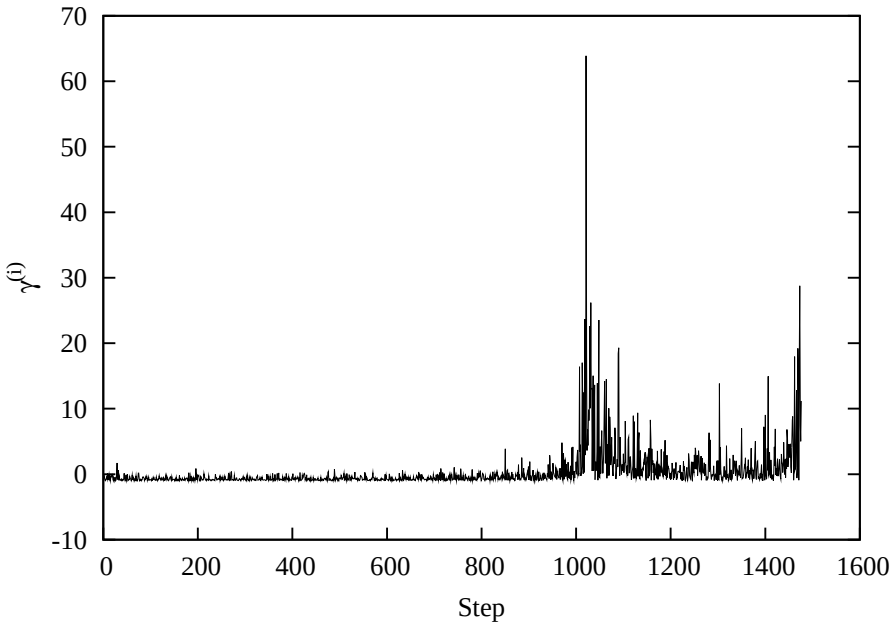


Figure 2.29. A sample plot of the quality indicator function $\gamma^{(i)}$. Exponential flow inter-arrival scenario.

Final clustering creation Finally, a partitional clustering procedure is run over the original dataset which includes all samples, using the optimal number of clusters N_c determined so far, therefore using the same procedures adopted in the first step (either the centroid, g percentile or single linkage methods). Then a fixed number of iterations of the

partitional approach are run, to permit a final refinement on the clustering definition. This third phase is not strictly required, given that at the end of the hierarchical procedure a partition is already given. However, it produces cluster of real points instead of centroids (which may not coincide with any data point). In addition, the computational cost of this phase is almost negligible compared to the previous one.

The complete clustering algorithm has been implemented by a number of Perl scripts, and by using the "R-Project for Statistical Computing" tool [Dal02], which efficiently implements many clustering algorithms.

2.7 Modelling Threshold and Clustering Based Systems

In this section we model the error probability of threshold based methods in detecting sessions.

2.7.1 Modelling Threshold Based Systems

Suppose user generates web sessions according to a stationary renewal process. During the active period of a session, a user generates TCP flows according to inter-arrivals which are i.i.d.; let $F_{T_{arr}}$ be the CDF of the flow inter-arrival process. After the last flow within a session has been sent, the session is said to be finished and the user become idle. The idle period is the time elapsed between the arrival times of the last flow of a session and the first flow of the following one. Idle periods are i.i.d.; let $F_{T_{OFF}}$ be the CDF of session idle times. Let N be the number of inter-arrivals in a data set. We can distinguish among inter-arrivals due to inter-session flow (N_{ON}) and intra session flows (N_{OFF}), so that $N = N_{ON} + N_{OFF}$.

Given a threshold η, an error occurs whenever either $T_{arr} > \eta$ or $T_{OFF} < \eta$. Therefore, we can evaluate

$$\begin{aligned} p_{arr} = \mathbb{P}\{T_{arr} > \eta\} &= 1 - F_{T_{arr}}(\eta) \\ p_{OFF} = \mathbb{P}\{T_{OFF} < \eta\} &= F_{T_{OFF}}(\eta) \end{aligned}$$

Let N_1 and N_2 be the random variables corresponding to the number of errors due to the first and second case respectively. Both variables have a Binomial distribution, $N_1 \sim B(N_{ON},p_1)$ and $N_2 \sim B(N_{OFF},p_2)$, because of the i.i.d. properties of the session and flow inter-arrival processes.

The random variable corresponding to the total number of errors is then given by the sum of N_1 and N_2, but its distribution is not binomial in general. However, it is possible to easily derive its generating function, but it is not interesting for our analysis, given that we are interested in $\epsilon(\eta)$, the average number of errors normalized over N samples. Given

that $\mathbb{E}[N_{ON}] = \mathbb{E}[N_{OFF}]\mathbb{E}[T_{ON}]/\mathbb{E}[T_{arr}]$ we write $\epsilon(\eta)$

$$\begin{aligned}\epsilon(\eta) &= \frac{\mathbb{E}[N_1 + N_2]}{\mathbb{E}[N]} = \\ &= \frac{\mathbb{E}[N_{ON}]\, p_1 + \mathbb{E}[N_{OFF}]\, p_2}{\mathbb{E}[T_{ON}]/\mathbb{E}[T_{arr}]\mathbb{E}[N_{OFF}] + \mathbb{E}[N_{OFF}]} \\ &= \rho p_1 + (1-\rho)p_2\end{aligned}$$

where

$$\rho = \frac{\mathbb{E}[T_{ON}]/\mathbb{E}[T_{arr}]}{\mathbb{E}[T_{ON}]/\mathbb{E}[T_{arr}] + 1}$$

Let us evaluate the minimum value for $\epsilon(\eta)$ and the corresponding value of the threshold by solving the following equation

$$\frac{\partial \epsilon}{\partial \eta} = -\rho f_{T_{arr}}(\eta) + (1-\rho) f_{T_{OFF}}(\eta) = 0$$

where f is the corresponding probability density function of the given random variable. By solving the previous equation, we obtain

$$\frac{f_{T_{OFF}}(\eta)}{f_{T_{arr}}(\eta)} = \frac{\rho}{1-\rho} \tag{2.17}$$

The solution depends on the assumed distributions. We will consider non negative random variables with a probability density function everywhere convex. In this case the solution is unique.

As an example, let us consider the case in which all the random variables are exponentially distributed, of parameters $\mu_{OFF} = \mathbb{E}[T_{OFF}]/\mathbb{E}[T_{arr}]$, $\mu_{ON} = \mathbb{E}[T_{ON}]/\mathbb{E}[T_{arr}]$ respectively. By solving eq.(2.17) we have:

$$\frac{\eta}{\mathbb{E}[T_{arr}]} = \frac{\mu_{OFF}}{\mu_{OFF} - 1} \log \mu_{ON}$$

The optimal threshold is much more sensitive to the ON period.

2.8 Performance Analysis: Artificial Traffic

In this section, we discuss the correctness properties of the proposed approach and compare it with respect to threshold based mechanisms.

Let us consider a simple artificial trace in which a *single* user generates sessions according to an ON/OFF process. Denote the ON and OFF period average durations as

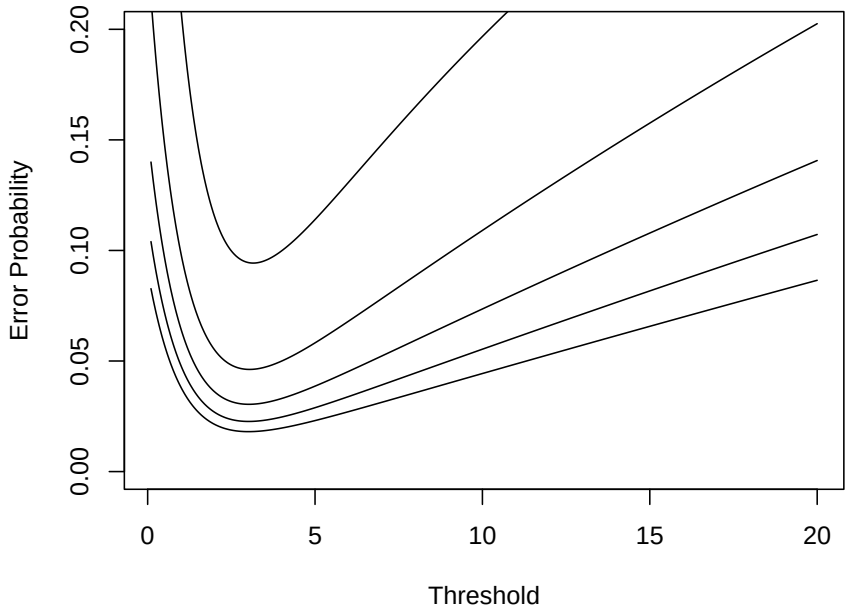


Figure 2.30. Error probability as a function of the threshold in the case of exponential random variables, with $T_{on} = 20s$ and $T_{arr} = 1s$ and for $20s \leq T_{off} \leq 200s$.

T_{on} and T_{off}. During each ON period of a session, a random number of TCP flows is generated, with average inter-arrival time denoted by T_{arr}. The session ON and OFF periods are random variables, exponentially distributed, with average $T_{on} = 20s$, whereas T_{off} ranges between $30s$ and $2000s$. For what concern flow arrivals, we set the average inter-arrival time $T_{arr} = 1s$ and we consider both the exponential and Pareto distributions. Indeed, considering the exponential distribution yields to a classic Poisson process, with no possible control on the variance of the flow inter-arrival process ($\sigma^2 = 1$). On the contrary, considering Pareto distribution allows us to control the variance of the process. Recall that the Pareto distribution $f(x)$ is characterized by two parameters a and b; the probability density function is:

$$f(x) = \frac{ab^a}{x^{a+1}} \quad x > b$$

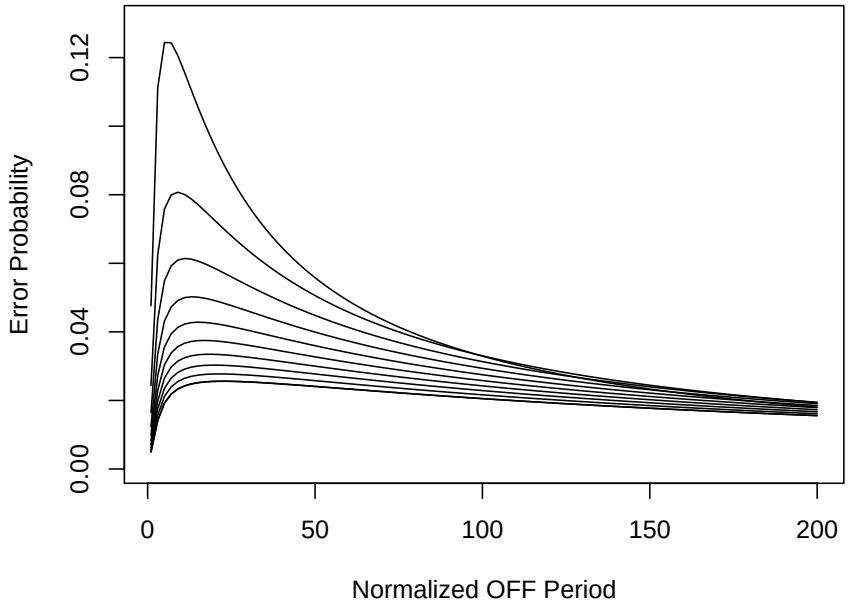


Figure 2.31. Optimal Error probability vs T_{off}, $20s \leq T_{on} \leq 200s$ and $T_{arr} = 1s$.

while the average μ and variance σ^2 are respectively:

$$\begin{aligned} \mu &= \frac{ab}{a-1} \\ \sigma^2 &= \frac{ab^2}{(a-1)^2(a-2)} \end{aligned}$$

The Pareto distribution is known to be an heavy tailed distribution, and depending on the value of a may have finite or infinite moments of order n. In particular, by selecting $a = 2$ we obtain a distribution with finite mean, but infinite variance. This choice yields to a larger probability that the inter-arrival of flows is comparable with the inter-arrival of sessions, therefore creating a stiffer scenario. To obtain the average value $T_{arr} = 1s$ we set $a = 2$ and $b = 0.5$.

We also examined Weibull distributed flow inter-arrivals to consider the case in which connection inter-arrival process exhibits a variance smaller than 1, but we do not report results for this distribution, since they are similar to those shown.

As performance metric, we use the percentage of misidentified sessions. Two types of errors are possible: (i) to erroneously separate a session in two or more clusters or (ii)

to merge two or more distinct sessions. The percentage of errors is then defined as 100 times the total number of errors observed divided by the total number of flow arrivals. All curves are averaged considering 50 different runs, each comprising 500 sessions (an average of 10000 flow arrivals per run).

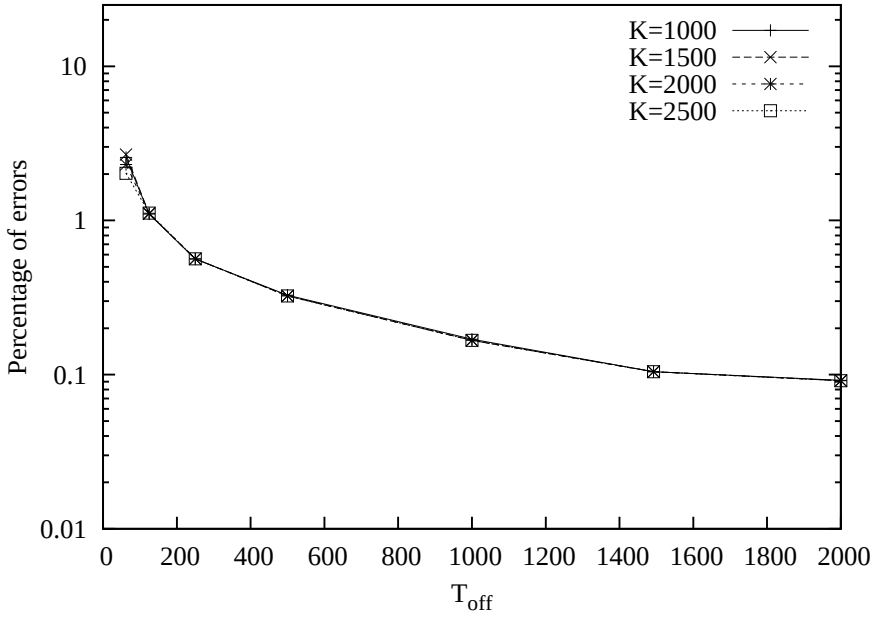


Figure 2.32. Sensitivity to the initial number of clusters K. Exponential flow inter-arrivals.

2.8.1 Parameter sensitivity

We start by evaluating the impact of the parameters given as input to the clustering methodology, namely the initial number of clusters K in the first phase, and the values of the percentile g during the hierarchical clustering. Considering the exponential flow inter-arrival scenario, in Fig. 2.32 we show that the error probability is practically independent from the value assumed for the initial number of clusters, which therefore is not a critical parameter (but it must be sufficiently larger than the supposed number of sessions). From now on we set $K = 1000$.

Fig. 2.33 shows instead the influence of the parameter g that determines the value of the percentile used to represent a cluster in the cluster-to-cluster distance. We report the single linkage method (which take the two extreme values in the sample distribution as representative, i.e., $g = 0$), the centroid method (which uses the average of the sample distribution) and curves for different values of g when using percentiles. Observe that the

best performance is obtained with the single linkage method. The centroid and the 45-th percentile methods exhibit the worst performance, and performance of percentile methods shows that small values of g are preferable. Indeed, in all the observed scenarios (i.e., different distributions) we got the best performance by using the single linkage method, which will therefore be the choice made in the remainder of the paper. This solves also the issue of setting a proper value for g, which again allows us to run the clustering scheme without being dependent, in terms of performance, by any external parameter.

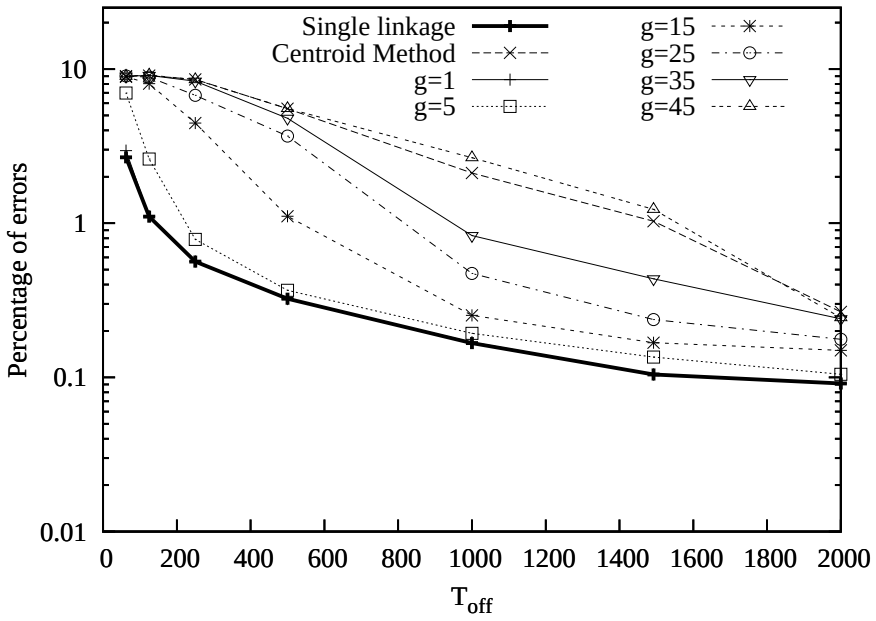


Figure 2.33. Sensitivity of the percentage of errors to the percentile g. Exponential flow inter-arrivals.

We now consider the percentage of misidentifies session to assess the quality of the results and to compare them with traditional threshold based approach. Results are reported in Fig. 2.34, where we show, as a function of the average OFF period T_{off}, the percentage of errors obtained using the proposed clustering scheme (three steps, initial number of cluster K set to 1000, single linkage hierarchical clustering) and the classical threshold schemes for variable values of the threshold η. Exponential flow inter-arrivals are reported in the top plot, Pareto flow inter-arrivals in the bottom plot; clustering scheme performance is shown with a solid black line.

For all schemes, performance obviously improves as T_{off} increases, since long silent period among user sessions is easily detected with respect to short silent period among flow arrivals within an ON period. Threshold based mechanisms may perform better, provided that the proper threshold value is chosen. However, if the threshold is not correctly selected, errors are much higher than those obtained when using the clustering scheme. In

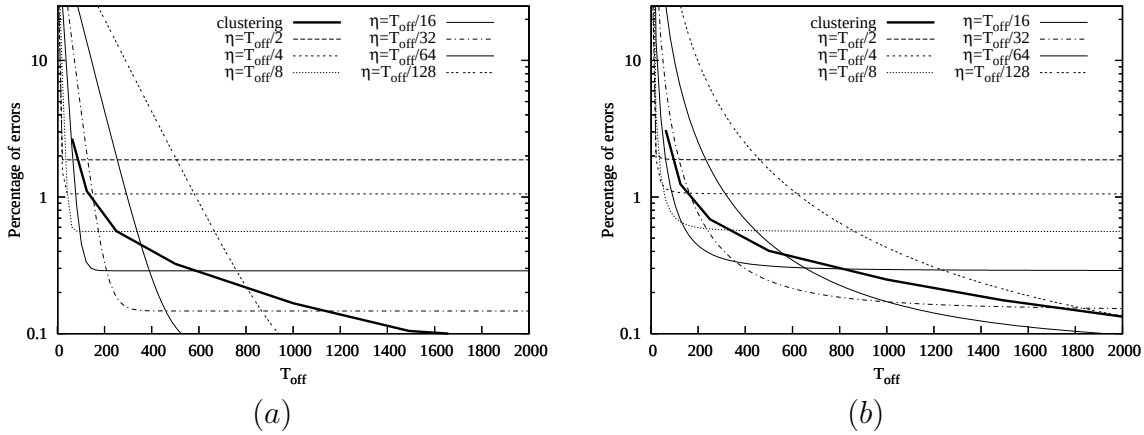


Figure 2.34. Percentage of errors. Exponential and Pareto flow inter-arrivals in top and bottom plot respectively.

general, large values of η exhibit a higher percentage of errors, due to the probability of erroneously merging two subsequent different sessions. On the contrary, a sharp increase in error probability occurs for small values of η when T_{off} goes below a given value, i.e., when T_{off} becomes closer to T_{arr}.

In general, the clustering scheme shows a percentage of errors never larger than 2%, and it is less sensitive to variations of T_{off}. Moreover, it does not require to set any parameter like the threshold value that can dramatically affect the error probability. Notice also that, when considering the Pareto distribution of flow inter-arrivals, no big differences are noticeable. Only an increase in the percentage of errors when using the threshold based approach for small values of T_{off} is visible. This is due to the increase in the probability that a flow inter-arrival is quite large, thus becoming comparable with the OFF duration. The clustering approach is less affected by such events.

2.8.2 Mean inter-arrival estimation errors

To gather the accuracy of the methodology in the estimation of the system parameters, Fig. 2.35 reports the percentage of errors when determining $E[T_{arr}]$, the mean connection inter-arrival time, and $E[T_{off}]$, the mean session OFF period duration as obtained after running either clustering algorithm (lines), or threshold based approach with different values of η (lines with points). Also in this case we consider the exponential scenario, in which T_{arr} is fixed to $1s$, and $T_{off} = 62.5s$. Errors are averaged over 5 runs each comprising 500 sessions.

It is immediate to notice that the clustering approach is very accurate in the estimation of both T_{off} and T_{arr}, exhibiting a relative error of about -1.4% in the estimation of T_{off},

and 4.4% in the estimation of T_{arr}. On the contrary, the threshold based approach is very sensitive to the choice of η. In particular, in the case $\eta < 5$, a larger number of session are identified, therefore under-estimating both T_{arr} and T_{off}, while for $\eta > 5$, an over-estimation of system parameters is due to the large number of erroneous merging of sessions.

In summary, the clustering approach proposed in this paper is less error prone than threshold based approaches during the identification of Web user-sessions, and does not require any ad hoc parameter settings which largely affects the performance of simple threshold-based approaches.

Results show that threshold based mechanisms are very sensitive to the choice of η. This makes their use very questionable, especially when considering real traffic datasets, as a bad choice of η may lead to large errors. We therefore avoid to report results using traditional threshold method and only adopt the novel technique we devised so far.

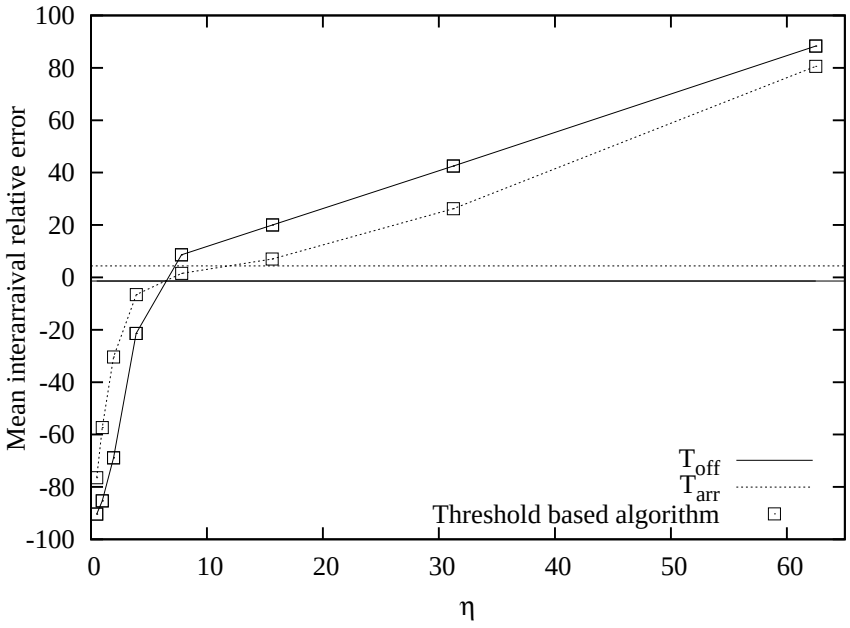


Figure 2.35. Percentage of errors on the estimation of the mean OFF session time (solid lines) and flow inter-arrival time (dotted lines). Clustering (no points) and threshold (square points) algorithms for different values of the threshold η for classic approach.

2.9 Performance Analysis of Trace Data Set

Recall that we determine user sessions by running the clustering algorithm for each user separately, according to the trace pre-processing algorithm previously described.

2.9.1 Web user-session characterization

In Figs. 2.36, 2.37, and 2.38 the major characteristics of the Web user-session identified during APR.04 are shown. The OCT.02 data show similar characteristics and are not reported. We plot Probability Density Functions (PDFs) using a linear/log scale: since the support is quite large, the PDF is represented only for a limited range of values, and the complementary Cumulative Distribution Function (CDF) is shown using a log/log scale in an inset to highlight the characteristics of the distribution tail.

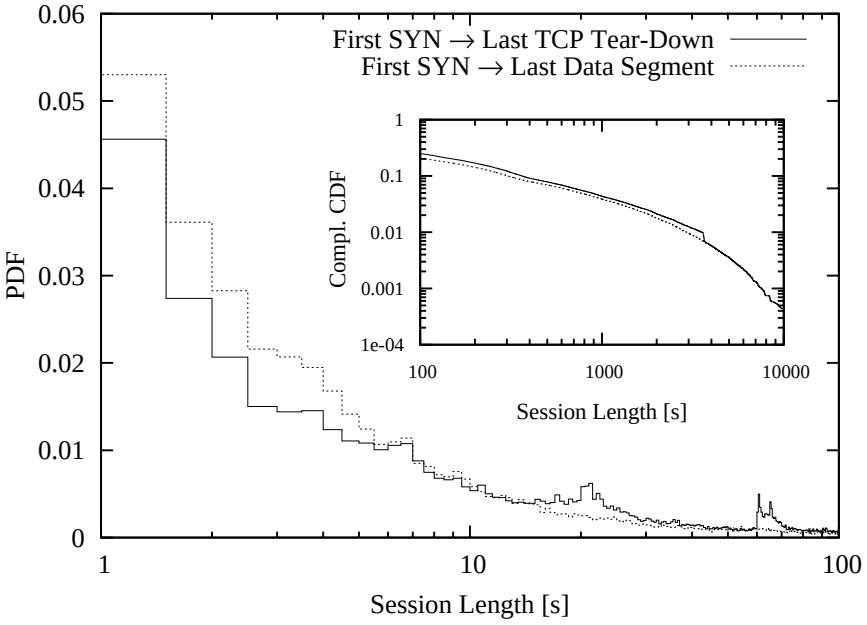


Figure 2.36. PDF (and Complementary CDF in the inset) of the session length.

Fig.2.36 shows the probability density function for the duration of the identified sessions. The two different distributions shown in Fig. 2.36 represent the effect of different definitions for the Web user-session. Considering TCP connections belonging to the same session, we define the session duration as the time between the first SYN segment of the first connection and: (i) for "protocol session", the last segment observed during the last connection tear-down (solid line); (ii) for "user" session, the last segment carrying payload of the last connection (dotted line). Therefore, using the notation introduced in Sec. 2.6.4, for a given session/cluster C, we can define

$$\begin{aligned} \Delta T_e &= \max_{f_{id} \in C}(t_e(f_{id})) - \min_{f_{id} \in C}(t(f_{id})) \\ \Delta T_d &= \max_{f_{id} \in C}(t_d(f_{id})) - \min_{f_{id} \in C}(t(f_{id})) \end{aligned}$$

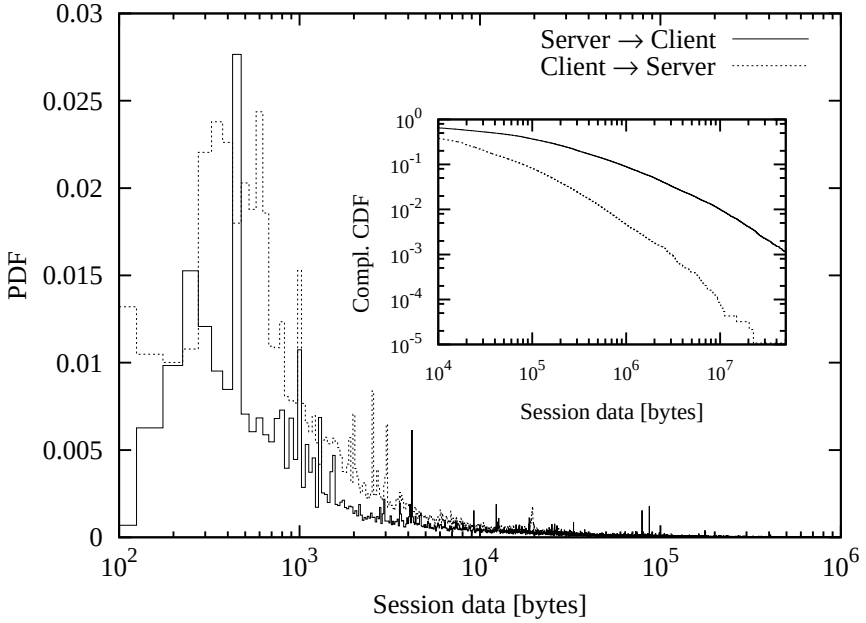


Figure 2.37. PDF (and Complementary CDF in the inset) of the client-to-server and server-to-client data sent in each session.

in which ΔT_e and ΔT_d are the protocol and user session duration respectively. The first definition is relevant for example when either web server or client resources are considered, since TCP connections must be managed until the tear-down procedure has ended. The second definition on the contrary is relevant to model the user behavior, since users are satisfied when all data have been sent/received.

Clearly, the distribution of the protocol session duration shows longer duration, but also biased peaks at $20s$, $60s$ and $3600s$, corresponding to application layer timers imposed by Web browsers or HTTP servers which trigger connection tear-down procedure after idle periods. For example, due to HTTP protocol settings, servers may wait for a timer to expire (usually set to 20 seconds) before closing the connection; similarly, HTTP 1.1 and Persistent-HTTP 1.0 protocols use an additional timer, usually set to a multiple of 60 seconds. Therefore, for protocol session duration, the bias induced by those timers is evident. The same bias disappears when the session duration is evaluated considering user session duration.

Notice that the session duration distributions have a quite large support, showing a great variability in users behavior. Indeed, there is a percentage of very short sessions (less than few seconds), but also users whose activities last for several hours. Indeed, the tail of the complementary CDF shown in the inset underlines the heavy-tailed distribution of session duration, which can be a possible cause of Long Range Dependence (LRD) at both the connection and packet layers.

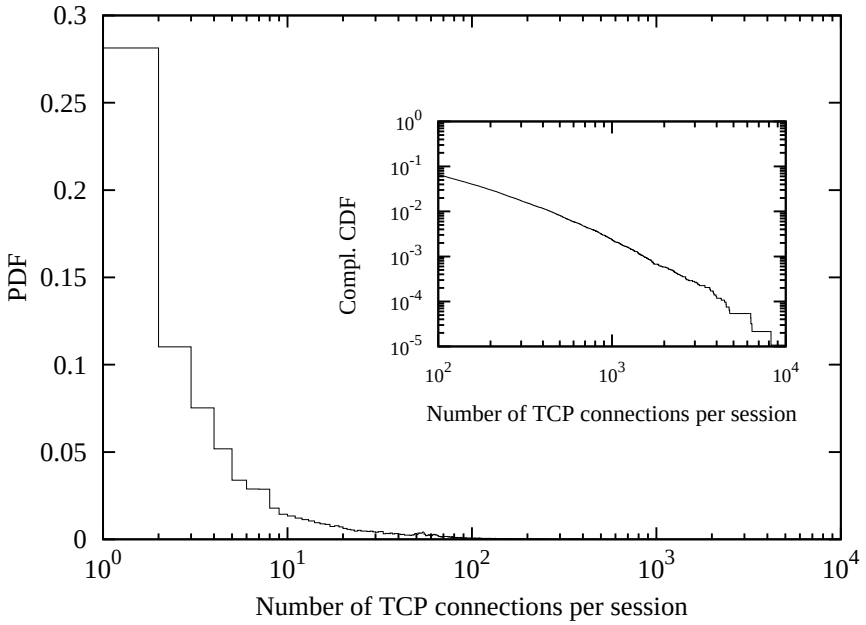


Figure 2.38. PDF (and Complementary CDF in the inset) of the number of TCP connection in each session.

In Fig. 2.37 we have the PDF for the volume of data exchanged during each session in the client-server (dashed lines) and the server-client (solid lines) directions : for a given session/cluster C, $D_c = \sum_{f_{id} \in C}(B_c(f_{id}))$ and $D_s = \sum_{f_{id} \in C}(B_s(f_{id}))$ define the session data volumes. As expected, much less data are transfered from clients to servers and the distribution tail is shorter; the number of sessions transferring more than 10 Mbytes in the server-client direction is not negligible. A peculiar number of peaks are present in the initial part of both PDFs. Investigating further, we discovered that those peaks are due to the identification of sessions which are not generated by users, but instead by automatic reload procedure imposed by the Web page being browsed. For example, news or trading on line services impose periodic updates of pages which causes the client to automatically reload the pages. If the automatic reload is triggered periodically, the clustering algorithm tends to identify for each connection a separate session, thus causing a bias in the session data distribution.

This is clearly evident also from Fig. 2.38, which reports the number of TCP connections per session $|C|$. Indeed, more than 25% of sessions count for only one connection. Moreover, most of the identified sessions is built by very few connections (about 50% by 4 connections or less), indicating also that i) the client is usually able to obtain all the required data over few TCP connections, ii) the number of external objects required is limited, and iii) the time spent by the users over one web page is large enough to define each web transaction as a session.

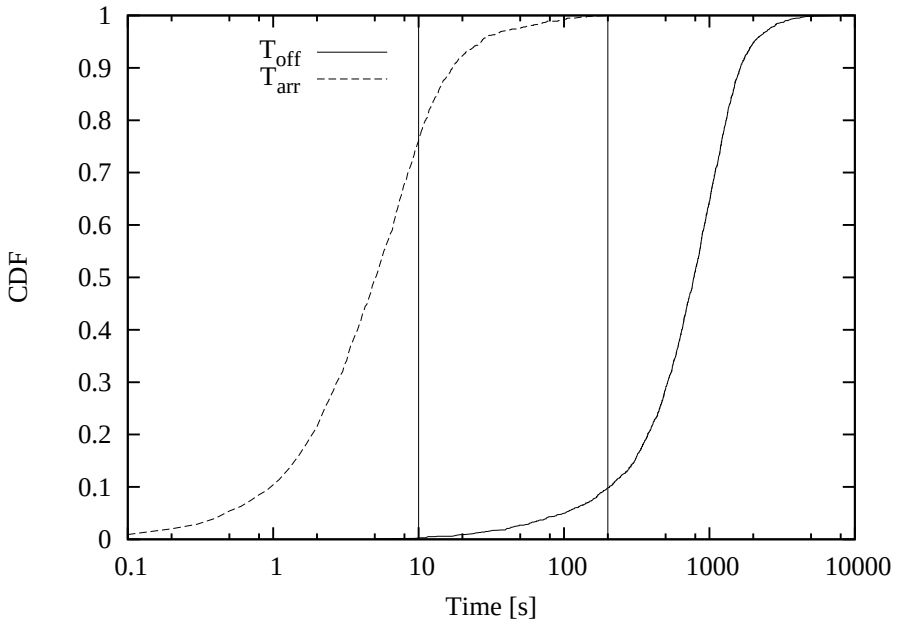


Figure 2.39. CDF of the mean user session inter-arrival and mean users flow inter-arrival.

The complementary distribution of the number of TCP connections per session reported in the inset of Fig. 2.38 shows a linear trend, highlighting that the distribution has an heavy tail. This could be one of the possible causes of LRD properties at the flow level, as already known.

Fig. 2.39 reports the CDF of the *average* user flow inter-arrival T_{arr} and average user session OFF period T_{off}, i.e., for each user, we evaluated the average T_{arr} and T_{off}, and then derived the corresponding distribution over users. Both OCT.02 and APR.04 dataset are considered, to show how similar are the results obtained during both time frame during which the session analysis was performed. As expected, T_{arr} assumes smaller values than T_{off}, but the two distributions overlap as underlined by the two vertical lines. This shows the variability in user behavior.

Notice that the overlapping of the two distributions would have not appeared if a threshold methodology had been applied, whichever adopted threshold. Moreover the variability of the mean T_{off} and mean T_{arr} makes it very difficult to select an appropriate values for η. This confirms the limits of threshold based approaches and the need of using clustering approaches that automatically adapt to different scenarios.

2.9.2 Statistical properties of session arrival process

Finally, statistical properties of the aggregate session inter-arrival times are investigated. We obtained a trace of session arrivals by multiplexing all sessions identified for each IP source address (user) during the same time period, i.e, by considering the Web user-session arrival process to the access router of our institution.

Fig. 2.40 reports the Q-Q plot of the aggregate session inter-arrival distribution with respect to the best fitted Weibull distribution over the same data set. The parameters a,b of the Weibull distribution represent the so called "shape" and "scale" parameters.

$$\mathbb{P}\{X > x\} = e^{-(\frac{x}{b})^a} \mathbb{1}_{(0,+\infty)}(x)$$

When the shape parameter is set to 1, the Weibull distribution degenerates into an exponential distribution. When it is smaller than 1, the tail of the distribution tends to be heavier, while for values of a larger than 1 the shape of the distribution tends to assume a dumbbell form. The classical maximum likelihood method was used to obtain the best a and b parameters for the fitting procedure. We embed the exponential distribution into a Weibull but a Gamma would also be possible.

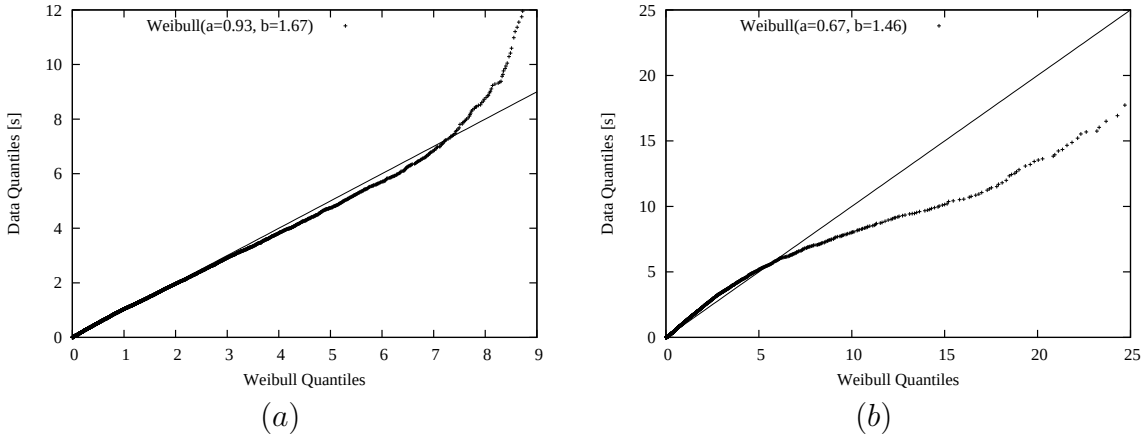


Figure 2.40. Fit of user session inter-arrivals to a Weibull distribution: normal day in the top plot and during a worm attack on the bottom plot.

The upper plot of Fig. 2.40 refers to a typical measurement day and shows a good matching of the data samples with the indicated Weibull distribution. Being $a = 0.93$, it also shows that the distribution is also very close to an exponential distribution, therefore hinting that the arrival process of Web user-session tends to be Poisson as pointed out by previous studies [CIWA02]. The Q-Q plot shows also that the tail of the distribution is in general less heavy than the tail of the fitted Weibull distribution. There is therefore a bias toward small values of session inter-arrivals, with large inter-arrivals which are rare. The

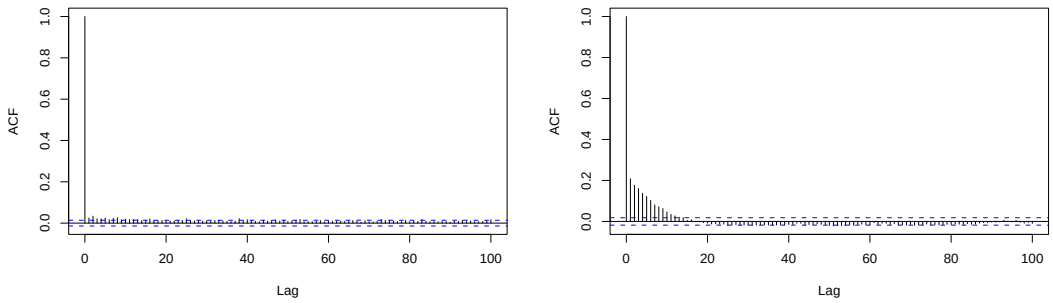


Figure 2.41. Auto correlation function of the inter-arrival process of sessions: normal day in the top plot and during a worm attack on the bottom plot.

Q-Q plot of the same data with respect to the best-fitted exponential distribution showed almost the same behavior.

On the contrary, the lower plot in Fig. 2.40, which refers to the samples collected on May 5th, 2004, shows a distribution that cannot be fitted by a Weibull distribution. The best fit is obtained by a shape parameters $a = 0.67$ indicating an heavy tailed distribution. Moreover, the best fit Weibull distribution deviates from the measured dataset on large quantiles, showing that the tail of the real distribution is heavier than the one of the best fit distribution. The anomalous behavior is due to the spreading of the Sasser.B worm [tSW04] in our institution, as previously described, and to the subsequent download of the needed OS patches and Anti-virus updates, which introduced correlation in the arrival process of sessions, being driven by the (large) download time of files.

To double-check that the anomalies were not introduced by our methodology, we also fit the session inter-arrival distribution as identified by a threshold procedure. The qualitative results are the same, even if the quantitative measurements (e.g., average inter-arrival time) are different depending on the threshold selected (as discussed in Fig. 2.35).

Fig.2.41 finally reports the autocorrelation function evaluated on the session inter-arrival obtained during a typical day on top plot, while bottom plot refers to the autocorrelation estimated during the anomalous day during the worm attack. Top plot confirms that Poisson assumption holds for normal day, being the autocorrelation function almost negligible except that in the origin. Similarly, the autocorrelation function among session inter-arrivals can be quite relevant on days during which user activities is driven by external factors, as shown by the bottom plot which refers to the day of the worm infection.

2.10 Conclusions

We propose an application of clustering techniques to a large set of real Internet traffic traces to identify Web user-sessions. We compared our methodology with the classical threshold based methods. The effectiveness and robustness of our clustering method has been assessed applying it to an artificial data set, showing its ability in the identification of user-sessions without requiring the a priori definition of threshold values.

The proposed clustering method has been applied to measurement trace data sets to study the characteristics of Web user-sessions, showing that session arrival process tends to be Poisson distributed. Moreover, the analysis of the identified user-sessions shows a wide range of behavior that cannot be captured by any threshold based method. We think that the clustering algorithms proposed in this work can be helpful in studying other traffic properties. We intend to apply them to different and type of traffic, and to extend the study on the arrival process statistical properties of the identified sessions.

Chapter 3

Traffic Models

3.1 Traffic Measurement and Analysis

In order to collect traffic traces, we observed the data flow on the Internet access link[1] of our institution, i.e., we focus on the data flow between the edge router of Politecnico di Torino and the access router of GARR/B-TEN [top05], the Italian and European Research network. For traces collection and processing we used `tcpdump` [MLJ05] and `Tstat` [Mel05, MCN05]. `Tstat` is a new software tool developed at Politecnico di Torino, which analyzes traces, and derives traffic characteristics at both the IP and TCP levels. For the analysis at the TCP level, `Tstat` rebuilds each TCP connection status by looking at the TCP header in the forward and backward packet streams. In order to do so, `Tstat` requires as input a trace collected on an edge node, such that both data segments on the forward stream and ACKs on the backward stream can be analyzed. When `Tstat` observes a TCP connection opening and closing, it marks the flow as *complete*, and proceeds by analyzing it. Additional information about `Tstat` and statistical analysis performed on collected traces can be found in [Mel05] and [MCN05].

The Politecnico LAN comprises approximately 7,000 hosts; most of them act as clients, but several servers are also regularly accessed from outside hosts. Data were collected on files storing 6 or 3 hours long traces (to avoid exceeding the File System limitation on the file size), for a total of more than 100 Gbytes of compressed data. Traces were collected during different periods, which correspond to different phases of the network topology evolution. In this paper, we present results considering two periods which are characterized by a significant upgrade in network capacity:

- April 2000, from 4/11/2000 to 4/14/2000: the bandwidth of the access link was 4 Mbit/s, and the link between the GARR network and the corresponding US peering was 45 Mbit/s

[1] The data-link level exploits an AAL-5 ATM virtual circuit (OC-3).

Table 3.1. Summary of the analyzed traces

Name	Date	Start time	Stop time	IP packets (10^6)	TCP flows (10^3)
Peak'01	2 Feb 01	10:52	13:52	11	540
Night'01	2 Feb 01	04:52	07:52	0.43	30
Peak'00	13 Apr 00	08:10	14:10	12	564
Night'00	13 Apr 00	02:10	08:10	0.92	79

- February 2001, from 2/1/2001 to 2/19/2001: the bandwidth of the access link was 16 Mbit/s, and of the link between the GARR network and the corresponding US peering was 622 Mbit/s.

The campus access link was a bottleneck during April 2000, while it was not during February 2001. The same consideration applies to the GARR-US peering capacity, which plays a key role, since most of the traffic comes from US research sites.

Among all the traces we collected, we report here results from four traces, which we consider representative of different network scenarios. Table 3.1 summarizes the key parameters of the selected traces; the last two columns report the number of samples in a trace, i.e., the number of IP packets and of TCP flows.

Since our campus network can be mainly considered as a "client" network, i.e., hosts in the network are mainly destinations of information, in the remaining of this paper we will present results considering incoming streams of data only, both at the TCP flow level and at the IP packet level.

3.2 Markovian Models

3.2.1 Markov Modulated Poisson Process: The MMPP Traffic Model

In order to derive a model which emulates the behavior of a given real trace, we now need to map the three model entities (sessions, flows and packets) into entities which can be observed in real traces. Packets and flows can be easily identified on the real trace; in particular, a flow is a single TCP connection, which starts by the three-way-handshake procedure and ends by the closing procedure[2]. On the contrary, it is more difficult to provide a specific and unique definition of a session. Indeed, many different definitions of a session could be proposed on the basis of traces. All the web pages downloaded by a user from the same web server in a limited period of time can form a session; a ftp connection from a user that requests many files from a server can form a session; all

[2]In case no packets are observed for more than 30 min, the flow is declared closed as well.

the e-mail messages generated by a user that replies to all the previously downloaded e-mails, or even the user activating its connection to the Internet (for example by switching on its computer), are all possible definitions of a session. Thus, we have the following problem. On the one hand, sessions are difficult to define, and to recognize in real traces. On the other hand, we need a notion of session in order to account for correlation over long time scales, which a model based on flows and packets alone cannot catch. We resort to defining a session as a generic set of correlated flows that are submitted to a network interface; then, we use the flow and packet levels of the model to fit the metrics which are easy to measure on real traces, and we specify the model session level so that the LRD of the real traces is accurately approximated. We now make the following assumptions

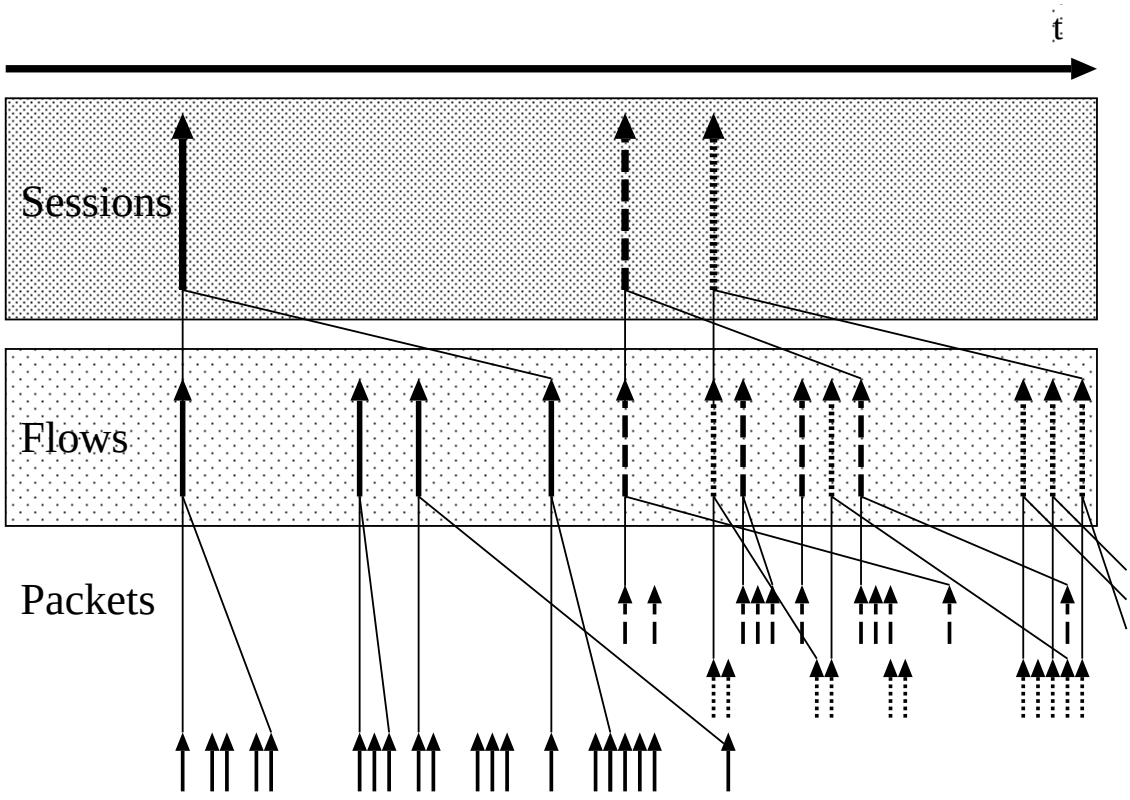


Figure 3.1. Sessions, flows, and packets as seen by the model

concerning the behavior of sessions, flows and packets:

- Sessions are generated according to a Poisson process with rate λ_s; each session starts by generating a new flow and ends when it generates the last flow belonging to the session.

- The number of flows generated by a session is a geometrically distributed random variable with mean value equal to N_f.
- Flows belonging to the same session are generated according to a Poisson process with rate λ_f; each flow starts by generating a packet and ends after having generated the last packet of the flow.
- The number of packets generated by a flow is a geometrically distributed random variable with mean value equal to N_p.
- Packets belonging to the same flow are generated according to a Poisson process with rate λ_p.

Due to the above assumptions, both the packet arrival process and the flow arrival process are MMPP, whose memory is given by two variables: the number of active flows, which accounts for short term correlation, and the number of active sessions, which determines correlation over long time lags. Observe that the model analysis can also use the same formalism used in [FH99]

3.2.2 Setting the Model Parameters

The MMPP model is completely described by five parameters:

λ_s : the arrival rate of new sessions

λ_f : the flow arrival rate per active session

λ_p : the packet arrival rate per active flow

N_f : the average number of flows per session

N_p : the average number of packets per flow.

In order to tune the model so that the generated synthetic traffic emulates the characteristics of a real trace, we need to properly set the model parameters. The parameters N_p and λ_p are simply set so as to match the packet and flow behaviors of a given trace; i.e., they are set equal to the measured average number of packets per flow and average packet arrival rate per flow. For what concerns N_f and λ_s, since they are related to the session behavior, they are harder to measure from traces. Thus, we set N_f and λ_s so as to match the Hurst parameters $\hat{H}_f$ and $\hat{H}_p$; this is done by means of an iterative procedure described below. Finally, given the session behavior, the parameter λ_f, the flow arrival rate per session, is simply set to match the average flow arrival rate observed from the traces. The fitting procedure is summarized in Fig. 3.2 (notice that, for our convenience,

1. From the traffic traces estimate $\hat{H}_p$, $\hat{H}_f$, $\hat{\Lambda}_f$, $\hat{\lambda}_p$, $\hat{N}_p$
2. Set $N_p = \hat{N}_p$ and $\lambda_p = \hat{\lambda}_p$
3. Set the initial values $N_f = 1$ and $C = 1$ (C is defined as $C = \lambda_s/\lambda_f$)
4. Compute $\lambda_s = \frac{\hat{\Lambda}_f}{N_f}$ and $\lambda_f = \frac{\lambda_s}{C}$
5. Generate a synthetic sequence with the same number of samples as the real trace
6. Estimate the Hurst parameter at both packet and flow level of the synthetic trace and compare them with $\hat{H}_p$, $\hat{H}_f$
7. If the fitting is good, the procedure ends else assign new values to N_f and C and go to 4

Figure 3.2. Fitting procedure to derive the MMPP model parameters from a measured trace

in the fitting procedure we normalize the session arrival rate λ_s to the flow arrival rate per session λ_f, and denote the normalized arrival rate by C):

The selection of N_f and C by means of the fitting procedure at steps 4–7 can be performed according to different definitions of accuracy of the fit and to different criteria for the assignment of new values to the parameters. The detailed procedure which we followed is reported in Fig. 3.3.

$C = 1$; $N_f = 1$;
$eps_f = eps_p = 0.05$; fit = 0;
while !(fit){
  generate a synthetic sequence and estimate H_f,H_p;
  fit = $(|H_f - \hat{H}_f| < eps_f)$ &&$(|H_p - \hat{H}_p| < eps_p)$;
  if $(H_f < \hat{H}_f)$ then $N_f = N_f + 5$;
  else if $(H_f > \hat{H}_f)$ then $N_f = N_f - 1$;
  if $(H_p > \hat{H}_p)$ then $C = 3 * C$;
  else if $(H_p < \hat{H}_p)$ then $C = C/2$;
}

Figure 3.3. Selection of new parameters in the iteration

The criteria to assign new values to N_f and C were chosen after having studied the sensitivity of the Hurst parameters H_f and H_p to changes of N_f and C by means of

Table 3.2. Flow level analysis of traces

Trace	Peak'01			Night'01		
	$\hat{H}_f$	$\hat{c_f}$	$1/\hat{\Lambda}_f$	$\hat{H}_f$	$\hat{c_f}$	$1/\hat{\Lambda}_f$
$I\ [ms]$	0.74	82.4	20.01	0.86	8491	358.9
N_{1s}	0.76	59.4	49.9	0.76	1.66	2.79
N_{100ms}	0.75	2.01	4.99	0.73	0.07	0.28
N_{10ms}	0.74	0.07	0.49	0.80	0.001	0.028
Trace	Peak'00			Night'00		
	$\hat{H}_f$	$\hat{c_f}$	$1/\hat{\Lambda}_f$	$\hat{H}_f$	$\hat{c_f}$	$1/\hat{\Lambda}_f$
$I\ [ms]$	0.76	275.1	39.6	0.74	2414	271.6
N_{1s}	0.75	28.4	25.9	0.78	1.54	3.68
N_{100ms}	0.74	0.79	2.53	0.76	0.06	0.37
N_{10ms}	0.75	0.015	0.25	0.78	0.001	0.04

Table 3.3. Packet level analysis of traces

Trace	Peak'01			Night'01		
	$\hat{H}_p$	$\hat{c_p}$	$1/\hat{\Lambda}_p$	$\hat{H}_p$	$\hat{c_p}$	$1/\hat{\Lambda}_p$
$I\ [ms]$	0.87	0.01	0.89	0.71	$5 \cdot 10^{-4}$	0.025
N_{1s}	0.88	5232	1113	0.73	457.8	40.06
N_{100ms}	0.88	91.4	111.3	0.72	16.6	4.00
N_{10ms}	0.88	1.50	11.1	0.76	0.30	0.4
Trace	Peak'00			Night'00		
	$\hat{H}_p$	$\hat{c_p}$	$1/\hat{\Lambda}_p$	$\hat{H}_p$	$\hat{c_p}$	$1/\hat{\Lambda}_p$
$I\ [ms]$	0.84	0.17	2.25	0.84	40.54	15.74
N_{1s}	0.86	504.19	444.75	0.83	133.93	63.51
N_{100ms}	0.87	14.50	44.49	0.83	3.61	6.35
N_{10ms}	0.88	0.23	4.45	0.87	0.04	0.63

the graphs shown in Fig. 3.4. The Hurst parameter at both the flow and packet levels increases as N_f increases, consistently with the intuition that a larger value of N_f introduces a higher degree of memory in the system. Moreover, at the packet level, there is a higher degree of memory and correlation since packets are generated by flows which are generated by sessions. Let us now focus on C. The larger C is, the more bursty the generation of flows per session is. The influence of C on the Hurst parameter is quite complex. At the flow level, a higher degree of burstiness tends to induce a larger value of the Hurst parameter, while the opposite is true at the packet level.

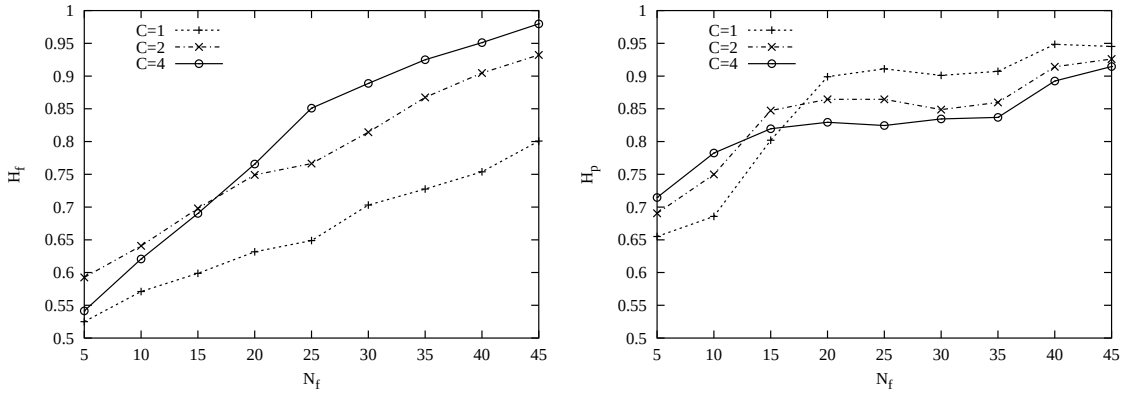


Figure 3.4. Impact of N_f on the Hurst parameters of the flow arrival process, for different values of the normalized session arrival rate $C = \lambda_s/\lambda_f$

The fitting procedure requires only a few iterations (approximately 10 in our tests). In order to verify that the synthetic traffic generated by the model accurately emulates the real traces, we report in Table 3.4 the characteristics of the synthetic traces measured by the *AV* tool when the model parameters are fitted to the '01 traces; the values must be compared with those of Tables 3.2. Observe that, thanks to the fitting procedure, the Hurst parameters are well matched, while the values for the c parameters are less accurate, though the qualitative behavior is the same.

Table 3.4. Model: flow and packet level results fitting the '01 traces

Trace	Peak'01			Night'01		
	H_f	c_f	$1/\Lambda_f$	H_f	c_f	$1/\Lambda_f$
$I\ [ms]$	0.74	19.2	20.1	0.84	7799	356.8
N_{1s}	0.71	76.7	49.8	0.82	0.51	2.81
N_{100ms}	0.71	2.27	4.98	0.83	$7.6 \cdot 10^{-3}$	0.28
N_{10ms}	0.78	0.013	0.50	0.79	$1 \cdot 10^{-4}$	0.03
Trace	Peak'01			Night'01		
	H_p	c_p	$1/\Lambda_p$	H_p	c_p	$1/\Lambda_p$
$I\ [ms]$	0.84	0.038	0.90	0.82	$1 \cdot 10^{-4}$	0.025
N_{1s}	0.87	5943	1113	0.87	35.2	40.06
N_{100ms}	0.82	178.2	111.4	0.79	2.05	4.01
N_{10ms}	0.84	3.13	11.14	0.86	$5 \cdot 10^{-3}$	0.41

3.2.3 The Modulating Markov Chain

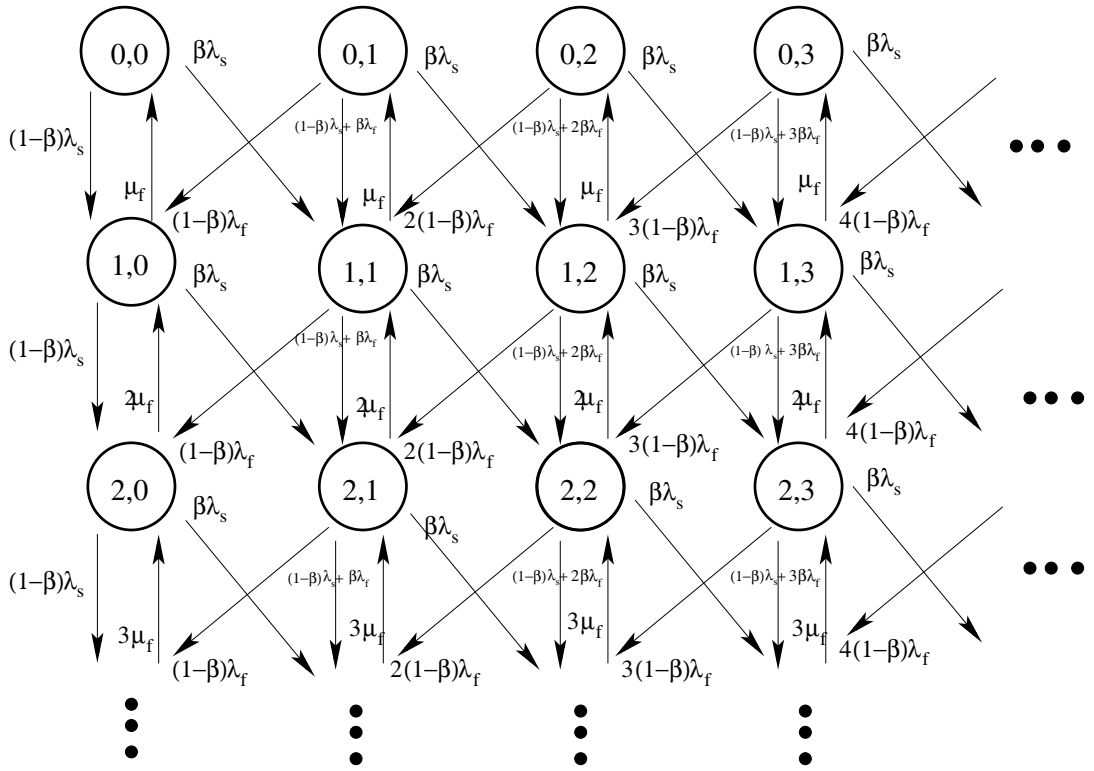


Figure 3.5. State-transition diagram of the Continuous-Time Markov Chain that modulates arrivals

The Continuous-Time Markov Chain (CTMC) which modulates the packet and flow arrival processes is defined by the state variable $\overline{s} = (n_f,n_s)$, where n_f and n_s denote the number of active flows and the number of active sessions, respectively. A state-transition diagram of the CTMC model of the modulating process is reported in Fig. 3.5.

The transition from state (n_f,n_s) to state $(n_f - 1,n_s)$ corresponds to the termination of a flow; its rate is $n_f\mu_f$, where $\mu_f = \lambda_p/(N_p - 1)$. The generation of a new flow makes the chain move from state (n_f,n_s) to state $(n_f + 1,n_s)$ with rate $\beta n_s\lambda_f$ if the flow is not the last one of the session, and to state $(n_f + 1,n_s - 1)$ with rate $(1 - \beta)n_s\lambda_f$ if the flow is the last one. The probability β that a generated flow is the last one of a session is given by $\beta = 1 - 1/N_f$. In state (n_f,n_s) a session starts with rate λ_s and generates a new flow. If the session is composed of one flow only, the chain moves from (n_f,n_s) to $(n_f + 1,n_s)$ with rate $(1 - \beta)\lambda_s$; otherwise, the chain moves from (n_f,n_s) to $(n_f + 1,n_s + 1)$ with rate $\beta\lambda_s$.

The infinitesimal generator of the CTMC is infinite due to the unbounded values that n_s and n_f can take. Thus, in order to analyze the property of the MMPP or to evaluate the

performance of a queue fed by the synthetic traffic generated by the MMPP we have two alternatives: i) we resort to simulation, ii) we truncate the CTMC so that the infinitesimal generator matrix becomes finite. In what follows we consider both cases. The criterion used for truncating the CTMC is described in Section 3.2.4. For the moment, assume that the CTMC has been truncated so that the number of active flows varies in the range $[f_m,f_M]$ and the number of active sessions varies in $[s_m,s_M]$.

We denote by $\{J(t),t \in \mathbb{R}^+\}$ the finite CTMC obtained by truncating the MMPP according to the above ranges. The state space is given by

$$S = \{(n_f,n_s) \in \mathbb{N}^2, f_m \leq n_f \leq f_M, s_m \leq n_s \leq s_M\}$$

The infinitesimal generator $Q \in \mathbb{R}^{n \times n}$ with $n = (f_M - f_m + 1) \cdot (s_M - s_m + 1)$ is given by,

$$Q = \begin{pmatrix} Q_{f_m} & Q^+ & 0 & \cdots & 0 \\ (f_m+1)Q^- & Q_{f_m+1} & Q^+ & \cdots & \vdots \\ 0 & (f_m+2)Q^- & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & \ddots & Q_{f_M-1} & Q^+ \\ 0 & \cdots & \cdots & f_M\, Q^- & Q_{f_M} \end{pmatrix}$$

$Q^- = \mu_f\, I_s$,
$Q^+ = (1-\beta)\lambda_f\, N_s^- + \beta\lambda_f\, N_s + (1-\beta)\lambda_s\, I_s + \beta\lambda_s\, I_s^+$

$$I_s^+ = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & 0 & 1 & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \quad N_s^- = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ s_m+1 & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & s_M & 0 \end{pmatrix}$$

$I_s, I_s^+, N_s, N_s^- \in \mathbb{R}^{(s_M - s_m + 1) \times (s_M - s_m + 1)}$, I_s is the identity matrix. $I_s^- = I_s^{+T}$
$N_s = diags\{s_m, s_m+1, \cdots, s_M\}$ [3],
$Q_i = diags\{-(s_m\beta\lambda_f + \lambda_s + i\,\mu_f), -[(s_m+1)\lambda_f + \lambda_s + i\,\mu_f], \cdots, -[(s_M-1)\lambda_f + \lambda_s + i\,\mu_f], -[s_M\lambda_f + (1-\beta)\lambda_s + i\,\mu_f]\}$, for $f_m < i < f_M$
$Q_{f_m} = diags\{-(s_m\beta\lambda_f+\lambda_s), -[(s_m+1)\lambda_f+\lambda_s], \cdots, -[(s_M-1)\lambda_f+\lambda_s], -(s_M\lambda_f+\lambda_s)\}$
$Q_{f_M} = diags\{-f_M\mu_f, -f_M\mu_f, \cdots, -f_M\mu_f\}$. Q is an homogeneous irreducible infinitesimal generator if $\beta \neq 0,1$. The steady-state probability vector π is given by

$$\pi Q = 0, \ \pi\, e = 1$$

where $e = (1,1,\cdots,1)^T$. The rate matrix is $\Lambda \in \mathbb{R}^{n \times n}$, with $\Lambda = \lambda_p\, diags\{f_m\, I_s, (f_m+1)\, I_s, (f_m+2)\, I_s, \cdots, f_M\, I_s\}$.

[3] The operator $diags\{x_1, x_2, \cdots, x_n\}$ defines a diagonal matrix in $\mathbb{R}^{n \times n}$ whose elements along the diagonal are given by $x_1, x_2, \cdots, x_n$.

3.2.4 Truncating the Modulating CTMC

To truncate the modulating CTMC, we have to trade-off between the opposite needs of i) reducing the dimension of the CTMC as much as possible in order to make the solution efficient and fast, and ii) keeping the CTMC dimension large enough so that the truncated chain accurately approximates the original infinite one. In order to find a proper truncation criterion, we first discuss the marginal distributions of n_s and n_f.

Let us focus on the number of active sessions, n_s. Sessions are generated according to a Poisson process with rate λ_s. The lifetime of a session, i.e., the time a session spends in the system, is given by the sum of the interarrival times of the flows generated by the session. Interarrival times between flows of a session are i.i.d. negative exponential random variables X with rate λ_f. The lifetime Y of a session is thus given by,

$$Y = \sum_{i=0}^{\infty} \beta^i (1-\beta) X_i \tag{3.1}$$

where $\{X_i\}_{i\in\mathbb{N}}$ is a sequence of i.i.d. exponentially distributed random variables. It can be easily shown that Y is negative exponential distributed with mean value $E[Y] = E[X]\beta/(1-\beta) = \beta/[\lambda_f(1-\beta)]$. Since sessions are independent from each other, the evolution of the number of sessions in the system can be modeled by an M/M/∞ queue, whose arrival rate is λ_s and whose mean service time is $E[Y]$. It follows that the distribution of n_s is Poisson with parameter $\delta = \lambda_s\, E[Y] = \lambda_s\beta/[\lambda_f(1-\beta)]$.

We now consider the number of active flows, n_f. In order to model the flow arrival process, we number flows according to the order in which they are generated by the session they belong to: A flow of type i is the $i-$th flow generated by a session. We model the flow arrival process by the infinite queuing network reported in Fig. 3.6.a, where arrivals at queue i represent the arrivals of type i flows. The service time of a customer in queue i represents the interarrival time between the i-th flow and the $(i+1)-$th flow of a session and is distributed according to a negative exponential distribution with rate λ_f. Since type 1 flows are generated at the arrival of sessions, type 1 flows arrive at queue 1 according to a Poisson process with rate λ_s. A customer leaving queue 1 enters queue 2 with probability $(1-\beta)$, which is the probability that the flow was not the last one of the session. Since the departure process from an infinite queue with Poisson arrivals is Poisson too, the arrival process at queue 2 is Poisson with rate $(1-\beta)\lambda_s$. Thus, the arrival process at queue i is Poisson with rate $\beta^{i-1}\lambda_s$. This queuing network is equivalent to the queue with feedback shown in Fig. 3.6.b. Notice that, as is well known, despite the arrival processes at all queues in Fig. 3.6.a are Poisson, the infinite sum of the arrival processes which is the process entering the queue with feedback in Fig. 3.6.b is not Poisson.

Since the lifetime of flows in the system is negative exponential distributed with rate μ_f, the behavior of flows can be described by a set of infinite M/M/∞ queues, where the arrival rate at queue i is equal to $(1-\beta)^{i-1}\lambda_s$ and the average service time is given by

$1/\mu_f$; or equivalently by a queue with feedback as the one in Fig. 3.6.b with μ_f instead of λ_f. It results that the distribution of n_f, the number of flows in the system, is Poisson distributed with rate $\lambda_s/(\beta\mu_f)$. Given that the marginal distributions of n_s and n_f is Poisson distributed, we can truncate the infinitesimal generator choosing (s_m,s_M) and (f_m,f_M) such that

$$\sum_{k\in\{s_m,\cdots,s_M\}} p_{n_s}(k) = 0.9 \tag{3.2}$$

$$\sum_{k\in\{f_m,\cdots,f_M\}} p_{n_f}(k) = 0.9 \tag{3.3}$$

where $p_{n_s}(k)$ and $p_{n_f}(k)$ are the probability density functions of n_s and n_f.

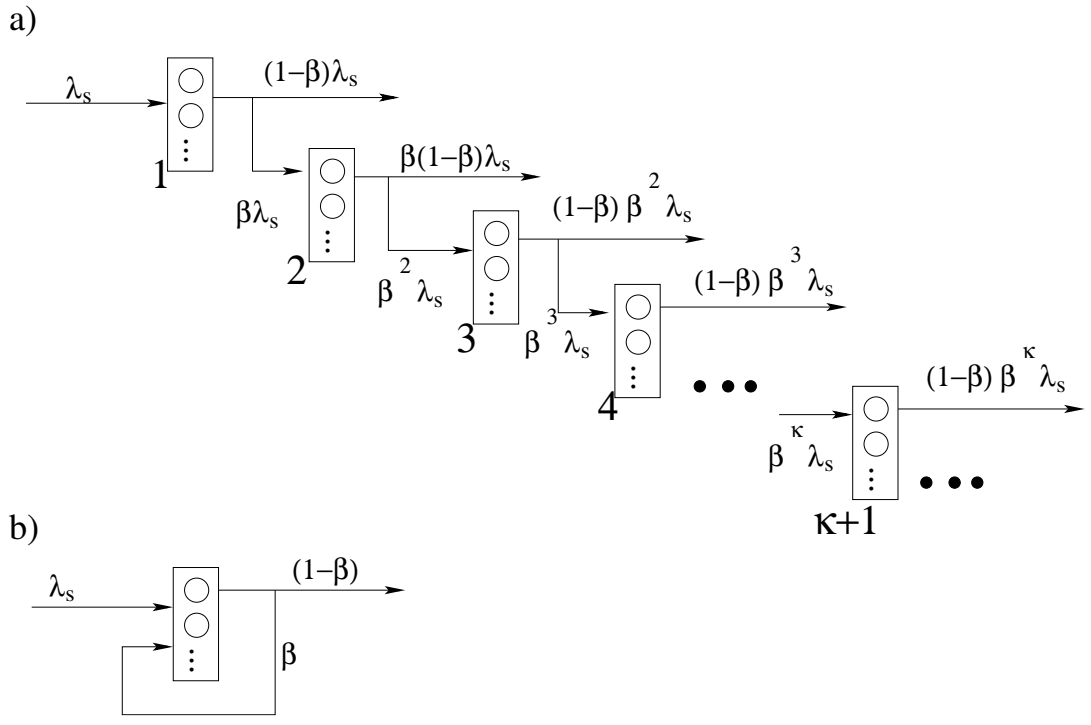


Figure 3.6. Sessions, flows, and packets as seen by the model

3.3 The Buffer Model (MMPP/M/1/m Queue)

For network planning and dimensioning, we are typically interested in the performance of a queue which represents the bottleneck of a network.

Besides the advantages of being simple to implement and efficient, a synthetic Markovian source as the one we propose has the additional advantage of allowing a Markovian model of a queue.

In general, the buffer model can be described by a MMPP/GI/1/m queue, where the service time represents the transmission time of a packet, and can be easily derived from the capacity of the link and the distribution of the packet length. The general service time distribution can be approximated by a phase type distribution. However, for simplicity, we consider the case of exponentially distributed service time; we validate the exponential assumption in Section 3.3.1 and we discuss the extension to phase type service times at the end of this section.

By adopting an exponential service time distribution, we obtain an MMPP/M/1/m queuing system. The stochastic process which describes the dynamic of the system is a vector process $\{Z(t),t \in \mathbb{R}^+\}$, $Z(t) = \{R(t),J(t)\}$, where $R(t)$ denotes the number of packets in the queue, $J(t) = (n_f(t),n_s(t))$ denotes the phase of the modulating chain and is a vector Markov process too.

The state space of the considered Markov process is $S = \{z = (r,n_f,n_s) \in \mathbb{N}^3$ with $0 \leq r \leq m$, $f_m \leq n_f \leq f_M$ and $s_m \leq n_s \leq s_M\}$. The infinitesimal generator of such a CTMC is A,

$$A = \begin{pmatrix} A_{00} & A_0 & 0 & \cdots & \cdots & 0 \\ A_{10} & A_1 & A_0 & \ddots & \cdots & \vdots \\ 0 & A_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & \ddots & A_2 & A_1 & A_0 \\ 0 & \cdots & \cdots & 0 & A_2 & A_{mm} \end{pmatrix}$$

where $A_{00} = Q - \Lambda$, $A_0 = \Lambda$, $A_1 = Q - \mu\ I_n - \Lambda$, $A_{mm} = Q - \mu\ I_n$,$A_2 = \mu\ I_n$. A_{00},A_{10},A_0,A_1,A_2,A_{mm},$I_n \in \mathbb{R}^{n\times\ n}$, and I_n is the identity matrix.

$Z(t)$ is a finite QBD (Quasi Birth Death) process whose solution has been studied in many papers, see for example [FH89, LR93, KNW98, BM95]. We adopt the solution proposed by [KNW98], the Improved Logarithmic Reduction Algorithm (ILRA). Let $\rho = \pi\ \Lambda\ e/\mu$ and let the steady-state distribution be $\hat{\pi}$. We are interested in the buffer length distribution π_r^q which can be obtained from [KNW98],

$$\pi_r^q = \hat{\pi}_r\ e = \lim_{t\to\ \infty} \mathbb{P}\{R(t) = r\}\ \ r = 0,\cdots,m$$

$$\hat{\pi}_r = x_0\ R^r + x_m\ S^{m-r}[I_n - (SR)^r] \tag{3.4}$$

where R,S are two matrix geometric terms and (x_0,x_m) are the solution of the two boundary conditions of the QBD finite process:

$$\hat{A}^{[0,m]} = \begin{pmatrix} \hat{A}_{00}^{[0,m]} & \hat{A}_{0m}^{[0,m]} \\ \hat{A}_{m0}^{[0,m]} & \hat{A}_{mm}^{[0,m]} \end{pmatrix}$$

$$\begin{aligned}
\hat{A}_{00}^{[0,m]} &= Q - \Lambda + A_0 G(R) && (3.5)\\
\hat{A}_{0m}^{[0,m]} &= R^m A_0 [I_n - G(R)] && (3.6)\\
\hat{A}_{m0}^{[0,m]} &= S^{m-1}[I - SR]A_2 && (3.7)\\
\hat{A}_{mm}^{[0,m]} &= W(S) + A_0 - (SR)^m\, A_0[I_n - G(R)] && (3.8)\\
G(R) &= [-(A_1 + R\, A_2)^{-1}]A_2\\
W(S) &= A_1 + S\, A_0
\end{aligned}$$

It can be shown that $\hat{A}^{[0,m]}$ is an infinitesimal generator of a CTMC with (x_0,x_m) as a steady-state vector,

$$(x_0,x_m)\hat{A}^{[0,m]} = 0,\ (x_0,x_m)\begin{pmatrix} B_1 \\ B_2 \end{pmatrix} e = 1$$

$$\begin{aligned}
B_1 &= I_n + R\sum_{r=0}^{m-1} R^r\\
B_2 &= \sum_{r=0}^{m-1} S^r - S^m R \sum_{r=0}^{m-1} R^r
\end{aligned}$$

If $\rho < 1$, then $\sum_{r=0}^{m-1} R^r = (I_n - R)^{-1}(I_n - R^m)$, else, if $\rho > 1$, $\sum_{r=0}^{m-1} S^r = (I_n - S)^{-1}(I_n - S^m)$, and the boundary conditions can be simplified taking into account the load value. The two terms R and S are two matrix geometric terms which are the minimal nonnegative solution of the two equations:

$$0 = R^2 A_2 + RA_1 + A_0 \tag{3.9}$$

$$0 = S^2 A_0 + SA_1 + A_2 \tag{3.10}$$

Observe that (3.4) can be extended to the infinite buffer case, which is the well known matrix geometric solution,

$$\hat{\pi}_r = x_0\, R^r,\ r \geq 0 \tag{3.11}$$

with boundary conditions:

$$x_0 \hat{A}_{00}^{[0,m]} = 0$$

$$x_0[I - R]^{-1} e = 1$$

The solution of (3.4) and (3.11) becomes more and more difficult as the matrix dimension increases; thus, the use of efficient iterative methods is a must [BBC+94]. The solution of the finite buffer case is more complex than that of the infinite case, in terms of both memory requirements and computation time. Indeed, the finite case needs the computation of two matrix geometric terms (instead of one) and the solution of a linear system twice as large as in the infinite case. However, despite its complexity, the finite buffer case must be solved in many cases of practical interest as, for example, for dimensioning the buffer size, in which case we typically need to compute the loss probability due to buffer

overflow. A nice feature of the method in [KNW98] is that the loss probability due to the buffer overflow can be directly evaluated through the recursive formula,

$$\pi_m^{q,(m)} = \pi^t(I_n - V_m) \tag{3.12}$$

$$\begin{aligned} \Psi &= Q - e\pi^t && (3.13)\\ V_1 &= I_n - [I_n + \mu(Q-\Lambda)^{-1}e\pi^t && (3.14)\\ &\quad +\mu(Q-\Lambda)^{-1}Q\Psi^{-1}]^{-1} && (3.15)\\ V_m &= A_2(-A_1^{-1})[I - V_{m-1}A_0(-A_1^{-1})]^{-1} && (3.16) \end{aligned}$$

This recursive formula is very useful because the computation of the loss probability for the case of buffer size equal to m provides, as a side product, the loss probability observed for all values of the buffer size smaller than m; thus, it is very efficient for network dimensioning.

As previously mentioned, the above problem can be easily extended to consider phase type distributions of the service time instead of exponential distributions. Such an extension requires the introduction of a random process component which accounts for the service phase. However, the increase of the dimension of the involved matrices makes the MMPP/Ph/1/m queue hard to solve in terms of time and memory requirements. Thus, in what follows we consider only exponentially distributed service times.

3.3.1 Performance Evaluation

The comparison of Table 3.4 with Tables 3.2 and 3.3 indicates that the proposed MMPP model captures the LRD characteristics of the traffic we measured at the edge router interconnecting the Politecnico di Torino network to GARR/B-TEN. This is hardly a surprise, since we tuned the model to obtain this result. It however indicates that the proposed simple MMPP model is capable of exhibiting LRD behaviors over the time scales of interest.

In this section we further evaluate the performance of the proposed MMPP model in two ways: i) we study the behavior of the synthetic traffic produced by the MMPP model when feeding a buffer in front of a transmission link, and compare the results against those produced by measured traffic traces, ii) we investigate the predictability and tunability of the model when used as a synthetic source of aggregate Internet traffic, i.e., we discuss the model effectiveness in representing different traffic scenarios.

The analysis in this section considers the 2001 traces, for which the values of the measured parameters are reported in Table 3.5.

For the evaluation of the queuing performance of the traffic generated by our MMPP model, we consider three different network topologies. Topology T1 comprises a single link and a single source; topology T2 aggregates four different sources on a single link;

Table 3.5. Measured values from real traces used in setting the model parameters

	Peak '01	Night '01
$\hat{N}_p$	22.03	14.35
$\hat{\lambda}_p$	29.57	352.9
$\hat{\Lambda}_f$	49.95	2.78
$\hat{\Lambda}_p$	1113	40.06

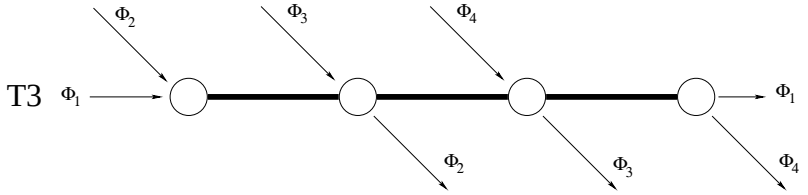


Figure 3.7. The third topology T3.

topology T3, shown in Fig. 3.7 is a tandem network with three links and four traffic relations, arranged so as to aggregate two traffic relations on every link, with relation Φ_1 crossing all three links, and all other relations interfering with it on a different link.

We test the queuing behavior by simulation using "*ns-2*" comparing the behavior of the synthetic traces to that of the original measured traces. To avoid useless repetitions, we report the results for the Peak'01 trace only, since all other results are equivalent. When the measured traces are used, the service times at nodes depend on the measured packet length. The MMPP traffic model, instead, generates only arrival instants, and the packet length, determining the service time at the nodes, is randomly associated with packets. The packet length value is drawn from a distribution fitted on the measured one, which exhibits the well-known multi-mode behavior, with peaks for very short packets and for the different MTUs (Maximum Transfer Units) in the network. The peak of the Ethernet frame at 1,500 bytes dominates the lot. Notice that this procedure destroys all possible correlation between the arrival process and the packet length distribution.

To simplify reading the plots (unless otherwise stated), we normalize the link load, changing the link speed appropriately. Results at different load levels or with variable loads on different links in topology T3 yield similar results.

Queueing behavior is much more interesting (and realistic) in the finite buffer case. We now consider the three topologies with five different values of buffer size: $B = 32$, 64, 128, 256, and 512 packets. Fig. 3.8 refers to the T1 and T2 topologies. The matching is satisfactory, though the trace-driven simulations always show a higher probability of full buffer, that is reflected in a slightly higher loss rate. The Poisson driven simulations refer only to $B = 32$, and show how inaccurate is a simple Poisson model, even for a very short buffer size. All other curves for Poisson arrivals are practically coincident with

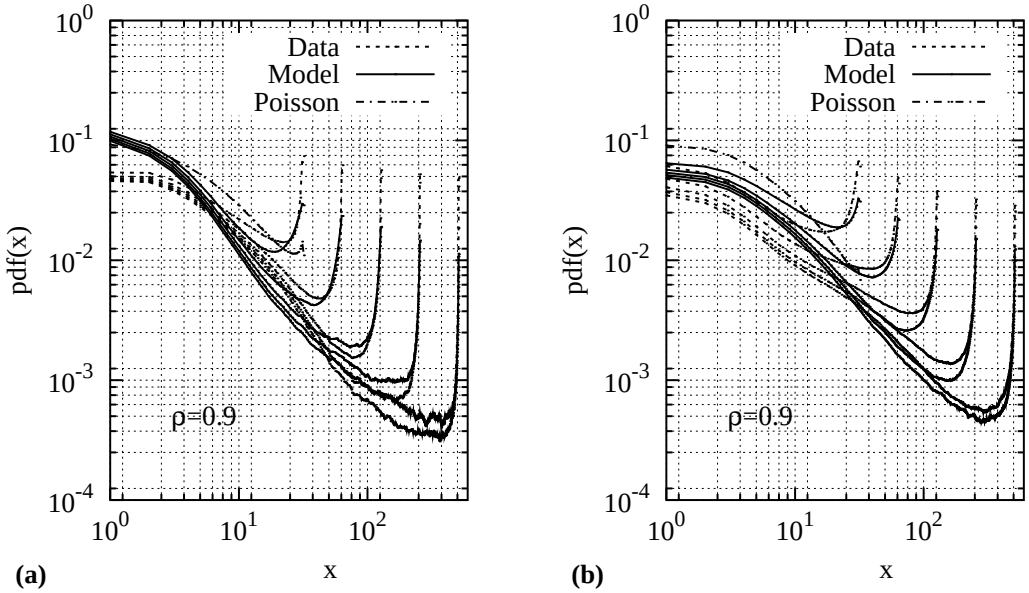


Figure 3.8. Buffer occupancy distribution, $B = 32, 64, 128, 256, 512$ packets; **(a)** topology T1 and **(b)** topology T2.

this one, and are not reported to avoid cluttering the graph. Multiplexing traffic does not change the situation, and actually the accuracy of the results obtained with the MMPP traffic model is even better. Fig. 3.9, finally, refers to the multi-node T3 topology, and reports the behavior of all three buffers. Once again, the MMPP model behavior is very close to that of traces, and we can observe that traffic crossing several bottleneck (as the traffic relation Φ_1 does) does not alter results significantly.

3.3.2 Queueing Models: Matrix Analitic Solutions

In order to evaluate the accuracy of the model as a synthetic traffic source, we consider a queuing system and we compare the performance obtained by feeding the queue with the synthetic traffic generated by the MMPP model with the results obtained from the real trace. For comparison purposes only, and in order to show the importance of introducing some memory in the input traffic model, we also plot the queue performance obtained when Poisson traffic feeds the queue.

We first consider the case of infinite buffer. The service time distribution reflects the packet length distribution measured from the real traces, which is reported in Fig. 3.10, and exhibits the well-known multi-mode behavior, with peaks for very short packets and for the different MTUs (Maximum Transfer Units) in the network, with a dominating peak at 1,500 bytes, due to the size of Ethernet frame. In order to evaluate the performance of the queue under different values of the load, we change the average service time while

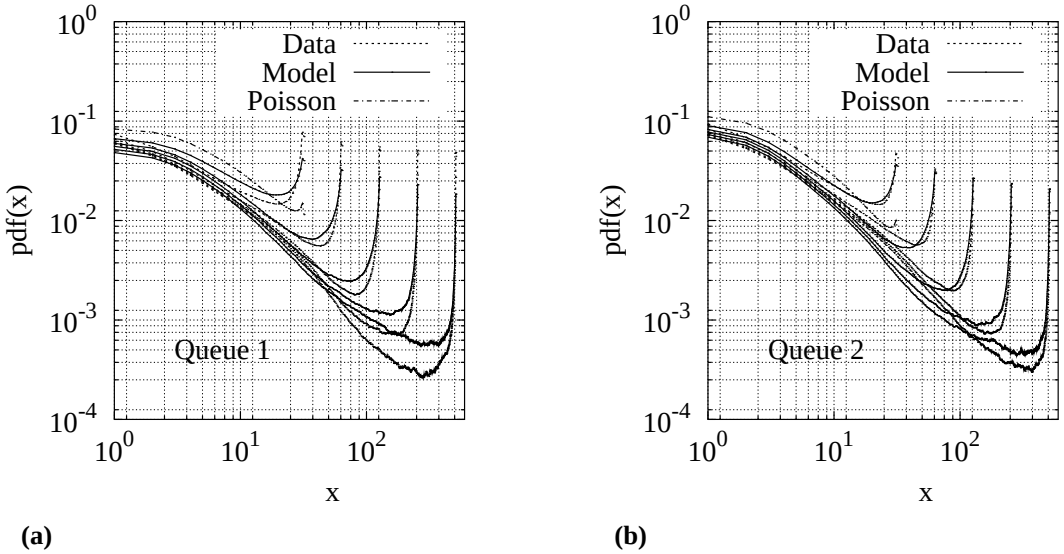
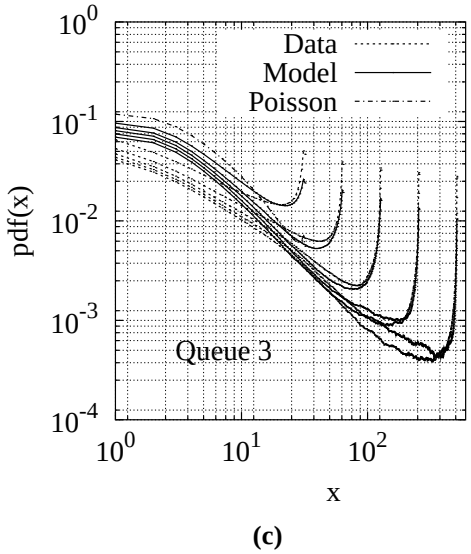


Figure 3.9. Buffer occupancy distribution, topology T3, $B = 32, 64, 128, 256, 512$ packets; the load is normalized on each link; **(a)** first queue, **(b)** second queue, **(c)** third queue.



keeping the packet length distribution unchanged. Notice that the load of the queue has no relation with the actual load of the link where the traces were collected. The results for the synthetic traffic are obtained by simulating the corresponding MMPP/GI/1 queue.

Fig. 3.11 reports the queue length distribution for the Peak'01 trace. The thin dashed line is obtained by using as input the measured trace, the solid line is obtained with the

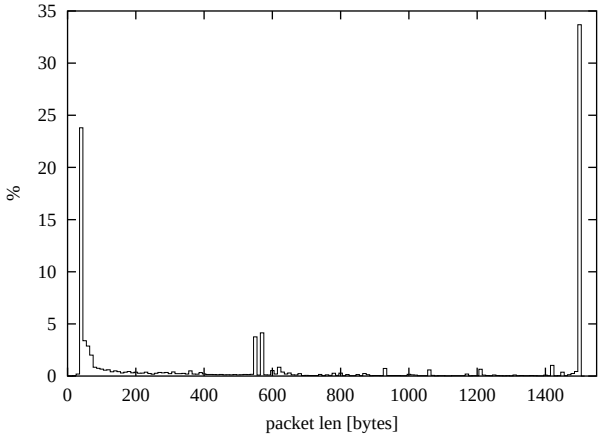


Figure 3.10. Packet length distribution used in simulations

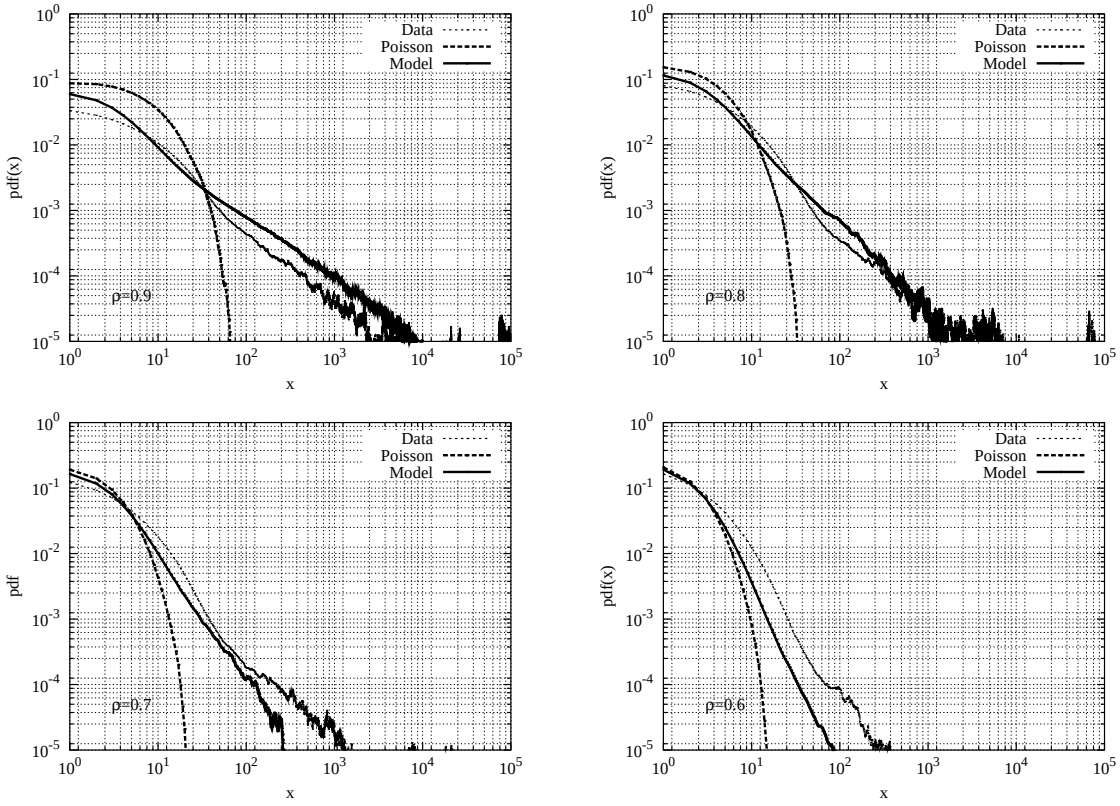


Figure 3.11. Buffer occupancy distribution for the model, the Peak'01 trace and Poisson arrivals; load ρ = 0.9, 0.8, 0.7 and 0.6

MMPP model and the thick dashed line with a simple Poisson process whose rate matches the average packet arrival rate measured on the trace. Plots refer to four different loads: 0.9, 0.8, 0.7, and 0.6. For the real trace, the tail of the distributions below 10^{-4} becomes noisy due to lack of samples. The buffering behavior of the model matches quite well that of the measured traces, while, as expected, the Poisson model underestimates the buffer level of orders of magnitude. Notice that the accuracy of the MMPP model predictions tends to increase for large values of the load, which correspond to the most interesting cases for the system designer.

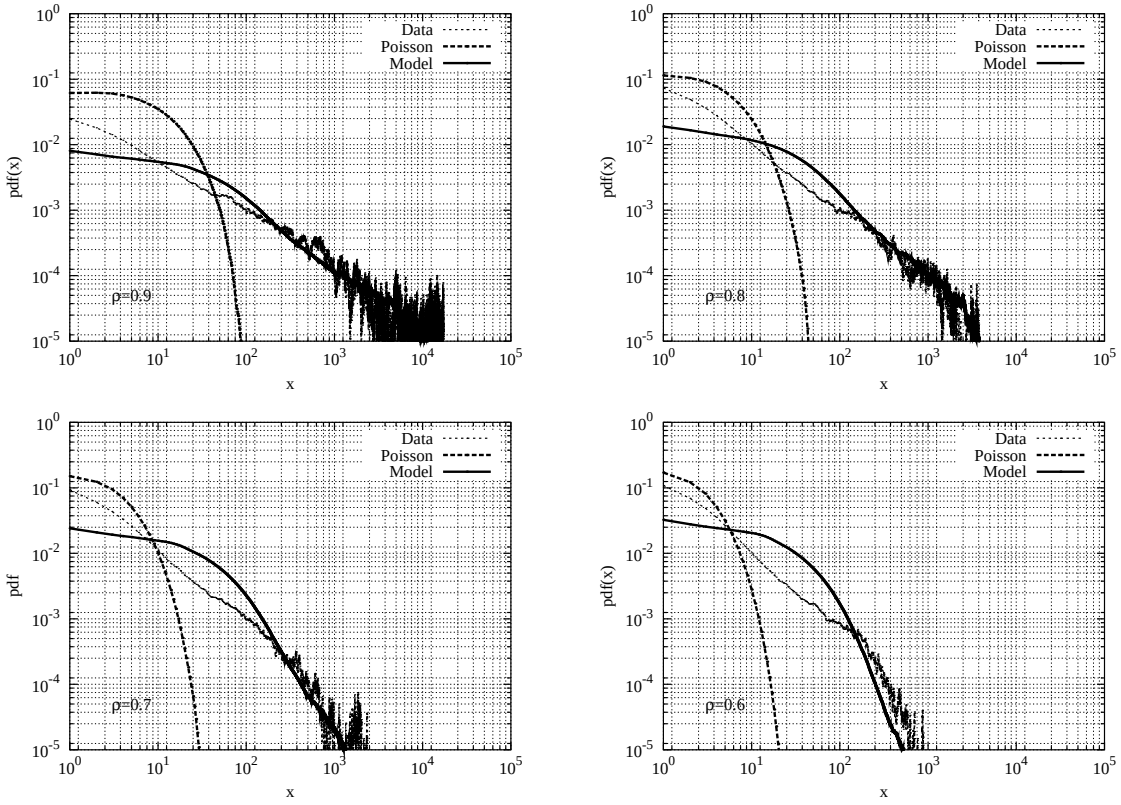


Figure 3.12. Buffer occupancy distribution for the model, the Night'01 trace and Poisson arrivals; load ρ = 0.9, 0.8, 0.7 and 0.6

Similar results are shown in Fig. 3.12 for the Night'01 trace. In this case, the accuracy of the model is less satisfactory, especially when the load is light. It must be noted, however, that this scenario is both less interesting and somewhat more "artificial" than the previous one. First of all, the number of points in the measured trace is about 50 times smaller than during peak hours (see Table 3.1) and this explains why trace-driven curves are noisier. Second, the real link load at night is extremely low, thus artificially forcing the load to values higher than 0.6 significantly modifies the overall scenario. Yet, despite

of this, the MMPP model matches quite well the tail of the distribution, which is typically the most crucial part of the curve.

We now look at the case of finite buffers. The results for the MMPP model are derived analytically, by assuming that service times are exponentially distributed. Fig. 3.13 reports the queue length distribution for a finite buffer queue driven by the same arrival process as in the previous scenarios for the Peak'01 trace, with ρ=0.9. The considered buffer sizes are $B = 32$, 64, 128, 256, 512 packets. Notice again the accuracy of the model in evaluating the queue performance. The curves for the Poisson traffic with different values of the buffer size are indistinguishable.

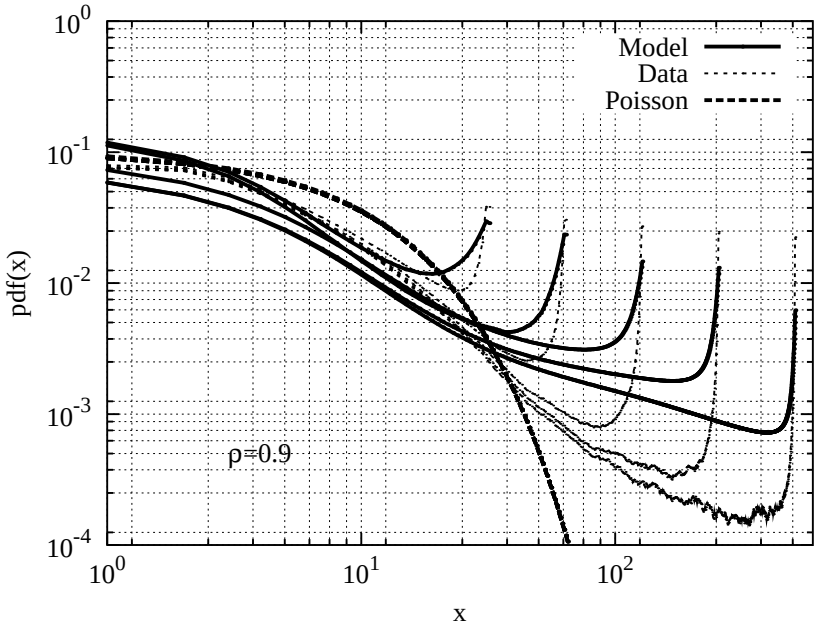


Figure 3.13. Buffer occupancy distribution for the model, the Peak'01 trace and Poisson arrivals; finite buffer with capacity equal to 32, 64, 128, 256, 512 packets and load $\rho = 0.9$

We now consider the typical dimensioning problem: the evaluation of the impact of the buffer size on the packet loss probability. Results are shown in Fig. 3.14. Two curves refer to the MMPP model. The solid line reports analytical results obtained with the assumption that service times are exponentially distributed; the markers are derived from the model by simulation using the same distribution of the service time as for the real trace. The figure proves that the impact of the exponential assumption for service times is marginal. Again, notice that analytical predictions are very accurate.

In terms of complexity, the numerical approach depends mainly on the size of the matrices which describe the Markov process. The computation of the steady-state distribution of the MMPP is efficient because the matrix Q is sparse, and finding the matrix

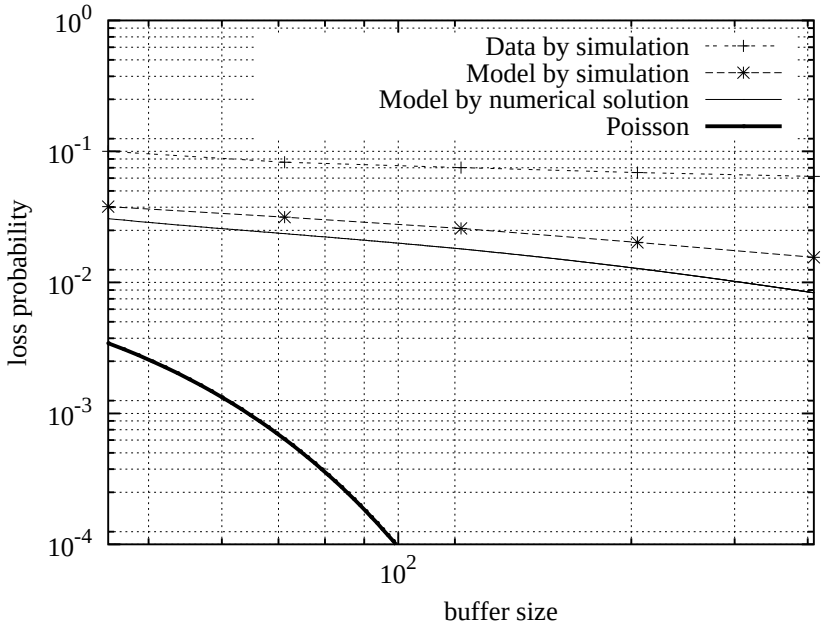


Figure 3.14. Buffer loss probability for the model, the Peak'01 trace and Poisson arrivals; finite buffer with capacity equal from 32, 64, 128, 256, 512 packets and load ρ = 0.9

geometric terms is also efficient thank to the ILRA solution method, even for very large matrices. On the contrary, the computation of the boundaries can be a difficult task because it requires solving a dense linear system.

3.3.3 Sensitivity Analysis

We now discuss the impact of the MMPP model parameters on both the Hurst parameters and the queuing behavior. In particular, we focus on the effect of the average number of flows per session, N_f, which represents the long term memory of the model. We proceed as follows. We first derive the model parameters as described in Section 3.2.2. Then, we set a new value of N_f and, accordingly, we change λ_s so that the average number of flows generated in the time unit does not change. All the other model parameters are kept unchanged. Observe that the load is kept equal to the desired value (0.9) even if N_f is changed, as shown by the following equations:

$$\mathbb{E}[\Lambda_f] = \lambda_s + \lambda_f \mathbb{E}[n_s] = \lambda_s + \lambda_f \frac{\lambda_s}{\lambda_f}(N_f - 1) \tag{3.17}$$

$$= \hat{\Lambda}_f \tag{3.18}$$

$$\mathbb{E}[\Lambda_p] = \mathbb{E}[\Lambda_f] N_p = \hat{\Lambda}_f \hat{N}_p = \hat{\Lambda}_p \tag{3.19}$$

This result is obvious, if we recall that N_f and C do not change the first order statistic but they act only on the second order statistics.

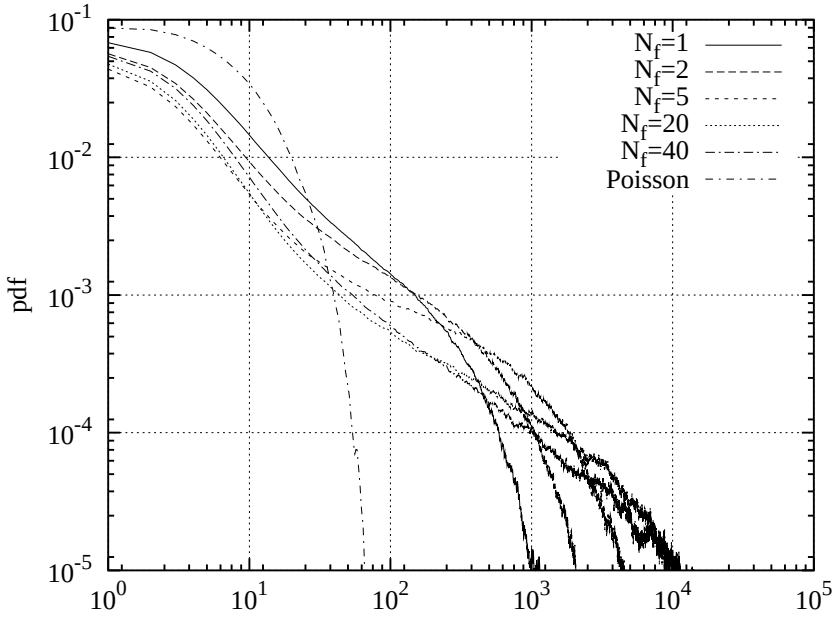


Figure 3.15. Buffer occupancy distribution for the model with different values of N_f and λ_s. Offered load equal to 0.9

Results are shown in Fig. 3.15 for different values of N_f. Again, for comparison purposes only, we report the curve derived with Poisson packet arrivals. The case $N_f = 1$ corresponds to one flow only per session, which implies that the flow arrival process is Poisson. Increasing the number of flows per session makes the queue tail heavier: as N_f increases, the range where the queue decay follows roughly a power law becomes longer. Clearly, there is always a value beyond which the queue decay is exponential and this value increases with N_f. Indeed, the behavior of our model confirms the results obtained in [HRT00], where the Hurst parameter of measured traffic, as well as the queuing behavior of the same traffic, are fitted through the use of phase type distributions.

Thanks to its simplicity and to the intuitive meaning of the parameters, the model can be effectively used as a manageable tool for IP network dimensioning and design. Indeed, having proved that the model accurately represents real traffic behavior, the model can be used to assess the performance of a system under variable traffic conditions, by simply changing the value of the parameters. Consider for example that for the network design we are interested in evaluating the impact of different traffic mixes on a finite buffer. We fix a value of N_f and modify accordingly N_p so that the average offered load to the buffer is constant. This corresponds to traffic mixes composed by either a small number of long

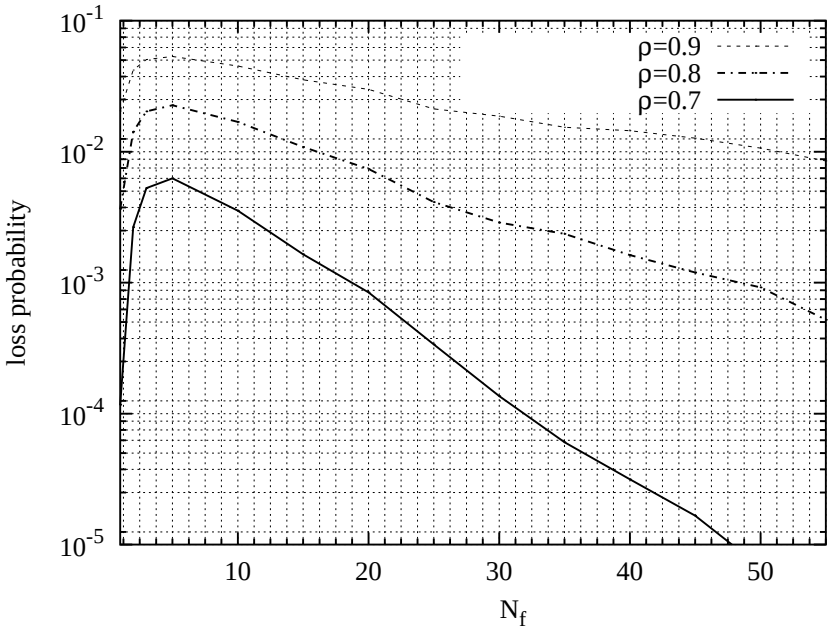


Figure 3.16. Loss Probability with different values of N_f and N_p. Buffer size equal to 512.

flows or a large number of short flows. Fig. 3.16 plots the loss probability for a finite buffer of 512 packets, and for average load equal to $0.7, 0.8, 0.9$. The Figure shows that the loss probability initially increases for increasing values of N_f, but for values of N_f larger than about 5, the loss probability decreases. Indeed, the loss probability is strictly related to the correlation in the packet arrival process: for N_f close to one (and therefore for large values of N_p), we expect less correlation in the traffic mix, having just one long flow per session, i.e., flows and sessions tend to coincide. Similarly, large values of N_f force small values of N_p, which leads to many one-packet long flows, i.e., flows and packets tend to coincide, thus reducing correlation.

Chapter 4

Impact of Traffic on the Internet: The Scalability of Fair Queueing

This chapter is devoted to a very specific problem striclty related to the fair queueing algorithm as a mean to realize statistical bandwidth sharing in the Internet. The comprehensive Internet architecture that makes use of fair queueing has been conceived in two recent papers [KOR04, KMOR05b] that mix per-flow fair queueing and flow admission control leading to a novel flow-aware Internet architecture. The chapter 5 includes an optimized implementation of per-flow fair queuing as detailed in [KMOR05b]. This chapter includes most of the published material in [KMOR05a, KMOR04].

4.1 Introduction

The number of flows traversing a network link is a random process that varies as a large population of potential users independently initiate and complete their applications. Most flows are established to transfer some kind of digital document and can adjust their rate to make the best use of available bandwidth. The way these elastic flows share bandwidth is currently determined by end-to-end transport protocols like TCP that hopefully implement standardized congestion control algorithms. In this work we consider an alternative approach where network mechanisms are employed to govern bandwidth sharing, without having to rely on end user cooperation.

The assumed sharing objective is max-min fairness between user flows [BG92a]. This can be realized by performing per-flow fair queueing on all network links [Hah91]. Users must still perform end-to-end congestion control but only to ensure they fully use the allocated rate.

The proposal to implement fair queueing is clearly not new and it has already been demonstrated that this is possible on high rate links [SLSC99]. Nevertheless, it is widely considered that generalized implementation is not scalable: complexity increases with the

number of flows in progress and this number increases with link rate; required operating speed therefore increases too fast. The objective of the present work is to demonstrate that this belief is false and that fair queueing is both scalable and feasible.

The desirability of max-min fairness as a bandwidth sharing objective has been called into question [KMT98]. In particular, if users gain different utilities from a given bandwidth allocation, there are sound economic reasons to realize shares that reflect these differences. This argument is less compelling, however, when one takes account of the fact that the population of flows in progress varies. It turns out that, under a realistic stochastic traffic model, the differences in the performance of policies like max-min fairness and weighted proportional fairness are hardly significant and not necessarily to the advantage of those providing greater discrimination [BM01].

We study the scalability of fair queueing by means of trace driven simulations and analytical modelling. The traces are recorded on networks with quite different traffic characteristics. In all cases, the simulations demonstrate that the number of flows that the scheduler needs to be aware of at any instant is measured in hundreds, even when there are tens of thousands of flows in progress. The analytical model, based on a quasi-stationary separation of flow level and packet level processes, accurately predicts the simulation results. The model shows how different traffic characteristics impact performance, highlighting the significant role of the flows' exogenous rate limits.

Fair queueing scheduling has been widely studied since the early proposal of Nagle [Nag85]. The main focus of research has been on the development of algorithms with reduced complexity, precise fairness and provable performance bounds. For present purposes, a rough degree of fairness is adequate and we are not interested in delay bounds since there are no constraints on incoming traffic. Our objective is to evaluate the complexity of fair queueing in dynamic, random traffic. To this end we consider the specific case of Start-time Fair Queueing [GVC96]. However, a parallel analysis could be performed to show the equivalence of alternative schedulers, like Deficit Round Robin, for example [SV96].

In view of the imagined complexity of schedulers, a number of active queue management algorithms have been proposed that realize approximate fair sharing [LM97, MFW01, PBPS02]. Our analysis would also apply roughly to an evaluation of the number of flows that need to be handled by these schemes.

The characteristics of individual IP flows have been catalogued according to a variety of criteria including size, rate, duration and burstiness [BC02, SRB01, ZBPS02]. The most significant characteristic for present purposes is the exogenous flow rate: the rate the flow would have if the considered link were of infinite capacity. The measurement study by Zhang *et al.* [ZBPS02] shows that the rate varies widely from flow to flow. Our own analysis shows that the distribution of rates can vary significantly depending on the considered network.

We have previously claimed that fair queueing is scalable in [KOR04, KMOR04] and provided preliminary evidence to back up this claim. The present work goes much

further in both experimental validation using trace driven simulation and in developing a comprehensive analytical model.

We first discuss relevant characteristics of IP traffic at flow level and introduce the three network traces used in the simulations. In Section 4.3, we describe the considered fair queueing algorithm and present simulation results illustrating its performance under the traffic of the three traces. The analytical model is developed in Section 4.4 and evaluated using data representative of the traces. The impact on performance of particular traffic characteristics at flow and packet level is discussed in Section 4.5. We also highlight a number of issues raised by the previous analysis.

4.2 Flow level characteristics of IP traffic

We elaborate on the representation of traffic as a stochastic process at flow and session levels and introduce the traffic traces used in our evaluation.

4.2.1 Traffic as a stochastic process

IP traffic on a network link can be considered as a superposition of independent sessions, each session relating to some piece of user activity and being manifested by the transmission of a collection of flows. Assuming sessions are mutually independent and are generated by a large population of users leads naturally to a Poisson session arrival process. This characteristic has been confirmed empirically and recognized as one of the rare invariants of IP traffic [FP01].

Sessions and flows are defined locally at a considered network element. Flows can generally be identified by common values in packet header fields (e.g., the 5-tuple of IP addresses, port numbers and transport protocol) and the fact that the interval between such packets is less than some time out value (20s, say). It is not usually possible to identify sessions just from data in packets and this notion cannot therefore be used for resource allocation.

A useful traffic model is to suppose flows in a session are emitted one after the other, separated by "think times". The number of flows in a session, the sizes of successive flows and think times and their correlation can vary widely depending on the underlying applications (mail, Web, P2P,...). We refer to a Poisson process of sessions of alternating flows and think times with otherwise general characteristics as the *Poisson session model*. It forms the basis of the analytical model introduced later.

An obvious discrepancy with reality arises if we must identify flows using the 5-tuple of header fields. For instance, the transfer of a Web document would be considered as a single flow in the session model but is frequently realized using several distinct TCP connections in parallel. This discrepancy does not have a significant impact on the present

evaluation of the complexity of per-flow fair queueing, however, as discussed later in Section 4.5.

A more significant flow characteristic is the exogenous peak rate at which a flow can be emitted. This is the highest rate the flow would attain if the link were of unlimited capacity. This limit may be due to the user access line capacity, the maximum TCP receive window or the current available bandwidth on other links of the path, for instance.

The complexity of fair queueing depends somewhat on packet length. IP packets are well known to have a size distribution concentrated around a few particular values such as 40, 570 and 1500 bytes [FML+03]. This length distribution is generally correlated with flow characteristics such as the size or peak rate.

4.2.2 Statistics of trace data

Traffic characteristics used in the present study are derived from trace data for three network links:

ADSL : an OC3 link concentrating the traffic outgoing to several thousand ADSL users; the trace represents 5 minutes of data recorded on August 25 2003;
Ab-I : an OC48 link on the Abilene research network between Indianapolis and Kansas City; the trace represents 5 minutes of data recorded on August 14 2002 (10:30 to 10:35 am);
Ab-III : an OC192 link on Abilene III between Indianapolis and Chicago; the trace represents 2 minutes of trace data recorded on June 1 2004 (7:31 to 7:33 pm).
The Abilene traces are publicly available on the NLANR website [1]. The ADSL data is from a commercial network.

Summary statistics for these three traces are shown in Table 4.1. The table refers to 5-tuple flows defined with reference to a 20s time out. These statistics demonstrate significant differences between the traces. In particular, flows are much bigger on average on Ab-III than on the other two links. This is due to a small number of large flows with a very high rate. The utilization of all three links is very low. The absence of congestion means measured flow rates can reasonably be assimilated to the exogenous rates discussed in Section 4.2.1. The last line of the table gives the average number of flows in progress over the trace lifetime. This number is derived on considering flows to be still "in progress" during the 20s time out. Some 95% of bytes in all traces are in TCP connections.

Figure 4.1 presents the empirical complementary distribution of average flow rates calculated, as in [ZBPS02], for all flows lasting longer than 100ms. Flow rates in the ADSL trace are limited by the offered line rates (all less than 1Mbps). Rates in the Abilene traces can be much higher though most flows still have an average rate less than

[1] See http://pma.nlanr.net/Traces/Traces/long/ipls/1/ and http://pma.nlanr.net/Special/ipls3.html.

Table 4.1. Packet trace statistics summary

	ADSL	*Ab-I*	*Ab-III*
bandwidth	155Mbps	2.5Gbps	10Gbps
total packets	2.6M	19M	156M
total flows	850K	2.3M	683K
utilization	28%	13%	19%
flows in progress	24000	37000	62000

1Mbps. The Ab-III trace is somewhat atypical in that it includes some flows with the exceptionally high average rate of around 300Mbps. These turn out to be TCP flows using 9000 byte jumbo packets.

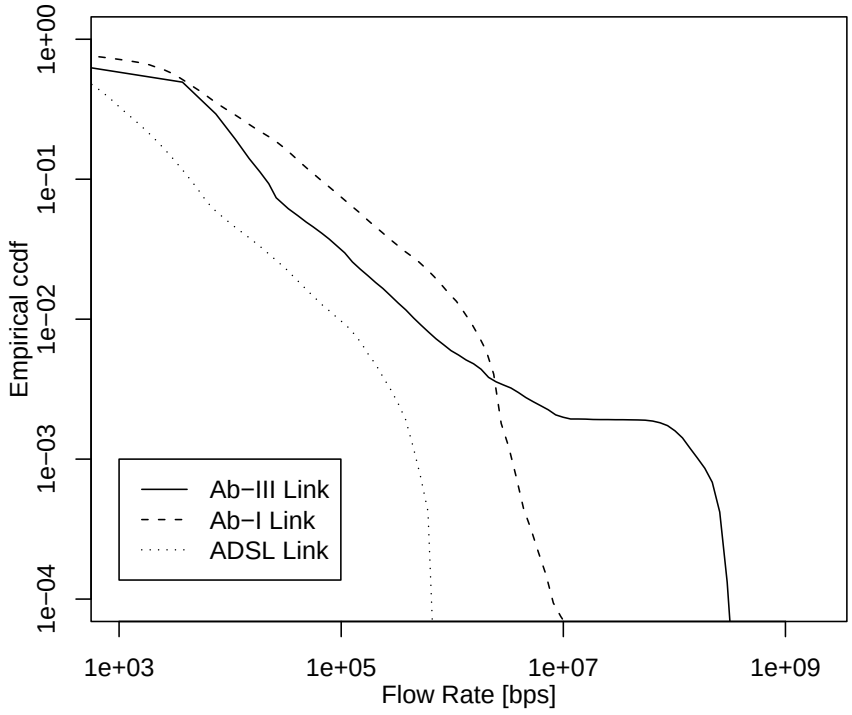


Figure 4.1. Empirical complementary distribution of average flow rates.

As an alternative measure of the exogenous rate we have measured the peak rate attained in successive 10ms intervals over the flow lifetime (the length of the interval is not critical since similar statistics were obtained using 100 ms). Results presented in Figure

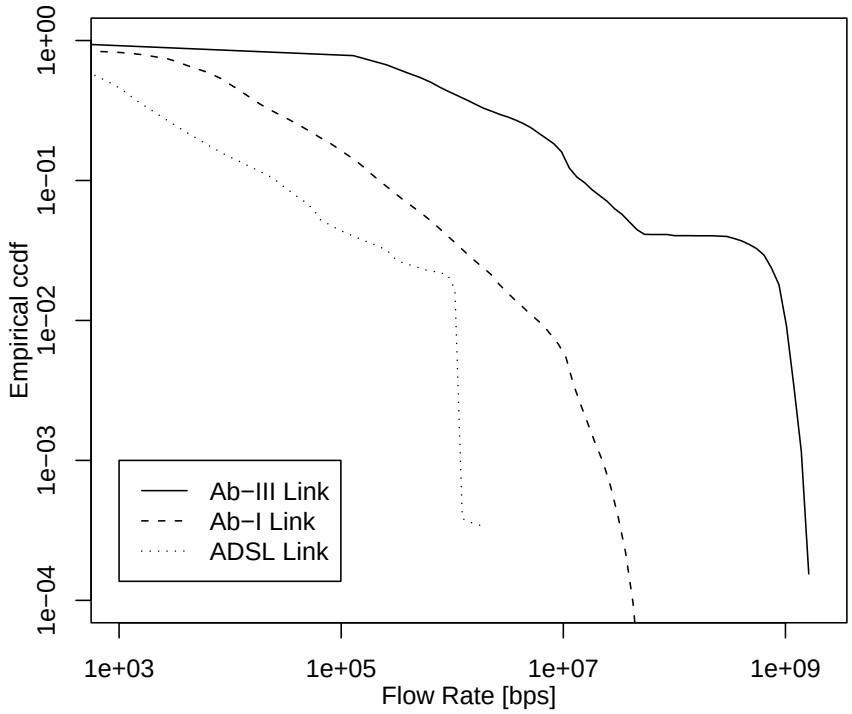


Figure 4.2. Empirical complementary distribution of flow peak rates.

4.2 accentuate the differences between the three traces revealing notably that flows in Ab-III can attain more than 1Gbps while rates of ADSL flows are all less than 1Mbps.

Packet lengths are correlated with rate for the Abilene traces. On Ab-I, all packets are less than 1500 bytes. Most packets in flows whose peak rate is greater than 10Mbps are of this length. Lower rate flows, that include flows of TCP ACKs, have a much higher proportion of 40 byte packets. Some 67% of bytes in the Ab-III trace are in flows of peak rate greater than 500Mbps. Almost all packets in these flows are of 9000 bytes. Lower rate flows in this trace have more typical packet sizes ranging from 40 to 1500 bytes. The distribution of packet sizes in the ADSL trace is clustered around 40, 570 and 1500 bytes and is broadly independent of flow rate.

4.3 Complexity of per-flow fair queueing

We choose to exemplify the class of fair queueing schedulers by the self-clocked Start-time Fair Queueing (SFQ) algorithm of Goyal et al. [GVC96]. We discuss how complexity of SFQ in a dynamic traffic setting depends on the number of so-called *active* flows.

We then evaluate the distribution of this number using trace driven simulations.

4.3.1 Start-time Fair Queueing

Pseudocode for SFQ adapted to the present context of a dynamically varying flow population is given in Figure 4.3. Fairness properties of the algorithm are discussed in [GVC96]. It is shown, for example, that the amount of data transmitted by any continuously backlogged flow is always within one maximum sized packet the ideal fair share.

```
1.   on arrival of l-byte packet p of flow f:
2.   if f ∈ ActiveList do
3.      TimeStamp.p = FinishTag.f
4.      FinishTag.f += l
5.   else
6.      add f to ActiveList
7.      TimeStamp.p = VirtualTime
8.      FinishTag.f = VirtualTime + l

9.   transmit packets in increasing TimeStamp order

10.  at the start of transmission of packet p:
11.     VirtualTime = TimeStamp.p
12.     for all flows f ∈ ActiveList
13.        if (FinishTag.f ≤ VirtualTime) remove f

14.  if no packets in scheduler VirtualTime = 0
```

Figure 4.3. Pseudocode of SFQ with a dynamic list of active flows.

The complexity of the algorithm is usually evaluated in terms of the number of operations necessary per packet under the assumption that the flow population is fixed. This complexity is determined by the sort operation implicit in line 9 of the pseudocode and thus typically increases with the logarithm of the number of flows. In the present dynamic version, the sort must be performed on flows that have a packet with a time stamp greater than the current value of VirtualTime. This is the number of *bottlenecked* flows since this condition only occurs when the flow incoming rate exceeds the current fair rate.

The number of bottlenecked flows is usually much smaller than the number of *active* flows. Active flows are flows with an entry in ActiveList. This includes any bottlenecked flow but also any flow that has recently emitted a packet, as manifested by the fact that its FinishTag is still greater than VirtualTime.

Pseudocode operations 2 and 12-13 depend on the number of active flows. Operations 12-13 are trivial if the list is sorted in FinishTag order, implying log complexity for a sort operation whenever a finish tag is updated. The test in operation 2 can have O(1) complexity if the size of ActiveList is small enough to be realized using content addressable memory. The overall complexity of the algorithm thus depends mainly on the number of active flows and secondarily on the number of these that are bottlenecked.

Note that, in any realization, the size of ActiveList is finite and it is necessary to limit the probability of overflow. It is thus important to understand how the distribution of the number of active flows depends on the intensity and characteristics of offered traffic. It is also necessary to specify what happens when the list is full. We assume packets of new flows that cannot be recorded are scheduled with time stamp equal to VirtualTime. List saturation thus has no impact on scheduling non-bottlenecked flows. Bottlenecked flows will not be slowed to the current fair rate until a subsequent packet arrival when the list is no longer saturated.

4.3.2 Trace driven simulations

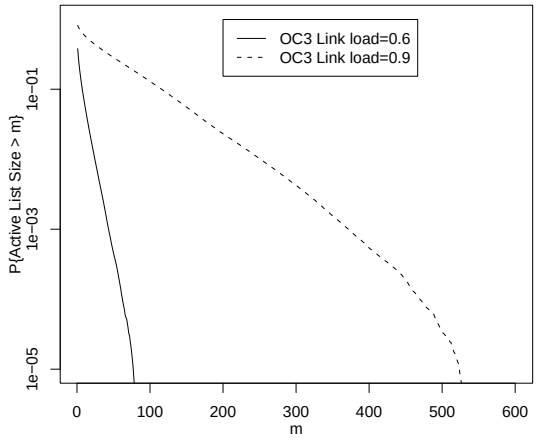
The link utilization for the three traces presented in Section 4.2 is too low to produce any interesting scheduling behaviour. For example, the maximum size of ActiveList observed on the OC192 link was 11 flows. To evaluate the performance of SFQ we therefore apply the trace data to a simulated scheduler operating at a rate considerably less than the rate of the link on which the trace was measured. The different simulated link capacities are given in Table 4.2. For each trace in Table 4.1, link capacities were calculated to produce a load of 0.6 and 0.9, where the load is defined as flow arrival rate $\times$ mean flow size, divided by the link capacity. This produces realistic results under the assumption that the transport protocol controlling each flow is capable of using the current fair rate, whenever this is less than the exogenous peak rate. In the simulation, this condition is satisfied since sufficient buffering is provided to avoid any packet loss. Figure 4.4 shows

Table 4.2. Simulated link capacities in bps, all traces

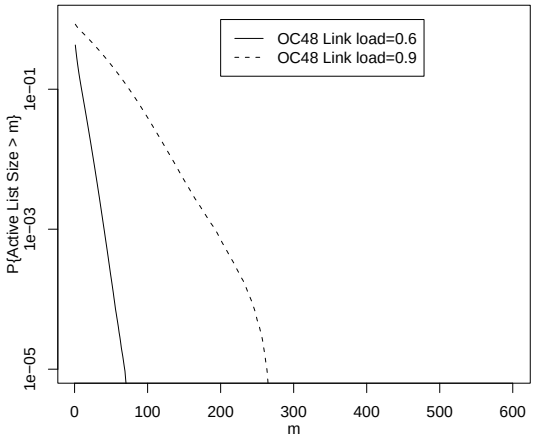
Load	ADSL	Ab-I	Ab-III
0.6	72M	541M	3.16G
0.9	48M	361M	2.11G

the complementary distributions of the observed ActiveList size.

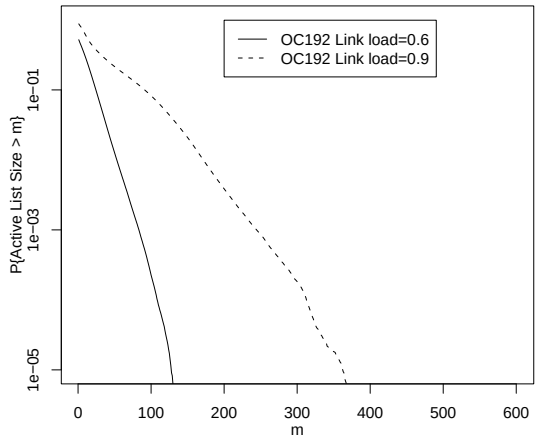
The most significant observation is that the number of active flows in all three cases, even at 90% utilization, is very much smaller than the average number of flows in progress (see Table 4.1). Secondly, the number does not increase with link rate. Indeed, the worst



$(a) ADSL$



$(b) Ab - I$



$(c) Ab - III$

Figure 4.4. Complementary distribution of ActiveList size from trace driven simulations at utilizations of 60% and 90%

case is that of the ADSL link. This is due to the fact that most flows are not bottlenecked on this link because of the limited peak rate. The distinction between the impact of bottlenecked and non-bottlenecked flows is clarified in the development of the analytical model in the next section.

These results are significant in that they suggest fair queueing is scalable and feasible. It is scalable since complexity does not increase with link speed; it is feasible since the required ActiveList size is relatively small.

4.4 Analytical model

In this section we focus on analytical models to corroborate our conjectures based on simulations so far. We present different models in order to highlight with different levels of approximation what influences, and why, the scalability of this class of algorithms under random traffic.

4.4.1 A traffi c model for homogeneous flows

To explain the difference in the numbers of in progress, active and bottlenecked flows, we introduce the following simple model. Flows are generated according to the general Poisson session model described in Section share a link of capacity C. We assume all flows emit data at the same constant rate and, to simplify the presentation, we choose this rate to be C/m with m an integer. Let the number of flows in progress be N_m. When $N_m \geq m$, the SFQ scheduler imposes that each flow realizes the current fair rate on output. We assume the transport protocol is perfectly efficient in ensuring each flow is able to emit at this rate.

The above assumptions correspond to the fluid traffic model considered by Ben Fredj *et al.* [FBP+01]. The distribution of the number of flows in progress depends only on the link load ρ (flow arrival rate $\times$ mean size / C) and the rate parameter m. Denoting this distribution by $\pi_m(n)$, we have:

$$\pi_m(n) = (1-\rho) f(\rho) \times \begin{cases} \frac{m!}{n!}(\rho m)^{n-m}, & \text{for } n < m, \\ \rho^{n-m}, & \text{for } n \geq m, \end{cases} \tag{4.1}$$

where

$$f(\rho) = \frac{(m\rho)^m/m!}{(1-\rho)\sum_{k=0}^{m-1}(m\rho)^k/k! + (m\rho)^m/m!}.$$

Let A_m represent the number of active flows when a new entry needs to be added to this list (triggered by a packet arrival when $n < m$ or a new flow arrival when $n \geq m$). To derive the distribution of A_m, we assume all packets have the same size. Considering the operation of SFQ, when N_m is less than m, the number of active flows is non-zero only in a busy period of the packet queue. During the busy period, A_m increases by one on each packet arrival, these packets coming necessarily from distinct flows, and is reset to zero at the end of the busy period. When N_m is greater than or equal to m, the active list always contains exactly N_m flows.

The distribution $b_m(i,n)$ of the number of packets in a busy period can be derived from known results for the nD/D/1 queue [Vir94]. We have, for $1 \leq n < m$ and $1 \leq i \leq n$:

$$b_m(i,n) = \binom{n-1}{i-1} \frac{m-n}{m-n+1} \frac{i^{i-2}(m-i)^{n-i-1}}{n^{n-1}} \tag{4.2}$$

Let $d_m(j,n)$ denote the probability of the event $A_m = j$, given $N_m = n$. We have:

$$d_m(j,n) = \begin{cases} \left(1 - \dfrac{n-1}{m}\right) \displaystyle\sum_{i=j+1}^{n} b_m(i,n), \text{for} 0 \le j < n < m \\ 1, \text{ for } n \ge m \text{ and } j = n, \text{ or } j = n = 0 \\ 0, \text{ otherwise.} \end{cases}$$

The first line derives from the fact that one packet in i sees an active list of size j in every busy period comprising i packets whenever $i > j$. Finally, the distribution of the number of active flows $a_m(j)$ is derived by deconditioning with respect to the distribution (4.9):

$$a_m(j) = \sum_{n \ge 0} d_m(j,n)\pi_m(n). \tag{4.3}$$

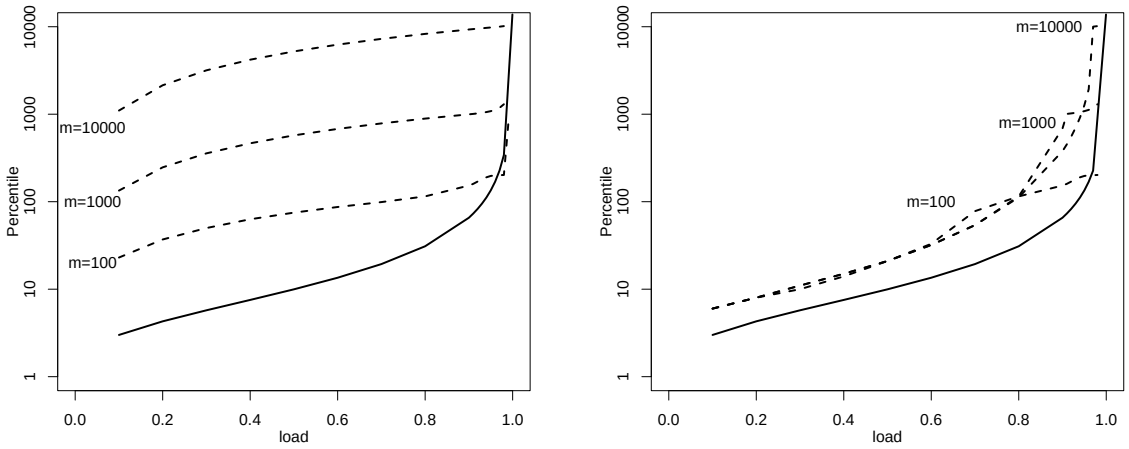
Let B_m denote the number of bottlenecked flows. B_m is equal to n with probability $\pi_m(n)$, for $n \ge m+1$, and is 0 with probability $\sum_{0 \le i \le m} \pi_m(i)$.

Noting that A_m is unbounded, any finite active list size leads to some probability of overflow. We would like to know how big the list needs to be to satisfy some upper limit on this probability. We therefore choose to represent the performance of this traffic model in terms of a remote percentile of the distribution.

Figure 4.5 plots the 99.9% percentile of the distributions of N_m and A_m against link load ρ for a range of rate parameters m. When $m = 1$, all flows in progress are active and bottlenecked and N_1, A_1 and B_1 have the same geometric distribution. The percentile of this distribution is represented as the continuous line in Fig. 4.5 (a) and (b). We do not plot results for B_m since the percentile is trivially small except when ρ is very close to 1. The plots for A_m in Fig. 4.5(b) behave non-uniformly as load increases due to the abrupt change in the form of its conditional distribution (4.3) as N_m exceeds m.

The results illustrate two important points about fair queueing:
1) Scheduling is *scalable* in that the numbers of active and bottlenecked flows do not increase with link capacity C. In fact, for a fixed exogenous flow rate and a given link utilization, the percentile for bottlenecked flows decreases as C increases while that of active flows tends to a limit.
2) Scheduling is *feasible* up to high load (utilization $< 90\%$, say) since the number of active flows remains small. However, this number explodes as utilization approaches 100% under demand overload (traffic intensity $>$ capacity).

In this section we develop an analytical model to explain why fair queueing is scalable and to understand the impact of the various traffic characteristics. An earlier model outlined in [KMOR04] successfully reproduced simulation results for another trace but relied for this on a traffic classification derived from the simulation. The present model relies only on intrinsic traffic characteristics.



(a) Flows in progress N_m

(b) Active flows A_m

Figure 4.5. 99.9 percentiles of flow number distributions against link load: $m = 1,100,1000,10000$.

4.4.2 Traffi c model

We assume traffic is generated according to the Poisson session model introduced in Section 4.2.1. We further suppose flows have a *constant* peak rate drawn from a set $\{c_1,c_2,\ldots,$
$c_M\}$ with $c_1 > c_2 > \cdots > c_M$. Flows with peak rate c_i constitute class i. The number of class i flows in progress at time t is denoted $X_i(t)$.

The analysis is based on a quasi-stationary timescale separation. We first suppose the X_i are fixed and evaluate the distribution of the number of active flows for each state $x = (x_1,\ldots,x_M)$. We then introduce assumptions allowing us to estimate the stationary distribution of X in order to derive the required unconditional distribution.

4.4.3 Bottlenecked and active flows

For notational convenience, define $c_0 = C$ and $c_{M+1} = 0$. If $\sum_{i=1}^{M} x_i c_i > C$, fair queueing realizes max-min fair sharing with fair rate θ given by

$$\theta = \frac{C - \sum_{i>J} x_i c_i}{\sum_{i \le J} x_i} \tag{4.4}$$

where $J = J(x)$ is the unique integer, $1 \le J \le M$, such that $c_J \ge \theta > c_{J+1}$.

Flows of peak rate greater than or equal to c_J are bottlenecked and realize the fair rate θ while the others preserve their peak rate through the scheduler. If $\sum_{i=1}^{M} x_i c_i \le C$, all flows preserve their peak rate. Let $J(x) = 0$ in this case. Note that J defines a partition of the state space.

Consider now the operation of the SFQ algorithm. All bottlenecked flows are included in ActiveList. In addition, any non-bottlenecked flow having emitted a packet in the recent past, such that its finish tag is less than VirtualTime, is also included. To evaluate the stationary distribution of the total number we must make some assumptions about packet lengths and the arrival process of packets from non-bottlenecked flows.

The most convenient assumption is to suppose all packets of bottlenecked flows have maximum size MTU. This is reasonable given the packet size statistics of high peak rate flows reported above and is arguably a worst case assumption for the distribution of ActiveList size[2]. Given this assumption, and taking MTU as the unit of virtual time, it is easy to see that VirtualTime takes only integer values.

The periods when at least one flow is bottlenecked and VirtualTime is constant constitute "busy cycles". Any packet from a non-bottlenecked flow arriving in a busy cycle acquires the time stamp VirtualTime and further perpetuates the cycle for the duration of its own transmission. This flow remains in ActiveList until the end of the cycle. It is then removed since its finish tag cannot be greater than VirtualTime + 1 (it has size $l \leq 1$ in MTU units). Figure 4.6 illustrates the notion of busy cycle.

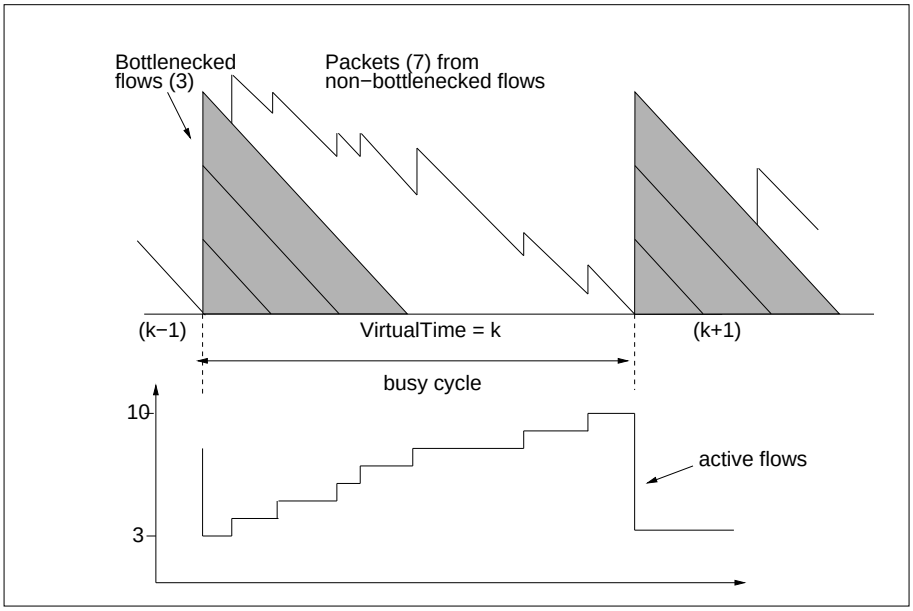


Figure 4.6. Illustration of the notion of busy cycle.

Busy cycles also occur when there are no bottlenecked flows in progress. In this case a busy cycle is initiated by a packet arriving to an empty system. VirtualTime remains equal to zero in such cycles.

[2]This is because VirtualTime changes more rarely so that small non-backlogged packets remain longer in the list.

The size of ActiveList is largest just before the end of a busy cycle. To evaluate the stationary distribution of this number *MaxList*, we assume each non-bottlenecked flow emits at most one packet in a cycle[3].

To model the arrival process we assume each flow independently emits its packet between time t and $t + dt$ after the start of the busy cycle with probability $\alpha dt + o(dt)$. This finite source Poisson process facilitates analysis while accurately accounting for the number of contributing flows $\sum_{i>J} x_i$.

4.4.4 Conditional distribution of ActiveList size

From the above description it may be recognized that an estimate for the number of non-bottlenecked flows contributing to *MaxList* can be derived from an evaluation of the busy period of a queue with exceptional first service. The latter corresponds to the transmission of one MTU sized packet from each of the $\sum_{i\leq J} x_i$ bottlenecked flows [4].

The arrival process results from $N = \sum_{i>J} x_i$ flows independently emitting at most one packet according to the process described at the end of the last section. The service time distribution derives from that of the packet length $F(s)$. We assume packet lengths are i.i.d. for all non-bottlenecked flows[5]. Let packet length be measured in units of MTU and denote the mean by σ.

The assumed per-flow arrival rate α is the average rate for all non-bottlenecked flows: $\alpha = R/(N\sigma)$ where $R = \sum_{i>J} x_i c_i$ is their overall bit rate. The probability k packets arrive in an interval of length u from the start of the busy cycle is then:

$$\Lambda_N(k,u) = \binom{N}{k}(1 - e^{-\alpha u})^k e^{-(N-k)\alpha u}\mathbb{1}_{(0,+\infty)}(u) \tag{4.5}$$

The conditional distribution of *MaxList* is given by the following proposition.

Proposition 3 *Let $g(m;b,n,r)$ be the conditional distribution of* MaxList *given b bottlenecked flows and n non-bottlenecked flows contributing overall rate r. Assuming packets arrive according to process with distribution Λ and have independent length drawn from distribution F, we have:*

$$g(m;b,n,r) = \begin{cases} \frac{1}{m}\int \Lambda_n(m-1,u)dF^{[m]}(u), & b = 0,\ m \geq 1, \\ \Lambda_n(0,b), & b > 0,\ m = b, \\ b\int \Lambda_n(m-b-1,u)/u dF^{[m-b]}(u-b), & b > 0,\ m > b, \end{cases} \tag{4.6}$$

where $F^{[m]}$ is the m-fold convolution of F.

[3] A flow cannot emit more than MTU bytes since this would contradict its non-bottlenecked status.

[4] The order of service of packets with the same time stamp has no impact on *MaxList*; it is convenient to suppose the bottlenecked packets are emitted first.

[5] We therefore ignore the fact that this distribution depends on the number of flows in each class and its specific distribution.

Proof We adapt classical results for the M/G/1 queue by substituting Λ_n for the corresponding Poisson distribution. This is possible because the considered arrival process has the same indiscernible properties as the Poisson process allowing us to apply the combinatorial lemma of Takàcs [Tak62, page 231] and its generalization by Niu and Cooper [NC89]. The first case, when no flows are bottlenecked, derives from the classical result for the number of customers served in an M/G/1 busy period [Tak62, page 63]. The second case just expresses the probability that no non-bottlenecked flow emits a packet in the busy cycle. The third case is derived from results in [NC89] for the number of customers served in a busy period starting with an exceptional first service corresponding to b packets from the bottlenecked flows.

Example 4 *Let us consider the case packets belong to a discrete distribution with a finite support.* $dF(u) = \sum_{k=1}^{K} \alpha_k \delta(u - x_k)du$, $\rho = \lambda \mathbb{E}[S]$, $\mathbb{E}[S] = \sum_{k=1}^{K} \alpha_k x_k$.
If $b = 0$, $m \geq 0$ *then* $g(m;0,n,r) =$

$$\begin{aligned} &= \frac{1}{m} \int_{\mathbb{R}} \Lambda_n(m-1,u) \sum_{\substack{k_1,\ldots,k_K \\ \sum_{i=1}^{K} k_i = m-1}} M\left(\{k_i\}; m-1, \{\alpha_i\}\right) \\ &\quad \delta\left[u - \left(\sum_{i=1}^{K} k_i x_i + 1\right)\right] du \\ &= \sum_{\substack{k_1,\ldots,k_K \\ \sum_{i=1}^{K} k_i = m-1}} M\left(\{k_i\}; m-1, \{\alpha_i\}\right) \frac{\Lambda_n(m-1, \sum_{i=1}^{K} k_i x_i + 1)}{\sum_{i=1}^{K} k_i x_i + 1} \end{aligned}$$

If $b > 0$, $m = b$ *then* $g(m;0,n,r) = \Lambda_n(0,b)$
If $b > 0$, $m \geq 0$ *then* $g(m;b,n,r) =$

$$\begin{aligned} &= b \int_{\mathbb{R}} \Lambda_n(m-b-1,u)/u \sum_{\substack{k_1,\ldots,k_K \\ \sum_{i=1}^{K} k_i = m-b-1}} M\left(\{k_i\}; m-b-1, \{\alpha_i\}\right) \\ &\quad \delta\left[u - \left(\sum_{i=1}^{K} k_i x_i + 1\right)\right] du \\ &= b \sum_{\substack{k_1,\ldots,k_K \\ \sum_{i=1}^{K} k_i = m-b-1}} M\left(\{k_i\}; m-b-1, \{\alpha_i\}\right) \frac{\Lambda_n(m-b-1, \sum_{i=1}^{K} k_i x_i + 1)}{\sum_{i=1}^{K} k_i x_i + 1} \end{aligned}$$

where $M\left(\{k_i\}; m-b-1, \{\alpha_i\}\right)$ *is the multinomial distribution function.*

$$M\left(\{k_i\}; m, \{\alpha_i\}\right) = m! \prod_{i=1}^{K} \frac{\alpha_i^{k_i}}{k_i!}, \quad \sum_{i=1}^{K} k_i = m \tag{4.7}$$

From the distribution of *MaxList* we can derive the distribution of the active list size at the moment when a flow should be added. Let this random variable be *AddToList*. Under the quasi-stationary assumption, the flow population x is fixed and only non-bottlenecked flows are ever *added* to the list. The following proposition gives the conditional distribution of *AddToList*.

Proposition 5 *Let $h(m;b,n,r)$ be the conditional distribution of* AddToList *given that there are b bottlenecked flows and n non-bottlenecked flows contributing overall rate r. We have, for $m \geq b \geq 0, n > 0$:*

$$h(m;b,n,r) = \frac{\sum_{k>m} g(k;b,n,r)}{\sum_{l>b}(l-b)g(l;b,n,r)} \tag{4.8}$$

Proof The distribution $g(k;b,n,r)$ gives the relative frequency of busy cycles starting with ActiveList size b and terminating with size k due to $k-b$ packet arrivals. The probability that that an arbitrary arrival occurs in such a busy cycle is thus $(k-b)g(k;b,n,r)/\sum_{l>b}(l-b)g(l;b,n,r)$. The probability that arrival encounters m flows in ActiveList is then $1/(k-b)$ for $b \leq m < k$. The expression for h is derived on multiplying the two probabilities and summing over all possible values of k.

4.4.5 Flows in progress

Consider now the stationary distribution of $X(t)$ assuming a fluid model where flow rates adjust instantly as flows start and end. The fair queueing scheduler realizes max-min fair sharing with fair rate θ as defined in equation (4.4). Unfortunately, analysing max-min sharing is intractable under any realistic traffic assumptions. We therefore consider the alternative sharing objective of *balanced fairness*. This leads to a relatively simple expression for the distribution of X [BP02]. The assumption, that we partially justify later, is that the distribution of X is approximately the same under max-min and balanced fairness.

Balanced fair allocation

Under balanced fairness it is possible to evaluate exactly the stationary distribution of X. Moreover, this distribution is insensitive to all characteristics of the Poisson session traffic model defined in Section 4.2 other than the respective class demands a_i (= class i flow arrival rate $\times$ average class i flow size). The load is equal to a/C where $a = \sum_i^M a_i$.

Following Bonald and Virtamo [BV05] we adopt the shorthand $c^x = \prod_{i=1}^M c_i^{x_i}$, $a^x = \prod_{i=1}^M a_i^{x_i}$ and $x! = \prod_{i=1}^M x_i!$ and denote by e_i the M vector with 1 in position i and 0 elsewhere. The stationary distribution π_{BF} is then given by:

$$\pi_{BF}(x) = \Phi(x)a^x/G \tag{4.9}$$

where G is a normalizing constant ([BV05] provides a recursive formula to evaluate G) and the balance function Φ is determined recursively as follows:

$$\Phi(x) = \begin{cases} 1/(x!c^x) & \text{if } xc^T \leq C, \\ \sum_{i=1}^{M} \Phi(x - e_i)/C & \text{if } xc^T > C. \end{cases} \tag{4.10}$$

Recurrence relations

Let $B_j = \sum_{i \leq j} X_i$, $N_j = \sum_{i>j} X_i$ and $R_j = \sum_{i \leq j} X_i c_i$ for $0 \leq j \leq M$. We need to evaluate, from $\pi_{BF}(x)$, the joint stationary distribution of B_J, N_J and R_J where J is the index defined in Section 4.4.3. Note that as J defines a partition of states x, it also partitions allowable values of (B_J,N_J,R_J) (we have, for example, $\theta = (C - R)/B$ and $c_J \leq \theta < c_{J+1}$).

Let $Q_J^{BNR}(b,n,r)$ denote the joint distribution of B_J, N_J and R_J. To evaluate this distribution directly is impractical in view of the huge size of the corresponding array. We therefore evaluate the following approximation:

$$\begin{aligned} Q_J^{BNR}(b,n,r) &= \mathbb{P}\{B_J = b | N_J = n, R_J = r\} Q_J^{NR}(n,r) \\ &\approx Q_J^{BR}(b,r) Q_J^{NR}(n,r)/Q_J^{R}(r), \end{aligned} \tag{4.11}$$

with the obvious interpretation for the distributions on the right hand side. This is reasonable since N_J and R_J are closely correlated. To evaluate the two-term distributions we apply the recurrence relations stated in the following propositions.

Proposition 6 *The joint distribution of B_J and R_J, for $0 \leq J \leq M$, is given by $Q_J^{BR}(b,r) = q_J(b,r)$, where the $q_j(b,r)$, for $0 \leq j \leq M$, satisfy the following recurrence relations:*

$$\begin{aligned} q_j(b,r) &= \sum_{i \leq j} \frac{a_i}{C} [q_j(b-1,r) + \sum_{s=C-r-c_i+1}^{C-r} p_j(b-1,s,r)] \\ &+ \sum_{i>j} \frac{a_i}{C} [q_j(b,r-c_i) + \sum_{s=C-r+1}^{C-r+c_i} p_j(b,s,r-c_i)] \end{aligned} \tag{4.12}$$

and

$$\begin{aligned} p_j(b,s,r) &= \sum_{i \leq j} \frac{a_i}{(s+r) \wedge C} p_j(b-1,s-c_i,r) \\ &+ \sum_{i>j} \frac{a_i}{(s+r) \wedge C} p_j(b,s,r-c_i). \end{aligned} \tag{4.13}$$

Values of q_j and p_j are zero if any argument is negative and $q_j(0,0) = p_j(0,0,0) = \pi_{BF}(0)$ is determined by the normalization condition.

Proof Define variables $S_j = \sum_{i \le j} x_i c_i$ and sets $\mathcal{S}_j(b,s,r) = \{x : B_j = b, S_j = s, R_j = r\}$. Consider the probabilities $p_j(b,s,r) = \mathbb{P}\{B_j = b, S_j = s, R_j = r\}$ for $j > 0$:

$$\begin{aligned} p_j(b,s,r) &= \sum_{x \in \mathcal{S}_j(b,s,r)} \pi(x) \\ &= \sum_{x \in S_j(b,s,r)} \sum_{i=1}^{N} \frac{a_i}{(s+r) \wedge C} \pi(x - e_i) \\ &= \sum_{i \le j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(b-1, s-c_i, r)} \pi(x) + \\ &+ \sum_{i > j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(b,s,r-c_i)} \pi(x), \end{aligned}$$

and this is (4.13). The second step uses properties (4.9) and (4.10), (see [BV05]). If $b = 0$ then $q(0,r)$, for $r = 0, \ldots, C$, is directly given by the Kaufman-Roberts recursion (see below and [BV05]). Observe that when $b > 0$, $S_J + R_J > C$. Thus we can define $q_j(b,r) = \sum_{s > C-r} p_j(b,s,r)$ for $j > 0$ so that

$$\begin{aligned} q_j(b,r) &= \sum_{s > C-r} p_j(b,s,r) = \\ &= \sum_{i \le j} \frac{a_i}{C} \sum_{s > C-r} p_j(b-1, s-c_i, r) \\ &+ \sum_{i > j} \frac{a_i}{C} \sum_{s > C-r} p_j(b,s,r-c_i), \\ &= \sum_{i \le j} \frac{a_i}{C} \sum_{s > C-r-c_i} p_j(b-1,s,r) \\ &+ \sum_{i > j} \frac{a_i}{C} \sum_{s > C-r} p_j(b,s,r-c_i), \end{aligned}$$

The last two sums can be split as in (4.12). $Q_J^{BR}(b,r)$ is given limiting $q_j(b,r)$ in the state space partition induced by J.

Proposition 7 *The joint distributions of N_J and R_J, for $1 \le J \le M$, is given by*

$Q_J^{NR}(n,r) = q_J(n,r)$, where the $q_j(n,r)$, for $1 \le j \le M$, satisfy the following recurrence relations:

$$(1 - \sum_{i \le j} \frac{a_i}{C}) q_j(n,r) = \sum_{i \le j} \frac{a_i}{C} \sum_{s=C-r-c_i+1}^{C-r} p_j(s,n,r) + \sum_{i>j} \frac{a_i}{C} [q_j(n-1,r-c_i) + \sum_{s=C-r+1}^{C-r+c_i} p_j(s,n-1,r-c_i)] \tag{4.14}$$

and

$$\begin{aligned} p_j(s,n,r) &= \sum_{i \le j} \frac{a_i}{(s+r) \wedge C} p_j(s-c_i,n,r) \\ &+ \sum_{i>j} \frac{a_i}{(s+r) \wedge C} p_j(s,n-1,r-c_i) \end{aligned} \tag{4.15}$$

Values of q_j and p_j are zero if any argument is negative and $p_j(0,0,0) = \pi_{BF}(0)$ is determined by the normalization condition.

Proof Define variables $S_j = \sum_{i \le j} x_i c_i$ and sets $\mathcal{S}_j(s,n,r) = \{x : S_j = s, N_j = n, R_j = r\}$. Consider the probabilities $p_j(s,n,r) = \mathbb{P}\{S_j = s, N_j = n, R_j = r\}$ for $j > 0$:

$$\begin{aligned} p_j(s,n,r) &= \sum_{x \in \mathcal{S}_j(s,n,r)} \pi(x) \\ &= \sum_{x \in S_j(s,n,r)} \sum_{i=1}^{N} \frac{a_i}{(s+r) \wedge C} \pi(x - e_i) \\ &= \sum_{i \le j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(s-c_i,n,r)} \pi(x) + \\ &+ \sum_{i>j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(s,n-1,r-c_i)} \pi(x), \end{aligned}$$

and this is (4.15). The second step uses properties (4.9) and (4.10) (see [BV05]) as in the

previous proof. Now let $q_j(n,r) = \sum_{s>C-r} p_j(s,n,r)$ so that

$$\begin{aligned} q_j(n,r) &= \sum_{s>C-r} p_j(s,n,r) = \\ &= \sum_{i\le j} \frac{a_i}{C} \sum_{s>C-r} p_j(s-c_i,n,r) \\ &+ \sum_{i>j} \frac{a_i}{C} \sum_{s>C-r} p_j(s,n-1,r-c_i), \\ &= \sum_{i\le j} \frac{a_i}{C} \sum_{s>C-r-c_i} p_j(s,n,r) \\ &+ \sum_{i>j} \frac{a_i}{C} \sum_{s>C-r} p_j(s,n-1,r-c_i), \end{aligned}$$

The last two sums can be split as in (4.14) where the first term does not depend on i. If $J = 0$ we obtain again the Kaufman-Roberts recursion. $Q_J^{NR}(n,r)$ is given considering $q_j(n,r)$ in the state space partition induced by J.

Proposition 8 *Kaufman-Roberts Recursion*
$\mathbb{P}\{X_1c_1 + \cdots + X_Mc_M = n\} = q(n)$ *for* $n \le C$ *satisfies the following recursion:*

$$q(n) = \sum_{i=1}^{M} \frac{a_i}{n} q(n-c_i)$$

with $q(n) = 0$ *if* $n < 0$.

Proof The joint distribution defined in (4.9),(4.10) in the case $xc^T \le C$ satisfies the relation $x_ic_i\pi(x) = a_i\pi(x-e_i)$. Thus, summing over i, $n\pi(x) = \sum_{i=1}^{M} a_i\pi(x-e_i)$. Finally, summing over all x such that $xc^T = n$ we conclude the proof.

Comparison between max-min and balanced fair sharing

Previous work has shown that max-min and balanced fairness result in similar flow level performance [BP02]. This has been confirmed in the present case by evaluating the behaviour of max-min fair sharing by fluid simulation and comparing the marginal distributions of B and R (over all J) with that derived for balanced fairness.

We assume a Poisson arrival process of flows having a Weibull size distribution with shape parameter equal to 0.3 (yielding a long tailed distribution) in the first simulation scenario, and an exponential size distribution with the same mean size equal to 1Mbit in the second one. Arrival rates were chosen to produce the demand proportion associated with each class. Flow peak rates in Mbps are $c = \{1500, 1000, 800, 500, 100, 50, 10, 1\}$

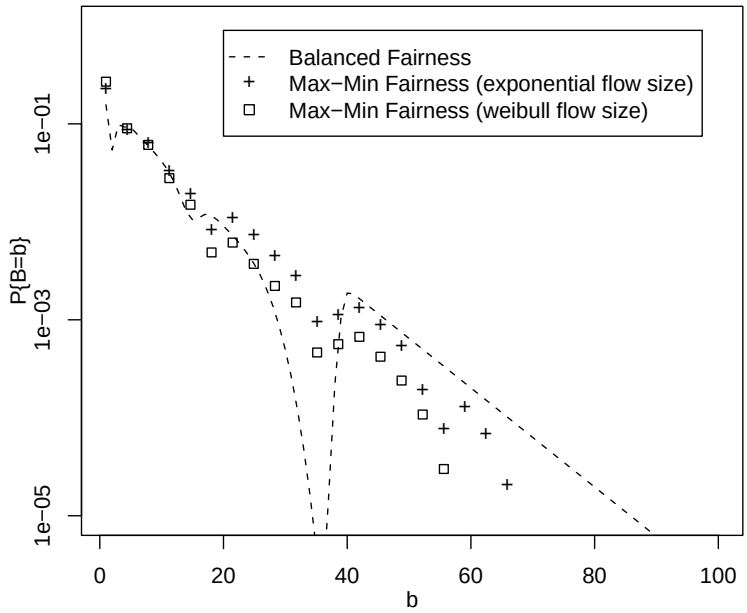


Figure 4.7. Comparison between balanced and max-min allocations, load = 0.9,AbileneIII trace.

with respective demands in proportion to $a = \{21, 22, 17, 7, 6, 17, 6, 4\}$. These data are representative of the Abilene III OC192 trace.

Figures 4.7 and 4.8 compare the marginal distributions of B and R, respectively, for link load 0.9. The rough agreement is typical of results for data corresponding to the other traces. Note the somewhat erratic fluctuations in these distributions due to the discrete mixture of peak rates. In this fluid model, all flows belonging to the same class become bottlenecked or non bottlenecked simultaneously which explains the observed jumps in the distribution of B. Errors introduced by the approximation do not appear to be too significant. At load 0.6, there is little controlled sharing (i.e., $J = 0$ with high probability) and max-min and balanced fairness are practically the same.

4.4.6 Unconditional ActiveList size distribution

We can now estimate the distribution of ActiveList size by combining the results of Sections 4.4.4 and 4.4.5, i.e.

$$\mathbb{P}\{\text{ActiveList size} = m\} = \begin{cases} \sum_{(b,n,r):J<M} h(m;b,n,r) Q_J^{BNR}(b,n,r) & J < M \\ Q_M^{BNR}(m,0,0) & J = M \end{cases}$$

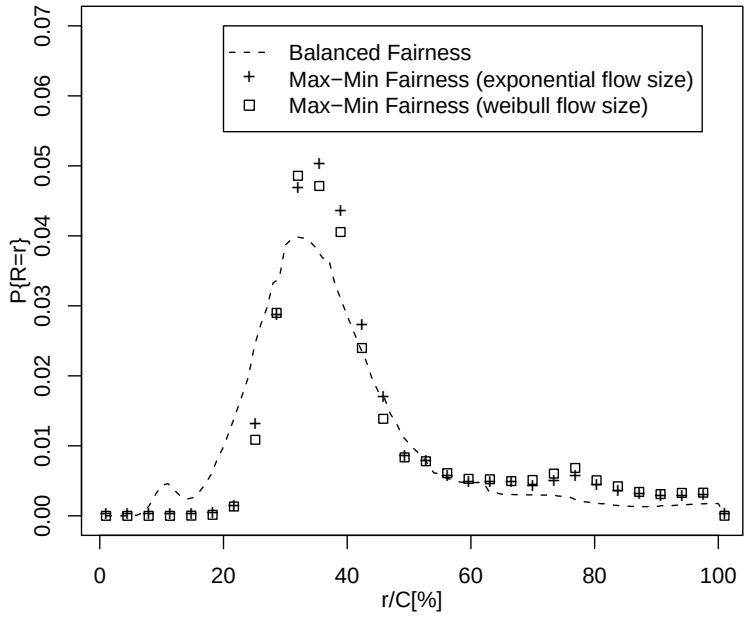


Figure 4.8. Comparison between balanced and max-min allocations, load = 0.9, AbileneIII trace.

Table 4.3. Flow classes and traffic proportions

	ADSL		Ab-I		Ab-III	
Class	c [bps]	a_i/a [%]	c [bps]	a_i/a [%]	c [bps]	a_i/a [%]
$\mathcal{C}_1$	1M	52	100M	24	1.5G	21
$\mathcal{C}_2$	512K	26	20M	10	1G	22
$\mathcal{C}_3$	128K	12	15M	9	800M	17
$\mathcal{C}_4$	50K	10	12M	14	500M	7
$\mathcal{C}_5$	-	-	10M	14	100M	6
$\mathcal{C}_6$	-	-	5M	11	50M	17
$\mathcal{C}_7$	-	-	1M	4	10M	6
$\mathcal{C}_8$	-	-	500K	12	1M	4

We perform the calculations with data representative of the three traces described in Section 4.2.2 The peak rate classes (in bps) and respective traffic proportions (in %) are given in Table 4.3.

These values are derived by choosing a set of peak rates roughly covering the domain of the distributions in Figure 4.2 and attributing to each class the traffic from flows whose peak rate is closest. We set C and adjust the absolute values of the demand vectors a to attain link loads of 0.6 and 0.9, as in the simulations described in Section 4.3.2 and summarized in Table 4.2.

For the packet size of non-bottlenecked flows we have used a distribution with just two values as it is difficult to numerically evaluate the convolution in Proposition 3 for more terms. We chose packet sizes of 40 and 1500 bytes with relative proportions derived roughly from the trace statistics (50% for each packet size for the ADSL and Abilene traces, 30% and 70% respectively for the Abilene III trace). The value of MTU was 1500 bytes for the first two traces and 9000 bytes for the third.

Results are presented in Figure 4.9. As can be seen, the analytical model predicts the simulated ActiveList size distribution quite accurately. These results are discussed in the next section.

4.5 Discussion

In this section we discuss the relative importance of traffic characteristics at flow and packet timescales and highlight a number of important issues.

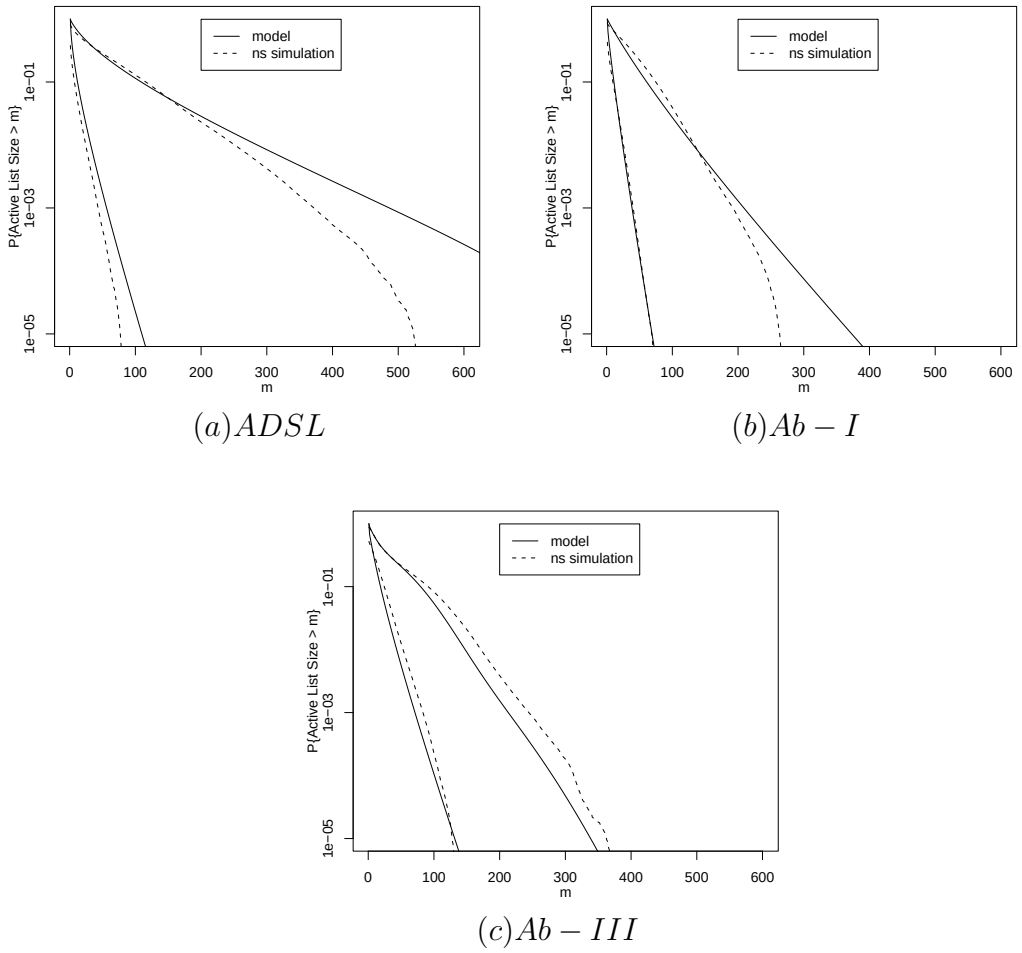
4.5.1 Impact of traffic characteristics

First recall the lack of impact on the population of flows in progress X of the detailed session structure under the assumed Poisson session traffic model. This insensitivity is strictly valid only for balanced fairness but we believe it to be true also for all practical purposes for max-min fairness. The only significant characteristics are the overall load and the distribution of this load over the range of exogenous flow peak rates.

The assumption in the model that the peak rate is one of a finite set of values and is the same throughout the flow lifetime is clearly a simplification. It nevertheless appears to predict performance reasonably accurately. The main impact of the rate distribution is to determine for any load which proportion of traffic is in bottlenecked flows and which is not. These two types of traffic are shown to have qualitatively different impacts on ActiveList size.

Bottlenecked flows are few in number up to high loads of around 90%. The majority of active flows correspond to isolated packets emitted by relatively low rate non-bottlenecked flows. This explains why the ADSL trace gives rise to the largest list: its peak rate limited flows are rarely bottlenecked even at 90% load. The very high rate flows in Ab-III, on the other hand, have a beneficial impact on the statistics of ActiveList size.

The packet length of non-bottlenecked flows has a significant impact. The smaller the packets, the greater the packet arrival rate and, in consequence, the greater the number of



$(a) ADSL$ $(b) Ab-I$

$(c) Ab-III$

Figure 4.9. Complementary distribution of ActiveList size from analytical model at utilizations of 60% and 90%

flows contributing to a busy cycle and momentarily entering ActiveList.

4.5.2 Benefits of fair queueing

Imposed fairness relieves the network from relying on user cooperation to ensure satisfactory performance. It does remain possible, however, for users to establish multiple parallel flows and thus acquire a greater rate than the fair share. The present models show that such advantage would only accrue to the small proportion of users whose flows are bottlenecked. Such users already realize high throughput. If they nevertheless "cheat" by setting up parallel flows, this should be fairly easy to detect and counter given their small number.

The Poisson session model (with alternating flows and think times) remains appropriate for users whose combined peak rate is less than the fair rate, whatever the number of 5-tuple flows they actually use in parallel. This is because packets of the combination still arrive in isolation (i.e., ≤ 1 in any busy cycle) and add 1 to the current ActiveList size independently of the particular 5-tuple flow to which they belong.

When fairness is assured by the network, new transport protocols can be tested and introduced without having to ensure they are not overly aggressive to legacy TCP variants. This is an appreciable advantage, notably for the design of protocols adapted to high speed transmission (see Section 4.5.5 below). It is also possible to introduce known techniques for improving the behaviour of TCP slow-start.

A particularly interesting benefit of fair queueing is the protection afforded to relatively low rate real time flows. As long as the rate of audio and video flows is less than the current fair rate (and our models demonstrate that this is typically quite high), their packets achieve low latency in traversing the FQ scheduler. This effectively realizes service differentiation without the cumbersome requirement to explicitly mark packets as real time or delay tolerant. Packet latency can be further reduced for flows whose peak rate is less than the current fair rate by giving priority to their packets (i.e., packets that acquire time stamp VirtualTime in line 7 of the SFQ code go to the head of the schedule), as proposed in [KOR04].

4.5.3 Alternative realizations

We have used the SFQ algorithm to exemplify the performance of fair queueing but it is clearly possible to use different algorithms. Similar results would apply for other self-clocking schedulers and to round robin variants like DRR [SV96].

Some authors have proposed to use active queue management schemes to avoid the pretended scalability issues of scheduling [LM97, MFW01, PBPS02]. The present discussion would also apply (with somewhat less precision) to determining the performance of these algorithms in a realistic dynamic traffic setting. These algorithm aim to operate only on the bottlenecked flows but would presumably need to account momentarily for active flows in order to be able to detect new bottlenecked ones.

Core stateless fair queueing [SSZ03] distributes the complexity of scheduling to the network edge here the current rate of each flow is estimated. Note that all in progress flows must be tracked and measured with this approach.

Of course, the congestion control of TCP currently realizes approximate fair sharing with simple FIFO queues (albeit with well known biases and inefficiencies). The present models are again useful in understanding how the number of flows actually contending for resources on a given link (the bottlenecked flows) is typically very small.

4.5.4 Avoiding overload

In this work we have shown that fair queueing is scalable and feasible at normal to heavy loads. When demand exceeds around 90% of link capacity, however, the number of in progress and active flows increases rapidly. In such overload the scheduler (like the above mentioned AQM mechanisms) tends to revert to a FIFO queue. Per-flow throughput gets smaller and smaller as an increasing number of flows compete for the available bandwidth.

It is clearly desirable to avoid such overload, whatever queue management scheme is employed. A possible solution avoiding the current requirement for excessive over-provisioning would be to use per-flow admission control as advocated in [KOR04]. The measure of the current fair rate realized by the scheduler is a convenient criterion for deciding when new flows should be blocked.

4.5.5 Buffer sizing

It has recently been observed that the rule of thumb whereby router buffers should be able to store some 200 ms of data at line rate is excessive and unsustainable as link bandwidth increases [AKM04]. The present discussion on the numbers of in progress, active and bottlenecked flows throws more light on typical buffer requirements.

It is shown in [AKM04] that buffer requirements decrease substantially as the number of flows in progress increases, under the assumption that these flows are all bottlenecked. This trend is confirmed using different models in [DJD05], again assuming traffic is composed mainly of "locally bottlenecked persistent flows". In fact, our results demonstrate that the number of locally bottlenecked flows is typically relatively small, the vast majority of flows having a peak rate considerably less than the current fair rate. If more than a hundred flows *are* actually bottlenecked, this is usually a sign of severe overload that buffering alone cannot control, as discussed in the previous section.

If only a handful of flows are bottlenecked (there is a non-negligible probability that the number is only one or two), they necessarily have a high rate and the models in [AKM04] show that a large buffer is indeed necessary to sustain 100% utilization using TCP. We might therefore argue that the above rule of thumb should be applied, at least to the sizeable line rate fraction used by the bottlenecked flows. However, this is not what we would advocate.

Most of the time, on most links, no flow is bottlenecked and a small buffer (for around 100 packets) is adequate to ensure low loss. Rather than applying the rule of thumb to cater for the small number of exceptional flows that can saturate the residual link capacity, it seems more appropriate to require that these flows employ one of the recently proposed high speed TCP variants (e.g., [Flo03, Kel03, JWL04]). A feature of these protocols is that they require much less buffering to attain 100% link utilisation.

4.6 Conclusion

We have shown that the advantages of network assured max-min fair sharing can be realized because per-flow fair queueing is *scalable* - since the number of flows that need to be scheduled is independent of link speed - and *feasible* - since this number is relatively small. Performance has been evaluated using trace driven simulations and explained by means of an analytical model.

The model demonstrates that, after the average link load, the most significant traffic characteristics are the exogenous peak rates the flows can attain in the absence of link congestion. These determine the proportion of flows that are bottlenecked and those that are not. The model shows how these flows combine to determine the complexity of the scheduler and explains why this remains low up to utilizations of around 90%.

The number of bottlenecked flows is typically measured in tens. Many more flows are in progress but they are non-bottlenecked and only contribute episodically to the list of active flows to be accounted for by the scheduled. The overall number of flows that need to be scheduled is limited to a few hundreds, independently of the proportions of traffic in bottlenecked and non-bottlenecked flows.

In addition to protecting the quality of service of individual flows against possible user misbehaviour, the use of fair queueing would bring two significant advantages. First, it would be possible to test and introduce new, more efficient transport protocols without requiring that they be TCP-friendly. This is useful notably for very high speed transfers for which the congestion avoidance algorithm of TCP is known to waste bandwidth. Second, fair queueing ensures low packet latency for flows whose rate is less than the current fair rate, the non-bottlenecked flows. This allows audio and video flows to coexist with high speed data transfers without the need to explicitly identify the former.

The present work suggests several directions for further research. The more approximate fairness realized by AQM mechanisms may be sufficient for elastic data traffic but it remains to evaluate the performance they provide for below fair rate audio and video flows. How could a transport protocol best exploit the assurance of max-min fairness in order to improve the current slow-start and congestion avoidance algorithms? Fair queueing is feasible as envisaged only up to a load of around 90%: it is necessary therefore to implement some means to prevent overload. This may take the form of dynamic load balancing or adaptive routing. A necessary component appears to be some form of flow-level admission control, as envisaged in [KOR04].

This work presents credible technical arguments, based on analysis, simulations and recent Internet measurements, in favour of the feasibility and desirability of fair queueing. We hope these arguments will encourage router architects to reconsider the opportunity of implementing fair queueing in the design of next generation routers allowing the development of a more efficient and reliable Internet.

Chapter 5

A Flow Aware Internet Architecture

Limitations of the current best effort datagram-based Internet architecture have led to proposals for enhanced QoS empowered architectures. Early proposals for an integrated services network with per-flow connection set up, based on RSVP signalling [BCS94], have not been widely deployed for reasons of scalability. The currently favoured solution is to apply class-based service differentiation [BBC$^+$98], coupled with the traffic engineering facilities of MPLS [FL03], to perform traffic management based on broad flow aggregates. In fact, analysis of the way performance depends on the statistical nature of traffic suggests that neither Intserv nor Diffserv constitutes a truly effective service model enabling the necessary quality assurances. Both impose a significant cost penalty compared to the best effort network, in terms of both capital and operational expenditure [Rob04].

The present work contributes to the specification of an alternative flow-aware networking (FAN) architecture [KOR04, OR05, KMOR05b]. Here we present part of the specifications proposed in [KMOR05b] that focus on per-flow fair queueing.

The objective of FAN is to augment the current Internet as sparingly as possible in order to provide necessary performance assurances for real time and data transfer applications. According to the analysis in [Rob04] and [KOR04], the necessary mechanisms are per-flow fair queueing and flow level admission control. This chapter discusses the implementation overhead of these mechanisms and proposes original designs that aim to minimize complexity.

A flow in the present context is a flight of datagrams, localized in time (packets are spaced by no more than a certain interval, TimeOut) and space (packets in question are observed at a particular interface) and having the same unique identifier. The identifier is deduced from header fields including IP addresses and user-specified fields like the IPv6 flow label or IPv4 port numbers. The expectation is that users define flows to correspond to a particular instance of some application such as a video stream or a document transfer.

Space locality means a given end-to-end flow has multiple instances, one at each network element on its path. To be flow-aware, these elements identify flows on the fly by

examining the packet header. There is thus no requirement for signalling and no modification to the network control plane. Minimal soft state is established, as necessary, for scheduling and admission control. As is well known, fair queueing provides effective traffic separation, avoiding current reliance on cooperative user implementation of congestion responsive end-to-end protocols. Fair queueing also realizes a certain implicit differentiation between real time streaming flows and bursty, potentially high rate, data transfers. This is because packet latency is limited for any flow that is not bottle-necked and this is the case for most audio and video applications.

The feasibility of per-flow fair queueing has recently been demonstrated by showing that the number of active flows (i.e., the flows that need to be known to the scheduler at any instant) is not more than several hundred, even at link loads as high as 90%, and this for any link capacity [KMOR04, KMOR05a, KMOR05b]. The active flows are composed of a relatively small number of bottle-necked flows together with the subset of non-bottlenecked flows that currently have a packet in the buffer. In this chapter, we show how complexity can be further reduced by limiting scheduling to just the bottlenecked flows, the maximum number of active flows then being limited to around 100.

If demand (flow arrival rate $\times$ average flow size) were to exceed link capacity, the number of flows to be schedule would grow to much more than 100. To preserve scalability, and to preserve the performance of flows in progress, it is necessary to limit utilization by performing flow level admission control. Since the number of flows in progress increases with link rate and could attain a million or more for the highest capacities, the amount of state required for admission control is considerably more than that necessary for scheduling. We discuss two approaches for identifying the current set of in-progress flows: leveraging a general purpose flow table that maintains per-flow state for a variety of applications in addition to admission control; designing an *ad hoc* data structure that reduces memory requirements and complexity by exploiting the specific features of the admission control application. The architecture is structured in two main devoted to fair queueing scheduling and admission control, respectively. We base the discussion of scheduling on Deficit Round Robin [SV96], showing how this algorithm can be modified to identify just the bottlenecked flows whose rate needs to be controlled. The section on admission control is not presented here because it has not been subject of the present thesis work, but for sake of completeness the interested reader would find the complete architecture in [KMOR05b].

5.1 Per-flow fair queueing

The advantages of per-flow fair queueing have long been recognized [Nag85, DKS90]. As well ensuring protection against malicious use, network assured fairness frees applications form the current requirement to be “TCP-friendly”. It would be possible, for example, to introduce more efficient high speed transport protocols without concern that

they might be unfair to legacy TCP connections. Max-min fairness also provides low latency to the packets of flows that have a peak rate less than the current fair rate. This realizes an implicit service differentiation on recognizing that most streaming applications fall into this category [1]. We present the design of a reduced complexity fair queueing scheduler based on Deficit Round Robin [SV96]. We have performed similar developments for the self-clocked time stamp based scheduler Start-time Fair Queueing [GVC97] but omit details in the interests of conciseness.

5.1.1 Scalability

The huge amount of research on fair queueing since Nagle's pioneering proposal has mainly been concerned devising schemes that realize tight fairness and delay bounds for a given set of rate controlled sources. Our focus is on ensuring approximate fairness for a highly dynamic population of active flows identified on the fly. The main objective is to realize fair queueing with low complexity.

Two broad classes of schedulers can be used: self-clocked fair queueing where packet emissions are ordered according to an assigned time stamp (exemplified by Start-time Fair Queueing (SFQ) [GVC97]) and round-robin algorithms where dynamically constituted queues are visited in cyclic order (exemplified by Deficit Round Robin (DRR) [SV96]). Weighted fairness can be applied in the present context if the class of a flow can be determined on the fly from packet headers.

The complexity of SFQ and DRR depends on the number of flows that need to be scheduled. For link loads[2] of up to 90%, it has been shown that with high probability this number does not exceed a few hundreds, for any link rate [KMOR04, KMOR05a]. In Section 5.1.3 below, it is shown that the number can be further reduced to around 100.

Most flows in an IP network are short lived. The number of flows in progress at any instant on a high speed link is a variable that can attain a value measured in tens or hundreds of thousands. However, the flows that need to be accounted for by a fair queueing scheduler are considerably fewer. The scalability of fair queueing derives from the fact that most in-progress flows enter the schedule rarely, only when they have a packet to emit.

The schedule only concerns flows that have a packet in the router buffer. This fact makes per-flow scheduling scalable since the number of such flows is largely independent of the link rate and is measured only in hundreds [KOR04, KMOR04, KMOR05a].

Some of the flows to be scheduled are bottlenecked in that they could attain a higher rate if the link in question had unlimited capacity. Traffic models predict that the number of bottlenecked flows in progress is less than 100 with high probability as long as link load is not higher than 90%.

[1] Latency of low rate flows is further reduced in the FQ implementation described in Sec. 5.1.3

[2] flow arrival rate $\times$ average flow size / link capacity

Most flows in progress at any instant are not bottlenecked. Their rate is limited by other constraints on their path (access links, notably) to a peak value less than the fair rate offered by the link. Only the bottlenecked flows strictly need scheduling. We explore how DRR can be modified to separate out the other flows leading to a less onerous implementation.

5.1.2 DRR scheduling for a dynamic set of active flows

Flows that need to be accounted for by the DRR scheduler are logged in a table called ActiveList. Necessary per-flow state for a given flow f using DRR is as follows:

- Identifier.f - the flow identifier (possibly a hash of the relevant header fields)
- Queue.f - current length in bytes of flow f queue
- Quantum.f - value of flow f quantum ($\geq$ MTU bytes)
- Deficit.f - current flow deficit
- FIFO.f - addresses of head and tail packets of a linked list forming the flow f FIFO
- Next.f - the next flow in the DRR schedule following flow f.

Succinct pseudo-code for DRR is given in Figure 5.1. Flows are temporarily logged in ActiveList when they first emit a packet after a period of inactivity and remain there until they have no packets to emit at the end of one of their scheduled service quanta. A given in-progress flow may enter and leave ActiveList several times during its lifetime. In particular, non-bottlenecked flows enter and leave this structure once for every packet.

The ActiveList data for one flow in a typical realization amounts to some 16 bytes of memory. The overall memory requirement depends on the maximum number of flows that need to be recorded at any time. To limit this requirement we propose to add an additional data structure and modify the scheduling algorithm to distinguish between bottlenecked and non-bottlenecked flows, as described in the next section. Since ActiveList can then be limited to a capacity of around 100 flows (see below), it is feasible to use content addressable memory allowing rapid consultation of flow status in line 2 of Figure 5.1.

5.1.3 Identifying bottlenecked flow

We aim to identify non-bottlenecked flows and avoid inserting their packets in the DRR schedule. In addition, we seek to emit the packets of these flows with priority, ahead of the packets of bottlenecked flows. As explained in [KOR04], this allows a form of implicit service differentiation since streaming flows typically have a peak rate less than the fair rate and thus naturally fall into the non-bottlenecked category. Their packets

```
1.  on arrival of l-byte packet p of flow f:
2.  if f ∈ ActiveList do
3.     update FIFO.f
4.     Queue.f + = l
5.  else
6.     add f to ActiveList
7.     initialize FIFO.f
8.     Queue.f = l
9.     Deficit.f = 0
10.    add f at end of DRR schedule

11. transmit packets in DRR schedule order

12. when flow f is scheduled:
13. Deficit.f + = Quantum.f
14. while (Queue.f ¿ 0)
15.    get head packet from FIFO.f; l = packet size
16.    if (l > Deficit.f) skip while loop
17.    else
18.        emit packet
19.        Queue.f − = l
20.        Deficit.f − = l
21. if (Queue.f == 0) remove f from ActiveList
```

Figure 5.1. Pseudocode of DRR with a dynamic list of active flows.

are forwarded with very low latency. The maximum flow rate for which low latency is assured can be engineered by choosing admission control thresholds that maintain the fair rate high enough, even under overload.

To identify bottlenecked flows, we use an additional data structure called NewFlows. NewFlows is an array of M words of b bits. For each flow f there corresponds a unique NewFlows word i, determined by a hash function applied to the flow identifier. The word content Bytes.i counts the number of bytes emitted by flows that map to that address. We assume the NewFlows address of a given flow is distributed uniformly between 1 and M.

All NewFlows words are reset to zero at particular instants (defined below) such that the count for a flow would exceed a certain threshold between two resets if that flow were bottlenecked. Thus, with a certain degree of imprecision, any packet that does not lead to overflow is considered to belong to a non-bottlenecked flow and is sent to the priority

```
1.   on arrival of l-byte packet p of flow f:
2.   if f ∈ ActiveList
3.     update FIFO.f
4.     Queue.f+ = l
5.   else
6.     compute the NewFlows address i of flow f
7.     if (Bytes.i = 0)
8.       Bytes.i = 1
9.       send packet p to priority queue
10.    else
11.      add f to ActiveList
12.      initialize FIFO.f
13.      Queue.f = l
14.      Deficit.f = 0
15.      add f at end of DRR schedule
```

Figure 5.2. Modified DRR packet enqueueing pseudocode using NewFlows.

queue.

It is possible to perform the above operations using a flow dependent quantum when defining the bottlenecked status (i.e., when DRR is used to perform weighted sharing)[3]. A flow mapping to word i would be deemed bottlenecked on a packet arrival if the packet size added to the current value of Bytes.i exceeds the flow quantum.

The choice of word size b determines the precision of the bottlenecked/non-bottlenecked classification. To maximize precision for a given b, the increment for a packet of length l would be $\lceil (2^b - 1)l/\text{Quantum} \rceil$.

In fact, the simplest option of setting $b = 1$ works satisfactorily on the traces we have tested. NewFlows is then simply a bitmap and any flow, whatever its quantum, is deemed to be bottlenecked and included in ActiveList if more than one packet arrive between two resets. The modification to the pseudo-code for DRR packet enqueueing with this choice of b is shown in Figure 5.2.

Packets sent to the priority queue are served until the queue empties. Service then passes to the ActiveList flow that is currently at the head of the DRR schedule. This flow emits its quantum according to lines 12 to 21 in Figure 5.1. Service returns to the priority queue if necessary between the quanta of two successive bottlenecked flows.

[3]This weight would need to be derived from packet header fields.

NewFlows is reset at times t_n, $n = 1,2,\ldots$ Epoch t_{n+1} is determined iteratively as follows:

$t_{n+1} = t_n +$ 8 MTU/FairRate(t_n)

where FairRate(t_n) is an estimate of the fair rate in bits/s currently realized by the DRR scheduler. By fair rate, we mean the rate that would currently be realized by a bottlenecked flow with the minimal quantum of MTU bytes.

The fair rate estimate can be realized in several ways. In our experiments, we evaluate the rate a fictitious bottlenecked flow could have achieved in successive constant length intervals and perform an exponentially weighted average of these rates. The choice of interval length and smoothing parameter settings are not highly critical though it is preferable to average out random fluctuations that occur on the scale of the DRR cycle time. For the results reported below we have used an interval of 100ms and a smoothing parameter of 0.9 (i.e., new average = 0.9 $\times$ old average + 0.1 $\times$ new measure).

5.1.4 Accuracy

The identification of bottlenecked flows is necessarily imperfect. The above algorithm introduces both false positives (a flow is wrongly supposed to be bottlenecked) and false negatives (a bottlenecked flow is not detected as such).

False positive errors occur in the following cases:

- a small word size b leads to an overestimation of the number of emitted bytes;
- two or more flows map to the same NewFlows address and jointly lead to overflow.

The probability of the first type of error depends on traffic characteristics and the relative quanta of different flows. For trace data we have tested, a single bit is almost as effective as 8 when all quanta are equal (see Section 5.1.5). A greater value may be necessary in weighted sharing if it is necessary to preserve low packet latency for flows doted with a relatively large quantum.

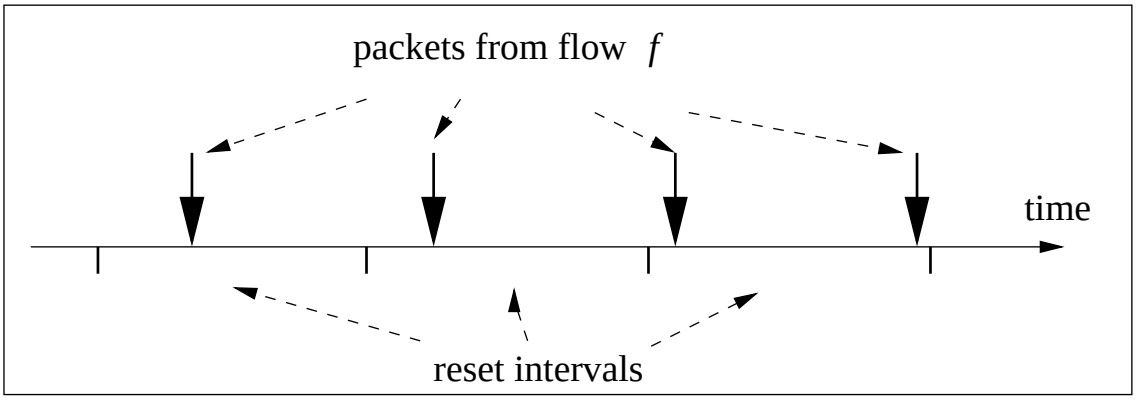
The probability of the second type of error can be controlled by the choice of the number of elements in the array NewFlows. Consider a bitmap realization ($b = 1$) and assume n packets arrive from distinct flows in some NewFlows reset interval. These arrivals give rise to x false positives if their images map into exactly $n - x$ elements. The probability of this event can be derived from classical results on occupancy theory [Fel68, page 102]. The number of false positives has approximately a Poisson distribution of mean $n^2/2M$ when M is large, indeed following the Feller's calculations we have

$$\begin{aligned}\mathbb{E}[x] = Me^{-\frac{n}{M}} - (M-n) &\sim M\left[1 - \frac{n}{M} + \frac{1}{2}\left(\frac{n}{M}^2\right) + o\left(\frac{n}{M}^2\right) - (M-n)\right] \\ &= \frac{n^2}{2M} + o\left(\frac{n^2}{M}\right)\end{aligned}$$

We discuss typical values of n is Section 5.1.5 with respect to trace data. It turns out that a small bitmap of a few thousand bits is sufficient to limit the rate of false positives to a fraction of 1%. The memory requirement for a given rate of false positives can of course be minimized by using Bloom filters [Blo70] or multistage filters [EV02], at the expense of calculating several addresses for each flow identity.

Note that the consequence of a false positive on perceived performance is slight. The impact on bottlenecked flows is negligible since their packet latency is slightly improved. Only the latency of wrongly classified non-bottlenecked flow packets is greater that what it would have been in the priority queue. However, this latency is not high (one DRR round) and still low enough for most applications.

False negative errors can occur for a flow whose incoming rate is slightly larger than the fair rate. This is illustrated in Figure 5.3 where the flow is only recognized as being bottlenecked when two packets arrive in the same reset interval. Clearly, only flows whose rate is marginally greater than the fair rate are affected by this kind of error. The impact is to extend somewhat the grey area where flows are sometimes bottlenecked and sometimes not because of fluctuations in the fair rate. This imprecision occurs naturally due to the statistical nature of traffic, whatever the scheduling algorithm.



The flow has a rate slightly greater than the fair rate but is only detected when it emits two packets in the third reset interval.

Figure 5.3. Delayed detection of a bottlenecked flow ($b = 1$)

5.1.5 Performance

To evaluate the effectiveness of the NewFlows structure in reducing the number of flows to be scheduled we have performed simulations using trace data with flows defined by the usual 5-tuple. Two traces are used:

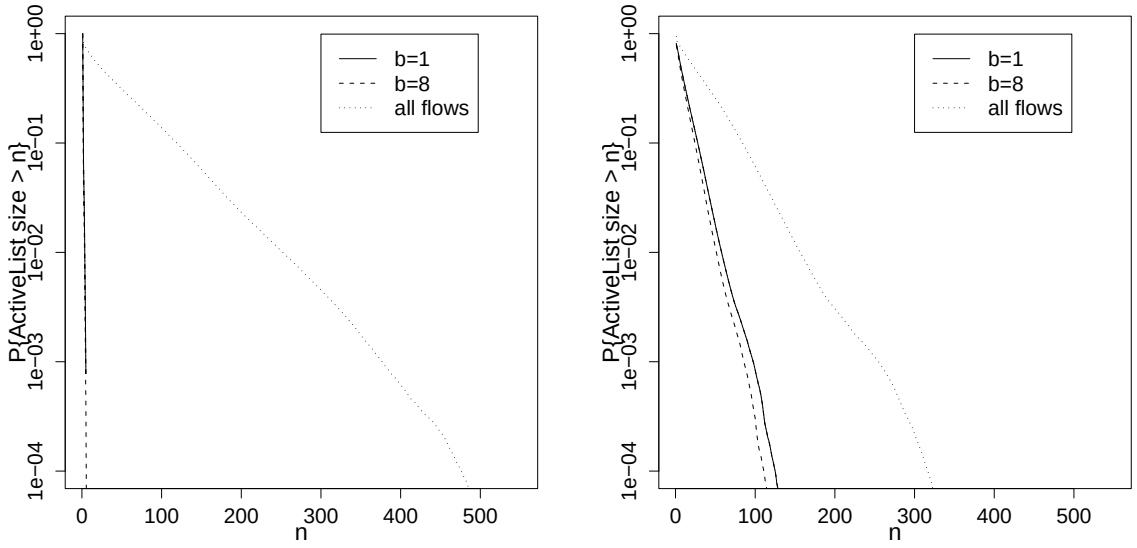


Figure 5.4. Complementary distribution of ActiveList size from trace driven simulations at utilizations 90%

- *ADSL*: five minutes of traffic recorded on a back-haul link concentrating the traffic of a large group of ADSL users: link rate 155 Mbps, utilization 28%, total number of flows 8.5×10^5;
- *Abilene*: five minutes of traffic recorded on the Abilene link between Indianapolis and Kansas: link rate 2.5 Gbps, utilization 13%, total flows 2.3×10^6.

We input the trace data to links of reduced capacity to artificially augment utilization. Figure 5.4 plots the complementary distribution of the number of flows in the DRR ActiveList for the two traces at load 90% for three configurations: without NewFlows, with a NewFlows structure with one byte words ($b = 8$), with a NewFlows bitmap ($b = 1$). All flows have the same quantum of MTU bytes.

The results show a dramatic reduction in ActiveList size for ADSL traffic. This is due to the fact that the peak rate of all flows is limited by their access line to less than 1 Mbps. No flow is bottlenecked in this case (and FIFO queueing would have been sufficient!). The reduction for the Abilene trace is roughly by a factor of 3. The bitmap works almost as well as an array of one-byte counters.

The above results were obtained with a NewFlows dimension $M = 16000$ resulting in a negligible proportion of false positives. Similar distributions are obtained with $M = 1000$, the rate of false positives increasing to around 5 10^{-3} for both traces. The formula in Section 5.1.4 predicts a rate per reset interval of $n/2M$ where n is the number of arriving packets. In the simulations, for both traces, the mean value of n is between 10 and 20

with rare peaks of 100. The formula thus predicts the empirical result.

The value of n depends on the arrival rate of packets from non-bottlenecked flows and the length of the reset interval. Using the queueing models presented in [KMOR05a], the average fair rate at 90% link utilization should not be less than $0.1C$ giving an average reset interval of at most MTU$/0.1C$. Assuming a mean size of non-bottlenecked flow packets of γMTU ($\gamma = 0.3$, say), the packet arrival rate is less than $0.9C/\gamma$MTU. The product of these two terms is $9/\gamma$ (or 27 if $\gamma = 0.3$).

Mean packet delay for flows handled by the scheduler is 0.82 ms for the Abilene trace (the transmission time of an MTU-sized packet is 33 μs). Packets sent with priority have a mean latency of only 0.05 ms. The penalty of a false positive for a non-bottlenecked is thus very small in this example.

By dimensioning ActiveList for a maximum of 100 flows, say, it will occasionally occur that a new flow should be added when the table is already full (with probability 10^{-3} for the Abilene trace). This flow might be allowed to continue transmitting its packets with priority (at the risk of longer latency for competing flows) or be constrained to share quanta with other bottlenecked flows. One approach would be to add the packets of such flows to the ActiveList flow that is currently last in the DRR cycle.

5.2 Conclusions

The present work builds on a proposal to enhance the current best effort Internet by making it flow-aware in sharing bandwidth and in controlling accessibility as necessary in overload, the latter not describe in this thesis. We have proposed reduced complexity per-flow scheduling and admission control mechanisms and evaluated their performance using trace driven simulation.

We have shown how the complexity of fair queueing, exemplified by Deficit Round Robin, can be reduced by identifying bottlenecked flows in a separate data structure. The size of the DRR schedule can then be limited to around 100 flows, this capacity being attained with low probability (10^{-3}) at relatively high loads (90% utilization). To identify the bottlenecked flows requires an additional low capacity data structure. A memory of just 1000 bits is sufficient for the trace simulations reported in this chapter.

The flow-aware networking paradigm introduced and discussed in [KOR04, OR05, KMOR05b] conforms to the original Internet design philosophy recorded by Clark [Cla88].

- It preserves the *flexibility* for edge-based service creation of the current Internet since the user-network interface is unchanged. Potential for designing new end-to-end applications is increased since developers no longer need to be friendly to legacy TCP transport.

- Network *survivability* characteristics are enhanced by the possibilities for load sensitive routing brought by admission control. Admission control is not just an alternative quality degradation to reduced throughput but a key element of any adaptive routing scheme.

- *Type of service* differentiation between delay sensitive streaming applications and throughput sensitive elastic data applications is assured without the need for explicit class of service marking. The packet latency of the former is particularly small in the fair queueing implementation proposed in this paper.

- While it remains necessary to more thoroughly evaluate the potential for new attacks brought by the flow-aware mechanisms, we believe the greater awareness of network traffic they bring reduces *vulnerability*. Knowledge of traffic at flow level is a frequent requirement for effective intrusion detection schemes.

- Congestion control algorithms, like those of *TCP*, remain essential for users to adjust emissions to the current fair rate. Network assured fairness makes it easier to introduce new transport protocols, better adapted to data transfer at very high speed, for example.

- Last in the list of features identified in [Cla88], flow-aware networking may be considered to improve *cost-effectiveness* and *accountability* by realizing assured performance without the considerable operational complexity of traditional QoS architectures.

We believe, therefore, that flow-aware networking represents a more desirable enhancement to the best effort Internet than current plans for a QoS architecture based on Diffserv-aware traffic engineering using MPLS. We hope the designs presented here will incite vendors to envisage more seriously this alternative and to explore the prospects of realization the mechanisms in their routers.

Chapter 6

Mathematical Appendix

This appendix is a author's free elaboration of known results exposed to make the reader familiar to the mathematical tools used along the thesis.
The first part is a short introduction to the theory of copula as a statistical tool to model dependence between random variables [ELM01].

The second part is a pretty complete survey on scale invariance and most of the material here presented come from a course taught by Prof. Daniele Veneziano at Politecnico di Torino during spring 2002. Prof. Veneziano teaches this short course at M.I.T. in Cambridge, MA, USA, [Ven02]. Part of the monography [DOT02] has also been a source of the presented material.

6.1 Modeling Dependence with Copula

Definition 9 *A 2-dimensional copula is a function C with domain $[0,1]^2$ such that*

(i) *C is grounded and 2-increasing*

(ii) *C has margins C_1, C_2 which satisfy $C_1(u) = u$, $C_2(u) = u \; \forall u \in [0,1]$*

It is clear by the definition that any probability measure of a random vector with uniform margins is a valid copula. What makes copulas interesting is the following simple theorem.

Theorem 10 *(Sklar's Theorem) Consider the random couple (X,Y) with probability measure $H(x,y) = \mathbb{P}\{X \leq x, Y \leq y\}$ with margins $F_X(x)$, $F_Y(y)$ then there exists a copula $C(u,v)$ such that for all $(x,y) \in \mathbb{R}^2$*

$$H(x,y) = C(F_X(x),F_Y(y)) \tag{6.1}$$

C is uniquely determined by

$$C(u,v) = H(F_X^{-1}(u),F_Y^{-1}(v)) \tag{6.2}$$

Example 11 *(Copula of indipendent variables)* X *and* Y *are indipendent if and only if*

$$C(u,v) = uv \tag{6.3}$$

Let us now consider the main measure of dependence and study their relation with copulas.

Definition 12 *Let* (X,Y) *a random couple with non zero finite variances the linear correlation coefficient is*

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} \tag{6.4}$$

Linear correlation is a measure of linear dependence. $Y = aX + b$ $(a \neq 0)$ almost surely if and only if and $|\rho(X,Y)| = 1$. In all other cases $\rho(X,Y) \in (0,1)$ and it is invariant under strictly increasing linear transformations. There are two main drawbacks of this measure of dependence, first because it is not even defined when the margins X,Y have no finite variance and second they might be misleading if margins have not joint elliptical distribution, e.g. the normal distribution.

We introduce now two different measure of dependence that are perhaps the best measure of dependence for non-elliptical distribution. They are the Kendall's tau and the Spearman's rho.

Definition 13 *Kendall's tau for the random vector* (X,Y) *is*

$$\tau(X,Y) = \mathbb{P}\{(X - \hat{X})(Y - \hat{Y}) > 0\} - \mathbb{P}\{(X - \hat{X})(Y - \hat{Y}) < 0\} \tag{6.5}$$

where $\hat{X}$,$\hat{Y}$ *are indipendent copy of* X,Y.

Hence τ is simply defined as the probability of concordance minus the probability of discordance.

Theorem 14

$$\tau(X,Y) = 4\mathbb{E}[C(U,V)] - 1 \tag{6.6}$$

where U,V $U(0,1)$.

Notice that if X,Y are indipendent $\tau(X,Y) = 4\mathbb{E}[C(U,V)] - 1 = 4\frac{1}{4} - 1 = 0$

Definition 15 *Spearman's rho for the random vector* (X,Y) *is*

$$\rho_S(X,Y) = 3\mathbb{P}\{(X - \hat{X})(Y - Y^{'}) > 0\} - \mathbb{P}\{(X - \hat{X})(Y - Y^{'}) < 0\} \tag{6.7}$$

where $\hat{X}$,$Y^{'}$ *are indipendent copy of* X,Y.

Theorem 16

$$\rho_S(X,Y) = \rho(F_X(X),F_Y(Y)) \tag{6.8}$$

where $U,V\ U(0,1)$.

These two different measure of dependence are invariant under strictly-increasing component wise transformation.

Copula can be used to describe the dependence of random variables accross the distribution especially the tail. Therefore we define a measure of tail dependence and we show how useful copulas are to calculate it.

Definition 17 *The coefficient of upper tail dependence* $\lambda_U \in (0,1]$

$$\lambda_U = \lim_{u \nearrow 1} \mathbb{P}\{F_X(X) > u | F_Y(Y) > u\} \tag{6.9}$$

if the limit exists

Definition 18 *The coefficient of lower tail dependence* $\lambda_L \in (0,1]$

$$\lambda_L = \lim_{u \searrow 0} \mathbb{P}\{F_X(X) \leq u | F_Y(Y) \leq u\} \tag{6.10}$$

if the limit exists

λ_U and λ_L can be calculated through copulas

$$\lambda_U = \lim_{u \nearrow 1} \frac{1 - 2u + C(u,u)}{1-u} \tag{6.11}$$

$$\lambda_L = \lim_{u \searrow 0} \frac{C(u,u)}{u} \tag{6.12}$$

If λ_U (λ_L) are not equal to zero we say that X and Y are upper (lower) tail dependent or asymptotically dependent. Tail dependence is a copula property.

Example 19 *(Normal Copula) Le* Φ *the standard normal univariate distribution function and* Φ^2_ρ *the standard bivariate normal distributino function with linear correlation coefficient* ρ*. Then*

$$C(u,v) = \Phi^2_\rho(\Phi^{-1}(u),\Phi^{-1}(v)) \tag{6.13}$$

is the normal bivariate copula. Note that correlation coefficient is sufficient to describe all the association of two jointly normal random variables.

6.1.1 Marshall-Olkin Copulas

Let us consider a copula that is able to model tail dependence and which will be used in the following sections as a possible model of Internet flow traffic. The structure of the dependence of this copula comes out from the following construstion. Let X_1, X_2 denote two component's lifetimes with crashes driven by three independent Poisson processes with parameter λ_1, λ_2 and λ_{12} which model crashes of the first component or the second or both. X_1, X_2 are exponentially distributed with parameters $\lambda_1 + \lambda_{12}$, $\lambda_2 + \lambda_{12}$ and the times of occurence of crashes are then independent and exponentially distributed Z_1, Z_2, Z_{12}. It is then simple to study the dependence structure using the copula.

Example 20 *Let us evaluate the complementary Copula of this model.*

$$\begin{aligned}
\bar{H}(x_1,x_2) &= \mathbb{P}\{X_1 > x_1, X_2 > x_2\} \\
&= \mathbb{P}\{Z_1 > x_1\}\mathbb{P}\{Z_1 > x_1\}\mathbb{P}\{Z_{12} > \max(x_1,x_2)\} \\
&= \exp\{-(\lambda_1 + \lambda_{12}) - (\lambda_2 + \lambda_{12}) + \lambda_{12}\min(x_1,x_2)\} \\
&= \bar{F}_{X_1}(x_1)\bar{F}_{X_1}(x_1)\min\{\exp(\lambda_{12}x_1, \exp(\lambda_{12}x_2)\} \\
&= \bar{F}_{X_1}(x_1)\bar{F}_{X_2}(x_2)\min\{\bar{F}_{X_1}(x_1)^{-\alpha_1}, \bar{F}_{X_2}(x_2)^{-\alpha_2}\}
\end{aligned}$$

Therefore

$$\hat{C}(u,v) = \bar{H}(\bar{F}_{X_1}^{-1}(u), \bar{F}_{X_2}^{-1}(v)) = uv\min(u^{-\alpha_1}, v^{-\alpha_2}) \tag{6.14}$$

where $\alpha_1 = \lambda_1/(\lambda_1 + \lambda_{12})$, $\alpha_2 = \lambda_2/(\lambda_2 + \lambda_{12})$.

It is a simple exercise to evalutate other useful parameters from the Copula.

$$C(u,v) = 1 - u - v - \hat{C}(u,v) = \min(u^{1-\alpha_1}v, uv^{1-\alpha_2}) \tag{6.15}$$

$$\tau(C) = \tau(X,Y) = 4\mathbb{E}[C(U,V)] - 1 = \frac{\alpha_1\alpha_2}{\alpha_1 + \alpha_2 - \alpha_1\alpha_2} \tag{6.16}$$

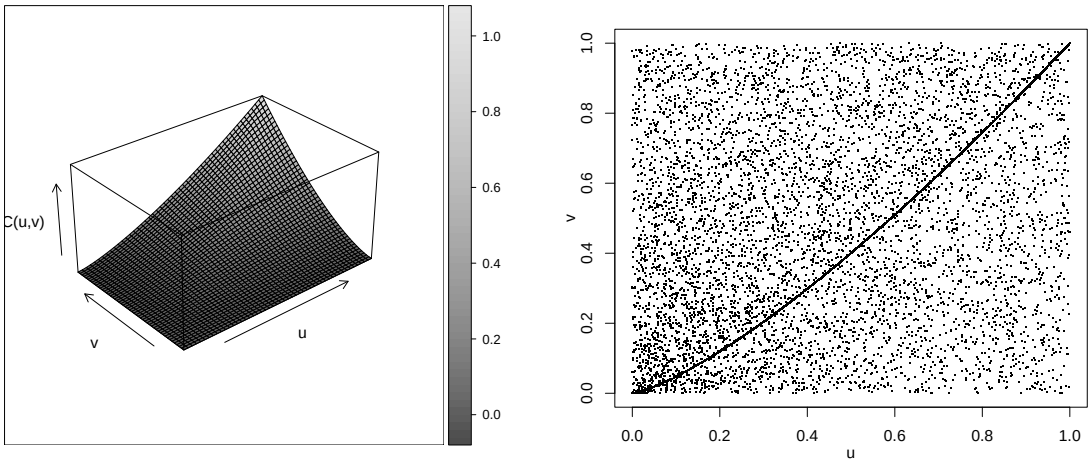
$$\rho_S(C) = \rho_S(X_1,X_2) = \rho(F_{X_1}(X_1), F_{X_2}(X_2)) = \frac{3\alpha_1\alpha_2}{2\alpha_1 + 2\alpha_2 - \alpha_1\alpha_2} \tag{6.17}$$

$$\lambda_U = \lim_{u \searrow 0} \frac{\hat{C}(u,u)}{u} = \min(\alpha_1\alpha_2) \tag{6.18}$$

This copula is easy to fit from to a data sample, because α_1 and α_2 can be directly expressed through τ and λ_U which can be estimated from a sample and this is better shown in the following sections.

$$\min\{\alpha_1,\alpha_2\} = \lambda_U \tag{6.19}$$

$$\max\{\alpha_1,\alpha_2\} = \frac{\lambda_U\tau}{\lambda_U + \lambda_U\tau - \tau} \tag{6.20}$$



(a) (b)

Figure 6.1. Marshal-Olkin Copula with $\tau = 0.3$, $\lambda_U = 0.4$. C(u,v) in the subplot (a) and the simulation the copula in the subplot (b)

6.2 Long Range Dependence, Self Simularity, Multifractals

Lévy stable distributions

The class of *L-stable* distributions received growing attention in the last thirty years thanks to the well developed theory in the field of applied probability and the large number of applications that make use of them. This class of distributions comes out from the normal distribution and can be seen as a generalization of it.

Let us recall the following problem related to the normal distribution. Let X,Y two independent zero mean normal random variables with $\mathbb{E}[X^2] = \sigma_X^2$, $\mathbb{E}[Y^2] = \sigma_Y^2$ then $X + Y$ is itself Gaussian with $\mathbb{E}[(X + Y)^2] = \sigma_X^2 + \sigma_Y^2$. The normal distribution is invariant under the addition of independent random variables and it is the unique solution of the following functional problem if $\mathbb{E}[X^2] < \infty$.

$$\begin{cases} s_1 X_1 + s_2 X_2 = sX \\ s_1^2 + s_2^2 = s^2 \end{cases}$$

where X_1,X_2 are independent copies of X. Let us consider a different distribution, e.g. the Cauchy distribution, with density $1/\pi(1 + x^2)$, which is often used as a counterexample. Indeed the Cauchy distribution is the unique solution of the following problem

$$\begin{cases} s_1 X_1 + s_2 X_2 = sX \\ s_1 + s_2 = s \end{cases}$$

and with the condition $s_1^{1/2}+s_2^{1/2} = s$, the solution is a distribution with density $1/\sqrt{2\pi}x^{-3/2}e^{-x/2}$. The study of this functional problem has been deeply studied by Lévy and the problem is usually referred as *Lévy stability*

$$\begin{cases} s_1 X_1 + s_2 X_2 = sX \\ s_1^\alpha + s_2^\alpha = s^\alpha \end{cases}$$

which has a closed form solution for $0 < \alpha \leq 2$ given by the weighted difference of two identically distributed random variables with weights $1/2(1+\beta)$,$1/2(1-\beta)$. The distribution is known in a general form through its characteristic function $f(t) = \mathbb{E}[e^{itX}]$ due to Lévy. When $\alpha > 2$ there is no closed form solution.

$$\log \mathbb{E}[e^{itX}] = i\mu\, t - \gamma\, t^\alpha \left[1 + i\beta \left(\frac{t}{|t|}\right) \tan\left(\alpha\frac{\pi}{2}\right)\right] \tag{6.21}$$

Let us use the following notation for a Lévy stable distribution $S(\alpha,\beta,\gamma,\mu)$ though it can be easily verified that $S(2,\beta,\sigma^2/2,\mu) = \mathcal{N}(\mu,\sigma^2)$ (β arbitrary choosen) and $S(1,0,\gamma,\mu)$ is a non centered Cauchy distribution. This important class of distributions generalizes the centrality of the normal distribution at the light of the following result. The *domain of attraction* of an L-stable distribution is the set of all the distributions $F_\Theta(\theta)$ whose i.i.d. sequence $\{\Theta_i\}_{i=1}^n$ satisfies the following relation

$$\zeta_n = \frac{1}{b_n}\sum_{i=1}^{n} \Theta_i - a_n \tag{6.22}$$

$$\zeta_n \rightarrow S(\alpha,\beta,\gamma,\mu) \tag{6.23}$$

for certain a_n, b_n. In particular $b_n \sim n^{-1/\alpha}$. All the finite variance random variables stay in the domain of attraction of the normal distribution (in this special case $\alpha = 2$) and $b_n \sim n^{-1/2}$ which is the well known rate of convergence of the finite variance sum of random variables to the normal distribution (Central Limit Theorem).

6.2.1 Self similar Process (self-affine)

Let us consider a real-valued stochastic process $Z = \{Z(t)\}_{t\in\mathbb{R}}$. Z is said to be *H (statistical) self similar* (H-ss) if, for any $a > 0$

$$\{Z(t),t \in \mathbb{R}\} \stackrel{d}{=} \{a^{-H}Z(at),t \in \mathbb{R}\},\ H > 0^1 \tag{6.24}$$

Sometimes the term *self-affine* is used in lieu of *self similar* because the transformation scales time and space differently. We continue to use the more frequently used *self similar*

[1]The equality is for finite dimensional distributions

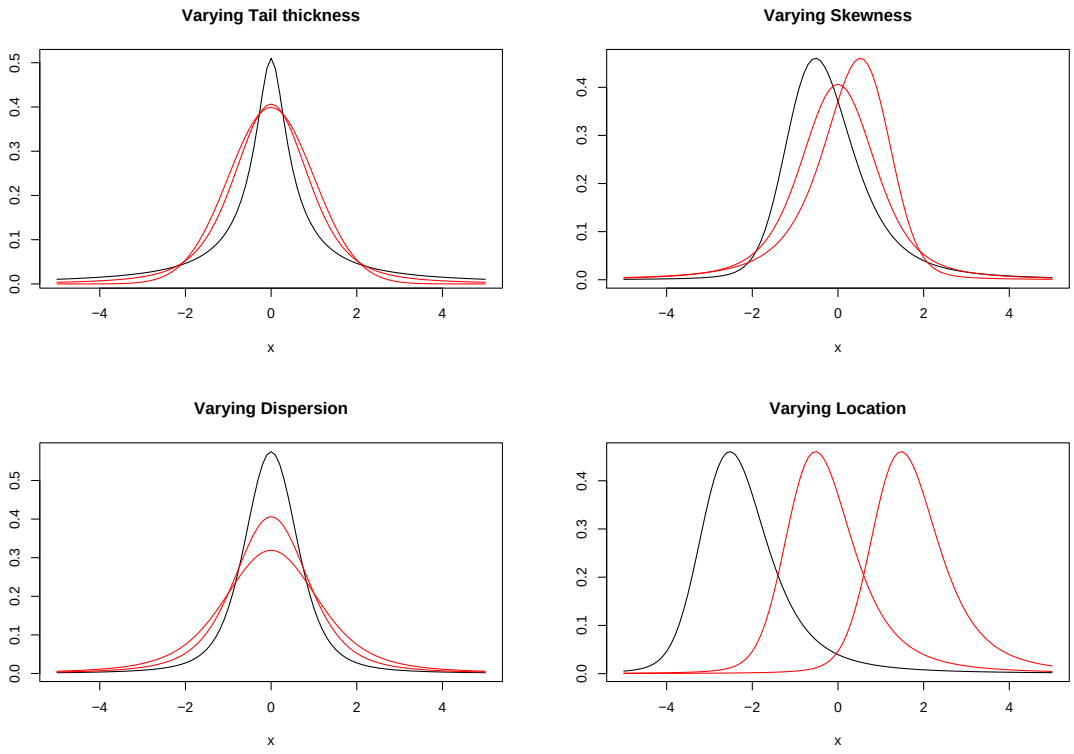


Figure 6.2. L stable distributions varying tailness α, skewness β, dispersion γ, location μ.

and show some interesting properties of this class of stochastic process. Let us observe a simple but fundamental fact for $H-ss$ process.

$$\mathbb{E}[Z(t)] = a^{-H}\mathbb{E}[Z(at)] \Rightarrow \mathbb{E}[Z(1)]/\mathbb{E}[Z(a)] \neq 1 \tag{6.25}$$

which means that a non degenerate $H-ss$ process ($Z \equiv 0$ is degenerate) cannot be stationary.

Proposition 21 *If $\{Z(t), t \in \mathbb{R}\}$ is H-ss then the process $\{Y(t) = e^{-tH}Z(e^t), t \in \mathbb{R}\}$ is stationary. Conversely if $\{Y(t)\}_{t\in\mathbb{R}}$ is stationary $Z(t) = t^H Y(lnt)$ is H-ss.*

Proposition 22 *The first two moment of an $H-ss$ process are*

$$\mathbb{E}[X(t)] = t^H\mathbb{E}[X(1)] \tag{6.26}$$
$$\mathbb{E}[X(t)^2] = t^{2H}\mathbb{E}[X(1)^2] \tag{6.27}$$

Proof

$$\begin{aligned}
\mathbb{E}[Y(t)] &= \mathbb{E}[Y(0)] = \mathbb{E}[X(1)] \\
\mathbb{E}[Y(t)^2] &= \mathbb{E}[Y(0)^2] = \mathbb{E}[X(1)^2] \\
\mathbb{E}[X(t)] &= \mathbb{E}[t^H Y(\ln t)] = t^H \mathbb{E}[Y(\ln t)] \\
&= t^H \mathbb{E}[X(1)] \\
\mathbb{E}[X(t)^2] &= \mathbb{E}[t^{2H} Y(\ln t)^2] = t^{2H} \mathbb{E}[Y(\ln t)] \\
&= t^{2H} \mathbb{E}[X(1)^2]
\end{aligned}$$

Our interest in $H-ss$ is actually motivated from the fact that a subclass of self-similar processes has stationary increments which are Long Range Dependent.

Definition 23 *(Long Range Dependence) Let $\{Z_k\}_{k\in\mathbb{Z}}$ be a wide sense stationary random sequence, μ its mean, $\gamma(n)$ its autocovariance function, and $f(\nu)$ $\nu \in [-\pi,\pi]$ its spectral density,*

$$\begin{gathered}
\mathbb{E}[Z_k] = \mu \\
\mathbb{E}[(Z_k - \mu)(Z_{k+n} - \mu)] = \gamma(n),\ n \in \mathbb{Z} \\
f(\nu) = \tfrac{1}{2\pi} \sum_{n=-\infty}^{+\infty} e^{-i\nu n} \gamma(n) \\
\gamma(n) = \int_{-\pi}^{\pi} f(\nu) e^{-i\nu n} d\nu
\end{gathered} \tag{6.28}$$

$\{X_k\}_{k\in\mathbb{Z}}$ is said to be Long-Range Dependent *if*

$$\gamma(n) \sim a n^{\alpha-1},\ \ n \to \infty,\ \alpha \in (0,1) \tag{6.29}$$

or if

$$f(\nu) \sim c|\nu|^{-\alpha}, \nu \to 0,\ \alpha \in (0,1) \tag{6.30}$$

where $f(x) \sim g(x)$ as $x \to x_0$ means $\lim_{x\to x_0} f(x)/g(x) = 1$. Equations (6.29) and (6.30) are equivalent if $\gamma(n)$ is monotone. In the following we use the definition in (6.30), that is based on two parameters: (α,c). The parameter $\alpha \in [0,1)$ is the dimensionless scaling exponent, and describes the "intensity" of LRD; for a non-LRD stationary process, $\alpha = 0$ at large scales. The parameter $c \in \mathbb{R}^+$ has the same dimension of the variance of the process and describes the quantitative aspect of LRD often referred to as the *LRD size*. LRD implies that the sum of correlations over all lags is infinite; however, individually, their sizes at large lags are proportional to c, and can be arbitrarily small. LRD is usually associated with *self similar processes with stationary increments (H-sssi[2])* and in particular with normal marginals.

Definition 24 *A H-sssi process is an H-ss with $\Delta\, Z(t) = Z(t+\tau) - Z(t)$ wide sense stationary.*

[2] *H-selfsimilar with stationary increments*

Such a class of processes $\{Z(t)\}_{t\in\mathbb{R}}$ satisfies a number of properties:

(i) $Z(0) = 0$ a.s. (almost surely) which follows from $Z(0) = Z(a0) \stackrel{d}{=} a^H Z(0)$ for any $a > 0$.

(ii) if $H \neq 1 \Rightarrow \mathbb{E}Z(t) = 0 \ \forall\, t \in \mathbb{R}$. This follows from self-similarity $\mathbb{E}Z(2t) = 2^H \mathbb{E}Z(t)$ while stationarity of $\Delta\, Z(t)$ implies $\mathbb{E}Z(2t) = \mathbb{E}(Z(2t) - Z(t)) + \mathbb{E}(t) = 2\mathbb{E}Z(t)$.

(iii) by (ii) and stationarity of $\Delta\, Z(t)$, $Z(-t) = Z(-t) - Z(0) \stackrel{d}{=} Z(t) - Z(0) = Z(t)$.

(iv) by (iii) and selfsimilarity, $\mathbb{E}Z^2(t) = \mathbb{E}Z^2(|t|sgn(t)) = |t|^{2H}\mathbb{E}Z^2 sgn(t) = |t|^{2H}\mathbb{E}Z^2(1)$ if $\mathbb{E}Z^2(1) = \sigma^2 = 1$ we will say that the process $Z(t)$ is standard.

(v) by (iv) and stationarity of $\Delta\, Z(t)$, the covariance function $\Gamma(s,t) = Cov(Z(s),Z(t)),s,t \in \mathbb{R}$ is given by

$$\begin{aligned} \Gamma(s,t) &= \frac{1}{2}\Big[(Z(s)^2 + Z(t)^2 - (Z(s) - Z(t))^2\Big] \\ &= \frac{\sigma^2}{2}\Big[(|s|^2 + |t|^2 - |s-t|^2\Big] \end{aligned} \tag{6.31}$$

(vi) $H \leq 1$. Indeed

$$\begin{aligned} \mathbb{E}|Z(2)| &= \mathbb{E}|Z(2) - Z(1) + Z(1))| \leq \mathbb{E}|Z(2) - Z(1)| + \mathbb{E}|Z(1))| \\ &= 2\mathbb{E}[Z(1)] \\ \mathbb{E}|Z(2)| &= 2^H \mathbb{E}[Z(1)] \text{because of self-similarity} \\ \Rightarrow 2^H &\leq 2 \Rightarrow H \leq 1 \end{aligned}$$

(vii) $H = 1$ is a trivial case where the process degenerate. Indeed

$$\begin{aligned} \mathbb{E}[Z(t) - tZ(1)]^2 &= \mathbb{E}[Z(t)^2] + t^2\mathbb{E}[Z(1)^2] - 2t\mathbb{E}[Z(t)Z(1)] \\ &= (t^2 + t^2 - 2t^2)\mathbb{E}[Z(1)^2] = 0 \end{aligned}$$

$\Rightarrow Z(t) \stackrel{a.s.}{=} tZ(1)$ degenerated in a line.

Proposition 25 *if $\{Z(t)\}_{t\in\mathbb{R}}$ is H-sssi its paths are continuous and non differentiable.*

Proof Continuity follows from the Kolmogorov criterion which says that every process admits a version with continuous paths if there exist $p \geq 1$ and $\theta > 1$ such that

$$\mathbb{E} \mid Z(t_2) - Z(t_1) \mid^p \leq\ c|t_2 - t_1|^\theta \tag{6.32}$$

for H-sssi $p > 1/H$

$$\mathbb{E} \mid Z(t_2) - Z(t_1) \mid^p = \mathbb{E}|Z(1)|^p |t_2 - t_1|^{pH} \tag{6.33}$$

and non differentiability follow from

$$\lim_{t_2 \to t_1} \mathbb{E}\Big| \frac{Z(t_2) - Z(t_1)}{t_2 - t_1} \Big|^p = \mathbb{E}|Z(1)|^p |t_2 - t_1|^{pH-2} \to \infty \tag{6.34}$$

Definition 26 *(Fractional Brownian Motion) A Gaussian H-sssi process $B_H(t)_{t\in\mathbb{R}}$ with $0 < H \leq 1$ is called* fractional Brownian motion *(fBm). If $\mathbb{V}[B_H(1)] = 1$ it is called standard. $B_H(t)$ is the unique Gaussian H-sssi process. For $H = 1/2$ $B_H(t)$ is the* Wiener Brownian motion *and for $H = 1$ $B_1(t) = B_1(1)t$, a line with random Gaussian slope. The process of the stationary increments of $B_H(t)$ is called* fractional Gaussian noise *and is Long Range Dependent if $1/2 < H < 1$.*

Proof (of Long Range Dependence)

$$\begin{aligned}
\gamma(k) &= \mathbb{E}[\Delta X(t) \Delta X(t+k)] \\
&= \mathbb{E}[\Delta X(0) \Delta X(k)] \\
&= \mathbb{E}[X(\tau)(X(\tau+k) - X(k))] \\
&= \mathbb{E}[X(\tau)X(\tau+k)] - \mathbf{E}[X(\tau)X(\tau+k)] \\
&= \frac{1}{2}\mathbf{E}[\tau^{2H} + (\tau+k)^{2H} - k^{2H} - \tau^{2H} + \\
&\quad -k^{2H} - (\tau-k)^{2H}]\mathbf{E}[X(1)^2] \\
&= \frac{1}{2}\mathbb{E}[(k+\tau)^{2H} - 2k^{2H} - (k-\tau)^{2H}]\mathbb{E}[X(1)^2] \\
&\sim H(2H-1)k^{2H-2}\mathbb{E}[X(1)^2]_{\ k\to\infty} = c_\gamma k^{2H-2}_{\ \ k\to\infty}
\end{aligned}$$

6.2.2 Multifractal Process (stochastic self similar)

Let us introduce the class of Multifractal processes which can be viewed as a generalization of the class of selfsimilar process. Let us define this class in a formal way.

Definition 27 *A stochastic process $\{X(t), t \in \mathbb{R}\}$ is said to be Multifractal if*

$$X(at) \stackrel{d}{=} M(a)X(t) \tag{6.35}$$

with $M(a) = \log_a H(a)$ where X and M are independent. The we can rewrite (6.35): $X(at) \stackrel{d}{=} a^{H(a)}X(t)$.

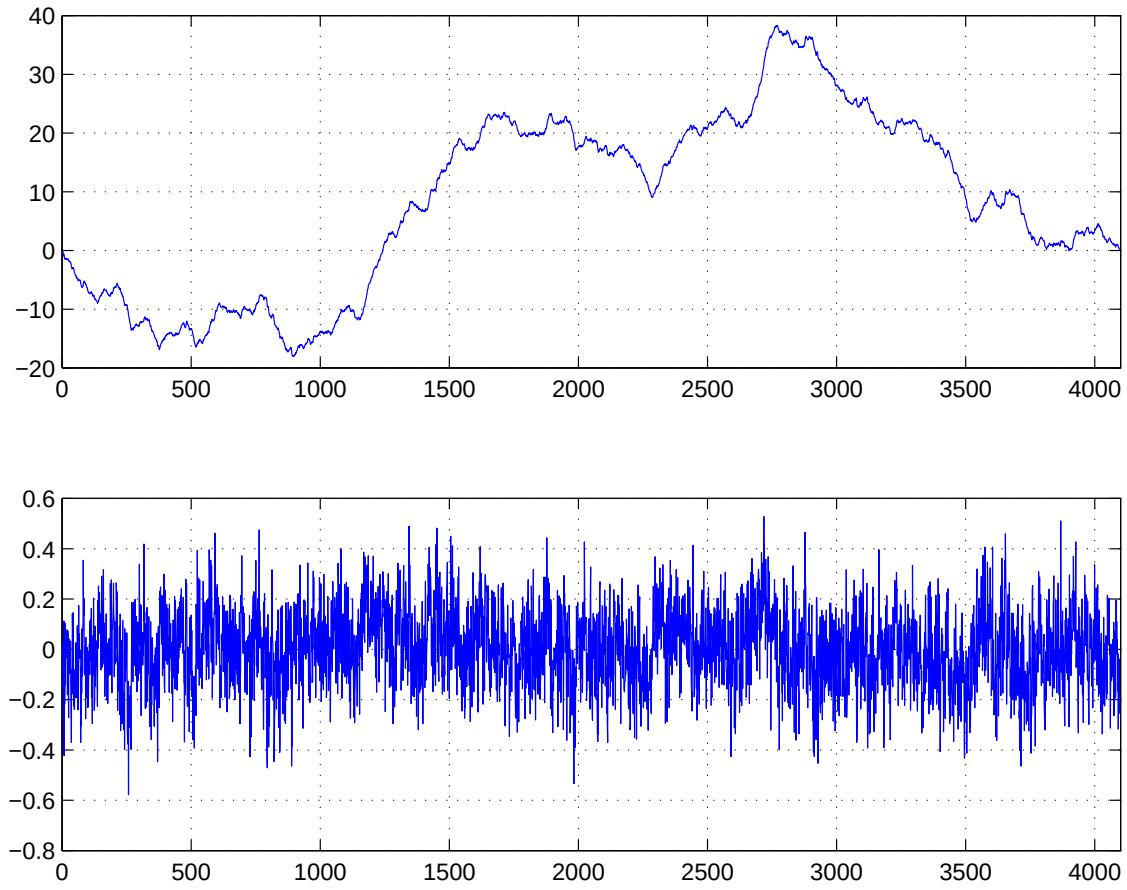


Figure 6.3. fBm (top plot) and fGn (bottom plot) with H=0.8

It is clear that the class of self similar process is a subclass of Multifractals sometimes called *stochastic self similar* process (some authors abbreviate stochastic self similar into H-sss but it can be misleading and we prefer to avoid this practice). Let us observe the main properties of a mf (multifractal). First it is clear that a mf is completely described by the distribution of $H(a)$. Given that $H(a)$ is distributed it is possible to have stationary mf processes. The distribution of $H(a)$ is in a class of distributions called infinitely divisible (i.d.).

Definition 28 *A random variable X is said to be i.d. if it can be written as the sum i.i.d. random variables* $\{\theta_i\}_{i=1}^n$.

$$X = \sum_{i=1}^{n} \theta_i$$

The characteristic function $\psi(t)$ of a i.d. distribution is known in a very general form and due to *Lévy and Khintchine*. The Normal, Gamma, Poisson are for example i.d.

$$\psi(t) = e^{\phi(t)} \tag{6.36}$$

Proposition 29 *H(a) is i.d. distributed.*

Proof

$$\begin{aligned} X(t) &= a_1^{-H_{a_1}} X(a_1 t) = Y(t) \\ X(t) &= a_2^{-H_{a_2}} X(a_2 t) \\ Y(t) &= a_2^{-H_{a_2}} Y(a_2 t) = a_1^{-H_{a_1}} a_2^{-H_{a_2}} X(a_1 a_2 t) = X(t) \end{aligned}$$

Iterating this procedure n times for different values of $a_i = a^{1/n}$ than we have

$$\begin{aligned} X(t) &= \left(\prod_{i=1}^{n} a_i^{-H_{a_2}} \right) X \left(\prod_{i=1}^{n} a_i t \right) \\ X(t) &= \left(a^{-\sum_{i=1}^{n} H_{a^{1/n},i}} \right) X(at) \\ H_a &= \sum_{i=1}^{n} H_{a^{1/n},i} \end{aligned}$$

and $H_{a^{1/n},i}$ i.i.d. with the distribution of $H_{a^{1/n}}$. It is now clear the distribution of $H(a)$ for given a permit to evalate the distribution for all a.

$$\psi_{H_a}(t) = \left[\psi_{H_{a'}}(t) \right]^{log_{a'}(a)}$$

Observe again that this approach is equivalent to study the moments $\mathbb{E}[|X(t)|^q]$ of the mf process. Indeed

$$\begin{aligned} \mathbb{E}[|X(t)|^q] &= c(q) t^{\zeta(q)} \\ \mathbb{E}[|a^{-H_a}|^q] &= \mathbb{E}[e^{H_a q ln(1/a)}] = e^{\zeta(q)} \end{aligned}$$

Therefore the analysis cab be limited to the function $\zeta(q)$ which include all the scaling properties of $X(t)$. A multifractal process, differently from a self similar, has a local degree of regularity which is variable along the path of the trajectory. Suppose $X(t)$ to be stationary then $X(t_1) \stackrel{d}{=} X(t_2)$

$$\mathbb{E}[|X(t+dt) - X(t)|] = c(1)\mathbb{E}[|X(t+adt) - X(t)|] \leq dt^{\zeta(1)}$$

Let us consider $\{X(t),t \in [0,1]\}$ without loss of generality, and the dyadic subinterval of $[0,1]$, $I^n_{k_n} \in [0,1]$, $I_{k_n} = [2^{-n}k,2^{-n}(k+1)]$, $k_n = 0,\ldots,2^n-1$. Let

$$\begin{aligned} \Delta^n_{k_n}[X] &= |X(2^{-n}(k+1)) - X(2^{-n}k)| \\ S_q(n) &= 2^{-n}\mathbb{E}\left[\sum_{k_n=0}^{2^n-1}(\Delta^n_{k_n}[X])^q\right] \\ \zeta(q) &= \lim_{n\to\infty} -\frac{1}{n} log_2 S_q(n) \end{aligned}$$

Example 30 *(the binomial measure) The binomial measure is a multiplicative measure for construction, and it is a illuminating example to understand how to build a multifractal measure useful to model trading time. Let us consider our dyadic representation of* $[0,1]$*. At the first we divide* $[0,1]$ *in two equal part whome we associate a mass* m_0 *with probability* m_0 *and mass* $m_1 = 1 - m_0$ *with probability* m_1*. Iterating the procedure until the step* n *we have associated a mass* $m_{\epsilon_1}m_{\epsilon_2}\ldots m_{\epsilon_n}$ *to the interval* $I_{\epsilon_1\epsilon_2\ldots\epsilon_n}$ *with* $\epsilon_i \in \{0,1\}$*. Then a measure is induced* $\mu_k(I_{\epsilon_1\epsilon_2\ldots\epsilon_n}) = \mu_n(I_{\epsilon_1\epsilon_2\ldots\epsilon_n})$ *per* $k \geq n$*. Then* μ *is the limit measure of* μ_n *where* $\mu(I_{\epsilon_1\epsilon_2\ldots\epsilon_n}) = m_{\epsilon_1}m_{\epsilon_2}\ldots m_{\epsilon_n}$*. At each step every interval is equally divided in two interval then the length of the interval is reduced by a* $1/2$ *factor while the relative mass is reduced by a* m_i *factor. Then it easy to see that the measure satisfy the following:*

$$\mu([a,b]) = m_0\mu([2a,2b]) + m_1\mu([2a-1,2b-1])$$

with $a,b \in [0,1]$*. This measure is called* binomial measure *or binomial cascade. It is straight forward to extend this constraction to the* $[0,t]$ *interval. The cumulative distribution function of* μ *is* $M(t) = \int_0^t \mu(u)du$ *can be used as* trading time *for a compound process to induce mf properties.*

6.2.3 Multifractal analysis: examples

In this section we will analyze three important examples. The fBm, the binomial cascade and the fBm in multifractal time and we show the mf analysis evaluating the $\zeta(q)$ in these simples cases.

Fractional Brownian Motion. Let us evaluate $\zeta(q)$ for fBm. Given that the fBm has stationary increments $\Delta^n_{k_n}[B] = \Delta^n[B]$ and that $B(0) = 0$, than we have

$$\begin{aligned} S_q(n) &= 2^{-n}\mathbb{E}\left[\sum_{k_n=0}^{2^n-1}(\Delta^n_{k_n}[B])^q\right] \\ &= \mathbb{E}[|B(2^{-n})|^q] = 2^{-nqH}\mathbb{E}[|B(1)|^q] \\ \zeta(q) &= \lim_{n\to\infty} -\frac{1}{n} log_2 S_q(n) = \lim_{n\to\infty} -\frac{1}{n} log_2 2^{-nqH}\mathbb{E}[|B(1)|^q] = qH \end{aligned}$$

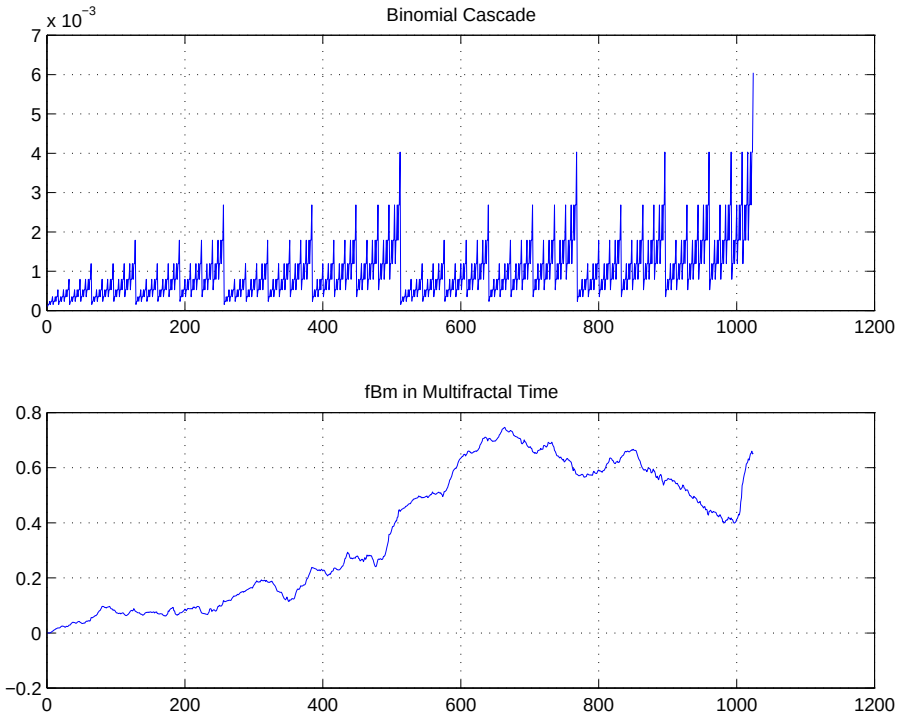


Figure 6.4. Binomial cascade with $m_0 = 0.4$ (top) and fBm in multifractal time (bottom)

The binomial cascade. Let us observe that the construction procedure of a binomial cascade is describe by a random tree and a Bernoulli scheme can be associated to a leaf of the tree in order to evaluate the probability measure of a path along the tree from the root node to a certain leaf. Then the calculus is straightforward.

$$\begin{aligned} S_q(n) &= 2^{-n}\mathbb{E}\left[\sum_{k_n=0}^{2^n-1}(\Delta_{k_n}^n[M])^q\right] \\ &= 2^{-n}\sum_{k_n=0}^{2^n-1}\mathbb{E}[|\mu(I_{k_n}^n)|^q] = 2^{-n}\sum_{k_n=0}^{2^n-1}\mathbb{E}\left[\prod_{s=0}^{n} m_{\epsilon_s}^q\right] = 2^{-n}\sum_{k_n=0}^{2^n-1}\prod_{s=0}^{n}\mathbb{E}[m_{\epsilon_s}^q] \qquad (6.37) \\ &= \mathbb{E}[m_{\epsilon_0}^q]^n \\ \zeta(q) &= \lim_{n\to\infty} -\frac{1}{n} log_2 S_q(n) = \lim_{n\to\infty} -\frac{1}{n} log_2 \mathbb{E}[m_{\epsilon_0}^q]^n = -log_2\left[m_0^{q+1} + (1-m_0)^{q+1}\right] \end{aligned}$$

fBm in multifractal time. This process $X(t)$ is a compound process obtained from a fBm and a multifractal non decreasing process like the c.d.f. $M(t)$ of a mf measure. Then

$X(t) \equiv B_H(M(t))$.

$$\begin{aligned} S_q(n) &= 2^{-n}\mathbb{E}\left[\sum_{k_n=0}^{2^n-1} |\Delta_{k_n}^n[B(M)]|^q\right] \\ &= \mathbb{E}[|B(1)|^q]2^{-n}\sum_{k_n=0}^{2^n-1}\mathbb{E}|\Delta_{k_n}^n[M]|^{Hq} \\ &= \mathbb{E}[|B(1)|^q]\mathbb{E}[m_{\epsilon_0}^{Hq}]^n \\ \zeta(q) &= \lim_{n\to\infty} -\frac{1}{n} log_2 S_q(n) = \lim_{n\to\infty} -\frac{1}{n} log_2 \mathbb{E}[|B(1)|^q]\mathbb{E}[m_{\epsilon_0}^{Hq}]^n \\ &= -log_2\left[m_0^{Hq+1} + (1-m_0)^{Hq+1}\right] = \zeta_{M(t)}(Hq) \end{aligned}$$

because of

$$\mathbb{E}[|B_H(M(t+\Delta\, t)) - B_H(M(t))|^q] = \mathbb{E}[|M(t+\Delta\, t) - M(t)|^{Hq}]\mathbb{E}[|B(1)|^q]$$

This result is useful in practice because it shows that the multifractal component can be modeled directly on the mf random measure $M(t)$.

6.2.4 Multi Resolution Analysis

Let us consider the Hilbert space $V = L^2(\mathbb{R}) = \{x(t) : \int_{-\infty}^{+\infty} x(t)^2\, dt < \infty\}$, with the usual euclidean scalar product. Consider now the sub-vector spaces of V so defined

$$\ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots$$

such that they satisfies the following property

$$\bigcup_{m=-\infty}^{+\infty} V_m = V$$

$$\bigcap_{m=-\infty}^{+\infty} V_m = \emptyset$$

It is evident that these sub-spaces are also Hilbert spaces. Given $x(t) \in V$, we write

$$A_m\{x(t)\} = \underset{v(t)\,\in\, V_m}{argmin}\ ||x(t) - v(t)|| \tag{6.38}$$

then

$$\lim_{m\to+\infty} A_m\{x(t)\} = x(t)$$

$$\lim_{m\to-\infty} A_m\{x(t)\} = 0$$

While m increases A_m represents a progressively better estimation of $x(t)$. We define the following set of functions

$$\ldots,\phi^m_{-2}(t),\phi^m_{-1}(t),\phi^m_0(t),\phi^m_1(t),\phi^m_2(t),\ldots$$

such that

$$A_m\{x(t)\} = \sum_{n=-\infty}^{+\infty} a^m_n \phi^m_n(t) \tag{6.39}$$

$$a^m_n = \int_{-\infty}^{+\infty} x(t)\phi^m_n(t)\,dt = a^m_n = \langle x(t),\phi^m_n(t)\rangle \tag{6.40}$$

$\langle,\rangle$ denotes the scalar product. $\phi^m_n(t)$ with $n,m \in \mathbb{Z}$ are choosen such that:

$$v(t) \in V_m \Rightarrow v(t-a) \in V_m \tag{6.41}$$

$$v(t) \in V_m \Rightarrow v(2t) \in V_{m+1} \tag{6.42}$$

by 6.41 and 6.42 all these function can be obtained from a *father* function $\phi(t)$ such that an orthonormal base for each sub-space is given by

$$\phi^m_n(t) = 2^{m/2}\phi(2^m t - n)$$

$\forall m,n \in \mathbb{Z}$. Every function in V is approximated iteratively from finer to coarser time scales and at each iteration the error of the representation has the follwing form.

$$A_{m+1}\{x(t)\} - A_m\{x(t)\} = D_m\{x(t)\}$$

Let us consider the vectors $v(t) \in O_m$ where O_m is such that $O_m \oplus V_m = V_{m+1}$ and $O_m \perp V_m$, i.e. the subspace of the functions $d_m(t)$ such that $v_{m+1}(t) - v_m(t) = d_m(t)$ with $v_{m+1}(t) \in V_{m+1}$ and $v_m(t) \in V_m$. By iterating the procedure we obtain

$$V_{M+1} = O_M \oplus V_M = O_M \oplus O_{M-1} \oplus V_{M-1} = \ldots = \bigoplus_{m=-\infty}^{M} O_m$$

which defines an equivalent representation of V through the subspaces O_m whose base is

$$\ldots,\psi^m_{-2}(t),\psi^m_{-1}(t),\psi^m_0(t),\psi^m_1(t),\psi^m_2(t),\ldots$$

such that $D_m\{x(t)\} \in O_m$

$$D_m\{x(t)\} = \sum_{n=-\infty}^{+\infty} x^m_n \psi^m_n(t)$$

where

$$x_n^m = \int_{-\infty}^{+\infty} x(t)\psi_n^m(t)\,dt$$

Every function can be written

$$\begin{aligned} x(t) &= \sum_{n=-\infty}^{+\infty} a_n^M \phi_n^M(t) + \sum_{m=-\infty}^{M}\sum_{n=-\infty}^{+\infty} x_n^m \psi_n^m(t) \qquad (6.43)\\ &= \lim_{M\to+\infty} \sum_{n=-\infty}^{+\infty} a_n^M \phi_n^M(t) \\ &= \sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty} x_n^m \psi_n^m(t) \end{aligned}$$

$$a_n^m = \langle x(t), \phi_n^m(t)\rangle$$

$$x_n^m = \langle x(t), \psi_n^m(t)\rangle$$

$$\phi_n^m(t) = 2^{m/2}\phi(2^m t - n)$$

$$\psi_n^m(t) = 2^{m/2}\psi(2^m t - n)$$

$\phi(t)$ e $\psi(t)$ are called *father* and *mother wavelet* respectively. The *mother wavelet* is chosen such that the series 6.43 are convergent on V. A necessary and sufficient condition is that

$$\int_{-\infty}^{+\infty} \psi(t)\,dt = 0$$

The *mother wavelet* has also N *vanishing moments* if

$$\int_{-\infty}^{+\infty} t^N \psi(t)\,dt = 0$$

Discrete Wavelet Transform (DWT)

DWT is the algorithm that allows to efficiently evaluate scales and details a_n^m, x_n^m. Let us consider the two discrete sequences

$$h[n] = \int_{-\infty}^{+\infty} \phi_n^1(t)\phi_0^0(t)\,dt$$

$$g[n] = \int_{-\infty}^{+\infty} \phi_n^1(t)\psi_0^0(t)\,dt$$

and a_n^m, x_n^m can be obtained as

$$a_n^m = \langle\, x(t),\phi_n^m(t)\rangle = \{x(t) * \phi_0^m(-t)\}|_{t=2^{-m}n} = \sum_{l=-\infty}^{+\infty} h[l-2n]a_l^{m+1}$$

$$x_n^m = \langle\, x(t),\psi_n^m(t)\rangle = \{x(t) * \psi_0^m(-t)\}|_{t=2^{-m}n} = \sum_{l=-\infty}^{+\infty} g[l-2n]a_l^{m+1}$$

because $\phi_0^0(t),\psi_0^0(t) \in V_0 \subset V_1$ they can be represented in V_1, hence

$$\phi_0^0(t) = \sum_{l=-\infty}^{+\infty} h[l]\phi_l^1(t)$$

$$\psi_0^0(t) = \sum_{l=-\infty}^{+\infty} g[l]\psi_l^1(t)$$

$h[l],g[l]$ are projections of $\phi_0^0(t),\psi_0^0(t)$ onto V_1. Now, by multiplying $2^m/2$ to both sides and with the change of variable $t = 2^m t' - n$ we rewrite

$$\phi_n^m(t) = \sum_{l=-\infty}^{+\infty} h[l-2n]\phi_l^{m+1}(t)$$

$$\psi_n^m(t) = \sum_{l=-\infty}^{+\infty} g[l-2n]\psi_l^{m+1}(t)$$

and

$$h[l-2n] = \langle\phi_n^m(t),\phi_l^{m+1}(t)\rangle$$
$$g[l-2n] = \langle\psi_n^m(t),\psi_l^{m+1}(t)\rangle$$

At last the searched set of coefficients can be evaluated just through $h[n],g[n]$ from the finest to the coarsest time scales $\phi_n^{m+1}(t) \in \{V_m \oplus O_m\}$,

$$\begin{aligned}\phi_n^{m+1}(t) &= A_m\{\phi_n^{m+1}(t)\} + D_m\{\phi_n^{m+1}(t)\} \\ &= \sum_{l=-\infty}^{+\infty} \alpha_{l,n}\phi_l^m(t) + \sum_{l=-\infty}^{+\infty} \beta_{l,n}\psi_l^m(t)\end{aligned} \tag{6.44}$$

where

$$\alpha_{l,n} = \langle\phi_n^{m+1}(t),\phi_l^m(t)\rangle = h[n-2l]$$
$$\beta_{l,n} = \langle\phi_n^{m+1}(t),\psi_l^m(t)\rangle = g[n-2l]$$

thus

$$a_n^{m+1} = \sum_{l=-\infty}^{+\infty} h[n-2l]a_l^m + g[n-2l]x_l^m$$

$h[n]$,$g[n]$ are FIR filters (Finite Impulse Response) which are easy to implement in a very efficient way. $h[n]$ gives an approximation of $x(t)$ at the resolution m given that at $m+1$, i.e., $a_n^{m+1} \rightarrow a_n^m$. By contrast $g[n]$ gives the error of approximation at the resolution m known $x(t)$ at $m+1$, i.e., $a_n^{m+1} \rightarrow x_n^m$.

$x[n]$ with finite support

Let us consider a finite discrete series of samples $x[n]$ with $n = 0,\ldots,N$. The representation of $x(t)$ is then

$$x[n] = A_M\{x(t)\} = \{x(t) * \phi_n^M(-t)\}|_{t=2^{-M}n}$$

with $m = M_0,\ldots,K$ and $n = 0,\ldots,2^m - 1$, where M_0 and K are the minumum and maximum time scale in $x(t)$. The support is finite an equal to $[0,2^K - 1]$

$$x(t) = \sum_{n=0}^{2^{M_0}-1} a_n^{M_0}\phi_n^{M_0}(t) + \sum_{n=M_0}^{K}\sum_{n=0}^{2^m-1} x_n^m\psi_n^m(t)$$

A simple case: the Haar Wavelet

Given a scale function $\phi(t)$ it is always possible to find $\psi(t)$. It can be shown that ortogonality of O_m e V_m implies

$$g[n] = (-1)^n h[1-n]$$

we consider

$$\phi(t) = \begin{cases} 1 & 0 < t \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

that leads to

$$\psi(t) = \begin{cases} 1 & 0 < t \leq 1/2 \\ -1 & 1/2 < t \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

which defines the set of the *Haar Wavelets*. The two filters $h[n]$,$g[n]$ can be obtained easily for this set of wavelets.

$$h[n] = \int_{-\infty}^{+\infty} \phi_n^1(t)\phi_0^0(t)\,dt = \int_0^1 \phi_n^1(t)\,dt = \frac{\sqrt{2}}{2}(\delta[n] + \delta[n-1])$$

$$g[n] = \int_{-\infty}^{+\infty} \phi_n^1(t)\psi_0^0(t)\,dt = \int_0^{1/2} \phi_n^1(t)\,dt - \int_{1/2}^{1/4} \phi_n^1(t)\,dt = \frac{\sqrt{2}}{2}(\delta[n] - \delta[n-1])$$

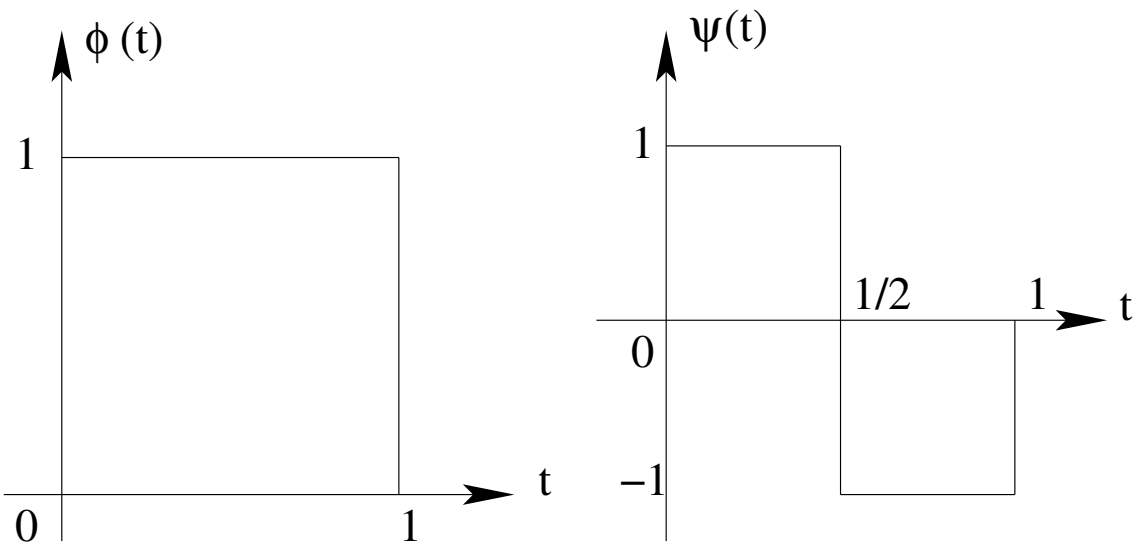


Figure 6.5. Haar Wavelets

and using 6.44 e 6.44 we write

$$a_n^m = \sum_{l=-\infty}^{+\infty} h[l-2n]a_l^{m+1} = \frac{\sqrt{2}}{2}(a_{2n}^{m+1} + a_{2n+1}^{m+1})$$

$$x_n^m = \sum_{l=-\infty}^{+\infty} g[l-2n]a_l^{m+1} = \frac{\sqrt{2}}{2}(a_{2n}^{m+1} - a_{2n+1}^{m+1})$$

solving the system of equation

$$a_{2n}^{m+1} = \frac{\sqrt{2}}{2}(a_n^m + x_n^m) \tag{6.45}$$

$$a_{2n+1}^{m+1} = \frac{\sqrt{2}}{2}(a_n^m - x_n^m) \tag{6.46}$$

which is a very fast algorithm to generate all scales and details coefficients.

6.3 LogScale Diagram

In this section we describe a tool that has become very popular in the computer networks community. This tool has been extensively studied by Abry and Veitch et. al in [VA98, VA01, AFTV00, RVA$^+$01] and it is wavelet based.

Scale invariance has been very shortly introduced in this notes without the use of wavelets and, indeed, this theory developed without the help of this representation technique that proved to be very useful in fields like image/video compression especially.

On the contrary it is a relatively recent finding that wavelts has very good properties as an estimator of scale invariance in time series. This novel approach to deal with scale invariance is more than a simple estimator but actually defines a comprehensive framework.

Let us see why wavelets can help to study stochastic processes with scale invariance. Let $X(t)$ a stochastic process and x_n^j the wavelet coefficients of $X(t)$.

$$x_n^j = \int_{-\infty}^{+\infty} X(t)\psi_n^j(t)\,dt$$

$$\phi_n^j(t) = 2^{-j/2}\phi(2^{-j}t - n)$$

Notice that in this case we use $j = -m$. While 2^m has the dimension of a time scale 2^j is a frequency, therefore j is called octave. It is implicit in the wavelet structure that if a stochastic process is scale invariant (H-ss,H-sssi or multifractal) x_n^j keeps some regular structure in j. This applies for any kind of scale invariant process, stationary or non-stationary.

An interesting result from [Fla92] states that for stationary gaussian scale invariant series

$$Cov(x_n^j, x_{n'}^j) = O(|n - n'|^{\alpha - 2N - 1})$$

where N is the number of *vanishing moments* of $\psi_n^j(t)$. The more regular the mother wavelet is the less the wavelets coefficient are correlated. This results explains why this approach works very well on sampled data.

We list a set of results that introduce a simple diagram that is often used in practice to infer information on a given sample.

- $\mathbf{E_n}[|\mathbf{x_n^j}|^\mathbf{2}] = \sigma^\mathbf{2}\mathbf{C2^{j\alpha}}\ \forall j \in \mathbf{Z} \Rightarrow X(t)$ is H-sssi
- $\mathbf{E_n}[|\mathbf{x_n^j}|^\mathbf{2}] \sim \mathbf{c_f C2^{j\alpha}}\ j \to +\infty \Rightarrow X(t)$ is Long Range Dependent
- $\mathbf{E_n}[|\mathbf{x_n^j}|^\mathbf{2}] \sim \sigma^\mathbf{2}\mathbf{C2^{j\alpha}}\ j \to -\infty \Rightarrow X(t)$ is Multifractal

Where c_f and α are related to power spectra of the process globally or locally defined depending whether the process is stationary or not, $\Gamma(f) = c_f \frac{1}{f^\alpha}$

The so called *Log-Scale Diagram* is a plot of $log_2\mathbf{E_n}[|\mathbf{x_n^j}|^\mathbf{2}]$ vs j that can be used to estimate scale invariant coefficients. For a finite sample we have $j \in [j_{min}, j_{max}]$ over which we seek a linear slope in the log-scale diagram as in the figure 6.6 for the fractional Brownian motion.

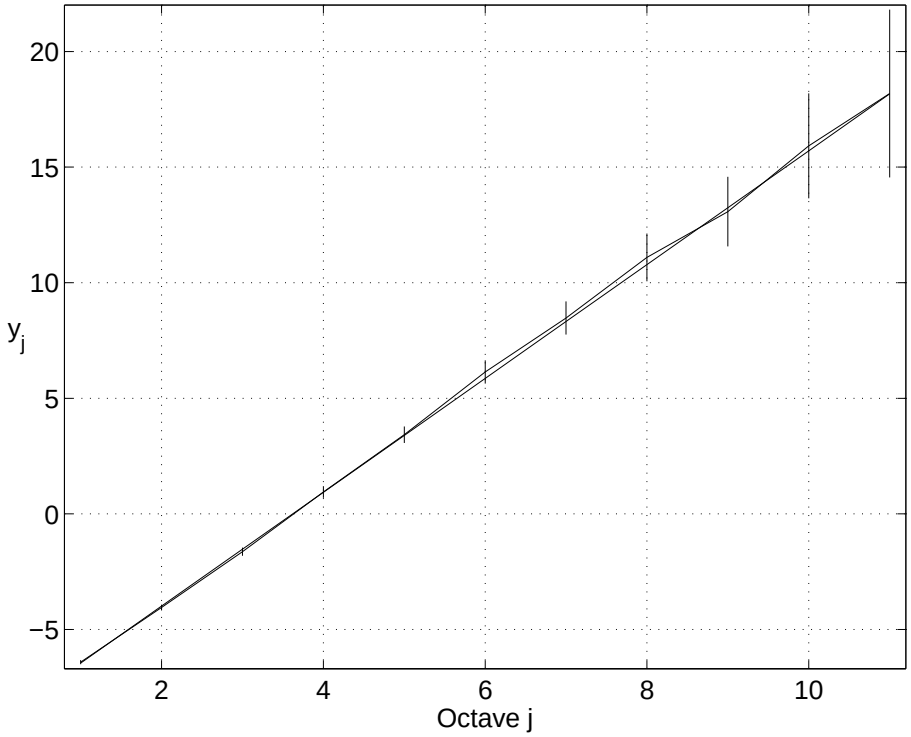


Figure 6.6. LogScale Diagram for fBm with H=0.8

Bibliography

[AAA+03] U. Ayesta, K.E. Avrachenkov, E. Altman, C. Barakat, and P. Dube. Multilevel approach for modeling short TCP sessions. In *Proc. of 18th International Teletraffic Congress*, pages 661–670, Berlin, Germany, August 2003.

[AFTV00] P. Abry, P. Flandrin, M.S. Taqqu, and D. Veitch. Self-similarity and long-range dependence through the wavelet lens. *Long Range Dependence: Theory and Applications, Doukhan, Oppenheim*, 2000.

[AKM04] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *ACM SIGCOMM*, Portland, Oregon, USA, 2004.

[BBC+94] R. Barret, M. Berry, T.F. Chan, J.Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R.Pozo, C. Romine, and H. Van der Vorst. Templates for the solution of Linear Systems: Building Blocks for Iterative Methods. *SIAM, 1994, Philadelphia, PA, USA*, 1994.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. In *RFC 2475, IETF*, Munich, 1998.

[BC02] N. Brownlee and KC Claffy. Understanding Internet Traffic Streams: Dragonflies and Tortoises. *IEEE Communications Magazine*, 40(10):110–117, October 2002.

[BCS94] B. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. *RFC 1633*, 44(69–100), 1994.

[BG92a] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, Englwood Cliffs, N.J., 1992.

[BG92b] D. Bertsekas and R. Gallager. *Data networks, 2nd edition*. Prentice-Hall, second edition, 1992.

[Blo70] B. Bloom. Space/time tradeoffs in hash coding with allowable errors. *Commununication of the ACM*, 13(7):422–426, 1970.

[BM95] D. Bini and B. Meini. n cyclic reduction applied to a class of Toeplitz-like matrices arising in queueing problems. *Computations with Markov Chains, W. J. Stewart, Ed. Kluwer Academic Publisher, Boston*, pages 21–38, 1995.

[BM01] T. Bonald and L. Massoulié. Impact of Fairness on Internet Performance.

In *ACM SIGMETRICS / IFIP Performance*, Cambridge, Massachusetts, USA, 2001.

[BMM+05a] A. Bianco, G. Mardente, M. Mellia, M. Munafò, and L. Muscariello. Exploiting clustering techniques for web user-session inference. In *3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurement IPS MOME*, Warsaw, Poland, 2005.

[BMM+05b] A. Bianco, G. Mardente, M. Mellia, M. Munafò, and L. Muscariello. Web user session characterization via clustering techniques. In *In proceedings of IEEE GLOBECOM 2005*, St. Louis, MO, USA, 2005.

[BP02] T. Bonald and A. Proutière. Insensitivity in Processor Sharing Networks. In *IFIP Performance*, Rome, Italy, 2002.

[BP03] T. Bonald and A. Proutière. Insensitive bandwidth sharing in data networks. *Queuing Systems*, 44:69–100, 2003.

[BPRJ01] T. Bonald, A. Proutière, G. Régnié, and J.W.Roberts. Insensitivity results in statistical bandwidth sharing. *ITC17, Volume 4, pp.125-136*, 2001.

[BPS99] J.C.R. Bennett, C.C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6):789–798, 1999.

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

[BT01] T. Bu and D. Towsley. Fixed point approximations for TCP behavior in an AQM network. In *Proc. of SIGMETRICS 2001*, pages 216–225, Cambridge, MA, USA, June 2001.

[BV05] T. Bonald and J. Virtamo. A recursive formula for multirate systems with elastic traffic. *IEEE Communications Letters*, 9(8):753–755, August 2005.

[CB97] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM transactions on networking*, 5(6):835–846, 1997.

[CDJM91] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel. `tcplib`: A library of tcp internetwork traffic characteristics. In *USC Technical report*, 1991.

[CIWA02] C.Nuzman, I.Saniee, W.Sweldens, and A.Weiss. A compound model for tcp connection arrivals, with applications to lan and wans. *Computer Networks, Special Issue on Long-Range Dependent Traffic*, 4(3):319–337, 2002.

[Cla88] D. Clark. The design philosophy of the darpa internet protocols. In *ACM SIGCOMM*, August 1988.

[Coh79] J.W. Cohen. The multitype phase service network with generalized processor sharing. *Acta Informatica*, 12:245–284, 1979.

[Dal02] P. Dalgaard. *Introductory Statistics with R*. Springer, 2002.

[DJ91] P. Danzig and S. Jamin. Characteristics of wide-area tcp/ip conversations. In *ACM SIGCOMM*, 1991.

[DJC+92] P. Danzig, S. Jaminand, R. Caceresand, D. Mitzeland, and D. Mestrin. An empirical workload model for driving wide-area tcp/ip network simulations. *IEEE/ACM Transactions on Networking*, 3(1):1–26, 1992.

[DJD05] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer sizing for congested Internet links. In *IEEE INFOCOM*, Miami, FL, USA, 2005.

[DKS90] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and experience (Also in proceedings of ACM Sigcomm 89)*, 1:3–26, 1990.

[DOT02] Paul Doukhan, George Oppenheim, and Murad S. Taqqu. *Theory and Applications of Long-Range Dependence*. 2002.

[ELM01] Paul Embrechts, Filip Lindskog, and Alexander McNeil. Modeling dependence with copulas and applications to risk management. *Handbook of Heavy Tailed Distributions in Finance, eds S.Racher, Elsevier*, pages 329–384, 2001.

[ENW96] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM transactions on networking*, 4(2):209–223, 1996.

[EV02] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *ACM SIGCOMM*, 2002.

[FBP+01] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J. Roberts. Statistical bandwidth sharing: A study of congestion at flow level. In *Proceedings of SIGCOMM*, San Diego, California, USA, August 2001.

[Fel68] Feller. *Introduction to probability theory and its applications: Vol I.* Wiley International, 1968.

[FGWK98] A. Feldmann, A. Gilbert, W. Willinger, and T. Kurtz. The changing nature of network traffic: Scaling phenomena. *Computer Communication Review*, 28(2), 1998.

[FH89] W. Fisher and K. Meier Hellstern. Structured Stochastic Matrices of M/G/1 Type And Their Applications. *Marcel Dekker, New York*, 1989.

[FH99] W. Fisher and K. Meier Hellstern. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18:149–171, 1999.

[FL03] F. Le Faucheur and W. Lai. Requirements for support of differentiated services-aware mpls traffic engineering. *RFC 3564, IETF*, 2003.

[Fla92] P. Flandrin. Wavelet analysis and synthesis of fractional brownian motion. *IEEE Transactions on Information Theory, 38(2), pp. 910-917*, March 1992.

[FLMT02] D. R. Figueiredo, B. Liu, V. Misra, and D. F. Towsley. On the autocorrelation structure of tcp traffic. *Computer Networks*, 40(3):339–361, 2002.

[Flo03] S. Floyd. HighSpeed TCP for Large Congestion Windows. In *IETF RFC 3649*, 2003.

[FMI80] G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many job classes. *Journal of the ACM*, 27:519–532, 1980.

[FML+03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, D. Moll M. Khan, R. Rockell, T. Seely, and C. Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 17(6):6–16, November-December 2003.

[FP01] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001.

[FRW04] J. Fermanian, D. Radulovic, and M. Wegkamp. Weak convergence of empirical copula processes. *Bernoulli*, 10(5):847–860, 2004.

[GKZ93] S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin. *Logistic of Production and Inventory*. Nemhauser and Rinnoy Kan (editors), Handbooks in Operation Research and Management Science Vol.4, North-Holland, 1993.

[GMMR04] P. Giaccone, M. Mellia, L. Muscariello, and D. Rossi. Switches under real internet traffic. In *IEEE Workshop on High Performance Switching and Routing*, Phoenix, Arizona, USA, 2004.

[GVC96] P. Goyal, H. Vin, and H. Cheng. Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. *IEEE/ACM Transactions on Networking*, 5(5):690–704, October 1996.

[GVC97] P. Goyal, H. Vin, and H. Cheng. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transactions on Networking*, 5(5):690–704, October 1997.

[Hah91] E.L. Hahne. Round-Robin Scheduling for Max-Min Fairness in Data Networks. *IEEE Journal on Selected Areas in Communications*, 9(7):1024–1039, September 1991.

[HCJP+05] F. Hernandez-Campos, K. Jeffay, C. Park, J. S. Marron, and S. I. Resnick. Extremal dependence: Internet traffic applications. *Stochastic Models, Taylor & Francis*, 22(1):1–35, 2005.

[HMS01] D.J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[HRT00] A. Horváth, G.I. Rózsa, and M. Telek. A map fitting method to approximate real traffic behaviour. In *8-th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks*, San Francisco, CA, USA, 2000.

[HVA02] N. Hohn, D. Veitch, and P. Abry. Does fractal scaling at the ip level depend on tcp flow arrival processes ? In *ACM Internet Measurement Workshop*, Marseille, France, 2002.

[IDG+01] G. Iannaccone, C. Diot, I. Graham, , and N. McKeown. Monitoring very high speed links. In *ACM Internet Measurement Workshop*, San Francisco, CA, USA, November 2001.

[JD05] H. Jiang and C. Dovrolis. Why is the internet traffic bursty in short (sub-*RTT*) time scales? In *ACM SIGMETRICS*, Banff, AL, Canada, 2005.

[JID+03] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. In *IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[JWL04] C. Jin, D. X. Wei, and S. H. Low. FAST TCP: Motivation, Architecture, Algorithms. In *IEEE INFOCOM*, Honk Hong, China, 2004.

[Kel03] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *Computer Communication Review*, 32(2), April 2003.

[KMOR04] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. On the Scalability of Fair Queueing. In *ACM SIGCOMM HotNets III*, San Diego, CA, USA, 2004.

[KMOR05a] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *SIGMETRICS*, Banff, AL, Canada, 2005.

[KMOR05b] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Minimizing the overhead of implementing flow-aware networking. In *Symposium on Architectures for Networking and Communications Systems ANCS*, Princeton, New Jersey, USA, October 26-28, 2005.

[KMT98] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[KNW98] U.R. Krieger, V. Naoumov, and D. Wagner. Analysis Of A Finite Buffer In An Advanced Packet-Switched Network. *IEICE Transaction On Communication, Special Issue on ATM Traffic Control and Performance Evaluation*, E00-B(4), 1998.

[KOR04] A. Kortebi, S. Oueslati, and J. Roberts. Cross-protect: Implicit Service Differentiation and Admission Control. In *IEEE HPSR*, Phoenix, AZ, USA, 2004.

[LBMK03] P. Lassila, H. Van Den Berg, M. Mandjes, and R. Kooij. An integrated packet/flow model for tcp performance analysis. In *Proceedings of 18th International Teletraffic Congress (ITC-18)*, pages 651–660, August 2003.

[LH03] Kun Chan Lan and John Heidemann. On the correlation of internet flow characteristics. Technical Report ISI-TR-574, USC/Information Sciences Institute, July 2003.

[LM97] D. Lin and R. Morris. Dynamics of Random Early Detection. In *ACM SIGCOMM*, Cannes, French Riviera, FRANCE, 1997.

[LR93] WG. Latouche and V. Ramaswami. A logarithmic Reduction Algorithm for quasi-birth-death processes. *Journal of Applied Probability*, 30:650–674, 1993.

[LTWW94] W.E. Leland, M.S. Taqqu, W.Willinger, and V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended version). *IEEE/ACM Transaction on Networking*, 2(1):1–15, January 1994.

[MCC02] M. Mellia, A. Carpani, and R.Lo Cigno. Measuring ip and tcp behavior on edge nodes. In *IEEE GLOBECOM*, pages 2533 – 2537 vol.3, Taipei, TW, 2002.

[MCN05] M. Mellia, R. Lo Cigno, and F. Neri. Measuring ip and tcp behavior on edge nodes with tstat. *Computer Networks*, 47(1):1–21, 2005.

[Mel05] Marco Mellia. `Tstat` home page `http://www.tlc-networks.polito.it/tstat`. 2005.

[MFW01] R. Mahajan, S. Floyd, and D. Weatherall. Controling High-Bandwidth Flows at the Congested Router. In *IEEE ICNP*, Riverside, CA, USA, 2001.

[MLJ05] S. McCanne, C. Leres, and V. Jacobson. Tcpdump. *http://www.tcpdump.org/*, 2005.

[MM05] R. Van De Meent and M. Mandjes. Evaluation of user-oriented and black-box traffic models for link provisioning. In *Proceedings of 1st EuroNGI Conference on Next Generation Internet Networks - Traffic Engineering*, Rome, Italy, April 2005.

[MMC05] M. Mellia, M. Meo, and C. Casetti. Tcp smart framing: a segmentation algorithm to reduce tcp latency. *IEEE/ACM Transactions on Networking*, 13(2):316–329, 2005.

[MMM+04a] L. Muscariello, M. Mellia, M. Meo, R. Lo Cigno, and M. Ajmone Marsan. Markov models of internet traffic and a new hierarchical mmpp model. *Computer Communications, Elsevier Science*, 10(5):847–860, 2004.

[MMM+04b] L. Muscariello, M. Mellia, M. Meo, R. Lo Cigno, and M. Ajmone Marsan. An mmpp-based hierarchical model of internet traffic. In *IEEE International Conference on Communications*, Paris, France, 2004.

[MMM05] M. Mellia, M. Meo, and L. Muscariello. Tcp anomalies: identification and analysis. In *Tyrrenian International Workshop on Digital Communication*, Sorrento, Italy, 2005.

[Nag85] J. Nagle. On Packet Switches with Infinite Storage. Technical report, RFC 970, IETF, 1985.

[NC89] S. Niu and R. Cooper. Duality and Other Results for M/G/1 and GI/M/1 Queues, via a New Ballot Theorem. *Mathematics of Operations Research*, 14(2):281–293, 1989.

[NLA05] NLANR. Abileneiii packet trace, `http://pma.nlanr.net/Special/ipls3.html`. 2005.

[OR05] S. Oueslati and J. Roberts. A new direction for quality of service: Flow aware networking. In *1st Conference on Next Generation Internet Design and EngineeringNGI 2005*, Rome, 2005.

[PA00] V. Paxson and M. Allman. Computing tcp's retransmission timer. *RFC 2988*, 2000.

[Pax94] V. Paxson. Empirically derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking*, 2(1):316–336, 1994.

[PBPS02] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Table-Based Design to Approximate Fairness. In *Hot Interconnects*, Palo Alto, CA, USA, 2002.

[PF95] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. *IEEE/ACM transactions on networking*, 3(3):226–244, 1995.

[RMM04] D. Rossi, L. Muscariello, and M. Mellia. On the properties of tcp flow arrival process. In *IEEE International Conference on Communications*, Paris, France, 2004.

[Rob04] J. Roberts. Internet traffic, qos and pricing. *Proceedings of the IEEE*, 92(9):1389 – 1399, 2004.

[Ros05] Dario Rossi. `diana_tools` home page `http://www.tlc-networks.polito.it/diana`. 2005.

[RVA+01] S. Roux, D. Veitch, P. Abry, L.Huang, P.Flandrin, and J.Micheel. Statistical scaling analysis of tcp/ip data. In *IEEE International Conference on Acoustics, Speech, and Sgnal Processing ICASSP*, 2001.

[SLSC99] B. Suter, T.V. Lakshman, D. Stiliadis, and A.K. Choudhury. Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queueing. *IEEE Journal on Selected Areas in Communications*, 17(6):1159–1169, June 1999.

[SRB01] S. Sarvotham, R. Riedi, and R. Baraniuk. Connection-Level Analysis and Modeling of Network Traffic. In *ACM Internet Measurement Workshop*, San Francisco, CA, USA, 2001.

[SSZ03] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: A scalable Architecture to Approximate Fair Bandwidth Allocations in High-Speed Networks. *IEEE/ACM Transactions on Networking*, 11(1):33–46, February 2003.

[SV96] M. Shreedhar and G. Varghese. Efficient Fair Queueing Using Deficit Round Robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.

[Tak62] L. Takàcs. Introduction to the Theory of Queues. *Oxford University Press, New York*, 1962.

[TG97] 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, 1997.

[top05] The GARR Network topology. `http://www.garr.it/mappagarr/garr-b-mappagarr.shtml`. 2005.

[tSW04] What You Should Know About the Sasser Worm. `http://www.microsoft.com/security/incident/sasser.mspx`. May

2004.

[VA98] D. Veitch and P. Abry. Wavelet analysis of long-range dependent traffic. *IEEE Transaction on Information Theory*, 44(1):2–15, 1998.

[VA01] D. Veitch and P. Abry. A statistical test for the time constancy of scaling exponents. *IEEE Transactions on Signal Processing*, 49(1):2325–2334, 2001.

[Vei] Darryl Veitch. Darryl veitch home page, wavelet estimation tools, `http://www.cubinlab.ee.mu.oz.au/\~darryl`on the auto-correlation structure of tcp traffic.

[Ven02] Daniele Veneziano. Eight lectures on scale invariance and engineering applications. support material for a lecture series by prof. d. veneziano. Technical report, MIT, 2002.

[Vir94] J. Virtamo. Idle and busy period distributions of an infinite capacity n*d/d/1 queue. In *Proceedings of ITC 14, Elsevier*, 1994.

[WTSW97] W. Willinger, M.S.and Taqqu, R. Sherman, and D. V. Wilson. Self-Similarity Through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. *IEEE/ACM Transaction on Networking*, 5(1):71–86, January 1997.

[WVM98] W.Willinger, V.Paxson, and M.S.Taqqu. Self-similarity and heavy tails: Structural modeling of network traffic. *In A Practical Guide to Heavy Tails: Statistical Techniques and Applications. R.Adler, R.Feldman, M.S.Taqqu editors, Birkhauser*, 1998.

[ZBPS02] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *ACM SIGCOMM*, Pittsburgh, PA, USA, 2002.