INGRESS TRAFFIC CONTROL IN DIFFERENTIATED SERVICES IP NETWORKS

By

Giovanni Neglia

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
UNIVERSITÀ DI PALERMO
VIALE DELLE SCIENZE, PALERMO
DECEMBER 2004

© Copyright by Giovanni Neglia, 2004

UNIVERSITÀ DI PALERMO
DEPARTMENT OF
INGEGNERIA ELETTRICA - DIE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Ingress Traffic Control in Differentiated Services IP Networks** " by **Giovanni Neglia** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dated: December 2004

External Examiner: _____

Research Supervisor: _____
Giuseppe Bianchi

Examing Committee: _____

_____

UNIVERSITÀ DI PALERMO

Date: **December 2004**

Author: **Giovanni Neglia**

Title: **Ingress Traffic Control in Differentiated Services IP Networks**

Department: **Ingegneria Elettrica - DIE**

Degree: **Ph.D.** Convocation: **February** Year: **2005**

Permission is herewith granted to Università di Palermo to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.

THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.

Table of Contents

List of Symbols

$\gamma(t, s)$	autocovariance function
$\rho(t, s)$	autocorrelation function
$\Gamma(\nu)$	spectral density function
H	Hurst parameter

Acronyms

AF	Assured Forwarding
APM	Adaptive Packet Marking, our algorithm
ARM	Active Rate Management
ATM	Asynchronous Transfer Mode
BB	Bandwidth Broker
BGP	Border Gateway Protocol
CAC	Call Admission Control
CE	Congestion Experienced
CHOKe	CHOse and Keep for responsive flows, CHOose and Kill for unresponsive flows
DCCS	Direct Congestion Control Scheme
DiffServ	Differentiated Services
DS	Differentiated Services
DSCP	Differentiated Services Code Point
EAC	Endpoint Admission Control
EBM	Equation-Based Marking
ECN	Explicit Congestion Notification
EF	Expedited Forwarding
E2E	End-to-End
FBM	Fractional Brownian Motion
FGN	Fractional Gaussian Noise

FIFO	First In First Out
FPA	Fixed Point Approximations
FRED	Flow RED
H-ss	Self Similar process with Hurst parameter H
H-sssi	Self Similar process with Stationary Increments and Hurst parameter H
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IntServ	Integrated Services
IVT	Intermediate Value Theorem
LRD	Long Range Dependence (or Dependent)
MBAC	Measurement Based Admission Control
MFT	Mean Field Theory
MIMO	Multiple Input Multiple Output
MLE	Maximum Likelihood Estimator
MPLS	Multi-Protocol Label Switching
MSS	Maximum Segment Size
NS	Network Simulator
PBAC	Parameter Based Admission Control
PDB	Per-Domain Behavior
PHB	Per-Hop Behavior
PME	Packet Marking Engine
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RED	Random Early Detection
REM	Rate-Envelope Multiplexing, or Random Exponential Marking
RSM	Rate Sharing Multiplexing
RTT	Round Trip Time

SLA	Service Level Agreement
SLS	Service Level Specifications
SN	Sequence Number
SRED	Stabilized RED
TCA	Traffic Conditioning Agreement
TCP	Transmission Control Protocol
TOS	Type Of Service
WDM	Wave Division Multiplexing
WPM	Web Packet Marking, algorithm proposed in [110]

Abstract

In this thesis we address issues concerned with three particular traffic control mechanisms employed in Differentiated Services IP networks: call admission control, active queue management and marking. In our study these mechanisms share the purpose to control traffic which is going to enter into the network, rather than traffic which is already in the network. It appears evident if we consider call admission control because it immediately changes the profile of ingress traffic by admissions ad rejections. At the same time active queue management and marking are considered as a way to drive TCP adaptation.

As regards call admission control, our study underlines that, when long range dependence takes place, measurement-based admission control exhibits good traffic forecasting properties which motivate better performance in comparison to traditional mechanisms based on a–priori knowledge. Besides, it appears to be able to change the stochastic properties of traffic aggregate, up to remove long range dependence.

As regards active queue management, we stress the spatially distributed feature of its control action, that is TCP flows may turn to be controlled at the same time by two or more nodes acting independently according to their specific settings. Configuration rules have been proposed in order to assure system stability from a modeling point of view, when a single node controls the traffic aggregate. Yet, we show that instability and poor network performance may arise when two nodes act at the same time.

Finally we propose a new marking algorithm, which explicitly takes into account the interaction among the TCP adaptation rule, the active queue management discipline, and the marking strategy. Extensive performance evaluation shows that the algorithm is able to assure significant improvements. We have developed an analytical fixed-point model exhibiting a novel approach for the queue behavior.

Acknowledgements

I would like to thank Prof. Giuseppe Bianchi, my supervisor, for his many suggestions and constant support during this research.

The results in this thesis have been obtained in collaboration with many other people: Vincenzo Mancuso, Vito Imburgia and Giuseppe Mazzola as regards call admission control; Prof. Laura Giarré, Dario Bauso and Daniele Di Bernardo as regards active queue management; Dario Lombardo, Marilena Sottile, Vincenzo Falletta as regards marking. My special thanks to all of them.

Finally I am grateful to all my friends and of course to my family: my parents and my sisters Nicoletta and Valeria.

Palermo, Italy — Giovanni Neglia
December 14, 2004

Chapter 1

Traffic Control

In this thesis the expression *traffic control* is employed in its broadest meaning, as the set of actions taken by the network which affect the traffic in order to achieve a given performance. These actions are detailed in what follows, anyway we can note that we are leaving out the source adaptation mechanisms, while we are including congestion control strategies inside the network.

1.1 About Terminology

In the ATM framework traffic control has a similar comprehensive meaning. For example, according to ITU-T Recommendation I.371 [1], "ATM layer traffic control refers to the set of actions taken by the network to avoid congested conditions", while "ATM layer congestion control refers to the set of actions taken by the network to minimize the intensity, spread and duration of congestion". The latter is defined as a state of network elements, in which the network is not able to meet the negotiated Quality of Service (QoS) objectives for the connections already established or for any new connection request, because of traffic overload or control-resource overload. According to these definitions of traffic control and congestion control, control mechanisms are often distinguished into two categories: preventive and reactive controls. In particular the following traffic control mechanisms are listed in [1]: resource provisioning, virtual path management, Connection Admission Control (CAC), fast resource management, call routing and load balancing, usage parameter control, priority control, traffic shaping. Explicit (forward or backward) congestion indication, selective cell discard, and reaction to usage or network parameter

control are indicated as congestion control techniques. Many of these mechanisms can be deployed also in the Internet and are described in Sec. 1.3. The distinction between traffic and congestion control can often be unclear: the same mechanism can be used in a preventive way or in a reactive way. For example an admission control mechanism discarding new incoming connections can be seen as a preventive mechanism if deterministic multiplexing is employed or a reactive one when statistical multiplexing occurs. This fuzziness appears also in [1]: for example selective cell discard (a congestion control mechanism) requires priority control (a traffic control mechanism). For this reason we decided to use the term "traffic control" to indicate both preventive and reactive control actions. More recently (see for example [43]) the term *traffic management* has been used with this meaning in the ATM framework.

The IP network framework is even more chaotic as regards terminology. ITU-T confirms the distinction between traffic control and congestion control [165]. In many contexts, "traffic control" is employed according to the wide meaning we adopted, see for example the Bonaventure's overview of traffic control and Quality of Service (QoS) [29], the topics of the conference "Scalability and Traffic Control in IP Networks" ([148, 149]), the Italian TANGO Project [155], and [85], [140]. In particular in TANGO project, under the traffic control umbrella there are CAC, Active Queue Management (AQM), dynamic routing, traffic conditioning (i.e. metering, marking, shaping, policing), Multi-Protocol Label Switching (MPLS) and even end-to-end (E2E) mechanisms (e.g. rate-adaptive protocols, like TCP). Conversely "traffic control" is employed with a more specific meaning in other significant contexts. In the Integrated Services (IntServ) framework it denotes the router functions that create different qualities of service [31]. It is implemented by three components: the packet scheduler, the classifier, and admission control. Traffic policing is considered one of the function of the scheduler. As regards Differentiated Services (DiffServ), the expression "traffic control" usually does not appear in related RFCs, but traffic conditioning is defined as the set of control functions performed to enforce rules specified in a Traffic Conditioning Agreement [27], including metering, marking, shaping, policing. Yet in [23] the authors use the terms "aggregate traffic control" and "DiffServ" interchangeably. Despite this lack of definition in the RFCs, an identification between traffic conditioning and traffic control is quite common (as in AQUILA project [7]). We think it is

due to the decision to name "Traffic Control Module" the software performing DiffServ traffic conditioning in Linux operating system [102]. Besides, this module performs functions similar (with the exception of CAC) to the "Traffic Control Module" of IntServ-enabled routers [32].

Finally traffic control techniques are often presented in the framework of *traffic engineering*. Traffic engineering is not limited to traffic control, but it encompasses the application of technology and scientific principles also to the measurement, characterization, modeling of Internet traffic. Moreover its main issue is performance evaluation and performance optimization of *operational* IP networks. As regards traffic engineering definition and overview see [12, 11, 10].

1.2 Traffic Control in Circuit-Switched Networks

Before describing traffic control techniques in a Differentiated Services scenario, we briefly present traffic control in circuit-switched networks, with reference to the Public Switched Telephone Network.

In circuit-switched networks, each connection is allocated a fixed amount of bandwidth, and a constant data rate in the network is provided to communicating entities throughout the duration of the connection. The call admission control is the criteria which is employed in order to admit new calls. In PSTN it is usually a very simple one: if a requested channel is available, the connection is established, otherwise it is rejected. This simple CAC is sufficient to control traffic in circuit-switched networks since the dedicated bandwidth is always available for a connection and there is no contention for network resources once a channel is allocated. Being the resource requirements constant during the call, the traffic control issue mainly comprises three aspects:

Provisioning or planning determines network resources at long time scales, i.e. adequate engineering of component elements (like circuits and exchanges) in terms of capacity, number and configuration.

Resource management allocates and configures network resources at middle or short time scales. Being PSTN dimensioned in order to satisfy

peak hour traffic requirements, resource reallocation or reconfiguration is usually required only when faults occur.

Call routing decides the path the call will follow. The early telephone network relied on static hierarchical routing. Upon the advent of digital switches and stored program control which were able to manage more complicated traffic engineering rules, dynamic routing was introduced to alleviate the routing inflexibility in the static hierarchical routing so that the network would operate more efficiently. This resulted in significant economic gains [84]. Dynamic routing typically reduces the overall loss probability by 10 to 20 percent (compared to static hierarchical routing) and can also improve network resilience by recalculating routes on a per-call basis and periodically updating routes. Three approaches to dynamic routing have been adopted in current networks: 1) in time-dependent routing, regular variations in traffic loads (such as time of day or day of week) are exploited in pre-planned routing tables; 2) in state-dependent routing, routing tables are updated online according to the current state of the network (e.g., traffic demand, utilization, etc.); 3) in event dependent routing, routing changes are incepted by events (such as call setups encountering congested or blocked links) whereupon new paths are searched out using learning models. A detailed description of the various routing strategies applied in telephone networks is included in [8].

These aspects require information about the status of the network. Digital exchange technology currently makes it possible to gather information on network status internally (meters) and makes it available to external systems. Meters reflect the call attempts number, completed calls, rejected calls, etc. From these basic meters it is possible to determine the degree of congestion, the level of utilisation, the percentage of free resources, etc.. There are global exchange meters and specific meters for every object in the exchange: route (connection between two exchanges), destination (numbering) and internal organ (internal component). Runtime measurements can be sent periodically or on threshold triggering, and are employed to reveal bad operation. Network planning is clearly based on information collected from the exchange over the day and has to be stored in order to be able to analyse daily, weekly, monthly

traffic and so forth. It is interesting to note that all the traffic controls (also the actions aiming to recover from faults) usually require human attendance (see for example Telefónica's traffic management system [35]).

1.3 Traffic Control in Differentiated Services Networks

In this section we present an overview of the large range of traffic control techniques in IP networks, where the Differentiated Services (DiffServ) approach is adopted. Firstly we illustrate main differences in comparison to PSTN that justify such extension of traffic control mechanisms, secondly we exemplify them according to a time-scale taxonomy. Despite the reference to DiffServ, the most part of the following considerations holds for a generic packet-switched or cell-switched network. The DiffServ framework is described in Appendix B, some traffic control mechanisms have been object of our research activity and are more deeply described in the following chapters.

Clearly the first discriminating element is the packetization of information to be delivered. It appears quite obvious but it allows network elements to operate on shorter time scale in comparison to call (or session) duration. At the same time packetization offers the possibility of statistical multiplexing gain. That is, taking advantage of variable sources data-rate, network elements can be dimensioned on average demand rather than on peak demand, allowing considerable resource sparing. Conversely when statistical multiplexing is used there is a non-null probability that at some point in time the offered load to a multiplex point will exceed its capacity. Hence packet level congestion can occur and it is faced by apposite traffic control mechanisms.

A second point is the large heterogeneity of application and user requirements. While PSTN is mainly intended for the transport of real-time content, the Internet is (or would become) a generic purpose network: file transfer as well as remote control or multimedia real-time communication should be supported. Many applications are *elastic*, i.e. they can work under a variety of network conditions and still perform correctly (for example File Transfer Protocol, emails transfer), others have QoS requirements (are inelastic). Besides such QoS requirements are usually different, not only quantitatively, but

also as regards the key performance issue (or combination of issues): throughput, delay, jitter, loss percentage, availability and security. Among inelastic applications some are *tolerant*, i.e. they can run in given range of QoS. For example Video on Demand can tolerate losses and a certain amount of delay, interactive real-time audio transmission is loss-tolerant to some extent (human brain can recover sporadic losses), but has stricter delay requirements, online trading is not loss tolerant. Tolerant applications are often *adaptive*, that is they try to maintain the perceived quality at an acceptable level, even under poor network conditions. This can be done by lowering the packet sending rate (VAT) or by lowering the resolution of the transmission (e.g., in video transfer) or even by using specialized compression and error-correction techniques. Adaptive applications may use extra buffering to compensate for network transients and allow for graceful degradation in performance. Most audio and video streaming applications on the Internet are adaptive. Tolerant and nonadaptive applications do not have the ability to cope with network transients: they can still tolerate some QoS degradation, but this directly affects the quality perceived by an end user. It has to be noted that Internet users are no more an undistinguished mass, and the same service can be provided with different characteristics of quality or availability depending on the user specific requirements. At the same time this heterogeneity of user desiderata corresponds to different pricing schemes. Further information on QoS needs of current Internet applications can be found in [52, 112].

Currently most of the traffic involves the transfer of static (or slowly changing) information on an adaptive reliable transport protocol (TCP). This has two main consequences: network traffic control mechanisms can be usefully aware of TCP adaptivity mechanisms, and the forecast of user requests can be advantageous.

In what follows we present the main traffic control techniques according to the time-scales over which they operate. This classification is quite common in the ATM framework [1, 72, 91] and it has also been proposed for Internet traffic control [29, 85, 167]. More in details we define the *operation time-scale* as the time interval between the begin of the control and the manifestation of its effects in the network. Although we have not found this strict definition in scientific literature, we remark that two different time intervals constitute the operation time-scale: the time needed to perform the control action (the *action*

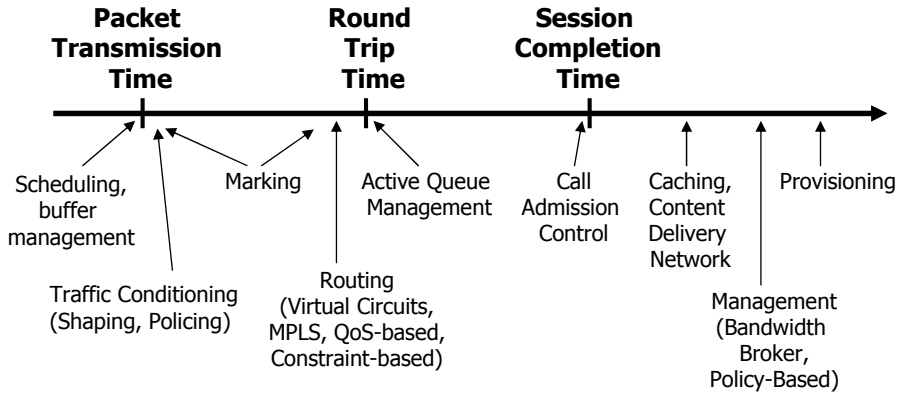

Figure 1.1: Traffic Control Techniques and their Time-Scales.

time), and the time needed to note its effects in the network (the *reaction time*). Often the classifications implicitly consider one of these time intervals as reference, sometimes they do not distinguish, but it can lead to ambiguity because the magnitude of these two interval times is sometimes comparable, but often one prevails. Our classification is shown in Fig. 1.1, we tried to make it as complete as possible. Clearly in some cases it may look questionable.

1.3.1 Provisioning

Provisioning determines the resources of the network. The goal of provisioning is to ensure that the network has enough resources to meet the expected demand with adequate QoS. Users expect some accessibility to the service, usually high, and providers need to plan the network to meet these expectations. With network provisioning, the challenge is to ensure that sufficient resources are available to accept all potential connections, while still maintaining a cost-effective network design. This leads to a tradeoff between the quantities of resources that should be placed in the network versus the expected utilization that they can achieve. The amount of resources depends on the degree of statistical multiplexing which is considered acceptable. Here pricing is fundamental to incentive the provider to expand capacity as demand grows in order to ensure that the performance is maintained. Certain aspects of provisioning, such as capacity planning, respond at quite large timescales, ranging from days to possibly years. The introduction of automatically switched optical transport networks (e.g., based on the Multi-protocol Lambda Switching concepts

[13]) could significantly reduce the lifecycle for capacity planning by expediting provisioning of optical bandwidth. For a recent overview of IP network planning refer to [144].

1.3.2 Management

While provisioning determines the resources, management allocates and configures them at middle or short time scales. Because of increasing complexity, management costs have become significant in current data networks. ISPs have been asking to vendors for management tools enabling the following features: centralized management, with the ability to perform tasks via the network; abstracted (or simplified) management data, which fits with the fewer interfaces objective by abstracting the functions and decisions criteria across multiple devices; automation of management tasks in order to support management information re-use, and to allow the network to operate with a minimum of human intervention; commonality across multi-vendor devices; fewer interfaces; consistency across interfaces.

One step towards this direction is the Bandwidth Broker employment, which is responsible of negotiating Service Level Specifications and configure edge routers to enforce resource allocation and admission control. More details on Bandwidth Broker are in Sec. B.7.

A more general solution is the Policy Based Management framework, which provides a way to allocate network resources, primarily network bandwidth, QoS, and security (firewalls), according to defined business policies. A policy-based management system allows administrators to define rules to address these issues and manage them in the policy system. These rules take the form "If condition, then action." A condition may be a user or a group, the time of day, the application type, or the network address. Policy rules are then distributed to network resources. Resources include devices that manage network bandwidth, security, IP addresses, storage, processors, and agents, as well as systems that manage services such as billing, accounting, and service mapping. Locator services are also required to help resource managers find one another. The IETF Policy Working Group has developed a policy management architecture that is considered the best approach for policy management on the Internet. It includes the following components:

Policy management service A graphical user interface for specifying, editing, and administering policy.

Policy repository A place to store and retrieve policy information, such as a Lightweight Directory Access Protocol (LDAP) server or a Directory Enabled Network (DEN) device.

Policy Decision Point (PDP) A resource manager or policy server that is responsible for handling events and making decisions based on those events (i.e., at time x do y), and updating the PEP configuration appropriately.

Policy Enforcement Point (PEP) PEP exists in network nodes such as routers, firewalls, and hosts. It enforces the policies based on the "if condition then action" rule sets it has received from the PDP.

Local Policy Decision Point (LPDP) This is a scaled-down PDP that exists within a network node and is used in cases when a policy server is not available. Basic policy decisions can be programmed into this component.

A variety of protocols may be used to communicate policy information between the PDP and the PEP. COPS (Common Open Policy Service) is the usual protocol, although DIAMETER or even SNMP may be used. COPS is a client/server protocol that provides transport services for moving policy information among IP network nodes. It also provides the transport for policy queries and responses. By moving policy information to different subnets, users can log on at other locations and receive the same service they receive from their home network. COPS was developed by the IETF RSVP Admission Policy (RAP) Working Group, which is developing a scalable policy control model for RSVP.

1.3.3 Caching and Content Delivery Networks

As we said above, most of the traffic involves the transfer of static (or slowly changing) information, like the Web traffic for example. Hence performance can be improved if user requests are predicted and contents are conveyed closer to the user. An example is the employment of a proxy server in the user

network. A proxy server sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server. Proxy servers can dramatically improve performance for groups of users. This is because the proxy server stores the results of all requests (let us consider web pages) for a certain amount of time. Hence if an user requests a page which has already been requested by another user formerly, the proxy server simply returns the page that it already fetched, instead of forwarding the request to the Web server where the page resides, which can be a time-consuming operation. Since the proxy server is often on the same network as the user, this is a much faster operation.

Content Delivery Networks (sometimes called content distribution networks) are an extension of this simpler approach: a content from a site is copied to geographically dispersed servers and, when a page is requested, faster delivery is achieved by identifying and serving it from the closest server to the user. Typically, high-traffic Web site owners and Internet service providers (ISPs) hire the services of the company that provides content delivery (e.g. Accelia [3], Akamai [4], Mirror Image [113]). A common content delivery approach involves the placement of cache servers at major Internet access points around the world and the use of a special routing code that redirects a Web page request to the closest server. When the Web user clicks on a URL that is content-delivery enabled, the content delivery network re-routes that user's request away from the site's originating server to a cache server closer to the user. The cache server determines what content in the request exists in the cache, serves that content, and retrieves any non-cached content from the originating server. Any new content is also cached locally. Other than faster loading times, the process is generally transparent to the user, except that the URL served may be different than the one requested. The three main techniques for content delivery are: HTTP redirection, Internet Protocol (IP) redirection, and domain name system (DNS) redirection. In general, DNS redirection is the most effective technique. Content delivery can also be used for specific high-traffic events such as live Web broadcasts by continually dispersing content from the originating server to other servers via satellite links.

1.3.4 Routing

The Internet deploys dynamic routing algorithms with distributed control to determine the paths that packets should take en-route to their destinations. Interior Gateway Protocol (IGP) such as RIP or OSPF are used to exchange routing information within an autonomous system[1] , while the Border Gateway Protocol (BGP) is an inter-autonomous system routing protocol. Here we focus on intra-autonomous system routing. The routing algorithms are adaptations of shortest path algorithms where costs are based on link metrics. The link metric can be based on static quantities, for example assigned administratively according to local criteria, or on dynamic quantities, depending for example by the network congestion measure such as delay or packet loss. Static link metric assignment, according to local criteria, has shown to be inadequate because it can easily lead to unfavorable scenarios in which some links become congested while others remain lightly loaded. Even if link metrics are assigned in accordance with the traffic matrix, forecasting errors or simply traffic dynamics can lead to unbalanced loads in the network. Also, the routing protocols does not take traffic attributes and capacity constraints into account when making routing decisions. This results in traffic concentration being localized in subsets of the network infrastructure and potentially causing congestion. One adopted solution is adjusting IGP and/or BGP parameters to route traffic away or towards certain segments of the network, but it tends to have network-wide effect. Consequently, undesirable and unanticipated traffic shifts can be triggered as a result.

Another solution is available when a virtual-circuit network, such as ATM, frame relay, or WDM, provides virtual-circuit connectivity between routers that are located at the edges of a virtual-circuit cloud. In this mode, two routers that are connected through a virtual circuit see a direct adjacency between themselves independent of the physical route taken by the virtual circuit through the ATM, frame relay, or WDM network. Being the router logical topology decoupled from the physical topology, it can be re–configured to make it correlate more closely with the spatial traffic distribution using the underlying path-oriented technology. This approach is called the *overlay model* [10].

[1] An autonomous system is a network or group of networks under a common administration and with common routing policies.

According to our presentation it requires the management of two separate networks with different technologies resulting in increased operational complexity and cost. The IP over ATM technique is no longer viewed favorably due to recent advances in MPLS (which allows virtual circuit employment) and router hardware technology. Both these adaptive solutions employ a measurement system that monitors changes in traffic distribution, traffic shifts, and network resource utilization and subsequently provides feedback to the online and/or offline traffic engineering mechanisms and tools which employ this feedback information to trigger certain control actions to occur within the network.

The traffic engineering mechanisms and tools can be implemented in a distributed fashion or in a centralized fashion, and may have a hierarchical structure or a flat structure. The above solutions rely on the legacy Internet interior gateway routing system, currently there is there is strong interest in path oriented technology with explicit routing and constraint-based routing capability such as MPLS. Constraint-based routing refers to a class of routing systems that compute routes through a network subject to the satisfaction of a set of constraints and requirements. It is a generalization of QoS routing [45], which selects paths to be used by a flow based on the QoS requirements of the flow itself. In the most general setting, constraint-based routing may also seek to optimize overall network performance while minimizing costs. The constraints and requirements may be imposed by the network itself or by administrative policies. Constraints may include bandwidth, hop count, delay, and policy instruments such as resource class attributes. Constraints may also include domain specific attributes of certain network technologies and contexts which impose restrictions on the solution space of the routing function. Path oriented technologies such as MPLS have made constraint-based routing feasible and attractive in public IP networks. The concept of constraint-based routing within the context of MPLS traffic engineering requirements in IP networks was first defined in [12].

1.3.5 Admission Control

As we said in Sec. 1.2, admission control evaluates if the network can provide to the flow the requested service while maintaining the service promised to

the other flows. The admission control needs to determine (or estimate) the available resources and the requested resources. The service is provided if available resource are not smaller than requested resources. Available and requested resources can be determined by explicitly declared traffic parameters (e.g. by conformance to a token bucket) or through measurements, or both. The admission control method could be centralized or distributed. We can observe that the time-scale of the admission process is related to the call arrival rate, while the time-scale of congestion recovery (i.e. when the admission control estimates that no new call can be admitted) is related to the call duration time. A more detailed overview of CAC is in Sec. 2.1.

1.3.6 Active Queue Management

The most part of current Internet traffic employs TCP as transport protocol. TCP is responsive to congestion: it assumes that packet discard is due to congestion, hence it uses the receipt of the three duplicate acknowledgements or the expiration of a retransmit timer as indication of congestion and consequently reduces its transmission rate. The traditional technique for managing router queue lengths is to accept packets for the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until there is space in the queue because a packet has been transmitted (this technique is known as *tail drop*. The basic idea of Active Queue Management (AQM) is to take into account TCP responsiveness, and drop packets before a queue becomes full, so that end nodes can respond to congestion before buffers overflow. Hence AQM appears to be a proactive approach to queue management. By preventive dropping and introducing some randomness AQM mechanism are able to overcome droptail overcomes like *lock-Out* and *full queues*. A more detailed overview of AQM is in Sec. 3.1.

1.3.7 Traffic Conditioning

Traffic conditioning, which enforces that the traffic entering the network of the flow follows the rules agreed for the service. In fact, there is a traffic profile description that allows to classify each packet as in-profile or out-profile packet. In-profile packets receive some QoS while out-profile receive some other (lower), this depending on the services rules. Therefore, traffic conditioning

involves some traffic metering and comparison to a profile, and some actions. These actions could be the following: delaying an out–profile packet until becoming in–profile (also called shaping), discarding an out–profile packet (also called policing), and changing its DiffServ codepoint (also called re–marking). A more detailed overview of traffic conditioning is in Sec. B.4.

1.3.8 Queue Discipline

DiffServ Per-Hop-Behaviors (PHB, see Sec. B.5) are implemented in nodes by means of some buffer management and packet scheduling mechanisms.

Buffer managements algorithms, which decide when a packet is discarded, have been already illustrated as AQM mechanisms, whose purpose is to preventively notify incipient congestion, and as policing mechanism in the framework of traffic conditioning. On shorter time-scales, buffer management is a way to cope with current congestion at the node. Drop tail, *random drop on full* (a randomly selected packet is dropped) and *drop front on full* (the packet at the front of the queue is dropped) are typical buffer management mechanisms.

Scheduling algorithms decide which packet has to be sent among those contending for the same transmission link. The packets can be in the same buffer or in different buffers. A main distinction of schedulers is between work-conserving algorithms and non-work-conserving algorithms. The former schedulers are never idle, when there are packets waiting to be sent, that is, a packet is always sent unless there are no packets in the buffers. The latter schedulers may be idle, even when there are packets waiting to be sent, because they wait for packets to become eligible for transmission. While scheduling is a long-time studied topic, the advantages of non-working-conserving disciplines have not been realized at the begin. The reason is twofold. Firstly, in most of performance analysis the major performance indices were the average delay of all packets and the average throughput of the server. The average delay is the same for all the working-conserving discipline, and non-working-conserving disciplines cannot achieve lower delay. Secondly, queuing analysis assumed often a single serve environment, and the potential advantages of non-working-conserving disciplines arise when a complex networking environment is considered. In guaranteed performance service, delay bounds are more

important performance indexes than average delay. In order to derive end-to-end delay bounds traffic needs to be characterized inside the network on a per connection (or per aggregate) basis. With work-conserving disciplines, even if traffic of a connection can be characterized at the entrance to the network, traffic pattern may be distorted inside the network, thus making the source characterization not applicable at the servers traversed by the connection. Some examples of working-conserving schedulers are First-In-First-Out, Priority, Round Robin, Virtual Clock, Weighted Fair Queuing, Worst-case Fair Weighted Fair Queueing, Self-Clocked Fair Queueing and Delay Earliest-Due-Date. Some examples of non-work-conserving schedulers are Jitter Earliest-Due-Date, Stop-and-Go, Hierarchical Round Robin, Rate Controlled Static Priority, Jitter-Virtual Clock and Core-Jitter-Virtual Clock. Further information on scheduling disciplines can be found in [167] and its bibliography.

1.4 Contribution and Organization of the Thesis

In this thesis we address three particular traffic control mechanisms: Call Admission Control (CAC), Active Queue Management (AQM) and marking. They operate at middle time-scale ranging from the Round Trip Time (RTT) to the call (or session) duration. In the title of this thesis we refer to *ingress* traffic control for two reasons. Firstly CAC and marking are deployed at the ingress routers of a DiffServ domain. Secondly the specific traffic controls we are considering have not the purpose to control the traffic already in the network, but to affect the traffic which is going to enter into the network. This appears evident if we consider Call Admission Control because it immediately changes the profile of ingress traffic by admissions ad rejections. At the same time active queue management does not aim to give immediate relief to the network, but to signal incipient congestion to TCP sources. In this way AQM action reduces the amount of traffic which will be offered to the network after a RTT. Finally our marking algorithm is also a way to drive TCP adaptation according to the network status, hence the same considerations hold.

The thesis is organized as follows. In Chapter 2 we focus on CAC and compare Parameter Based Admission Control (PBAC) and Measurement Based Admission Control (MBAC) in the presence of Short Range Dependent and

Long Range Dependent (LRD) traffic. In Chapter 3 we study a control-theoretic approach to AQM and show that network instability can arise in a multiple bottleneck scenario, even when configuration rules assure local stability at each node. In Chapter 4 we propose a new adaptive marking algorithm together with its performance evaluation and analytical model. The three chapters are almost independent, but the reader can find useful the AQM overview in Chapter 3 before tackling Chapter 4. Three appendixes completes the thesis: Appendix A provides analytical background on LRD and Self-Similarity, Appendix B briefly describes the DiffServ approach and architecture, while Appendix C explains results on the effect of reordering shown in Chapter 4.

In our opinion, the main contributions of this thesis are the following:

- The study on CAC confirms that MBAC mechanisms can achieve better performance in comparison to the PBAC when traffic LRD is considered.
- Besides, to the best of our knowledge, it is the first study showing that MBAC algorithms are able to change the traffic correlation structure and to filter away Long Range Dependence.
- The study on AQM highlights the distributed nature of TCP control by such mechanisms, and the limits of configuration rules ignoring this aspect.
- A new adaptive marking algorithm has been proposed. Extensive performance evaluation shows the benefits of its adoption. An analytical model provides deeper understanding and can be used for configuration purpose.
- A novel approach to model the queue behavior has been adopted and can be used in different contexts.

Chapter 2

The effect of Measurement-Based Admission Control on Traffic Long Range Dependence

In this chapter we compare two approaches to admission control: Parameter-Based Admission Control (PBAC) and Measurement-Based Admission Control (MBAC). Although the measurement-based algorithms are usually considered "approximations" of the parameter-based ones, we support the thesis that it is not true, on the contrary MBAC schemes are *in principle superior* to PBAC schemes when Long Range Dependence (LRD) comes into play. This result is due to two main interrelated effects: 1) MBAC ability to forecast traffic is more useful than PBAC a priori knowledge when the traffic exhibits long scale temporal correlation, 2) MBAC is able to highly reduce traffic correlation by compensating traffic variability through the admission process.

The chapter is organized as follows. Sections 2.1 and 2.2 briefly overview call admission control and results about LRD in data networks. Sec. 2.3 intuitively explains why MBAC could outperform PBAC. The specific MBAC scheme adopted, the simulation details and the statistical analysis are described in Sec. 2.4. Finally performance results are presented and discussed in section Sec. 2.5. Analytical background on LRD, self-similarity and heavy-tailedness is provided in Appendix A.

2.1 Call Admission Control

The role of any admission control algorithm is to ensure that admittance of a new flow into a resource constrained network does not violate service commitments made by the network to admitted flows (see also Sec. 1.3.5).

There are two basic approaches to admission control: 1) the parameter-based approach computes the amount of network resources required to support a set of flows given a priori flow characteristics, 2) the measurement-based approach relies on measurement of actual traffic load in making admission decisions.

Parameter-Based Admission Control (PBAC) algorithms usually require that the user declares its traffic characteristics at the connection setup time. The flow is then enforced to declared value by policing at the edge routers (see Sec. B.4). The set of traffic characteristics the user declares is called *traffic descriptor*. It needs to be easily specified by the user, easily monitored by the network, and suitable for online performance evaluation. The traffic descriptor has been standardized in the ATM framework [1, 43]: it includes peak cell rate and cell delay variation tolerance, sustainable cell rate, and burst tolerance. Bibliography about PBAC algorithms can be found in [151].

Parameter-Based Admission Control algorithms can be more easily analyzed by formal methods in comparison to measurement-based ones. Theoretical analysis is quite different depending whether the buffering effect is taken into account in evaluating performance. Methods in which the buffering effect is considered are called Rate-Sharing Multiplexing (RSM) methods, and those in which the buffering effect is not considered are called Rate-Envelope Multiplexing (REM) methods. If RSM methods are considered, we need to model the queuing process at the output port buffer in the ATM switch. Many techniques exist for modeling such queuing, such as MMPP/D/1/K, MMBP/D/1/K, and so on. By solving the queuing model, one can evaluate packet loss probability or queuing delay or link utilization. A strength of the RSM methods is that they can achieve high efficiency because they consider the buffering effect. But in general, they require a significant amount of processing power. In addition, RSM methods are dependent on the input traffic model. By contrast, with REM methods the queuing process at the output port buffer needs not be considered. When the aggregate data rate exceeds

link capacity, excess packets are deemed to be discarded immediately. In this case the virtual packet loss probability is calculated by using the peak and average rate; it does not require any assumptions on burst length or inter-burst length distributions.

While traditional CAC methods rely on the a priori knowledge of the statistical characterization of the offered traffic, Measurement-Based Admission Control (MBAC) algorithms base the decision whether to accept or reject an incoming call on runtime measurements on the traffic aggregate. They are usually considered as the second-best solution when traffic descriptor are unknown or uncertain or when limited resources do not allow the network to track the number of active connections and their traffic descriptor. As we are going to show in this chapter MBAC can be a much more powerful tool.

A large number of MBAC algorithms have been proposed in literature (e.g. [62, 69, 92, 75]). In [75] it has been observed that many MBAC schemes rely on certainty equivalence assumption. These methods use a static AC algorithm, but insert measured quantities rather than a priori known traffic descriptors (and these measured quantities are assumed to be the "real ones"). Despite the attractive feature of reusing existing PBAC algorithms, the authors show that this assumption can grossly compromise the target performance of the system: overload is due to jointly occurring "misleading measurements" (giving an overly optimistic impression of the momentary load) and rare behavior in the period after the measurement. Certainty equivalence methods neglect the first type of error. At the same time [33] introduces the separation between the admission criterion and the measurement procedure. The admission criterion determines on the basis of a number of traffic characteristics (of the existing flows and the new flow) whether or not to accept the new flow. The measurement procedure captures the required traffic characteristic from the flows that are currently present. The authors show that different MBAC schemes behave very similarly in terms of throughput/loss performance. In particular it appears that the measurement process, and in particular the length of the averaging periods and the way in which new flows are taken into account, are much more important than the specific admission criteria (either heuristic or theoretical) in determining how close MBAC schemes approach ideal CAC performance.

The admission control method can be centralized by one entity that maintains the topology as well as the state of all nodes in the network, thus eliminating the need of a distributed reservation state. An example of centralized solution in a DiffServ scenario is the bandwidth broker (Sec. B.7). A different approach is distributed admission control.

In the IntServ framework each node tracks the available resources for its links and performs a local decision upon receiving a service request and hop-by-hop resources are reserved in a path [92]. In the DiffServ framework core router are not flow-conscious and distributed admission control is performed at edge routers, which can have a limited knowledge of network status. Anyway starting from 1998, a number of proposals have shown that per-flow Distributed Admission Control schemes can be deployed over a DiffServ architecture (e.g. [94, 70, 39, 53, 26, 25]), resulting in statistical per-flow QoS guarantees. Such solutions are referred to as Endpoint Admission Control (EAC) schemes. EAC builds upon the idea that admission control can be managed by pure end-to-end operation, involving only the source and destination host. At connection set-up, each sender-receiver pair starts a "probing phase" whose goal is to determine whether the considered connection can be admitted into the network. Although the described scheme looks elegant and promising (it is scalable, it does not involve inner routers), a number of subtle issues come out when we look for QoS performance. A scheme purely based on endpoint measurements suffers of performance drawbacks mostly related to the necessarily limited (few hundreds of ms, for reasonably bounded call setup times) measurement time spent at the destination. Measurements taken over such a short time cannot capture stationary network states, and thus the decision whether to admit or reject a call is taken over a snapshot of the network status, which can be quite an unrealistic picture of the network congestion level. The simplest solution to the above issue is to attempt to convey more reliable network state information to the edge of the network. In such a view, EAC can be supported by a Measurement Based Admission Control (MBAC) that runs internally to the network (i.e., in a whole domain or in each internal router or in the edge devices), e.g. [39, 70, 53]. In [25] the authors propose GRIP (Gauge&Gate Reservation with Independent Probing) a reservation framework where Endpoint Admission Control decisions are driven by probing packet losses occurring in the internal network routers. The Gauge&Gate acronym stems from

the assumption that network routers are able to drive probing packet discarding (Gate) on the basis of accepted traffic measurements (Gauge), and thus implicitly convey congestion status information to the edge of the network by means of reception/lack of reception of probes independently generated by edge nodes.

2.2 Self-Similarity in IP Networks

The experimental evidence that packet network traffic shows self-similarity was first given in [99], where a thorough statistical study of large Ethernet traffic traces was carried out. This paper stimulated the research community to explore the various taste of self-similarity. This phenomenon has been also observed in wide area Internet traffic [159, 46], and many of the causes that contribute to self-similarity for both TCP [139, 55] and UDP [22] traffic aggregates have been now more fully understood. As regards practical consequences, many works [138, 117, 74, 166] show that self-similarity has a severe detrimental impact on network performance.

From an analytical point of view it is known that LRD asymptotically arises when many random variables, exhibiting heavy-tailedness are superposed (see Sec. A.5). At the same time there is widespread evidence [46, 162] that human as well as computer sources behave as heavy-tailed ON/OFF sources, so this result should be considered a physical explanation of the traffic self-similarity - independent of network or protocol characteristics [46] - rather than a mere way to generate self-similar traces.

An extensive bibliographical guide to self-similar traffic in data networks can be found in [161]. A more recent one is in [140].

2.3 Intuition

As we said in Sec. 2.1, it is frequently considered “obvious” that the ultimate goal of any MBAC scheme is to reach the “ideal” performance of a parameter-based CAC scheme. In fact, MBAC schemes are traditionally meant to approximate the operation of a parameter-based CAC (i.e. by estimating the

status of the system). They cannot rely on the detailed a priori knowledge of the statistical traffic characteristics, as this information is not easy supplied by the network customer. Therefore, their admission control (AC) decisions are based on an estimate of the network load obtained via a measurement process that runs on the accepted traffic aggregate[1].

However, a closer look at the basic principles underlying MBAC suggests that, in particular traffic conditions, these schemes might outperform traditional parameter-based CAC approaches. An initial insight into the performance benefits of MBAC versus parameter-based algorithms in an LRD traffic scenario is given in [33]. In this paper, we present additional results that confirm the superiority of MBAC and, in addition, we justify them showing that MBAC algorithms are able to reduce the self-similarity of the traffic aggregate generated by the admitted HT sources. In other words, we support the thesis that MBAC schemes are not just "approximations" of parameter-based CAC, but they are *in principle superior* to traditional CAC schemes when self-similarity comes into play.

An intuitive justification can be drawn by looking at the simulations presented in figures 2.1 and 2.2. Each figure shows two selected 200 s simulation samples, which for convenience have been placed adjacently. The y-axis represents the normalized link utilization. The figures report: i) the normalized number of accommodated calls[2]; ii) the link load, for graphical convenience averaged over a 1 s time window, and iii) the smoothed link load, as measured by the autoregressive filter adopted in the MBAC, whose time constant is of the order of 10 seconds.

Figure 2.1 plots results for the PBAC scheme. According to this scheme, a new flow is accepted only if the number of already admitted flows is lower than a maximum threshold N_t. In the simulation run N_t has been set to 129, which corresponds to a target link-utilization of about 88%, and a very high offered load (650%) was adopted. As a consequence, the number of flows admitted to the link sticks, in practice, to the upper limit.

The leftmost 200 simulation seconds, represented in figure 2.1, show that,

[1]The reader can recognize the underlying certainty equivalence assumption, described in Sec. 2.1.

[2]The simulation details are described in section 2.4. For what concerns the considered plots, a 100% link utilization in terms of calls, i.e. nominal average offered load equal to the link capacity, corresponds to 146.8 accommodated flows

owing to LRD of the accepted traffic, the load offered by the admitted sources is well above the nominal average load. Traffic bursts even greater than the link capacity are very frequent. On the other hand, as shown by the rightmost 200 seconds, there are long periods of time in which the system remains under-utilized. This is the so-called Joseph effect [105]. The criticality of self-similarity lies in the fact that the described situation occurs at time scales, which dramatically affect the loss/delay performance.

MBAC schemes behave very differently, as reported in figure 2.2 for the simple scheme described in section 2.4.2. In this case, new calls are blocked when the offered-load measurement is higher that 89%[3]. We see that the offered-load fluctuates slightly around the threshold. However, long term traffic bursts are dynamically compensated by a significant decrease in the number of admitted calls (leftmost plot). The opposite situation occurs when the admitted calls continually emit below their nominal average rate (rightmost plot): in these periods the number of admitted calls significantly increases. This "compensation" capability of MBAC schemes leads us to conclude that MBAC is very suited to operate in LRD traffic conditions, as quantitatively confirmed in section 2.5.

[3]The values 129 in PBAC and 89% in MBAC were selected so that the resulting average throughputs were the same.

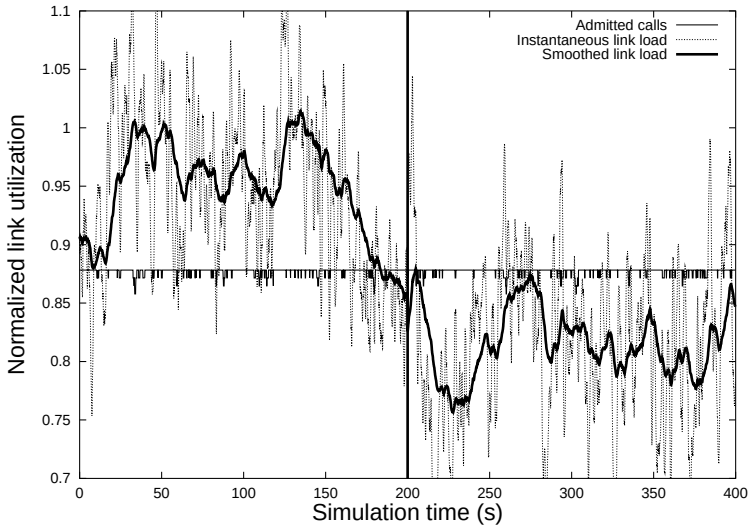

Figure 2.1: Traditional Admission Control operation

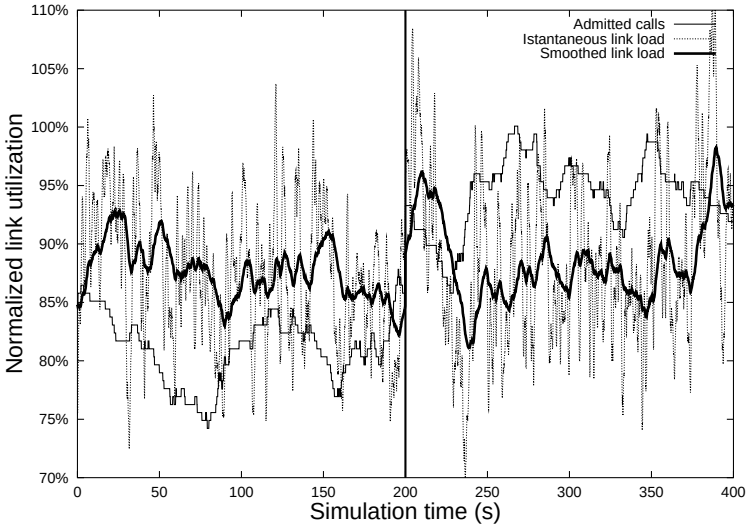

Figure 2.2: Measurement-Based Admission Control operation

2.4 The Simulation Scenario

To obtain simulation results, we have developed a C++ event-driven simulator. A batch simulation approach was adopted. The simulation time is divided into 101 intervals, each lasting 300 simulated minutes, and results collected in the first "warm-up" time interval are discarded.

As in many other admission control works [75, 33], the network model consists of a single bottleneck link. The reason is that the basic performance aspects of MBAC are most easily revealed in this simple network configuration rather than in a multi-link scenario. The link capacity was set equal to 2 Mbps and 5 Mbps. Most results are related to an infinite buffer size scenario. Thus, QoS is characterized by the delay (average and 99th delay percentiles) experienced by data packets rather than packet loss as in [33]. The rationale for using delay instead of loss is threefold. Firstly, loss performance depends on the buffer size adopted in the simulation runs, while delay performance does not require a choice of buffer size (we have actually used infinite buffer size). Secondly, the loss performance magnitude may be easily inferred, for a given buffer size, from the analysis of the distribution of the delay, which can be well summarized via selected delay percentiles. Thirdly, and most importantly, a limited buffer size acts as a smoothing mechanism for traffic bursts. Large packet losses, occurring during severe and persistent traffic bursts (as that expected for self-similar traffic), have a beneficial congestion control effect on the system performance. Conversely, in a very large buffer scenario, the system is forced to keep memory of non-smoothed traffic bursts and therefore performance is further degraded in the presence of high traffic variability[4].

As our performance figures, we evaluated link utilization (throughput) and delay distribution, summarized, for convenience of presentation, by the average and 99th delay percentile. The 95% confidence intervals have been evaluated.

[4] Specifically, this justifies the very different performance results we obtain in high utilization conditions when compared with the loss-utilization performance frontier presented in [33] for LRD sources. In that paper, unlike our results presented in figures 2.4, 2.5 and 2.6, it appears that performance of MBAC schemes tend to converge to the performance of traditional CAC schemes - i.e. the PBAC algorithm - as the utilization increases. A theoretical justification for this behavior can be found in [74], where the authors derive a formula to estimate the "correlation horizon" (which results to scale in linear proportion to the buffer size), beyond which the impact on loss performance of the correlation in the arrival process becomes nil.

In all cases, throughput results show a confidence interval always lower than 0.3%. Instead, despite the very long simulation time, higher confidence intervals occur for 99th delay percentile results: less than 5% for MBAC results, and as much as 25% for PBAC results (this is an obvious consequence of the self-similarity of the PBAC traffic aggregate). Nevertheless, even accounting for such an uncertainty on the results, the PBAC and MBAC delay performance are very different (Sec. 2.5).

2.4.1 Traffic Sources

For simplicity, we have considered a scenario composed of homogeneous flows. Each traffic source is modelled as an ON/OFF source. While in the ON state, a source transmits 1000 bit fixed size packets at a Peak Constant Rate (PCR) randomly generated in the small interval 31 to 33 Kbps (to avoid source synchronization effects at the packet level). Conversely, while in the OFF state, it remains idle. The mean value of the ON and OFF periods were set, respectively, equal to 1 s and 1.35 s (Brady model for voice traffic). This results in an average source rate $r = 0.4255 \cdot E[PCR] \approx 13.6$ Kbps. ON and OFF periods were drawn from two Pareto distributions with the same shaping parameter $c = 1.5$ (so they exhibit heavy-tails, in particular the variance is infinite), hence the aggregated traffic is self-similar [162].

Simulation experiments were obtained in a dynamic scenario consisting of randomly arriving flows. Each flow requests service from the network, and the decision whether to admit or reject the flow is taken by the specific simulated admission control algorithm. A rejected flow departs from the network without sending any data, and does not retry its service request again. The duration of an accepted flow is taken from a lognormal distribution [28] with mean 300 s and standard deviation 676 s (we adopted unitary variance for the corresponding normal distribution as reported in [28]), but call duration is extended to the end of the last ON or OFF period. Because of this, the real call-lifetime exhibits longer mean (320 s) and infinite variance. If the last burst were cut off, the process variance would become finite.

The flow arrival process is Poisson with arrival rate λ calls per second. For convenience, we refer to the normalized offered load $\rho = \lambda \cdot r \cdot T_{hold}/C_{link}$, being r the mean source rate, T_{hold} the average call duration and C_{link} the link

capacity. Depending on the simulation experiment, the arrival rate ranges from underload conditions (less than 50% of the link capacity) to severe overload conditions (up to 650%).

2.4.2 The MBAC Algorithm

Rather than using complex MBAC proposals, we have implemented a very basic MBAC approach. The rationale for the choice of a very simple MBAC scheme is twofold. Firstly, it has been shown [33] that different MBAC schemes behave very similarly in terms of throughput/loss performance. It appears that the length of the averaging periods and the way in which new flows are taken into account, are much more important than the specific admission criteria. Secondly, and more importantly, our goal is to show that the introduction of measurement in the admission control decision is the key to obtain performance advantages versus the PBAC approach, rather than the careful design of the MBAC algorithm. In this perspective the simpler the MBAC scheme is, the more general the conclusions are.

The specific MBAC implementation is described as follows. A discrete time scale is adopted, with sample time $T = 100$ ms. Let $X(k)$ be the load, in bits/sec, entering the link buffer during the time slot k, and let $B(k)$ be a running bandwidth estimate, smoothed by a simple first order autoregressive filter

$$B(k) = \alpha B(k-1) + (1-\alpha)X(k)$$

We chose $\alpha = 0.99$, corresponding to about 10 s time constant in the filter memory.

Consider now a call requesting admission during the slot $k+1$. The call is admitted if the estimated bandwidth $B(k)$ is less than a predetermined percentage of the link bandwidth. By tuning this percentage, performance figures can be obtained for various accepted load conditions.

An additional well-known issue in MBAC algorithm design [62, 92] is that, when a new flow is admitted, the slow responsiveness of the load estimate will not immediately reflect the presence of the new flow. A solution to prevent this performance-impairing situation is to artificially increase the load estimate to account for the new flow. In our implementation, the actual bandwidth

estimate $B(k)$ is updated by adding the average rate of the flow (i.e. $B(k) := B(k) + r$).

2.4.3 Statistical Analysis of Self-Similarity

The Hurst parameter H is able to quantify the self-similarity of the accepted traffic aggregate (Appendix A). For a wide range of stochastic processes $H = 0.5$ corresponds to uncorrelated observations, $H > 0.5$ to LRD processes and $H < 0.5$ to SRD processes.

In order to evaluate H, we used three well known methods: the *aggregate variance*, the *rescaled adjusted range* and the *wavelet estimators*. The methods are described in Sec. A.6.

We implemented by ourselves the first two methods, for the third a MatLab implementation is freely distributed [160]. All methods receive as input a realization $X(i)$ of the discrete-time stochastic process representing the load offered, during a 100 ms time window, to the link buffer by the accepted traffic aggregate. A common problem is to determine over which scales LRD property exists, or equivalently the alignment region in the logscale diagrams. Using the fit test of the matlab tool [160] we determined for our traces the range from 2000 s -11th octave- to 250000 s -18th octave- (the two last octaves were discarded because there were too few values). All the three methods were applied over this scale.

2.5 Performance results

The results shown in this section appeared in [122, 123, 124].

A problem arising in the comparison of different CAC schemes is the definition of a throughput/performance operational trade-off. In general, CAC schemes have some tunable parameters that allow the network operator to set a suitable *utilization target* and a consequent QoS provisioning. For example, in the case of the ideal PBAC algorithm, a higher setting of the threshold value results in an increased system throughput, at the expense of delay performance. By adjusting these parameters, CAC rules can be designed to be more aggressive or conservative with regard to the number of flows admitted.

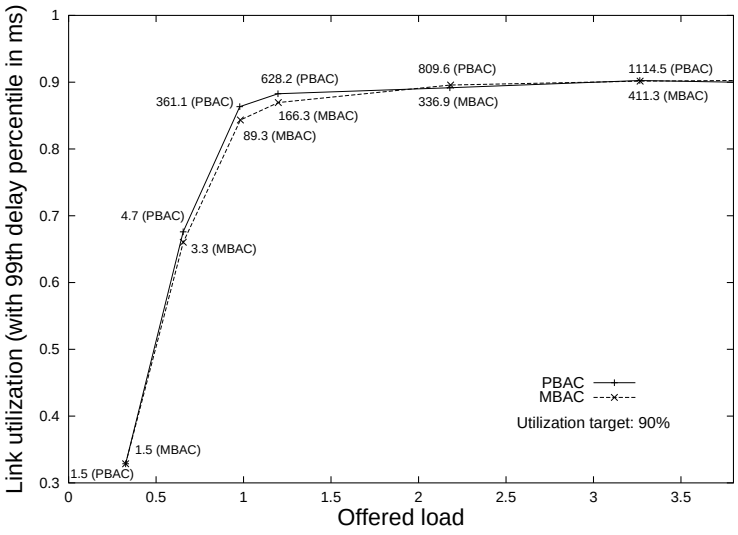

Figure 2.3: Link utilization vs offered load

Results presented in Fig. 2.3 were obtained by setting the PBAC and MBAC tuning parameters so that a target 90% link-utilization performance is achieved in overload conditions. The figure compares the throughput/delay performance (99th delay percentiles, measured in ms, are reported) of MBAC and PBAC, versus the normalized offered load. Minor differences can be noted in the capability of the considered schemes to achieve the performance target (as expected, PBAC converges faster than MBAC to the utilization target). A much more interesting result is the significantly lower MBAC 99th delay performance versus the PBAC one.

It is restrictive to limit the investigation to a single level of performance, but it is preferable to compare different CAC schemes for a wide range of link utilization targets (and, correspondingly, QoS performance), obtained by varying the CAC threshold parameters. Unless otherwise specified, all results presented in what follows are obtained in large overload conditions (650% offered load).

Rather than varying the offered load, Fig. 2.4 compares MBAC and PBAC by plotting their QoS performance versus the link utilization (following [33], the QoS versus utilization curve is called *Performance Frontier*). The figure reports the delay/utilization performance frontiers of PBAC and MBAC in terms of 99th delay percentiles. The figure reports the results obtained for both LRD and Markovian flows. It is shown that better performance are obtained using a

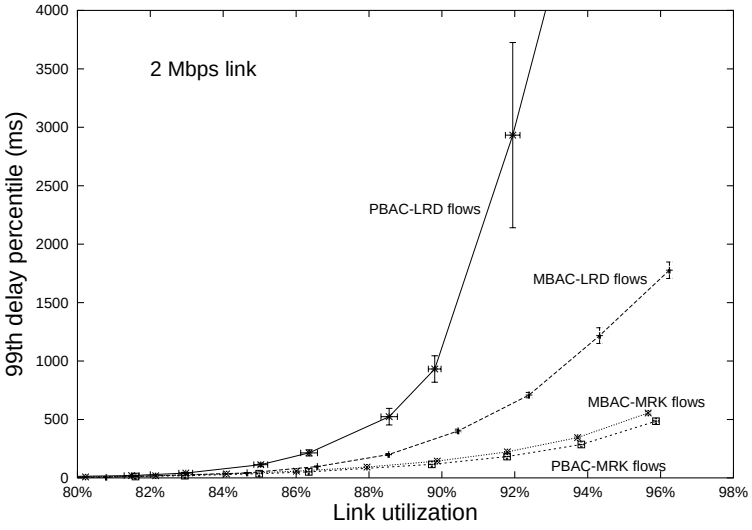

Figure 2.4: Delay performance vs link utilization (2 Mbps link)

Markovian traffic model, but it is also enlightened the remarkable performance improvement provided by MBAC with respect to PBAC in LRD assumptions, especially for large link utilization. Considering Markovian flows, PBAC acts slightly better than MBAC, in fact no memory arises in offered traffic process, thus the best bandwidth control simply consists in monitoring the number of admitted connections. Instead, with LRD flows, MBAC performance frontiers assume intermediate values, between PBAC-LRD and Markovian curves. Thus, MBAC appears to be more robust than PBAC to the traffic statistical properties. Average delays with LRD flows are shown in Fig. 2.6. Moreover, in Fig. 2.5 the performance frontiers are plotted for a 5 Mbps link. Beside the general performance improvement in comparison to the 2 Mbps link scenario shown in Fig. 2.4, one can see that MBAC behavior, with Markovian traffic, is closer to the PBAC behavior, since the traffic granularity is reduced and the impact of a flow erroneously admitted is less significant.

We argue that the performance enhancement of MBAC over PBAC is due to the beneficial effect of MBAC in reducing the self-similarity of the accepted traffic aggregate. A visual comparison of the temporal behavior of MBAC and PBAC is proposed in Fig. 2.7. For a time scale 10 times greater than that used in figures 2.1 and 2.2, Fig. 2.7 reports, for both PBAC (left) and MBAC (right) algorithms, i) the number of admitted calls, ii) the instantaneous offered load (averaged over a 1 s period) and iii) the offered load measured by the

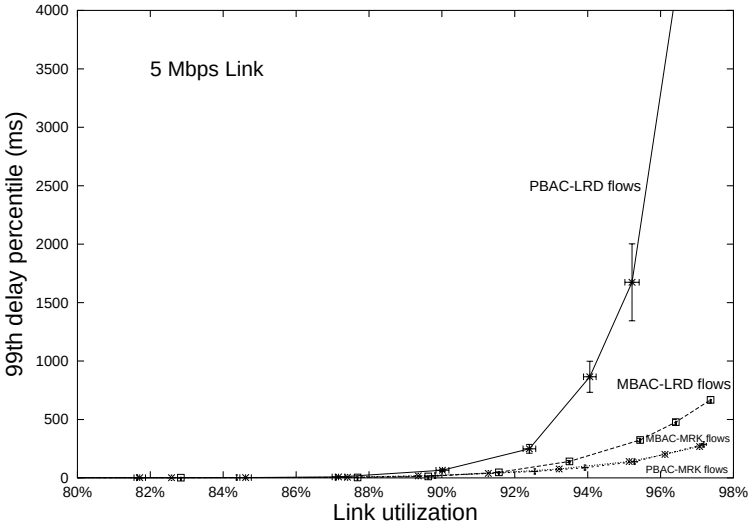

Figure 2.5: Delay performance vs link utilization (5 Mbps link)

autoregressive MBAC filter. For convenience of visualization, the three plots have been separated on the y-axis scale. The different temporal behavior of PBAC and MBAC is clearly visible.

To quantify the time behavior of the two PBAC and MBAC traffic aggregate time series, Fig. 2.8 reports a log-log plot of the aggregate variance, computed as described in Sec. 2.4.3. While the two curves exhibit similar behavior for small values of the aggregation scale, the asymptotic slope of the PBAC plot is very different from the MBAC one, suggesting that the MBAC-controlled traffic is not self-similar ($H \sim 0.5$). We recall that the asymptotic slope β is related to H by $\beta = 2H - 2$. The lines corresponding to $H = 0.50$, $H = 0.55$, $H = 0.75$ and $H = 0.80$ are plotted in the figure as reference comparison.

Similar considerations can be drawn by looking at Fig. 2.9, which plots the estimated squared wavelet coefficients $d_x^2(j, l)$ versus the basis-function time scale. 95% confidence intervals under gaussian assumption are depicted. For reference purposes, the lines corresponding to $H = 0.50$, and $H = 0.80$ are also plotted in the figure. An interesting consideration is that in both figures the MBAC curve departs from the PBAC curve at a time scale of the order of about 100 seconds. Although a thorough understanding of the emergence of such a specific time scale is outside the scope of the present paper, we suggest that it might have a close relationship with the concept of "critical time scale"

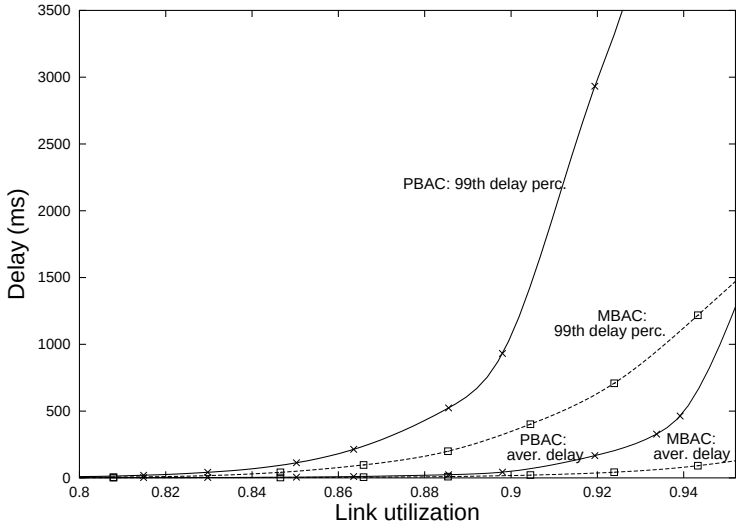

Figure 2.6: Delay performance vs link utilization with LRD flows (2 Mbps link)

outlined in [75].

The H estimates are reported in tables 2.1 and 2.2, with the corresponding CAC settings (the maximum call number for PBAC and the maximum link utilization for MBAC), and the achieved link utilization. For the wavelet estimates 95% confidence interval are also indicated. The three methods described in Sec. 2.4.3, provide congruent estimates. Results are impressive, and show that H decreases from about 0.75, in the case of PBAC, to about 0.5 for MBAC. It is interesting to note that 0.75 is the H value theoretically calculated in [73], [162] and [100] under different assumptions (see also Theorem A.5.1 and Theorem A.5.2), when a flow has HT periods of activity/inactivity with a shaping parameter $c = 1.5$ (the formula is $H = (3 - c)/2$). We note that, as expected, H does not depend on the link utilization. In conclusion, table 2.2 quantitatively supports our thesis that self-similarity is a marginal phenomenon for MBAC controlled traffic (H close to 0.5).

Further results obtained by considering a finite buffer scenario are depicted in Fig. 2.10 and 2.11, where buffers having respectively 100 and 2000 packets length, are considered. Note that in the former case, in which a short buffer of 100 packets is considered, PBAC performance is comparable to the MBAC one only in extreme underload condition or conversely in a harsh over-utilized scenario. In the latter case, with a 2000 packets buffer, MBAC performance does

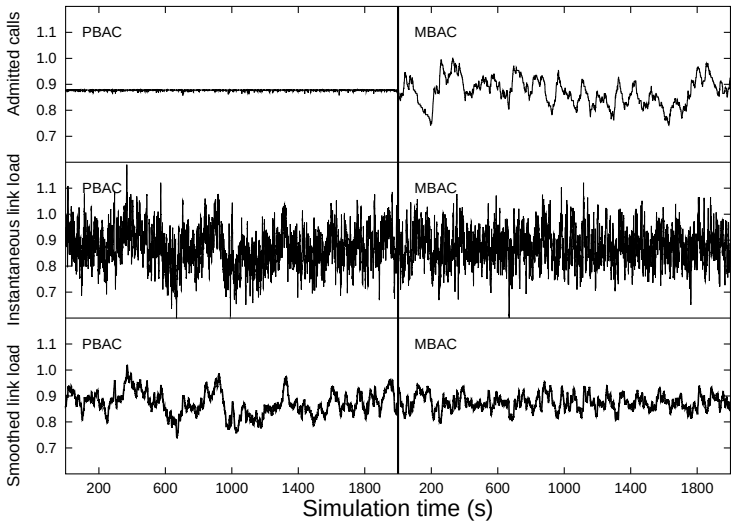

Figure 2.7: Time traces for PBAC and MBAC

not saturate at all, while in the PBAC scenario a greater buffer length should be needed in order to avoid losses occurring when the buffer is entirely filled. These figures show, in the same conditions as before, i.e. in very high offered load conditions, that the performance gain resulting from MBAC scheme adoption is not impaired by a finite buffer application. Moreover Fig. 2.12 shows how MBAC approach allows a significantly improvement in packet loss ratio, which is controlled well under 1% even in a high offered load and a high link utilization scenario.

Table 2.3 reports the Hurst parameter estimate obtained via the Wavelet method, when a finite and short buffer is employed (the link bandwidth is 2Mbps). Even if these results relate to a short buffer scenario (100 packets instead of infinite), the values reported in table 2.3 for H are very similar to those of tables 2.1 and 2.2, where an infinite buffer size was adopted.

The impact of the connection duration is drawn in Fig. 2.13. The ability of reducing traffic LRD is more effective as the duration increases. In fact, MBAC measurements are not able to efficiently track traffic variability when the holding time is too short in comparison to the filter memory.

Most of the previous results were obtained under constant overload conditions. The aim of the Fig. 2.14 is to show the behavior of the two CAC schemes in a wide range of offered load conditions, with a fixed target link-utilization (the thresholds chosen, 110 connection for PBAC and 77% for MBAC, give

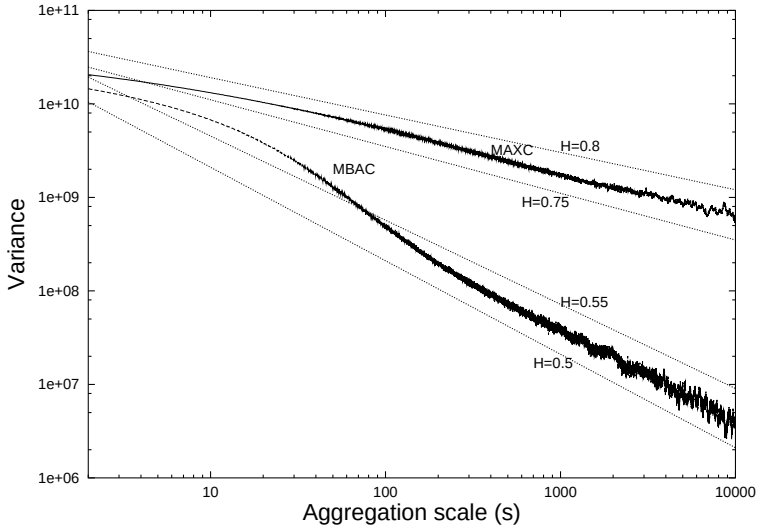

Figure 2.8: Aggregate variance plot

very similar throughput performance under the same offered load). The vertical dashed line corresponds to this target. Hurst-parameter was estimated by the wavelet estimator. When the offered load is below the target, the Hurst-parameter estimates[5] are quite similar because MBAC and PBAC do not enforce any rejection. By the way, in this situation, no need of access control arises and delay/loss performance copes with high QoS requirements. Instead, the effect of CAC rules becomes evident when the offered load exceeds the target utilization: the PBAC curve approaches to $H = 0.75$, while the MBAC one decays and approaches to non LRD values. Moreover, the uncertainty of statistical results is shown by plotting several points for each simulated scenario, obtained with different seeds for the random generator.

[5]In accordance with the fit test of the matlab tool [160] the LRD hypothesis on the range from 2000 s to 250000 s (see section 2.4.3) should not be rejected.

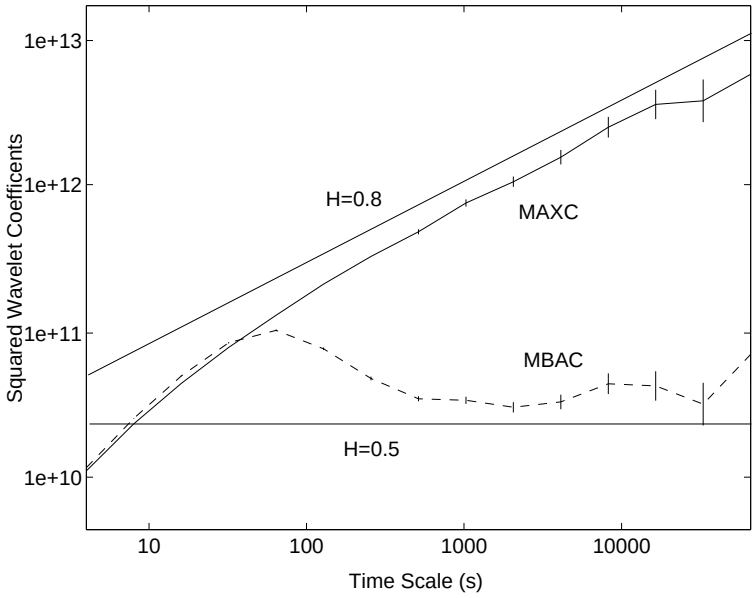

Figure 2.9: Wavelet coefficients plot

	Thresh (calls)	Thrput%	H-Variance	H-R/S	H-Wavelet
	105	71.8	0.73	0.79	0.78 [0.74,0.82]
M	115	78.3	0.74	0.78	0.80 [0.76,0.84]
A	125	84.5	0.71	0.79	0.75 [0.71,0.79]
X	130	88.7	0.78	0.76	0.75 [0.71,0.79]
C	135	91.7	0.72	0.72	0.77 [0.74,0.81]
	140	94.7	0.78	0.80	0.74 [0.70,0.78]

Table 2.1: Hurst-parameter estimate for PBAC controlled traffic

	Thresh (util%)	Thrput%	H-Variance	H-R/S	H-Wavelet
	70	69.1	0.55	0.48	0.55 [0.51,0.58]
M	78	76.9	0.58	0.54	0.58 [0.54,0.62]
B	86	84.6	0.55	0.51	0.60 [0.56,0.64]
A	90	88.5	0.60	0.52	0.57 [0.53,0.60]
C	94	92.4	0.51	0.46	0.56 [0.52,0.60]
	96	94.3	0.58	0.52	0.58 [0.54,0.62]

Table 2.2: Hurst-parameter estimate for MBAC controlled traffic

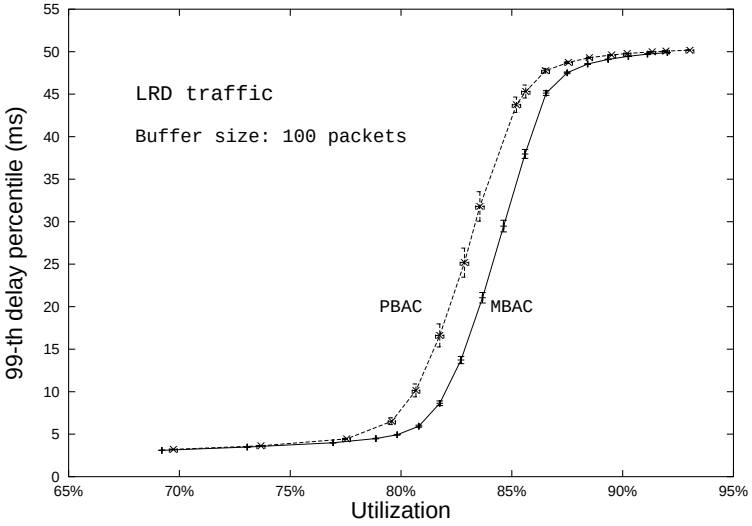

Figure 2.10: Delay performance vs link utilization, using a 100 packets buffer

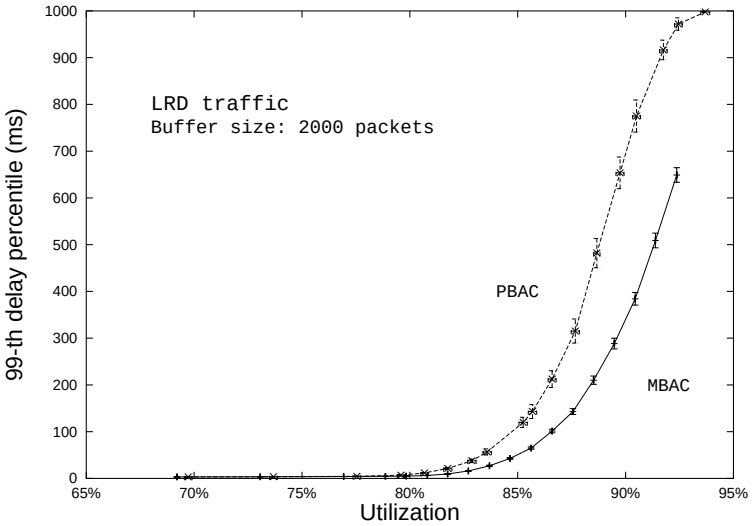

Figure 2.11: Delay performance vs link utilization, using a 2000 packets buffer

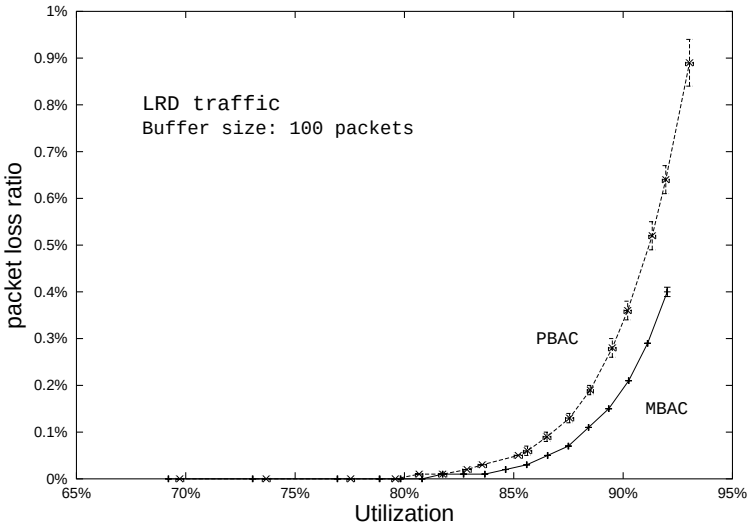

Figure 2.12: Packet loss ratio in high offered load condition, using a 100 packet buffer

Thresh (util%)	Hurst PBAC	Hurst MBAC
70	0.75 [0.72,0.80]	0.51 [0.47,0.55]
80	0.74 [0.70,0.78]	0.60 [0.56,0.63]
85	0.75 [0.71,0.79]	0.51 [0.48,0.55]
90	0.75 [0.71,0.79]	0.51 [0.48,0.55]
94	0.79 [0.75,0.82]	0.51 [0.48,0.55]

Table 2.3: Hurst-parameter Wavelet estimate for PBAC and MBAC, using a 100 packets buffer

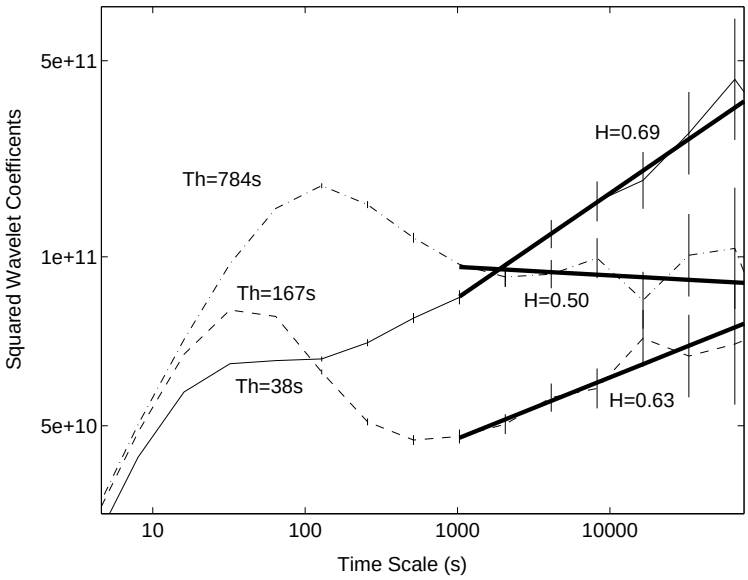

Figure 2.13: Wavelet coefficients plot for different holding time

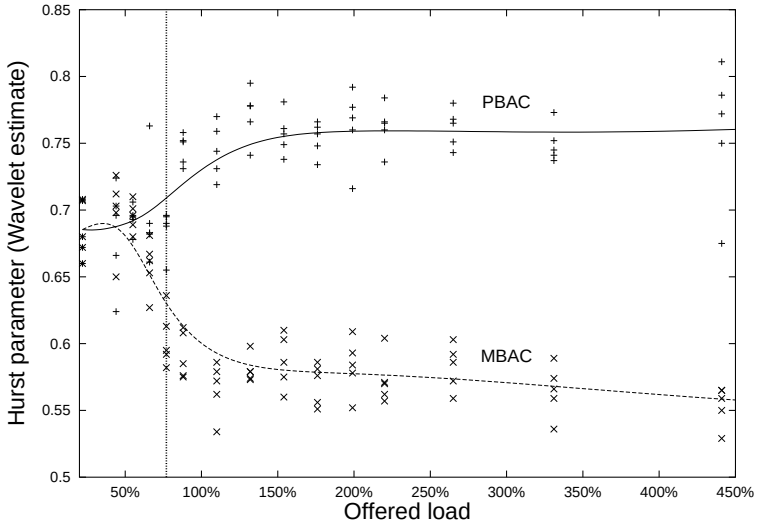

Figure 2.14: Hurst parameter vs offered load

Chapter 3

Active Queue Management Stability in Multiple Bottleneck Networks

In this chapter we stress the spatially distributed feature of Active Queue Management (AQM) control on TCP flows. In particular we discuss the influence of multiple bottlenecks on the stability of AQM controllers, usually configured on a single bottleneck basis. To see this, we consider a network scenario where Random Early Detection (RED) -a well known AQM scheme- is configured at each router according to previously developed control theoretic techniques. These configuration rules assure stability in a single bottleneck scenario. Yet, we show that instability may arise when two links become congested. We justify this result through a multiple bottleneck model using the Generalized Nyquist stability criterion.

The chapter is organized as follows. After an overview of AQM in Sec. 3.1, the motivation of our research is presented in Sec. 3.2. Then Sec. 3.3 recollects some results about RED stability and configuration criteria from [83]. In Sec. 3.4 we present our multiple bottleneck counterexample, and prove analytically that instability can arise. In Sec. 3.5 we show, by simulation, results what instability implies as regards network performance. Finally, conclusive remarks are given in Section 3.6.

3.1 AQM Overview

In this section we firstly present the AQM rationale (following [30]), then we describe the two AQM schemes considered in this thesis (RED and RIO) and other AQM proposals. Usually AQM schemes are intended to drop packets in order to implicitly signal congestion, but an apposite form of Explicit Congestion Notification (ECN) is also contemplated (Sec. 3.1.5).

3.1.1 Need for AQM

The common method for managing router queue lengths is known as Tail Drop: a maximum length for each queue is fixed, all the incoming packets are accepted until this limit is reached, then subsequent incoming packets are refused until the queue decreases because a packet from the queue has been transmitted. This is the traditional mechanism that has been used in the Internet for years, but it has two important disadvantages, as stated in [30].

Lock-Out. The tail drop mechanism may allow a monopolization of queue resources by a single or few flows, denying other connections the possibility to find place in the router buffer.

Full Queues. Buffering in the network is needed in order to absorb data bursts and to transmit them during the ensuing bursts of silence. At the same time we would like to have normally-small queues in routers in order to keep low end-to-end delay. The tail drop discipline allows queues to maintain a full (or almost full) status for long periods of time, since tail drop signals congestion (via a packet drop) only when the queue has become full. As a consequence end-to-end delay is high. Besides if the queue is full or almost full, an arriving burst will cause multiple packets to be dropped. This can result in a global synchronization of flows throttling back, followed by a sustained period of lowered link utilization, reducing overall throughput. It is important to decouple the steady-state queue size, and the ability to absorb traffic bursts.

Approaches like “random drop on full” or “drop front on full” are similar to Tail Drop. When the queue is full these disciplines respectively discard a

random selected packet and the packet at the front of the queue. They usually solve the lock-out problem, but the full queues problem holds.

In the Internet many flows are *responsive*, i.e. they throttle back in response to congestion notification. In particular most of the traffic in the current Internet employs the Transmission Control Protocol (TCP) as transport protocol. TCP is a connection-oriented, end-to-end, reliable protocol and reacts to congestion by reducing its transmission rate (specifically the sender sliding window size). One TCP assumption is that packet loss is not due to damage but to congestion in the network[1]. TCP detects packet losses by the retransmit timer timeout or by the reception of a triple-duplicate ack (when fast recovery is implemented). Other protocols employ adaptation rules similar to TCP in order to achieve fair resource employment. For this reason they are said to be *TCP-friendly*.

TCP responsiveness is the basis of *active* queue management. The idea is to employ packets dropping as a congestion notification to end nodes: in particular packets can be dropped before a queue becomes full in order to prevent an incipient congestion. Such proactive approach has three main advantages.

1. AQM solves the full queues problem. In fact it is able to keep the average queue size (hence the average queueing delay) small. At the same time packet bursts can be absorbed reducing the number of packet dropped.

2. AQM can prevent lock-out behavior by ensuring that there will almost always be space in the buffer for an incoming packet. This effect increases fairness, even if general fairness among flows requires per-flow state, which is not provided by queue management. Per-flow scheduling can be useful employed with AQM in order to achieve fairness.

3. AQM allows routers to control when and how many packets to drop. Hence the router can avoid multiple subsequent packets to be dropped, which can cause global synchronization or some flows starvation. In fact TCP recovers with more difficulty from a burst of packet drops than from a single packet drop.

[1]This assumption is realistic in wired network. On the contrary TCP performance are poor in wireless networks, which are characterized by losses due to transmission errors and handoffs.

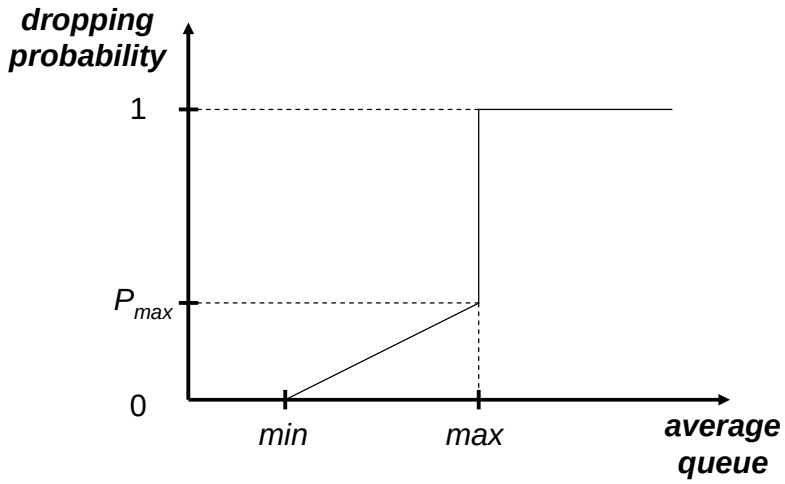

Figure 3.1: RED dropping function

3.1.2 Random Early Detection (RED)

RED has been proposed in [65] and is the most popular AQM mechanism. The RED algorithm drops arriving packets probabilistically. The probability depends on the estimated average queue size.

RED gateway calculates the average queue size (avg), using a low-pass filter with an exponential weighted moving average (avg) of the instantaneous queue (q): $avg = w_q * q + (1 - w_q) * avg$. The average queue size is compared to two thresholds, a minimum threshold and a maximum threshold.

When the average queue size is less than the minimum threshold, no packets are dropped. When it is greater than the maximum threshold, every arriving packet is dropped. When the average queue size is between the minimum and the maximum threshold, each arriving packet is dropped with probability p, where p is a linearly increasing function of the average queue size as it is shown in Fig. 3.1. Hence RED configuration is specified through three parameters: the minimum and the maximum threshold and the maximum dropping probability in the region of random discard P_{max}.

Given a steady average queue avg, the number of packets that arrive, after a dropped packet, until the next packet is dropped -say X- is a geometric random variable with mean value $1/p$. It is undesirable to have too many marked packets close together, and it is also undesirable to have too long an interval between marked packets. For this reason another version of the

algorithm has been proposed. It produces a uniform distribution for X. The variable p is evaluated according to the previous description, but the final dropping probability p_b is different and it is evaluated as $p_b = p/(1-count*p)$, where *count* is the number of undropped packets that have arrived since the last dropped packet.

3.1.3 RED with In/Out bit (RIO)

RIO has been proposed in [42]. It uses two twin RED algorithms for dropping packets, one for IN packets and one for OUT packets which share the same physical queue. So RIO is configured with two sets of RED parameters: (min_{in},max_{in},$P_{max_{in}}$) and (min_{out},max_{out},$P_{max_{out}}$). RIO discriminates against OUT packets in times of congestion essentially in two way: firstly IN dropping probability depends on the average queue for the IN packets, while OUT dropping probability on the average total queue; secondly parameter are opportunely chosen for the two kinds of traffic. In [42] the authors suggest the following rules:

$$min_{out} < min_{in}, \quad max_{out} << max_{in}, \quad P_{max_{out}} > P_{max_{in}},$$

and in the paper they choose $max_{out} < min_{in}$.

3.1.4 Other Mechanisms

Many other mechanisms have been proposed. Sometimes they are variants of the original RED aiming to improve some aspect. Here we briefly present some of them.

Adaptive RED (ARED) [58, 64]. It focuses on the problem of parameterizing the RED algorithm in order to reach good performance in each possible scenario. RED performance, as regards both average queueing delay and throughput have shown to be dependent on the traffic load. In particular a more aggressive RED is required when the number of TCP flows is higher and vice versa. Adaptive RED provide an adaptive variation of RED parameters based on the average queue. The key idea is to adapt P_{max} in order to keep the average queue size between the two thresholds.

RED with Penalty Box [143]. It is intended to distinguish responsive users from unresponsive users. It is based on the observation that high bandwidth flows see proportionally larger amounts of packet loss. It maintains a list of the recent packet loss events in order to identify such unresponsive flows. The rate of these flows is limited using mechanisms like as class-based queueing.

Flow RED (FRED) [101]. It tries to solve some particular cases of unfairness allowed in RED. In fact RED does not guarantee to give each connection the same fraction of the total resources and ignore the control on misbehaving flows. FRED is an improvement of RED able to provide major protection for bursty and low-speed flows by keeping an information state for those flows that have packets buffered in the router. It introduces new parameters that allow a more accurate estimation of discarding probability. FRED allows each connection to buffer $minq$ packets and apply discard probability to the subsequent packets. It never permits a flow to buffer more than $maxq$ packets and it counts how many times a flow tries to exceed $maxq$. Flows with high $strike$ values experiment higher dropping probability and they cannot buffer more than $avgcq$ packets.

Stabilized RED (SRED) [134]. It is a RED-derived mechanism that attempts to improve RED performance by estimating the number of flows going through the router. The basic idea is to compare every new incoming packet to one entry randomly taken from the so-called Zombie list. The discard probability is entirely based on the instantaneous queue length and also on the estimated number of active flows. This improvement has the advantage to stabilize the buffer occupancy, independently of the number of active connections and to provide a way to detect misbehaving flows.

CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows (CHOKe) [137]. It is another proposal, based on the RED algorithm, whose goal is to approximate fair queueing and be, at the same time, simple to implement. It is based on the assumption that the occupancy of a FIFO buffer is a reliable indication of which flows are consuming a great amount of resources. If a packet arrives and $avg > max$, the packet is

discarded, like in normal RED, if $avg > min$ the new packet is compared to a randomly chosen packet from the FIFO buffer. If they belong to the same flow they are both discarded, otherwise the randomly drawn packet is left in the queue and the new one is admitted with a probability calculated according to RED operation.

Random Exponential Marking (REM) [9]. It decouples congestion measure from performance measure such as loss, queue length or delay. While congestion measure indicates excess demand for bandwidth and must track the number of users, performance measure should be stabilized around their targets independently of the number of users. This approach is different from RED, where the queue occupancy is both a congestion measure and a performance index.

BLUE [57]. It uses a constant dropping probability p. If the router buffer often saturates p is incremented. On the other hand, if the queue is almost empty and the link is idle the p is reduced. In order to avoid instability a $freeze\_time$ is chosen as the minimum time interval between two successive updates of p. Stochastic Fair BLUE [59] is an extension of BLUE developed to solve the problem of non-responsive flows.

3.1.5 Explicit Congestion Notification

In an effort to reduce losses in TCP/IP networks, Explicit Congestion Notification (ECN) has been proposed as an additional congestion signal for TCP flows [143]. ECN allows to mark packets with a Congestion Experienced (CE) codepoint. When a packet marked with the CE codepoint is received by its destination, the data is acknowledged with a packet containing the CE-ECHO codepoint. When the CE-ECHO marked acknowledgment reaches the sender, the sender reduces its throughput, as if a loss had happened in the network.

3.2 Motivation

AQM algorithms usually rely on some heuristics and their performances appear to be highly dependent on the considered network scenario (see, e.g., [41, 108, 60], as regards the RED algorithm).

Our research has been motivated by the consideration that the distributed fashion of TCP flows control across the network has not been explicitly considered up to now. As a matter of fact TCP flows may turn to be controlled at the same time by two or more nodes acting independently according to their AQM settings. According to our opinion, this can hardly affect AQM algorithms performance. In particular, we propose a counterexample to show that RED controllers, configured according to [83], do not prevent from instability if two or more nodes face congestion at the same time (this is referred to as *multiple bottleneck scenario*). In our papers [118, 120, 119] instability in multiple bottleneck has been tackled by considering a distributed Multi-Input Multi-Output model, whose stability has been studied considering the poles of the rational Linear Time Invariant model obtained through linearization and Padé approximation for time delays. Here the MIMO linearized model is analyzed via the Generalized Nyquist stability criterion.

3.3 Single bottleneck model

The starting point in [83] is the model described by the following coupled, nonlinear differential equations:

$$\begin{aligned} \dot{W}(t) &= \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))}p(t-R(t)) \\ \dot{q}(t) &= \frac{W(t)}{R(t)}N(t) - 1_{q(t)}C \end{aligned} \tag{3.3.1}$$

where $1_q = 1$ if $q > 0$, $1_q = 0$ otherwise. Symbols used in the model above are summarized in the following table.

W	expected TCP window size (packets);
q	expected queue length (packets);
R	round-trip time;
C	link capacity (packets/sec);
R_0	propagation delay (secs);
N	load factor (number of TCP sessions);
p	probability of packet drop;

The first equation represents the TCP window, that increases by one every round trip time, and halves when a packet loss occurs. Packet loss rate is computed as the dropping probability times the number of packets sent per time unit. The round trip time is related to the propagation delay and the queue occupancy by the following relation: $R = R_0 + \frac{q}{C}$. The second equation represents the variation of queue occupancy as the difference between the input traffic and the link capacity.

AQM schemes determine the relation between the dropping probability and the nodes congestion status.

Here we considered RED as AQM scheme. RED configuration is specified through four parameters: the minimum and the maximum threshold (THR_{min}, THR_{max}), the maximum dropping probability in the region of random discard P_{max}, and the memory coefficient w_q. RED can be modelled by the following equations (refer to Sec. 3.1.2 for RED operation):

$$\begin{aligned} \dot{x}(t) &= -kx(t) + kq(t) \\ p(x) &= \begin{cases} 0, & 0 \leq x < THR_{min} \\ \frac{(x-THR_{min})P_{max}}{THR_{max}-THR_{min}}, & THR_{min} \leq x < THR_{max} \\ 1, & THR_{max} \leq x, \end{cases} \end{aligned} \tag{3.3.2}$$

where $k = -\ln(1-\alpha)/\delta$ and δ is the time between two queue samples. The time interval δ can be assumed to be equal to $1/C$ for a congested node.

The linearized system (TCP sources, congested node queue and AQM controller) can be represented by the block diagram of Fig. 3.2. In the block diagram $L_{RED} = P_{max}/(THR_{max} - THR_{min})$.

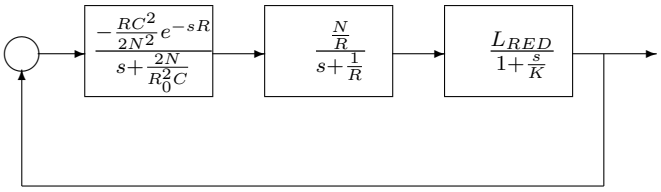

Figure 3.2: Linearized control system for a single bottleneck scenario

The open-loop transfer function of the system in Fig. 3.2 is:

$$H(s) = \frac{L_{RED}\frac{(RC)^3}{(2N)^2}e^{-sR}}{\left(1+\frac{s}{k}\right)\left(1+\frac{s}{\frac{2N}{R^2C}}\right)\left(1+\frac{s}{\frac{1}{R}}\right)} \tag{3.3.3}$$

In [83] the authors present RED configuration rules, that guarantee the stability of the linear feedback control system in Fig. 3.2 for $N \geq N^-$ and $R_0 \leq R^+$.

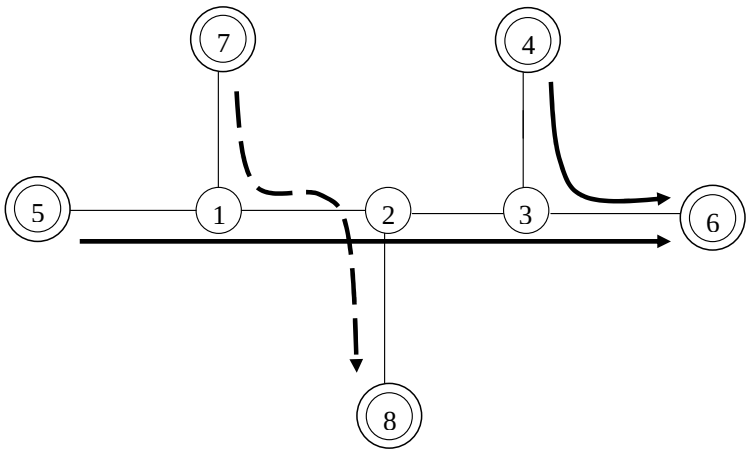

Figure 3.3: Network topology

3.4 An Instability Example

In this section we show the example of a network where each router has been configured so that stability is guaranteed when congestion occurs at a single link, but instability arises when there are two bottlenecks at the same time.

We consider a parking lot network whose topology is depicted in Fig. 3.3. The capacity and the propagation delay of each link are reported in Table 3.1.

Table 3.1: Network Parameters

Link	Capacity (Mbps)	Propagation Delay (ms)
5-1	20	2.1
1-2	10	10
2-3	20	2.1
4-3	20	19.1
3-6	10	10
7-1	20	9.4
2-8	20	9.4

Packet size is 1500 bytes. Links between nodes 3 and 6 and between nodes 1 and 2 will play the role of bottlenecks.

The RED algorithm is deployed at nodes 3 and 1, respectively to manage the output queues for the link $3-6$ and $1-2$. In what follows we refer to these buffers simply as node 3 buffer and node 1 buffer, without specifying the link.

We consider TCP flows aggregates from node 5 to node 6, from node 4 to node 6 and from node 7 to node 8. We indicate the number of flows of these aggregates respectively N_5, N_4 and N_7.

Our RED configuration relies on the control theoretic analysis of RED presented in [83]. Nevertheless, we do not adopt exactly the configuration rules proposed there, since their high stability margins do not allow simple counter-example. Then, we verify RED-configuration stability directly through the Nyquist plot of the open loop transfer function.

We recall that the Nyquist criterion allows one to study the stability of the closed loop system through the polar plot of the open loop transfer function $H(j\omega)$. For the functions we are interested in, the closed loop system is stable if and only if the plot does not encircle the point $(-1, 0)$.

We choose $THR_{min} = 2$, $THR_{max} = 20$, $P_{max} = 9\%$, and $w_q = 0.0017$.

The analysis in [83] shows that stability increases as the number of flows increases and the RTT decreases. Given a network scenario these quantities are not independent because the queueing delay, and hence the RTT, depends on the number of flows. In particular the higher the number of flows the bigger the queue. In fact in order to evaluate the equilibrium point, we can impose

the derivatives in equation system (3.3.1) equal to zero, obtaining

$$pW^2 = 2, \quad \frac{W}{R}N = C, \quad R = R_0 + \frac{q}{C},$$

and from equation system (3.3.2) considering that $q > THR_{min}$:

$$p = L_{RED}(q - THR_{min}).$$

It follows that the equilibrium queue value has to satisfy the following equation

$$(q - THR_{min})^2(q + CR_0)^2 = 2\frac{N^2}{L_{RED}^2}. \tag{3.4.1}$$

It is easy to see that as N increases, q increases.

The previous equation shows also that the most critical condition is the lowest number of flows. Indeed q is approximatively proportional to $\sqrt{N}$, hence the gain of $H(s)$ is proportional to $1/\sqrt{N}$. In conclusion we expect that if stability is guaranteed for a value N_- with a certain gain margin, then the gain margin is greater for $N > N^-$.

Given the RED configuration, if there are $N_5^- = N_4^- = 3$ flows (and $N_7^- = 0$) link $3-6$ is congested, the average queue is $q = 7$ packets. The maximum RTT is equal to 69 ms and the open loop transfer function $H(j\omega)$ does not encircle the point $(-1, 0)$. Hence the system is stable, in particular the gain margin is equal to 1.35. This gain margin assures stability even if flows number reduces to 5 and if the average queue length increases to 12 packets.

As it is expected, if $N_5 > N_5^-$ and $N_6 > N_6^-$ the gain margin increases. For example it is equal to 2.5 when $N_5 = N_4 = 5$.

Link bandwidths and propagation delays are such that the path from 4 to 6 and that from 7 to 8 have the same characteristics. Hence the same numerical results hold if only the link $1-2$ is congested due to aggregate 5 and aggregate 7, while $N_4 = 0$.

Let us assume that both links are congested and $N_5 = N_5^-$, $N_4 = N_4^-$ and $N_7 = N_7^-$. If we evaluate the new equilibrium point according to following equations, it holds $q_4 = q_2 \approx 5.48$. In comparison to the above situation, the RTT of aggregate 5 increases due to queueing delays at both node 3 and node 1 buffers, but the maximum RTT is always lower than 69 ms. Hence the *local* stability conditions are satisfied and we would expect the network to be stable and the gain margin should be even greater if $N_5 \geq N_5^-$, $N_4 \geq N_4^-$

and $N_7 \geq N_7^-$. Conversely we are going to show that a refined two-bottleneck model predicts instability.

3.4.1 Two Bottleneck Model

We extend the single bottleneck congestion model described in Section 3.3 to the case of two congested nodes. With reference to the network topology depicted in Fig. 3.3, according to notations introduced in [77] we obtain:

$$\begin{cases} \dot{W}_4 = \frac{1}{R_4} - \frac{W_4^2}{2R_4} p_3(t - \overleftarrow{R}_{34}) \\ \dot{W}_5 = \frac{1}{R_5} - \frac{W_5^2}{2R_5}(p_3(t - \overleftarrow{R}_{35}) + p_1(t - \overleftarrow{R}_{15})) \\ \dot{W}_7 = \frac{1}{R_7} - \frac{W_7^2}{2R_7} p_1(t - \overleftarrow{R}_{17}) \\ \dot{q}_3 = -C_3 I_{q>0} + \frac{N_4}{R_4} W_4(t - \overrightarrow{R}_{34}) + \frac{N_5}{R_5} W_5(t - \overrightarrow{R}_{35}) \\ \dot{q}_1 = -C_1 I_{q>0} + \frac{N_5}{R_5} W_5(t - \overrightarrow{R}_{15}) + \frac{N_7}{R_7} W_7(t - \overrightarrow{R}_{17}) \end{cases} \tag{3.4.2}$$

where W_j is the average window of the flows originating at node j, p_i is the dropping probability at node i buffer, $\overleftarrow{R}_{ij}$ represents the backward delay from node i buffer to source j (including queuing delay) and $\overrightarrow{R}_{ij}$ the forward delay from source j to node i buffer and $I_{q>0}$ represents a logical function that is equal to 1 if the queue $q > 0$ and zero otherwise. For sake of simplicity in equation system (3.4.2), the time dependence is indicated only for delayed function values.

Now, we linearize the Model 3.4.2 around the equilibrium point $(\hat{W}_4, \hat{W}_7, \hat{W}_5, \hat{p}_3, \hat{p}_1)$ that is the solution of the following system:

$$\begin{cases} W_4^2 p_3 = 2 \\ W_5^2(p_1 + p_3) = 2 \\ W_7^2 p_1 = 2 \\ \frac{W_4}{R_4} N_4 + \frac{W_5}{R_5} N_5 = C_3 \\ \frac{W_5}{R_5} N_5 + \frac{W_7}{R_7} N_7 = C_1 \\ p_3 = L_{RED3}(q_3 - THR_{min}) \\ p_1 = L_{RED1}(q_1 - THR_{min}) \\ R_4 = R_{40} + \frac{q_3}{C_3} \\ R_7 = R_{70} + \frac{q_1}{C_1} \\ R_5 = R_{50} + \frac{q_1}{C_1} + \frac{q_3}{C_3} \end{cases} \tag{3.4.3}$$

where R_{k0} is the round trip time experienced from source k when no congestion is present, and we omitted chapeau to simplify notation. In the Laplace domain we obtain the following relations:

$$w(s) = -F(s)\overleftarrow{R}(s)^T p(s) \tag{3.4.4}$$

$$q(s) = (sI + \Omega)^{-1}\overrightarrow{R}(s)NT^{-1}w(s) = G(s)w(s) \tag{3.4.5}$$

where $N = diag\{N_j\}$ and $T = diag\{R_j\}$ for $j = 4, 7, 5$ and

$$F(s) = \begin{bmatrix} \frac{W_4^2}{s+\frac{2}{W_4 R_4}} & 0 & 0 \\ 0 & \frac{W_7^2}{s+\frac{2}{W_7 R_7}} & 0 \\ 0 & 0 & \frac{W_5^2}{s+\frac{2}{W_5 R_5}} \end{bmatrix} \tag{3.4.6}$$

$$\Omega = \begin{bmatrix} \frac{N_5 W_5}{R_5^2 C_3} + \frac{N_4 W_4}{R_4^2 C_3} & \frac{N_5 W_5}{R_5^2 C_1} \\ \frac{N_5 W_5}{R_5^2 C_3} & \frac{N_5 W_5}{R_5^2 C_1} + \frac{N_7 W_7}{R_7^2 C_1} \end{bmatrix}$$

In the delay matrixes $\overleftarrow{R}(s)$ and $\overrightarrow{R}(s)$ the columns correspond to sources 4, 7 and 5, and the rows to buffer 3 and 1. In particular

$$\overleftarrow{R}(s) = \begin{bmatrix} e^{-s\overleftarrow{R}_{34}} & 0 & e^{-s\overleftarrow{R}_{35}} \\ 0 & e^{-s\overleftarrow{R}_{17}} & e^{-s\overleftarrow{R}_{15}} \end{bmatrix}$$

$$\overrightarrow{R}(s) = \begin{bmatrix} e^{-s\overrightarrow{R}_{34}} & 0 & e^{-s\overrightarrow{R}_{35}} \\ 0 & e^{-s\overrightarrow{R}_{17}} & e^{-s\overrightarrow{R}_{15}} \end{bmatrix}$$

Note that $\overrightarrow{R}(s=0) = \overleftarrow{R}(s=0)$ is the so called routing matrix of our scenario.

$$R(0) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{3.4.7}$$

The RED AQM control law that computes the packet marking probability p as a function of measured queue length q is $p(s) = K(s)q(s)$, where

$$K(s) = \begin{bmatrix} \frac{k_3 L_{RED3}}{s+k_3} & 0 \\ 0 & \frac{k_1 L_{RED1}}{s+k_1} \end{bmatrix} \tag{3.4.8}$$

The overall feedback loop is depicted in Fig. 3.4.

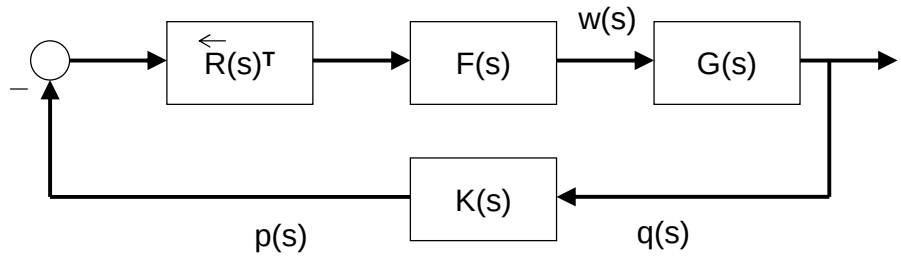

Figure 3.4: Linearized control system for the multiple bottleneck scenario

3.4.2 Stability Analysis

Many methods have been developed in the past fifty years to analyze the stability of time-delay system, (i.e. see [76] for a detailed survey), but for our purpose, the classical result of the Generalized Nyquist stability criterion [50] is sufficient and gives us an analytical tool that is easy to be verified. We recall it hereafter:

Theorem 3.4.1. *(**Generalized Nyquist Criterion**) If the open loop matrix $L(s)$ has $P0$ unstable poles, then the closed-loop system with return ratio $-kL(s)$ is stable if and only if the characteristic loci of $kL(s)$, taken together, encircle the point $(-1, 0)$, $P0$ times anticlockwise.*

As proved in [50], the above theorem is valid not only for the *lumped* case when $L(s)$ is a square rational transfer matrix $L(s)$, but it has been extended also to the *distributed* case. In the lumped case, $L(s)$ is factorized in two part, and $P0$ are the poles the rational part. In our case of interest, the open loop matrix is given by

$$L(s) = K(s)(sI + \Omega)^{-1}\overrightarrow{R}(s)NT^{-1}F(s)\overleftarrow{R}(s)^T$$

Applying the Generalized Nyquist criterion, we analyze the characteristic loci of $L(s)$ in the case of $N_4 = N_5 = N_7 = 3$, reported in Fig. 3.5.

Since the open loop is stable, and the characteristic loci of $L(j\omega)$ encircle the point $(-1, 0)$, two times anticlockwise, as shown in Fig. 3.6, the closed loop system is unstable.

This example shows the limits of local AQM configuration ignoring the distributed nature of TCP flows control in a multiple bottleneck scenario. If

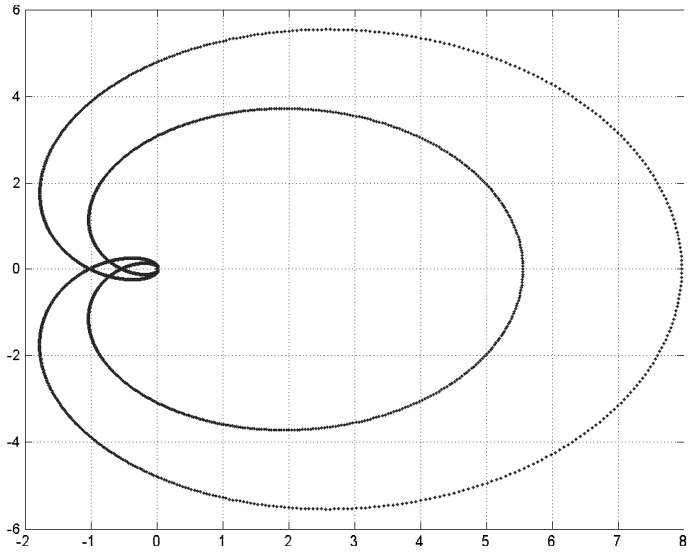

Figure 3.5: $L(s)$ characteristic loci in the case of $N_4 = N_5 = N_7 = 3$

we consider the configuration rules given in [83], instability probably does not arise in such a simple example, but there is a reduction of stability margins. This modifies the system dynamic response and reduces the system robustness to the flows number and the round trip time variation.

3.5 Simulation Results

In this section we investigate through simulations the previous analytical result. Firstly we are going to discuss what instability implies as regards network performance in the single bottleneck scenario. Secondly we show that the two bottleneck scenario with $N_5 = N_4 = N_7 = 3$ (unstable according to the analysis in Sec. 3.4.2) exhibits performance similar to those identified as instable in the single bottleneck scenario. Finally we observe similar results even in other scenarios, which are stable according to the two-bottleneck model.

Simulations were conducted through ns v2.1b9a [130]. We used TCP Reno implementation.

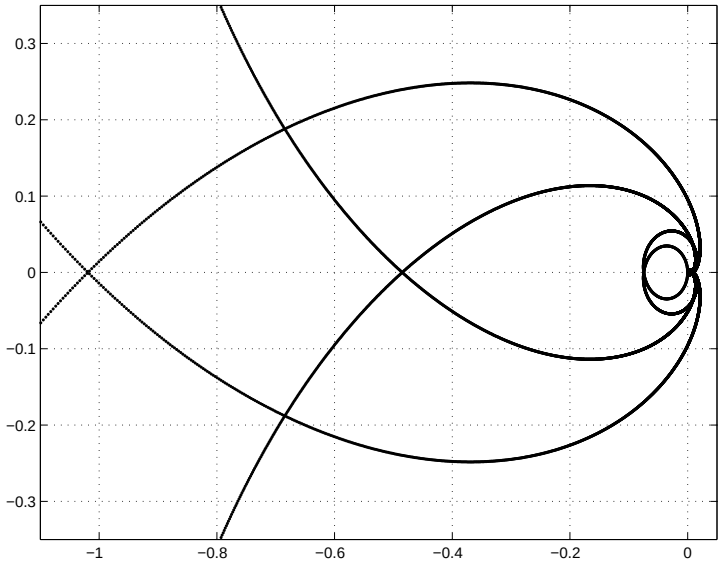

Figure 3.6: Particular of the $L(s)$ characteristic loci in the case of $N_4 = N_5 = N_7 = 3$

3.5.1 Single Bottleneck

Now, in order to analytically show how instability of the linear model concretely affects the network performance, we first present some results regarding the single bottleneck scenario.

When dealing with simulation tests a first issue regards the choice of a metric able to catch potential instability phenomena. For example [103] shows the oscillations of the TCP window and a deterministic limit cycles in the average window, averaged over all the flows of the same aggregate, and [135] shows nonlinear phenomena, such as bifurcations, using Liapunov exponents as a measure. Following the same line as in [83] we will look at the oscillating nature of the queue length to distinguish between stable or unstable behaviors. Differently from those papers, our results suggest that the amplitude of queue oscillations is not significant by itself when RED is considered. As we are going to show it can be more appropriate to consider the amplitude of queue oscillations in relation to the average queue value.

Let us consider two aggregates, each one of five TCP flows ($N = 10$), entering the network through node 4 and node 5 with destination node 6 (solid lines in Fig. 3.3). The link between nodes 3 and 6 is congested. Fig. 3.7 shows the instantaneous queue occupancy time-plot for the buffer at node 3.

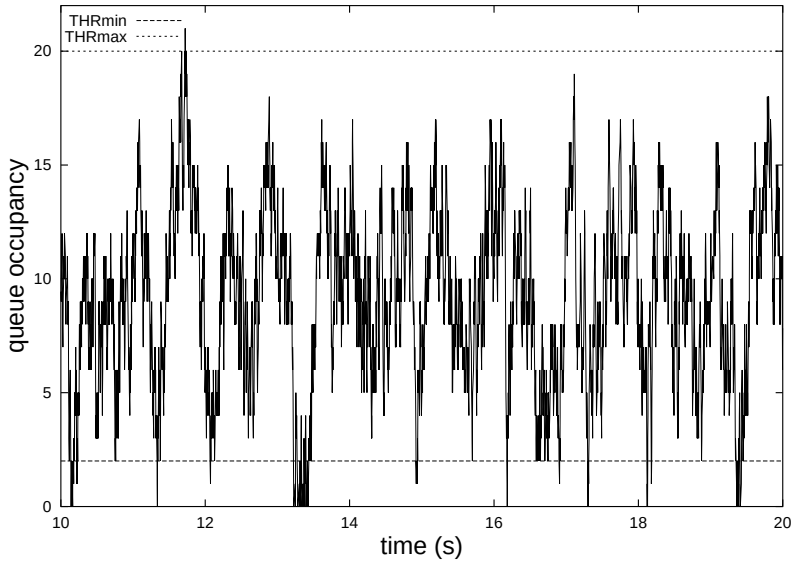

Figure 3.7: Instantaneous buffer occupancy with number of flows $N = 10$

RED should be able to keep the queue occupancy within the two thresholds (dotted lines).

Numerical results for the throughput and the queue are shown in Table 3.2. In particular the average queue size is in column "queue occupancy", while the standard deviation is in column "queue oscillation". The value in parentheses is the normalized queue oscillation, i.e. the ratio between the standard deviation and the average queue value.

Let us reduce the number of flows through the network and see if instability occurs as claimed in [83]. In Fig. 3.8 the buffer occupancy is shown to revisit with a higher frequency the regions associated to buffer underload (out of RED thresholds). As the total flow number decreases from 10 to 6 we note that i) the throughput over the link $3 - 6$ ($Thr_5 + Thr_4$) reduces from 9.91 Mbps to 9.74, ii) both the average queue occupancy and the oscillation amplitude decrease, respectively from 9.06 to 6.71 and from 4.72 to 4.09, and iii) the normalized standard deviation, i.e. the ratio between standard deviation and mean, increases from 0.52 to 0.61.

If we reduce drastically the number of flows to 2, the above RED configuration, turns to be too aggressive, which is evidenced by higher frequencies of buffer occupancy oscillations and further reduction of the throughput. Even longer periods, where buffer is underloaded results from Fig. 3.9.

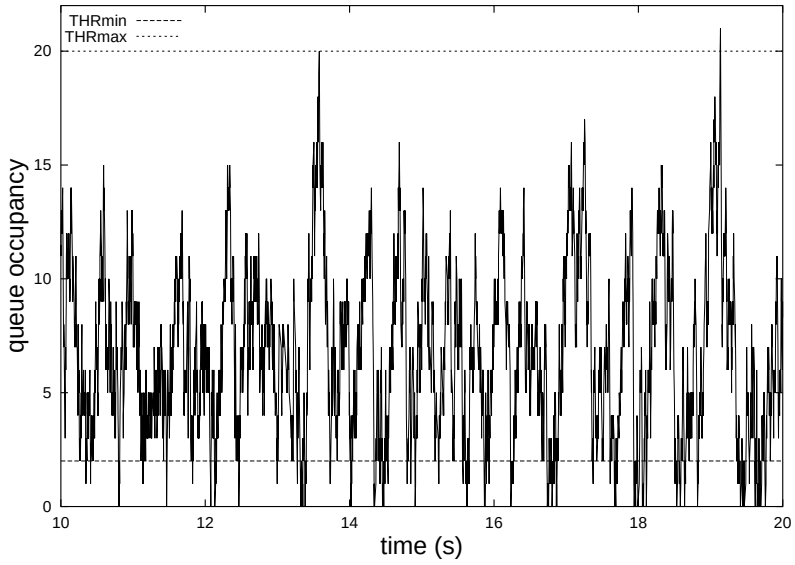

Figure 3.8: Instantaneous buffer occupancy with $N = 6$

In general, experimental results show that instability predicted by the model in [83] leads to reduced link utilization and queue oscillation and higher normalized oscillations (higher jitter in percentage).

Hence, while one could expect larger queue oscillations when the number of flows decreases and the stability margins decrease, this intuition is not confirmed by simulative results. The explanation is straightforward: RED couples queue length and loss probability, in particular a lower number of flows needs a lower global dropping probability, hence a lower average queue (from a control theoretic point of view one says that the RED controller has steady state regulation errors). As the average queue size decreases, the physical constraint of positive queue values can determine smaller oscillations.

On the contrary the normalized standard deviation looks to reflect system change from stability to instability. So in what follows we focus on it to analyze instability phenomena.

3.5.2 Two Bottlenecks

Now we consider an additional aggregate entering the network from node 7, with destination node 8 (dotted line in Fig. 3.3). In Fig. 3.10 node 3 buffer appears to be empty more often than in Fig. 3.8.

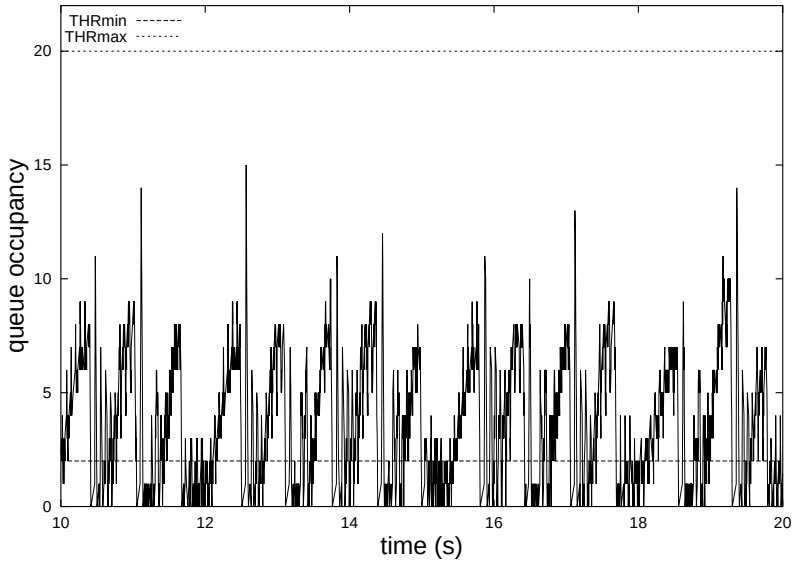

Figure 3.9: Instantaneous buffer occupancy with $N = 2$

Table 3.2: Numerical Results

N_5	N_5	N_7	Thr_5	Thr_4	Thr_7	$queue_3$ occupancy	$queue_3$ oscillation	$queue_1$ occupancy	$queue_1$ oscillation
5	5	0	5.12	4.79	-	9.06	4.72 (0.52)	0.95	0.066 (0.27)
4	4	0	5.09	4.75	-	7.93	4.48 (0.56)	0.95	0.057 (0.25)
3	3	0	5.12	4.62	-	6.71	4.09 (0.61)	0.95	0.067 (0.27)
2	2	0	5.06	4.49	-	5.32	3.57 (0.67)	0.97	0.075 (0.28)
1	1	0	4.80	4.47	-	3.49	2.64 (0.76)	0.97	0.085 (0.30)
5	5	5	3.40	6.41	6.33	7.76	5.29 (0.68)	7.83	5.47 (0.70)
4	4	4	3.43	6.28	6.20	6.85	4.98 (0.73)	6.93	5.13 (0.74)
3	3	3	3.62	5.99	5.91	5.76	4.38 (0.76)	5.94	4.59 (0.77)
2	2	2	3.80	5.60	5.52	4.72	4.01 (0.85)	4.75	4.06 (0.85)

The numerical values stored in Table 3.2 support quantitatively our claims rising from Fig. 3.10. In particular the normalized oscillation values of queue 3 is 0.76, equal to the value stored in the fifth row, corresponding to a single bottleneck high instability scenario due to a low number of flows ($N_5+N_4 = 2$).

Note that, though the number of flows at each node and the flow round trip time should assure stable operation, instability arises due to the traffic aggregate from 5 to 6, which traverses both the congested links.

Moreover results in Table 3.2 indicate that even stable scenarios (according to the two bottleneck model in Sec. 3.4) like that corresponding to $N_5 = N_4 = N_7 = 5$ cannot be easily distinguished by an instable scenario. This remark suggests that the study through linear models could provide not signifivative

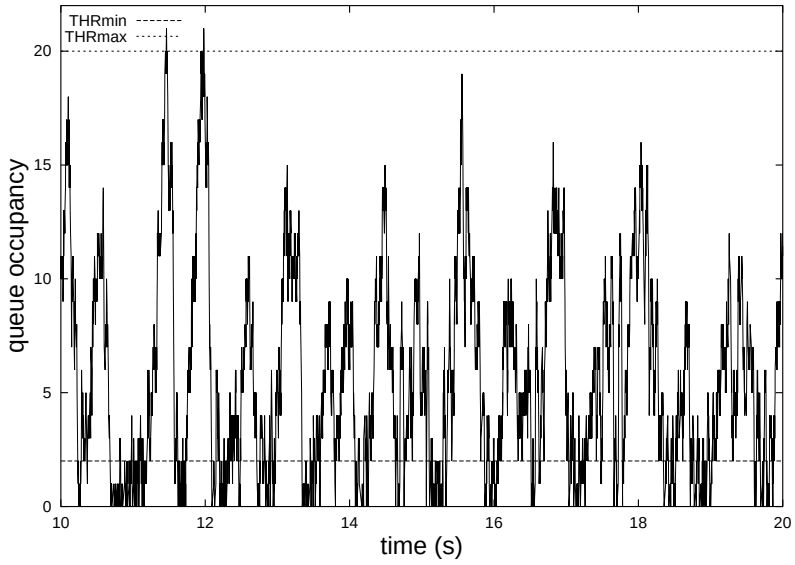

Figure 3.10: Instantaneous node 4 buffer occupancy in a two bottleneck scenario

insight on the stability of network with TCP traffic.

3.6 Conclusive Remarks

Our investigation showed that RED configuration based on a single-bottleneck assumption may not prevent from traffic instability when congestion occurs, at the same time, in two different locations of the network.

This suggests that the effect of multiple bottlenecks could be counteracted by robust configuration of AQM controllers. In particular our results in [118] suggest that the minimum number of flows N^- should not take into account flows being controlled by other nodes. Hence the network administrator should evaluate not only the minimum number of flows at each node and their round trip time, but he should also get more sophisticated information about traffic matrix across the network and contemporaneously congested nodes.

In a similar way some robust design techniques for multiple bottleneck scenarios have been developed recently [77], but they appear to be very conservative, unless information about traffic matrix and congested nodes is available.

Another approach would be to implement new cooperative AQM controllers, that base their control action on information about the congestion

status of the other nodes. Simplicity is an obvious requirement, particularly for signaling among nodes.

We think that ECN (Sec. 3.1.5) could be usefully employed for internodes signaling. ECN has been proposed as a light in-band signaling form between nodes and client, but it appears to be a simple way for nodes to transmit downstream information about their congestion status. The advantages of ECN employment are: no further network transmission resources are required, information travels along the data path, and can be used by all the nodes controlling the flow.

AQM controller should monitor the ingoing traffic, evaluate the share of traffic controlled elsewhere, by the percentage of packets with the Congestion Experienced codepoint set (CE packets) and set some tunable parameters according to the controlled traffic share. For example a RED controller could decrease the dropping curve slope L_{RED} as the percentage of CE packets increases in order to maintain a stable operation. Even if it is difficult to manage such kind of aggregated information some improvements could be achieved in comparison to a situation where each node is unaware of other nodes actions.

Chapter 4

A New Adaptive TCP Marker

In this chapter we propose a new marking algorithm. After an overview of marking schemes in Sec. 4.1, the algorithm is described through Sec. 4.2 and 4.3. Extensive performance evaluation is presented in Sec. 4.4. An analytical model (with two variants) is proposed in Sec. 4.5 together with its simulative validation. Appendix B provides a survey of Differentiated Services architecture.

4.1 Need for Adaptivity in TCP Marking

Several packet marking algorithms have been proposed to provide service differentiation among a set of TCP flows that share network resources. All packet marking mechanisms have a common basic approach. Packets of each individual flow are marked based on a suitably chosen profile at an edge router. Then, marked packets are aggregated in the network, and receive a different treatment in the network core routers.

Generally, a two-level marking scheme is adopted, where packets labelled as IN receive better treatment (lower dropping rate) than packets marked as OUT. Within the network, dropping priority mechanisms are implemented in active queue management schemes such as RIO - Random Early Discard with IN/OUT packets (Sec. 3.1.3).

The basic idea of proposed algorithms is that a suitable marking profile (e.g. a token bucket which marks IN/OUT profile packets) may provide some form of protection in the case of congestion. The purpose is to give the customer the assurance of a minimum throughput, even during periods of congestion, while allowing him to consume more bandwidth when the network load is low.

Thus a connection using the assured service should achieve a throughput equal to the subscribed minimum rate, also called target rate, plus some share of the remaining bandwidth gained by competing with all the active best-effort connections.

A large number of papers [42, 88, 150, 166, 78, 147, 56, 40, 164, 68, 51, 98, 163] have thoroughly studied marking mechanisms for service differentiation, and have evaluated how the service marking parameters influence the achieved rate.

This approach was first introduced in [42], where the authors propose a time-sliding window marker. In [88] token bucket appears to achieve better performance in comparison to time-sliding window. According to the authors a correctly configured token bucket will allow for the natural burstiness of TCP by marking as IN all the packets within a burst, while with an average rate estimator some packets will be marked OUT giving them drop preference. At the same time the authors claim that marking cannot offer a quantifiable service to TCP traffic due to the interaction of TCP dynamics with priority dropping: when IN packets and OUT packets are mixed in a single TCP connection, drops of OUT packets negatively impact the connection's performance. Afterwards token bucket and time-sliding window markers have been extended to three colors [80, 81, 54].

Following studies confirm the difficulty of marker configuration. A detailed experimental study of the main factors that impact the throughput of TCP flows in a RIO based DiffServ network is provided in [150]. The article shows that in an over-provisioned network all target rates are achieved, but unfair shares of excess bandwidth are obtained. However, as the network approaches an under-provisioned state, not all target rates can be achieved. In [166] it is shown that it is possible to improve the throughput significantly even when a small portion of traffic is sent as in-profile packets. At the same time the authors observe that, in order to fully utilize the benefit of out-profile packets, the amount of out-profile packets sent in addition to the in-profile packets has to be carefully determined. In [78] a set of experimental measures is presented. The main result is that the differentiation among the transmission rates of TCP flows can be achieved, but it is difficult to provide the required rates with a good approximation. In [147] the limits of token bucket are deeply investigated. It appears that (i) the achieved rate is not proportional to the

assured rate, (ii) it is not always possible to achieve the assured rate and, (iii) there exist ranges of values of the achieved rate for which token bucket parameters have no influence.

These results suggested the need to introduce some adaptivity in order to cope with TCP dynamics. In [56] the Packet Marking Engine (PME) monitors and sustains the requested level of service by setting the DS-field in the packet headers appropriately. By default, all packets are generated as low priority packets. PME is idle, if the observed service rate at the low priority level either meets or exceeds the requested service rate. If the observed throughput falls below the minimum target rate the PME starts prioritizing packets until the desired target rate is reached. Once the target is reached, it strives to reduce the number of priority packets without falling below the minimum requested rate. The Active Rate Management (ARM) is proposed in [40] in order to provide minimum throughputs to traffic aggregates. It is a classical, linear, time-invariant controller, which sets the token bucket parameters (specifically the token bucket rate) adapting to changes in the network. The same issue is tackled by [164]. The adaptive dual token bucket in [68] regulates the amount of OUT packets in order to prevent TCP packet losses caused by excess low-priority traffic in the network. This adaptive technique requires a congestion signaling procedure from internal routers to border routers. When the RIO buffer is not congested, the token bucket filter can increase the percentage of out-of-profile traffic in order to exploit the bandwidth available on the output line. Conversely, when the RIO buffer is congested, the token bucket filter must reduce this percentage.

The *Equation-Based Marking* (EBM) [51] is someway similar to ours because it senses the current network conditions, in particular it estimates the loss probability and the RTT experienced by a TCP flow (without any signaling with core routers), and adapts the packet marking probabilities accordingly. In particular it uses the TCP model in [136] and these estimates in order to identify the target loss probabilities, corresponding to target throughput rates. Then, it uses the current loss probability estimate as well as these target loss probabilities to calculate the packet-marking probabilities. Main targets are fairness among heterogeneous TCP flows and protection against non-assured traffic. Fairness is also the main focus of [98] and [163]: the first concentrates

on the effect of different RTTs, the second propose the Direct Congestion Control Scheme (DCCS) to achieve fairness between responsive and unresponsive aggregate.

The proposals described above share the purpose to assure a minimum throughput to TCP individual flows or aggregates. More recently, TCP marking has been proposed as a way to achieve better than best effort performance [15, 116, 110]. The idea is that packet marking can be adopted also in a scenario of homogeneous flows (i.e. all marked according to the same profile), with the goal of increasing the performance of all flows. In particular [110] focuses on WWW traffic and proposes a new scheme able to reduce the completion time of a http session. This scheme is described more in detail in Sec. 4.4.3. The TCP-friendly marker in [15, 116] considers long lived flows and adopt goodput and loss as performance metrics. The main guidelines are: 1) to protect small-window flows and retransmitted packets from losses by marking them IN; 2) to avoid, if possible, to mark OUT consecutive packets in order to reduce the possibility of burst loss of packets. Our approach share the purpose to space as much as possible packet losses, at the same time many differences hold. In the TCP-friendly marker a fixed number of IN tokens is available for each time interval and it has to be distributed among the flows, on the contrary our scheme adaptively set the length of IN packets burst (i.e. the number of flow consecutive packets that are marked IN) according to the network status. Besides, all the marking schemes share the idea that packets marked IN will be protected against network congestion, while our algorithm operates according to the someway opposite philosophy to employ OUT packets as *probes* (see Sec. 4.2). Finally, our approach is much simpler.

4.2 Rationale behind our Marking strategy

The fundamental difference of our algorithm with respect to the previously mentioned marking strategies is that our algorithm marks most of the packets as IN, and interleaves IN packets with occasional OUT packets. The length of an IN-packets burst is adaptively set based on an heuristic estimation of the experienced packet loss ratio.

Since the large majority of packets in the network are of type IN, by marking a packet as OUT we dramatically increase the probability that this packet is dropped. In essence, the role of the OUT packet is that of a *probe*, whose goal is to early discover whether the network is getting congested.

We remark that Random Early Discarding (RED) techniques have been designed with the same philosophy in mind. By randomly dropping packets when the queue size increases above a given threshold, RED provides early feedbacks to the TCP congestion control mechanism running at the network edge. However, RED provides only a "loose" form of control mechanism; dropped packets may belong to a subset of offered flows (which consequently reduce the emission rate), while other TCP flows may not experience packet dropping and thus remain unaware of the congestion situation. Moreover, in the same time, a few TCP flows may even experience multiple packet dropping and therefore be severely penalized.

Conversely, our proposed marking strategy provides a stronger form of control mechanism, as the packets that are likely to experience dropping are not randomly chosen among all the offered packets, but are the ones specifically marked as OUT. This operation allows to smoothly and fairly drive the TCP congestion control operation and makes the traffic offered to the network more regular, so better performance are achieved. Moreover, as we showed in [125] the higher the dropping rate of OUT packets versus the IN dropping rate, the better the performance gain is. Ideally, the optimal operational condition in the network should be that of a 100% loss rate of the OUT packets but still no loss encountered by IN packets.

When compared with the schemes proposed in [15, 116, 110], our marking strategy presents two major differences. First, the majority of packets are IN. Second, the performance takes advantage of a very high OUT packet loss rate. The fact that our strategy performs well is apparently in conflict with some results presented in [110], which show that interleaving IN and OUT packets may have a highly negative impact on the TCP throughput, if the loss rate of the OUT traffic is much larger than that of the IN traffic. In particular, a throughput reduction may be encountered as long as the percentage of IN traffic becomes greater than a given threshold. Indeed, we too have observed performance impairments for both a token-bucket marker and for a marking scheme very similar to the one proposed in [15, 116] (protection

of small window and retransmitted packets, an OUT packet inserted every n IN packets). However, we remark that these schemes are not designed to be adaptive to the network congestion status, while ours uses some heuristics to provide adaptability.

4.3 The Marking Algorithm

A first version of the algorithm was presented in [125, 127]. A new version has been presented in [121, 129]. Here we describe the most recent version and conclude this section with some remarks about the first one.

The packet marking algorithm we proposed can be implemented at the ingress router. It acts on a per-flow basis. When the ingress router detects a TCP SYN packet, meaning that a new flow is offered, it reserves a state for the flow. This state is composed of the following variables:

- SN_h: This counter stores the highest Sequence Number (SN) encountered in the flow. It is initially set to the ISN (Initial Sequence Number) value. It is updated whenever a non-empty packet (i.e. non ACK) with a higher SN[1] arrives at the router.

- L_{seq}: It is initially set to zero. It is increased by one unit for each new arrived packet (i.e. in-sequence packet), while is reset to zero every time an out-of-sequence packet arrives.

- A_{seq}: It stores the average length of in-sequence packet burst between two consecutive losses, using an auto-regressive filter on the previous values of L_{seq}. It is initially set equal to a design parameter A_0.

- C_{IN}: It counts the number of IN-packets in the burst. It is reset to zero when it exceeds A_{seq} and an OUT packet is sent.

Note that we distinguish between two different bursts of packets: the in-sequence packet one and the IN-marked packet one. An IN-marked packet burst is a group of packets which are marked IN at the ingress router. An

[1] in a cyclical sense - recall that sequence numbers wrap when the value $2^{32}-1$ is reached

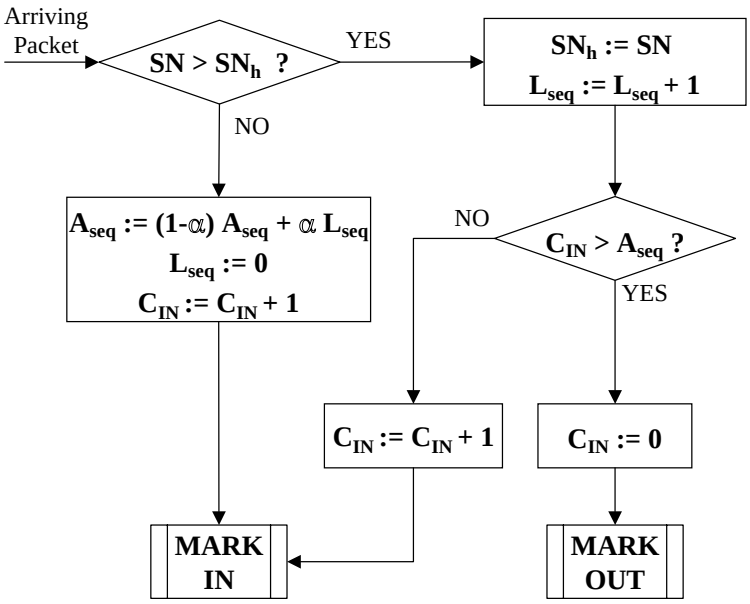

Figure 4.1: Packet Marking Algorithm

OUT-marked packet separates two consecutive IN-marked packet bursts. The variable C_{IN} stores the number of IN-packets in the current burst. An in-sequence packet burst is a group of packets which arrive at the ingress router with increasing SNs. Such group is bordered by two out-of-sequence packets (one at the begin, the other at the end of the burst). According to [87], we say that a packet is out-of-sequence if its sequence number is less than that of a previously observed sequence number in that connection. The variable L_{seq} stores the instantaneous length of the in-sequence burst, while the variable A_{seq} tracks the average value of such length.

Note that an out-of-sequence packet can be caused by three different events:

- Retransmission. In this case, a sender infers that a packet has been lost and retransmits the packet. The retransmitted packet will have a sequence number that is smaller than previously observed packets at the measurement point and hence will be deemed out-of-sequence.

- Network duplication. In this case, a non-sender retransmitted copy of a packet is observed. This can occur when the router is within a routing loop (and hence the same packet is observed more than once), or if the network itself creates a duplicate copy of the packet.

- In network-reordering. In this case, the network inverts the order of two packets in a connection (for example, because of parallelism within a router [1] or a route change).

Network duplication and reordering should be rarer events in comparison to retransmissions, hence the algorithm assumes that all the out-of-sequence events correspond to retransmission events[2]. Under such assumption the A_{seq} is a good estimate of the average number of packets transmitted between two consecutive losses. The algorithm aims to pace the TCP flow adaptation by marking a packet OUT after a burst of A_{seq} IN packets. For this reason in Sec. 4.2 we said that the length of an IN-packet burst is adaptively set based on an heuristic estimation of the experienced packet loss ratio.

Hence this marking algorithm enforces TCP source adaptation through OUT-packets higher sensibility to network congestion, but it does not change the basic TCP adaptation law to the network congestion. In fact the algorithm estimates per-flow losses through information implicitly conveyed by TCP packets, and it simply tries to space the losses as far as possible. Performance improvements are expected, because TCP adaptation should be more regular and the traffic variability is reduced. Performance evaluation in Sec. 4.4 confirms such intuition.

The details of the algorithm are in the flow-chart in Fig. 4.1. When a non-empty packet arrives at the router, its sequence number SN is read. According to the new SN value, and the recorded highest sequence number encountered before, we face two possible situations. If $SN \leq SN_h$, then the incoming packet is a replica of a previously transmitted packet. This means that such packet has probably been lost. Conversely, if $SN > SN_h$ the incoming packet is a new one.

Our algorithm distinguishes these two cases. In the case of packet loss, the value A_{seq} is updated as the weighted sum of the previous estimate with the current value of L_{seq}. The L_{seq} value is then reset to 0. The out-of-sequence packet is delivered marked as IN, hence the C_{IN} counter is increased by one. In the case of a new incoming packet the current in-sequence packet burst size is increased by one. The packet is then marked as IN if the IN-packet

[2]More details on reordering are in Appendix C, while the some results on the effect of reordering on the performance of the algorithm are shown in Sec. 4.4.6

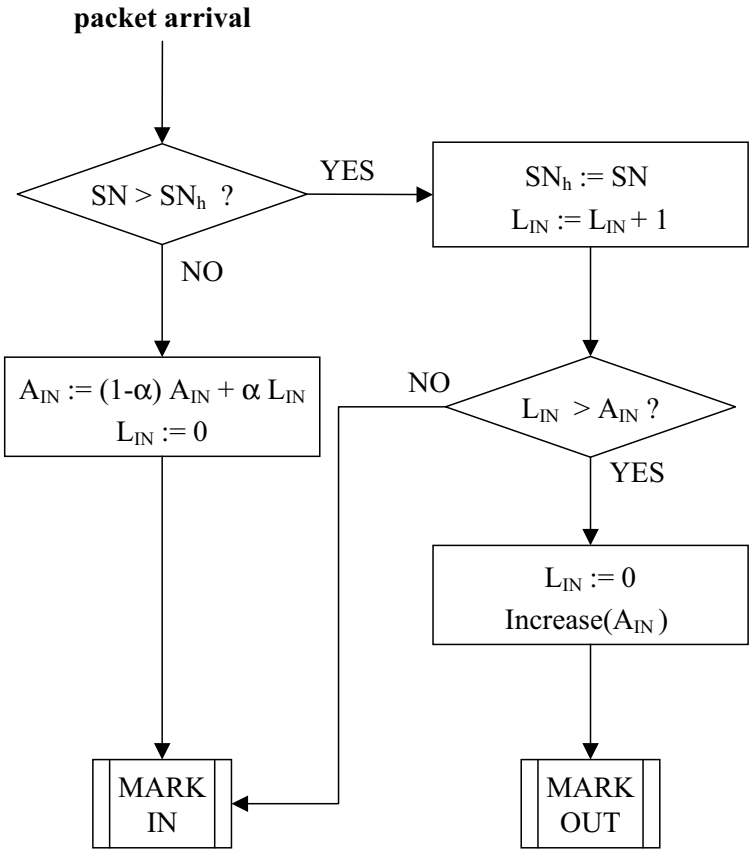

Figure 4.2: The first version of the Packet Marking Algorithm

counter C_{IN} is shorter than the value A_{seq}. Conversely, if the actual burst of IN-marked packets has become longer than A_{seq}, the actual packet is marked as OUT, and a new burst begins ($C_{IN} = 0$).

As regards the configuration parameters α and A_0, unless otherwise specified, we have adopted $\alpha = 0.5$, $A_0 = 7$. These parameters have not been chosen after an optimization procedure, nevertheless the performance of the algorithm are very good.

The first version. As we said, the above algorithm is an improvement of the original version presented in [125, 127] and shown in Fig. 4.2. The previous algorithm was simpler, because a single variable (L_{IN}) was taking into account the number of in-sequence packets (as L_{seq} actually does) and the number

of IN packets of the actual IN-packets burst (as C_{IN} actually does). This coupling required an artificial increase of the variable A_{IN} (the average value of L_{IN}) after marking an OUT packet. In Fig. 4.2, this operation is generically indicated as increase(A_{IN}). In fact, when congestion conditions occur, several packet losses may be encountered, and thus the value A_{IN} decreased (left part of Fig. 4.2). In [125, 127] we chose $A_{IN} := 2A_{IN} + 1$ but its correct amount was dependent from network condition.

To better understand how this increment should be quantitatively accounted, consider the situation in which all packets labelled as IN are successfully received, while all packets labelled as OUT are discarded. This means that the congestion level in the network has reached a given stationary target value. To remain in such stationary conditions, the OUT marking rate should not vary with time, i.e. an OUT packet should be marked every $\bar{A}_{IN}$ IN packets, being $\bar{A}_{IN}$ a constant[3] In the assumption of stop&wait TCP operation[4], no IN packet loss, and 100% OUT packet loss, it is easy to see that A_{IN} remains constant to an initial value $\bar{A}_{IN}$ if the increase rule is $A_{IN} := A_{IN}/(1 - \alpha)$.

4.4 Performance Evaluation

4.4.1 The Simulation Scenario

Usually we considered a simple dumbbell network topology like that in Fig. 4.3, where there is a single bottleneck link. Unless otherwise specified the numerical values for link capacities and propagation delays, are those in Fig. 4.3. The capacity of the bottleneck is set equal to 6 Mbps and the RTTs of the different TCP flow paths are different (from 124 ms to 198 ms, the average value is 160 ms). The different RTTs together with random start of source transmissions avoid phase effects, i.e. misleading simulation results due artificial sources synchronization.

Each router deploys RIO (Sec. 3.1.3) as Active Queue Management. As

[3]It depends (in a non trivial manner) on the RIO configuration at the bottleneck link and on the number of offered flows.

[4]For general values of the contention window, such an analysis is much more complex as it further depends on how many packets have been sent when a triplicate ACK arrives at the sender.

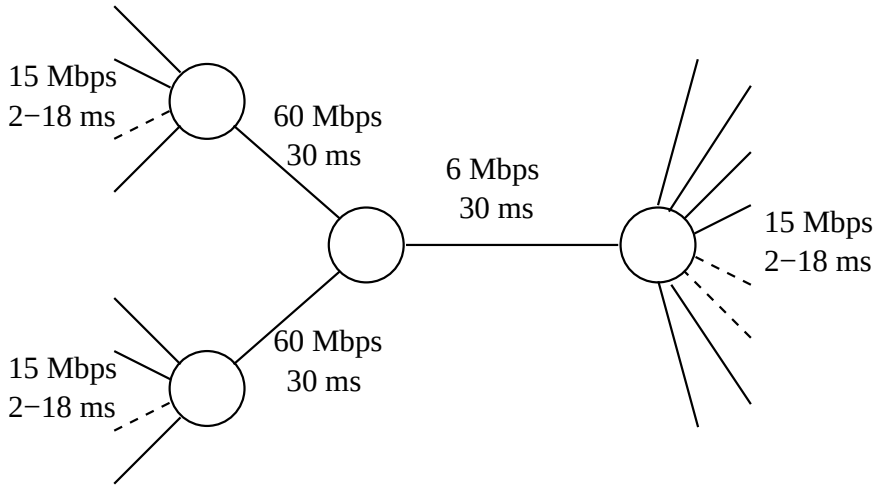

Figure 4.3: Network topology

regards RIO configuration, the thresholds and $P_{max_{in}}$ are chosen according to [63], the filter coefficient w_q according to [64], i.e. $max = 3min$, $P_{max_{in}} = 0.1$ and $w_q = 1 - exp(-M/(C * 10 * RTT)) = 0.0012$, where C is the link capacity, M is the packet size and RTT is the Round Trip Time.

RIO configuration allows the network provider to trade off link utilization and delay performance: the higher the RED thresholds, the higher link utilization and delay. For this reason different settings were considered, and the results are presented through "performance frontiers". Lastly queue physical lengths were chosen so that packet losses occurred only in the core router, due to RIO (not to physical queue overflow).

Simulations were conducted through Network Simulator (NS) [130], which is nearly a "standard de facto" for IP networks simulations. In particular we used two different releases of NS: the v2.1b8 and the v2.1b9a. Unless otherwise specified we used TCP Reno implementation.

4.4.2 Long-Lived Flows

The expression long-lived flow refer to a TCP flow whose duration is some order of magnitude greater than its RTT. It corresponds to the transfer of big amount of data. In this situation TCP congestion control and congestion avoidance

play a fundamental role in order to determine user performance. Usually long-lived flows are modelled by infinite load sources, which have always data to transmit. Each source is able to employ all the available capacity along the flow path, for this reason these flows are also referred as greedy ones. When long-lived flows are considered the throughput is determined only by the network conditions and TCP dynamics.

The results in this section appeared in [125].

We considered two different load conditions with 10 and 40 TCP long-lived flows. Each source starts to transmit randomly in the interval 0-1 s, in order to avoid synchronization (Sec. 4.4.1).

Different RIO settings were considered. As regards the IN traffic, the min_{in} threshold values goes from 2 to 80 packets. As regards the OUT traffic we considered two different scenarios: in the first the OUT traffic settings vary according to IN traffic parameters, $max_{out} = 3min_{out} = min_{in}$ and $P_{max_{out}} = 0.2$, in the second they are fixed to $min_{out} = 2$, $max_{out} = 6$ and $P_{max_{out}} = 0.2$. In what follows we refer to this two settings respectively as *soft differentiation* and *hard differentiation*.

As regards the IN traffic, the min_{in} threshold values goes from 2 to 80 packets. As regards the OUT traffic we considered two different scenarios: in the first the OUT traffic settings vary according to IN traffic parameters, $max_{out} = 3min_{out} = min_{in}$ and $P_{max_{out}} = 0.2$, in the second they are fixed to $min_{out} = 2$, $max_{out} = 6$ and $P_{max_{out}} = 0.2$. In what follows we refer to this two settings respectively as *soft differentiation* and *hard differentiation*.

We compared the proposed marker with a no-marker situation, where all the packets are treated as IN packet. For each of the threshold setting we evaluated link utilization (goodput) and average delay, and plotted them as "performance frontiers".

For each configuration at least 5 simulations with different random seeds were run. Each simulation lasted 1000 simulated seconds, statistics were collected after 50 seconds. In the figures we present in the following section, standard deviation of goodput and average delay is always less than 1% of their numerical value. These simulations were conducted through ns v2.1b8.

Figures 4.4 and 4.5 show the performance frontiers respectively for 10 flows and 40 flows in soft differentiation. In figure 4.4 RIO threshold settings are reported for three points in the form (min_{out},max_{out})

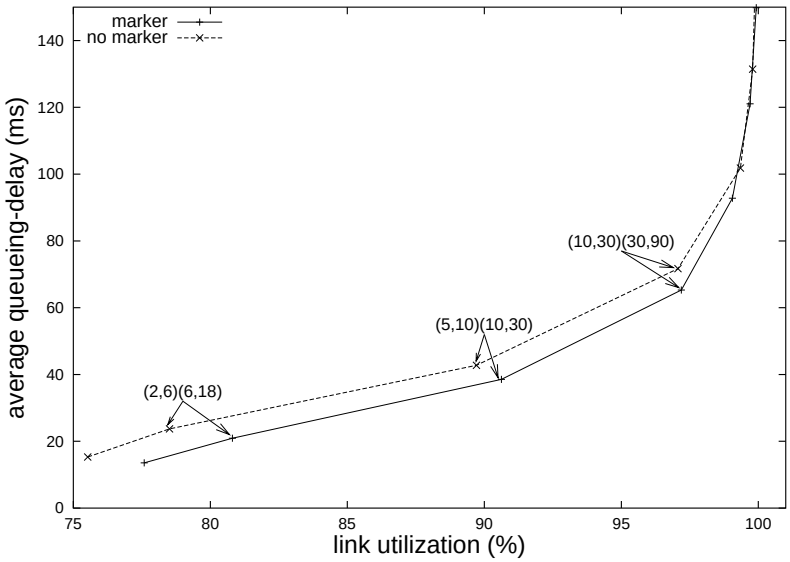

Figure 4.4: Delay vs link utilization - 10 flows, soft differentiation

(min_{in},max_{in}). Performance improvement provided by the marker employment is remarkable under high load condition.

The improvement is more significant when IN and OUT packets receive more different services from the network, as one can see in figures 4.6 and 4.7.

As regards the number of packet marked IN by the algorithm, it increases as thresholds are higher and link utilization increases. For both soft and hard differentiation IN packet percentage varies from about 98% to more than 99% for the tested configuration with 10 sources and from about 94% to 97% with 40 sources (losses increase with the number of flows). Figure 4.8 shows global, IN and OUT loss percentage for 10 flows. We see that for high goodput values in hard differentiation OUT loss percentage is near 100% while IN loss percentage is very small: source behavior becomes almost "deterministic", the variance of the offered load is highly reduced so performance are significantly improved.

4.4.3 Short-Lived Flows

In the previous section we considered long-lived TCP flows, so it could appear questionable whether our scheme is able to improve performance in a more

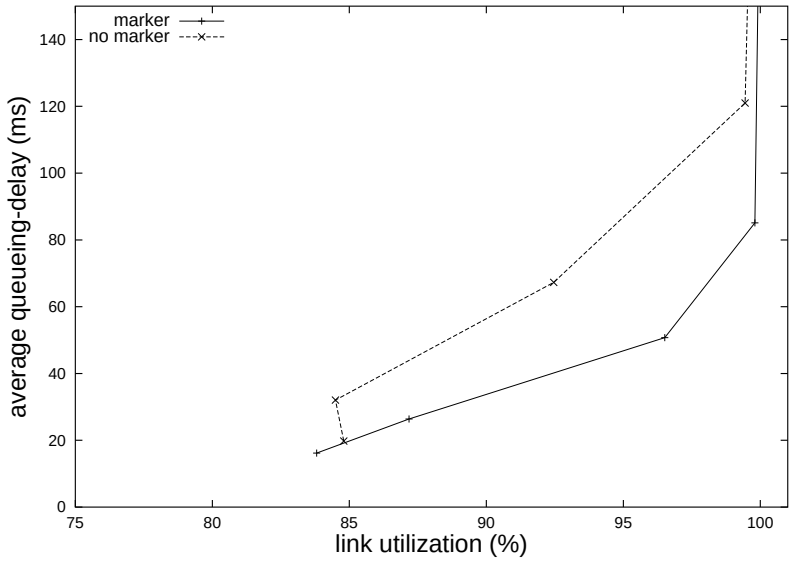

Figure 4.5: Delay vs link utilization - 40 flows, soft differentiation

realistic scenario, where very short flows do not allow the algorithm to reach a *steady state*. Actually short flows are a significant part of Internet traffic because of Web traffic (usually html pages are few tens of kBytes).

For this reason we here extend the performance evaluation to a more realistic scenario, coming from a Internet measurement campaign [109]. We compare our algorithm with a best effort scenario but also with another algorithm explicitly developed in order to reduce the completion time for short-lived http flows [110]. In what follows we refer to this scheme as Web Packet Marking (WPM).

This section is organized as follows. Firstly the Web Packet Marking (WPM) strategy proposed in [110] is briefly presented. Secondly, the performance evaluation of the APM and the comparison with WPM is carried out.

The results in this section appeared in [127].

Web Packet Marking

As our algorithm, WPM can be implemented at the ingress router and act on a per-flow basis. In particular in [110] the authors present two packet

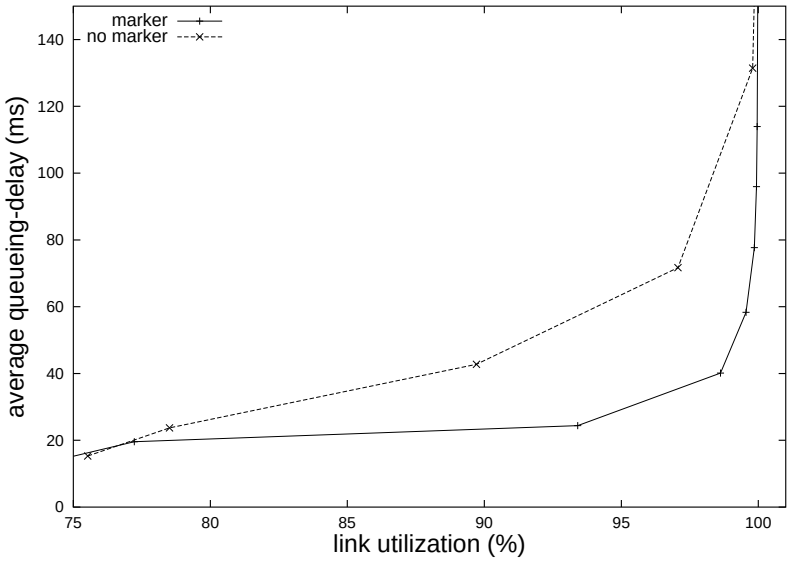

Figure 4.6: Delay vs link utilization - 10 flows, hard differentiation

marking schemes. The first one is tightly integrated with the TCP protocol: the source is allowed to send up to N_s IN packets when it starts, and then up to $N_a = sstresh$ at the beginning of a Slow Start phase, and up to $N_a = cwnd$ at the beginning of a Fast Recovery phase. The second scheme does not require the knowledge of internal TCP variables, but it uses a constant value $N_a = N_s = 5$, hence this scheme can be implemented at ingress router. In what follows we compare the second scheme with our adaptive scheme. The rationale behind WPM is that packets marked as IN will be protected against network congestion, hence marking can be useful employed to protect flows with small window, when packet losses cannot be recovered via the fast retransmission algorithm. In fact when the window is small, a loss trigger a timeout, which reduce TCP source throughput.

We present some remarks about this rationale in order to stress the novelty of our approach. In a DiffServ Assured Forwarding (AF) scenario, the differentiation between traffic classes is relative. For example RIO configuration described in Sec.4.4.1 assures that IN packets dropping probability is lower than OUT packets one, but no bound is guaranteed. For this reason the protection of IN packets in WPM relies on the assumption that most of the packets in the network are of type OUT, hence IN packets will receive a “good-enough”

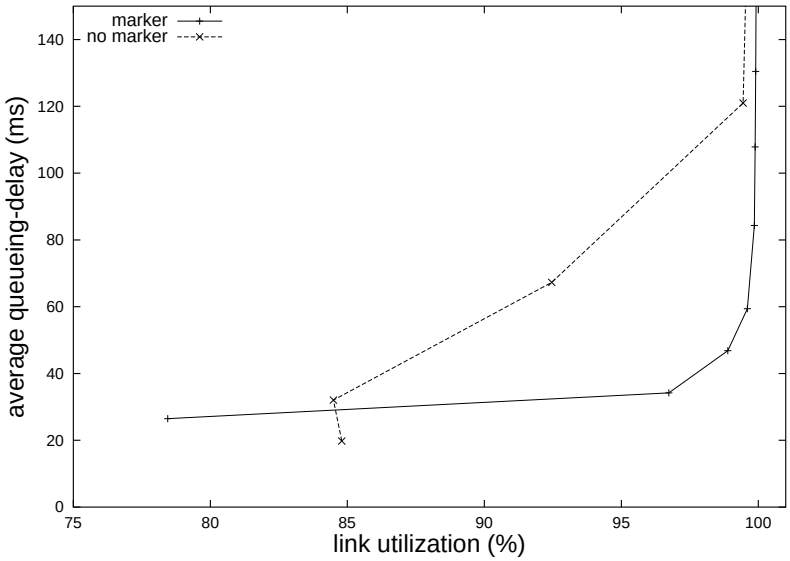

Figure 4.7: Delay vs link utilization - 40 flows, hard differentiation

service. In fact in [110] the authors show that a throughput reduction may be encountered as long as the percentage of IN traffic becomes greater than a given threshold. The authors claim that the problem is interleaving IN and OUT packets, when the loss rate of the OUT traffic is much larger than that of the IN traffic. We want to stress that the IN packet protection vanishes as IN traffic increases. Indeed, we too have observed performance impairments for both a token-bucket marker and for a marking scheme very similar to the one proposed in [15, 116] (protection of small window and retransmitted packets, an OUT packet inserted every n IN packets.

Hence our approach shows two main differences (Sec. 4.4.2): 1) the majority of packets are IN, 2) the APM performance takes advantage of a very high OUT packet loss rate. The apparent conflict with results in [110] and with similar results for the marker proposed in [15, 116] relies on the adaptivity. These schemes are not designed to be adaptive to the network congestion status, while ours uses some heuristics to provide adaptivity.

In this section we show performance evaluation results when also short-lived flows are present in the network. To simulate WWW-like traffic, a number of TCP-Reno sources are connected to each ingress router in the topology of Fig. 4.3. The flow's arrival rate is modeled as a Poisson process and the flow

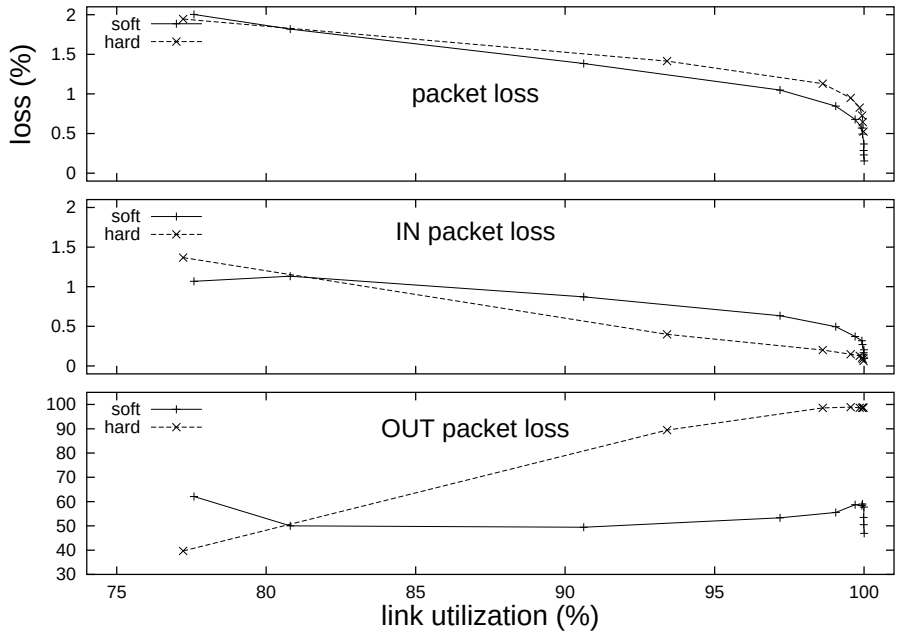

Figure 4.8: Loss percentage - 10 flows, soft and hard differentiation

lengths are drawn from the distribution given in table 4.1. This distribution has been constructed from data collected at the end of 2002 through the Tstat tool [109] on the Internet access link of Politecnico di Torino, i.e. between the border router of Politecnico and the ingress router of GARR, the Italian research network. Collected flow length data have been ordered from the shortest to the longest and divided in 15 groups, each corresponding to a 0.066% probability. Table 4.1 reports, for each group, the average flow length measured both in bytes and in IP packets (for simplicity, we have considered 1500 bytes packets).

In our simulations we have considered two different WWW offered loads, corresponding to 64% and 90% of the bottleneck link capacity. Since results are very similar, in what follows only 64% load results are presented. One continuous backlogged TCP flow has been added.

Different kind of RIO settings have been considered for APM, WPM and the "no marker" situation (NM in what follows). In Sec.4.4.2 we showed that APM performance is better as the service differentiation among OUT and IN packets increases. So here we let the min_{in} threshold values going from 9 to 240

Table 4.1: Flow lenght distribution

Group	Bytes	Packets	Group	Bytes	Packets
1	119	1	9	1650	2
2	179	1	10	2861	2
3	251	1	11	4706	4
4	334	1	12	8015	6
5	428	1	13	13681	10
6	529	1	14	26641	18
7	658	1	15	284454	190
8	948	1			

packets, while the OUT traffic settings are fixed to $min_{out} = 2$, $max_{out} = 6$ and $P_{max_{out}} = 0.2$. For the WPM algorithm, where the majority of the packets are marked OUT, we let the min_{out} threshold values going from 9 to 240 packets, while the min_{in} threshold is so high that no IN packets are dropped. When no marker is applied, all the packets are considered IN and RIO configuration is the APM one.

We considered two main performance figures: the average packet delay and the average flow completion time, i.e. the time from the emission of the SYN packet to the reception of the last data packet (no connection closing has been simulated). These values are plotted versus the average goodput (at the application level) for each of the threshold setting, so we can obtain the "performance frontiers".

Fig. 4.9 shows delay vs goodput. APM does not act on very short flows (less than 7 packets), so the remarkable advantage in figure 4.9 is achieved by controlling the longest flows (groups 13, 14, 15 and the long-lived TCP flow), whose throughput is regulated by the insertion of OUT probe packets. Instead, we note that the performance of WPM are very similar to that experienced in the case of no marking (NM) algorithm employed (all packets marked as IN). This is expected, as WPM is not specifically designed to improve goodput/delay performance, but it is designed to reduce the completion time for short flows.

Completion time results are shown in figures 4.10, for the three cases of i) single packet flows; ii) 18 packet flows, and iii) 190 packet flows. Let us first focus on the cases of 1 and 18 packet flows. As expected, WPM is effective (and

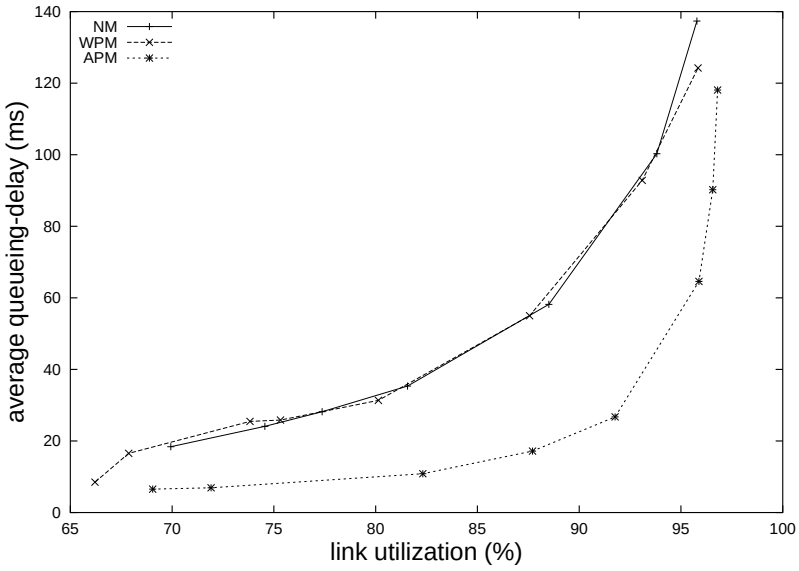

Figure 4.9: Delay vs goodput

slightly better than APM) when the link utilization is low, essentially because it protects the first packets in the flow, whose loss can be recovered only via retransmission timeout expiration (and not via fast retransmit). At high utilization, this "protection" effect reduces, as packet loss percentage decreases (results in table 4.2), and queueing delay becomes the main contribute to completion time. For this reason APM provides results consistently better than NM and WPM, because it is the only marking strategy that allows to keep the queue occupancy low in high utilization conditions [5].

[5]We remark that these insights on the performance of the considered algorithms were made possible only by the choice of presenting results in terms of performance frontiers rather than selecting a specific RED configuration. Indeed, several contrasting results presented in the TCP literature are motivated by different behaviors in different operational conditions - selecting a single RED configuration allows to achieve performance for just a single operational condition.

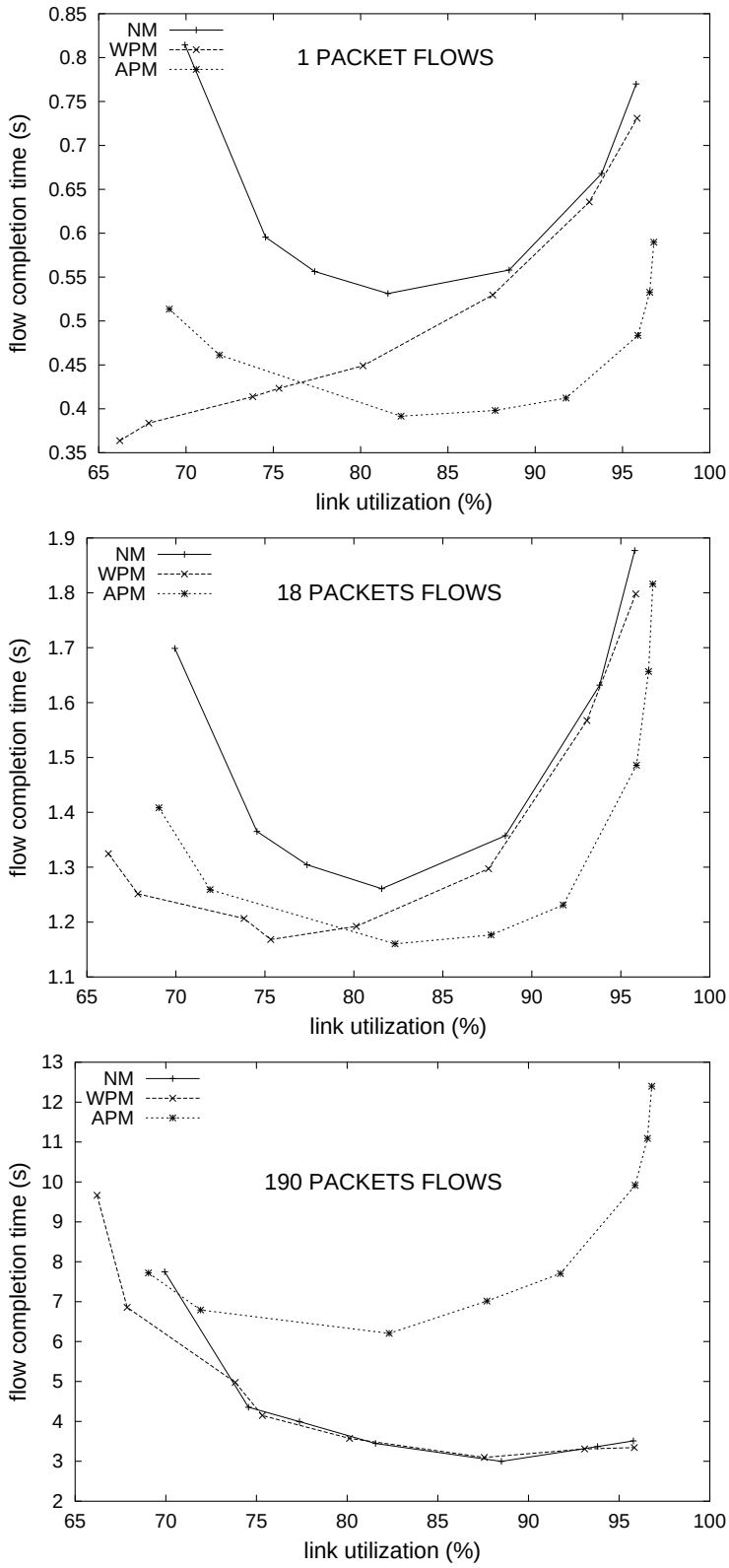

Figure 4.10: Flow completion time vs goodput

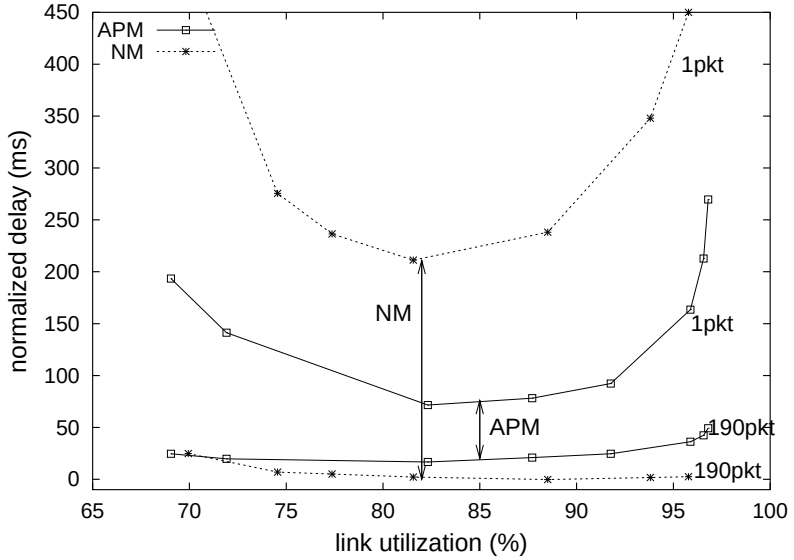

Figure 4.11: Normalized per-packet delay

The completion time results for the case of 190 packet flows (third plot of figure 4.10) shows that, in high utilization conditions, APM "pays" the improved completion time for short-lived flows with a significantly higher completion time for long-lived flows. The reason is that, in high utilization conditions, long-lived flows are given sufficient time to "adapt" (i.e. increase the number of OUT-marked packets) to the congestion situation, while short-lived flows behave in a more aggressive manner.

Despite this significant worsening in comparison to NM and WPM, we remark that this is not necessarily an impairment. To prove this point, figure 4.11 compares the normalized per-packet delay of APM with that of NM, plotted versus the link utilization, for 1 packet flows and 190 packet flows. This normalized delay is evaluated as the difference between the average completion time and the minimum completion time (when the network is unloaded), divided by the total number of data packets. From figure 4.11 it appears that, with APM, this normalized delay is less sensitive to the network configuration (as it is the average per-packet delay in figure 4.9). More importantly, APM is able to reduce the variability for different flow lengths in comparison to the no-marker scenario. So APM control on longest flows can be seen as a way to reduce the common TCP unfairness between short and long flows, caused by the fact that long flows can rely on the Fast Retransmit and Fast Recovery

Table 4.2: IN packets and dropped packets percentage

	NM	WPM	APM
IN%	100	26-13	97-98
DROP% TOT	5.3-0.0	4.5-0.0	2.9-2.0
DROP% IN	5.3-0.0	0.0-0.0	1.1-0.0
DROP% OUT	–	6.2-0.0	64-98

algorithms and have better Round Trip Time estimates. This unfairness holds also for WPM traffic for high link utilization: WPM normalized-delay curves are very similar to NM ones.

Finally, the difference among NM, WPM and APM in terms of percentage of IN packets, and percentage of dropped packets appears evident from table 4.2, where their range of variation for the different RIO settings is shown (from lower thresholds to higher one).

4.4.4 A new three color marker

From previous results it appears clear that our scheme and WPM are in some way orthogonal, so we can think to merge them. In order to elaborate a new scheme we need three different kind of packets. In a DiffServ framework, the domain administrator can dedicate an AF class (Sec. B.5) with three different dropping level to marked TCP traffic, say i the class and AFi1, AFi2 and AFi3 the levels, ordered for increasing dropping probability in core routers. Vulnerable packets according to the WPM scheme can be marked as AFi1, probing packets according to APM can be marked as AFi3, while the majority of packets are marked as AFi2. Performance evaluation shows that this new scheme, called Merge Packet Marking (MPM) is able to sensibly improve APM marking behavior in low link utilization without any other drawbacks [126]. For example figure 4.12 shows the average completion time for 1 packet flow when WPM, APM and MPM are employed.

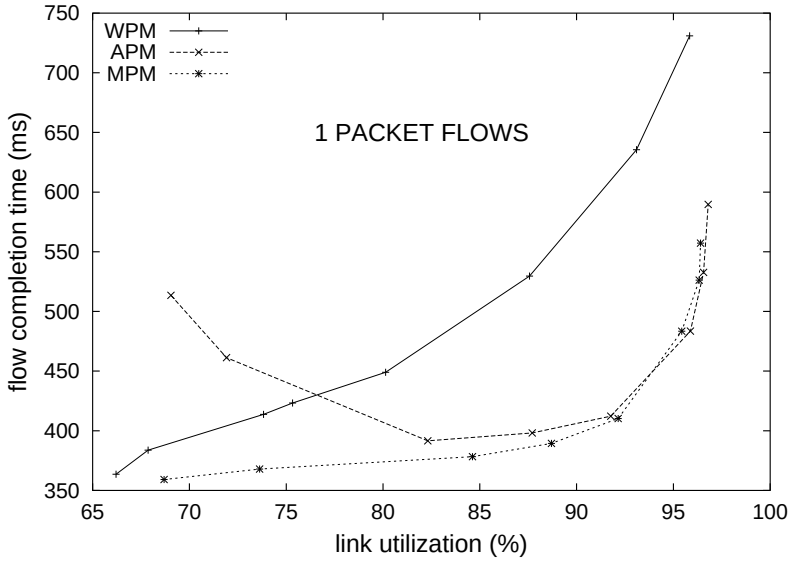

Figure 4.12: Flow completion time vs goodput

4.4.5 Reverse Traffic

It has been argued that reverse traffic can significantly impair TCP performance as this affects the flow of ACKs (acknowledgments) back to the TCP sources. The reason is threefold. Firstly there is a fundamental ambiguity in

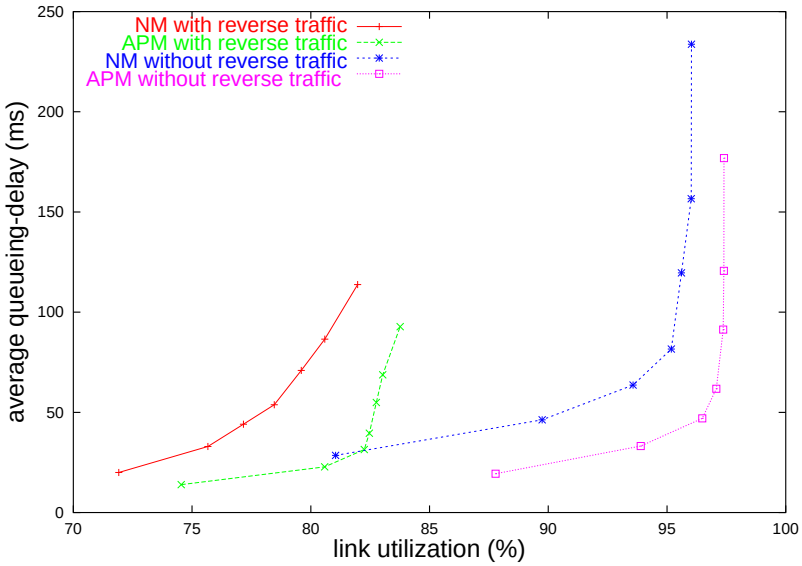

Figure 4.13: The effect of reverse traffic

the TCP operation: TCP mechanisms aiming to detect congestion are not able to distinguish between forward path and reverse path congestion. This can lead in significant underutilization of forward path resources when congestion arises on the reverse path. Secondly, when ACKs are queued behind other traffic for appreciable periods of time, the burst nature of TCP traffic and self-synchronizing effects can result in an effect known as ACK compression, which reduces the throughput of TCP. Specifically ACKs which are queued behind a burst of other packets become compressed in time. This results in an intense burst of data packets in the other direction, in response to the burst of compressed ACKs arriving at the server. It can become a self-reinforcing effect. Thirdly, an out-of-phase synchronization mode can occur. In the one-way traffic configuration, all of the connections are synchronized in–phase in that the flow control windows of the various connections all increase and decrease at the same time. With two-way traffic, under certain conditions the connections in different directions are synchronized out-of-phase in that the flow control window of one connection is increasing while that of the other is decreasing. This phenomenon has the effect of keeping the bottleneck utilization below optimal, even in the limit of infinite buffers. These effects becomes more significant when there is a high degree asymmetry between forward and reverse path. For further details on the effect of reverse traffic refer to [97, 19, 18, 168].

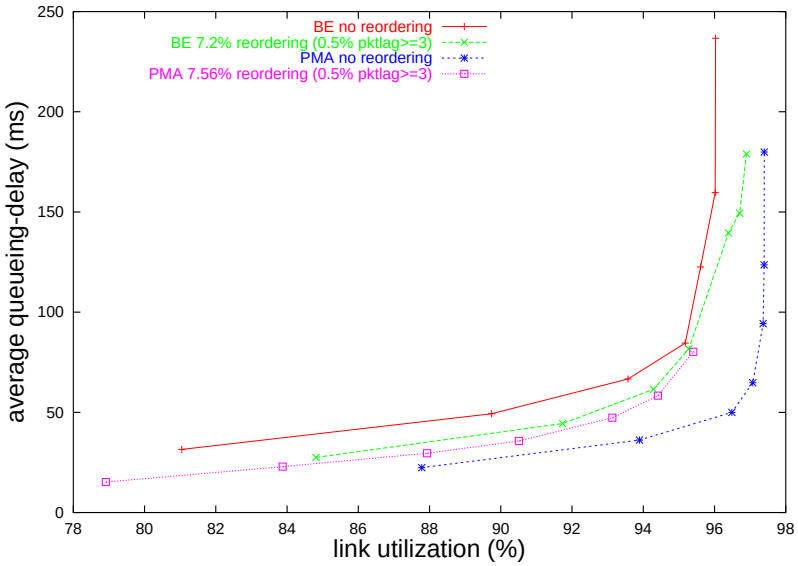

Figure 4.14: The effect of upstream reordering

Fig. 4.13 shows that reverse traffic produces a decrease of link utilization both in the no-marker scenario and in the APM scenario. Anyway also in this case APM produces significant benefits.

4.4.6 Packet Reordering

Packet reordering is a phenomenon which occurs when packets belonging to a same flow are received in a different order than the packet transmission one. Possible causes of reordering in IP networks and its effects on the TCP performance are presented in Appendix C. Here we are interested into reordering, mainly because if packet reordering occurs upstream the marker, the algorithm poorly estimates the number of packets between two consecutive losses. In fact reordered packets are misleadingly considered as retransmitted.

Fig. 4.14 shows that performance of APM worsen when reordering occurs. An odd result can be noted for the no-marker scenario: reordering appears to produce a performance improvements. An explanation of this effect and the description of the simulation scenario are in Appendix C. APM and NM performance frontiers almost overlap when the reordering probability is about 7%.

When reordering occurs downstream the marker likewise affects APM and

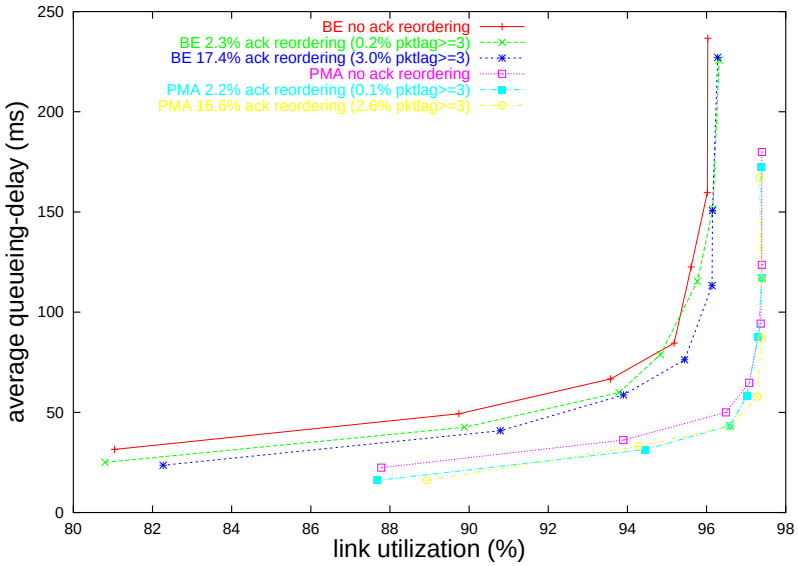

Figure 4.15: The effect of ack reordering

NM scenario. For example Fig. 4.15 shows analogous shifts of performance frontiers.

4.5 Analytical Models

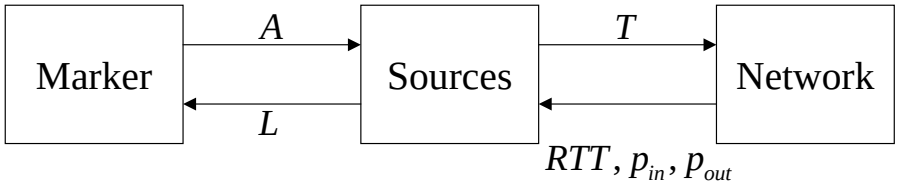

Figure 4.16: The three-block model.

In this section we are going to present an analytical model (with two variants) of PMA. The model is based on a Fixed Point Approximation (FPA), a modeling technique deeply described in the following subsection. According to FPA the system is divided into its three main components as shown in Fig. 4.16: the TCP sources, the network and the marker. Each element is modeled separately, taking into account the effects of the others through the parameters shown in figure. For example TCP sources depend on the network by the RTT and the dropping probabilities p_{in} and p_{out}, and on the marker by the average length (A) of an IN packet burst.

After an overview of FPA methods in Sec. 4.5.1, the submodels for the TCP sources, for the marker and for the network are respectively presented in sections 4.5.2, 4.5.3 and 4.5.4. In particular two different submodels for the network are introduced, and this leads to two variants for the global model. In Sec. 4.5.4 there are considerations about the existence and the uniqueness of the model equation systems. Finally Sec. 4.6 shows simulative results validating the models.

4.5.1 About Fixed Point Approximations

The expression *Fixed Point Approximation* (FPA) refers to a particular modeling technique, which we are going to describe in this section. This name is quite spread in scientific literature [34, 111, 95], but also other names appear: fixed point models [71, 146], fixed point approach [14], reciprocal model tuning [38]. Other papers [104, 114] refer the expression "fixed point" to the specific method employed to solve the model equation system, rather than to the modeling technique.

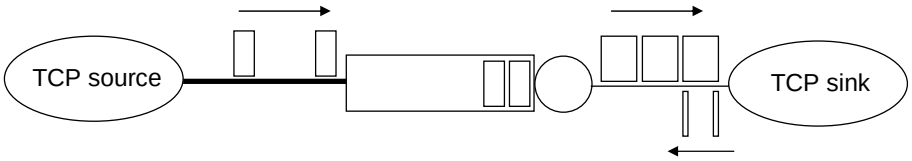

Figure 4.17: FPA single-bottleneck single-source example.

This section is organized as it follows. Firstly we introduce the idea of FPA and its main assumption by an example, and we explain the origin of the expression *fixed point*. Secondly we recall some mathematical results about fixed point theorems. After a look at other assumptions often employed in FPA, we present telecommunications works employing this kind of modeling technique.

FPA main assumption by an example. Let us assume we want to model a single long-lived TCP flow, which feeds a simple FIFO Drop Tail queue as in Fig. 4.17.

Suppose we are interested into some average values, like the TCP throughput or the queue occupancy. If we know all the parameters characterizing the network (i.e. link capacities, link delays, buffer size) and the TCP sender (like for example the TCP version, the maximum congestion window size, the timer granularity) and the TCP receiver, we are able to describe exactly the behavior of each element of the network and to evaluate the instantaneous throughput of the TCP sender or the instantaneous queue occupancy. If we would able to describe the evolution of these quantities in a closed form, we could evaluate their average value, by integrating the analytical expressions, but in general it is not the case.

In order to achieve our purpose we have to sacrifice the exact description of the system. A way to make the problem analytically tractable is to divide the system into two parts (e.g. the TCP source and the queue), to assume some simplifying assumptions about their interaction, and then to develop an analytical model for each part.

According to the FPA approach, the main assumptions are that we model each part considering the other in a steady state, and that this state is independent by the behavior of the part we are modeling. In our example we know

that the throughput of the TCP source is dependent from the network instantaneous RTT and from packet discard at the queue. At the same time the TCP traffic generates the queue in the network and causes eventually packet discard when the buffer is full. Nevertheless, in order to model the TCP behavior, we assume that the network is in a steady state: e.g. we consider that the RTT and the dropping probability (p) are constant, independently from the present TCP throughput. If we add some further hypothesis (not essential to the FPA rationale), for example that the packet discard is a bernoullian process with mean value p, and we discard TCP timeout, we can evaluate the long-term steady-state TCP throughput (packets per second) as in [133]:

$$T = f(RTT, p) = \frac{1}{RTT}\sqrt{\frac{3}{2bp}} \tag{4.5.1}$$

A different formula is known to be more suitable when the timeout probabilities are not negligible [136].

In the same manner, in order to model the network, we assume the TCP source offers a constant traffic intensity to the network, independently from the present network status (queue occupancy and packet discard probability). If we add the hypothesis that the packet arrival at the buffer is a Poisson process (mean value T) and that service times are exponential (mean value $1/C$), we can model the buffer as a $M/M/1/K$ queue (K is the buffer length), and derive the mean number of packets in the router (X, including the packet being transmitted) and the mean dropping probability (p), i.e.:

$$p = g(T) = \frac{1-\rho}{1-\rho^{K+2}}\rho^{K+1} \tag{4.5.2}$$

$$X = \frac{\rho}{1-\rho} - \frac{(K+2)\rho^{K+2}}{1-\rho^{K+2}} \tag{4.5.3}$$

where $\rho = T/C$. From Eq. 4.5.3 the average RTT can be easily obtained, being $RTT = R_0 + X/C$, where R_0 is the propagation delay in the network. Hence the following equation holds for the RTT:

$$RTT = h(T) = R_0 + \frac{1}{C}\left(\frac{\rho}{1-\rho} - \frac{(K+2)\rho^{K+2}}{1-\rho^{K+2}}\right) \tag{4.5.4}$$

In order to determine T, RTT and p, we need to solve the system of equations (4.5.1),(4.5.2) and (4.5.4). If we define the function $F : \Re^3 \rightarrow \Re^3$ as

$$F(T, p, RTT) = [f(RTT, p), g(T), h(T)],$$

then we can note that a solution of such system ($[T^*, p^*, RTT^*]$), if any, satisfies the following relation:

$$[T^*, p^*, RTT^*] = F(T^*, p^*, RTT^*), \tag{4.5.5}$$

i.e. the point $[T^*, p^*, RTT^*]$ is a fixed point for the $\Re^3 \rightarrow \Re^3$ mapping, established by the function F. This remark justifies the name of FPA.

Under some proper conditions about the function F and its definition set, fixed-point theorems can be used to conclude that at least a solution exists. Some of these theorems are shown in what follows. The question of uniqueness is more difficult, eventual monotonicity greatly constraints the possible dynamics.

Different methods can be employed in order to solve Eq. 4.5.5. In particular repeated substitution takes into account the following relation:

$$[T_{i+1}, p_{i+1}, RTT_{i+1}] = F(T_i, p_i, RTT_i), \tag{4.5.6}$$

assuming that

$$\lim_{i->\infty} [T_i, p_i, RTT_i] = [T^*, p^*, RTT^*].$$

This kind of solution is particularly appealing, because Eq. 4.5.6 can be read as a dynamical system, describing the network operation [111, 60]: in our example the TCP source starts assuming zero loss probability ($p = 0$) and injects traffic in the network; the buffer provides a new (different) value of p by dropping packets. The source reacts to this packet loss probability adjusting its sending rate until convergence is reached. Despite this striking interpretation, it is not clear how close Eq. 4.5.6 actually describes the network operation. By the way, convergence of Eq. 4.5.6 is not guaranteed.

Fixed point theorems. In this section we recall some results about the existence of fixed points, starting from the simple and well-known Intermediate Value Theorem, also known as Bolzano's Theorem. The reader interested to these topics can read [89] or [48].

- Intermediate Value Theorem (IVT).

  Theorem 4.5.1. *Let $f : \Re \rightarrow \Re$ be a continuous function, where $[a, b]$ is an interval of $\Re$ and $f(a)f(b) < 0$, then there exists a $x^* \in [a, b]$ such that $f(x^*) = 0$.*

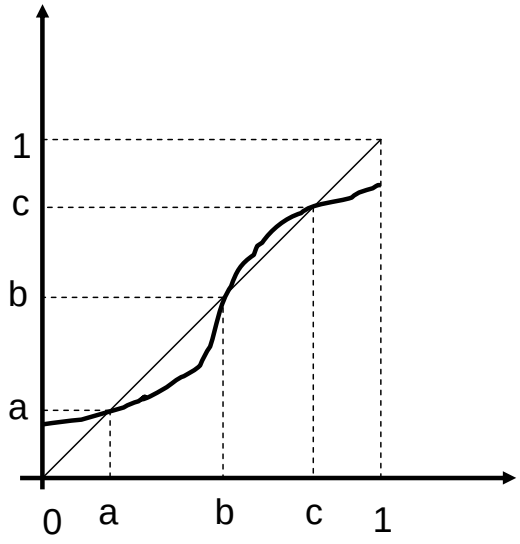

Figure 4.18: An example of the Brouwer's fixed point theorem.

Corollary 4.5.2. *Let $f : [0,1] \to [0,1]$ be a continuous function, then, there exists a fixed point, i.e. $x^* \in [0,1]$ such that $f(x^*) = x^*$.*

- Brouwer's Fixed Point Theorem.

  Brouwer's fixed point theorem (Theorem 1.10.1 in [48]) is a generalization of the corollary to the IVT set out above. Specifically:

  Theorem 4.5.3. *Let $f : S \to S$ be a continuous function from a non-empty, compact, convex set $S \subset \Re^n$ into itself, then there is a $x^* \in S$ such that $f(x^*) = x^*$.*

  Thus, the previous corollary is simply a special case (where $S = [0,1]$) of Brouwer's fixed point theorem. The intuition can be gathered from Fig. 4.18, where we have a function f mapping from $[0,1]$ to $[0,1]$. The curve $(x, f(x))$ intersects the bisector at three points - a, b and c - all of which represent different fixed points as, for instance, at point a, $a = f(a)$.

- Kakutani's Fixed Point Theorem.

The following, Kakutani's fixed-point theorem for correspondences (Theorem 1.10.2 in [48]), can be derived from Brouwer's Fixed Point Theorem via a continuous selection argument.

Theorem 4.5.4. *Let $\phi : S \to S$ be an upper semi-continuous correspondence from a non-empty, compact, convex set $S \subset \Re^n$ into itself such that for all $x \in S$, the set $\phi(x)$ is convex and non-empty, then $\phi(.)$ has a fixed point, i.e. there is a $x^* \in S$ such that $x^* \in \phi(x^*)$.*

Other common assumptions in FPA. In this section we deal with some further assumptions one can find when the FPA is applied to more complex (and realistic) scenarios, when there are many TCP flows and many routers in the network. These assumptions cannot often be easily distinguished from a FPA approach extended to all the network elements, i.e. when we divide the network into as many parts as the number of TCP sources plus the number of the network routers.

Firstly we consider the interaction of many TCP flows, sharing the same network path. Clearly the instantaneous throughput of each source depends on the current throughputs of the other flows. An exact analysis of their interaction is quite complex. For example, a differential equation approach would lead to write one equation for each TCP source. This kind of many-body interactions is quite common in physics: for example consider electrons subject to a magnetic field, where the electron spin contributes to the field. A common simplification is the *Mean Field Theory* (MFT). The great difficulty (e.g. when computing the partition function of the system) is the treatment of combinatorics generated by the interaction terms in the Hamiltonian when summing over all states. The main idea of MFT is to replace all interactions to anyone body with an average or effective interaction. This reduces any multi-body problem into an effective one-body problem. The ease of solving MFT problems means that a lot of insight into the behavior of the system can be obtained at a relatively low cost.

For example in field theory, the Hamiltonian may be expanded in terms of the magnitude of fluctuations around the mean of the field. In this context, MFT can be viewed as the zero-th order expansion of the Hamiltonian

in fluctuations. Physically, this means a MFT system has no fluctuations, but this coincides with the idea that one is replacing all interactions with a "mean field". Quite often, in the formalism of fluctuations, MFT provides a convenient launch-point to studying first or second order fluctuations.

In general, dimensionality plays a strong role in determining whether a mean-field approach will work for any particular problem. In MFT, many interactions are replaced by one effective interaction. Then it naturally follows that if the field or particle exhibits many interactions in the original system, MFT will be more accurate for such a system. This is true in cases of high dimensionality, or when the Hamiltonian includes long-range forces.

MFT is known under many names and guises, e.g. Self Consistent Field Theory, Bragg-Williams Approximation, Bethe Approximation, Landau Theory. In particular the expression "self consistent" refers to the fact that the mean field acting on the particle spin is determined by its own mean value.

As regards TCP sources sharing the same bottleneck, the MFT leads to consider that each flow experiences a link capacity reduced by the average throughput of the other flows, and a longer RTT due to the average queue occupancy from other flow packets. If the paths of the N TCP sources are identical, it is equivalent to consider a single equivalent source, whose throughput is N times that of a TCP source. Explicit references to MFT application for TCP modeling can be found in [16, 17].

The other common assumption concerns the networks of queues, and it is known as *Kleinrock's independence approximation*. Our overview follows [24].

Also in a network of queues there is a form of interaction, in the sense that a traffic stream departing from one queue enters one or more other queues, perhaps after merging with portions of other traffic streams departing from yet other queues. Analytically, this has the unfortunate effect of complicating the character of the arrival process at downstream queues. The difficulty is that packet interarrival times become strongly correlated with packet lengths once packets have traveled beyond their entry queue. As a result, even if packet arrive from outside the network according to a Poisson process, it is impossible to carry out a precise and effective analysis comparable to the one for the $M/M/1$ and $M/G/1$ systems.

Kleinrock suggested that merging several packet streams on a transmission line has an effect akin to restoring the independence of interarrival times and

packet lengths. It was concluded that it is often appropriate to adopt $M/M/1$ queueing model for each communication link regardless of the interaction of traffic on this link with traffic on other links. This is known as the *Kleinrock independence approximation* and seems to be a reasonably good approximation for systems involving Poisson stream arrivals at the entry points, packet length that are nearly exponentially distributed, a densely connected network, and moderate-to-heavy traffic loads.

It turns out that if somehow this correlation is eliminated and randomization is used to divide traffic among different routes, then the average number of packets in the system can be derived as if each queue in the network were $M/M/1$. This is an important result known as Jackson's Theorem [24].

Scientific literature. The employment of FPA techniques to model networks is not a novelty. For example there is a considerable body of literature on the application of fixed point methods to estimating blocking probabilities in circuit switched networks. One of the first work is [95], where the author proposes *Erlang Fixed Point Approximation* in order to evaluate call blocking probability in a network. The Erlang Fixed Point approximation assumes the network is a collection of single-link networks which block independently (similarly to the Kleinrock hypothesis). The existence of a fixed point is proven using the Brouwer's fixed point theorem, Theorem 4.5.3. Other applications can be found in [145].

More recently FPA has been widely used to model the interaction of TCP sources with the network. An overview of some important scientific works follows.

In [114] two approximate techniques for analyzing the window size distribution of TCP flows sharing a RED-like bottleneck queue are presented. Both methods presented above use a fixed point algorithm to obtain the mean window sizes of the flows, and the mean queue length in the bottleneck buffer.

[71] has probably the merit to be the first paper where the FPA approach is clearly stated and presented as a method "which allows the adaptive nature of TCP sources to be accounted for". Based on an input matrix of a mean number of sessions per route and per traffic class, the parameters of packet loss, link utilization and mean TCP throughputs are determined. The paper relies on the Kleinrock's assumption -even if it is not called so- and buffers are

modelled as $M/M/1/K$ queues.

Anyway in the same year two other papers, employing FPA, appeared [38, 60]. In [38] the authors divide the system into two parts: the TCP sources and the network; then they develop an analytical model for each part. The first model uses a Markovian representation to describe the TCP source through the interaction between application level behavior (ON/OFF sources) and transport-layer protocol dynamics. The second model describes the network as a queueing network, bringing into the picture aspects such as topology, queueing capacity at the nodes, link capacity. The authors clearly state main assumptions: the segment generation process as seen by the network is Poisson; average measures instead of complete distributions for losses and delay; decoupling the states of the network and of the sources. In [60] the authors model TCP sources sharing the same RED queue as a feedback control system, with the controlled systems being the TCP senders, the controlling element being RED, which acts on the dropping probability and on the queueing delay. As the dropping probability or the queueing delay increases, the TCP throughput decreases [136]. The authors assume that link's bandwidth is fully utilized by TCP sources if the dropping probability is below a given threshold, under these circumstances they can determine what they call the "queue function" (or queue law) $q = G(p)$. The relation simply follows from the TCP formula in [136] by imposing the total throughput equal to the link bandwidth. The steady-state of the system is obtained by the queue function and the "control function" $p = H(q)$ -i.e. the RED dropping probability function. The two equations are interpreted as the description of a dynamic system having the average queue size and average packet drop rate as state parameters -we discussed above about such interpretation of FPA equations. Some improvements to this approach are in [61]. The contribution is threefold: TCP throughput formula in [136] is improved by taking into account clock granularity and ack loss; the model in [60] is extended to heterogeneous flows (different RTTs), short-lived flows and UDP flows; a method to find the equilibrium of a generic multiple bottleneck network is proposed. The starting point is the relation of TCP throughput as a function of queueing delays and dropping probabilities in the network. The method tries to maximize the throughput of each source, while satisfying the conditions on bandwidth constraints at each link.

In [34] the authors explore the use of fixed point methods to model the

behavior of a large population of TCP flows traversing a network of routers implementing active queue management (AQM) such as RED (random early detection). Both AQM routers that drop and that mark packets are considered along with infinite and finite duration TCP flows. As regards the case of infinite duration flows, the approach appears to be an extension of [60] to multiple bottleneck scenarios. In the case of finite duration flows, the authors restrict themselves to networks containing one congested router. In all cases, they formulate a fixed point problem with the router average queue lengths as unknowns. Once the average queue lengths are obtained, other metrics such as router loss probability, TCP flow throughput, TCP flow end-to-end loss rates, average round trip time, and average session duration can be easily obtained.

In [146] the authors address the problem of predicting long-lived TCP performance in multiple bottleneck networks. TCP flows are characterized by the square root formula, and the buffer from a simple equation predicting that the loss rate is just the proportional excess of the send rate over the capacity.

The same problem appears in [6]. The approach considers the whole network and try to predict the throughput of all connections simultaneously, taking into account their mutual interaction. The authors propose a model for the network and present three equivalent formulations (Complementary Problem, Fixed Point and Nonlinear Programming) of it. In particular, FPA and Nonlinear Programming formulations lead to efficient computational procedures and FPA formulation helps to prove the existence of a solution. In comparison to similar works, like [34], the presented model does not require the pre-identification of bottleneck links and include the possibility of the source rate limitation, but it neglects queueing delay. For this reason it is suitable for networks with large delay-bandwidth product. The authors employ Th.4.5.3 in order to prove the existence of a solution.

The same authors -together with others- consider the effect of queueing delays in [14], where each buffer is modeled as a $M/M/1/K$ queue or as a $M^{[X]}/M/1/K$ queue with batch arrivals. In the first part of the paper they discuss the admissibility of the Poisson hypothesis. They prove the existence and the uniqueness of the solution when the nominal load is less than one for short and long lived TCP flows.

Recently, the authors of [111] investigated some theoretical issues concerning the existence, the uniqueness and the stability of equilibrium points, which

have been determined by FPA. In particular the authors consider a single-bottleneck scenario and short-lived flows. They adopt $M^{[X]}/M/1/K$ models for the queue, they consider the X constant and equal to 1 or to W_M, where W_M is the maximum window size in order to obtain performance bounds. From their analysis the effect of packet retransmissions appears important. If we consider an ideal protocol that provides a perfect selective retransmission mechanism of lost packets, then the solution of the FPA exists, it is unique, and it is stable under the only condition that the application goodput -i.e. the rate of packets delivered by the transport layer to the upper layers- is less than one. A real protocol can retransmit a packet unnecessarily, for example when a TCP source sends again a segment of data that was already received by the destination or that is still in flight along the path towards the destination. As regards TCP, it often happens when a timeout occurs and all the packets with sequence number higher than the last segment acknowledged are considered lost. Under the assumption that unnecessary retransmissions are proportional to necessary retransmissions, two thresholds can be found. If the application goodput is below the lower threshold, a unique and stable solution of the FPA exists, if the goodput is between the thresholds, there are two solutions, one stable and the other unstable, finally if the goodput is greater than the upper threshold, the system does not admit a stable operating point.

As a final remark we note that there has been related work focusing on the development and solution of a set of differential equations describing the transient behavior of TCP flows and queue dynamics [115]. FPA complements this approach. The fixed point approach is much more efficient computationally as the number of unknowns equals the number of links in the network, whereas the differential equations approach requires the solution of a number of equations equal to the number of routers plus the number of TCP flows. On the other hand, the differential equations approach can be used to study transient behavior.

4.5.2 The TCP Sources Model

According to the previous description, we aim to obtain an expression of the average TCP throughput (T, the input to the Network block) and of the average length of the in-sequence packet burst (L, the input to the marker block),

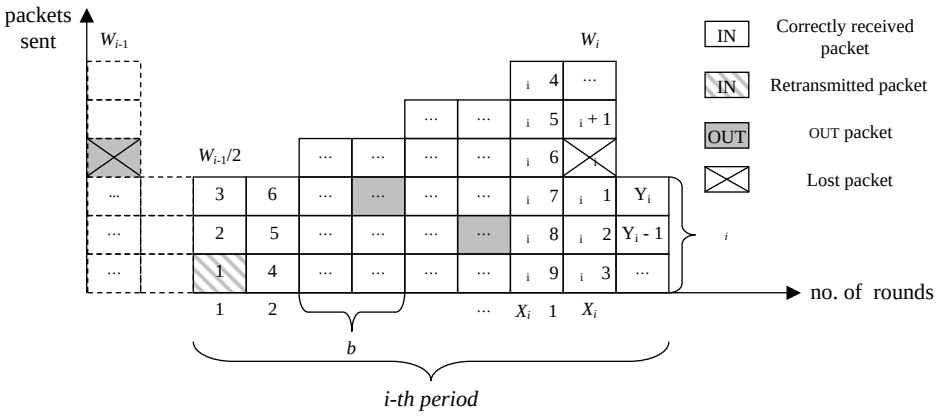

Figure 4.19: Timeline and transmitted packets

given the marking profile (A) and the network status (RTT, p_{in}, p_{out}). We have conjectured a regenerative process for TCP congestion window ($cwnd$), thus extending the arguments in [136] to include two different service classes, with different priority levels.

We only consider loss indications due to triple duplicated acks, which turn on TCP fast retransmit mechanism. We don't consider in our analysis the fast recovery mechanism neither the time-out loss events, for the sake of simplicity. As regards time-out neglecting, this approximation appears to be not critical because PMA spaces OUT packets and hence loss events. For this reason errors are usually recovered by fast retransmission, not by time-out. Such intuition is confirmed by our simulation results, where the number of time-outs appear to be significantly reduced in comparison to a no-marker scenario.

A period of our regenerative process starts when the sender congestion window is halved due to a loss indication. Figure 4.19 shows $cwnd$ trend as rounds succeed. W_{i-1} is the $cwnd$ value at the end of the $(i-1)$-th period, hence in the i-th period $cwnd$ starts from $W_{i-1}/2$ and it is incremented by one every b rounds (b is equal to 2 or 1, respectively if the receiver supports or not the delayed ack algorithm). Notice that, due to neglecting fast recovery and timeouts, each period starts with an IN retransmitted packet, hence the number of packets sent in the period (Y_i) is equal to $L_{seq}+1$, according to the marker description in section 4.3.

In the i-th period we define also the following random variables: I_i is the length of the period; β_i is the number of packets transmitted in the last

round; α_i is the number of the first lost packet since the beginning of the period, while γ_i is the number of packets transmitted between the two losses occurred in the $(i-1)$-th and in the i-th period. We get $Y_i = \alpha_i + W_i - 1$ and $\alpha_i = \gamma_i - (W_{i-1} - 1)$.

Due to the renewal-reward theorem we can obtain the expression for the average throughput of n sources sharing the same path:

$$T(A, RTT, p_{in}, p_{out}) = n\frac{E[Y_i]}{E[I_i]}$$

We first compute $E[Y_i]$. The relation between α_i and γ_i allows us to explicit $E[Y_i]$ as a function of the marking profile (A) and the network status (in particular p_{in}, p_{out}). In general $Y_i \neq \gamma_i$, however if we consider their mean values, it holds:

$$\begin{aligned} E[Y_i] &= E[\alpha_i] + E[W_i] - 1 = \\ &= E[\gamma_i] - (E[W_{i-1}] - 1) + E[W_i] - 1 = \\ &= E[\gamma_i] \end{aligned}$$

Let us denote by N the expected value $E[\gamma_i]$. We compute N as:

$$N = \sum_{n=0}^{\infty} np(n) = \sum_{n=0}^{\infty} (1 - P(n)) = \sum_{n=0}^{\infty} Q(n)$$

where $p(n)$ is the probability of losing the n-th packet after $(n-1)$-th successful transmission, $P(n) = \sum_{l=0}^{n} p(l)$ is cumulative distribution function, and so $Q(n) = 1 - P(n)$ represents the probability of *not* losing any packet among these n. If we put n as $n = k(A+1) + h$, with $0 \leq h < (A+1)$ we can write $Q(n)$ as

$$Q(n) = s_{in}^{kA+h} s_{out}^{k}$$

where $s_{in} = 1 - p_{in}$, $s_{out} = 1 - p_{out}$. The expression of N can be rewritten as

$$N = \sum_{k=0}^{\infty} \sum_{h=0}^{A} s_{in}^{kA+h} s_{out}^{k}$$

and can be solved in a close form:

$$N = \frac{s_{in}^{A+1} - 1}{s_{in} - 1} \frac{1}{1 - s_{in}^{A} s_{out}} \tag{4.5.7}$$

Now we compute $E[I_i]$. Denoting with X_i the round in the i-th period when a packet is lost, we obtain the period length as $I_i = \sum_{j=1}^{X_i+1} r_{ij}$, where $r_{i,j}$ is the j-th round trip time length. Supposing r_{ij} independent of the round number j (i.e. independent of *cwnd* size), taking expectation we find

$$E[I_i] = (E[X] + 1)E[r]$$

where $E[r] = RTT$ is average round trip time.

In the i-th period *cwnd* size grows from $W_{i-1}/2$ to W_i with linear slope $1/b$, so[6]

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b} - 1$$

and taking expectation we get

$$E[W] = \frac{2}{b}\left(E[X] - b\right)$$

To simplify our computations we assume $W_{i-1}/2$ and X_i/b to be integers. Now let us count up all the packets:

$$\begin{aligned} Y_i &= \sum_{k=0}^{X_i/b-1} \left(\frac{W_{i-1}}{2} + k\right) b + \beta_i \\ &= \frac{X_i W_{i-1}}{2} + \frac{X_i}{2}\left(\frac{X_i}{b} - 1\right) + \beta_i \\ &= \frac{X_i}{2}\left(W_{i-1} + \frac{X_i}{b} - 1\right) + \beta_i \\ &= \frac{X_i}{2}\left(W_i + \frac{W_{i-1}}{2}\right) + \beta_i \end{aligned}$$

and taking again expectation it follows

$$N = \frac{E[X]}{2}\left(E[W] + \frac{E[W]}{2}\right) + E[\beta]$$

Assuming β identically distributed between 1 and $W_i - 1$ we can write $E[\beta] =$

[6]There are actually different ways to represent *cwnd* linear growth above the i-th period in the continuous; period bounds are chosen respectively at the beginning of the first round and at the end of the last round, but while in [136] *cwnd* starts from $W_{i-1}/2$ at the beginning of the period, in our analysis it reaches $W_{i-1}/2$ only after $b/2$ rounds.

$E[W]/2$; therefore, solving for $E[X]$:

$$\begin{aligned} E[X] &= \frac{b}{2}\left(-\frac{2+3b}{3b}+\sqrt{\frac{8N}{3b}+\left(\frac{2+3b}{3b}\right)^2}+2\right) \\ &= \frac{3b-2}{6}+\sqrt{\frac{2bN}{3}+\left(\frac{2+3b}{6}\right)^2} \end{aligned}$$

then it follows

$$E[I_i] = RTT\left(\frac{3b-2}{6}+\sqrt{\frac{2bN}{3}+\left(\frac{2+3b}{6}\right)^2}+1\right)$$

Now we can write down the throughput formula:

$$\begin{aligned} T(N,RTT) &= n\frac{N}{RTT(E[X]+1)} \\ &= \frac{nN}{RTT}\ \frac{1}{\frac{3b-2}{6}+\sqrt{\frac{2bN}{3}+\left(\frac{2+3b}{6}\right)^2}+1} \end{aligned} \tag{4.5.8}$$

Throughput dependence from A, p_{in} and p_{out} is included in N through eq.(4.5.7).

Note that if $A_{seq} = A = 0$ (i.e. there is only one class of packets) and $p_{out} = p \to 0$ we get the well-known formula [136]:

$$T(p,RTT) \simeq \frac{n}{RTT}\sqrt{\frac{3}{2bp}}$$

.

Finally, as regards the average length of the in-sequence packet burst (L), from previous remarks it simply follows:

$$L = E[Y_i] - 1 = N - 1 \tag{4.5.9}$$

4.5.3 The Marker Model

We have discussed before about PMA in Sec. 4.3, and we have seen how the procedure acts marking one packet OUT every A_{seq} IN, where A_{seq} is obtained filtering L_{seq} with an autoregressive unitary-gain filter. Hence, given A and L

respectively the average values of A_{seq} and L_{seq}, they are tied by the relation $A = L$[7]. The relation between A_{seq} and L_{seq} has been chosen according to the rationale discussed in section 4.3. Anyway the relation between A and L can be considered a project choice:

$$A = a(L) \tag{4.5.10}$$

A change of the $a()$ law leads to a different marking algorithm, for example pursuing a different target.

As regards the fixed-point approach approximation, we observe that the previous relation looks more suitable as long as the system reaches the state where $p_{in} \simeq 0$ and $p_{out} \simeq 1$. In fact, in the case of $p_{in} = 0, p_{out} = 1$ we would have $A_{seq} = L_{seq}$, not simply $A = L$. In Sec. 4.4.2 we have shown that the algorithm exhibits optimal performance under *hard differentiation* setting, which leads to $p_{in} \simeq 0$ and $p_{out} \simeq 1$. Hence fixed-point approximation appears justified for PMA.

[7] A closer look to the algorithm reveals that this is an approximation due to the update $A := A + 1$ after each OUT-packet transmission.

4.5.4 Network Models

As regards the network, we limited our analysis to a single bottleneck scenario. We considered two different models, which are detailed in the following sections.

The Maximum Utilization Model

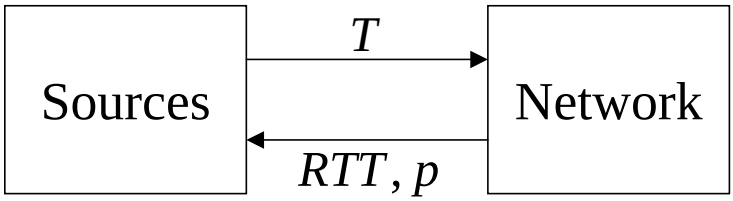

Figure 4.20: Interaction between the Network Model and the Sources Model.

This model appeared in [121].

The network model has been developed following the approach proposed in [60], which presents a fixed-point model for a best-effort scenario with long-lived flows. The system diagram in Fig. 4.16 reduces to that in Fig. 4.20, where no marker appears and there is only one dropping probability p. The dropping probability p and the Round Trip Time RTT can be immediately derived from the queue size. In facts:

$$RTT = R_0 + q/c \tag{4.5.11}$$

where c is the bottleneck link capacity and R_0 is the propagation and transmission delay, and

$$p = H(q) \tag{4.5.12}$$

where $H()$ is referred in [60] as "control function" and depends on the drop module, for example it can be the RED dropping function.

As regards q the authors assume that TCP sources achieve full bottleneck utilization, then for each flow

$$T(p, RTT) = c/n$$

where n is the number of TCP flows. For this reason we denote such model as *the maximum utilization network model*. If we denote by $T^{-1}_{RTT}(p, y)$ the

inverse function of $T(p, RTT)$ in RTT, then

$$RTT = T_{RTT}^{-1}(p, c/n)$$

From eq.(4.5.11), if we consider that q is greater equal than 0 and less equal than the maximum buffer size q_x,

$$q = \max\big(\min\big(c\left(T_{RTT}^{-1}(p, c/n) - R_0\right), q_x\big), 0\big) \tag{4.5.13}$$

This relation is referred in [60] as the 'queue law'. The value of q can be obtained from eq.(4.5.12) and eq.(4.5.13). In Fig. 4.21 the solution of the two equations is shown as the intersection of the curves $q = G(p)$ and $p = H(q)$.

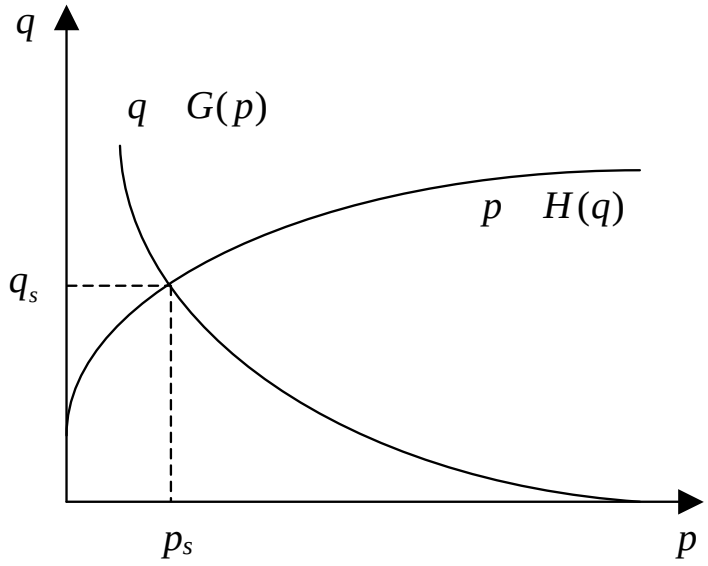

Figure 4.21: Steady state (p_s, q_s) as intersection of queue and control laws.

Now we are going to present our extension to this model. In our DiffServ scenario we have two virtual queue q_{in} and q_{out}, and hence two control law H_{in} and H_{out} for IN and OUT packets respectively. According to RIO behavior:

$$\begin{cases} p_{in} = H_{in}(q_{in}), & (i) \\ p_{out} = H_{out}(q_{in} + q_{out}), & (ii) \end{cases} \tag{4.5.14}$$

The same arguments of [60] lead to the following relation:

$$\begin{aligned} q_{tot} &= q_{in} + q_{out} = \\ &= \max\big(\min\big(q_x, c\left(T_{RTT}^{-1}(N, c/n) - R_0\right)\big), 0\big) \end{aligned} \tag{4.5.15}$$

where $T_{RTT}^{-1}(N, y)$ the inverse function in RTT of the eq.(4.5.8). Note that N depends on A, p_{in}, p_{out}.

The model has 7 variables (q_{in},q_{out},p_{in},p_{out},N,A,L) and 6 equations (4.5.7), (4.5.8), (4.5.9), (4.5.10), (4.5.14) and (4.5.15). We need an equation relating q_{in} and q_{out}, given the traffic offered to the network. If there are not other sources apart from the TCP ones (as we are assuming), a simple relation can be $q_{out} = q_{in}/A$. Usually it holds $A \gg 1$, for this reason we considered $q_{out} \approx 0$.

A further simplification allows us to get again the simple two-variables model in [60]. In fact if H_{in} is invertible, p_{out} is univocally individuated by p_{in}: $p_{out} = H_{out}(H_{in}^{-1}(p_{in}))$. The network is now characterized by the following equations:

$$p_{in}=H_{in}(q_{in}) \tag{4.5.16}$$

$$q_{in}=G(p_{in}) = \tag{4.5.17}$$

$$=\max\left(\min\left(q_x, c\left(T_{RTT}^{-1}(N, c/n) - R_0\right)\right), 0\right)$$

Given A, the operation point (q_{in}, p_{in}) can be found setting up an iterative procedure which can be implemented numerically.

As regards the assumption for a RED law of being invertible, we know there are some intervals where this inversion cannot be accomplished (see Fig. 4.22):

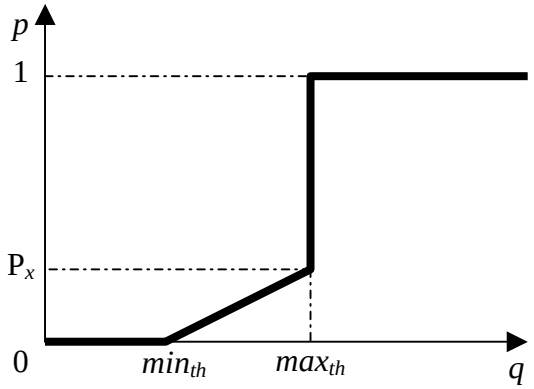

Figure 4.22: RED law.

For $0 \leq q \leq min_{th}$ and $max_{th} \leq q \leq q_x$ is not possible to define the inverse function $q = H^{-1}(p)$; we need to introduce a slight slope to eliminate flats, creating a new "RED$_{inv}$" law for IN class, which is invertible.

About the solutions of the system under the maximum utilization model. Summarizing, our model relies on equations (4.5.7), (4.5.8), (4.5.9), (4.5.10), (4.5.16) and (4.5.17). In this section we afford existence and uniqueness of solutions for this system. Let us focus on the expression of the throughput (4.5.8). We can express the throughput as a function of q_{in} and q_{out}, it appears that it is a not-increasing monotone function of the queues values. In fact it is immediate to note that RTT is an increasing function of the queues and that N, given A, is a not-increasing function of the dropping probabilities and hence of the queues (4.5.7). N is also a not-decreasing function of A if $p_{in} < p_{out}$. The dependence of A on the queues is more complex. From equations (4.5.7) and (4.5.10), we obtain

$$A + 1 = \frac{s_{in}^{A+1} - 1}{s_{in} - 1} \frac{1}{1 - s_{in}^{A} s_{out}} \tag{4.5.18}$$

Remember that our PMA is described by $A = L$, i.e. $A = N - 1$. A is solution of the above equation. It can be shown that if $p_{in} < p_{out}$ A is a decreasing function of p_{in} and p_{out}, hence a not-increasing function of q_{in} and q_{out}.

According to the considerations in the previous section, we can consider only dependence from q_{in}.

The following results hold:

$$\lim_{q_{in} \to 0} T(q_{in}) = +\infty$$

$$\lim_{q_{in} \to +\infty} T(q_{in}) = 0$$

If H_{in} and H_{out} are continuous functions, also $T(q_{in})$ is a continuous functions.

From the previous considerations and hypotheses it follows that the system admits at least one solution, i.e. it exists always a value q_{in}, such that $T(q_{in}) = c/n$ and all the equations are satisfied. One has only to verify that $q_{in} < q_x$. Being the throughput a not-increasing monotone function of the queues values, the solutions set is an interval (eventually reducing to a single point). Finally if H_{in} is a strictly increasing function of q_{in} and $0 < p_{in}, p_{out} < 1$, the throughput is a strictly decreasing function of the queue and hence the solution is unique. It is possible to set up an iterative procedure to find numerically this solution, and this is just what we did using MATLAB.

Now we want to address the solutions in a particular context. Let us remove the previous hypothesis about invertibility and consider equation (4.5.18). We consider the RIO settings in Fig. 4.23, where $max_{out} < min_{in}$. In the range $[max_{out}, min_{in}]$, $s_{in} = 1$ and $s_{out} = 0$, hence equation (4.5.18) reduces to an identity, and the system admits as solution the whole set of values $[max_{out}, min_{in}]$. In Fig. 4.23 we have put in evidence this interval.

As we said, we have introduced a slight slope to RED, in order to make it invertible. Hence the previous range should reduce to a point near the value max_{out}. Despite of this, the MATLAB procedure is affected by numerical approximations, in particular p_{in} can be undistinguishable from 0 in the range $[max_{out}, min_{in}]$, so also the MATLAB procedure can find different solutions in this range, depending on the initial conditions chosen. In particular, unless we start from a point inside the range, the system will converge to the left or right extremity of the interval.

When the model predicts a range of solutions, the dynamics of the system play a fundamental role to determine the final solution. Such dynamics are not considered in a fixed point approach. According to a preliminary study it appears that the model exhibits a higher sensitivity to state perturbations for higher values of the queues, this could justify the simulation results and it suggests that the system dynamics could be recovered by inserting in the numerical procedure a sort of model noise.

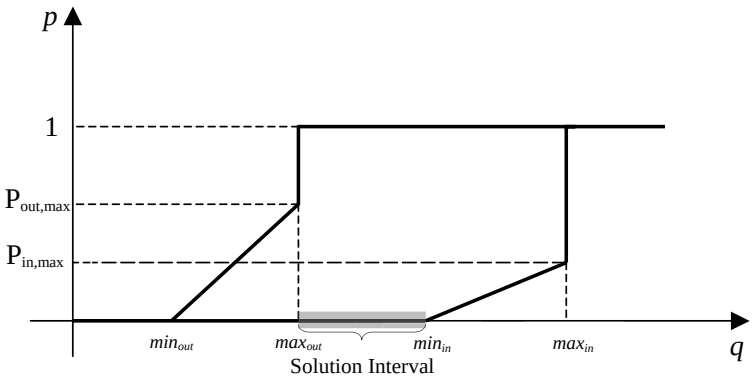

Figure 4.23: Solution Interval.

The Markovian Queue Model

This model appeared in [129].

In the previous section we have proposed a network submodel, extending the approach proposed in [60] for a best-effort scenario to a DiffServ one. As we have seen a limit of this approach is that TCP sources are intrinsically assumed to achieve full bottleneck utilization, hence the model is able to predict average queue occupation, not link utilization. Besides the model predicts a range of solutions when $max_{out} < min_{in}$. These problems could be overcome introducing in the model queue variability.

Anyway here we propose a radically different approach: we consider that the queue can be modelled as a $M/M/1/K$ queueing system. This allows us to evaluate the stationary distribution of the queue for a given offered load T, and then the average values we are interested in, i.e. RTT, p_{in} and p_{out}.

As regards the assumption of Markovian arrivals, it seems to be justified when the TCP connection rate increases [36]. This issue is tackled in [14] by an extensive simulation campaign of a bottleneck link crossed by TCP traffic. The paper shows that the packet arrival process at the input of the bottleneck link is close to Poisson when the number of access links is large and when the speed of these links is slow. This observation scales well with the bandwidth of the bottleneck and it is insensitive to the distribution of file sizes. When short-lived flows are considered, the authors find that, for a certain number of access links, there is some speed of access links that makes the packet arrival process very close to Poisson. They give an empirical expression for this threshold speed $(N * C_{ac})/C \leq 2$, where N is the number of access links, C_{ac} is the access link capacity, and C is the bottleneck capacity. When long-lived flows are considered, the number of flows appears to be the key factor, almost independently from the access link capacity. The authors justify some deviations from the exponential interarrival time distribution. They also note that even when interarrival times look exponential, they are correlated. One cause of correlation are packet pairs sent in the Slow-Start phase from the same access link.

Anyway $M/M/1/K$ models have been widely employed in literature and have shown good performance [107, 71, 104, 67, 153]. Some researchers suggest that a refinement of such models can be obtained if batch arrivals are

introduced in order to account for the burstiness of TCP transmissions within the same window of data. It has been shown [66] that an $M^{[X]}/M/1/K$, a queue with a Poisson arrival of batches of packets distributed according to a batch size distribution $[X]$ closely related to the window size distribution of the senders, provides a good estimate of the packet loss probability for a wide range of values of the offered load. Unfortunately, the $M^{[X]}/M/1/K$ queue does not allow a closed-form expression of the dropping probability p as a function of the offered load. In order to carry on the analysis in [111] the authors resort to an approximation using the formula of the $M/M/1/K$: suppose that all batches are of equal size b packets (of the same size), and that a batch is completely lost if it cannot be entirely received into the queue. Hence K is the number of complete batch that can be received by the buffer, i.e. $K = \lfloor B/b \rfloor$, where B is the buffer size (in packets). In general, $b = 1$ provides a lower bound to p, because TCP traffic is more bursty than a Poisson process, while $b = W_M$, where W_M is TCP maximum window size, provides an upper bound that roughly correspond to the behavior of earlier versions of TCP [142] before congestion control was introduced [90]. The true behavior of TCP, whatever version we consider, should be between the upper and lower bounds that can be analyzed in closed form. The same approach can be found in [14] with two main differences: the worst case is numerically computed evaluating the window distribution under the assumption that the session does not experience any loss and that it will remain always in Slow-Start phase, if a packet discard does not imply the whole batch discard. Despite this considerations, the authors of [66] claim that the difference between the correct value of p and the one obtained with a simple $M/M/1/K$ becomes smaller and smaller as the link becomes more congested, so that an $M/M/1/K$ queue alone indeed provides a good approximation of the packet loss probability (this is due to the fact that the burstiness of the traffic is reduced because the TCP congestion window is small).

Our framework is similar to those of [104] and [153], which model respectively Token Bucket and Single Rate Three Color Marker, but it differentiates because it assumes state dependent arrivals, rather than uniform ones.

These models take into account the presence of different class of traffic and the effect of AQM mechanism like RIO, but they assume that dropping probability depends only on the instantaneous queue size, disregarding the

effect of filtering.

According to [104], the stationary distribution of the queue can be evaluated as:

$$\pi(i) = \pi(0)\left(\frac{T}{C}\right)^i \prod_{j=0}^{i-1}(1-p(j)), i = 1, 2, ...max_{in}$$

where C is the bottleneck capacity, $\pi(0)$ is given by the normalization equation

$$\pi(0) = \left(1 + \sum_{i=1}^{max_{in}}\left(\frac{T}{C}\right)^i \prod_{j=0}^{i-1}(1-p(j))\right)^{-1}$$

and

$$p(i) = \frac{T_{in}p_{in}(i) + T_{out}p_{out}(i)}{T_{in} + T_{out}} = \frac{Ap_{in}(i) + p_{out}(i)}{A+1}$$

Note that we assumed $max_{out} < max_{in}$, and that it is useless considering queue values greater than max_{in} because RIO drops all the incoming packets when the instantaneous queue is equal to max_{in}.

Once $\pi(i)$ has been obtained RTT, p_{in} and p_{out} can be evaluated as

$$RTT = R_0 + q/C = R_0 + \frac{1}{C}\sum_{i=0}^{max_{in}} i\pi(i) \tag{4.5.19}$$

$$p_{in} = \sum_{i=0}^{max_{in}} p_{in}(i)\pi(i) \tag{4.5.20}$$

$$p_{out} = \sum_{i=0}^{max_{out}} p_{out}(i)\pi(i) \tag{4.5.21}$$

where R_0 is the propagation and transmission delay.

We have followed such approach, but results are unsatisfactory. The physical explanation appears from figures 4.24(b) and 4.24(a), which show the empirical distribution coming from simulations and the queue distribution predicted by the model given the same average load, for three different configurations. The RIO settings in the legend are given in the form $(min_{out}, max_{out}, Pmax_{out}) - (min_{in}, max_{in}, Pmax_{in})$. According to the model the queue should exhibit a spread distribution, with high probability for low queue values (in particular the queue distribution decreases if $T < C$), while the empirical distribution looks like a gaussian one: the dynamic adaptive throughput of the TCP sources, which increase their throughput when RTT decreases and vice versa, appear to be able to create a sort of "constant bias".

In order to capture this behavior, we have modified the model in [104], by introducing arrival dependence from the network status. The input to the sub-model is

$$\begin{aligned} F(N) &= T * RTT \\ &= \frac{N}{\frac{3b-2}{6} + \sqrt{\frac{2bN}{3} + \left(\frac{2+3b}{6}\right)^2} + 1} \end{aligned} \tag{4.5.22}$$

and the arrival rate when the there are i packets in the queue is:

$$T(i) = \frac{F}{R_0 + \frac{q}{C}}$$

Now the stationary distribution can be evaluated as:

$$\pi(i) = \pi(0) \prod_{j=0}^{i-1} \frac{T(j)}{C}(1 - p(j)), i = 1, 2, ...max_{in}$$

Figure 4.24(c) shows the queue distribution evaluated by the new model. The similarity with figure 4.24(c) is impressive, the only difference is for the first configuration ($(2, 6, 0.2) - (8, 24, 0.05)$), as regards low queue occupancy. The peak for $q = 0$ is probably due to timeouts, which are more common with low RIO settings, and make TCP throughput less uniform and hence the markovian arrival assumption less accurate.

About the solutions of the system under the markovian queue model. Summarizing, our model has 8 variables (N,A,T,L,RTT,p_{in},p_{out},F) and 8 equations (4.5.7), (4.5.8), (4.5.9), (4.5.10), (4.5.19), (4.5.20), (4.5.21) and (4.5.22). In this section we afford existence and uniqueness of solutions for this system.

Firstly, let us note that F can be expressed as a function of A by equations (4.5.7) and (4.5.22), hence, given A, the variables $q(A)$, $p_{in}(A)$ and $p_{out}(A)$ are determined in a univocal manner. Besides it can be proven that $\pi(i + 1)/\pi(i)$ increases with A, and therefore q, p_{in} and p_{out} are continuous increasing function of A. Hence they are invertible and one can express p_{in} and p_{out} as (increasing) functions of q.

Besides, the following results hold:

$$q(F=0)=0$$

$$\lim_{F\to+\infty} q(F) = max_{in}$$

As regards A, we recall Eq. (4.5.18)

$$A+1 = \frac{s_{in}^{A+1}-1}{s_{in}-1}\frac{1}{1-s_{in}^{A}s_{out}}$$

Being $p_{in} < p_{out}$ A is a decreasing function of p_{in} and p_{out}, hence a decreasing function of q. Besides, being:

$$\lim_{q\to 0} N(q) = +\infty,$$

it holds

$$\lim_{q\to 0} A(q) = +\infty$$

Let us focus on the expression of F (4.5.22). It appears that it is a decreasing function of the queues values, because it is an increasing function of A.

The following results hold:

$$\lim_{q\to 0} F(q) = +\infty$$

$$\lim_{q\to+\infty} F(q) = 0$$

From the previous considerations and hypotheses it follows that the simplified system in F and q admits only one solution, as it is qualitatively shown in figure 4.25. Being all the function monotone, the original system admits only one solution.

It is possible to set up an iterative procedure to find numerically this solution, and this is just what we did using MATLAB.

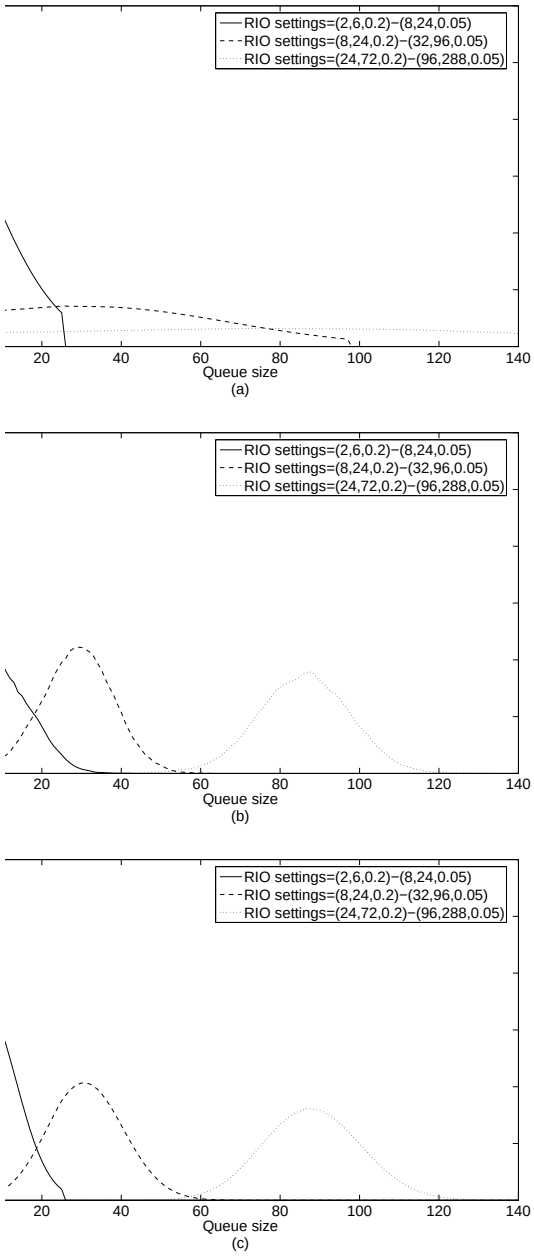

Figure 4.24: (a) Queue distribution predicted by the model with uniform arrivals. (b) Queue distribution obtained by simulations. (c) Queue distribution predicted by the model with state dependent arrivals.

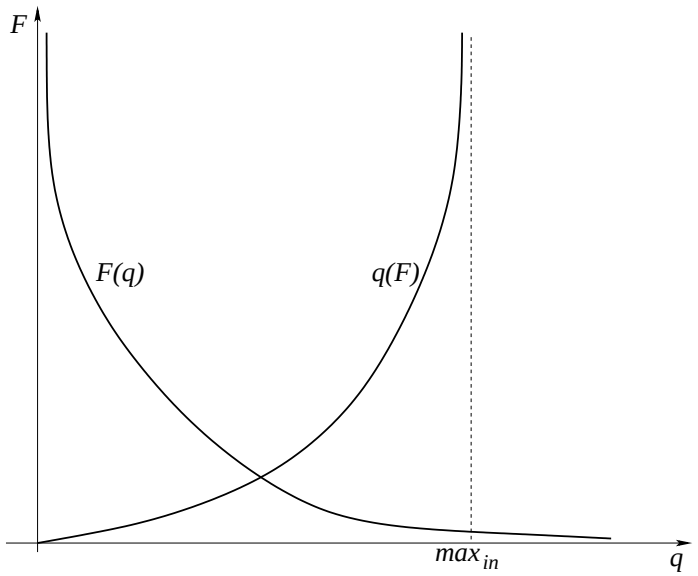

Figure 4.25: Existence and uniqueness of the solution.

Table 4.3: Queue occupation vs RIO settings

RIO (*non overlapping*)	predicted q	measured q
(2,6)(8,24)	{6;8}	8,29
(4,12)(16,48)	{12;16}	12,90
(8,24)(32,96)	{24;32}	27,77
(16,48)(64,192)	{48;64}	54,87
(32,96)(128,384)	{96;128}	108,85
RIO (*contiguous*)	predicted q	measured q
(2,6)(6,18)	6	7,59
(4,12)(12,36)	12	13,32
(8,24)(24,72)	24	23,68
(16,48)(48,144)	48	44,39
(32,96)(96,288)	96	87,77
RIO (*overlapping*)	predicted q	measured q
(2,8)(6,24)	8,00	8,87
(4,16)(12,48)	15,84	15,39
(8,32)(24,96)	30,07	27,01
(16,64)(48,192)	56,36	50,76
(32,128)(96,384)	105,61	100,69

4.6 Model Validation

In this section we do not show validation results for each submodel, but for the global fixed point model, which join the three different submodels (for the TCP sources, for the marker and for the network). In Sec. 4.5.4 we presented two different models for the network. Hence it follows that we have two different fixed point models, whose results we show in the two following subsection. For the sake of simplicity, we refer to the two global models by the name of the specific network submodels.

4.6.1 Maximum Utilization Model

These results appeared in [121].

To validate our model we considered the network topology showed in Fig.4.26, which is the same encountered in [125], consisting of a single bottleneck link with capacity equal to 6Mbps. The Round Trip Time goes from 128ms to 192ms, for an average value of R_0=160ms. The IP packet size is chosen to be 1500 Bytes, for a bottleneck link capacity of c = 500packets/s. We started three different simulation sets, each one related to a different way

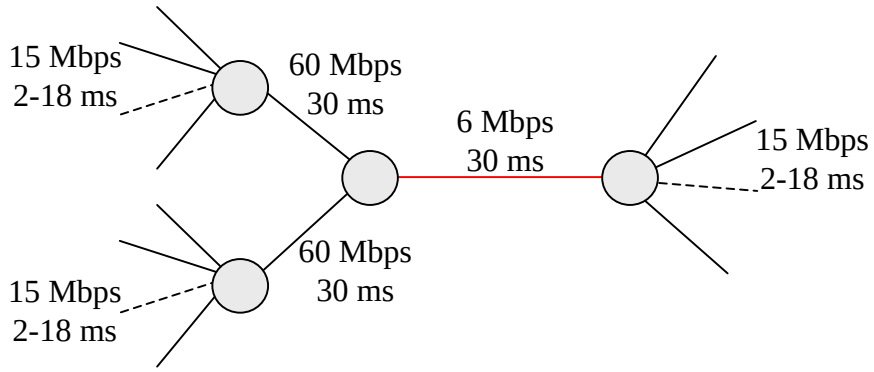

Figure 4.26: Network topology

of configuring RIO thresholds.

We denote as *overlapping* a RIO configuration in which $min_{in} < max_{out}$. We set $max_{out} = 4min_{out}$, $max_{in} = 4min_{in}$ and $min_{in} = 3min_{out}$. We denote as *contiguous* a configuration where RIO thresholds are set so that $max_{out} = min_{in}$. We set $max_{out} = 3min_{out}$, $max_{in} = 3min_{in}$. Finally we denote as *non overlapping* a RIO configuration in which $max_{out} < min_{in}$, more precisely we choose $max_{out} = 3min_{out}$, $max_{in} = 3min_{in}$ and $min_{in} = 4max_{out}$. For each of this settings criteria, we have tested five different configurations, varying min_{out} from 2 up to 32.

We ran our simulations using ns v2.1b9a, with the Reno version of TCP. In Table 4.3 we report the results of our analysis in terms of queue occupation for all tested configurations, while in Fig.4.27 , 4.28 and 4.29 we can see the same results in a more readable form.

As regards the *overlapping* and *contiguous* RIO configurations, the analytical model predicts a unique solution, according to the considerations in Sec. 4.5.4. Figures 4.27 and 4.28 show that model results are quite accurate if compared to simulation results.

As regards the *non overlapping* RIO settings, due to the small slope of the "RED$_{inv}$" curve, the solution interval coincides with the range $[max_{out}, min_{in}]$, as it is shown in Table 4.3. According to the initial value of A our iterative procedure converges to the lower or to the higher value of the solution interval. Actually, with a starting value of $A_0 = 7$ the procedure converges to the lower value, while with $A_0 = 300$ the higher value is held. In Fig.4.29 we reported

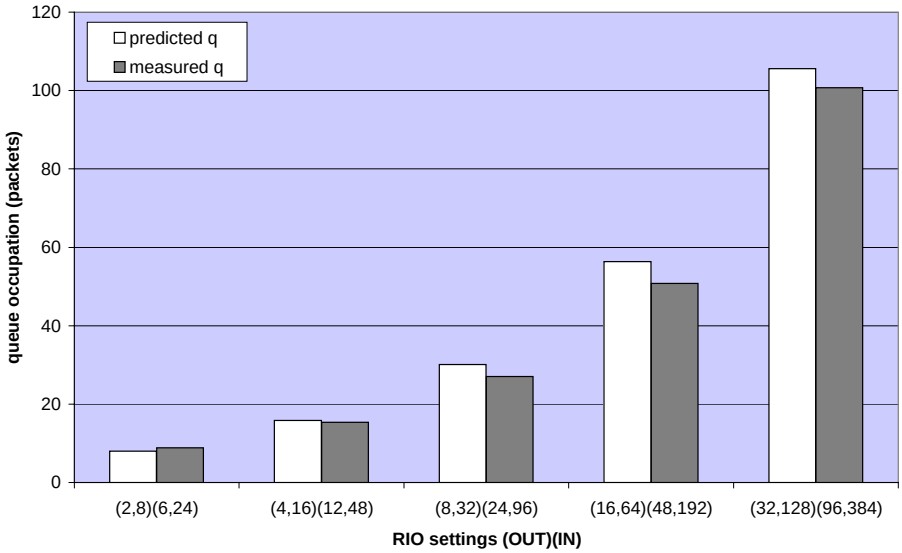

Figure 4.27: Queue occupation vs RIO settings (*overlapping*).

for each setting the measured queue occupation (got from ns simulations) and both lower and higher predictions (respectively horizontal and vertical lines in the bars). The fact that the measured solutions stay between the predicted values is a proof that our speculations are correct. Yet we cannot say anything about the real adaptive dynamics, whose inclusion in the model will be one of our future issues to investigate.

4.6.2 Markovian Queue Model

To validate our model we considered the network topology showed in Fig.4.30, consisting of a single bottleneck link with capacity equal to 6Mbps. Considering both the transmission and the propagation delay of packets and acks in the network, the average Round Trip Time is $R_0 \cong 138$ms. The IP packet size is chosen to be 1500 Bytes, for a bottleneck link capacity of $c = 500$packets/s.

As regards RIO configurations we considered *non overlapping* the ones in which $max_{out} < min_{in}$, more precisely we choose $max_{out} = 3min_{out}$, $max_{in} = 3min_{in}$ and $min_{in} = 4max_{out}$. In previous performance evaluation this kind of settings showed better results in comparison with a *overlapping* RIO configuration in which, $max_{out} >= min_{in}$. We tested seven different configurations,

Table 4.4: Model vs simulation with 10 flows

RIO	T (pkt/s)		G (pkt/s)		q (pkt)		$Pdrop$ (%)		$Pdrop_{in}$ (%)		$Pdrop_{out}$ (%)		A (pkt)	
	mod	sim	mod	sim	mod	sim	mod	sim	mod	sim	mod	sim	mod	sim
(2,6)(8,24)	486.12	476.19	476.04	469.14	8.79	8.51	2.073	1.457	0.844	0.190	68.605	76.498	54.15	58.12
(4,12)(16,48)	488.25	500.60	480.12	494.84	16.72	15.94	1.666	1.178	0.580	0.085	74.590	85.898	67.17	76.60
(6,18)(24,72)	501.29	503.64	494.25	499.19	24.52	21.71	1.405	0.935	0.408	0.050	79.988	88.121	78.83	98.06
(8,24)(32,96)	501.10	504.38	495.03	499.89	32.04	29.29	1.211	0.937	0.300	0.039	83.490	89.652	90.33	98.23
(12,36)(48,144)	500.39	503.76	495.68	499.98	46.63	44.12	0.943	0.777	0.184	0.015	87.848	90.962	114.60	118.21
(16,48)(64,192)	499.93	503.32	495.47	499.99	60.89	58.77	0.892	0.687	0.120	0.012	90.450	88.296	115.94	128.66
(24,72)(96,288)	499.44	502.39	496.84	500.00	89.00	86.31	0.521	0.517	0.059	0.014	93.627	81.998	201.19	160.50
mean error (%)	-0.49		-0.84		6.14		30.76		659.78		-3.77		-4.91	
max error (%)	-2.47		-2.97		12.92		50.29		1128.05		14.18		25.35	

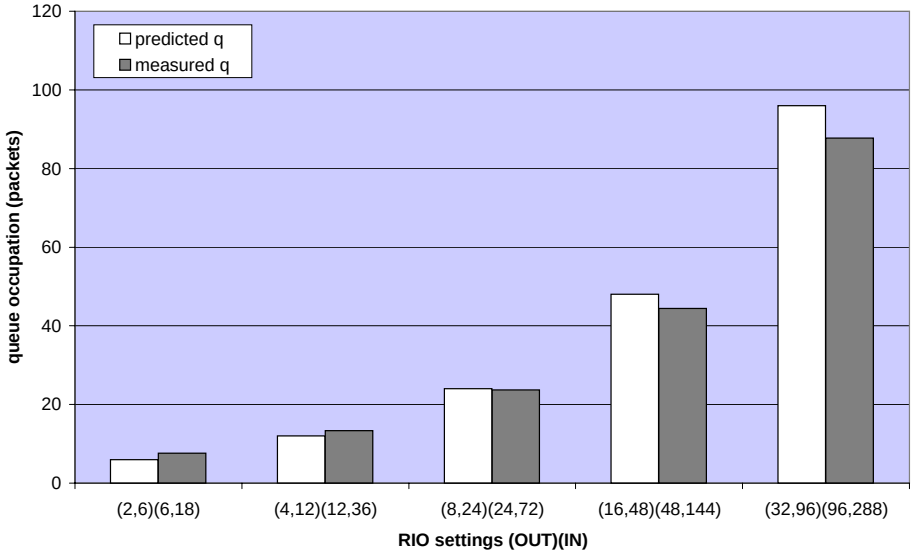

Figure 4.28: Queue occupation vs RIO settings (*contiguous*).

varying min_{out} from 2 up to 24, and for each configuration we gathered statistics from 10 trials of 1000 seconds each. We ran our simulations using ns v2.1b9a, with the Reno version of TCP.

Table 4.4 compares model predictions with simulation results when the number of flows is equal to $n = 10$, as regards throughput (T), goodput[8](G), queue occupancy (q), the dropping probability for the generic packet, for IN packets and for OUT packets (respectively $Pdrop$,$Pdrop_{in}$, $Pdrop_{out}$), and the average length of IN packets bursts. The average mean error over the different settings and the maximum error are shown in the last two rows. The model appears to be able to predict with significant accuracy throughput, goodput and queue occupancy, which are the most relevant performance indexes when we consider TCP long lived performance flows. On the contrary dropping probability estimates are very inaccurate, in particular as regards $Pdrop_{in}$. We think the reason is that the model neglects the effect of filtering on dropping probability calculation from RIO routers. In fact some preliminary results which take into account filtering seem to suggest that filtering: i) can be neglected in order to evaluate the dynamic of the instantaneous queue, ii) it is significant for the evaluation of the dropping probabilities. In particular probabilities estimates look better. At the moment we have introduced the

[8]The goodput is estimated as $G = T(1 - Pdrop)$.

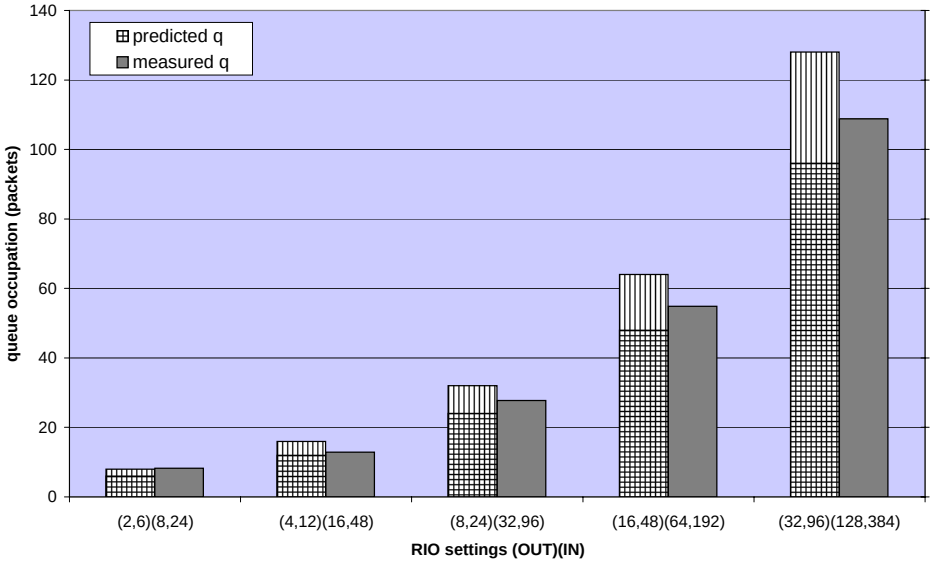

Figure 4.29: Queue occupation vs RIO settings (*non overlapping*).

Table 4.5: Model vs simulation with 6 and 20 flows

$n = 6$	T (pkt/s)	G (pkt/s)	q (pkt)	$Pdrop$ (%)	$Pdrop_{in}$ (%)	$Pdrop_{out}$ (%)	A (pkt)
mean error (%)	-0.87	-0.90	-2.28	3.71	760.33	-2.87	37.33
max error (%)	-1.49	-1.53	-25.96	9.47	1227.52	28.15	61.31
$n = 20$	T (pkt/s)	G (pkt/s)	q (pkt)	$Pdrop$ (%)	$Pdrop_{in}$ (%)	$Pdrop_{out}$ (%)	A (pkt)
mean error (%)	1.75	-0.24	2.76	110.99	607.65	-1.37	-45.67
max error (%)	5.93	2.15	19.37	157.58	798.09	-10.19	-54.12

effect of filtering by considering a two dimensional Markov chain where the status is the pair of instantaneous queue and filtered queue (whose values have been quantized). This approach is particularly heavy from the computational point of view, for this reason, at the moment, we have not adopted it.

We evaluated also the model with the same network topology with a different number of flows ($n = 6$, $n = 20$). The differences between model predictions and simulation results are similar to those observed for $n = 10$ flows. The relative errors for these two scenarios are shown in table 4.5.

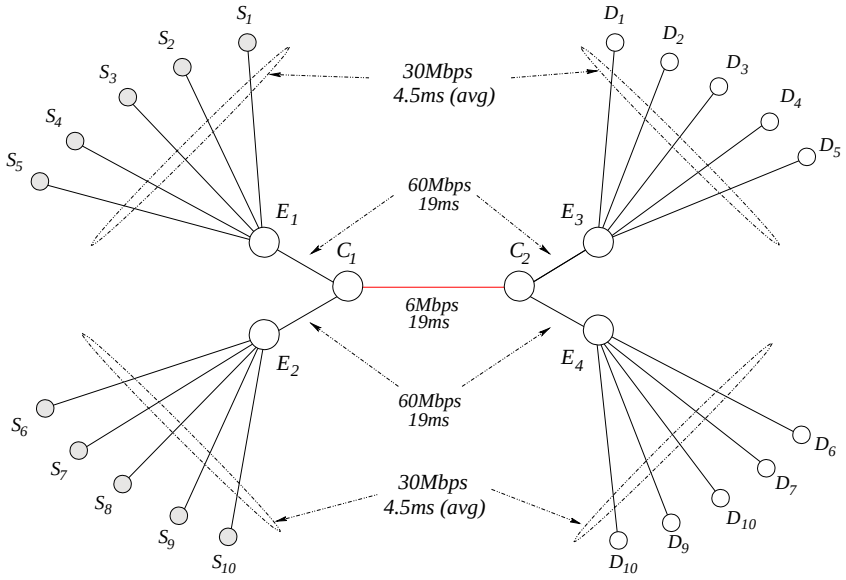

Figure 4.30: Network topology.

Appendix A

Technical Background on Self-Similarity and Long-Range Dependence

A.1 Distributional Self-Similarity

Here we give a definition of self-similarity for continuous time processes in the sense of finite-dimensional distributions. Let $Y(t)$ be a continuous time process.

Definition A.1.1. (H-ss) $Y(t)$ is self-similar with self-similarity parameter, that is Hurst parameter, H, denoted H-ss, if for all $a > 0$ and $t \geq 0$,

$$Y(t) =_d a^{-H} Y(at) \qquad \text{for } t > 0 \tag{A.1.1}$$

Thus $Y(t)$ and its time scaled version $Y(at)$ -after normalizing by a^{-H}- must follow the same distribution.

By noting that[1]

$$Y(t) =_d t^H Y(1)$$

[1]Substitute $t = 1$ and $a = t$ in Eq. A.1.1.

the limiting behavior of $Y(t)$ as t tends to infinity and to zero can be derived. As a consequence of such limiting behavior, $Y(t)$ cannot be stationary, unless $Y(t)$ is degenerate, that is $Y(t) = 0$ for all $t > 0$, or $H = 0$. The exception $H = 0$ is not interesting, as it implies that $Y(t) = Y(1)$ for all $t > 0$. Anyway its increment process $X(t) = Y(t) - Y(t-1)$ can be stationary. When $Y(t)$ is H-ss and has stationary increments, we say $Y(t)$ is *H-sssi*. When considering H-sssi processes the range of H can be restricted to $H > 0$, otherwise $Y(t)$ is not a measurable process. If $Y(t)$ has finite variance, it can be checked that $E[Y(t)] = 0$, $E[Y^2(t)] = \sigma^2|t|^{2H}$, and the autocovariance function is:

$$\gamma(t,s) = \frac{\sigma^2}{2}(|t|^{2H} - |t-s|^{2H} + |s|^{2H}), \tag{A.1.2}$$

where σ^2 is the variance of $X(t)$. The increment process $X(t)$ has mean 0 and autocovariance $\gamma(t+k,t) = \gamma(k)$:

$$\gamma(k) = \frac{\sigma^2}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}).$$

Often it is more convenient to refer to the autocorrelation function $\rho(k) = \gamma(k)/\sigma^2$

$$\rho(k) = \frac{1}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}). \tag{A.1.3}$$

As regards H, note that for $H = 1$ Eq. A.1.3 implies $\rho(k) \equiv 1$. This case is hardly of any practical importance. For $H > 1$, $\rho(k)$ diverges to infinity as k diverges. This contradicts the fact that $\rho(k)$ must be between -1 and 1. We can conclude that if covariances exist and $\lim_{k\to\infty} \rho(k) = 0$, then

$$0 < H < 1.$$

For $H = 1/2$, all correlations at non-zero lags are zero, i.e. the $X(t)$ are uncorrelated. For $0 < H < 1$ and $H \neq 1/2$

$$\rho(k) \underset{k\to\infty}{\sim} H(2H-1)k^{2H-2}. \tag{A.1.4}$$

Here we give the definition of a widely employed H-sssi process: the Fractional Brownian Motion (FBM).

Definition A.1.2. (FBM) $Y(t)$, $t \in \mathbb{R}$, is called *Fractional Brownian Motion* with parameter H, $0 < H < 1$, if $Y(t)$ is gaussian and H-sssi.

The increment process of a FBM is of interest too.

Definition A.1.3. (FGN) $X(t)$, $t \in \mathbb{Z}^+$, is called *Fractional Gaussian Noise* with parameter H, $0 < H < 1$, if $X(t)$ is the increment process of FBM with parameter H.

Note that when $H = 1/2$ FBM reduces to Brownian motion and FGN to white gaussian noise.

The gaussian structure of FBM and FGN renders them especially useful as aggregate traffic models where the aggregation of independent traffic sources -by the central limit theorem- leads to the gaussian property.

A.2 Second-Order Self-Similarity

Let us consider a second-order stationary discrete-time process $X(t)$ and define the *aggregated process* X^m of X at aggregation level m,

$$X^m(i) = \frac{1}{m} \sum_{t=m(i-1)+1}^{mi} X(t).$$

That is $X(t)$ is partitioned into non overlapping blocks of size m, $X^m(i)$ is the average of the i-th block. Let $\rho(k)$ and $\rho^{(m)}(k)$ respectively denote the autocovariance functions of X and $X^{(m)}$. We say that

Definition A.2.1. (Second-Order Self-Similarity) $X(t)$ is *exactly second-order self-similar* with Hurst parameter H $(1/2 < H < 1)$, if for all $k \geq 1$

$$\rho(k) = \frac{1}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}) \tag{A.2.1}$$

$X(t)$ is *asymptotically second-order self-similar* if

$$\lim_{m\to\infty} \rho^{(m)}(k) = \frac{1}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}) \tag{A.2.2}$$

It can be checked that if condition A.2.1 holds, then $\rho(k) = \rho^{(m)}(k)$ for all $m \geq 1$[2]. Thus, second-order self-similarity captures the property

[2]Note that in [140] there is an error in the expression of the autocovariance function $\gamma^{(m)}$ and the incorrect relation $\gamma(k) = \gamma^{(m)}(k)$ is shown rather than $\rho(k) = \rho^m(k)$

that the correlation structure is exactly -condition A.2.1- or asymptotically -condition A.2.2- preserved under time aggregation.

A.2.1 Distributional Self-Similarity versus Second-Order Self-Similarity

Firstly, we remark that distributional self-similarity has been defined for continuous time processes and second-order self-similarity for discrete time processes.

It is immediate to note that if the process $Y(t)$ is H-sssi with $1/2 < H < 1$, and we sample its increment process $X(t)$, i.e. we consider $X(t)$ for $t \in \mathbb{Z}$, this discrete time process is exactly second-order self-similar. In fact Eq. A.1.3 and Eq. A.2.1 coincide.

Besides, while for exactly second-order self-similarity $\rho(k) = \rho^{(m)}(k)$, the increment process of $Y(t)$ satisfies a stronger relation:

$$X =_d m^{1-H} X^{(m)}. \tag{A.2.3}$$

Similarly to the distinction between exactly second-order self-similarity and asymptotically second-order self-similarity, a discrete time process which satisfies Eq. A.2.3 for all $m \geq 0$ is called *exactly self-similar*, while if Eq. A.2.3 holds only in the limit as $m \to \infty$, the process is called *asymptotically self-similar*.

A.3 Long Range Dependence

Let us consider a second-order stationary discrete-time process $X(t)$. $X(t)$ is long-range dependent if $r(k)$ decays slowly. Formally:

Definition A.3.1. (Long Range Dependence, LRD) $X(t)$ is a *long-range dependent* if there exists $\alpha \in (0, 1)$ and a constant $c_{\rho>0}$ such that

$$\rho(k) \underset{\nu \to \infty}{\sim} c_\rho |k|^{-\alpha} \tag{A.3.1}$$

An equivalent definition can be given in the frequency domain where the *spectral density* $\Gamma(\nu) = (2\pi)^{-}1 \sum_{k=-\infty}^{+\infty} \gamma(k) e^{ik\nu}$ has to diverge around the origin, implying ever larger contributions by low-frequency components

$$\Gamma(\nu) \underset{\nu \to 0}{\sim} c_f |\nu|^{-\beta}, \qquad \beta \in (0,1),$$

where $\beta = 1 - \alpha$.

A.4 Self-Similarity versus Long Range Dependence

Let us consider an exactly second-order self-similar process $X(t)$, with Hurst parameter $H \in (1/2, 1)$. From Eq. A.1.4 it follows that $\rho(k)$ behaves as $c_\rho k^{-\alpha}$ as $k \to \infty$, where $c_\rho = H(2H-1) > 0$ and $\alpha = 2 - 2H \in (0,1)$. Hence the process $X(t)$ is long range dependent.

This result can be generalized.

Theorem A.4.1. *A second-order stationary discrete-time process is asymptotically second-order self-similar, with $H \in (1/2, 1)$ if and only if it is long-range dependent.*

A.5 Heavy Tails and Long Range Dependence

Heavy-tailed distributions are strongly related to long-range dependence. From a mathematical point of view LRD asymptotically arises when many random variables, exhibiting heavy-tailedness are superposed, as we are going to detail in this section. Besides heavy-tailedness of certain network-related variables can be shown to underlie the root cause of LRD and Self-Similarity in IP networks (Sec. 2.2).

Definition A.5.1. A random variable Z has a *heavy-tailed distribution* if:

$$\Pr\{Z > x\} \underset{x \to \infty}{\sim} a x^{-c} \tag{A.5.1}$$

where $0 < c < 2$ is called the *tail index* or *shape parameter* and a is a positive constant.

Hence the tail of the distribution decays hyperbolically. In terms of the cumulative distribution function $F_Z(x)$, Z has a heavy-tailed distribution if

$$F_Z(x) \underset{x\to\infty}{\sim} 1 - ax^{-c}.$$

A distribution, which satisfies Eq. A.5.1 has the following property: if $\alpha \leq n \in \mathbb{N}$ than the $n-th$ moment of the distribution is infinite. In particular a heavy-tailed distribution $0 < \alpha < 2$ has infinite variance, and if $0 < \alpha \leq 1$, it also has unbounded mean.

A frequently used heavy-tailed distribution is the *Pareto distribution*, which satisfies Eq. A.5.1 exactly, i.e.:

$$\Pr\{Z > z\} = ax^{-c} = \left(\frac{x}{b}\right)^{-c}, \quad \text{for } x \geq b,$$

where b is called the *location parameter*. Sometimes one prefers to define the Pareto variable for $x \geq 0$, in this case it follows:

$$\Pr\{Z > z\} = \left(\frac{x+b}{b}\right)^{-c}, \quad \text{for } x \geq 0.$$

Now we are going to analytically detail the relation between heavy tails and LRD by introducing two theorems.

Firstly let us consider N independent reward renewal processes $X_i(t), \quad i \in 1, 2, ...N$. Each process alternating takes the value 1 (on) and 0 (off) with i.i.d. on periods (τ_{on}) and i.i.d. off periods (τ_{off}). Each process could model the on/off transmission activity of a source in data networks. Let $S_N(t) = \sum_{i=1}^{N} X_i(t)$ denote the aggregate traffic at time t. Consider the cumulative process $Y_N(Tt)$ defined as

$$Y_N(Tt) = \int_0^{Tt} \left(\sum_{i=1}^{N} X_i(s)\right) \mathrm{d}s$$

Thus, following our network interpretation, $Y_N(Tt)$ measures the total traffic up to time Tt.

Let us assume that τ_{on} is heavy tailed with $1 < \alpha < 2$ (hence it admits mean value), while τ_{off} can be either heavy tailed with $1 < \alpha < 2$ or not. It can be shown [156, 162]:

Theorem A.5.1. *(On/Off Model and FBM) As T and N diverge $Y_N(Tt)$ behaves statistically as it follows*

$$Y_N(Tt) \underset{N,T\to\infty}{\sim} \frac{E[\tau_{on}]}{E[\tau_{on}] + E[\tau_{off}]} NTt + CN^{1/2}T^H B_H(t), \tag{A.5.2}$$

where $H = (3-\alpha)/2$, $B_H(t)$ is FBM with parameter H, and $C > 0$ is a quantity depending only on the distributions of τ_{on} and τ_{off}.

Thus $Y_N(Tt)$ asymptotically behaves as a zero-mean fractional brownian motion fluctuating around $NTtE[\tau_{on}]/(E[\tau_{on}] + E[\tau_{off}])$ when suitably normalized. It is long-range dependent $(1/2 < H < 1)$ if and only if $1 < \alpha < 2$, that is τ_{on}'s distribution is heavy tailed.

Now we consider a different model: an infinite set of processes $X_i(t), \quad i \in \mathbb{N}$, each process $X_i(t)$ is equal to 1 if t belongs to a specific interval $([t_i, t_i + \tau_i), \quad t_i \in \mathbb{N}, \quad \tau_i \in \mathbb{N})$ otherwise $X_i(t)$ is equal to 0. The time instants t_i are determined by a Poisson process, τ_i are i.i.d.. Let us consider the process which indicates how many connections are active at time t:

$$X(t) = \sum_{i\in\mathbb{N}} X_i(t).$$

Each process $X_i(t)$ can model a source transmitting a single packet train (or a single variable size packet), and $X(t)$ denotes the aggregate traffic at time t.

The process $X(t)$ has been studied in [44] as an $M/G/\infty$ queue. In this case t_i are the arrival times of packets at the queue, τ_i are the service times, $X(t)$ coincides with the number of busy servers at time t, known as the busy server process.

Theorem A.5.2. *($M/G/\infty$ and LRD) If the service time distribution has heavy tails with tail index $1 < \alpha < 2$ then $X(t)$, $t \in \mathbb{N}$, is asymptotically second-order self-similar with parameter $H = (3-\alpha)/2$.*

$X(t)$ is long-range dependent. Anyway it has Poisson marginals, but it can be shown that FBM arises naturally as a limiting process by appropriately scaling the Poisson arrival rate and service times [96].

Note that heavy-tailedness is not necessary to generate long-range dependence in aggregate traffic (see for example [21]), anyway there is evidence that LRD in data networks is caused by heavy tails (Sec. 2.2).

A.6 Statistical Analysis of Self-Similarity

The Hurst parameter H is able to quantify the degree of self-similarity (or of LRD) of a process.

In what follows we briefly describe three methods able to provide H estimates: the *aggregate variance*, the *rescaled adjusted range* and the *wavelet estimators*. These are the methods employed in our work (see Sec. 2.4).

A comprehensive overview of LRD statistical-parameters estimation, including the first two methods, can be found in [21]. Aggregated variance and rescaled adjusted range have the advantage of being simple and practical to implement, but often exhibit poor statistical properties [157]. Maximum Likelihood Estimator (MLE) techniques, not used in this work, have better statistical properties, but involve minimization procedures which are complex and slow and need parametric assumptions. The wavelet-based joint estimator is faster than MLE techniques and, according to [2], displays statistical performance comparable to MLE techniques when their parametric assumptions are satisfied and greater robustness under departures from them.

A.6.1 Aggregate Variance

The methods relies on the following striking property of LRD processes. If $\overline{X}_n = 1/n \sum_{i=1}^{n} X_i$ it holds

$$\mathrm{var}(\overline{X}_n) \underset{n\to\infty}{\sim} cn^{2H-2},$$

where $c > 0$ [21].

This suggests the following method for estimating H. The original series $X(i)$ is divided into M blocks of size m and the aggregated series $X^{(m)}(k)$ is

calculated as

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X(i) \qquad k = 1, 2, \ldots.$$

The overall mean is

$$\overline{X}^{(m)} = \frac{1}{M} \sum_{k=1}^{M} X^{(m)}(k).$$

The sample variance of $X^{(m)}(k)$,

$$s^2(m) = \frac{1}{M-1} \sum_{k=1}^{M} \left(X^{(m)}(k) - \overline{X}^{(m)} \right)^2,$$

is an estimator of $Var\left(X^{(m)}\right)$, hence, asymptotically:

$$s^2(m) \approx cm^{2H-2}$$

If we plot $\log(s^2(m))$ versus $\log m$, for large values of m, the points in the plot are expected to be scattered around a straight line with negative slope $2H - 2$. The slope, and hence the Hurst parameter, can be estimated by least-squares regression.

A.6.2 Rescaled Adjusted Range (R/S)

Let $X(t)$ be discrete-time process, and $Y(t) = \sum_{k=1}^{t} X(t)$ its partial sum. Let us define the *adjusted range* as

$$\begin{aligned} R(t,m) = & \max_{0 \le p \le m} \left(Y(t+p) - Y(t) - \frac{p}{m}(Y(t+m) - Y(t)) \right) \\ & - \min_{0 \le p \le m} \left(Y(t+p) - Y(t) - \frac{p}{m}(Y(t+m) - Y(t)) \right). \end{aligned}$$

In order to study the properties that are independent of the scale, $R(t,k)$ is standardized by $S(t,m)$, where $S^2(t,m)$ is the sample (biased) variance of the sequence $(X(t+1), X(t+2), ...X(t+m))$. The ratio

$$\frac{R}{S}(t,m) = \frac{R(t,m)}{S(t,m)}$$

is called the *rescaled adjusted range* or *R/S-statistic.*

It can be shown that if $X(t)$ is long-range dependent with Hurst parameter H, then

$$\frac{R}{S}(t,m) \underset{m\to\infty}{\sim} cm^H.$$

This suggests the following method for estimating H. Given a sample of n observations $X(1), X(2), ..., X(n)$, one subdivides the whole sample into M non overlapping blocks and computes the rescaled adjusted range $R/S(t_i, m)$ for each batch m, starting at points $t_i = (i-1)n/M + 1$ for $i = 1, 2...M$. For values of m smaller than n/M one gets M different estimates of R/S. For values of m approaching n, one gets fewer values, as few as 1 when $t1+m > n$.

If we plot $\log(R/S(t_i, m))$ versus $\log m$, for large values of m, the points in the plot are expected to be scattered around a straight line with positive slope H. The slope can be estimated by least-squares regression.

A.6.3 Wavelet Estimator

We recall that the spectrum of an LRD process $X(t)$ exhibits power-law divergence at the origin:

$$\Gamma(\nu) \underset{\nu\to 0}{\sim} c_f|\nu|^{-\beta}, \qquad \beta \in (0,1).$$

The method, proposed in [2] recovers the power-law exponent $1-2H$ and the coefficient c_f turning to account the following relation

$$E\left\{d_X^2(j,l)\right\} = 2^{j(1-2H)}c_f C$$

where $d_X(j,l) =< X, \psi_{l,j} >$ are the coefficients of the discrete wavelet transform of the signal $X(t)$, i.e. its projections on the basis functions $\psi_{l,j}$, constructed by the mother wavelet through scaling and translation (2^j and l are respectively the scaling and the translation factor).

If we plot $\log(d_X^2(j,l))$ versus j, for large values of j, the points in the plot are expected to be scattered around a straight line with positive slope $1-2H$.

If the process $X(i)$ is gaussian, this estimator provides confidence intervals for H.

Appendix B

An Overview of Differentiated Services

The purpose of this appendix is to briefly present Differentiated Services approach and terminology. The text is mainly based on the RFCs from the Differentiated Services IETF Group (in particular [27]) and on two overview papers from experts in Internet Quality of Service [37, 52].

B.1 The Quality of Service Issue in the Internet

The expression *Quality of Service* (QoS) does not have a universally accepted meaning. Here we limit its meaning to a set of measurable network performance parameters, such as delay, throughput, and loss rate, that can be attached to some identifiable subset of IP traffic through a given network domain. The identifiable subset of traffic belongs to a "user", where "user" spans from a single application program to an entire company. Providing guarantees about the values of network performance parameters requires the implementation and deployment of physical mechanisms throughout the network and then configuring these mechanisms in such a way that their effect, when viewed from the edges of the network, composes into the desired QoS.

The wireline telephone network has often been considered as an undiscussed high-level reference for QoS. Although the telephone network has evolved considerably, a reserved circuit and a single grade of quality are the two defining characteristics of a standard telephone call.

The situation has never been this simple in packet switching networks including the Internet. Firstly there is no requirement in the Internet for a reserved path from source to destination: individual packets are routed separately according to routing tables at each hop and their paths may change due to extraneous circumstances (though in most cases they are constant throughout a session). At the same time, packets are of variable length, typically between 20–4000 bytes. The arrival of packets at network links is both random and bursty, and cannot be modeled by a Poisson distribution [99, 159]. Secondly a given link or router in the core of the Internet may be carrying traffic for thousands or millions of sessions from a wide variety of application types whose intrinsic QoS requirements vary enormously. Unlike the telephone network, measurements show that the dominant traffic pattern is typified by short sessions with a handful of packets in each direction, hence setup costs (e.g. to instantiate circuits) are not acceptable. In summary, the Internet has no standard QoS requirement and no analytically tractable (e.g., Poisson) traffic model.

There have been many attempts to solve the QoS problem in IP networks, before the proposal of the differentiated services approach, e.g. IP precedence and type of service [141, 5], Internet stream protocol [49], integrated services [31, 106].

B.2 The Differentiated Services Approach

Differentiated Services (DiffServ) is an architecture for implementing scalable service differentiation in the Internet. A *service* defines some significant characteristics of packet transmission in one direction across a set of one or more paths within a network. These characteristics may be specified in quantitative or statistical terms of throughput, delay, jitter, and/or loss, or may otherwise be specified in terms of some relative priority of access to network resources. Service differentiation is desired to accommodate heterogeneous application

requirements and user expectations, and to permit differentiated pricing of Internet service. DiffServ is currently being finalized by the IETF and implemented by various vendors.

Diffserv approach recognizes that the relevant entity for effecting service guarantees in the Internet is the administrative domain of a single network operator (either an enterprise network, or a single ISP). Thus, the model is oriented toward edge-to-edge service across a single domain, with an appropriate Service Level Agreement (SLA) assumed to be in place at the edges of the domain. In this way, simple but effective QoS can be built from the components during early deployments and Internet-wide QoS can evolve into a more sophisticated structure as roll-outs and experience increases. Internet-wide differentiated service levels will, of necessity, require agreements between adjacent network providers. The technical part of these agreements will consist of the edge-to-edge Service Level Specifications (SLS) to which a network provider is committed.

Also, the emphasis has been on developing QoS building blocks first rather than the services, recognizing the need for highly scalable mechanisms with minimum impact on the data path elements of high-speed core routers. In facts Diffserv uses simple mechanisms in complex composition, allowing the details of the composition to evolve while the mechanisms, part of the network infrastructure, can remain the same. Such mechanisms are functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and policing. The Diffserv architecture also recognizes the highly variable traffic profiles encountered in the Internet. Unlike IP Precedence, it aims at "sophisticated simplicity" in which the data path mechanisms are simple to implement, but allow very rich network behaviors to be created

Finally, this architecture achieves scalability by traffic aggregation. That is, traffic flows are not independently forwarded, but they are aggregate in a certain number of macroflows, according to their QoS requirements. In order to constitute traffic flow aggregates, packet marking is used. Complex classification and conditioning functions can be implemented only at network boundary nodes, and per-hop behaviors are applied to aggregates of traffic. Hence, unlike the Internet Stream protocol and Integrated Services, per-application

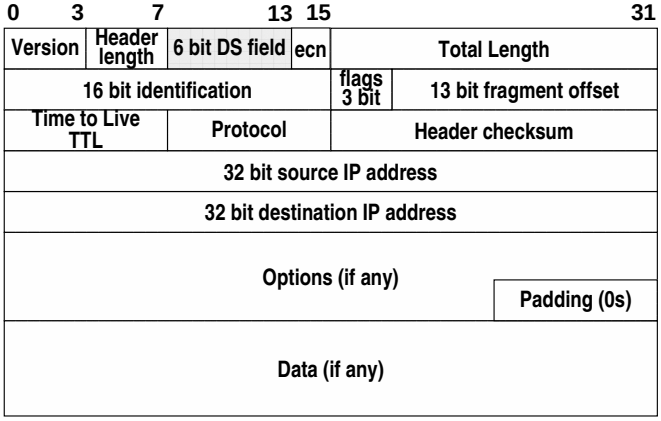

Figure B.1: The DS field in the IP v4 packet.

flow or per-customer forwarding state need not be maintained within the core of the network. Diffserv takes the approach that connections that are visible at the network's edge do not necessarily mean connections that are visible in the interior of the network.

From the previous description, it follows that a distinction is maintained among:

- the service provided to a traffic aggregate,
- the conditioning functions and per-hop behaviors used to realize services,
- the Differentiated Services field value (DS codepoint) used to mark packets to select a per-hop behavior, and
- the particular node implementation mechanisms which realize a per-hop behavior.

These aspects are detailed in the following sections.

B.3 The Differentiated Services Field

The DiffServ architecture is based on the use of the Differentiated Services field (DS field, [131]) in the IP header. The DS field comes from a redefinition of the 8-bit Type of Service (TOS) of the IPv4 header (Fig. B.1), which is commonly unused in today's traffic. In IPv6, there is an equivalent byte called

the Traffic Class octet. The TOS field has been divided into two parts: the 6-bit DS field and a 2-bit ECN field. The DS field is marked with a specific bit-pattern called a DS codepoint (or DSCP) used to indicate how each router should treat the packet. Diffserv packets must have a suitable value in the DSCP field. To emphasize the fact that no session information needs to be stored, this treatment is known as a per-hop behavior (PHB).

The DSCP setting operation is called *marking*. It may occur in two places: the original source of the traffic, e.g. a web server or IP telephony gateway, or a router such as the first router the traffic encounters or the router at the customer/ISP boundary. Marking at the source has the advantage that the classifier may have explicit knowledge of the application in use and can therefore mark packets in an application-dependent way. The second solution has the advantage that no change is needed to servers, but it requires some extra "smarts" in the router. Fortunately, many routers have a very similar capability already, for use with IntServ/RSVP. A different solution is the possibility for the server to use the IntServ/RSVP model to communicate with the boundary router of the DiffServ domain.

B.4 Traffic Classification and Conditioning

Differentiated services are extended across a DS domain boundary by establishing a SLA between an upstream network and a downstream DS domain. From the SLA a Traffic Conditioning Agreement (TCA) is derived. It specifies classifier rules and any corresponding traffic profiles and metering, marking, discarding and/or shaping rules which are to apply to the traffic streams selected by the classifier. A TCA encompasses all of the traffic conditioning rules explicitly specified within a SLA along with all of the rules implicit from the relevant service requirements and/or from a DS domain's service provisioning policy.

As we said a TCA can specify a *traffic profile*. Traffic profile is an optional component of a TCA, which characterizes the temporal properties of a traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in-profile or out-of-profile. For example, a profile could be based on a token bucket, in this case out-of-profile packets are those packets

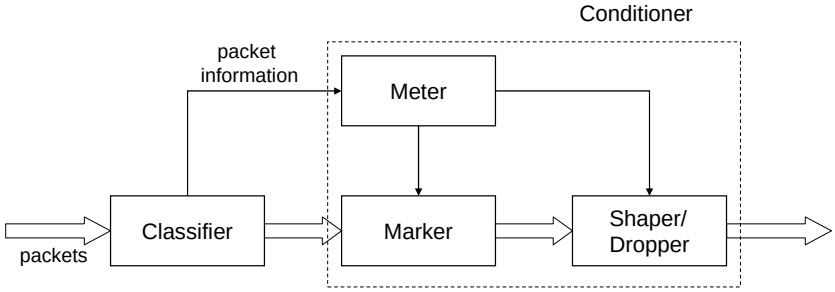

Figure B.2: Logical View of a Packet Classifier and Traffic Conditioner.

in the traffic stream which arrive when insufficient tokens are available in the bucket. The concept of in- and out-of-profile can be extended to more than two levels, e.g., multiple levels of conformance with a profile may be defined and enforced. Different conditioning actions may be applied to the in-profile packets and out-of-profile packets, or different accounting actions may be triggered.

Fig. B.2 shows the relation between a packet classifier and traffic conditioner in a DS router.

B.4.1 Traffic Classifiers

The packet classification policy identifies the subset of traffic which may receive a differentiated service by being conditioned and/or mapped to one or more behavior aggregates (by DS codepoint re–marking) within the DS domain.

Packet classifiers select packets in a traffic stream based on the content of some portion of the packet header. Two types of classifiers have been identified. The Behavior Aggregate classifier classifies packets based on the DS codepoint only. The Multi-Field classifier selects packets based on the value of a combination of one or more header fields, such as source address, destination address, DS field, protocol ID, source port and destination port numbers, and other information such as incoming interface.

Classifiers are used to “steer” packets matching some specified rule to an element of a traffic conditioner for further processing.

B.4.2 Traffic Conditioners

Traffic conditioning performs metering, shaping, policing and/or re–marking to ensure that the traffic entering the DS domain conforms to the rules specified in the TCA, in accordance with the domain's service provisioning policy. The extent of traffic conditioning required is dependent on the specifics of the service offering, and may range from simple codepoint re–marking to complex policing and shaping operations.

For example in-profile packets may be allowed to enter the DS domain without further conditioning; or, alternatively, their DS codepoint may be changed. The latter happens when the DS codepoint is set to a non-Default value for the first time [131], or when the packets enter a DS domain that uses a different PHB group or codepoint→PHB mapping policy for this traffic stream. Out-of-profile packets may be queued until they are in-profile (shaped), discarded (policed), marked with a new codepoint (re–marked), or forwarded unchanged while triggering some accounting procedure. Out-of-profile packets may be mapped to one or more behavior aggregates that are "inferior" in some dimension of forwarding performance to the behavior aggregate into which in-profile packets are mapped.

As it is shown in Fig. B.2, a traffic conditioner may contain the following elements:

Meters measure the temporal properties of the stream of packets selected by a classifier against a traffic profile specified in a TCA. A meter passes state information to other conditioning functions to trigger a particular action for each packet which is either in- or out-of-profile (to some extent).

Markers set the DS field of a packet to a particular codepoint, adding the marked packet to a particular DS behavior aggregate. The marker may be configured to mark all packets which are steered to it to a single codepoint, or may be configured to mark a packet to one of a set of codepoints used to select a PHB in a PHB group, according to the state of a meter. When the marker changes the codepoint in a packet it is said to have *re-marked* the packet.

Shapers delay some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. A shaper usually has a

finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold the delayed packets.

Droppers discard some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is know as *policing* the stream.

B.5 Per-Hop Behaviors

When a packet enters a router, routing logic selects its output port and the DSCP value is used to steer the packet to a specific queue or treatment at that port. The particular handling depends on the definition of the corresponding PHB, which is implemented in nodes by means of some buffer management and packet scheduling mechanisms. The particular PHB is configured by a network management mechanism setting up the QoS behavior table inside the router.

A few PHBs have been standardized so far:

Default behavior [131]: here the DSCP value is zero and the service to be expected is exactly today's default Internet service (i.e., with congestion and loss completely uncontrolled).

Class selector behaviors [131]: here seven DSCP values run from 001000 to 111000 and are specified to select up to seven behaviors, each of which has a higher probability of timely forwarding than its predecessor. The default behavior plus the class selectors exactly mirror the original eight IP Precedence values.

Expedited Forwarding (EF) behavior [47]: the recommended DSCP value is 101110 and the behavior is defined as being such that the departure rate of EF traffic must equal or exceed a configurable rate. EF is intended to allow the creation of realtime services with a configured throughput rate.

Assured Forwarding (AF) behaviors [79]: an AF behavior, say AFi actually consists of three subbehaviors AFi1, AFi2 and AFi3 with increasing dropping probability when congestion occurs. Thus, within the AF

class, differential drop probabilities are available, but otherwise the class behaves as a single PHB. The standard actually defines four independent AF classes, hence $i = 1, 2, 3, 4$.

As regards packet marking schemes for the AF PHB refer to Chapter 4, in particular Sec. 4.1. To the best of our knowledge, at the time of writing this document, there is no published work on marking schemes for EF packets. A simple way of marking EF packets is to meter the traffic against a token bucket profile. Conformant packets are marked as EF and nonconformant packets are either dropped or unmarked (default behavior, Sec. B.5).

B.6 Per-Domain Behaviors

The IETF has adopted the phrase "per domain behavior" to capture the notion of a precise description, including quantitative parameters, of the service provided to an identifiable or target group of packets between the edges of a given differentiated services network domain [132]. A particular PHB (or, if applicable, list of PHBs) and traffic conditioning requirements are associated with each PDB. Each PDB has measurable, quantifiable attributes that can be used to describe what happens to its packets as they enter and cross the DS domain. These derive from the characteristics of the traffic aggregate that results from application of classification and traffic conditioning during the entry of packets into the DS domain and the forwarding treatment (PHB) the packets get inside the domain, but can also depend on the entering traffic loads and the domain's topology. Despite of this, an important consideration for the scalability of any PDB is that its attributes should be almost independent of the amount of traffic entering the domain or the path taken by this traffic inside the DS domain. Furthermore, the edge-to-edge attributes of a PDB should hold regardless of any splitting or merging of the traffic aggregates inside the domain. The attributes of PDBs (throughput, drop rate, delay bound, etc.) are advertised as service-level specifications (SLSs) at the edges of the domain. They are usually listed as statistical bounds or percentiles and not as fixed values. Note that there is no one-to-one relationship between PHBs and PDBs. This means that more than one PDB can be based on the same PHB. On the other hand, a PDB can only be based on one or more PHBs of the

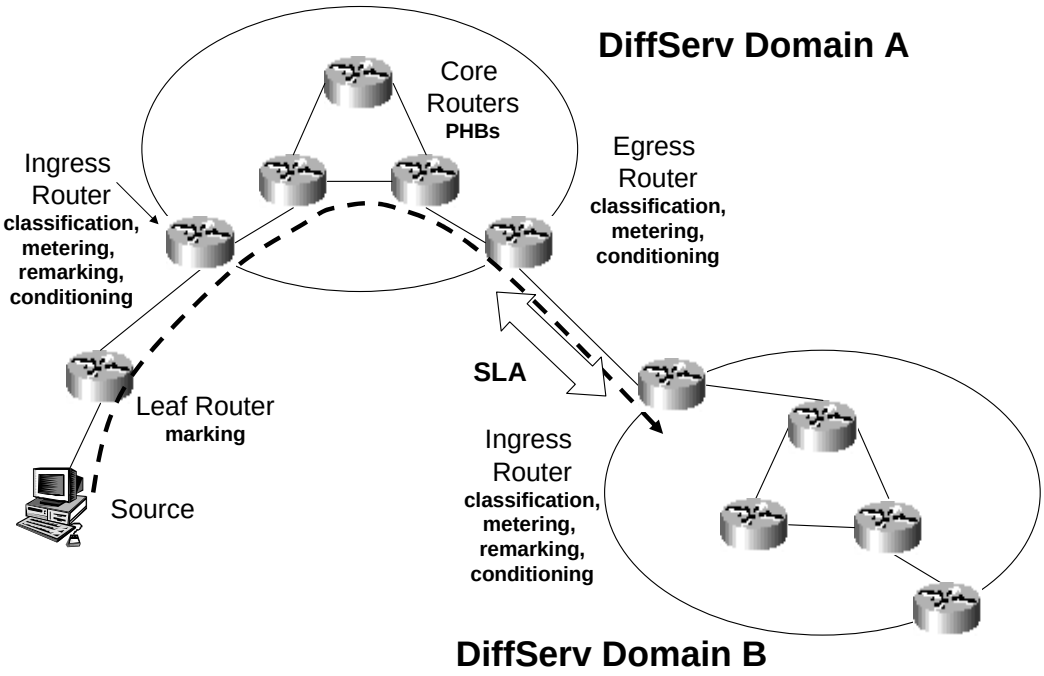

Figure B.3: An example of DiffServ operation.

same group within a single domain. This means that a large number of PDBs can be constructed from a small number of PHBs depending on many factors, such as PHB characteristics, available routes between each ingress–egress pair, and policy.

B.7 Bandwidth Broker

A SLA contains a Service Level Specification (SLS) that characterizes aggregates traffic profile and the PHB to be applied to each aggregate. To automate the process of SLS negotiation, admission control and configuration of network devices correctly to support the provisioned QoS, each DiffServ network may be added with a new component called a Bandwidth Broker (BB) [158]. An ISP, for example, can dynamically negotiate different service level agreements

and bandwidth guarantees with a given customer. Alternatively, a server provider could charge different rates for bandwidth depending on the demand. To do this, the bandwidth broker will contain the ability, using standards based protocols, to communicate with remote bandwidth broker in order to negotiate the Service Level Specification and with local enforcers to determine the state of the network as well as configure the network. The bandwidth broker will take into consideration the ability of the entire network to deliver the policy request. To do this, it will check the state of the network over a period as well as the number of other commitments that have been made before making any decision.

As we can see, the bandwidth broker is a complex entity that might need integration of several technologies such as standard interface for inter/intra domain communication, protocol entity for communication, standard protocol and database. Organizational policies can be configured by using the mechanism provided by BB. On the inter domain level BB is responsible of negotiating QoS parameters and setting up bilateral agreements with neighboring domains. On intra domain level BBs responsibilities include configuration of edge routers to enforce resource allocation and admission control.

Further information on bandwidth brokers can be found in [152].

B.8 An example

In this section we summarize DiffServ operation by a simple network example, illustrated in Fig. B.3.

A DiffServ-aware network consists of multiple DiffServ domains (DSs) that can be viewed as autonomous systems. The boundary routers of each domain perform the necessary traffic conditioning at the edges. Every DS domain makes a SLA specifying the services (in terms of SLSs) that this domain will provide. From the SLA one can derive a traffic conditioning agreement (TCA) that incoming traffic to this domain will be subjected to. Adjacent domains negotiate SLAs among themselves and with customers accessing their network. Each DS domain configures and provisions its internal nodes such that these SLAs can be met. This distribution of configuration responsibilities adds to the flexibility of the DiffServ architecture.

Let us follow a typical packet in the DiffServ scenario in Fig. B.3 from the time it leaves the source until it reaches domain B. As we said in Sec. B.4, the packet can be first metered against a certain traffic profile negotiated between the customer and the network service provider. The packet is then marked with an appropriate DSCP to meet a certain service level in the ISP network. Packet marking can be accomplished either at the host (source) or at the first-hop router (called leaf router) or even at the boundary routers. In our example, all the source traffic is simply marked by the leaf router, leaving to the ISP ingress router the re–marking task, if it is needed. Once the packet has been marked, it becomes part of a specific behavior aggregate with all other packets marked with the same DSCP. At the ingress router of the DS domain the packet is classified using either MF or BA classifiers. The packet is then metered against the negotiated traffic contract and undergoes a policer/shaper, if necessary. At this point, the packet may also be re–marked with a different DSCP if it is non-conformant (e.g. because the source exceeded the negotiated average throughput). The interior or core routers implement the necessary traffic forwarding treatments for different PHBs supported by the DS domain (Sec. B.5). At the exit of a DS domain, the packet may go through another level of traffic conditioning in the egress router of the domain. This level of traffic conditioning guarantees that the traffic leaving one DS domain and entering the adjacent domain follows the traffic contract agreed upon between the domains. Nevertheless traffic metering and conditioning can be repeated at the ingress of domain B, and traffic can be re–marked, even if all the traffic is conformant, because a different DSCP code has been adopted.

From the above example, it is clear that the DiffServ architecture pushes the complexity of managing the network to the edges and leaves packet handling and forwarding in the core of the network as simple and fast as possible. This is a major improvement in scalability of DiffServ over other schemes such as Integrated Services. Although aggregated traffic handling reduces the flexibility of providing QoS guarantees to individual flows, it improves the overall scalability of the architecture.

Appendix C

Some Remarks on the Effects of TCP Packet Reordering

This appendix integrates results of Sec. 4.4.6 about the effect of reordering on the proposed marking scheme. In Sec. C.1 we provide some information about the causes of reordering in IP networks and about its effects on the TCP performance. Although packet reordering is intuitively considered as a negative phenomenon, which may severely affect TCP traffic performance, we show via extensive simulation results, that a limited amount of reordering may improve the network performance in terms of delay versus utilization. I and other colleagues presented this odd result in [128]. In order to explain these apparently surprising and counter-intuitive results, we carry out an analogy with a simplified system model where TCP sources experience an artificially induced steady-state dropping probability. This simplified scenario can be analytically evaluated, and leads us to conclude that, in the presence of a "small" (to be quantified in what follows) packet loss probability, an improvement in terms of network performance is expected.

To the authors' knowledge, [128] is the first paper which claims that TCP packet reordering, rather than being harmful, may be a beneficial phenomenon in terms of overall network performance.

In Sec. C.2 we present the simulation scenario and parameters, focusing in particular on the way reordering has been obtained. We then, in Sec. C.3,

follow up by presenting simulation results which show that packet reordering may have a beneficial effect on the network performance. Then, we try to provide a theoretical justification for the obtained results by making an analogy with a system characterized by a given steady-state dropping probability. Finally conclusive remarks are drawn in Sec. C.4.

C.1 Introduction: Packet Reordering

Packet reordering is a phenomenon which occurs when packets belonging to a same flow are received in a different order than the packet transmission one[1].

IP networks do not provide any guarantee that packets belonging to the same flow are delivered in the correct order, anyway packet reordering was originally considered to be an uncommon event, occurring only in pathological network conditions, e.g. caused by router malfunctioning, route flapping, etc. However, recent studies have shown that in present IP networks there are several causes of reordering under absolutely normal network operation.

Previous research work [20] has identified two major causes for the occurrence of packet reordering: local parallelism and load balancing.

Local parallelism implies that packets may follow multiple paths within a device or logical link. Examples of local parallelism are link-level striping and switches which allow packets traveling between the same source and destination to take different paths through the switch. Local parallelism is an increasing phenomenon in modern Internet devices, because it leads to reduce the cost of equipments and trunks. It is often more cost effective to put two components in parallel than to use one component that has twice the speed. A concrete example of local parallelism at the data link layer is represented by the 802.3ad link aggregation control protocol, today extensively used in giga-ethernet switches.

Load balancing means distributing processing and communications activity evenly across a computer network so that no single device is overwhelmed. For example IS-IS, the Interior Gateway Protocol (IGP) used by Sprint provides a load balancing mechanism based on link weights [86] and can cause reordering.

[1]Hence packet reordering refers to the disorder caused by the network and not to the packet resequencing problem considered often in queueing theory papers like [93].

The amount of reordering in IP networks is quite debated: early experimental results [20] suggest that the probability of a session experiencing reordering is over 90%. More recent results [86, 87] state that the same probability appears to be below 3% and the number of packets that are reordered is no more than 2%. Anyway, whatever the amount of reordering is, reordering is usually considered harmful for TCP flows. In fact, in TCP, packets received out of order cause the transmission of a duplicate ACKs, which interfere with the normal operation of the TCP congestion control algorithm. Due to reordering, TCP flows may experience a great difficulty in opening their congestion windows and may end up in making inefficient usage of the available link bandwidth. Moreover, TCP flows may lose self-clocking and become highly bursty. Refer to [20] for a detailed overview of the effects caused by packet reordering. Here, we simply summarize that forward-path reordering has five side-effects: 1) it may trigger fast retransmission and cause unnecessary retransmissions; 2) it may trigger fast recovery and cause unnecessary slowdown of TCP window growth (cwnd and ssthresh); 3) it may obscure actual packet losses; 4) it may cause the round-trip estimator to poorly estimate the roundtrip time; and 5) it may reduce the efficiency of the receiving TCP. Reverse-path reordering has one major effect: a loss of self-clocking leading to highly bursty transmission patterns.

C.2 Simulation Scenario

We have run simulations using ns-2.1b9a [130]. In order to introduce packet reordering we have modified an existing ns module ("*hiccup*" by Morten Schlaeger [82], Technical University of Berlin). Hiccup provides functionalities to simulate link outages and segment reordering without loosing a packet. The hiccup class is derived from the ns queue class and integrated into the ns link object. The hiccup operates in four different modes. In HICCUP_IDLE mode, all packets are directly passed to the next link object. In HICCUP_DELAY mode packets are queued until the mode is changed back to HICCUP_IDLE, then all packets are passed to the neighbor object in a single, no time consuming, burst. In HICCUP_RESORT mode, the queueing of a single packet is delayed until `resort_len_` later packets are queued. The last mode, HICCUP_CONG,

allows to drop packets. The hiccup module drops all packets until it is set to a different mode.

In our study we employed *hiccup* only in HICCUP_RESORT and HICCUP_CONG modes. The hiccup operation in HICCUP_RESORT mode allows one to obtain specific packet lag[2] patterns. Despite of this, one can shortly see that this behavior is not directly related to the physical phenomenon that causes packet reordering. Besides it can produce spurious timeouts when the hiccup module operates on a per-flow basis (as in our simulations), in fact the hiccup module could wait for new packets before transmitting old ones, while the TCP window do not allow the sender to transmit them.

In order to overcome this problem and to reproduce a more realistic behavior we modified the HICCUP_RESORT mode so that it introduces a tunable per-packet random delay. The random delay is the sum of an exponential random variable with expected value τ and a constant term equal to $T - \tau$. So τ can be changed, but the average delay introduced by hiccup is constant and equal to T. Finally we have also modified the way the module works in HICCUP_CONG: packet are dropped independently with a constant probability (p_s) selected during the simulation set-up.

Our analysis has been developed on the network topology shown in figure C.1. The central part has a simple structure, with a 6Mbps single-bottleneck link between the core routers C_1 and C_2, where C_1 implements RED.

We have set up 10 sources ($S_i, i = 1, 2, ...10$), each one connected to one of the edges E_1 and E_2 by a 30Mbps link. We considered long-lived flows. The sources employ TCP Reno and have always data to transmit to destinations ($D_i, i = 1, 2, ...10$), so the throughput is determined only by the network conditions.

A *hiccup* module has been located between each source and the edge. It introduces an average delay $T = 6$ ms.

In order to avoid synchronization among the sources, each source starts to transmit randomly in the interval 0-1 s, and propagation delays of the access links are chosen so that Round Trip Time are different (from 124ms to 156ms,

[2]In [86] the packet lag of a reordered packet is defined as the number of packets with a sequence number greater than the one of the reordered packet, that are seen before the reordered packet itself; if the hiccup module operates on a per-flow basis, the packet lag coincides with the `resort_len_` parameter.

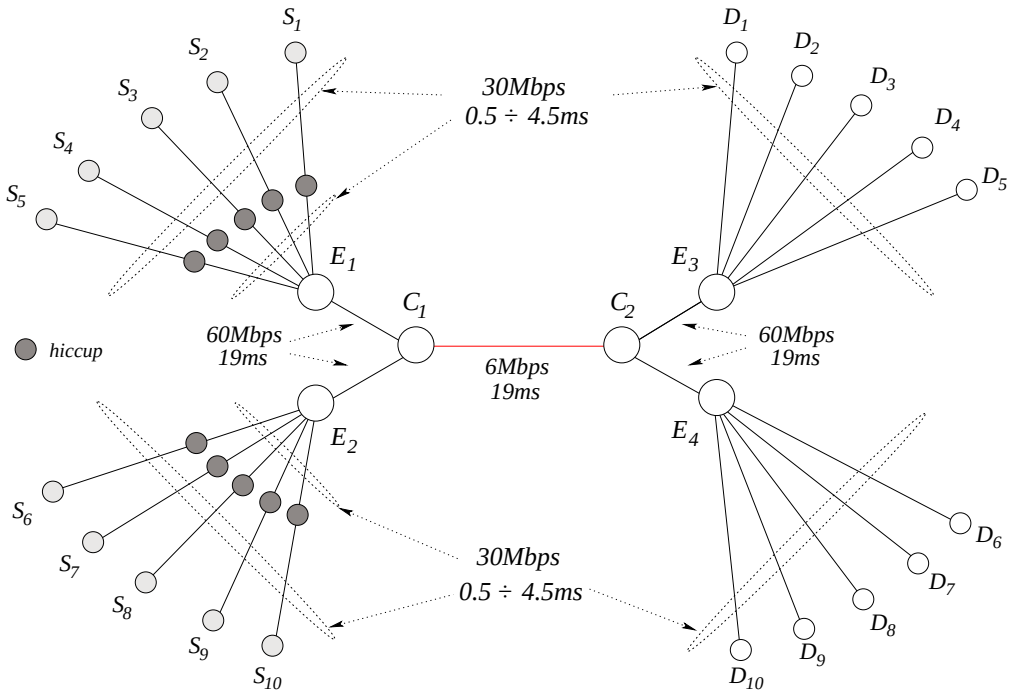

Figure C.1: Network topology

the average value is 140ms).

Each router deploys RED [65] as Active Queue Management. RED gateway calculates the average queue size (avg), using a low-pass filter with an exponential weighted moving average of the instantaneous queue (x): $avg = w_q x + (1 - w_q)avg$. The average queue size is compared to two thresholds, a minimum threshold (min_{th}) and a maximum threshold (max_{th}). When the average queue size is less than the minimum threshold, no packets are dropped. When the average queue size is greater than the maximum threshold, every incoming packet is dropped. When the average queue size is between the minimum and the maximum thresholds, each arriving packet is dropped with probability p, where p is a linearly increasing function of the average queue size. RED configuration is hence specified through three parameters: the minimum and the maximum threshold and the maximum dropping probability in the region of random discard (P_{max}). The thresholds and P_{max} are chosen according to [63], the filter coefficient w_q according to [64], i.e. $max_{th} = 3min_{th}$, $P_{max} = 0.1$ and $w_q = 1 - exp(-M/(10 * RTT * C)) = 0.0012$, where C is the link capacity, M is the packet size and RTT is the Round Trip Time.

RED configuration allows the network provider to trade off link utilization

Table C.1: RED thresholds settings

min_{th} (packets)	max_{th} (packets)
8	24
16	48
24	72
32	96
48	144
64	192
96	288

and delay performance: the higher the RED thresholds, the higher link utilization and delay. Different settings were considered and they are reported in table C.1.

Lastly queue physical lengths were chosen so that packet losses occurred only in the core router C_1, due to RED (not to physical queue overflow).

For each configuration at least 10 simulations with different random seeds were run. Each simulation lasted 1000 simulated seconds, statistics were collected after 50 seconds.

C.3 Simulation Results

The thorough performance evaluation of TCP traffic in the presence of RED routers is by no means a simple task. In fact, the specific configuration of the RED thresholds is proven to strongly affect the TCP traffic performance in terms of throughput and delay. Rather than pre-selecting a given RED threshold configuration (and thus evaluate the TCP traffic performance for the very specific RED configuration chosen), we have found very useful in the past [127] to rely on the so-called concept of "performance frontier".

A performance frontier is a delay versus utilization plot, where different network utilization levels are obtained via different settings of the RED thresholds. As regards link utilization we have not considered the throughput of the TCP sources, but rather the goodput, i.e. the data amount delivered to the applications at the receivers, without losses and retransmissions, because it

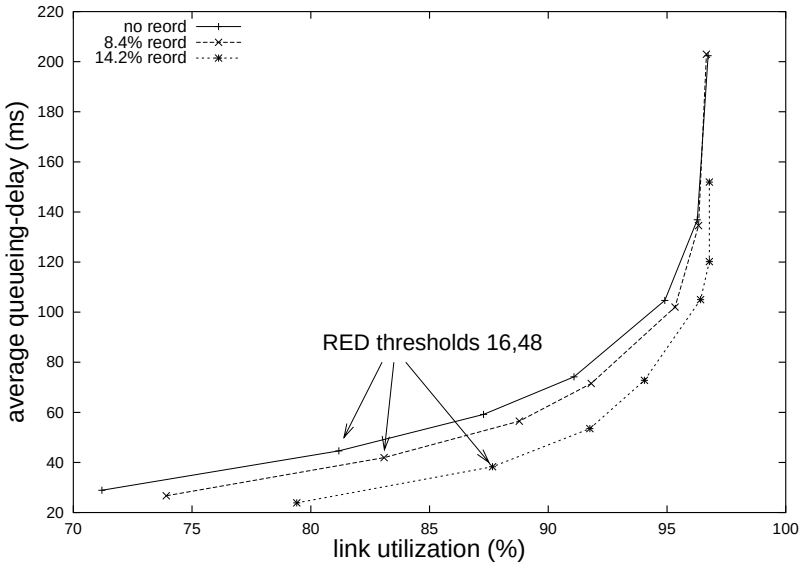

Figure C.2: Delay vs link utilization with packet reordering

is more significant from the user point of view. Fig C.2 shows the effect of reordering on the achievable performance frontiers. Each curve represents the average queueing delay versus the link utilization (goodput), for a specific setting of the *hiccup* module. In particular $\tau = 0, 2, 4$ ms have been chosen respectively for the continuous curve, the dashed one and the dotted one. The resulting percentage of packet reordering is reported in the legends of the figure[3]. Each point in a given curve corresponds to a specific configuration of the RED thresholds. The arrows in the figure show the points corresponding to a particular RED configuration ($min_{th} = 16, max_{th} = 48$). The other points are obtained with the RED threshold configurations summarized in table C.1.

We can note that, as reordering increases, each point moves towards lower queuing delay and higher link utilization. At first sight, we could be surprised seeing the frontiers moving right-down when we increase reordering, since this appears as an overall improvement of network performance. On the contrary we would expect a disturb like reordering to cause impairments to TCP operation.

[3]Actually, as τ increases, the packet reordering probability increases. Being τ constant, the reordering probability decreases slightly as we move from left (lower RED thresholds) to right (higher RED thresholds). Hence, to avoid complex notation, we have found simpler to label each curve with the average value of reordering probability.

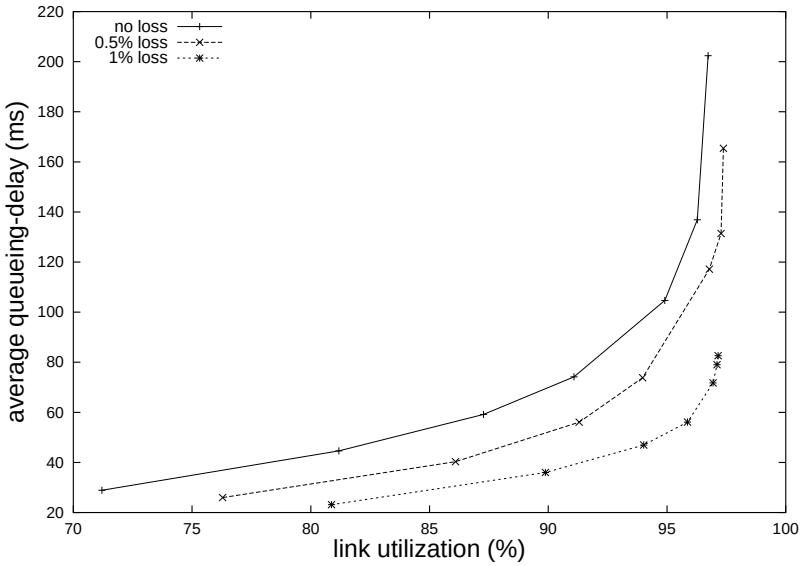

Figure C.3: Delay vs Link Utilization with packet dropping

C.3.1 An analogy: constant dropping probability

In order to explain this counter-intuitive result, we consider the effect of another form of disturb: a steady-state dropping probability independent from the network congestion status, as it could be introduced by a wireless link. It is advantageous because a lot of research has been done on the effect of random losses on TCP performance and different formulas are available which relate the throughput of a TCP source to the network dropping probability and the average Round Trip Time (RTT).

In order to introduce an additional dropping probability we configured the *hiccup* module in HICCUP_CONG mode. In figure C.3, the performance frontiers for different values of the dropping probability p_s appear similar to those in figure C.2.

The improvement for high threshold values is essentially a decrease of the average queueing delay. This reduction can be easily explained by the relation between throughput, average buffer occupancy (q) and packet dropping probability (p), we indicate it with $T(q, p)$ (see for example [136] for a common formula). $T()$ is a decreasing function of q and p. Usually packet discards are caused by routers, hence in well-configured AQM mechanism, dropping probability is a function of queue occupancy through the AQM law, i.e. $p = f_{AQM}(q)$

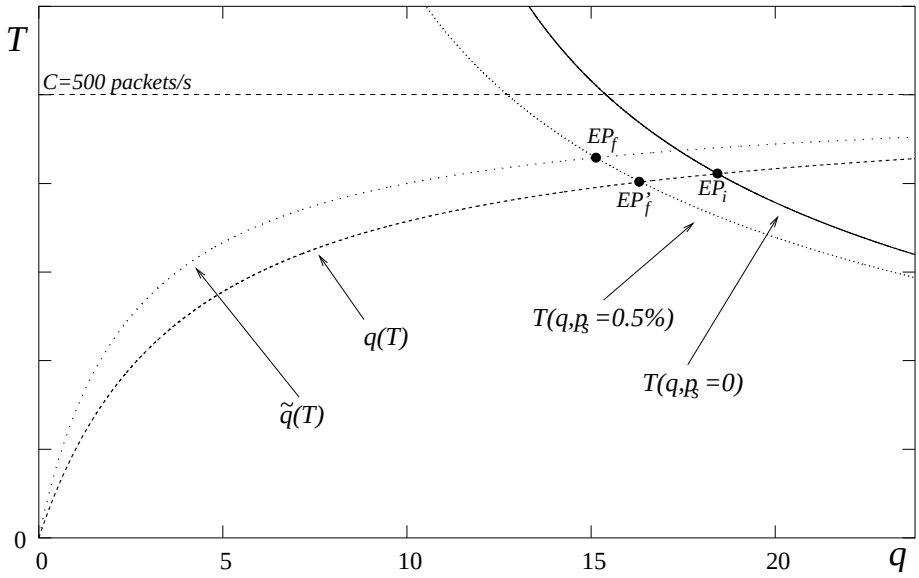

Figure C.4: Determination of equilibrium points

and $T = T(q, f_{AQM}(q)) = T(q)$. For high threshold values, the network equilibrium point can be found if we consider that TCP sources are able to achieve almost 100% utilization of the bottleneck bandwidth, i.e. $T \approx C$. The equilibrium point (p_i, q_i, T_i) satisfies the following relations: $T(q_i, f_{AQM}(q_i)) = C$, $p_i = f_{AQM}(q_i)$ and $T_i = C$.

When we add a steady dropping probability it holds: $p = p_s + f_{AQM}(q)$ and $T = T(q, p_s + f_{AQM}(q)) = T(q, p_s)$. If $p_s < p_i$, then we can assume that $T \approx C$ still holds and the new equilibrium point (p_f, q_f, T_f) satisfies the following relations: $T(q_f, p_s + f_{AQM}(q_f)) = C$, $p_f = f_{AQM}(q_f)$ and $T_f = C$. Being $T()$ a decreasing function of q and p, from the comparison of previous equations it follows $q_f < q_i$. Hence the queueing delay is reduced by the additional dropping probability.

If $p_s > p_i$ no solution is admissible with $T = C$, in particular it will be $T_f < C$. These arguments are inspired to those presented in [60] and justify the curves behavior in figure C.3 for high link utilization.

As regards low link utilization, i.e. low RED thresholds, this kind of justification is not adequate (in fact the reasoning we are going to carry out will lead to results in contrast with simulation results), and we have to proceed in a different way. Let us suppose that an analytical expression exists relating the average queue value to offered TCP traffic, i.e. a relation $q = q(T)$ similar

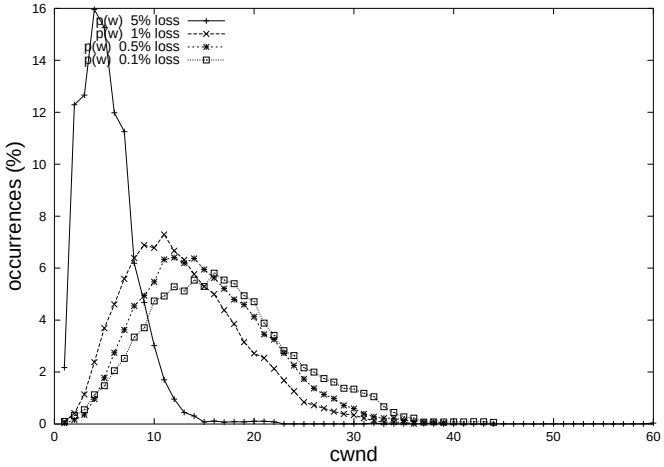

Figure C.5: *cwnd* probability density with packet dropping

to the relation for $M/M/1$ queue system $q = \frac{\rho}{1-\rho}$, where ρ is the normalized offered load. Using such relation and the previous one ($T = T(q)$) one would obtain the network equilibrium point EP_i, according to a fixed point approach. Even if such relation is not known, it is reasonable to assume that it is an increasing function of T and that q diverges as T approaches C. A qualitative curve for $q(T)$ is shown in figure C.4 together with the curve $T = T(q)$ for RED settings (8,24) predicted by the formula in [136]. The intersection of the two curves identifies the equilibrium point EP_i. Note that for high values of link utilization the curve $q(T)$ approaches the line $T = C$, hence this approach recovers the above considerations. At the same time it predicts that the introduction of p_s would move the equilibrium point towards lower delay but also to lower throughput. In facts, in figure C.4 the curve $T = T(q, p_s = 0.5\%)$ intersects the curve $q = q(T)$ in a new equilibrium point EP'_f with $q'_f < q_i$, $T'_f < T_i$, and $p'_f > p_i$. On the contrary Fig. C.3 shows that for low link utilization the throughput increases and the queueing delay is almost constant (a bit smaller).

The error of the previous reasoning is that the average TCP throughput is not sufficient to characterize the stochastic arrival process at the queue -as it is for exponential arrival-, but more complex statistics are needed.

The purpose to adequately characterize the TCP packets arrival is out of the scope of this paper. Anyway we assert that the additional dropping probability makes TCP throughput less variable, producing better network

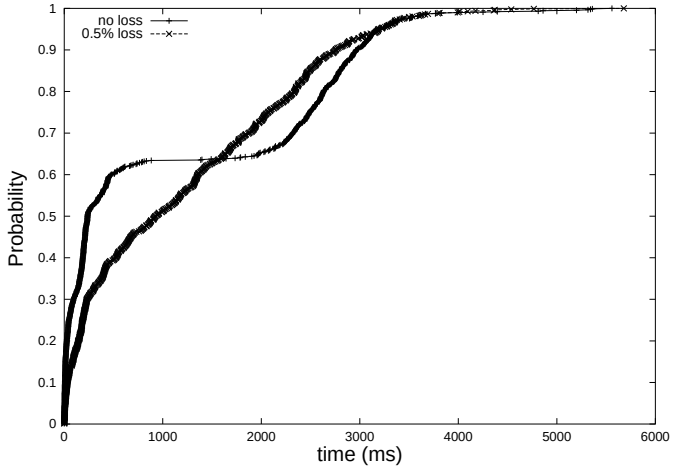

Figure C.6: Probability distribution of interloss time intervals

performance. In order to support such thesis, firstly, we present some results showing that TCP throughput is indeed more regular in the presence of the additional packet dropping probability; secondly we justify such results by RED operation; lastly we try to extend fixed-point argument to identify the new equilibrium point.

Fig. C.5 shows the congestion window *cwnd* probability density for a fixed RED setting and for different values of loss rate introduced by *hiccup*. This figure shows that increasing the loss rate results into a restraint of *cwnd* values so that the *cwnd* of each flow exhibits less variability and it has a lower average value. If the dropping probability exceeds a certain threshold the consequent average window reduction is excessive and network resources utilization is limited by the dropping probability, hence performance dramatically collapses. In particular, a loss rate of 5% is enough to heavily downgrade the functioning of the network, as is visible in the figure: the corresponding curve is extremely shrunk. We did not show the relative performance frontier in previous Fig. C.3 since it is out of range (throughput below 70%).

The different *cwnd* behavior can be explained looking at the packet discards pattern. Fig. C.6 shows the probability distribution of the time intervals between two consequent losses for $p_s = 0$ and $p_s = 0.5\%$. It appears that for $p_s = 0.5\%$ the curve resembles that for an exponential process[4], while for $p_s = 0$

[4]The exponential distribution may be viewed as a continuous counterpart of the geometric distribution originating from i.i.d. losses.

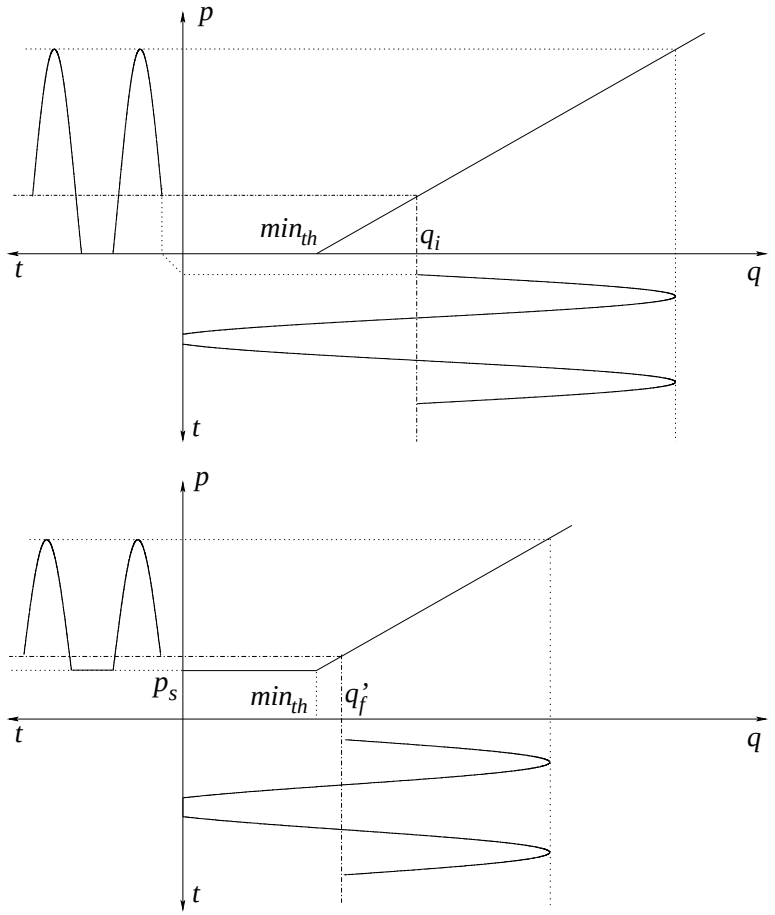

Figure C.7: The effect of queue oscillation on dropping probability

the curve suggests a bimodal dropping process. This can be explained by queue oscillation[5] between the region of null dropping probability ($q < min_{th}$) and the region of linear increasing dropping probability ($min_{th} < q < max_{th}$). Such oscillation causes the dropping probability oscillation and the consequent variability of the TCP window.

Let us observe that this oscillation is not due to an improper RED configuration: the parameters have been chosen according to commonly accepted heuristics (see previous section) and also control theoretic analysis developed in [83] predicts stable behavior. Simulations with $P_{max} = 0.05$, which increases stability margins according to [83], show similar results.

Let us consider what happens for $p_s \neq 0$. In order to determine the new

[5]For the sake of simplicity we do not distinguish between the instantaneous queue value and the filtered queue value calculated by RED in order to determine the dropping probability.

equilibrium point we can assume as a starting point for our consideration the equilibrium EP'_f shown in figure C.4, where $q'_f < q_i$, $T'_f < T_i$ and $p'_f > p_i$. The queue is expected to assume lower values and to exceed min_{th} threshold less frequently. As a consequence dropping probability is more constant. Figure C.7 illustrates qualitatively such behavior. In particular note how an equivalent RED curve has been considered, where the dropping probability is increased by p_s all over the range of queue values, with $q < max_{th}$. These considerations explain the quantitative results in Fig. C.6.

As we said, the lower variability of the dropping probability produces a lower variability of the TCP *cwnd*. If the throughput offered to the network is more regular, then the queue occupancy is lower, i.e. in the terms of the previous fixed-point approach, we should consider a different relation $q = \tilde{q}(T)$, where $\tilde{q}(T) < q(T)$. An hypothetical curve for this new relation is shown in Fig. C.4, and it shows that the new equilibrium point EP_f is characterized by: $q_f < q'_f < q_i$ and $T_f > T'_f$[6]. In particular from Fig. C.4 we note that T_f can even exceed T_i, as it appears from our simulative results.

In this subsection we have presented different results showing that an additional steady dropping probability can lead to better network performance, basically due to a better RED operation, and we have extended a fixed point approach in order to support such results.

C.3.2 Back to reordering

If we consider the introduction of reordering, we find similar results as regards the TCP *cwnd* and the inter-loss time interval. For example figure C.8 shows how *cwnd* density function varies as we introduce reordering. These results confirm the validity of the parallel between reordering and dropping. Now we are going to justify it in details.

In TCP Reno fast retransmit and fast recovery algorithms are implemented [154]. According to these mechanisms if the sender receives three duplicate acknowledgements, it assumes that the data segment indicated by the acknowledgements is lost, it immediately retransmits the lost segment (fast retransmit)

[6]We should evaluate the new packet discards pattern in correspondence to EP_f, and repeat again the same reasoning to iteratively evaluate better approximations for the equilibrium point, anyway the final results should satisfies the relations indicated for EP_f.

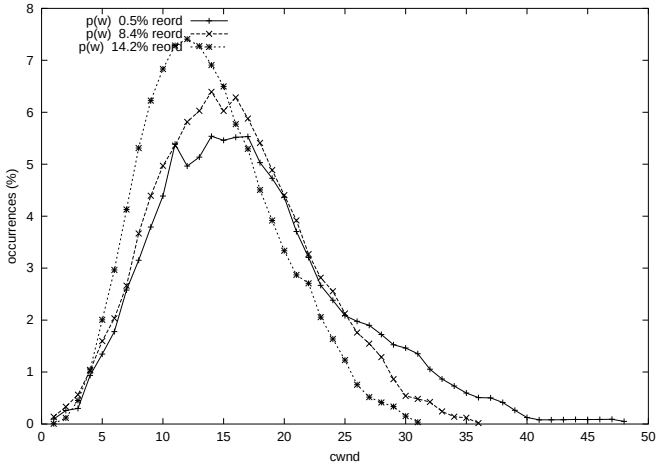

Figure C.8: *cwnd* probability density with packet reordering

and it halves the congestion window (fast recovery). We note that if reordering causes a packet lag greater than or equal to three, the receiver sends three duplicate acknowledgements and the event for the TCP sources is indistinguishable from a loss event. This is the reason of analogous effects on network performance.

In order to verify such hypothesis we have run some simulations where p_s has been chosen equal to the probability of a reorder with packet lag greater than or equal to three, that has been measured in the simulations corresponding to Fig. C.2. The resulting overlapping frontiers are shown in figure C.9. We observe that corresponding performance frontiers overlap, with the exception of the utmost RED configurations for high reordering percentage, where the frontiers branch. In particular the points obtained in HICCUP_RESORT mode are above the ones found in HICCUP_CONG mode. This difference is mainly due to the particular performance metric chosen, as we are going to explain. Even if beyond the threshold of *packet lag* three the TCP sources behave as if a packet loss happened, yet this is not the case: packets are only shuffled, they arrive at the receiver and retransmissions are useless. In Fig. C.9 we are considering goodput, hence such useless retransmissions do not appear, but we can note their effect on the delay. In facts, even if they constitute a small share of the total throughput (0.69%), for high link utilization, the delay is high dependent from the offered load, and they may produce a significantly increase of the delay. In order to support our thesis, we can plot the

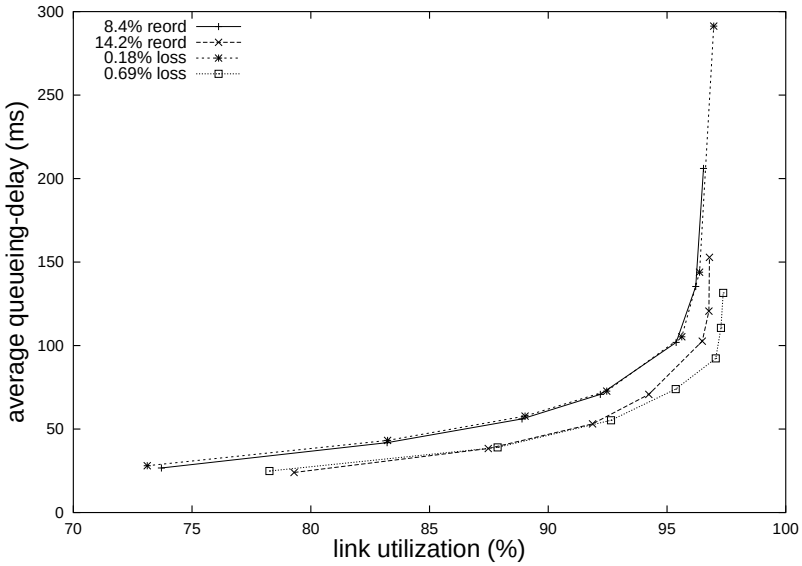

Figure C.9: Reordering frontiers versus Losses frontiers

performance frontiers considering the traffic offered to the bottleneck. In the case of reordering the traffic offered to the bottleneck is the throughput of the TCP sources, while in the case of dropping losses due to hiccup have to be taken into account. For high link utilization losses due to AQM are negligible in comparison to additional losses, hence the traffic offered to the queue is almost equal to the goodput of the TCP sources. Fig. C.10 shows the delay versus throughput and versus goodput respectively for the reordering case and the dropping case. It appears that the two frontiers do not split.

In this subsection we have shown that the similarity of results for reordering and dropping is due to the fact that packet reordering with packet lag greater than or equal to three triggers fast retransmit and fast recovery algorithms. Performance frontiers overlap when the dropping probability and reordering probability are chosen according to this consideration.

C.4 Final Remarks

In this appendix it has been shown that packet reordering is not necessarily a harmful effect in terms of network performance. In fact, our results show that a small amount of reordering can actually improve the network performance. *Small* is referred to the average dropping probability in absence of reordering.

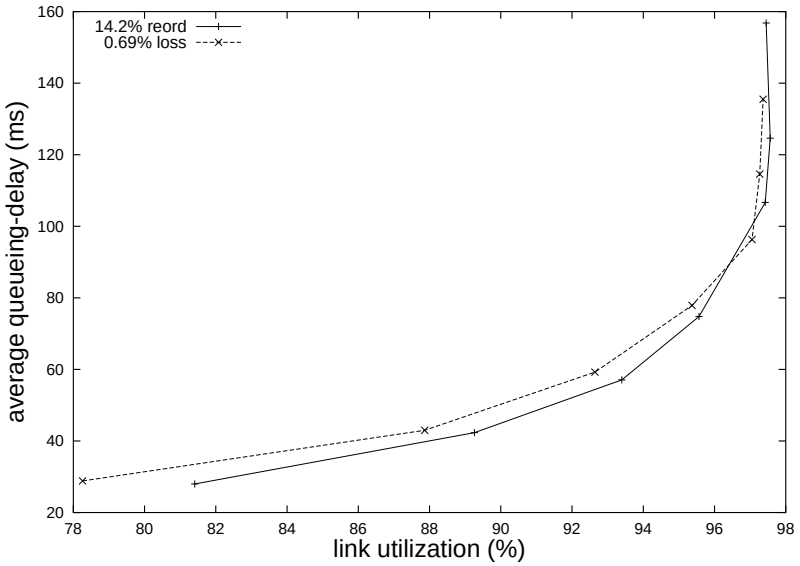

Figure C.10: Performance frontiers in terms of throughput and goodput respectively for congestion and resort mode

In the attempt to find a justification for this result, we have made an analogy with a system characterized by a constant steady-state dropping probability not related to the congestion status of the network (as it happens in wireless links), and we have shown (via both simulation and theoretical analysis) that it produces the same beneficial effects.

Even if the result can appear counter-intuitive and somewhat suprising, in reality we note three strong limitations:

- the improvement is highly dependent from the uniformity of the reordering (or dropping) probability;
- we think that the improvement is highly reduced if short lived flows are considered, or if reverse traffic reordering is taken into account;
- the amount of helpful reordering (dropping) depends from the specific network scenario, the same probability may be harmful for a different configuration.

We remark that the improvement is essentially due to a better operation of RED: basically we have shown that a RED with a different configuration, i.e. with a small dropping probability for low queue values would have performed better for the specific network scenario.

Bibliography

[1] ITU-T Study Group 13, *Traffic control and congestion control in B-ISDN*, ITU-T Recommendation I.371 (1996).

[2] P. Abry and D. Veitch, *Wavelet Analysis of Long-Range Dependent Traffic*, IEEE Transactions on Information Theory **44** (1998), no. 1, pp. 2–15.

[3] *Accelia*, http://www.accelia.net/.

[4] *Akamai*, http://www.akamai.com/.

[5] P. Almquist, *Type of service in the Internet protcol suite*, Request For Comments 1349 (1992).

[6] E. Altman, K. Avrachenkov, and C. Barakat, *TCP Network Calculus: The case of large delay-bandwidth product*, Proc. of IEEE Infocom (2002).

[7] *AQUILA Project, Adaptive Resource Control for QoS Using an IP-based Layered Architecture* , http://www-st.inf.tu-dresden.de/aquila/.

[8] J. Ash, *Dynamic routing in telecommunications networks*, McGraw Hill, 1998.

[9] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, *Rem: Active queue management*, IEEE Network **15** (2001), no. 3, pp. 48–53.

[10] D. Awduche, *Mpls and Traffic Engineering in IP Networks*, IEEE Communications Magazine **32** (1999), no. 12, 42–47.

[11] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, *Overview and Principles of Internet Traffic Engineering*, Request For Comments 3272 (2002).

[12] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *Requirements for Traffic Engineering Over MPLS*, Request For Comments 2702 (1999).

[13] D. Awduche and Y. Rekhter, *Multiprotocol lambda switching: combining MPLS traffic engineering control with optical crossconnects*, IEEE Communications Magazine **39** (2001), no. 3, 111–116.

[14] U. Ayesta, K. Avrachenkov, E. Altman, C. Barakat, and P. Dube, *Simulation Analysis and Fixed Point Approach for Multiplexed TCP flows*, INRIA Technical Report RR-4749 (2003).

[15] F. Azeem, A. Rao, and S. Kalyanaraman, *A TCP-Friendly traffic Marker for IP Differentiated Services*, Proc. of IwQoS (2000).

[16] F. Baccelli, D. Hong, and Z. Liu, *Fixed Point Methods for the Simulation of the Sharing of a Local Loop by a Large Number of Interacting TCP Connections*, INRIA Technical Report RR-4154 (2001).

[17] F. Baccelli, D. R. McDonald, and J. Reynier, *A Mean-Field Model for Multiple TCP Connections through a Buffer Implementing RED*, INRIA Technical Report RR-4449 (2002).

[18] H. Balakrishnan and V. N. Padmanabha, *How network asymmetry affects tcp*, IEEE Communications Magazine **39** (2001), no. 4.

[19] H. Balakrishnan, V. N. Padmanabhan, G. Fairhurst, and M. Sooriyabandara, *Tcp performance implications of network path asymmetry*, Request For Comments 3449 (2002).

[20] J. C. R. Bennett, C. Partridge, and N. Shectman, *Packet Reordering is Not Pathological Network Behavior*, IEEE/ACM Transactions on Networking **7** (1999), no. 6, pp. 789–798.

[21] J. Beran, *Statistics for long-memory processes*, Chapman & Hall, New York, 1994.

[22] J. Beran, R. Sherman, W. Willinger, and M.S.Taqqu, *Variable-bit-rate video traffic and long-range dependence*, IEEE Transactions on Communications (1995).

[23] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, *A Framework for Integrated Services Operation over Diffserv Networks*, Request For Comments 2998 (2000).

[24] D. Bertsekas and R. Gallagher, *Data Networks*, Prentice-Hall, Engelwood Cliffs, NY, USA, 1992.

[25] G. Bianchi and N. Blefari-Melazzi, *Performance evaluation of a measurement-based algorithm for distributed admission control in a diffserv framework*, Proc. of IEEE Globecom (2001).

[26] G. Bianchi, A. Capone, and C. Petrioli, *Throughput Analysis of End-to-end Measurement-Based Admission Control in IP*, Proc. of IEEE Infocom (2000).

[27] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Services*, Request For Comments 2475 (1998).

[28] V. Bolotin, *Modeling Call Holding Time Distributions for CCS Network Design and Performance Analysis*, IEEE Journal on Selected Areas in Communications **12** (1994), no. 3, pp. 433–438.

[29] O. Bonaventure, *Short bibliography on Traffic control and QoS in IP networks*, (2000), available at http://citeseer.ist.psu.edu/bonaventure00short.html.

[30] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Request For Comments 2309 (1998).

[31] R. Braden, D. Clark, and S. Shenker, *Integrated services in the internet architecture: an overview*, Request For Comments 1633 (1994).

[32] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin, *Resource reservation protocol (rsvp) – version 1 functional specification*, Request For Comments 2205 (1997).

[33] L. Breslau, S. Jamin, and S. Shenker, *Comments on the Performance of Measurement Based Admission Control Algorithms*, Proc. of IEEE Infocom (2000).

[34] T. Bu and D. Towsley, *Fixed Point Approximation for TCP behavior in an AQM Network*, Proc. of ACM SIGMETRICS (2001).

[35] José Fabián Roa Buendía, *Management of PSTN traffic*, Comunicaciones de Telefónica I+D **18** (2000).

[36] J. Cao, W. S. Cleveland, D. Lin, , and D. X. Sun, *Nonlinear Estimation and Classification, Chapter: Internet traffic tends toward poisson and independent as the load increases*, Springer, 2002.

[37] B. E. Carpenter and K. Nichols, *Differentiated Services in the Internet*, Proc. of the IEEE **90** (2002), no. 9, pp. 1479–1494.

[38] C. Casetti and M. Meo, *A New Approach to Model the Stationary Behavior of TCP Connections*, Proc. of IEEE Infocom (2000).

[39] C. Cetinkaya and E. Knightly, *Egress Admission Control*, Proc. of IEEE Infocom (2000).

[40] Y. Chait, C.V. Hollot, Vishal Misra, Don Towsley, H. Zhang, and John C.S. Lui, *Providing Throughput Differentiation for TCP Flows Using Adaptive Two-Color Marking and Two-Level AQM*, Proc. of IEEE Infocom (2002).

[41] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, *Tuning RED for web traffic*, Proc. of ACM SIGCOMM (2000).

[42] D. D. Clark and W. Fang, *Explicit Allocation of Best Effort packet delivery service*, IEEE Transactions on Networking **6** (1998), no. 4, pp. 362–373.

[43] The ATM Forum Technical Committee, *Traffic management specification, version 4.1*, (1999).

[44] D. R. Cox, *Statistics: An Appraisal, chapter: Long-range dependence: a review*, pp. pp. 55–74, Iowa State University Press, Ames, 1984.

[45] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, *A Framework for QoS-based Routing in the Internet*, Request For Comments 2386 (1998).

[46] M. E. Crovella and A. Bestavros, *Self-similarity in world wide web traffic: Evidence and possible causes*, IEEE Transactions on Networking **5** (1997), no. 6, pp. 835–846.

[47] B. Davie, A. Charny, J. C. R. Bennett, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, *An Expedited Forwarding PHB (Per-Hop Behavior)*, Request For Comments 3246 (2002).

[48] G. Debreu, *Theory of Value: An axiomatic analysis of economic equilibrium*, Wiley, New York, NY, USA, 1959.

[49] L. Delgrossi and L. Berger, *Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+*, Request For Comments 1819 (1995).

[50] C. A. Desoer and Y.T. Wang, *On the Generalized Nyquist Stability Criterion*, IEEE Transactions on Automatic Control **25** (1980), no. 2, pp. 187–195.

[51] M. El-Gendy and K. Shin, *Equation-Based Packet Marking for Assured Forwarding Services*, Proc. of IEEE Infocom (2002).

[52] M. A. El-Gendy, A. Bose, and K. G. Shin, *Evolution of the Internet QoS and Support for Soft Real-Time Applications*, Proc. of the IEEE **91** (2003), no. 7, pp. 1086–1104.

[53] V. Elek, G. Karlsson, and R. Ronngren, *Admission Control Based on End-to-end Measurements*, Proc. of IEEE Infocom (2000).

[54] W. Fang, N. Seddigh, and B. Nandy, *A time sliding window three colour marker (tswtcm)*, Request For Comments 2859 (2000).

[55] A. Feldmann, A. C. Gilbert, and W. Willinger, *Data network as cascades: Investigating the multifractal nature of the Internet WAN Traffic*, Computer Communications Review (1998), no. 28.

[56] W. Feng, D. Kandlur, D. Saha, and K. Shin, *Adaptive Packet Marking for mantaining end-to-end throughput in a Differentiated Services Internet*, IEEE/ACM Transactions on Networking **7** (1999), no. 5, pp. 685–697.

[57] ———, *Blue: A New Class of Active Queue Management Algorithms*, UM CSE-TR-387-99 (1999).

[58] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, *A SelfConfiguring RED Gateway*, Proc. of IEEE Infocom (1999).

[59] ———, *Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness*, Proc. of IEEE Infocom (2001).

[60] V. Firoiu and M. Borden, *A study of Active Queue Management for Congestion Control*, Proc. of IEEE Infocom (2000).

[61] V. Firoiu, I. Yeom, and X. Zhang, *A Framework for Practical Performance Evaluation and Traffic Engineering in IP Networks*, Proc. of IEEE International Conference on Telecommunications (2001).

[62] S. Floyd, *Comments on measurements based admission control for controlled-load services*, Technical report (1996).

[63] ———, *RED: Discussions of setting parameters*, (1997), email, available at http://www.icir.org/floyd/REDparameters.txt.

[64] S. Floyd, R. Gummadi, and S. Shenker, *Adaptive RED:an algorithm for increasing the robustness of RED's Active Queue Management*, August 2001.

[65] S. Floyd and V. Jacobson, *Random Early Detection gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking **1** (1993), no. 4, pp. 397–413.

[66] M. Garetto, R. Lo Cigno, M. Meo, E. Alessio, and M. Ajmone Marsan, *Modeling Short-Lived TCP Connections with Open Multiclass Queueing Networks*, Proc. of the 7th International Workshop on Protocols For High-Speed Networks (PfHSN 2002) (2002).

[67] M. Garetto, R. Lo Cigno, M. Meo, and M. Ajmone Marsan, *Closed Queueing Network Models of Interacting Long-Lived TCP flows*, IEEE/ACM Transactions on Networking **12** (2004), no. 2, pp. 300–311.

[68] P. Giacomazzi, L. Musumeci, and G. Verticale, *Transport of TCP/IP Traffic over Assured Forwarding IP-Differentiated Services*, IEEE Network, no. 5, pp. 18–28.

[69] R. Gibbens and F. P. Kelly, *Measurement-Based Connection Admission Control*, Proc. of 15th International Teletraffic Congress (1997).

[70] R. J. Gibbens and F. P. Kelly, *Distributed Connection Acceptance Control for a Connectionless Network*, Proc. of 16th International Teletraffic Conference (1999).

[71] R.J. Gibbens, S.K. Sargood, C. Van Eijl, F.P. Kelly, H. Azmoodeh, R.N. Macfadyen, and N.W. Macfadyen, *Fixed-Point Models for the End-to-End Performance Analysis of IP Networks*, Proc. of 13th ITC Specialis Seminar: IP Traffic Measurement, Modeling and Management (2000).

[72] H. Gilbert, O. Aboul-Magd, and V. Phung, *Developing a Cohesive Traffic Management Strategy for ATM Networks*, IEEE Communications Magazine (1991), pp. 36–45.

[73] M. Greiner, M. Jobmann, and C. Klüppelberg, *Telecommunication Traffic, Queueing Models, and Subexponential Distributions*, Queueing Systems: Theory and Applications **33** (1999), no. 1-3, 125–152.

[74] M. Grossglauser and J. Bolot, *On the Relevance of Long Range Dependence in Network Traffic*, IEEE/ACM Transactions on Networking **7** (1999), no. 5, 629–640.

[75] M. Grossglauser and D. Tse, *A Framework for Robust Measurement Based Admission Control*, IEEE Transactions on Networking **7** (1999), no. 3, 293–309.

[76] K. Gu and S. Niculescu, *Survey on Recent Results in the Stability and Control of Time-Delay Systems*, ASME Journal of Dynamic Systems, Measurement, and Control **125** (2003), pp. 158–165.

[77] H. Han, C. V. Hollot, Y. Chait, and V. Misra, *Tcp Networks Stabilized by Buffer-Based AQMs*, Proc. of IEEE Infocom (2004).

[78] J. Harju, Y. Koucheryavy, J. Laine, S. Saaristo, K. Kilkki, J. Ruutu, H. Waris, J. Forsten, and J. Oinonen, *Performance Measurements and Analysis of TCP Flows in a Differentiated Services WAN*, Proc. of the 25th Annual IEEE Conference on Local Computer Networks (2000).

[79] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, *Assured forwarding phb group*, Request For Comments 2597 (1999).

[80] J. Heinanen and R. Guerin, *A Single Rate Three Color Marker*, Request For Comments 2697 (1999).

[81] ———, *A Two Rate Three Color Marker*, Request For Comments 2698 (1999).

[82] *Hiccup Module*, http://www-tkn.ee.tu-berlin.de/∼morten/eifel/ns-eifel.html.

[83] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, *A Control Theoretic Analysis of RED*, Proc. of IEEE Infocom.

[84] B. R. Hurley, C. J. R. Seidl, and W. F. Sewel, *A Survey of Dynamic Routing Methods for Circuit-Switched Traffic*, IEEE Communications Magazine **25** (1987), no. 9, 13–21.

[85] Lluís Fàbrega i Soler, *A proposal of an admission control method for the assured service in the internet*, Ph.D. thesis, University of Girona, Department of Electronics, Informatics and Automatics, 2001.

[86] G. Iannaccone, S. Jaiswal, and C. Diot, *Packet reordering inside the Sprint backbone*, Sprintlabs technical report TR01-ATL-062917 (2001).

[87] G. Iannaccone, S. Jaiswal, C. Diot, J. Kurose, and D. Towsley, *Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone*, Proc. of IEEE Infocom (2003).

[88] J. Ibanez and K. Nichols, *Preliminary simulation evaluation of an assured service*, IETF draft (1998).

[89] V. Istratescu, *Fixed point theory*, Reidel, Dordrecht, Holland, 1981.

[90] V. Jacobson and M.J. Karel, *Congestion Avoidance and Control*, Proc. of ACM SIGCOMM (1988).

[91] Raj Jain, *Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey*, Computer Networks and ISDN Systems **28** (1995), no. 13, 1723–1738.

[92] S. Jamin, P. B. Danzig, S. Shenker, and L. A. Zhang, *A measurement-based admission control algorithm for integrated services packet networks*, IEEE Transactions on Networking **5** (1997), no. 1, pp. 56–70.

[93] F. Kamoun, L. Kleinrock, and R. Muntz, *Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism*, Proc. of the Second International Conference on Distributed Computing Systems (1981).

[94] G. Karlsson, *Providing Quality for Internet Video Services*, Proc. of CNIT/IEEE 10th International Tyrrhenian Workshop on Digital Communications (1998).

[95] F. P. Kelly, *Blocking probabilities in large circuit-switched networks*, Advances in Applied Probability **18** (1986), pp. 473–505.

[96] T. G. Kurtz, *Stochastic Networks: Theory and Applications, chapter: Limit theorems for workload input models*, pp. 55–74, Oxford University Press, Oxford, UK, 1996.

[97] T. Lakshman, U. Madhow, and B. Suter, *Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance*, Proc. of IEEE Infocom (1997).

[98] J. Hyeong Lee and C. K. Jeong, *Improvement of fairness between assured service TCP users in a differentiated service network*, Proc. of Joint 4th IEEE International Conference ATM (ICATM 2001) and High Speed Intelligent Internet Symposium (2001).

[99] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, *On the Self-Similar Nature of Ethernet Traffic*, IEEE Transactions on Networking **2** (1994), no. 1, pp. 1–15.

[100] N. Likhanov, B. Tsybakov, and N.Georganas, *Analysis of an atm buffer with self-similar ("fractal") input traffic*, Proc. of IEEE Infocom (1995).

[101] D. Lin and R. Morris, *Dynamics of random early detection*, Proc. of ACM SIGCOMM (1997).

[102] *Linux Advanced Routing & Traffic Control*, available at http://lartc.org/.

[103] S.H Low, F. Paganini, J. Wang, S. Adlakha, and J.C. Doyle, *Dynamics of TCP/RED and a scalable control*, Proc. of IEEE Infocom (2002).

[104] N. M. Malouch and Z. Liu, *Performance Analysis of TCP with RIO routers*, Proc. of IEEE Globecom (2002).

[105] B. B. Mandelbrot, *Fractals: Form, chance and dimension*, Freeman, San Francisco, 1977.

[106] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang, *Resource ReSerVation Protocol (RSVP)— Version 1 Applicability Statement Some Guidelines on Deployment*, Request For Comments 2208 (1997).

[107] M. May, J-C. Bolot, A. Jean-Marie, and C. Diot, *Simple performance models of tagging schemes for service differentiation in the Internet*, Proc. of the IEEE Infocom (1999).

[108] M. May, T. Bonald, and J. C. Bolot, Proc. of IEEE Infocom (2000).

[109] M. Mellia, A. Carpani, and R. Lo Cigno, *Measuring ip and tcp behavior on ege nodes*, Proc. of Globecom (2002).

[110] M. Mellia, I. Stoica, and H. Zhang, *Packet Marking for web traffic in networks with RIO routers*, Proc of Globecom (2001).

[111] M. Meo, M. Garetto, M. Ajmone Marsan, and Renato Lo Cigno, *On the Use of Fixed Point Approximations to Study Reliable Protocols over Congested Links*, Proc. of Globecom (2003).

[112] D. Miras, *A survey on network qos needs of advanced internet applications*, Working DocumentInternet2 Fellowship on Application QoS Needs (2002), available at http://www.internet2.edu/qos/wg/apps/fellowship/Docs/Internet2AppsQoSNeeds.pdf.

[113] *Mirror Image*, http://www.mirror-image.com/.

[114] A. Misra and T. Ott, *The Window Distribution of Idealized TCP Congestion Avoidance with Variable Packet Loss*, Proc. of IEEE Infocom (1999).

[115] V. Misra, W. Gong, and D. Towsley, *A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED*, Proc. of ACM SIGCOMM (2000).

[116] G. Lo Monaco, F. Azeem, S. Kalyanaraman, and Y.Xia, *TCP-Friendly Marking for scalable Best-Effort services on the Internet*, Computer Communication Review (CCR) **31** (2001), no. 5.

[117] P. R. Morin, *The Impact of Self-Similarity on Network Performance Analysis*, Ph.D. Dissertation, Carleton University (1995).

[118] G. Neglia, D. Bauso, and L. Giarré, *About the Stability of Active Queue Management Mechanism*, Proc. of ACC (2004).

[119] _____, *Active Queue Management Stability in Multiple Bottleneck Networks*, Proc. of ISCCSP (2004).

[120] _____, *AQM Stability in Multiple Bottleneck Networks*, Proc. of ICC **4** (2004).

[121] G. Neglia, G. Bianchi, and V. Falletta, *An Analytical Model of a new Packet Marking Algorithm for TCP flows: preliminary insights*, Proc. of ISCCSP (2004).

[122] G. Neglia, G. Bianchi, and V. Mancuso, *Is Admission Controlled Traffic Self-Similar?*, Proc. of Networking (2002).

[123] _____, *On the Self-Similarity of Admission Controlled Traffic*, Proc. of Globecom (2002).

[124] _____, *Performance Improvements on Self-Similar Traffic using Measurement-Based Admission Control*, Proc. of Internet Computer Symposium (2002).

[125] G. Neglia, G. Bianchi, F. Saitta, and D. Lombardo, *Adaptive Low Priority Packet Marking for Better TCP Performance*, Net-Con (2002).

[126] G. Neglia, G. Bianchi, and M. Sottile, *A new Three Color Marker for TCP flows*, Proc. of CCCT (2003).

[127] _____, *Performance Evaluation of a new Adaptive Packet Marking Scheme for TCP over DiffServ Networks*, Proc. of Globecom (2003).

[128] G. Neglia, V. Falletta, and G. Bianchi, *Is TCP Packet Reordering Always Harmful?*, Proc. of MASCOTS (2004).

[129] ———, *An Analytical Model of a new Packet Marking algorithm for TCP flows*, Proc. of QoS-IP (2005).

[130] *Network simulator*, http://www.isi.edu/nsnam.

[131] K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, Request For Comments 2474 (1998).

[132] K. Nichols and B. Carpenter, *Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification*, Request For Comments 3086 (2001).

[133] T. Ott, J. Kemperman, and M. Mathis, *The stationary behavior of the ideal tcp congestion avoidance*, available at http://web.njit.edu/ ott/Papers/.

[134] T. J. Ott, T. V. Lakshman, and L. H. Wong, *SRED: Stabilized RED*, Proc. of IEEE Infocom (1999).

[135] Ranjan P., E. H. Abed, and R. J. La, *Nonlinear instabilities in TCP-RED*, Proc. of IEEE Infocom.

[136] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, *Modeling TCP throughput: A simple Model and its empirical validation*, Proc. of ACM SIGCOMM (1998).

[137] R. Pan, B. Prabhakar, and K. Psounis, *Choke, A stateless active queue management scheme for approximating fair bandwidth allocation*, Proc. of IEEE Infocom (2000).

[138] K. Park, G. Kim, and M.Crovella, *On the Effect of Traffic Self-similarity On Network Performance*, Proc. of SPIE's International Symposium and Education Program on Voice, Video, and Data Communications (1997).

[139] K. Park, G. T. Kim, and M. E. Crovella, *On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic*, Proc. of the International Conference on Network Protocols (1996).

[140] K. Park and W. Willinger (eds.).

[141] J. Postel, *Internet protocol*, Request For Comments 791 (1981).

[142] ———, *Transmission Control Protocol*, Request For Comments 793 (1981).

[143] K. K. Ramakrishnan, S. Floyd, and D. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, Request For Comments 3168 (2001).

[144] A. Riedl, *Routing Optimization and Capacity Assignment in Multi-Service IP Network*, Ph.D. thesis, Technisce Universität München, 2004.

[145] K. W. Ross, *Multiservice loss networks for broadband telecommunication networks*, Springer Verlag, Secaucus, NJ, USA, 1995.

[146] M. Roughan, A. Erramilli, and D. Veitch, *Network performance for tcp networks, part i: Persistent sources*, Proc. of International Teletraffic Congress (2001).

[147] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu, *On achievable Service Differentiation with Token Bucket Marking for TCP*, Proc. of ACM SIGMETRICS'00 (2000).

[148] *Scalability and Traffic Control in IP Networks*, Conference (2001).

[149] *Scalability and Traffic Control in IP Networks II*, Conference (2002).

[150] N. Seddigh, B. Nandy, and P. Piedu, *Bandwith assurance issues for TCP flows in a Differentiated Services network*, Proc. of IEEE Globecom (1999).

[151] K. Shiomoto, N. Yamanaka, and T. Takahashi, *Overview of Measurement-Based Connection Admission Control Methods in ATM Networks*, IEEE Communication Surveys (1999), pp. 2–13.

[152] S. Sohail and S. Jha, *The Survey of Bandwidth Broker*, School of Computer Science and Engineering, The University of New South Wales, Technical Report UNSW-CSE-TR-0206 (2002).

[153] R. Stankiewicz and A. Jajszczyk, *Modeling of TCP behavior in a DiffServ Network supporting Assured Forwarding PHB*, Proc. of ICC (2004).

[154] W. Stevens, *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, Request For Comments 2001 (1997).

[155] *TANGO Project, Traffic models and Algorithms for Next Generation IP networks Optimization* , http://tango.isti.cnr.it/.

[156] M. Taqqu, W. Willinger, and R. Sherman, *Proof of a fundamental result in self-similar traffic modeling*, ACM SIGCOMM Computer Communication Review **27** (1997), no. 2, pp. 5–23.

[157] M. S. Taqqu, V. Teverosky, and W.Willinger, *Estimators for long-range dependence: an empirical study*, Fractals (1996).

[158] B. Teitelbaum and R. Geib, *Internet2 QBone: A Test Bed for differentiated service*, INET99, The Internet Global Summit (1999).

[159] V.Paxson and S.Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transactions on Networking **3** (1995), no. 3, pp. 226–244.

[160] *Wavelet Code for Long Range Dependence Analysis*, available at http://www.emulab.ee.mu.oz.au/~darryl/secondorder_code.html.

[161] W. Willinger, M. Taqqu, and A. Erramilli, *Stochastic Networks: Theory and Applications, Chapter: A bibliographical guide to self-similar traffic*

and performance modeling for modern high-speed networks, pp. 339–366, Oxford University Press, Oxford, UK, 1996.

[162] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, *Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level*, IEEE Transactions on Networking **5** (1997), no. 1, pp. 71–86.

[163] H. Wu, K. Long, S. Cheng, J. Ma, and Yanqun Le, *TCP Friendly Fairness in Differentiated Services IP Networks*, Proc. of 9th IEEE International Conference on Networks (ICON) (2001).

[164] I. Yeom and A. L. Narasimha Reddy, *Adaptive Marking for Aggregated Flows*, Proc. of IEEE Globecom (2001).

[165] S. Yoneda, *Traffic control and congestion control in IP-based Networks*, ITU-T Recommendation Y.1221 (2002).

[166] S. Tekinay Z. Sahinoglu, *On Multimedia Networks: Self-Similar Traffic and Network Performance*, IEEE Communications Magazine **37** (1999), no. 1, pp. 48–52.

[167] H. Zhang, *Service disciplines for guaranteed performance service in packet-switching networks*, Proc. of the IEEE **83** (1995), no. 10, 1374–1396.

[168] L. Zhang, S. Shenker, and D. D. Clark, *Observations and Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic*, ACM Computer Communications Review (CCR) **21** (1991), no. 4, pp. 133–147.