PERFORMANCE EVALUATION OF PRE-CONGESTION NOTIFICATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Xinyang Zhang

January 2009

PERFORMANCE EVALUATION OF PRE-CONGESTION NOTIFICATION

Xinyang Zhang, Ph.D.

Cornell University 2009

Pre-Congestion Notification (PCN) has been recently proposed at IETF as a practical framework for scalable measurement-based flow control system for support of inelastic real-time traffic. PCN encompasses two complementary functions: admission control of new flows and termination of some admitted flows when the available capacity is unexpectedly reduced after a network failure. Its simple structure and ad-hoc measurement-based nature warrant careful exploration of its applicability.

In this work, we conducted a comprehensive performance evaluation of two proposals that implement PCN, CL-PHB and Single-Marking. We showed that CL-PHB works reliably for both admission and termination control, which in turn demonstrates that the PCN framework is a viable architecture to provide QoS to inelastic real-time traffic. We also studied Single-Marking's performance tradeoffs associated with its implementation benefits, and analyzed its applicability range.

Our work also includes the development of several novel fluid models of different complexities and the investigation of their accuracy and applicability to various aspects of PCN admission and termination control. Aside from providing a tool to gain more insight in PCN behavior, we believe that some of our analysis is of a more general nature. Specifically, we believe that our results support a conjecture that while bottleneck behaviors are possible to predict with high accuracy using these models, per-aggregate behavior of admission control systems cannot be accurately predicted with any fluid model resulting in a bounded estimation error.

BIOGRAPHICAL SKETCH

Xinyang (Joy) Zhang was born on Jun 20, 1979 in Harbin, China. She attended University at Albany (State University of New York) from 1998 to 2002 and graduated with Master of Science in Computer Science. She came to Cornell University in the Fall of 2002 and began her doctoral studies in Computer Science. She pursued her research in networking under the guidance of Prof. Paul Francis and Dr. Anna Charny. She also pursued a minor in Statistics. Since 2006, she has been regularly working as an intern for Cisco System, Inc. at the Boxborough, MA. Her research interests include in Quality of Service and Internet Routing.

To my parents

ACKNOWLEDGEMENTS

First and foremost, I am extremely grateful to my thesis advisors Prof. Paul Francis and Dr. Anna Charny, for all their guidance and stimulation throughout this research work. The joy and enthusiasm they have for research was motivational for me, even during tough times in the Ph.D. pursuit.

I would like to thank other members of my committee, Prof. Ken Birman and Prof. Steven Schwager, for their insightful feedbacks and helpful comments. I also acknowledge my other colleagues at PCN working group (Philip Eardley, Bob Briscoe, Joe Babiarz, Michael Menth, and Vassilis Liatsos) with whom I have interacted a lot during the course of this research.

I am very grateful to Prof. Ravi for inspiring me to pursue a research career when I was an undergraduate, and for all his guidance and support throughout my student life. I feel very fortunate to have great friends like Stefanie, Lyublena, Barry, Ganesh, Kamal, Manpreet, Ruijie, Xiangyang, Vidya, Hitesh, Saikat and Vivek who made my stay at Cornell an enjoyable period. I also would like to thank Becky, Bill and Stephanie for all their help during my graduate study in Department of Computer Science.

Many thanks to Ting, who has always been there for me throughout every up and down in my life, and provides great encouragement and friendship. I am of course very grateful to my parents. They gave me the possibility to grow according to my talents and supported me in all my pursuits. I am forever indebted to them for their love, understanding, encouragement and endless patience when it was most required. And last but not least, Ralph has been a constant in my life since I came to Unite States, and he has certainly been a source of great inspiration. So, a very special “Thank you” to you, Ralph.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CL	CL-PHB Algorithm
DiffServ	Differentiated Services
DSCP	DiffServ Code-Point
ECMP	Equal Cost Multi Path Routing
EF	Expedited Forwarding
EWMA	Exponential Moving Average
IE-Aggregation	Ingress-Egress Aggregation
IntServ	Integrated Services
MBAC	Measurement-Based Admission Control
MPLS	Multi-Protocol Label Switching
PCN	Pre-Congestion Notification
PHB	Per-Hop Behaviors
QoS	Quality of Service
RED	Random Early Detection
RSVP	Resource ReSerVation Protocol
SM	Single-Marking Algorithm
VoIP	Voice over IP

CHAPTER 1

INTRODUCTION

Today, real-time applications are typically supported by a combination of queuing/scheduling differentiation between traffic classes (DiffServ) and *over-provisioning* (e.g. the amount of real-time traffic served in strict priority is smaller than the link bandwidth). To ensure these conditions, provisioning typically assumes the knowledge of the expected traffic matrix and network topology. However, unexpected events (e.g. an earthquake in California) may result in fast changes to the traffic matrix, known as "flash-crowds". At such times, the change of expected traffic patterns may cause severe congestion, resulting in uniformly poor service for *all* flows sharing the affected link. Furthermore, provisioning to ensure resiliency under multiple failures becomes prohibitively costly, especially as the amount of real-time traffic such as video increases.

These issues, which have long been recognized, provide the key motivation for admission control, which is intended to limit the admission of new calls to ensure that all admitted calls can maintain an acceptable level of Quality of Service (QoS). Over the years, a number of solutions of different complexity levels have been developed (e.g.[17][18][13][32][9][21][24] [31][34][36][7][25][14][4]).

However, admission control by itself may not be sufficient. When unplanned network failures occur, the traffic will be rerouted on a backup/new path. If bandwidth on these new paths is insufficient to support rerouted flows, then *all of the already admitted* real-time traffic on affected links may suffer from the resulting QoS degradation. For example, phone users typically hang up after 1-2 seconds of severe service degradation, so it is desirable to quickly terminate some calls while preserving quality of service for the remaining ones, rather than potentially have all callers on an affected link hang up.

Recently, a new Pre-Congestion Notification (PCN) framework has been proposed at IETF [27] primarily for support of inelastic real-time traffic. PCN encompasses two complementary functions - admission control of new flows and termination of some admitted flows when the available capacity changes unexpectedly. This framework assumes that each link in the network is associated with two utilization thresholds - an *admission-threshold*, which limits the amount of PCN traffic admitted under "normal" circumstances, and a *termination-threshold*. When PCN traffic exceeds the termination-threshold due to some unexpected circumstances, enough traffic is terminated to ensure that PCN utilization does not exceed termination-threshold. The important assumption is that the admission-threshold is less than the termination-threshold. It is assumed that the interior routers meter aggregate incoming PCN traffic against the two thresholds, and use two different markings to communicate the state of the system to the edge routers of the PCN domain. The edge devices make their admission and termination decisions based on this marking (more details in Chapter 3).

The two-threshold architecture of PCN is designed to provide architectural flexibility for implicitly reserving backup bandwidth which can be used to accommodate a configured amount of PCN traffic rerouted due to failures. However, this space separation of the admission and termination thresholds has an additional benefit of dramatically simplifying the job of admission algorithms. Indeed, over-admitting some limited amount of traffic over the admission-threshold does not carry the penalty of loss, but rather can be viewed as a bounded policy violation. As long as the error is small enough to not trigger termination, limited over-admission errors do not result in catastrophic consequences, and hence may be tolerable in practice. Likewise, as long as the error of the termination algorithm does not bring the utilization below the admission-threshold, it appears reasonable to think that a certain amount of over-termination may be tolerable, if it has to compare to loss of quality or service for all traffic on the link. This inherent tolerance of the PCN framework to such errors potentially allows simple ad-hoc algorithms to be practically sufficient.

1.1 Contributions

The primary focus of this dissertation is the comparative evaluation of two admission control algorithms proposed for implementing the PCN framework of [27]. One, proposed in [5], referred to as CL-PHB (CL), has its roots in veritable MBAC research. Even so, its practical performance limitations (e.g. appropriate parameter setting and sensitivity, levels of aggregation sufficient for practically tolerable performance, etc.) need to be understood. The second algorithm, proposed in [1], referred to as Single-Marking (SM), is a simple ad-hoc algorithm which was motivated by the desire to use just one marking and metering function at the internal routers for both admission and termination functions, (which is important, as metering and marking are performed in the data path of high-speed routers), and needs only a single marking encoding (which saves valuable real-estate in the packet header). In addition, its metering function is based on a token bucket implementation readily available in today's routing equipment. Yet, its ad-hoc nature warrants careful exploration of performance tradeoffs associated with the implementation and deployment benefits - the task we address in a large part of this work.

Our findings are twofold: First, we showed that CL-PHB works reliably for both admission and termination control, even in the presence of packet loss. Second, we showed that for Single-Marking, while giving adequate performance in most cases, the simpler mechanism does come at a price, especially in termination. At the same time, our results imply that as long as sufficient levels of aggregation are present, and if a larger termination error is acceptable, SM may nevertheless be a practicable solution.

Overall, the performance of two algorithms demonstrates that the PCN framework is a viable architecture to provide QoS to inelastic real-time traffic.

Our evaluation is a mix of theoretical analysis and extensive simulations. Many of the performance issues are fundamentally due to packet-level level effects and can therefore only be revealed when simulating with packet-based simulators. Yet, running packet-based simulation is very resource- and time-consuming, and greatly limits the ability to conduct large scale experiments. At the same time, a range of performance issues can be effectively represented by "fluid" approximation models. In this work we developed two separate simulation methods - one is a conventional discrete-event packet-based simulator, and the other is a novel fluid model that consists of both theoretical formulation and approximations. We used the two types of simulators in conjunction with each other and validated the accuracy of the fluid simulator against its packet counterpart. Our fluid simulator not only enables us to run large scale PCN experiments, but also provides some insight on the applicability of fluid models in general. In particular, we found that for admission control, the fluid simulator works remarkably well in modeling the various properties of bottleneck behavior, and yet it fails to predict per-aggregates' outcomes with bounded accuracy. Our analysis showed that the insufficiency may apply to fluid models in general, which leads us to hypothesize that an admission fluid model is fundamentally incapable of predicting the complete per-aggregate outcome of its packet counterpart in the long run.

To summarize, the key contributions of this dissertation are 1) a practical assessment of several PCN algorithms through careful study of their performance evaluations and understanding the associated practical tradeoffs between the algorithms, and 2) the development of a theoretical approximation of the fluid model for the PCN algorithms and the analysis of the applicability of the fluid model in a general context.

1.2 Organization

This work is structured as follows. Chapter 2 gives an introduction to the branch of quality of service (QoS) study that is most relevant to PCN, including related QoS mechanisms, architectures and admission control algorithms. Chapter 3 gives an overview of the PCN architecture and the algorithm descriptions of CL-PHB and Single-Marking. Chapter 4 contains the details of our evaluation methodology and comparative performance analysis of the two PCN algorithms. Chapter 5 describes our novel fluid simulation models, including their construction, accuracy and applicability. Finally Chapter 6 summarizes this work and provides directions for future work.

CHAPTER 2

BACKGROUND

Traditional IP-networks operate on a *best-effort* model, in which the network tries to deliver packets with no guarantee that the packets will arrive within certain time limits, or arrive in the same order as they are sent, or even arrive at all. Given that the early usage of the IP network was to support non-interactive data applications that are mostly throughput focused (e.g. file transfer), the best-effort service model was adequate, as long as the end systems had some mechanisms in place to ensure delivery reliability and can react to network congestion (most notably, TCP).

Recently, there is an increasing trend of using the IP-network to carry applications such as voice over IP (VoIP), video streaming, online gaming, etc. These applications, which belong to the category of *real-time* application, all have certain strict requirement for packet delivery quality. For instance, a VoIP application needs the timeliness of delivery, industry standards suggest that ~150 ms of end-to-end one-way delay is sufficient to ensure user satisfaction, while delay exceeding 400 ms is considered unacceptable [11]. For video streaming, the loss of a single IP packet may result in a visible impairment. Despite the fact that a sophisticated end system implementation might relax some quality constraints, the essential characteristic of real-time applications is that they demand certain delivery quality assurance ***from the network*** in order to guarantee end-to-end service satisfaction.

Clearly, a network with a best-effort model is not sufficient to support real-time applications. In addition, different applications have different quality requirements. Given the wide space of requirements, we need a service model that has several service classes, each available to meet the needs of some set of applications. This requires the network to be able to sort the traffic into appropriate classes, enforce that the traffic is treated appropriately at each hop along the path, and when the network resources run low, reject the request of some services to ensure the quality of the rest. A network that provides the above described services is said to support Quality of Service (QoS), and it is achieved by implementing a set of QoS mechanisms on all (or subset of) its routers in the context of a QoS architecture.

In fact, QoS is a vast research area with many aspects. In the rest of the chapter we only review the background and related work that are most relevant to PCN. More specifically, the focus of this chapter is three-fold: 1) to review the basic QoS mechanisms, most of which are used in PCN, 2) to describe the evolution in QoS architectures, and in doing so provide the context and the motivation for PCN, and 3) to outline some research that influences the design of the PCN algorithms, especially CL. The next section starts with basic QoS mechanisms.

2.1 QoS Mechanisms

Basic QoS mechanisms are either implemented in the routers' *data plane* or in the *control plane*. A data plane operation refers to the set of functions that a router performs on every passing packet, such as packet forwarding lookup. Such operations are typically implemented in hardware for maximum processing speed. A control plane operation, for example a routing protocol, involves the set up and maintenance of the state required by the data plane, and tends to be implemented as a software process.

Typical data plane QoS mechanisms include classification, metering, marking, scheduling and dropping. We will discuss each of these mechanisms in detail below.

Classification Traffic classification is the process of identifying packets into classes so that actions can be applied to these packets. A class might be as fine-grained as a flow, or as coarse-grained as an aggregate of flows. Classification is usually based on certain information in the packet's header, such as the IP address, port number, DSCP code point, etc. The actions to apply after classification could be another QoS mechanism.

Metering Traffic metering is the process of measuring the rate characteristics of a traffic stream, whether it be a flow or an aggregate of flows. Simple metering could be performed by taking the statistics of packets transmitted for a traffic stream over a certain interval. Alternatively, the metering function might consist of comparing a traffic stream against a QoS profile, to check if the stream conforms to a given profile, or identify the portion of traffic that exceeds the profile. An example of this type of metering is called token-bucket, and is illustrated in Figure 2.1. A token-bucket configured with a token-fill-rate of R and token-bucket-depth of D corresponds to a traffic profile of the average rate of R and burst size of D.[1] Those packets that find no available token upon their arrival will be identified as out of profile traffic.

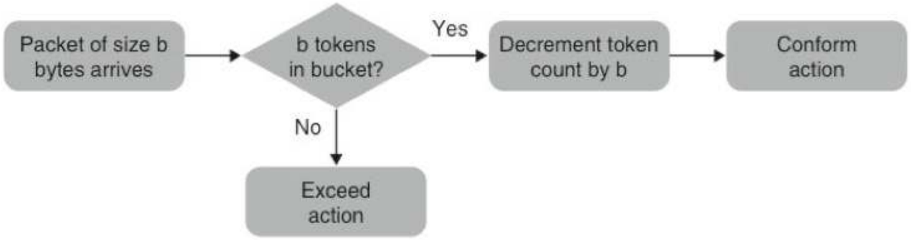

Figure 2.1: Token bucket metering

Marking Packet marking is the process of setting the value of the fields in the packet headers so that the traffic can easily be identified later.

Queuing/Scheduling/Dropping When recourse contention occurs, such as packets arriving faster than they can be transmitted, then packets are queued before they can be serviced. Queuing and scheduling are used in conjunction to prioritize some types of traffic over others, and to control queuing delays and bandwidth assurances to difference traffic types. For instance, delay sensitive traffic such as voice and video might use

[1] More precisely, a traffic stream that conforms to the token-bucket is allowed to send at most $R + D$ bytes at any time.

a separate queue than data traffic, and can be served with a high priority. Queues usually have finite space (not only because of physical router buffer memory, but also because a large queue might imply large queuing delay); if packets arrive faster than the queue's serving rate, eventually the queue will overflow and some packets will be dropped. The router may implement simple tail drop, or more complex preferential dropping in that it prefers dropping certain classes of traffic to others. Random early detection (RED) [35] is another dropping policy; it detects congestion before queue overflows and uses drops to provide feedback about congestion to the end system.

Control plane QOS mechanisms consist of all functions and operations performed by the network to set up and maintain the state required by the data path, as well as admission control and resource reservation. Control plane QOS functions are typically implemented by a signaling protocol. An example of a widely used control plane signaling protocol is RSVP [29].

QoS mechanisms are the basic building blocks of a QoS solution. In the next section, we will review several QoS architectures, which provide the context and define the structure within which we deploy QoS mechanisms to deliver quality assurance.

2.2 QoS Architectures

IntServ Architecture

Integrated services [28], or IntServ is a QoS architecture that supports admission control and resource reservation. IntServ is a fine-grained architecture in that it provides service level assurances to applications by explicitly managing bandwidth resources and schedulers on a per flow basis.

With IntServ, packet classification is performed on a per flow basis; at each IntServ-router, each packet is identified as part of a particular flow using the 5-tuple, that is, source and destination IP addresses, source and destination port numbers and the IP protocol number. To carry out this classification requires storing per-flow state at each IntServ-router. After classification, scheduling and queueing can also be managed on a per-flow basis to ensure the flow receives the appropriate service. IntServ defines three types of services: 1) Guaranteed service to support the traffic that requires low-delay, low-jitter, low-loss, assured bandwidth requirement, 2) Controlled load service to support the traffic that requires a QoS closely approximating the QoS that the same flow would receive from an unloaded network, and 3) Best effort service for the default traffic that requires neither of above two service types.

IntServ ensures there are sufficient resources provided to a flow by performing admission control at each IntServ-router. The flow provides its traffic specification (e.g. average rate and peak rate) and its requirement specification (e.g. Guaranteed service or Controlled load service). If there are sufficient resources at each hop along the way, the flow is admitted, else the flow is rejected. The most common practice is to implement the admission control and the subsequent resource reservation through the signaling protocol RSVP. This requires per-flow signaling and per-flow control plane state at each IntServ hop.

The main problem with IntServ is scalability, since it requires performing per-flow processing and storing per-flow control plane state (e.g. path and reservation state) and data plane state (e.g. classification and queueing state) at each IntServ router. The amount of state in each router is in the order of the number of flows passing through it, which can be rather high for core routers that connect to high-speed links. This prevents IntServ from being widely deployed.

DiffServ Architecture

Scalability concerns with IntServ lead to the design of Differentiated Services [33] or DiffServ. In contrast to IntServ, DiffServ is a more coarse-grained, class-based architecture, which achieves scalability by performing and maintaining per-flow QoS functions and states only at the edge of the DiffServ network.

Packets coming into the DiffServ domain may already be DiffServ-marked. If not, upon entering the network, the traffic is classified using simple or complex (per flow) classifications into a limit number of traffic classes, and then is marked with an appropriate DiffServ Code-point (DSCP). The subsequent DiffServ routers, or the internal (non-edge) router in the domain only need to perform a simple classification based on DSCP, hence there is no need to store per-flow state and perform per-flow classification.

Once the packets are appropriately marked and easily identified within the network, the goal of the internal routers is to ensure the quality of service is met for a given class by applying queuing and scheduling mechanisms on per-class basis. The DiffServ architecture uses the concept of Per-Hop Behaviors (PHB) that abstractly defines how the traffic should be treated at each hop. One example, Expedited Forwarding (EF), is used to support the traffic class that requires low-delay, low-loss and assured bandwidth (like the guaranteed service in IntServ), and is typically implemented using a strict priority queueing/scheduling mechanism.

Unlike Intserv, DiffServ configurations are provisioned, rather than being set up by a signaling protocol. In addition, in the absence of admission control, DiffServ needs to ensure that per-class resources, such as bandwidth, are provisioned at each hop by per-class capacity planning.

IntServ over DiffServ

DiffServ does not support an explicit mechanism for admission control. If the actual traffic load for a DiffServ class exceeds the available bandwidth allocated to that class, then all the flows in that class will suffer service quality degradation. Without admission control to ensure QoS, the bandwidth for any DiffServ class must be over-provisioned with respect to the peak load, which can be costly especially in failure situations.

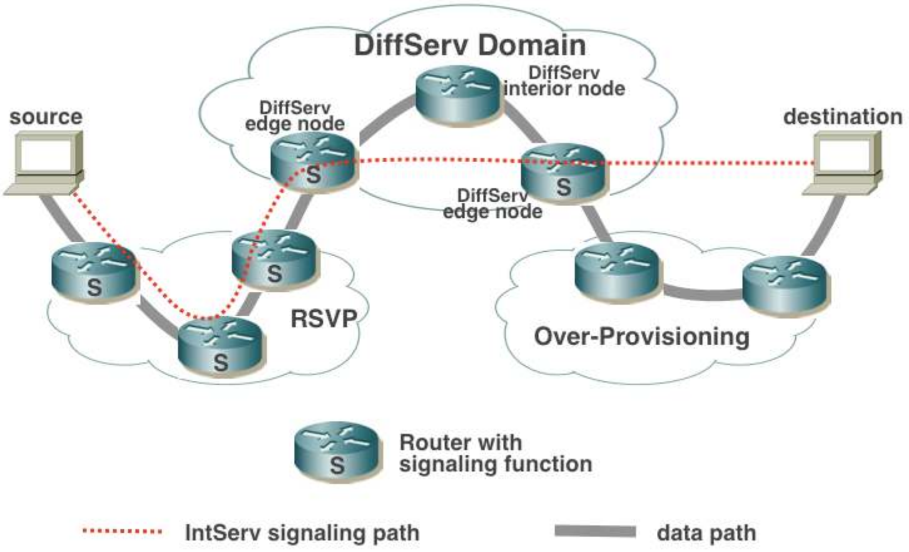

Figure 2.2: IntServ over DiffServ architecture

IntServ over DiffServ in [38], defines the architecture of using a DiffServ network in the end-to-end path of an IntServ reservation. Figure 2.2 shows three networks under in an end-to-end IntServ region. The left-most network follows IntServ architecture and uses RSVP to signal per-flow resource requirements and per-flow classification and scheduling at its routers along the path. The right-most network represents a network that relies on over-provisioning, hence the end-to-end RSVP just hop over the network and assume sufficient bandwidth is provided. The middle network is a DiffServ network. On the data path, the integration between the IntServ and DiffServ networks can be achieved by mapping a flow of an IntServ service into an appropriate DiffServ Per-

Hop Behaviors (PHB), e.g. Intserv Guaranteed Service maps to Expedited Forwarding PHB. At the edge of a DiffServ domain, per-flow states are kept so that packets can be identified into flows, then in turn mapped in to the appropriate class and marked with the corresponding DSCP.

On the control path, the admission control over the DiffServ domain can be handled in several ways. One is to have all the routers in the DiffServ domain to be RSVP-aware, and participate in RSVP signaling and admission control at each hop. With this scheme, the data path operation such as classification and scheduling are still done on a per-class basis. This approach takes advantage of the IntServ admission control, and avoids IntServs unscalable per-flow data path operations. Yet, it does not address the control plane scalability problem, since every router is involved in RSVP at per-flow level. The second approach is trying to reducing the control plane scaling concern by aggregating individual RSVP flow reservations over aggregate RSVP reservations [12].

Both these approaches can be considered as parameter-based admission algorithms. Such approaches use parameterized traffic descriptors to describe the flow's requirement, and comparable descriptors representing the network's available resources. The performance of such algorithm depends very much on the accuracy of the descriptor, and often has the problem of balancing between providing QoS guarantees and efficient bandwidth utilization. The counterpart of parameter-based algorithms, are measurement-based admission algorithms (MBAC), which are known for their ability to achieve higher network utilization. This leads to the question of using measurement-based admission control over a DiffServ domain, which is essentially what PCN is about. PCN is a measurement-based control (admission + termination) over a DiffServ domain, typically deployed in the IntServ over DiffServ context. In the next section, we will review some of existing work on measurement-based admission algorithms.

2.3 Measurement-Based Admission Algorithms

Before referencing relevant literature in the area of the MBAC, we should first discuss the general applicability of the measurement-based admission control algorithms. The fundamental assumption of MBAC is that past traffic measurement accurately predicts future behavior. In practice, this assumption does not always hold, especially when the aggregation (number of flows) on the links is low. Therefore, MBAC cannot provide a worst-case bound for any quality metric such as delay or loss. Consequently, it should only be used to provide services that do not require hard guarantees, such as controlled-load service. (Recall, we describe the controlled-load service in Section 2.2 as the service type that is suitable to support adaptive real-time applications which require QoS without a worst-case guarantee.)

MBAC research is a vast and well-developed field. Here we do not attempt to give a comprehensive overview of existing MBAC work. Instead we briefly mention some of the results most relevant for PCN, and point the reader to appropriate surveys. An excellent overview and comparison of "traditional" hop-by-hop MBAC research can be found in [18], where it is argued that there is little difference between different MBAC algorithms (formula-based or ad-hoc) with respect to the utilization levels they can achieve. At the same time, none of these algorithms reliably meet loss targets. As a result, the authors of [18] conclude that little can be gained by refining the details of the MBAC algorithms, effectively closing the field. Why then, one may ask, is yet another study on the properties of yet another MBAC algorithm necessary?

One answer to that question lies in the desire to improve the scalability of per-hop signaling used in the algorithms studied in [18], providing motivation for a large body of work on the so-called end-point admission control. In this immediate ancestor of PCN, probe packets are marked by the routers, and this marking is used to make admission decisions at the end nodes (or in some studies by the edge-nodes of a network region). A comprehensive survey and simulation analysis of different marking algorithms can be found in [21], where it is concluded that such probe-based end-point admission control is a reasonable low-cost alternative to hop-by-hop MBAC. Most relevant for PCN are the ideas in [31], which in turn motivated the work in [7][25]. The results of this work directly motivated the choice of the marking algorithms first proposed for PCN in CL.

CHAPTER 3

PCN ARCHITECTURE AND ALGORITHMS

3.1 Overview of PCN Framework

In a typical deployment scenario [27] (as illustrated in Figure 3.1), a signaling protocol (e.g. RSVP or SIP) controls admission of flows end-to-end outside the PCN domain, treating the entire PCN domain as a single signaling "hop". PCN controls admission and termination decisions over this hop. Per-flow signaling state is maintained only at the PCN edge (where the flow may in itself be an aggregate), interior routers have no per-flow, or even per-ingress-egress aggregate state. Although not strictly required by the architecture, it is typically assumed that PCN traffic is served at the highest priority. The remaining bandwidth is assumed to be used by non-PCN traffic and hence is not wasted.

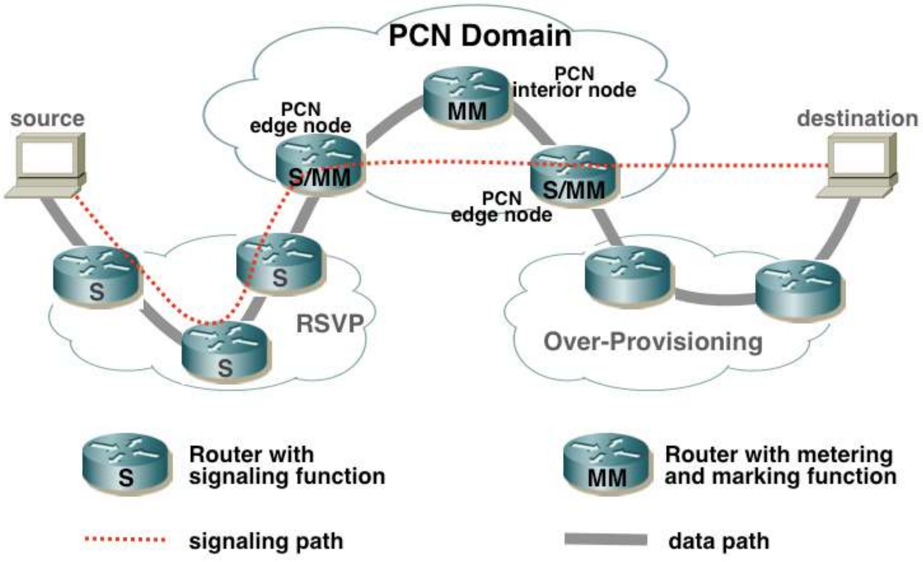

Figure 3.1: Overview of PCN framework

Each link in the PCN domain is configured with two rate thresholds, the ***admission-***

threshold and the ***termination-threshold***. Admission of new flows stops when the admission-threshold is reached. Flows are terminated if the termination-threshold is exceeded. The relationship between the load condition and PCN control action is summarized in Figure 3.2. Interior routers in the PCN domain meter the aggregate amount of PCN traffic on a per-output link basis against the two thresholds. If the admission-threshold is exceeded, some packets are admission-marked. When the termination-threshold is exceeded, some packets are termination-marked. PCN edge nodes make their admission and termination decisions based on the marked packets. The remainder of this chapter describes two algorithms for implementing this high-level framework.

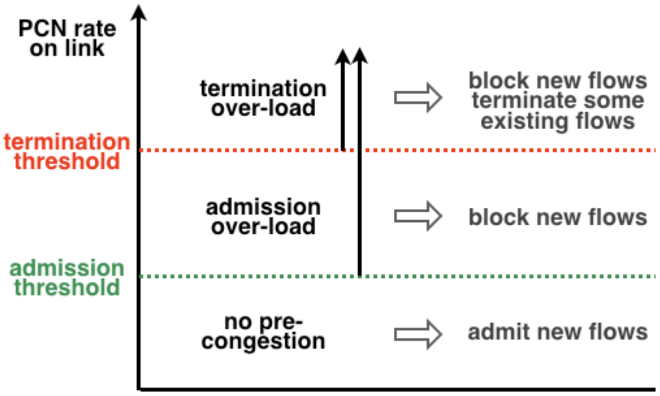

Figure 3.2: PCN control action

3.2 CL-PHB Algorithm

Admission

In CL, a PCN-enabled interior router maintains a *virtual-queue* for each of its links, along with a real queue used for PCN traffic. An arriving PCN packet simultaneously enters the virtual-queue and the real queue. The virtual-queue is drained at the rate equal to admission-threshold. The router admission-marks incoming PCN packets with the probability given by a piecewise-linear function of the virtual-queue depth: no packets are marked until the virtual-queue has reached a lower threshold (*vq-min-threshold*), all packets are marked when the virtual-queue exceeds the upper threshold (*vq-max-threshold*), and between the two thresholds the marking probability linearly increases from 0 to 1. A third threshold (*vq-upper-limit*) is used to limit the size of the virtual-queue to some maximum number. When the virtual-queue reaches vq-upper-limit, newly arriving packets are not added to the virtual-queue. Note that if a steady overload occurs, all PCN packets will be admission-marked. The egress of the PCN domain measures the so-called Congestion-Level Estimate (*cle*) which is the fraction of admission-marked PCN traffic over the total PCN traffic. The cle is measured and computed on an interval basis, then smoothed as a weighted moving average. We refer to the interval sample as *interval-cle*, the smoothed value as *ewma-cle* and the weight parameter as *ewma-weight*. The ewma-cle is periodically signaled to the corresponding ingress. For example, if RSVP is used outside the PCN domain, ewma-cle may be piggy-backed on an RSVP message traveling between this particular ingress-egress pair. The ingress node admits a new flow to a given egress if the latest ewma-cle it received from that egress is less than a chosen *cle-threshold*, and blocks admission otherwise.

Termination

The termination algorithm in CL is as follows. Interior routers in the PCN domain maintain a *token-bucket* for each of their links. The token-bucket is filled at a rate equal to the termination-threshold. The router termination-marks incoming PCN packets if it finds no tokens in the bucket. Note, at subsequent hops only ***unmarked*** PCN traffic is metered against the termination threshold. When the egress node detects a termination-marked packet, it measures the rate of PCN traffic from a given ingress *excluding* any termination-marked packets, again on an interval basis. This amount of unmarked traffic is termed *sustainable-rate*. Sustainable-rate reflects the amount of traffic of an ingress-egress PCN aggregate that “safely got through” without exceeding the bottleneck termination threshold along the path from ingress to egress. The sustainable-rate measurement is then reported for that ingress. When the ingress receives this sustainable-rate, it measures the current PCN traffic rate it sends to the appropriate egress (*sending-rate*), and terminates sufficient PCN flows to bring its sending-rate to that egress down to the sustainable-rate if necessary.

3.3 Single-Marking Algorithm

SM is motivated by the desire to use just one marking and metering function at the interior routers for both admission and termination functions, instead of two, which is very attractive as it substantially reduces the work expected to be performed in the data path of the high-speed routing equipment. In addition, as its name indicates, Single Marking needs only a single marking encoding instead of two required by the other PCN proposals, thus saving valuable real estate in the packet header (which is especially valuable in the MPLS environments [6]). Furthermore, its metering function can be implemented by a simple token-bucket readily available in today’s routing equipment.

Admission

In SM, a PCN interior router implements *excess rate* admission marking instead of virtual-queue-based marking for its admission function. It meters traffic exceeding the admission-threshold and marks excess traffic (e.g. using token-bucket-based marking in exactly the same manner as the termination function of CL). Note that in the case of a steady overload, the amount of admission-marked traffic corresponds to the excess rate of PCN traffic over its admission-threshold (which differs from the virtual-queue marking for which steady overload results in 100% of PCN traffic being marked). Just as in CL, the egress measures and computes the ewma-cle every measurement interval and reports it to the ingress, which stops admission of new flows when it exceeds the cle-threshold.

Termination

In SM, PCN internal routers do not implement termination marking at all, and no explicit termination-threshold is set. Instead, the termination-threshold is *implicitly* assumed to be a PCN domain-wide factor (***K***) times the admission-threshold on all links. The egress node measures (per ingress) the amount of PCN traffic that remains unmarked just as in the termination function of CL (except here it is measured against admission-marked packets). The egress then reports to the ingress the amount of unmarked traffic along with ewma-cle. To decide whether termination is needed, the ingress first multiplies the unmarked traffic by the system-wide constant K described above to obtain the effective *sustainable-rate* for this egress, and if necessary, terminates enough flows to keep its sending-rate below the sustainable-rate.

CHAPTER 4

PERFORMANCE EVALUATION OF CL-PHB AND SINGLE-MARKING

4.1 Evaluation Setup

4.1.1 Performance Metrics

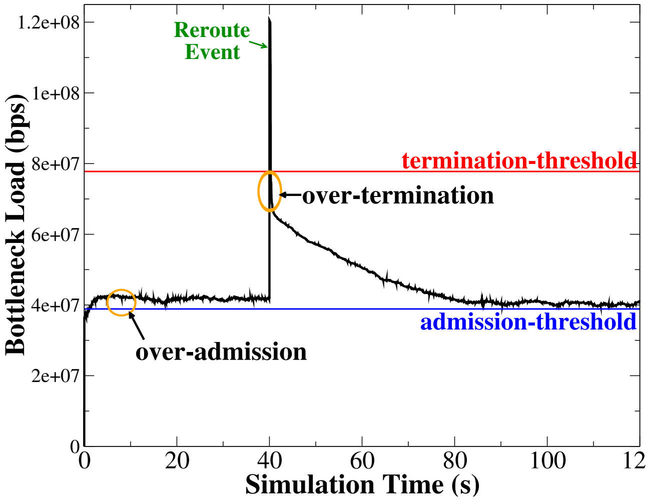

Figure 4.1: An example of bottleneck behavior with PCN

We start with describing our performance metrics and the intuition behind the choice of these metrics through an example. Figure 4.1 illustrates the load on a bottleneck when both PCN admission and termination controls are enabled. An effective admission algorithm should maintain the load (represented by the black line) around the admission threshold under the “normal” conditions (up until 40s of simulation time). At about 40s, a failure event elsewhere in the network causes a sharp surge of traffic on the link in question, at which point the termination mechanism should bring the load just below the termination threshold. After that, natural flow departures over time reduce the load down to the admission threshold. One way to evaluate a PCN algorithm is to measure the percent deviation of the actual load achieved in the experiment from the admission and termination thresholds respectively. We term these "***over-admission***" and "***over-termination***" percentages respectively. More precisely, our experiments measure the actual achieved load at the granularity of 50 ms, and then compute the average of these 50ms rate samples over the duration of the experiment. We also compute the deviation of the load from the average. In all our experiments the deviation was consistently small (as we will show in Section 4.2.2), which allowed us to report experimental averages.

A note is due on the relevance of these metrics. As mentioned in the Chapter 1, the space separation of the admission and termination thresholds, both of which are assumed to be below the link speed, makes traditional loss-based performance evaluation metrics less relevant for admission control, as small over-admission is not likely to bear the cost of packet loss and can be viewed as a bounded policy violation. However if over-admission errors were large enough to trigger termination, and over-termination errors were to cause the amount of traffic to fall below the admission threshold, then the system will oscillate between admission and termination states, and newly admitted flows may be immediately terminated under normal conditions. Hence, our goal is to investigate, for a given PCN algorithm, the range of these errors, and in doing so provide practical guidelines on how far apart the admission and termination thresholds must be configured to avoid system oscillation.

Clearly, the magnitude of over-admission and over-termination errors may depend on many factors, such as configuration parameters, traffic models, differences in round trips, the levels of aggregation, etc. To understand the sensitivity of the studied algorithms to these factors, we used a range of network topologies, traffic types and simulation models, which are described in detail in the remainder of this section.

Another metric of interest with respect to termination is the algorithm's reaction time, which is defined as the time it takes for the termination mechanism to bring the load below the termination-threshold after a failure/reroute event. The importance of this metric is intuitive, and like reason we need the termination function in the first place, it is to quickly remove some flows in case of such events, and ensure the QoS of the remaining ones; a termination control with a long reaction time loses its purpose. In practice, a reaction time within 2 seconds is considered to be acceptable.

We also note here that the time and space separation of the areas of admission and termination algorithms implied by Figure 4.1 enables us to effectively discuss the properties of these algorithms independently of each other in most cases. We emphasize, however, that although we report our results separately for admission and termination algorithms, in our experiments we ran both algorithms simultaneously, and our results actually reflect the two algorithms working in tandem.

4.1.2 Network Topologies

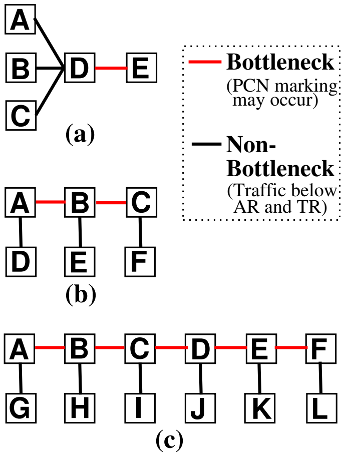

Figure 4.2: Network topologies

Most of our simulations used two types of topologies, shown in Figure 4.2. The MultiLink (MLK) topology in Figure 4.2(a) models the case when flows from several ingresses (A,B,C) going to the same egress (E) share a single bottleneck link DE. By varying the number of ingresses and the propagation delays from different ingresses to node D, we were able to explore sensitivities to the differences in propagation delay, levels of ingress-egress aggregation, etc. Figures 4.2(b) and (c) show a class of multi-bottleneck topologies termed PLT-N, with N representing the number of potential bottleneck links. A simple traffic matrix on these topologies, has a single "long-haul" ingress-egress aggregate traversing all the N bottlenecks, while the other N ingress-egress aggregates traverse a single bottleneck link each, exiting at the next "hop", emulating local cross-traffic. We also experimented a class of more generalized traffic matrices, which we term G-PLT-N, that allows the aggregates to traverse any number of bottlenecks up to N. For instance, a G-PLT-5 topology consists of 5 (potential) bottlenecks, and 15 (potential) aggregates traversing bottlenecks ranging from one to all five. These classes of topology model the cross-traffic and multiple congestion points that can occur in the hierarchically structured networks deployed by many network providers.

Despite the seeming simplicity of these topologies, we believe they are sufficiently versatile not only to represent a wide range of situations expected in real networks, but also to allow stressing the PCN algorithms in many dimensions, as we show below. We believe that through comprehensive evaluations of this limited class of topologies, we can gain insights into performance issues relevant to the more general settings expected in practice.

4.1.3 Traffic Types

Table 4.1 summarized the parameters of all traffic types we used. The first two rows correspond to an older version of G.711 voice codec (with and without silence compression), and are referred to as just "CBR" and "VBR" throughout the paper. We also simulated traffic mixes drawn from other voice codecs shown in rows 3-8 of the table. Specifically, the "MIX" used in this paper consists of equal proportions of all voice codecs from Table 4.1. Other mixes yield very similar results and are not reported here.[1] Where applicable, the on and off periods were exponentially distributed with the specified means.

Table 4.1: Simulated codecs

Codecs	Packet Size (B)	Inter-Arrival Time (ms)	On/Off Ratio	Rate (Kbps)
"CBR"	160	20	1	64
"VBR"	160	20	0.34	21.75
G.711 CBR	200	20	1	80
G.711 VBR	200	20	0.4	32
G.711 CBR	120	10	1	96
G.711 VBR	120	10	0.4	38.4
G.729 CBR	60	20	1	24
G.729 VBR	60	20	0.4	9.6
"SVD"	1500	1	0.34	4096
"VTR"	1500	1	—	769

The last two rows of the table summarize our "video" traffic models. The first one, referred to as Synthetic Video (SVD) is in fact just an on-off traffic model with video-like mean-to-peak ratio and mean rate approximating that of an MPEG-2 video stream. We expect this model to be more challenging for a measurement-based algorithm than the actual, much "smoother" MPEG video, and make no attempt to simulate any other aspects of a video stream in this model. We expect that "good" or "reasonable" performance on SVD will likely indicate that MPEG traffic should perform at least as well. Our second "video" model (referred to as Video Traces, or VTR) is more realistic: we used a publicly available library of frame size traces of long MPEG-4 and H.263 encoded video obtained from.[2] Each trace is roughly 60 minutes long, consisting of a list

[1] Note, our MIX traffic model consists of voice traffic only, not a mix of voice and video traffic because we assume in practice voice and video traffic will be run from separating queues.

[2] http://www.tkn.tu-berlin.de/research/trace/trace.html

of records in the format of <FrameArrivalTime, FrameSize>. Among the 160 available traces, we randomly picked two. We then randomly chose segments with a mean duration of 1 minute from this trace, and have verified by direct simulation that the expected rate of these 1-minute segments is roughly the same as the trace's average. Finally we packetize each frame using a packet size equal to the one in SVD. Traffic characteristics of VTR are summarized in the last row of Table 4.1.

4.1.4 Traffic Generation

Apart from network topology and traffic types, another important simulation parameter is bottleneck call-arrival-rate. However, instead of specifying a value for call-arrival-rate for each experiment, we prefer to use another, more illustrative parameter, that we term *demand-load*. Demand-load, which is used in admission experiments and often written as a ratio to the admission threshold, represents the amount of traffic that "would be" on the bottleneck if admission control was not activated and any arrived flow were admitted (assuming the bottleneck has infinite capacity). Clearly, the bottleneck flow-arrival-rate can be computed from demand-overload for a given flow type, flow duration and the flow arrival model. In almost all our experiments we used Poisson flow arrivals, with the flows duration exponentially distributed with a mean of 1 minute. A subset of experiments uses what we call a BATCH flow arrival model, where batches of flows arrive according to a Poisson distribution. The number of flows in a batch was chosen from a geometrical distribution with the mean of 5.

A similar parameter called *event-load* is used for termination experiments. It represents the bottleneck load *after* the failure (and the corresponding reroute) event occurs, but *before* the termination occurs, and is often written as a ratio to termination threshold. Note that the event-load actually consists of the load originally on the bottleneck and the rerouted traffic. We generated rerouted traffic by simulating the actual reroute of the packets of already active flows, and so the arrivals of rerouted traffic on a link appear as a "large surge" which we believe is more realistic than modeling them as Poisson arrivals.

4.2 Basic Experiments on Single Link

In this section, we take the first step of our PCN evaluation. One of the assumptions in PCN architecture (as just about in any measurement-based algorithm) is that there is a "sufficient" level of aggregation at the bottleneck link. Hence our foremost task is to quantify exactly how much is "sufficient". Our next task is to understand the effect of each of the configurable parameters in the algorithms, in particular, whether the performance of the algorithms is sensitive to the settings of these parameters. In essence, we will perform a sanity check to see, given sufficient bottleneck aggregation and under the most simple topology (a single link), whether the PCN algorithms can perform as expected. The results we obtain here (e.g. bottleneck aggregation, key parameter settings), provide a foundation for all experimental setups in the later sections.

4.2.1 Bottleneck Aggregation

In our experiments, the bottleneck aggregation needed to ensure that the over-admission percentage remained within 10%[3] of the admission-threshold is in the range of ~75 (for CBR traffic) to ~125 flows (for SVD traffic), with the rest of the traffic types falling

[3] In fact, for all traffic types other than SVD, the over-admission percentage is well within 5% for the above-mentioned bottleneck aggregation level

in-between. These numbers translate roughly to a 5 Mbps link for CBR voice and a 0.5 Gbps link for SVD. We showed by direct evaluation that the actual setting of the admission threshold and the actual link speed has no bearing on the observed performance as long as the aggregation level is sufficiently high. Our voice experiments therefore used an OC3 link (155Mbps), while video experiments run on link of 1Gbps or higher, and we maintained the admission threshold at 50% of the link bandwidth for all experiments reported here, without losing any generality.

4.2.2 Parameter Sensitivity of Admission

Table 4.2: Description of key admission control parameters

vq-upper-limit	CL	Marking	The maximum size a virtual-queue can grow
vq-min-threshold **vq-max-threshold**	CL	Marking	No packets are marked until virtual-queue size has reached vq-min-threshold, all packets are marked when it exceeds the vq-max-threshold, and between the two thresholds the marking probability linearly increases from 0 to 1
vq-threshold	CL	Marking	Combined vq-min-threshold and vq-max-threshold into one. Marks with probability 1 above it and 0 below it
tb-depth	SM	Marking	The depth of the token-bucket
ewma-weight	Both	Edge	Weight put on the history when computing ewma-cle
cle-threshold	Both	Edge	Stop admission if ewma-cle is smaller than the cle-threshold, allow admission otherwise

Now we focus on the parameter sensitivity of the PCN admission control. Before we present the results of our study, let us first review the PCN admission algorithms we study here and highlight some of the key difference between CL and SM. Recall, the admission control functions as follows: the packets are admission metered (and perhaps marked) at the interior PCN node. The edge node collects the measurements called *interval-cle* on an interval basis (interval-cle is the percentage of marked traffic over total traffic). The interval-cle is smoothed as an exponential weighted moving average, the *ewma-cle* is then compared to a *cle-threshold* to make the admission decision. All the key parameters involved in admission control for both algorithms are listed in Table 4.2. Note, the parameters in row 2 and 3 correspond to the two virtual-queue marking schemes which we termed "ramp" and "step". In our experiments, the use of only one threshold in the step-marking does not result in visible difference in the admission performance compared to the ramp version. Hence for the rest of this section, we focus our results on step-marking only. Since the edge functions are the same for both CL and SM, any difference in their performance must be rooted in the differences between their marking algorithms. SM with token-bucket marks the ***excess*** traffic over the admission threshold. For SM, the interval-cle is an indication of where the actual load is with respect to the admission threshold. In fact, a cle-threshold of x implies that the expected behavior of SM is to keep the mean load at x over the admission-threshold. In contrast CL with a virtual-queue has a different marking mechanism; for CL when the load is higher than the admission threshold, and the virtual-queue size is building up to exceed the vq-threshold, ***all*** traffic is marked. Therefore during the congestion, the semantics of interval-cle is different for the two algorithms, and the absolute value of interval-cle for CL tends to be much larger than for SM.

We now can turn our attention to the design of the experiments. First, we discovered that the behavior of the algorithms on single link topology can be predicted with our fluid model with remarkable accuracy. This encouraged us to utilize the efficiency of the fluid model to conduct large scale experiments that cover a wide range of parameter settings. As for the performance metric, recall that in Section 4.1, we presented a snapshot of the bottleneck load vs. time in the network running PCN algorithms in Figure 4.1. In particular, the snapshot shows actual load on the bottleneck oscillates around a (stable) average. As we mentioned then, our primary performance metric is the over-admission, which is the percent deviation of the average load achieved in the experiment from the admission threshold. In this section we will also quantify the magnitude of the oscillation around the average (expressed as deviation from the admission threshold as well). We organize our investigation as follow: for each of the parameters listed in Table 4.2, we study its effect by varying it through a range of values while keeping all other parameters' settings fixed. In addition, each experiment is run with three demand-load levels: 1.25x, 2x and 5x. We believe that the low-end of these load levels (i.e. 1.25x) represents the normal congestion, and the high-end level (i.e. 5x) bounds the extreme overload that is unlikely to happen in reality.

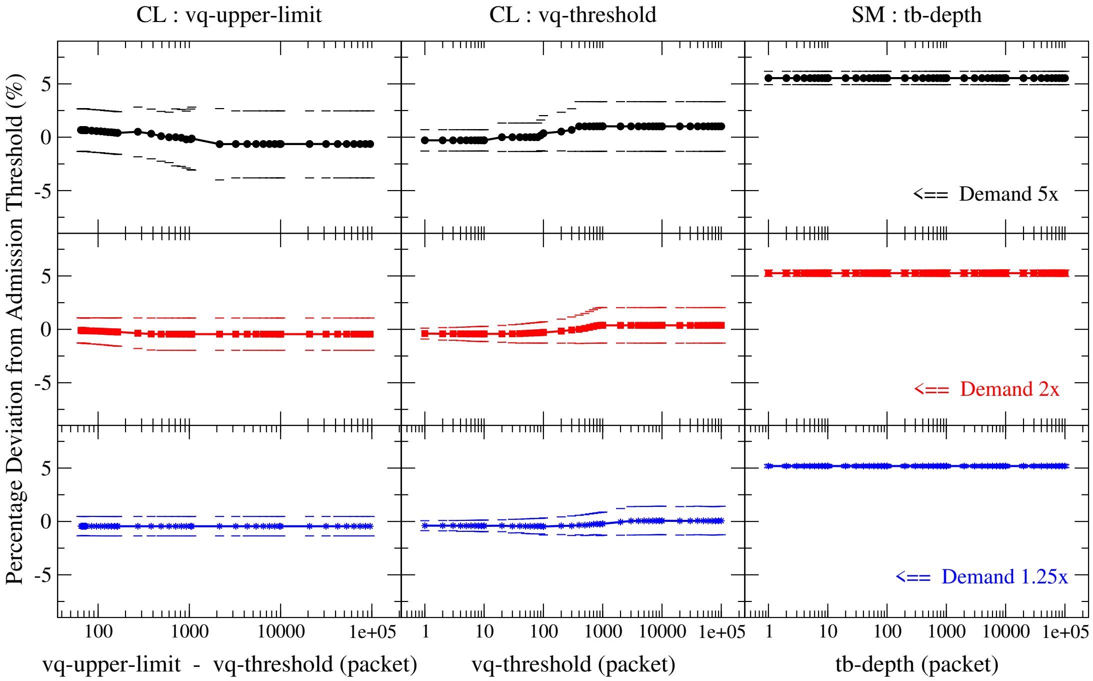

Figure 4.3: Sensitivity of the marking parameters in admission

Figure 4.3 summarizes our fluid simulation results on vq-upper-limit, vq-threshold and tb-depth (organized from left to right). We plot them in the same graph next to each other because these are all parameters in the marking function (in contrast to the edge parameters presented later). Each row in the graph represents a different demand-load level, with 5x demand on the top, 1.25x on the bottom. In each individual graph, the x axis gives the range of the value we varied for the parameter of interest. The results consist of the average load (represented with filled symbols), and the expected oscillation (represented by error bars above and below the average load). All results are expressed in terms of percentage deviation from the admission threshold (hence the value of the filled symbol is in fact the over-admission percentage). All these sets of experiments have ewma-weight set to 0.5, and cle-threshold to 0.05. For vq-upper-limit experiments, we fix the vq-threshold to be 256 packets, and then increase the difference between vq-threshold and vq-upper-limit (starting from 64 packets). For the vq-threshold experiments, we fix the difference between vq-threshold and vq-upper-limit to be 64 packets, and then increase the value vq-threshold. For tb-depth, we simply increase the value of tb-depth.

The results show that the two algorithms function as predicted in this simple topology. CL has almost no over-admission, while SM maintains the load at 5% of the admission threshold (since we set the cle-threshold to 0.05). Note that CL has a larger oscillation range. Intuitively, the oscillation range is a measure of how fast the system switches between the admission state and the congestion state (hence block admission). Recall that in the previous paragraph we mentioned that the difference in marking algorithms leads to much larger interval-cles in CL than SM during congestion (e.g. it could be 1.0 for CL, 0.05 for SM). Hence, CL takes longer to age the ewma-cle below the cle-threshold and to resume admission again. Furthermore, increasing the vq-upper-limit and vq-threshold both have the effect of prolonging the transition between the states. The larger the vq-threshold, the longer it takes to “fill up” the queue to signal congestion, and once filled up, the larger the vq-upper-limit, the longer it takes to “drain” the queue to clear the congestion signal. For SM, once the token-bucket is emptied and excess traffic marking starts (which initially could take a while with a large tb-depth), the system switches between congestion and admission state rather quickly and the ewma-cle value tightly oscillates around cle-threshold. Other than at the very beginning of the simulation, the token-bucket is never completely emptied or filled. Hence we see the resulting load oscillating around its average with larger deviation for CL than SM. In addition, CL shows slight sensitivity to its parameters – vq-upper-limit and vq-threshold, while SM shows none. The overall takeaway from Figure 4.3 is that both algorithms' performance is insensitive to their marking parameters' settings, especially under light demand-load. The average load and the oscillation range are bounded within 5% of the admission threshold.

Next we turn our attention to the two edge parameters, ewma-weight and cle-threshold. For the ewma-weight experiment in Figure 4.4, the settings for the rest of the parameters are as follows: cle-threshold = 0.05; vq-upper-limit = 320 packets; vq-threshold = 256 packets; tb-depth = 256 packets. The same settings are used for cle-threshold experiments in Figure 4.5 with ewma-weight set to 0.5.[4] Again, the results are as expected. CL shows more sensitivity to ewma-weight; due to its large interval-cle values, a heavy weight can significantly prolong the aging. The same effect is also observed on SM, though with a much smaller impact. As for the cle-threshold, CL is completely insensitive to its change. While for SM, the cle-threshold is a significant parameter, as we explained before, the over-admission is not caused by poor performance of the algorithm, but rather by design, as there is the positive correlation between cle-threshold and over-admission of SM. The effect is accurately reflected in Figure 4.5.

So far we investigated the effects of all the key admission parameters and showed

[4] Note, the settings for the marking parameters are chosen within the range of values we tested in the previously described experiments.

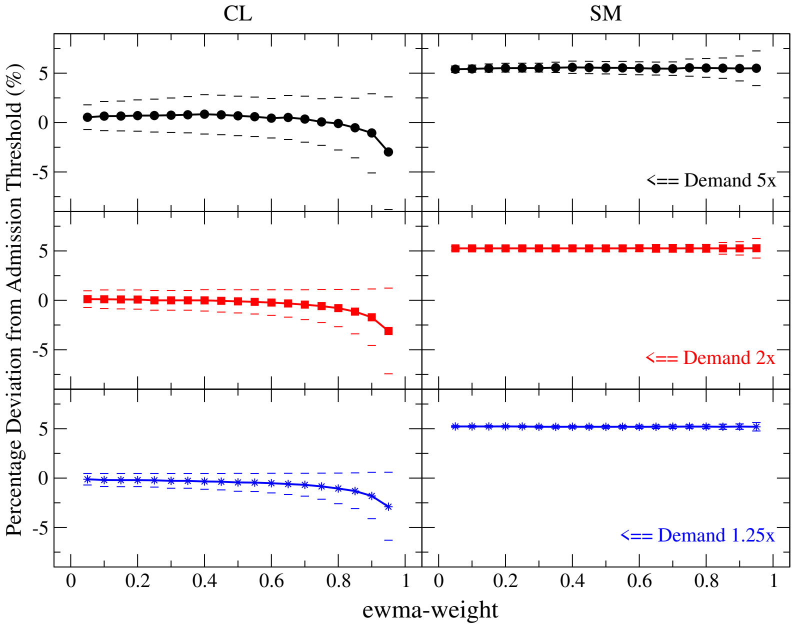

Figure 4.4: Sensitivity of ewma-weight in admission

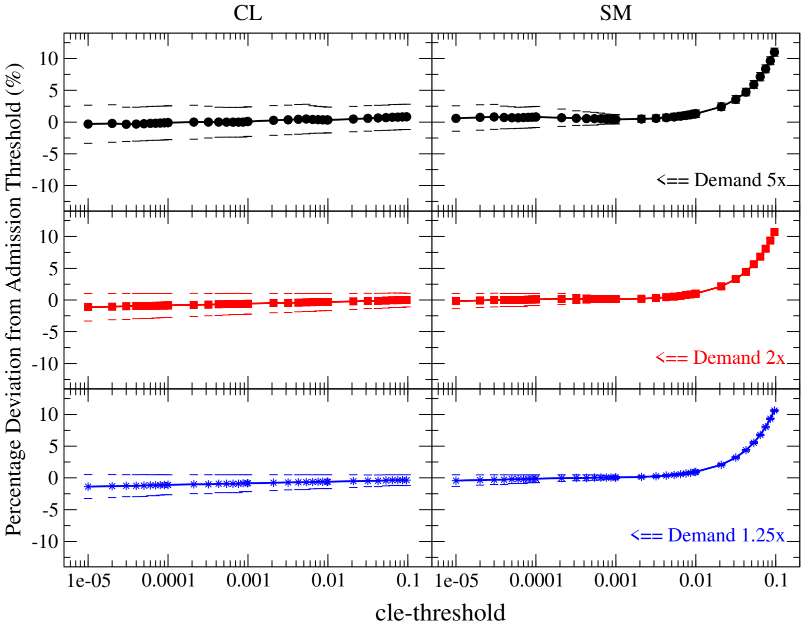

Figure 4.5: Sensitivity of cle-threshold in admission

through fluid simulation that the performance of both algorithms remains stable across a wide range of parameter settings. One point worth noting is that, while we claim that our fluid model predicts the system outcome with great accuracy, the results of the fluid model only resemble a network running with CBR traffic. Therefore, we complete our investigation by running a set of experiments with a packet simulator on all traffic types. The settings for the experiments are listed in Table 4.3.[5] We run the experiments with all traffic types and three demand-load levels for each combination of the settings.

Table 4.3: Parameter settings for admission experiments across all traffic types

vq-upper-limit	vq-threshold + 64 packets
vq-threshold	256, 2048 packets
tb-depth	256, 2048 packets
ewma-weight	0.1, 0.5, 0.9
cle-threshold	CL: 0.05, 0.15, 0.25 SM: 0.001, 0.01, 0.05

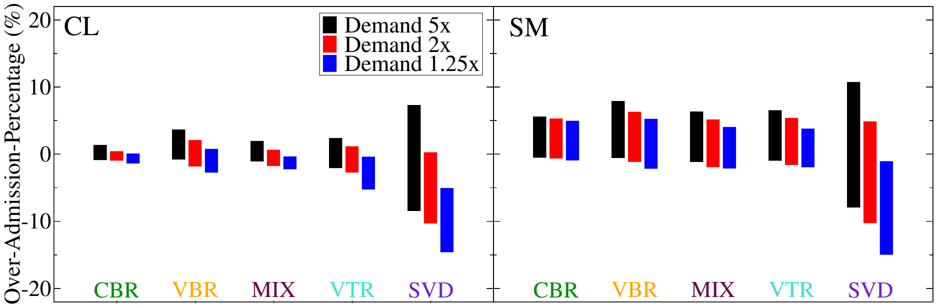

Figure 4.6: Parameter sensitivity in admission across all traffic types

The results are plotted in Figure 4.6. Each bar in the graph represents, for a given traffic type and demand-load, the range of the over-admission-percentage across all parameter setting combinations. The results show that our previous conclusion from the fluid model, that both algorithms are insensitive to these parameters, holds true for all traffic types except SVD. Over-admission is bounded within ±5% of the admission threshold for all traffic except SVD. We believe the large sensitivity for SVD should be traced to the burstiness of our crude traffic model. Recall, this traffic is designed to stretch the limit of the measurement base algorithm. We expect better performance on a real-world video model, as is in the case for our video trace VTR.

[5] Again, these settings are chosen within the tested range of the previously described experiments

4.2.3 Parameter Sensitivity of Termination

Compared to admission, there are fewer parameters involved in the termination control. Both algorithms use the same marking algorithms – token-bucket, and hence share a parameter – tb-depth. The difference between the algorithms lies in the relationship of the token-bucket rate to termination-threshold. While CL meters against the termination-threshold, SM meters against the admission-threshold, hence it has an additional parameter that is the ratio between the two thresholds, K. Intuitively, the change of either parameter should not effect the quantitative outcome of the termination, which is measured by the over-termination metric. However, a large tb-depth means it takes longer to empty the token-bucket and start marking, which in turn implies a longer termination reaction time. We set out to verify our intuition with a set of packet simulations similar to the ones in the last section for admission. The range of settings for the two parameters are listed in Table 4.4. Again we run the experiments with all traffic types and three event-load levels (1.25x, 2x, 5x) for each combination of the settings. The results are plotted in Figure 4.7. Like in the admission graph, each bar in the graph represents, for a given traffic type and event-load, the range of the over-termination-percentage across all parameter setting combinations. As the graph shows, we observe very little variation in the over-termination when changing the two parameters across a wide range.[6] Finally, our results indicate a correlation between the tb-depth and the termination reaction time, though, for all tests we conducted, the reaction time is within the acceptable range (which is 2 seconds as discussed in Section 4.1.1).

Summary In this section, we established the level of bottleneck aggregation required for the PCN algorithms, which is in the range of ~75 to ~150 flows, depending on the traffic types. In addition, we found that neither CL nor SM are sensitive to its key

[6] Note, SVD again shows a larger spread which again is linked to the burstiness of the model. We do not observe any consistent correlation between value of the parameter setting and the over-termination

parameter settings. We can then move beyond a single link topology and investigate the performance of PCN under more general settings.

Table 4.4: Parameter settings for termination experiments across all traffic types

tb-depth	Both	64, 256, 1024, 4096, 16384, 32768 packets
K	SM	1.5, 2.0, 2.5, 3.0

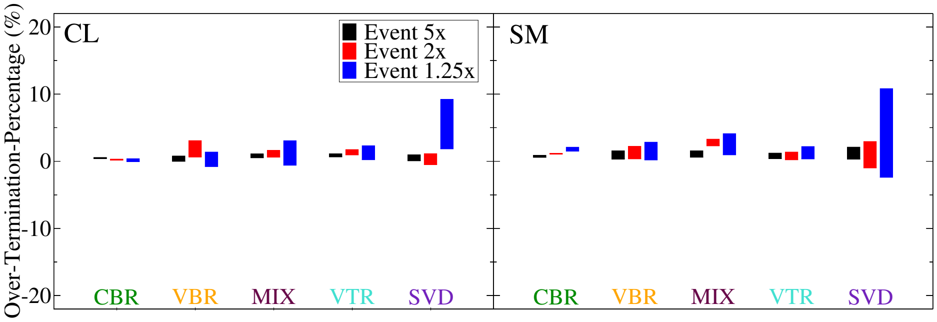

Figure 4.7: Parameter sensitivity in termination across all traffic types

4.3 Ingress-Egress Aggregation Effect on Admission

While the assumption of reasonable aggregation of PCN traffic at an internal bottleneck seems a relatively safe one (at least in the aggregated networks such as ISP cores), it is much less clear whether it is safe to assume that high ingress-egress aggregation level is a safe assumption in reality. In particular, with a SVD setup, only ~100 SVD flows can take up about 50% of a 1Gbps bottleneck link bandwidth and a scenario where all 100 flows come from different ingresses seems entirely plausible. It is therefore important to investigate if the algorithms can perform sufficiently robustly under these circumstances. For the rest of this section, we shorten the ingress-egress aggregation as IE-aggregation, or simple aggregation.

We run a range of experiments on MultiLink (MLK-N) topology (see Figure 4.2(b)). Recall this is the class of topology precisely designed for IE-aggregation experiments.

We fix the bottleneck capacity and vary the number of ingresses to obtain the desired level of IE-aggregation for that bottleneck load. The bottleneck aggregation is assumed to be sufficient with values suggested in Section 4.2.1. We also fix a particular ewma-weight, cle-threshold and demand-load level (ewma-weight=0.3, for CL, cle-threshold = 0.05, and 0.001 for SM, demand-load = 5x) for all experiments.

We plot the comparative results in Figure 4.8. The graph shows the change of over-admission with respect to the level of IE-aggregation. In each line, the leftmost datapoint represents the case with the lowest level of aggregation (expected flow-per-aggregation is 1 or 2), while the rightmost datapoint represents the largest IE-aggregation. The CL results show no sensitivity. On the other hand, SM performs significantly worse at lower levels of aggregation. For example for CBR, with expected 1 flow-per-aggregation, the over-admission-percentage can be as bad as 45%.

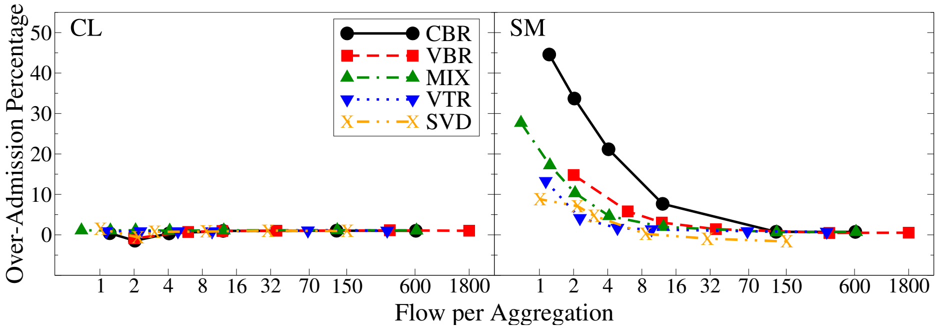

Figure 4.8: Ingress-egress aggregation effect on admission

Our investigation reveals the cause of this poor performance is that, contrary to our expectation that the admission marking will be more or less uniformly distributed among active flows, what actually happens is that some flows get all their packets marked, while other flows get no packets marked at all. It turns out that if the number of tokens arriving in the token-bucket in an packet-inter-interval-time of a single CBR flow is an integer multiple of a packet size, then if a packet of a flow is marked once, all the subsequent packets will find the same number of tokens in the token-bucket and will also be marked.

We refer to this effect as *Marking Synchronization*, *synchronization* for short in the rest of this section (the proof of this is in Appendix A). It is this phenomenon that, in the case of a single flow per aggregate, made only a few aggregates whose flows are marked react to the admission control, while the rest which receive no marked packets end up over-admitted. Since most of the other traffic patterns contain large CBR segments, this effect is seen with other traffic types as well, although to a different extent.

A natural initial reaction can be to write off this effect as purely a simulation artifact. In fact, one can expect that some randomization is introduced into the strict CBR traffic pattern when the initially strict CBR traffic traverses multiple links. Such randomization can be reasonably expected to break the CBR pattern. We set out to verify just how much randomization is actually necessary to break the synchronization. To do so we implement limited randomization of the base models for all of our traffic stream, by randomly moving the packet by a small amount of time around its transmission time in the corresponding base traffic model. We choose the ***randomization-interval*** to be a fraction from 0.0001 to 0.1 of packet-inter-arrival-time of the CBR portion of the corresponding base model. While we do not claim this is an accurate model for network-introduced jitter, we chose it for the simplicity of implementation as a means to gain insight on any simulation artifacts of strictly CBR traffic generation. We then re-ran all the experiments with the same topologies and parameter settings. The results are summarized in Figure 4.9. The lines labeled with f correspond to randomized traffic with randomization-interval $= f \times$ inter-packet-arrival. It also means on average, the packets are delayed by $f \times$ inter-packet-arrival $/ 2$. The lines labeled as "No-Rand" correspond to the SM results in Figure 4.8.

All graphs in Figure 4.9 except for VTR exhibit a clear pattern, namely, as the randomization-interval increases, the effect of aggregation (high over-admission) diminishes. It means that indeed introducing enough randomization does break the marking synchronization and the performance of the algorithm much improves. However, it requires a sufficient amount of randomization. For instance, in the CBR graph, the only line that shows no aggregation effect is the line labeled with “0.05”, which translates to expected packet jitter of 0.5ms. While 0.5ms per-hop jitter is not unreasonable to expect for voice traffic, in well provisioned networks with a relatively small amount of voice traffic in the priority queue one might in fact find lower delay levels. In any case, these results indicate that the synchronization effect cannot be completely written off as a simulation artifact. The good news, however, is that this effect is visible only at very low IE-aggregation levels. For example, in the CBR graph, the portion of the lines with more than 15 flows all collapse together, indicating that as the aggregation increases, the effect quickly disappears.

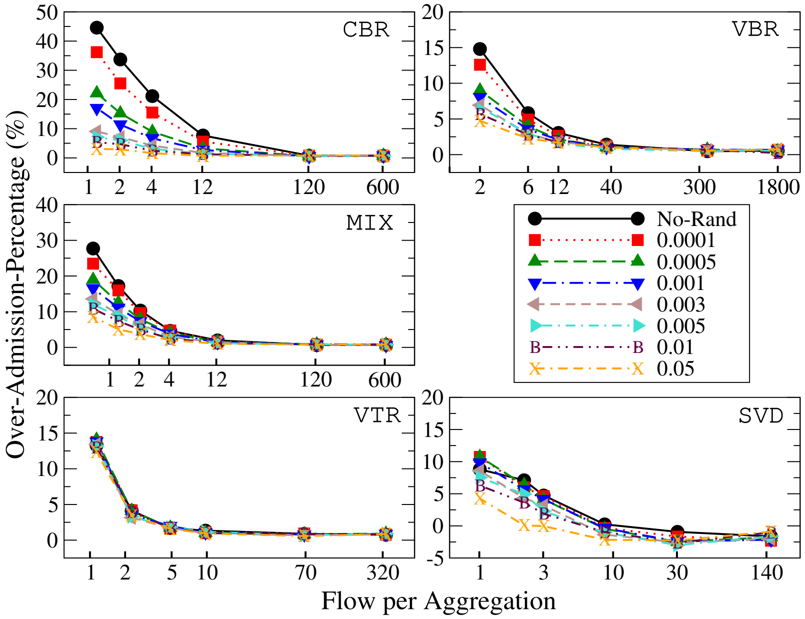

Figure 4.9: Synchronization effect vs. randomized traffic in SM admission

VTR is an exception among the graphs in Figure 4.9. It also exhibits some aggregation effect, however randomization of its CBR portion has almost no effect on performance. We believe this is because the randomization we perform is at the packet level, while the synchronization that seems to be causing the performance degradation for VTR traffic occurs at the frame-level. Our results show that if we calculate random deviation for our artificially induced jitter using frame-inter-arrival time instead of packet-inter-arrival time, we can reduce the over-admission for VTR to roughly 3%.[7]

Summary We studied the effect of ingress-egress aggregation on admission. While CL is completely insensitive to IE-aggregation level, SM shows substantially more over-admission at very low IE-aggregation. We found the cause to be the non-uniform distribution of markings, which we termed *synchronization effect*. Although the synchronization effect can be lifted by introducing jitter, the level of jitter required indicates it might not simply be a simulation issue. Nonetheless, the effect of low aggregation is only observable with fewer than 15 flows per IE-aggregate.

4.4 Ingress-Egress Aggregation Effect on Termination

Just as in the case for admission, the effect of ingress-egress aggregation on termination needs to be investigated. Though we do not anticipate the performance to be affected by the IE-aggregation level in general, there is a theoretical concern that the granularity of termination (which can operate on integer number of flows only) will result in large inaccuracies of the amount of terminated traffic when ingress-egress aggregation is very low. For instance, if $\frac{1}{3}$ of the bottleneck rate needs to be terminated to bring the bottleneck load down to the termination threshold, and if the bottleneck load consists of many ingress-egress aggregates with exactly 1 flow-per-aggregate, then ideally every flow will have $\frac{1}{3}$ of its packets marked, which will cause every ingress to terminate its only flow. It is therefore important to understand how bad this effect can get in practice,

[7] The details of the results are omitted for conciseness. It is unclear however, whether such randomization at the frame level meaningfully reflects network-introduced jitter.

for both CL and SM. Note, unlike the admission, the termination part of CL and SM are essentially based on the same marking algorithm, and we have demonstrated their comparable performance on the single link topology, hence it's natural to expect a similar behavior in this case.

We start our investigation again with a set of experiments using MLK-N topologies, as in the case for admission in Section 4.3. Recall, for MLK-N topology, we fix the load on the bottleneck (with sufficient aggregation), and then vary the number of ingress-egress-pairs that share the bottleneck to achieve the desired IE-aggregation level. In Figure 4.10 below we plot over-termination-percentage against the flow-per-aggregation for each traffic types. Again the left-most data-point on each line represents the lowest aggregation level.

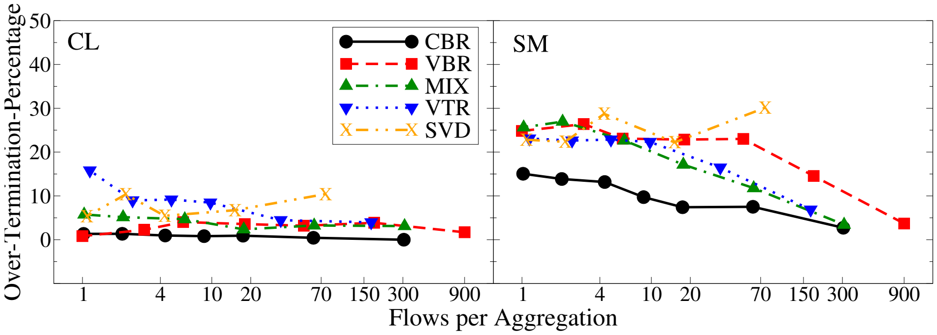

Figure 4.10: Ingress-egress aggregation effect on termination

There are two things in this graph that are peculiar. First, in the CL graph, none of the traffic types except VTR exhibits any sign of our theoretical worst case predication (e.g. the line for CBR is perfectly horizontal). Second, SM on the other hand does show signs of IE-aggregation effect, but gives an over-all worse performance compared to CL (over-termination is in the range of 20%). It's counter-intuitive for this very different behavior given that the two share the same marking algorithm. For the remainder of this section, we target our investigation to answer these two questions.

4.4.1 IE-Aggregation Effect with CL

It turns out that the cause of the unexpectedly good performance of CL termination is exactly the same as the cause of unexpectedly bad SM admission results - namely marking synchronization (see Section 4.3), when some flows are consistently marked, and some flows are not marked at all. In the case of termination, this effect is actually helpful. Take the worst case scenario we described at the beginning of the section, now instead of every aggregate getting $\frac{1}{3}$ of its traffic marked, the synchronization effect concentrates the markings on a subset of them, and hence spares the rest from terminating their only flow. To verify this argument, we re-ran the same set of termination experiment on the randomized version[8] of all traffic types, with random-deviation set to 10%. The results are plotted left graph in Figure 4.11 (the right one is the original CL with non-randomized traffic).

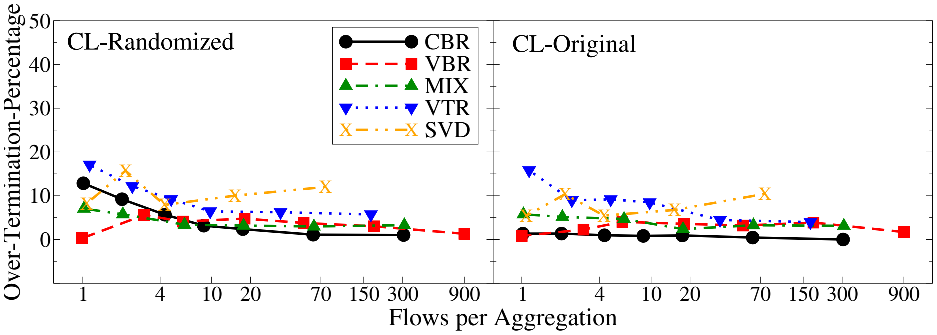

Figure 4.11: Synchronization effect vs. randomized traffic in CL termination

For CBR, as we expected, the randomization indeed breaks the synchronization, resulting in substantially more over-termination (14%) at low aggregation. This shows just as in the case of admission, one could not write off the good performance at low aggregation level due to synchronization merely as a simulation artifact. In any case, as the aggregation level increases, the worst-case behavior due to granularity of termination is no longer an issue anyway (the over-termination is down to %5 with only 4 flow-per-

[8] See Section 4.3 for the description of the randomized traffic

ie-aggregate). Note, we point out that even with the lowest aggregation (1 flow), our theoretical worst case analysis did not occur. This is because our aggregation level is expressed in expected values and the actual number of flows for any given aggregate at any time is a random variable. So at the instance of the event, some aggregates actually have 1, 2 or 3 flows, while others might have none at all and this prevents the worst case scenario from happening.

VBR shows some minor influence by the aggregation level, but at the left-most datapoint it did not give the worst over-termination as expected. We believe the reason for this outcome is the following: at the low aggregation level (with 1 or 2 flows per ingress), each VBR flow spends a large portion of its time in off-period. Once the aggregate is in its off period, it sends no packets, gets no markings, and hence will not react to the termination algorithm. Since a substantial portion of the aggregates can be in the off-period, only the ones that are in the on-period terminate their traffic.

The MIX essentially shows the combined effect of both CBR and VBR, in the sense that it shows a clear increase in the over-termination-percentage as the level of aggregation decreases, yet at the lowest aggregation, instead of having the highest over-termination (like CBR), the on-off effect of VBR dominates, hence we again do not see a significant over-termination.

The trace analysis of the SVD experiments indicates that there are a large number of partially marked flows, which indicates that synchronization could not have been responsible for the relatively bounded over-termination of about 10% in Figure 4.11. We believe this performance should be traced to the burstiness of our crude SVD traffic model at the time scales commensurate with the measurement period. In addition, just as for VBR, the on-off nature of the model dominates at low aggregation, which can also be used to explain why no aggregation effect is observed.

In summary, the over-termination can be expected at low aggregation for a variety of traffic, but in practice the degree of this over-termination is not as bad as the worst-case analysis might indicate. The over-termination vanishes as the level of ingress-egress aggregation becomes sufficiently large.

4.4.2 IE-Aggregation Effect with SM

We now investigate the performance differences between SM and CL. As shown in Figure 4.10, SM seems to produce worse over-termination at all aggregation levels for various traffic types. This problem is unlikely to be caused by marking synchronization; as we have shown in the case for CL, de-synchronizing the markings only has effects at very low aggregations. But we re-run the experiments with randomized traffic (the same configuration as in CL), and plot the results in Figure 4.12. The results confirm our intuition, by randomizing the traffic, the only visible difference is making CBR perform even worst at lower aggregation (the same thing happens to CL).

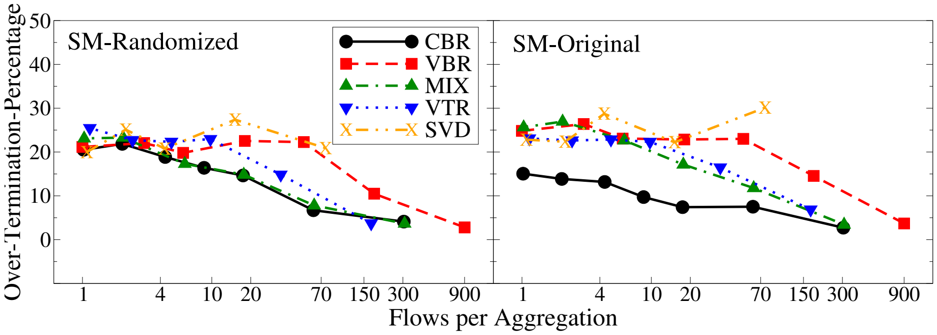

Figure 4.12: Synchronization effect vs. randomized traffic in SM termination

A careful look into output traces shows a phenomenon common to all SM experiments, that is, even after the bottleneck load is already below the termination threshold, some aggregates still initiate new rounds of termination, which in turn causes the over-termination we observed. It turns out that part of the reason for these extra (unnecessary) terminations lies in the non-uniformity in the marking distribution across different aggregates. We will illustrate this point through a simple example. Imagine two aggregates sharing a bottleneck. Both have a rate of 10Mbps and the termination threshold is at 20Mbps (exactly enough for both aggregates). Now suppose admission threshold is at 10Mbps and K is therefore 2 (recall K is the radio of termination- and admission-threshold). Recall under SM, marking is done against the admission threshold. Ideally, the bottleneck marks 10Mbps of traffic with each aggregate getting 5Mbps. The aggregates then compute its sustainable rate which is 5Mpbs $* K =$ 10Mbps. Since the sustainable rate is equal to the sending rate, the aggregates will not terminate any flow. Now imagine what if the markings are not uniformly distributed among two aggregates, instead of each getting 5Mbps, the split is 4 and 6Mbps. Then the aggregate that gets 6Mbps traffic marked will conclude it has a sending rate (10Mbps) larger than the sustainable rate ($(10 - 6) *$ 2Mpbs = 8Mbps), and hence start unnecessary termination.

It seems that we have pinpointed the problem to be the non-uniform marking distribution. However, this is a property of the token-bucket marking algorithm, which implements the termination control of both SM **and** CL. Why then, do we only see the performance penalty in SM? Let's go back to our simple example to see what is CL's behavior under the same setup. The answer is trivial, CL will not mark any traffic! No marking, hence no termination.[9] This reveals an important property of SM termination in contrast to CL. With CL, the bottleneck conditions with respect to the termination threshold, are conveyed directly through the termination markings. On the contrary, SM lacks such a signal, and needs to rely on the markings against the admission threshold to infer the conditions on the bottleneck. This inference process is vulnerable to various sources of inaccuracies, in this case, the non-uniformity in the marking distribution, and

[9] If the bottleneck load is above the termination threshold and CL marks the excess traffic, it is true that these markings might not be evenly distributed among the aggregates. However, this will only cause a slight unfairness among the aggregates, but ***not*** result in bottleneck over-termination

the net result is over-termination at the bottleneck. (We will see the occurrence of this property again in the multiple bottleneck environment later in this chapter.)

Having established the root cause of the problem, we can now consider the possible fixes. Since we cannot change the property inherent in the SM architecture, the only fix is in the direction of reducing the adverse effect coming from the non-uniformity of the marking distribution. Smoothing with exponential weight average (EWMA) is an obvious step to take. There are various interval egress measurement that can be smoothed, such as unmarked-rate, marked-rate, sending-rate etc. We have experimented with various combinations which give no significant performance difference. The one we presented here is based on the smoothing of interval-cle value. Recall, interval-cle is the ratio of marked-rate and the total-rate (it's a value also used by the admission algorithm). The intuition is to use the condition $\frac{\text{marked-rate}}{\text{total-rate}} > \frac{\text{termination-threshold} - \text{admission-threshold}}{\text{termination-threshold}}$ as an indication to trigger termination. More specifically, each aggregate computes a EWMA smoothed interval-cle, and then compares it to the value $\frac{K-1}{K}$. If the former is smaller, there will be no termination, even if the computed sustainable-rate is smaller than the sending-rate. Otherwise, the aggregate compares the sustainable-rate to the sending-rate to see if (and how much) to terminate. In a nutshell, we implement an additional control, based on smoothing, to counter the inaccuracy in the interval measurements and to safeguard the triggering of termination.

We implement the above algorithm and re-run the same experiments. One additional parameter is the ewma-weight(w) used in the smoothing of interval-cle[10]. A larger w corresponds to a longer history. We find that for this additional control to be effective, a very large w is needed. Below in Figure 4.13, we plot the outcomes of the smoothing

[10] Note this should not be confused with the ewma-weight in admission control. Interval-cle is also smoothed for admission, which should be done independently from what we introduced in this section

with $w = 0.9$. The improvement over the original SM is significant, the smoothed version of SM has a performance comparable to that of CL.

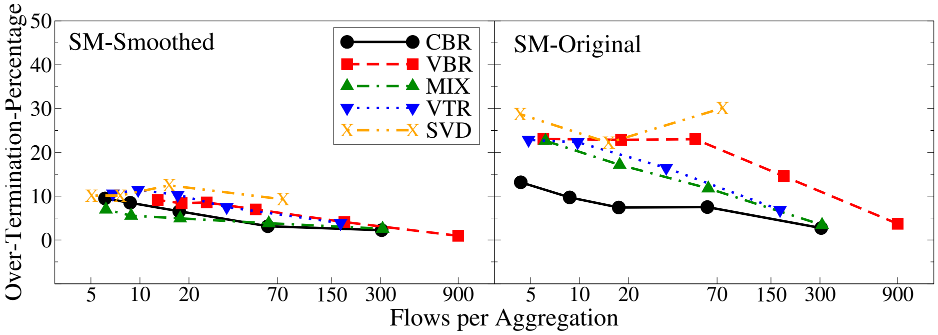

Figure 4.13: Effect of smoothing in SM termination

However, the improvement in the performance comes at a cost. It increases the reaction time of SM termination. The reason is as follows: before an event occurs that puts the bottleneck load over the termination threshold, the network, assuming admission control is functioning correctly, should have a load around the admission threshold, which means a interval-cle of 0 (or something very small, recall from Section 4.2.2). After the event has occured, the aggregate needs to wait for interval-cle to age from 0 to a value larger than $\frac{K-1}{K}$ before it can initiate the very first round of termination. Given that we need a large weight on the history for effective smoothing, this waiting time could be long, which leads to a long termination reaction time (recall in Section 4.1.1 we discussed that the reaction time should be within 2 seconds). It is therefore important to quantify this extra delay.

In fact, we can mathematically derive the expected delay. Its value depends on K, the smoothing weight (w) and the excess load over the termination threshold (R) which is defined as $\frac{\text{event-load} - \text{termination-threshold}}{\text{termination-threshold}}$. In the derivation below, C_n is the smoothed interval-cle in the nth measurement interval. C_{int} is the interval-cle value. Note that after the event occurs, and before the termination starts, C_{int} remains the same for each interval with a value equals to $\frac{\text{event-load} - \text{admission-threshold}}{\text{event-load}} =$

$\frac{K * A * (1 + R) - A}{K * A * (1 + R)} = \frac{K * (1 + R) - 1}{K * (1 + R)}$. To compute the C_n, we have:

$$\begin{aligned}
C_1 &= 0 * w + C_{int} * (1 - w) \\
C_2 &= C_1 * w + C_{int} * (1 - w) \\
&\vdots \\
C_n &= C_{n-1} * w + C_{int} * (1 - w) \\
C_n &= C_{int} * (1 - w) * \sum_{i=0}^{n-1} w^i \\
&= C_{int} * (1 - w) * (1 - w^n)/(1 - w) \\
&= C_{int} * (1 - w^n)
\end{aligned}$$

We can then compute the condition to trigger the termination by plugging in the value of C_{int}:

$$\begin{aligned}
\frac{K * (1 + R) - 1}{K * (1 + R)} * (1 - w^n) &> \frac{K - 1}{K} \\
1 - \frac{(K - 1) * (1 + R)}{K * (1 + R) - 1} &< w^n \\
\boxed{\frac{R}{K * R + K - 1} < w^n}
\end{aligned} \tag{4.1}$$

Equation (4.1) shows the number of intervals (n) needed to trigger termination for given value of K, R and w. The relationship is rather intuitive: the larger the R, the larger the C_{int}, hence the faster the aging. The smaller the K, the smaller the gap between the admission and termination thresholds and the faster the aging. In Table 4.5 below, we list the expected delay in seconds for some combination of K and R given $w = 0.9$ (we believe these values resemble the practical setup). These results assume each measurement interval is 100ms and the round-trip time is negligible.

We conclude the investigation by noting that this smoothing is only necessary at the lower range of the IE-aggregation levels we considered, and is not necessary as soon as the aggregation level reaches 50-150 flows (for different traffic types) in our experiments. For the lower aggregation levels, the smoothing may be useful, at the expense of the additional delay.

Table 4.5: Additional delay caused by smoothing

K / R	**0.15**	**0.2**	**0.25**	**0.3**	**0.35**	**0.4**	**0.45**	**0.5**
1.2	0.9	0.8	0.7	0.6	0.6	0.6	0.5	0.5
1.4	1.4	1.2	1.1	1.0	0.9	0.9	0.8	0.8
1.6	1.7	1.5	1.4	1.3	1.2	1.1	1.1	1.0
1.8	1.9	1.7	1.6	1.5	1.4	1.3	1.3	1.2
2.0	2.1	1.9	1.8	1.6	1.6	1.5	1.4	1.4

Summary We studied the effect of IE-aggregation level on the termination for CL and SM. For CL, we observed some over-termination at very low aggregation, though the scenario predicated by the theoretical worst case did not occur, partially due to the marking synchronization we introduced in Section 4.3. Overall, CL's performance is not very much affected by the IE-aggregation level, as the over-termination is bounded by %10 across all traffic types and aggregation levels. For SM, a consequence of using just a single metering and marking, in conjunction with non-uniformity in the marking distribution among the contending aggregates, results in false terminations when traffic is closely below the termination threshold at the bottleneck. This effect is especially pronounced for low and medium levels of aggregation. We found the results could be improved to be comparable with CL with respect to over-termination if the aggregates used EWMA smoothed interval-cle when triggering the termination event. Such smoothing, however, would add latency to the termination decision. In the range of parameters we investigated that seem practically reasonable, the additional latency is bounded by 1-2 sec. Such smoothing is not necessary at larger levels of ingress-egress aggregation. In conclusion, to avoid over-termination, SM needs substantially more IE-aggregation than CL, if no additional mechanisms are used to smooth the termination trigger.

4.5 Multi-Bottleneck Effect on Termination

Unfairness to long-haul aggregates traversing many bottlenecks (which we referred as ***multi-bottleneck effect*** or ***beatdown effect***), is a relatively well known phenomenon [17], but the exact cause of it (and possibly the extent) depends on the details of the algorithm. Another practical concern that arises from the beatdown effect is whether it will cause the under utilization of the bottlenecks, or if the loss of long-haul aggregates can be in some way compensated by the gain of the short-haul ones.

Our goal in this section is three-fold. First, we will study the cause and the property of the multi-bottleneck effect on termination, and demonstrate it using the CL algorithm. We will then investigate if the extent of the effect differs for CL and SM. Finally, we will quantify how much of the effect can be expected in practice on more generalized settings with fluid models.

4.5.1 Understanding Multi-bottleneck Effect on Termination

The primary cause of the beatdown effect for termination is excessive marking of the long-haul aggregate at the subsequent bottlenecks. We first recall that termination-metering at the internal node applies only to previously un-marked traffic, which is intended to partially mitigate the excess marking at subsequent bottlenecks. Nevertheless, as we show below it is not sufficient to prevent the beatdown effect completely.

Consider the following scenario on PLT-2 (see Figure 4.2(b)). At the time of the failure event, the bottleneck load is 75% of the link bandwidth, where long-haul aggregate (A–C) constitutes 50%, and 2 short-haul aggregates (D–E) and (E–F) each takes 25%. Assuming for the sake of example the termination threshold is at 50% of the link bandwidth, then an trivial "good" solution is to have three aggregates each terminate $(\frac{3}{4}-\frac{1}{2})/\frac{3}{4} = \frac{1}{3}$ of their traffic. With CL, when the long-haul aggregate traverses through the bottleneck1 (A–B), assuming marking is uniformly distributed, $\frac{1}{3}$ of its traffic will get termination-marked. (The short aggregate D–E will also get $\frac{1}{3}$ of its traffic marked). Next, $\frac{2}{3}$ of long-haul's traffic (the unmarked ones) together with short-haul (E–F)'s traffic will create a load of $\frac{2}{3} * \frac{1}{2} + \frac{1}{4} = \frac{7}{12}$ on the bottleneck2 (B–C). This implies that for the long-haul aggregate, an additional $(\frac{7}{12} - \frac{1}{2})/(\frac{7}{12}) = \frac{1}{7}$ percentage of the remaining unmarked traffic will be marked. As for short-haul E–F only $\frac{1}{7}$ of its traffic will be marked. Now back to bottleneck1, the short-haul aggregate terminates correctly, while the long-haul one over-terminates, with the net effect of over-termination at bottleneck1. We use the packet simulator to simulate the above simple scenario on PLT-2 and PLT-5. Recall in PLT-5 (Figure 4.2(c)), there are 5 one-hop short-haul aggregates and 1 long-haul aggregate that travels 5 hops. The load distribution among the aggregates is the same as in PLT-2 described above. Table 4.6 below summarizes the ***actual*** termination percentage (not over-termination-percentage) of the long-haul aggregate and first bottleneck for various traffic types.

Table 4.6: Multi-bottleneck effect on CL termination

Long-haul Aggregate	CBR	VBR	MIX	VTR	SVD
PLT-2	42.60	43.34	41.66	44.19	50.14
PLT-5	47.71	49.56	49.96	49.06	53.24

Bottleneck 1	CBR	VBR	MIX	VTR	SVD
PLT-2	39.96	40.77	39.36	41.86	44.78
PLT-5	43.27	44.53	45.58	45.39	48.46

Recall that, in the above setup, a trivial "good" solution is to have each aggregate terminate 33% of its traffic (which translate to a 33% bottleneck termination as well). The results show the beatdown effect for the long-haul aggregate, and also the subsequent over-termination at the upstream bottleneck are present for all tested traffic types. In addition, the difference between PLT-2 and PLT-5 shows that effect is more pronounced as the long-haul aggregate travels through more bottlenecks.

4.5.2 Comparison Between CL and SM

To understand if the extent of multi-bottleneck effect differs for CL and SM, we start by running the same simple scenarios described in Section 4.5.1. We keep all parameters the same, with one additional setting for *K* (the ratio between termination- and admission-threshold), which is set to 2. The comparative results are summarized in the Table 4.7 below.

Table 4.7: A comparison of multi-bottleneck effect on CL and SM termination

Long-haul Aggr	CBR		VBR		MIX		TRC		SVD	
	CL	SM	CL	SM	CL	SM	CL	SM	CL	SM
PLT-2	42.6	62.9	43.3	66.1	41.6	63.3	44.1	66.0	50.1	68.1
PLT-5	47.1	85.9	49.5	86.6	49.6	83.3	49.0	86.6	53.2	97.9

Bottleneck 1	CBR		VBR		MIX		TRC		SVD	
	CL	SM	CL	SM	CL	SM	CL	SM	CL	SM
PLT-2	39.9	53.2	40.7	55.4	39.3	53.8	41.8	55.2	44.7	57.5
PLT-5	43.2	68.7	44.5	68.7	45.5	66.8	45.3	69.1	48.4	75.1

First, the results for SM show the same trend as in CL, that is, the multi-bottleneck effect causes the over-termination of long-haul aggregate and at the upstream bottlenecks. However, magnitude-wise, SM suffers a much worse impact. Across both topologies and all traffic types, SM over-terminates more compared to CL.

In fact, the above observation can be verified mathematically. For the ease of illustration, we present our derivation for the behavior of long-haul aggregate on a PLT-2 topology. Let $L.T$ be the total traffic of long-haul aggregate (A–C), and $S1.T$, $S2.T$ represent the total traffic of short-haul aggregates (D–E) and (E–F), respectively. After traversing the first bottleneck and having part of the long-haul aggregate marked (termination or admission marking, depending on the algorithm), the rate of the remaining unmarked traffic is denoted as $L.U$. In addition, let P be the termination threshold on both links, A be the admission threshold, and K be the ratio of the two (the thresholds need not to be the same, we only set it this way to simplify the illustration). Finally recall that the aggregates compute their termination percentage with $\frac{\text{sending-rate} - \text{sustainable-rate}}{\text{sending-rate}}$, so given the same sending rate, it's sufficient to compare the sustainable rate under the two algorithms. The formulas below give the calculation of the sustainable rate $L.S$ of the long-haul aggregate under CL and SM:

$$L.S_{cl} = L.U_{cl} - \frac{L.U_{cl} + S2.T - P}{L.U_{cl} + S2.T} * L.U_{cl} = \boxed{\frac{P}{1 + S2.T/L.U_{cl}}} \tag{4.2}$$

$$\begin{aligned} L.S_{sm} &= \left(L.U_{sm} - \frac{L.U_{sm} + S2.T - A}{L.U_{sm} + S2.T} * L.U_{sm}\right) * K = \frac{A * K}{1 + S2.T/L.U_{sm}} \\ &= \boxed{\frac{P}{1 + S2.T * K/L.U_{cl}}} \qquad (P = K{*}A \quad L.U_{cl} = K{*}L.U_{sm}) \end{aligned} \tag{4.3}$$

The comparison between Equations (4.2) and (4.3) shows that, given the same setup, the long-haul aggregate's sustainable rate under SM is smaller than CL (hence the larger over-termination) and the difference grows larger as the K increases. This provides the first intuition of why SM performs worse. Second, recall from Section 4.4.2, we discovered that the lack of direct indication of the load condition on the bottleneck with respect to the termination threshold makes SM vulnerable to inaccuracies. The same problem arises here again. It's possible that an underestimation of sustainable rate will cause unnecessary termination even when all bottlenecks' loads are below the termination threshold. To illustrate this mathematically, consider the condition that triggers termination, that is, the total sending rate is larger than the sustainable rate:

$$\begin{aligned} L.S_{sm} \;=\; \frac{A * K}{1 + S2.T/L.U_{sm}} &\;<\; L.T \\ L.T + \frac{L.T}{L.U_{sm}} * S2.T &\;>\; A * K \\ L.T + \frac{L.T + S1.T}{A} * S2.T &\;>\; A * K \end{aligned}$$

$$\boxed{L.T + \frac{L.T + S1.T}{P} * S2.T * K \;>\; P} \tag{4.4}$$

Note, $L.T + S1.T$ and $L.T + S2.T$ are the event-loads on two bottlenecks, even with both values less than P, there still exist K to make the left-hand of Equation (4.4) larger than P. It follows that, the larger the K, the more likely to have Equation (4.4) satisfied and trigger unnecessary termination.

We can summarize the multi-bottleneck effect for CL and SM comparatively as follows: in the first round of termination, both CL and SM have over-termination at the long-haul aggregate with SM worse than CL. After which, CL stops terminating if the resulting loads are below the threshold, SM on the other hand, might continue to terminate as long as Equation (4.4) is satisfied, hence results in further over-termination.

4.5.3 Simulation Validation

Our simple analysis has shown that both CL and SM suffer from the multi-bottleneck effect, with SM showing a more severe impact. In this section, we set out to test if our analysis holds under more general settings, and if so, how much over-termination can be expected from practical configurations. Our results from the packet simulator indicate that the exact amount of termination is quite dependent on the specific traffic matrix and the amount of event-load at the bottleneck. At the same time, the magnitude of the beatdown effect (and the subsequent under-utilization) is very insensitive to the specific flow arrival pattern - an observation which encouraged us to use the fluid model which computes the amount of traffic to be terminated based on the expected number of flows on the bottleneck. This fluid simulator enabled us to run a very large set of experiments with different traffic matrices and different bottleneck overload levels, thus providing a view on the extent of over-termination that can be expected in a wide range of scenarios. (The details of the fluid model can be found in Chapter 5.)

We randomly generated ~45,000 traffic matrices on the G-PLT-5 topology. Recall from Section 4.1.2, a G-PLT-5 topology consists of 5 (potential) bottlenecks, and 15 (potential) aggregates traversing bottlenecks ranging from one to five. In these experiments, the event-load on the "tightest" bottleneck ranged from 1-10X of the termination threshold, and in different experiments the actual number of bottlenecks (where traffic exceeded threshold) ranged from 0 to 5. Each experiment is simulated in both CL and SM. For SM, the value of K is randomly generated for each experiment, selected from the range of $[1.2, 3.0]$. The rationale is, given that the admission accuracy results from our results in previous section call for a separation of the admission and termination thresholds by at least a factor of 1.2, implying the choice of $K \geq 1.2$. We also believe that in practical setups with large $K \geq 3.0$ should be rare.

Before we present the results, a note is due on the evaluation metric. All previously reported termination results used the metric over-termination, the usage of which is fine in a single bottleneck topology. However in a more generalized environment with multiple bottlenecks, the following scenario can often arise. Imagine a case when many of the aggregates sharing a given link are "bottlenecked" elsewhere, then after each aggregates resolves itself independently, the resulting load on the given link may be below the termination-threshold. This type of over-termination is the natural outcome of the setup and is *not* a sign of multi-bottleneck effect. Hence, we need a refined metric that can prune away the noise (i.e. the above described scenario) and only capture those over-termination caused by the multi-bottleneck effect. To that end, for each experiment we first compute a "reference solution" that represents the system behavior without the beatdown effect, then the difference between the "reference solution" and the actual experiment result will represent the over-termination caused by the beatdown effect. The "reference solution" is computed as follows. We take each link in the topology separately and compute the "rate-proportionally fair" rates that each aggregate sharing this bottleneck will need to be reduced to (in proportion to their demands), so that the load on that bottleneck independently becomes equal to the termination threshold. After this is done independently for each bottleneck, we assign each aggregate the smallest of its scaled down rates across all bottlenecks. We then compute the "reference" utilization on each link by summing up the scaled down rates of each aggregate sharing this link. For the remaining of this section, all the over-termination results are reported in reference to the "reference" utilization.

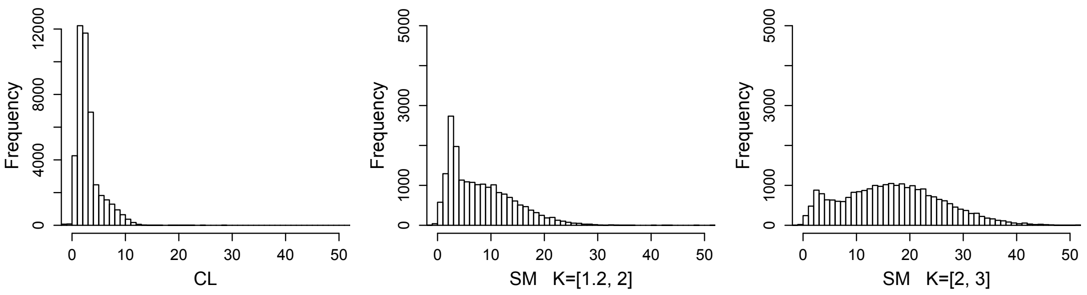

Figure 4.14: Multi-bottleneck effect on termination under general settings

Figure 4.14 shows three over-termination percentage distribution graphs, (a) for CL (b) for SM with $1.2 \leq K \leq 2.0$ (c) for SM with $2 \leq K \leq 3$. The encouraging results for CL shows that for the large range of multi-bottleneck scenarios we ran, the over-termination error is mostly within 10%, indicating the beatdown effect we report here is probably of limited practical significance. On the other hand, the error for SM shows a wider spread. In Figure 4.14 (b), which in our opinion is the case of practical importance, the over-termination for SM is below 10% in 65% of the experiments, is within 20% for about 30% of the experiments, and between 30% and 40% for the remaining 5%. The difference between Figure 4.14 (b) and (c), namely the even wider error range in (c), confirms the positive correlation between K and the magnitude of the over-termination error.

Summary In this section we studied the multi-bottleneck effect on the performance of termination control for CL and SM. We demonstrated that both algorithms' performance is adversely effected by the presence of multi-bottleneck. Yet, for CL, in all experiments we conducted, the magnitude of the over-termination was relatively small, indicating a reasonable performance in most cases. We show through both theoretical derivation and simulations that SM gives worse performance compared to CL, especially if K is large. This is in part because the over-termination error is amplified by the factor K, and also because the lack of a direct indication of the bottleneck condition with respect to the termination threshold again results in unnecessary termination. While we believe that the extent of this over-termination is tolerable for practical purposes, it needs to be taken into account when considering performance tradeoffs of the two mechanisms.

4.6 Multi-Bottleneck Effect on Admission

In the Section 4.5, we investigated the multi-bottleneck effect on termination, in this section we focus on its effect on admission. Recall that in termination, the multi-bottleneck effect is interpreted as the over-termination of the long-haul aggregates. In admission, it's natural to define the multi-bottleneck effect (or the beatdown) as the under-admission of the long-haul aggregates, that is, the long-haul aggregates are prevented from admitting their share of new flows. Note that if the short-haul aggregates do not have enough demand to compensate for the under-admission of the long-haul aggregates, then the bottlenecks could be under-utilized. We are interested in answering the same questions as in the case of termination, in particular 1) What is the exact cause of the beatdown of long-haul aggregates, 2) Is there any performance difference between CL and SM, and 3) What is the extent of unfairness and would it cause bottleneck under-utilization.

4.6.1 Understanding Multi-bottleneck Effect on Admission

Recall from Section 4.5, the primary cause of the beat-down effect for termination is excessive marking of the long-haul aggregates at the subsequent bottlenecks. In the case of admission, though the excessive marking is still present, as we will show, it is not the primary cause of the beat-down effect.

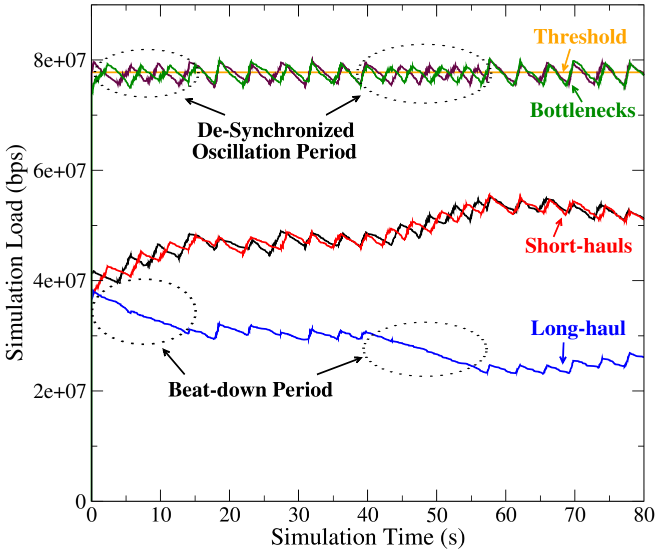

Figure 4.15: An illustration of multi-bottleneck effect on admission

Figure 4.15 depicts an 80-second snapshot of a network with PLT-2 topology running under the CL admission algorithm. The simulation is a setup with the intent of letting the long-haul (A-C) and two short-haul (D-E and E-F) aggregates have equal shares of the bottleneck links, that is, each takes half the bandwidth allowed by the admission threshold. In particular, each aggregate is given roughly the same starting load (with a little randomness), and each has exactly the same demand-load which is 5x its starting load. The point of the setup is to best capture the multi-bottleneck effect. If there was no unfairness to the long-haul aggregate, then in Figure 4.15 the lines for the three aggregates should be close to each other and stay roughly horizontal. On the other hand, with the presence of unfairness, the short-haul aggregates with their large enough demand (5x) will gradually "eat away" the bandwidth that belongs to the long-haul aggregates, hence we should see lines diverge from their starting position, with the long-haul aggregate going downwards. Figure 4.15 shows that during the 80 seconds, there are two periods of time showing beatdown of the long-haul aggregate, in particular, 0-15s and 40-58s during which period it did not admit new flows. The rest of time the long-haul aggregate does get the chance to admit, and hence maintained (and even regained) some of its share of bandwidth.

The immediate question is what is the exact cause of the multi-bottleneck effect, and why is the beatdown effect present but not persistent. To explain this, we first introduce the concept of *oscillation period*. Observe in Figure 4.15 that the lines representing the bottleneck loads exhibit a chain-saw pattern oscillating around the admission threshold. We refer to each interval where the bottleneck load first increases and then decreases as an oscillation period, since it roughly corresponds to the alternation between the two states 1) *admission-state* where the bottleneck load is below the admission threshold and traversing aggregates are allowed to admit new flows, 2) *congestion-state* where the bottleneck load is above the threshold and it marks enough packets to trigger no-admission at the relevant aggregates. It turn out the level of beatdown depends on the synchronicity between the bottlenecks' oscillation periods. To see this, imagine the following scenario: the long-haul aggregate (A–C) stops admitting because bottleneck1 (A–B) is in congestion-state; in the meantime, the second short-haul aggregate (E–F) has admitted enough flows to drive bottleneck2 (B–C) into congestion-state. Therefore, even when bottleneck1 now changes into the admission-state, the long-haul aggregate still cannot admit because of bottleneck2, leaving the first short-haul aggregate (D-E) alone admitting new flows. If the pattern repeats, the long-haul aggregate has far less time to admit new flows compared to its short-haul competitor, hence its share of the bandwidth will be gradually beaten down. Put another way, if both bottlenecks have synchronized oscillation periods, then there is no beatdown and long-haul aggregate will successfully admit new flows. With de-synchronized periods, the long-haul aggregate has to stop admitting as long as at least one of them is in the congestion state. This reflects the simple intuition that the aggregates traversing multiple bottlenecks see congestion more often, and therefore spend more time not admitting new flows than short-haul aggregates. Note in Figure 4.15, we highlight two beatdown periods of long-haul aggregates and the de-synchronized oscillation periods of the bottlenecks, which are clearly correlated in time.

Now that we know de-synchronization of the oscillation periods is the main cause of the beatdown, it remains to be determined the deciding factors for the de-synchronization of the oscillation period. Naturally, if the oscillation periods on the different bottlenecks have different sizes and shapes, (e.g. the aggregated flow arrival rate differs on each bottleneck), it would be impossible to synchronize them. The characteristics of the oscillation periods depend on many factors, such as expected flow arrival rate and flow length, as well as the admission algorithms' settings. Considering PLT-2, if we could set the fore-mentioned factors exactly the same for both bottlenecks, and start the simulation with two perfectly synchronized periods, the question is would it maintain in that way and hence eliminate the beatdown of the long-haul aggregate?

We discover that the answer is no. The problem lies in statistical variation of the flow arrivals. The way we usually characterize traffic (in theory or practice) is by its statistical expected properties, for instance, as mentioned in Section 4.1.4 our simulations assume that flow arrivals follow a Poisson distribution and have exponentially distributed length with expected means. However, the actual flow arrivals randomly deviate around the mean. It is this random variation that causes the originally synchronized periods become de-synchronized, and vice versa. This phenomenon is illustrated in Figure 4.15, in which we see the oscillation periods start out de-synchronized, then gradually become synchronized, and changed again later on. In fact, the sensitivity to the timing of flow arrivals makes it impossible to correctly estimate the extent of unfairness based on expected traffic arrival behavior. As an example, Figure 4.16 plots the load-vs-time graph from 4 experiments running SM on PLT-2. They are configured with the exact same parameter settings and expected traffic load, only different flow arrival patterns (The lines with the same color belong to the same experiments). We see that while the 4 experiments give virtually indistinguishable bottleneck utilization behaviors, the bandwidth distribution between the short-haul and long-haul aggregate looks drastically different. The graph conveys two points 1) our fluid model that had worked well in Section 4.2.2 would not predict its corresponding packet simulation with bounded accuracy in this situation, since it would generate the exactly same output for all 4 experiments. This prevents us from using the fluid model to conduct a large range of experiments. 2) Even with packet simulation, the already simulated period cannot be used to predict future behaviors (since 3 out of 4 experiments show no persistent trend). The two points together imply, unlike in termination, it would be impossible to accurately quantify the magnitude of unfairness under general settings (or even simple settings) with simulations.

Having said the above, there are still some simple qualitative arguments that can be made based on expected traffic load. For instance, we would expect a more severe beatdown if the competing short-haul aggregates have high traffic demands. Furthermore, despite the beatdown of the long-haul aggregates, as long as the short-haul ones sharing the bottleneck have enough demand to fill to bandwidth, there should not be any underutilization at the bottleneck level. There are some additional qualitative properties that are related to admission algorithms settings, which we will discuss in detail in the next section.

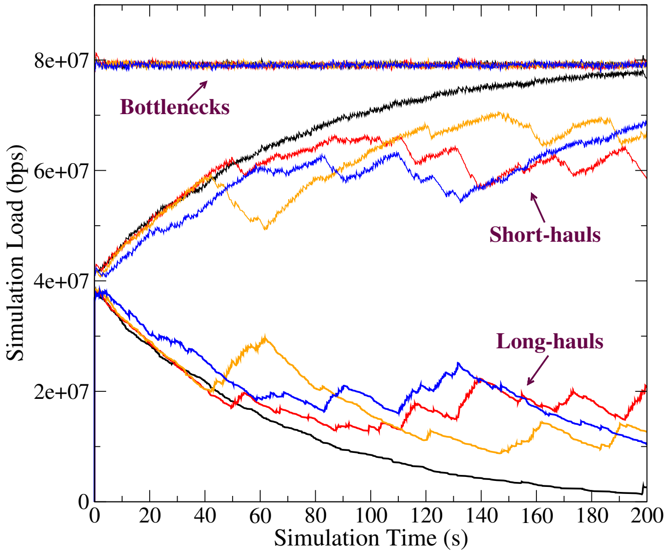

Figure 4.16: Sensitivity of IE-aggregation admission behavior to traffic arrivals

4.6.2 Comparison Between CL and SM

We have introduced the causes of the multi-bottleneck effect in admission, and argued why it is not possible to accurately predict the extent of the effect based on expected traffic input. Now we turn our attention to some of the admission algorithms details that might have an impact on unfairness. Though we still won't be able to exactly quantify the magnitude in a general case, we might be able to gain some intuitive understanding, in particular the difference (if any) between CL ad SM.

We start by running the same simple scenarios described in Section 4.6.1 (PLT-2, 5x demand-load, start with equal bandwidth distribution between the long-haul and short-haul aggregates). We keep all the settings the same for CL and SM, even the sequence of flow arrivals. (Recall, however, from Section 4.2.2, there is a difference in marking semantics between CL and SM, hence SM always has a smaller cle-threshold.) We find in these experiments, the behaviors of CL and SM are clearly different. Figure 4.17 shows such an example. While both CL and SM both show signs of beatdown (as the lines represent the load of the long-haul and short-haul aggregates diverge from each other), SM seems much more drastic with faster diverging lines. It is worth pointing out that we cannot make a general, long trend statement, such as "SM suffers from a more severe beatdown effect than CL" based on the interpretation of this graph. Because as we found out in Section 4.6.1, due to the sensitivity to the traffic arrival, the period we simulated (no matter how long) cannot be used to predict the future behavior of the network (imagine it is possible for CL to continue to diverge, and SM suddenly becomes converged). However we can say, for this short interval, the graph suggests more pronounced unfairness in SM than in CL. Now we set out to understand if the observation is true and why it is the case.

The explanation again lies in the "oscillation period" (defined in Section 4.6.1). We see from Figure 4.17 that CL had a much larger (and visible) oscillation period than SM (a magnifying window on the right shows the relative sizes of the two). Recall in Section 4.6.1, the beatdown effect is caused by the de-synchronization of the bottleneck oscillation period, and de-synchronization is in turn caused by the random variation in traffic arrivals. It's easy to see, given the same magnitude of random change, a small oscillation period can get de-synchronized much more easily. Then, why does SM have

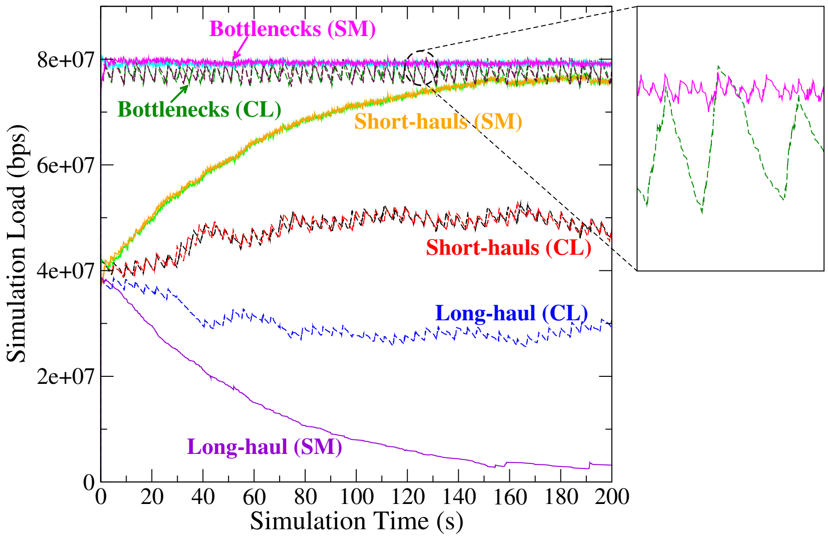

Figure 4.17: Comparison of multi-bottleneck effect on CL and SM admission

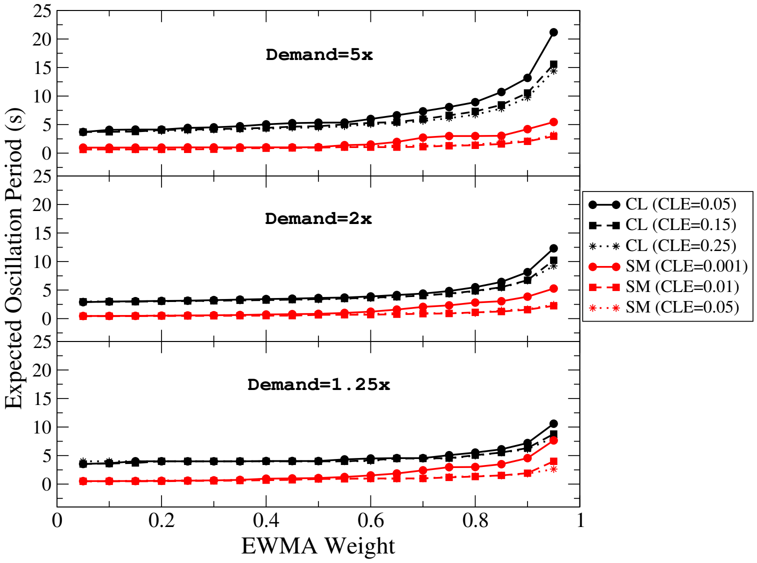

Figure 4.18: Expected length of oscillation period of CL and SM

a smaller oscillation period? It's due to the difference in the two algorithms' marking behavior. Recall, when the bottleneck load is above the admission threshold, CL, after filling up the virtual-queue, would mark ***all*** traffic, while SM only marks the rate ***exceeding*** the admission threshold. As a result, CL marks much more traffic, and it would take it longer to age out the interval-cle and to restart the admission. In fact, we have presented this very property in Section 4.2.2, only there we demonstrate it in the context of "oscillation range" (which is the deviation of the actual load around the average load). As in Section 4.2.2, we can use our fluid model to calculate the expected oscillation period given the algorithm, the parameter settings, and the expected traffic load.[11] In Figure 4.18, we plot the expected oscillation period under various combinations of demand-load, ewma-weight and cle-threshold. We can see that CL (displayed in black) indeed has a larger oscillation period comparing to SM, especially with large demand-load and ewma-weight.

To further verify our argument, we conduct a larger scale packet simulation of ~700 experiments on the G-PLT-5 topology. In these experiments, the all 5 bottlenecks have the same settings (capacity, admission threshold and arrival rate). The initial bandwidth distributions among the aggregates are randomly generated. The demand-load for each aggregate is x times their initial bandwidth, where x is 1.2, 2, 3.5 and 5. (Note, the x is same for all aggregates of the same experiment. This also means the demand-load on each bottleneck will be 1.2x, 2x, 3.5x or 5x). The rationale of this setup is again to capture the presence of multi-bottleneck effect, just as the example in Section 4.6.1. If there is no beatdown, then each aggregate should maintain its initial share of the bandwidth. This in turn provides us with a metric to quantify the effect, that is, for each aggregate, we calculate an average load over the simulation duration, then compare it to the aggregate's starting load. The more they differ, the more significant the beatdown effect. We should point out however, the absolute magnitude of the metric bears no practical meaning, it is used solely for comparison between two experiments. In fact, even as a comparison metric it should be used with caution, since it would be meaningful only when the aggregate's load during the entire simulation period exhibits roughly

[11] Even though in the previous paragraph we established that the fluid model is not suitable to predict the individual aggregate's behavior, the fluid model is quite accurate when it comes to predict various bottleneck behaviors, as we will show in Chapter 5. This is the reason why we can use it for calculating the expected oscillation period.

monotonic behavior (increase, decrease or stay the same). For our short simulation period (60s), this is indeed the case (confirmed by visual inspection). Finally, the cle-threshold and ewma-weight are randomly generated as well.

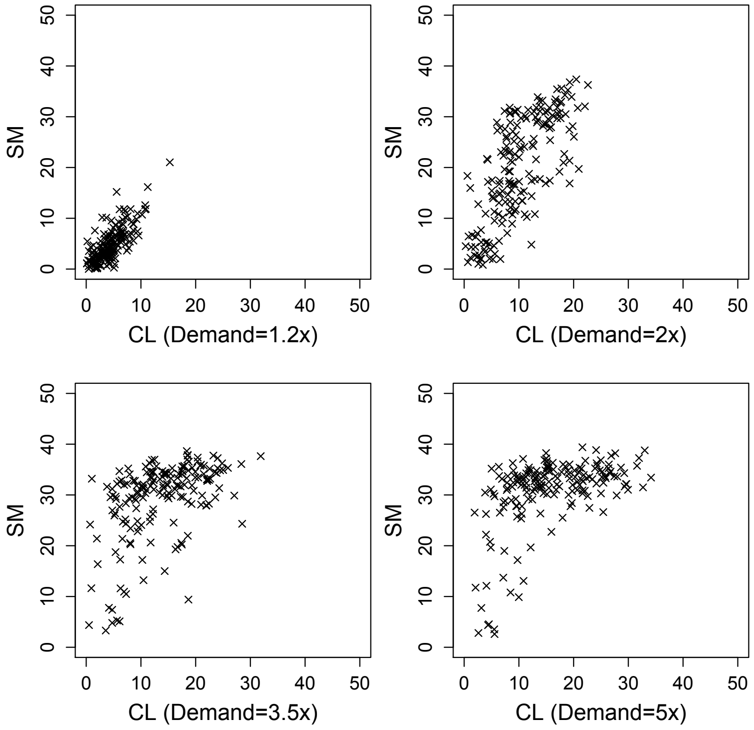

Figure 4.19: Comparison of multi-bottleneck effect on CL and SM admission under general settings

We produce the scatter plots of percentage deviations of the average load from its starting load. Note there are potential 15 aggregates on the G-PLT-5 topology; we use the one with the worst deviation. In Figure 4.19 CL and SM are plotted against each other for each demand-load level. The results show that when demand-load is small, such as 1.2x, there is virtually no difference between the two algorithms. As the demand increases, the deviation in SM is significantly larger. What can be concluded from this experiment is, while there is no way of stating the long trend behavior, if we zoom into short intervals, then with the presence of high demands, the long-haul aggregates tend to be beatdown more quickly in SM than in CL.

Summary In this section, we study the multi-bottleneck effect in admission. We identify the cause for the beatdown of the long-haul aggregates, its influencing factors, and the causes for the performance difference between CL and SM. Furthermore, we conclude that due to sensitivity to statistical deviations in flow arrival, the long trend property of beatdown or its exact magnitude at any point of time cannot be predicted accurately with simulation. We end this section on a somewhat brighter note with the observation that for the beatdown effect to be significant for admission, not only the demand-load should be substantial, but also the duration of the load should be long enough. Under "normal" conditions, one should not expect large prolonged substantial demand-loads, since networks are typically provisioned. In the exceptional cases (e.g. flash crowds) where high overloads can occur, they are not likely to be of very long duration. In any case, under those circumstances, unfairness and even starvation of some aggregates is still preferential to indiscriminately dropping packets of all flows that would occur in the absence of admission control. Hence, in practice, the effect of the multi-bottleneck effect we report here is probably of limited significance.

4.7 Queuing and Packet Loss Effect on Termination

All previously reported results assume no packet loss, now we will study CL and SM's performance in the presence of packet loss and queuing. Recall PCN has a two-threshold architecture, and the space separation between the admission and termination thresholds can shield the admission function from being affected by queuing and loss. Hence, in this work we focus our attention on the performance of the termination of the two algorithms. The problem with loss on termination is intuitively easy to understand. Recall both CL and SM rely on egress measurement of the unmarked traffic to calculate the aggregate's sustainable rate. If along the path, unmarked packets were dropped, the egress will have an under-estimated sustainable rate which in turn causes over-termination. One natural way to solve the problem is to implement preferential dropping and to have the bottlenecks preferably drop ***marked*** packets in the case of congestion. While this fix not only increases the complexity of the equipment, it also raises incompatibility with other proposed PCN algorithms (Some proposals require preferential dropping on unmarked packets). Therefore, it's important to go beyond simple intuition and actually evaluate the algorithms' performance under a variety of practical scenarios, so that we can quantify the impact of loss and queuing on termination and determine whether preferential dropping of the marked packets is indeed needed.

4.7.1 Single Bottleneck Scenario

For the remainder for this section, when referring to loss, we mean that tail dropping is performed (unless otherwise specified). We refer to the over-termination caused by tail dropping and queuing as the ***loss effect*** and the ***queuing effect***, respectively. In this section we analyze the properties of these two effects on the simplest topology, a single link. We start by providing an example to illustrate the loss effect and to test if preferential dropping can indeed avoid the over-termination. We then present another trivial example to capture the essence of the queuing effect. On a single link, CL and SM give comparable performance. We will therefore present the analysis and the results in the context of CL only.

An Example of the Loss Effect Consider the following scenario where two aggregates go through a bottleneck which has its termination threshold set to its link speed. Two aggregates have the same rate, and together they create 2x excess over the bottleneck link speed (and 2x over the termination threshold as well). Both aggregates have half of their traffic termination-marked already from previous bottlenecks. In a no loss scenario (suppose the bottlenecks' link speed were huge), the resulting system has the two aggregates each terminate half of their traffic. If preferential dropping is used, the bottleneck first drops termination-marked traffic, which, in this case, just happens to compensate all its overload, then the resulting system is the same. If tail dropping is used, assuming dropping is uniformly distributed, half of unmarked traffic will be dropped, resulting in the two aggregates terminating 75% of their traffic. We run a simulation on this simple example, and the actual termination percentages are summarized in the table below. The results confirm our intuition, that without preferential dropping, the dropping of unmarked packet can cause over-termination. Note that the expected worst case 75% termination is not achieved, because dropping is not uniformly distributed. In particular, marked packets are more likely to get dropped (since the packets arrivals are not uniform. Packets that come in burst are more likely to be marked and dropped.)

Table 4.8: Loss effect (tail drop) on termination and preferential dropping

	No Loss	Preferential Drop	Tail Drop
Aggregate 1	51.32	51.44	70.74
Aggregate 2	51.19	51.03	68.60

An Example of the Queuing Effect Consider the simple case where one aggregate travels through a bottleneck which has the termination threshold equal to its link speed. The aggregate creates 2x excess overload on the bottleneck. If the queue on the bottleneck is small enough to be neglectable, then half of traffic will be dropped and the remaining half is equal to termination threshold, hence unmarked. The aggregate will terminate half of its traffic. At the other extreme, if the queue is big enough to hold packets for at least the duration of one measurement period, then during the first round of termination there will be no dropped packets. Instead, the traffic will arrive at the egress at a rate which equals to the bottleneck link rate (half of its ingress sending rate) and half of the arriving traffic is termination marked. The aggregate will get a sustainable rate of only 1/4 of its sending rate and has to terminate 75% of traffic. Simply put, the increase in queue size will cause an under-estimation of the sustainable rate and in turn cause over-termination. We simulate this scenario with different queue sizes (measured in maximum possible queuing delay). The actual termination percentage is summarized in the table below. The results confirm our queuing effect analysis. Note, in this simple scenario, the deciding factor of the termination percentage is the sustainable rate estimation in the first measurement period. In our simulation the period is set to 100ms. Therefore, a queue that holds more than 100ms of traffic will not cause further over-termination.

Table 4.9: Queueing effect on termination

Queue Size	2.5ms	10ms	25ms	50ms	100ms	200ms
Termination Perc.	49.42	52.06	59.42	71.69	73.66	74.12

4.7.2 Multiple Bottlenecks Scenario

Recall from Section 4.5, the ***multi-bottleneck effect*** is the “beatdown” (over-termination) of long-haul aggregates in the presence of multiple bottlenecks (and the potential subsequent under-utilization of some bottleneck links). In this section we first refresh the reader on several key properties of the multi-bottleneck effect in the no loss scenario. Then we investigate how the multi-bottleneck effect interacts with the over-termination caused by queuing and loss (which we studied previously in Section 4.7.1). For the ease of illustration, we focus on the behavior of the long-haul aggregates and present our examples and theory on a simple two bottleneck topology PLT-2 (Figure 4.2 (c)). A comprehensive study of the bottleneck behavior with more general settings can be found in Section 4.7.3.

Multi-Bottleneck Effect on Termination, A Refresher Recall our main conclusion on the multi-bottleneck effect on termination is that both algorithms suffer from it, though SM to a larger extent. Here we briefly recite some equations from Section 4.5, that serve as a refresher as well as laying some theoretical background for our future analysis on multi-bottleneck effect involving queuing and loss. On the PLT-2 topology, let $L.T$ be the total traffic of the long-haul aggregate (A–C), and $S1.T$, $S2.T$ represents to total traffic of the short-haul aggregates (D–E) and (E–F), respectively. After traversing the first bottleneck, part of the long-haul aggregate will be marked (termination or admission marking, depending on the algorithm), we denoted the rate of the remaining unmarked traffic as $L.U$. In addition, let P be the termination threshold on both links, A the admission threshold, and K the ratio of the two. Equation (4.5) and (4.6)[12] the calculation of the sustainable rate $L.S$ of the long-haul flow under CL and SM, which shows that the long-haul aggregate's sustainable rate under SM is smaller than CL (this means SM over-terminates more)

$$L.S_{cl} = \frac{P}{1 + S2.T/L.U_{cl}} \tag{4.5}$$

$$L.S_{sm} = \frac{P}{1 + S2.T * K/L.U_{cl}} \tag{4.6}$$

Equation (4.7) describes the condition for SM to trigger termination, which implies, even if event-loads on the bottleneck are smaller than the threshold, there still could be false trigger terminations, given a large enough K. This is the second reason why SM performs worse than CL.

$$L.T + \frac{L.T + S1.T}{P} * S2.T * K > P \tag{4.7}$$

Loss vs. Multi-Bottleneck Effect To study the multi-bottleneck effect in the presence of loss, again consider an example with the PLT-2 topology but with a concrete traffic

[12] These two equations and Equation (4.7) below are the same as the ones in Section 4.5, we just list them here for viewing convenience

matrix. Suppose, at the time of the failure event, both bottlenecks' loads are twice the termination threshold, with long- and short-haul aggregates each constituting 50% of the load. The trivially "good" solution is to have the three aggregates each terminate half of their traffic. With CL in a no-loss scenario, the unmarked traffic from long-haul aggregate (A–C) plus the load from short-haul (E–F) creates 1/3 excess [13] over the termination threshold on the second link. These additional markings that long-haul aggregates experience reflect the multi-bottleneck penalty (which makes the total termination roughly 66%). Now assuming the link speed on the second bottleneck B–C is set so that the event-load creates x percent excess, in other words, B will drop x percent of its passing traffic. In the first case where $x > 1/3$, assuming dropping is uniformly distributed, more than 1/3 of the unmarked traffic will be dropped and the resulting unmarked traffic will be smaller than the termination threshold. Hence no additional multi-bottleneck penalty for the long-haul aggregates; the termination percentage will be decided by the loss effect alone. If $x < 1/3$, after dropping, the remaining unmarked traffic will still be larger than termination threshold, the multi-bottleneck effect will kick in and mark enough packets so that the combined effect (loss + new marking) is 1/3. In the table below we list the termination percentages of the long-haul aggregate from the packet simulation of the above scenario. It proves that with an overload larger than 1/3 (e.g. $x = 1/2$), the over-termination is dominated by the loss effect; while if x is less than 1/3, the presence of loss makes no difference as the over-termination is decided by the multi-bottleneck effect.

Table 4.10: Loss effect vs. multi-bottleneck effect

No Loss	Loss ($x = 1/5$)	Loss ($x = 1/3$)	Loss ($x = 1/2$)
62.80	63.00	63.66	79.18

[13] 1/3 excess is computed as follows: assuming the termination-threshold is unit 1, then all three aggregates have a load of unit 1 (since the sum of the load of long-haul aggregate and any short-haul aggregate creates a load 2x the termination threshold). Next the excess on the second bottleneck is equal to $(1 \times 1/2 + 1 - 1)/(1 \times 1/2 + 1) = 1/3$

To summarize, the intuition obtained from this example is that the penalty for loss and multi-bottleneck effect is not additive; instead it's the larger of the two. Since the magnitude of the multi-bottleneck effect has previously been quantified via simulation in Section 4.5.3, a natural question would be: are there any circumstances where the loss effect is significantly larger than the multi-bottleneck effect. The example above seems to suggest that the answer is when there is a large overload x. In fact, this is not always true. While a large x indeed causes a large loss effect, it might imply a large multi-bottleneck effect as well (by definition, lots of excess traffic over the link speed means even more traffic over the termination threshold.) A situation where it is true is when the overload consists of a lot of marked traffic, which means even though the traffic is exceeding link speed by a lot, the amount of unmarked traffic remains small. Then proportionally the long-haul aggregate will not get many additional markings, hence a small multi-bottleneck effect. We observe that if a scenario satisfies the following conditions: cross traffic is mostly unmarked short-haul traffic and the termination-threshold is close to the link threshold, then the multi-bottleneck effect is likely to dominate. Note, we only present the illustration in the context of CL, the same conclusion holds for SM, we omit the details for conciseness.

Queuing vs. Multi-Bottleneck Effect Recall from Section 4.7.1, with all other conditions being the same, changing the queue length will result in a change in the rate of the unmarked traffic exiting the queuing link over a relevant (short) timescale. Using the same PLT-2 example and terminology, now suppose link speed of the first bottleneck is small enough to cause queuing/dropping on Link A–B, then the long-haul aggregate $L.U$ value will change accordingly, in particular, the larger the queue length, the smaller the $L.U$. Recall that Equation (4.5) states the relationship between $L.U$ and the aggregate's sustainable rate $L.S_{cl}$ under CL. It's easy to see that a smaller $L.U$ means the reduction of the overall sustainable rate. Hence, for CL, the increase in queue length amplifies the multi-bottleneck effect.

For SM, Equation (4.6) indicates a similar behavior; however, it suffers less than CL. There are two potential reasons. First we will show that under certain common conditions, given the same change in $L.U$, the overall sustainable rate change for SM is smaller than for CL, especially if K is large.

$$\begin{aligned}
\Delta L.S_{cl} &= \frac{P}{1+S2.T/L.U_1} - \frac{P}{1+S2.T/L.U_2} \\
&= \frac{P * S2.T * (L.U_1 - L.U_2)}{(L.U_1 + S2.T)(L.U_2 + S2.T)} \\
\Delta L.S_{sm} &= \frac{P}{1+S2.T*K/L.U_1} - \frac{P}{1+S2.T*K/L.U_2} \\
&= \frac{P * S2.T * (L.U_1 - L.U_2)}{(L.U_1 + S2.T*K)(L.U_2 + S2.T*K)/K} \\
\Delta L.S_{cl} > \Delta L.S_{sm} &\Rightarrow (L.U_1{+}S2.T)(L.U_2{+}S2.T) \; < \; \frac{(L.U_1{+}S2.T*K)(L.U_2{+}S2.T*K)}{K} \\
&\Rightarrow K * L.U_1 * L.U_2 + (S2.T)^2 * K \; < \; L.U_1 * L.U_2 + (S2.T*K)^2 \\
&\Rightarrow (K-1) * L.U_1 * L.U_2 \; < \; K * (K-1) * (S2.T)^2 \\
&\Rightarrow \boxed{L.U_1 * L.U_2 \;\; < \;\; K * (S2.T)^2}
\end{aligned}$$

The intuition of our derivation is the following: for SM, if K is large, or on the second bottleneck the short-haul cross traffic is larger than the unmarked traffic from the long-haul aggregates; then SM's reduction in sustainable rate is smaller compared to CLs. (The condition is likely to arise if the link speed and threshold settings are comparable in value for both bottlenecks.)

The second reason is related to the property of SM implied by Equation (4.7). When K is large, the over-termination caused by this effect becomes dominant. Even though the presence of queuing causes the long-haul aggregates to terminate more in its first round, as long as Equation (4.7) is satisfied, the aggregates will keep on terminating until the bottleneck load falls below the “danger zone”. In these cases, the dominating over-termination from the multi-bottleneck effect will render the influence of queuing invisible. The table below contains the terminating percentage for the long-haul aggregate in each setup. The third row (the difference between LargeQueue and SmallQueue) shows that for the same queue size change, CL has the larger increase in its termination percentage, while for SM the differences are smaller; with a larger K ($K = 2$), it shows virtually no effect.

Table 4.11: Queueing effect vs. multi-bottleneck effect

	CL	SM (K=1.2)	SM (K=1.5)	SM (K=2.0)
SmallQueue (2.5ms)	63.46	69.02	76.25	88.05
LargeQueue (100ms)	78.81	81.73	84.51	87.74
Difference	15.35	12.71	8.26	-0.31

To summarize, for both CL and SM, an increase in queue length has the effect of amplifying the multi-bottleneck effect, and in turn causes larger over-termination. The effect is less pronounced for SM, especially when K is large.

4.7.3 Simulation Validation

To validate our findings in the previous sections under more general settings, we randomly generated 1200 traffic matrices on the G-PLT-5 topology (Section 4.1.2). In fact, these experiments are generated in a similar manner as those large scale multi-bottleneck simulations in Section 4.5.3. Yet there are some additions to the setup, that is, for each traffic matrix, we run three experiments with different link speeds and queue size settings: 1) set the link speed very large (hence no loss occurs), 2) randomly select the link speed from the range 1-2X the termination threshold, and set the queue size equals to 2.5ms of link capacity, and 3) use the same link speed as in 2), but with a larger queue randomly chosen between 256 to 5120 packets. We term the three types of experiments NoLoss, SMQueue and LGQueue, respectively. The results focus on the over-termination on the tightest bottleneck where the over-termination is compared to the "reference" over-termination defined in Section 4.5.3. (Note, given the aforementioned similarity in the setup, one would expect that the results for the NoLoss set to be comparable to the multi-bottleneck effect result we obtained in Section 4.5.3, which is indeed the case.)

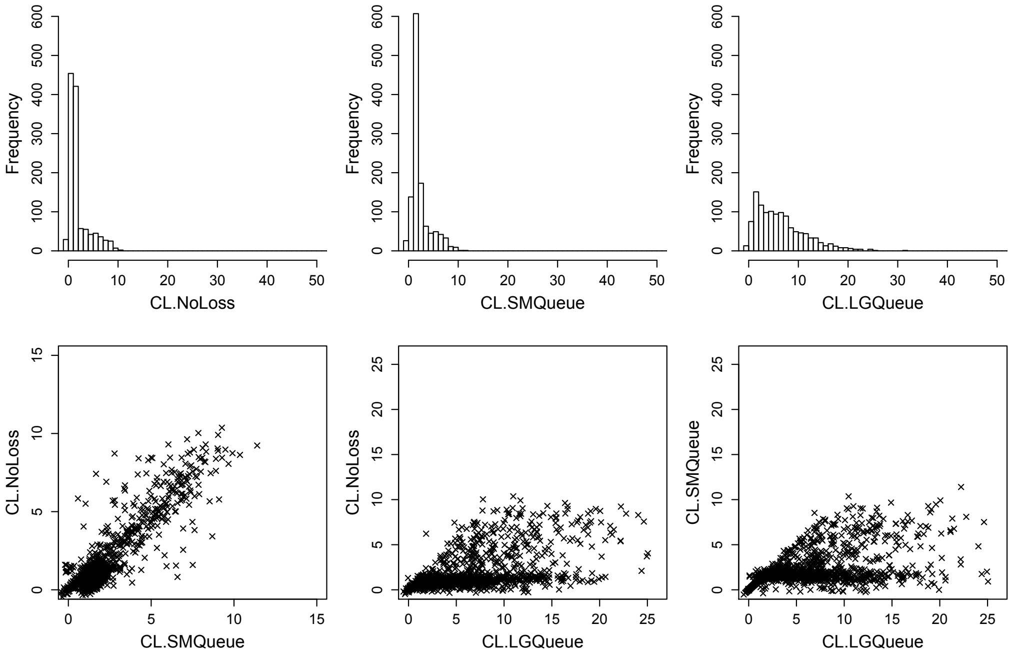

Figure 4.20: Effect of loss and queueing on CL termination

Figure 4.20 summarizes the results for CL. The three graphs in the top row give the distribution of the over-termination for each type of experiment (NoLoss, SMQueue and LGQueue). The second row consists of pair-wise scatter plots of the experiments. (The distribution graphs provide the overall statistics, the scatter plots give a performance comparison among NoLoss, SMQueue and LGQueue for each of 1200 experiments). The same information is plotted for SM in Figure 4.21 with the only difference being that for SM, we break the result into two categories based on the value of K. Experiments with a smaller K ($1.2 \leq K \leq 2.0$) are plotted in red, while the ones for larger K ($2.0 < K \leq 3.0$) are displayed in blue.

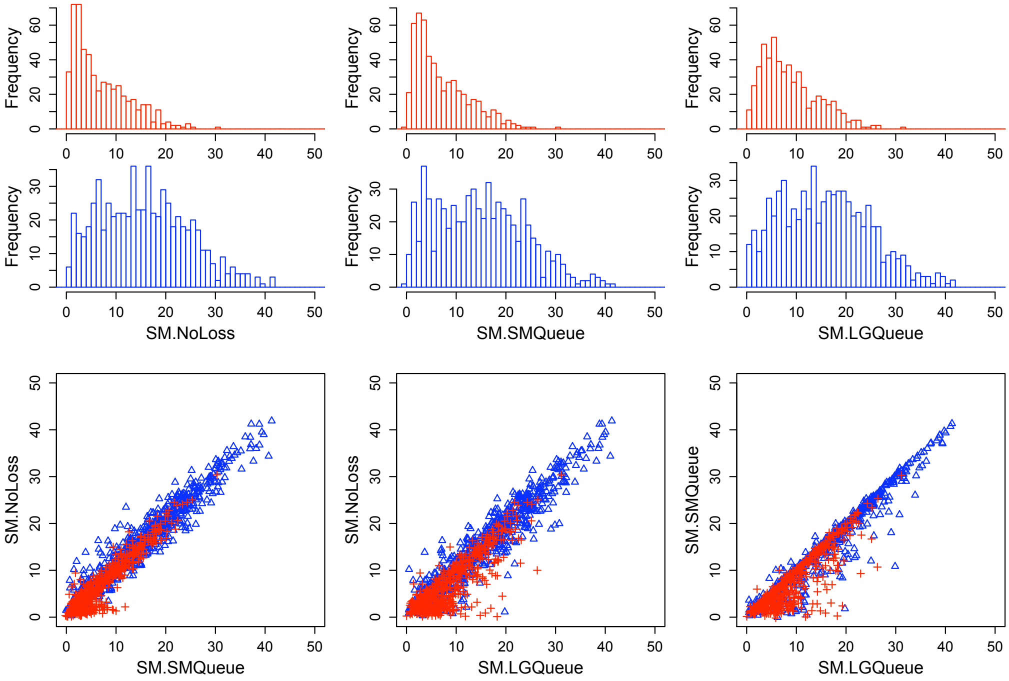

Figure 4.21: Effect of loss and queueing on SM termination

The simulation results confirm all our findings. For both CL and SM, the distribution graph for NoLoss and SMQueue show virtually no difference, indicating overall performance is dominated by the multi-bottleneck effect, despite the loss and the tail dropping (the scatter plot of NoLoss and SMQueue further confirms this). The increase in queue length shows a significant impact on CL (see the distribution graph for LGQueue and the scatter plot of SMQueue and LGQueue). While for SM, the effect is much less, and is not even visible if K is large. An interesting observation is that while in the no-loss situation CL does better than SM, the effect of queuing brings the performance of the two on the same level.

Summary In this work we studied the effect of loss and queuing on the performance of termination control for CL and SM. We demonstrated using simple examples that both loss (with tail drop) and queuing may adversely affect the termination performance. We then analyze the two effects in more generalized settings in conjunction with the multi-bottleneck effect. We found that in many practical scenarios, the multi-bottleneck effect tends to dominate the over-termination and the presence of loss does not further impact the performance much. The result seems to suggest that it may not be absolutely necessary to implement preferential dropping of marked packets with PCN. A difference between the two algorithms is that queuing affects the performance of CL much worse than that of SM. In fact, with large queues, the performance of CL and SM becomes comparable. This is an interesting fact to consider if PCN were to be used in an environment where traffic is mix of voice and video which requires the configuration of large queues.

4.8 Performance Summary

In this section, we conclude our performance evaluation work for CL and SM. The overall takeaway is that CL works stably across a wide range of scenarios; for SM, while giving adequate performance in most cases, the simpler mechanism does come at a cost, especially in termination. We organize our summary in two parts. The first part is dedicated to the performance of CL only. The second parts focuses on SM by highlighting the performance tradeoff and the its applicability range.

Performance of CL

In a nutshell, CL performs well in a large number of experiments. Our evaluation indicates that its admission control function works well at sufficient levels of bottlenecks for all real-time traffic types and flow arrival models we studied, and is rather insensitive to the settings of the key parameters and the level of ingress-egress aggregation. The algorithm suffers from the well-known phenomenon of unfairness towards flows traversing multiple bottlenecks, and long periods of relatively modest demand overload may result in a severe “beatdown” of the long-haul connections. However, this rather depressing property is not algorithm-specific and can be traced to the de-synchronization of congestion periods of different bottlenecks. We argue, however, that prolonged periods of steady demand overloads may not be common place in the PCN environment, and hence the practical significance of the beatdown effect may be limited.

Our investigation of the CL termination algorithm revealed that it worked well for the range of tests we conducted. We identified several factors that may result in over-termination: (1) inaccuracies of rate estimation and rounding errors due to the granularity of termination at very low ingress-egress-aggregation levels; (2) flows traversing multiple bottleneck in the presence of short-haul cross traffic. However, in all experiments we conducted, the magnitude of the over-termination was relatively small, indicating that the termination algorithm is likely to provide reasonable performance in most cases. In addition, the presence of packet loss does not impact the termination performance by much in most practical cases, suggesting that preferential dropping may not be a necessity for adequate termination performance.

Tradeoff and Applicability of SM

An obvious restriction necessary for the SM approach is that the ratio of admission- to termination-threshold remains the same on all links in the PCN region. While clearly a limitation [5][26], this does not appear to be particularly crippling, and does not appear to outweigh the benefits of reducing the overhead in the router implementation and savings in codepoints.

Performance-wise, within the constraints of this study, the performance tradeoffs observed between the queue-based technique for admission control suggested in CL and a simpler token-bucket-based excess rate measurement for admission control in SM do not appear to be a cause of substantial concern for cases when traffic aggregation is reasonably high at the bottleneck links as well as on a per ingress-egress pair basis. At very low ingress-egress aggregation (< 10 flows), SM is substantially more sensitive, and displays performance degradation caused by the synchronization effect. SM also exhibits the “beatdown” effect of the long-haul connections in the presence of multiple bottlenecks, and displays somewhat larger unfairness than CL. Having said that, one mitigating consideration in favor of the simpler mechanism is that in a typical DiffServ environment, the real-time traffic is expected to be served at a higher priority and/or the target admission rate is expected to be substantially below the speed at which the real-time queue is actually served. If these assumptions hold, then there is some margin of safety for an admission control algorithm, making the requirements for admission control more forgiving to bounded errors.

The termination mechanisms of SM and CL are both based on excess-rate metering and marking, so it may be inferred that their performance is similar. However, there is a subtle difference between the two mechanisms stemming from the fact that in SM, the bottleneck condition with respect to the termination-threshold is *not* directly conveyed through the markings. SM meters against the admission-threshold and infers the bottleneck condition with admission markings. The inference process is vulnerable to inaccuracies, such as non-uniformity in the marking distribution, and may result in over-termination compared to CL, especially when ingress-egress aggregation is low (< 50 flows) and in multi-bottleneck environments. While we believe that the extent of this over-termination is tolerable for practical purposes, it needs to be taken into account when considering the performance tradeoffs of the two mechanisms. As in the case of CL, the presence of loss does not affect the performance much; in addition, SM is more resilient to the negative impact of queuing, which makes it perform comparably to CL in settings with large queues.

CHAPTER 5

FLUID MODEL SIMULATION

Packet-base simulation has been widely used for performance evaluation. However running a packet-level simulator is very resource and time consuming, and does not scale well as the complexity of the subject network and the duration of the simulation increase. This limitation in the simulation effort will greatly constrain the depth and the breadth of the evaluation work. Consequently, the need for efficient simulation techniques is widely recognized. One method of speeding up the simulation is by using a model with higher level of abstraction. For instance, the abstracted model treats a flow as a continuous stream of fluid, instead of discrete packets. A simulation with such abstractions is called a ***fluid model***. Fluid models generally feature theoretical formulations and approximations, and offer superior efficiency compared to their packet counterparts.

Over the years, a large number of fluid models have been developed. From a structural point of view, some fluid models ([30][16][15][23][8]) treat each flow as a fluid stream, and the flow's rate adjustments due to network conditions (such as congestion) are tracked as events in the simulator; others ([3][39][37][22][20]) model flow traffic as a continuous amount of data which is shifted through the network at a fixed time step. Functionality-wise, some ([30][16][15][23][8]) are simply designed to model certain queuing disciplines; while others ([39][37][22][20][24][2]) with more complex capabilities such as throughput and delay approximation, are used to model systems with dynamic rate adjustment, such as TCP or admission control algorithms. While all existing works have demonstrated that their fluid models work accurately, most of the accuracy analysis are either performed on a simple topology (e.g. a single bottleneck) or focused on bottleneck behaviors only.

One of the contributions of our work is the development of several novel fluid models of different complexities and the investigation of their accuracy and applicability to various aspects of PCN termination and admission control. Aside from providing a tool to gain more insight in PCN behavior, we believe that some of our analysis is of a more general nature. Specifically, we believe that our results support a conjecture that while bottleneck behaviors are possible to predict with high accuracy using these models, per-aggregate behavior of admission control systems cannot be accurately predicted with any fluid model resulting in a bounded estimation error. The rest of this chapter is dedicated to the detailed description of the fluid models use to obtain the results of Chapter 4. We present the complete account of the model construction and assess its accuracy.

5.1 Fluid Model for PCN Admission Control

5.1.1 Model Description

Like our packet simulator, the admission fluid model is also an event-based simulator with the difference being it handles the events at an aggregate level rather than a packet level. Recall that in the PCN architecture, the aggregate 1) performs certain functions on an interval-basis, in particular, it collects various measurements of the previous interval and computes a ratio of the marked traffic over the total traffic, which we refer to as *interval-cle*. 2) It then smoothes the interval-cle as an exponential weighted average (*ewma-cle*). 3) It then uses ewma-cle to decide whether the aggregate will admit or not during the next interval by comparing it to cle-threshold. We refer to the time at which each aggregate perform the above functions as a ***decision-point***, and model it as an ***event***. In between the events, various aspects of the system are recorded as ***states***.

In a nut shell, the fluid simulation iterates through all the aggregates' decision-points as events, it processes the events by going through the above-described steps 1 to 3 based on the information stored in the states. This process continues until the desired length of the simulation is reached.

While above algorithm outline seems quite simple, the substantive core of the admission fluid model lies in step one, that is, given an aggregate, how the fluid model calculates its interval-cle. In fact the rest of this section is devoted to cover the implementation of this part, but first we need to introduce our concept of the ***state*** in more detail. The state is used to capture the dynamics of the system for a period of the time between two events, and it has a number of attributes to store information about the aggregates and the links. Some of these attributes are invariant throughout the state, such as the arrival-rate at the links (which is derived from the aggregates' admission decisions). Other attributes are subject to constant change, such as the load (number of flows) of the aggregates or on the links, the virtual-queue/token-bucket occupancy, the amount of marked/total traffic, etc. How to model these changing attributes and how to use them to compute an aggregate's interval-cle are the main focus of the admission fluid model, which we will present below in a step-by-step fashion.

Compute Arrival-Rate from Aggregates' Admission Decision

Given a state, an aggregate either decide to admit with a configured arrival-rate [1], or decide to block admission, which translate to an arrival-rate of zero. Since each aggregate's flow arrival is modeled as an independent Poisson process, the flow arrival on any bottleneck link is therefore also a Poisson process with an arrival-rate equal to the sum of the rates of the aggregates that traverse that bottleneck.

[1] The aggregate's configured arrival-rate is part of the input traffic matrix.

Compute Load from Arrival Rate

Next we will address the problem that given an arrival-rate and a time duration, how to capture the dynamics of the load (number of flows) on the bottleneck or aggregate. This is the crucial component for computing an aggregate's interval-cle. In what follows, we will introduce the theoretical background for translation between arrival-rate and load.

First, let's phrase the problem formally. We have flows arrive according to a Poisson process with rate λ. The flow duration is exponentially distributed with mean of $1/\mu$.[2] Given there are initially N_o flows in the system, we want to find the expected number of flows $N(t)$ in the system after time t. It turns out that this problem mirrors certain types of birth-death processes and can be solved using tools from queuing theory. First we break $N(t)$ into two independent components $N(t) = N_b(t) + N_d(t)$. The first component assumes a system starts with 0 flows and has an arrival rate of λ and $N_b(t)$ represents the expected number of flows in such a system after time t. This part can be modeled by a $M/M/\infty$ process. The second component is a pure death process; that is, it assumes the system starts with N_0 flows with no arrivals, and $N_d(t)$ is the expected number of flows left in the system after time t. Unlike most of the queuing theory applications that focus on the steady-state solution, we are interested in the transient-state analysis that is time dependent. Luckily both processes have an analytical (non-numerical) solution. For each process, we first compute its probability density function (pdf), in other words, the probability that the process has k flows at time t, denoted by $P^k(t)$, from which we can compute the expected number of flows at time t (The full derivation of $M/M/\infty$ process's pdf can be found in Appendix B, and pdf for the pure-death process is in [10])

$$\begin{aligned} P_b^k(t) &= \frac{1}{k!} \exp\left(-(1 - e^{-\mu t}) \frac{\lambda}{\mu}\right)\left((1 - e^{-\mu t}) \frac{\lambda}{\mu}\right)^k \\ P_d^k(t) &= \binom{N_o}{k} e^{-\mu t k}(1 - e^{-\mu t})^{N_o - k} \end{aligned} \tag{5.1}$$

[2] The mean of the flow duration is generally set to a fixed value such 1 or 2 minutes.

It's easy to see from Equation (5.1) that $P_b^k(t)$ has the shape of a Poisson distribution and $P_d^k(t)$ has the shape of a binomial distribution. We can then derive their expected values.

$$N_b(t) \;=\; (1 - e^{-\mu t})\,\frac{\lambda}{\mu}$$

$$N_d(t) \;=\; N_o\, e^{-\mu t}$$

$$\boxed{N(t) \;=\; (1 - e^{-\mu t})\,\frac{\lambda}{\mu} + N_o\, e^{-\mu t}} \qquad (5.2)$$

Given a state and the initial load (whether it be for aggregates or links), we can use Equation (5.2) to compute the load at any point of time within the state. We also use it to compute the end load of the state which is used to initialize the next state if necessary. Another situation that frequently arises requires finding out how long it takes for the load to change from N_1 to N_2, which requires a simple transformation of Equation (5.2), solving for t.

$$\boxed{t \;=\; -\frac{1}{\mu} * \ln\left(\frac{N_2 - \lambda/\mu}{N_1 - \lambda/\mu}\right)} \qquad (5.3)$$

Compute the Aggregate's Interval CLE from Load and Arrival Rate

An aggregate's interval-cle is simply the ratio of the aggregate's marked traffic over the total traffic given a measurement interval. To compute either value, we first backtrace through the list of states to identify the states that correspond to the interval of interest in time. Note this interval could span over several states in which case we will perform the calculation on each one and sum the results. Once the state and the (sub)interval of interest are located, all that is needed to compute the aggregate's total traffic is to utilize Equation (5.2), taking integral $N(t)$ with respect to t. The resulting formula is Equation (5.4). Here N_o represents the aggregate's load at the beginning of (sub)interval and R_t denote the total traffic.

$$\boxed{R_t \;=\; \int N(t)\,\mathrm{d}t \;=\; \frac{\lambda}{\mu} * t - (\frac{N_o}{\mu} - \frac{\lambda}{\mu^2}) * e^{-\mu t} + \frac{N_o}{\mu} - \frac{\lambda}{\mu^2}} \qquad (5.4)$$

The computation of marked traffic is much more involved. The question is how to model the dynamics of virtue-queue/token-bucket in the presence of a changing load, and in addition to that, how to deal with it if the aggregate traverses multiple bottlenecks. Given that we already know the aggregate's total traffic, our goal is to find the percentage of that total traffic that's marked which we refer to as *marking-percentage* or *marking-probability*, the product of the two will give us the marked traffic. Note that we cannot use the marking-percentage as interval-cle directly, because as we stated before the measurement interval might span several states with each one having different loads and marking-percentages. Hence the correct way is to compute the marked and total traffic for each state, sum them up, then divide the two to obtain the interval-cle. In what follows, we will illustrate our solution step by step. We start out by presenting how to compute the marking-percentage on the bottleneck if the load on the bottleneck is fixed. We then show how to carry out the computation with a changing load. Finally, we will show if an aggregate transverses multiple bottlenecks, how to obtain the aggregate's marking-percentage from the marking-percentages of all the bottlenecks it crosses.

If we have a fixed bottleneck load, then it's rather intuitive how to approximate the marking behavior with the fluid model. A virtual-queue will be filled at a rate equal to the *load*, and at the same time drained with the rate of the admission-threshold. If *load* is greater than admission-threshold, the *queue* will build up with a rate of (*load* − admission-threshold), once the *queue* grows over the vq-threshold, all traffic will be marked. If the *load* is smaller than admission-threshold, the *queue* will get drained with a rate of (admission-threshold − *load*), if the *queue* drops below vq-threshold, marking will stop. *Queue* cannot grow beyond vq-upper-limit. The function named *vq-marking-$fixload$* for this computation is presented in Algorithm (1). A token-bucket can be handled similarly, with the difference being that the bucket is filled with the rate of the admission-threshold and drained with the rate of *load*. Furthermore, when the bucket is empty, the traffic is marked with a rate of (*load* − admission-threshold). The function for the computation for the token-bucket named *tb-marking-fixload* is presented in Algorithm (2) (described below).

Algorithm 1: Function: *vq-marking-fixload*

```
input $<load, duration, queue>$

$total = load * duration$
if $load ==$ admission-threshold then
  if $queue >$ vq-threshold then
    $marked = load * duration$
  end if
else if $load >$ admission-threshold then
  $filltime = \frac{max(\text{vq-threshold} - queue,\ 0\,)}{load - \text{admission-threshold}}$
  $marked = max(duration - filltime,\ 0\,) * load$
  $queue = min((load - \text{admission-threshold}) * duration + queue, \text{vq-upper-limit})$
else
  $draintime = \frac{max(queue - \text{vq-threshold},\ 0\,)}{\text{admission-threshold} - load}$
  $marked = min(draintime, duration) * load$
  $queue = max(queue - (\text{admission-threshold} - load) * duration, \text{vq-upper-limit})$
end if

return $<marked, total, queue>$
```

The next step is to perform the above described computation in the presence of load change. If we know the starting and ending load[3] on the bottleneck for the duration of interest, then a reasonable approximation is to iterate from the starting load to the ending load with some step size (say ±1 flow per step). In each step, we first compute the time it takes to get to the next load step using Equation (5.3). This will be the step duration (*s-duration*) we pass in function *vq-marking-fixload* or *tb-marking-fixload* to compute the marked and total traffic, while keeping track of the changes of the virtual-queue/token-bucket. While we iterate through the changing loads, we sum up

[3] If the starting load is given, the ending load can be computed with Equation (5.2).

Algorithm 2: Function: *tb-marking-fixload*

```
input $< load, duration, queue >$

$total = load * duration$
if $load$ == admission-threshold then
    marked = 0;
else if $load$ > admission-threshold then
    $draintime = \frac{bucket}{load - \text{admission-threshold}}$
    $marked = max(duration - draintime,\ 0\ ) * (load - \text{admission-threshold})$
    $bucket = max(bucket - (load - \text{admission-threshold}) * duration, 0)$
else
    $marked = 0$
    $bucket = min(bucket + (\text{admission-threshold} - load) * duration, \text{tb-depth})$
end if

return $< marked, total, bucket >$
```

the marked and total traffic from each round of computation. At the end of this process, the ratio of the two gives us the marking-percentage on this bottleneck. The pseudocode of this approach for CL is presented in the algorithm below. (The code for SM is straightforward, we omit it for conciseness.)

Algorithm 3: Function: *cl-singlelink-marking-percentage*

```
input $< start\text{-}load, end\text{-}load, step, arrival\text{-}rate, queue >$

$marked = 0, total = 0$
for $load = start\text{-}load$ to $end\text{-}load$ do
    $s\text{-}duration \leftarrow$ compute with Equation (5.3): $N_1 = load$ and $N_2 = load + step$
    $(s\text{-}marked, s\text{-}total, n\text{-}queue) = vq\text{-}marking\text{-}fixload\,(load, s\text{-}duration, queue)$
    $marked \mathrel{+}= s\text{-}marked$
    $total \mathrel{+}= s\text{-}total$
    $queue = n\text{-}queue$
end for

return $marking\text{-}percentage = marked/total$
```

Now we know how to compute a marking-percentage on a single bottleneck, the next step is to turn the bottleneck's marking-percentage into an aggregate's marking-percentage. If the aggregate only goes through one bottleneck, then trivially the bottleneck's marking-percentage is same as the aggregate's. For the multi-bottleneck cases, we make an assumption that each bottleneck's marking behavior is independent, which allows us to compute each marking-percentage individually using the algorithms described above. Under this assumption, the value of our interest – the aggregate's marking-percentage which translates to the percentage of traffic that's being marked at at least one bottleneck – can then be viewed as the union of marking-probabilities of all the bottlenecks which the aggregate traverses. The computation of the probability of union is within the basic probability theory which we will not present here. We do however want to assess the validity of our assumption of independence. For CL, this assumption is perfectly valid since the virtual-queue algorithm meters and marks a packet regardless of whether or not the packet traverses (or has been marked at) previous bottlenecks. This resembles the independence of marking behavior at each bottleneck. SM however, excludes previous marked packets from its token-bucket metering, which in essence invalidates our assumption. While it's possible to implement a fluid simulator to model this particular aspect of token-bucket metering (like what we will do in the termination fluid model), we believe in SM admission, given that the marking-percentage is very small in magnitude, any inaccuracy in our assumption will not result in a large inaccuracy in the results.

Having described how to compute an aggregate's marking-percentage for a given state and duration within the state, we are finally ready to compute the aggregate's interval-cle which we already briefly outlined at the beginning for this subsection. Since the measurement interval, which is usually set to 100ms in our packet simulation, may span over several states, the fluid model will first identify the states (and each sub-interval within the state), then use the algorithms we described in this subsection to compute the total traffic, marking percentage, and hence the marked traffic for each state and duration. Finally, the marked and total traffic are summed up across the states and the ratio of the two gives the aggregate's interval-cle.

Summary To summarize, our admission fluid model is an event-based simulator that operates at the ingress-egress aggregate level. The simulator models each aggregate's admission decisions on an interval-basis. It uses a combination of analytical models and approximations to capture the load dynamics as well as the marking/metering behavior at the bottlenecks, which enables it to estimate a given aggregate's interval-cle, hence the admission decisions. We will verify the accuracy of this model in the next section.

5.1.2 Accuracy Analysis

In this section, we will assess the accuracy of our admission fluid model by running both a packet and a fluid simulator in conjunction with each other across a range of parameter settings and topologies. Note our evaluation goal in this section is not the performance of PCN algorithms, but rather the accuracy of the fluid model, hence when analyzing the result from difference metrics, our focus will not be the magnitude of the results itself, rather the magnitude of the difference between the two simulators.

Recall in Chapter 4 we used results from the fluid model in (Section 4.2.2 and in Section 4.6.2). These experiments were on a Single Link topology with wide range of parameter settings under both CL and SM algorithms. Here, we run ~250 from these experiments with both fluid and packet models. We present the same set of results as in Chapter 4 – average, expected max, expected min and length of oscillation period (see Section 4.6 for definition) which are metrics to describe the dynamics of the load. For each of the four metrics, we draw a scatter plot of the results of the fluid vs. the packet model (see Figure 5.1). For the first three graphs (average, max and min), the results are measured as percentage deviations from the admission-threshold. The results in the fourth graph (oscillation period) is measured in seconds.

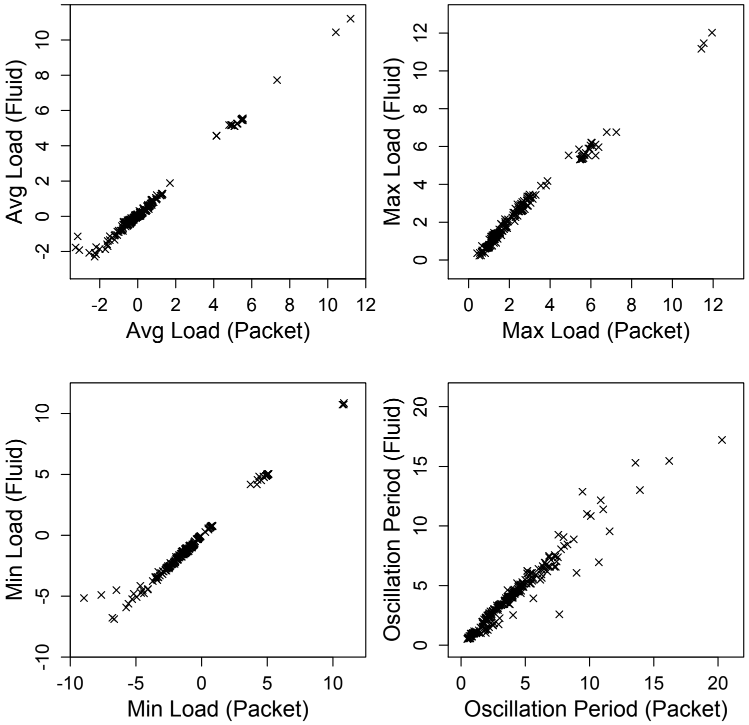

Figure 5.1: Accuracy of admission fluid model on single link topology

Most of data-points in the plots of Figure 5.1 lie on a 45 degree straight line, which indicates the results of fluid model are nearly identical to its packet-model counterpart. The few outliers in each plot correspond to the same set of experiments that have rather extreme parameter settings, such as an ewma-weight of 0.95. Overall, Figure 5.1 shows that on a Single Link topology, the fluid model accurately predicts the bottleneck behavior across a wide range of parameter settings.

Our next set of experiments is on multi-bottleneck topologies. Recall in Section 4.6.2, we ran large scale packet simulations on generalized PLT-5 (G-PLT-5) topologies for the purpose of comparing the admission performance of CL and SM in the multi-bottleneck scenario. Here we rerun exactly the same set of experiments with the fluid model. Figure 5.2 displays the distribution of the difference in bottleneck over-admission obtained by the fluid and packet models. It shows that for these ~8000 data-point from (including both CL and SM), the over-admission computed in the fluid model is mostly within 2% of that of the packet model, indicating again that the fluid model predicts the bottleneck behavior with great accuracy.

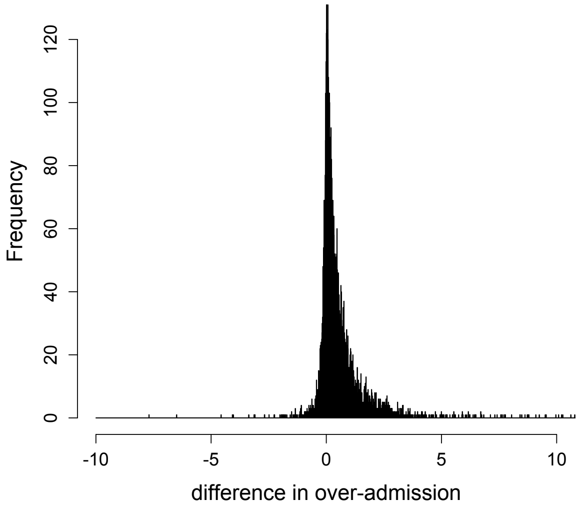

Figure 5.2: Accuracy of admission fluid model in predicting bottleneck behaviors under general settings

5.1.3 Model Applicability

We have shown that the fluid simulator works remarkably well when modeling the load behavior on the bottleneck; unfortunately, when it comes to predicting the load for individual aggregates, its accuracy greatly deceases. The main reason, as we have analyzed in Section 4.6.1, is that the individual aggregate's behavior is very sensitive to statistical variation in the traffic arrival, especially in the multi-bottleneck environment. The input to the simulation, both fluid and packet, is a traffic matrix which defines, among other things, the flow arrival rate of each aggregate. As described in our model, once the arrival-rates are fixed, the fluid simulator generates a ***deterministic*** outcome which can be viewed as an ***expected*** behavior of the system with the given traffic matrix. The packet simulator, on the other hand, models the aggregates' Poisson flow arrivals by actually generating a ***random*** inter-flow-arrival-time from an exponential distribution with a mean equal to the inverse of the arrival-rate. As we saw in Figure 4.16, experiments with different random seeds, which give different sequences of flow arrivals, can have drastically different IE-aggregate behaviors. This shows that our admission fluid simulator which is based on expected traffic arrival behavior will not be able to predict its corresponding packet simulation with bounded accuracy when it comes to the behavior of an individual aggregate, even though it works remarkably when modeling the bottleneck behavior.

The process of identifying the cause of the insufficiency of our fluid model sheds some light into a possible direction for improvement. The question is, whether there is a middle-ground between our aggregate-level expected-behavior-based fluid model and a packet simulator. The answer is yes, we could build a simulator that still models each flow as fluid (hence not getting into the packet-level details), but instead of computing the expected behavior with an analytical formula, we actually generate the random flow arrivals to capture the statistical variation in the flow arrivals. This version of the fluid model will no doubt run slower than the original, but it will still be orders of magnitude faster than any packet-based simulation. The question then becomes if we generate exactly the same flow arrival sequences and flow durations, will the new fluid model predict its packet counterpart accurately? We implemented our idea and tested it out. The results indicate that while the new fluid model shows great improvement in certain experiments, it still cannot accurately predict the aggregates' load behavior under all cases. The reason is as follows. Recall the core of the admission fluid model is the calculation of aggregate's interval-cle. If we just focus on any individual interval, we will discover that the revised fluid model gives a very accurate estimation of the interval-cle. However, the inaccuracy, no matter how small, has the potential to lead to a difference in admission decisions. For example, after smoothing, the ewma-cle from packet model is 0.3000001 and 0.2999999 for the fluid model. With a cle-threshold of 0.3, the seemingly neglectable error will cause a difference in admission decisions, and subsequently causes the difference in the load, the metering and the marking in the next interval, and these differences will likely cause a larger error when estimating the interval-cle for the next interval. The crucial point is to recognize that the admission control is a sequence of decisions, and as each decision is dependent on previous decisions, the inevitable inaccuracy in the each step of the approximation will ***accumulate***. While the model still predicts the bottleneck behavior very accurately, the effect caused by these accumulated errors will be more pronounced when predicting the aggregates' load, especially in a multi-bottleneck environment with some short-haul aggregates beating down the long-haul ones. This property leads us to conjecture the possibility that an admission fluid model is fundamentally incapable of predicting the complete per-aggregate outcome of its packet counterpart with bounded accuracy. This hypothesis remains to be investigated in depth in our future work.

5.2 Fluid Model for PCN Termination Control

5.2.1 Model Description

The termination fluid model shares many structural similarities with the admission one. It is also an event-based simulator that operates at the aggregate level. Just as in the case of admission, the simulation iterates through all aggregates' *decision-points* as event, recording various aspects of the system in *states*, and instead of making admission decisions, it decides the amount (if any) of flows to terminate by measuring the unmarked and total traffic.

Despite the similarities, there are certain differences in assumptions and modeling between the two fluid simulators. To understand the first difference, recall that in the case of admission, each state has a fixed arrival-rate, but a varying load. To the contrary, the termination model simplifies the computation by assuming that the load stays constant within a state; the only load change occurs if an event results in the termination of the flows. There are two observations justifying this simplification. First, in an environment where the network is overloaded to the point that termination is activated, it's natural to assume that any new flow arrived will not be admitted, which leads to an arrival-rate of zero. There are, however, still load changes caused by the natural departure of the current flows which brings to our second observation. The experiences from the packet simulations indicate that both CL and SM's termination algorithms have very fast reaction times, and usually bring the load below the termination-threshold within 200ms. This leads us to believe we can ignore the limited amount of flow departures and assume a constant load during the state without introducing too much inaccuracy. With a fixed load, the computation of marked (hence unmarked) traffic can be done exactly as described in Algorithm (2).

The second difference lies in the handling of metering/marking in the presence of multiple bottlenecks. Both CL and SM's termination algorithms use token-bucket metering as in the case of SM admission. Recall in the admission fluid model, we treated the metering and marking at each bottleneck independently, which is not entirely accurate for SM, because it does not meter (hence mark) any previously marked packets. We argued that due to the fact that the percentage of marked traffic in SM admission on any individual bottleneck is usually very small,[4] hence not removing this limited amount of marked traffic when metering at the downstream bottleneck will only be a minor inaccuracy. This is no longer true in the case of termination. Depending on the event-load, the percentage of marked traffic on any bottleneck can theoretically be anywhere between 0 to 100 percent and the inaccuracy will be too big to ignore. Therefore, we build the termination model ensuring that previously marked traffic is removed before metering and marking.[5]

There is no significant implementation difference between CL and SM for the termination fluid model. The only difference is as the algorithm suggests, CL measures unmarked traffic as a sustainable rate, while SM takes the additional step of multiplying the unmarked traffic with K (the ratio between the termination- and admission-thresholds). Naturally, CL and SM meter against different thresholds, one being the termination-threshold, the other being the admission-threshold. But this is not an implementation difference, just an input difference. Due to the time limitation, our termination fluid simulator does not model queuing or loss.

To summarize, our termination fluid model is an event-based simulator that operates at the ingress-egress aggregate level. The simulator models each aggregate's activity on an interval-base, computes the amount of unmarked traffic (hence the sustainable-rate) and total traffic (which is the sending-rate in the absence of loss), and finally computes the amount of flow to terminate (if any). The process continues until all bottlenecks' loads are below the termination-threshold. Compared to its admission counterpart, the

[4] The expected percentage of marked traffic is controlled by cle-threshold. In Section 4.2.2 we established for SM, there is a positive correlation between cle-threshold and the average load. Hence to avoid large over-admission, the cle-threshold for SM is usually set to a small value (< 0.05), which in turn implies a small percentage of marked traffic.

[5] The key is the order of the computation, starting from the most upstream bottleneck if possible, so that by the time we get to a more downstream bottleneck, we know the rate of already marked and unmarked traffic for each aggregate that traverses it. The computation is mostly implementation details rather than algorithmic insight, hence we omit the details for conciseness.

termination simulator made some adjustments in assumption and modeling that are more suited in a termination environment.

5.2.2 Accuracy Analysis

Like in the case of admission, we will assess the accuracy of our fluid model by running both a packet and a fluid simulator in conjunction with each other. Recall in Section 4.5.3, we used the termination fluid model to quantify the multi-bottleneck effect, in particular with respect to bottleneck utilization. Therefore, our goal in this section is to validate the termination fluid model in the multi-bottleneck environment.

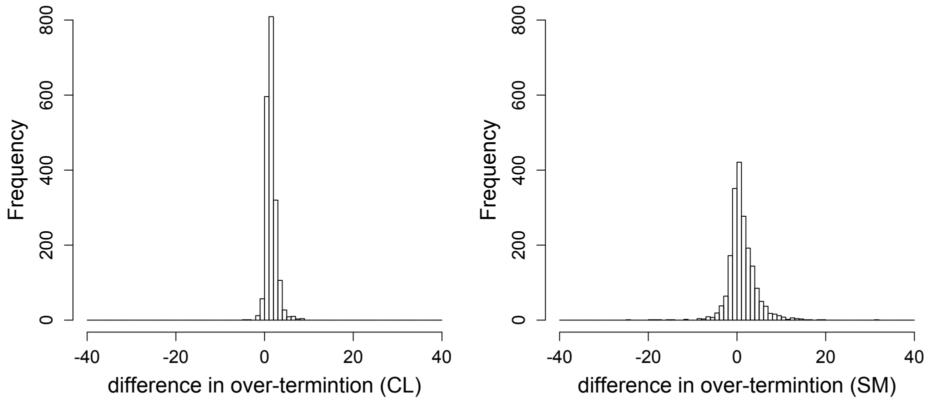

Figure 5.3: Accuracy of termination fluid model under general settings

We randomly select ~2000 from the ~45,000 traffic matrices used in the experiments in Section 4.5.3. These matrices are generated for Generalized PLT (G-PLT-5) topology. For each of matrix, we run a packet simulation for both CL and SM with all the same parameter settings as their fluid counterpart. The results metric, over-termination, is computed against a "reference" over-termination which was defined in Section 4.5.3. Again we emphasize, that the focus in this section is the difference in over-termination between the packet and the fluid model. Figure 5.3 displays the distribution of the differences in bottleneck over-termination obtained by the fluid and packet models. It shows that for these ~2000 experiments, the over-termination computed in fluid model is mostly within 5% of the packet model for CL, and 10% for SM.[6] The relatively small error in most experiments allows us to conjecture that the statistics of the fluid experiments adequately approximate the expected packet-level results in these experiments.

We conclude with an observation that despite the relative simplicity of the termination fluid model in comparison to the admission one, it does an adequate job of predicting the system outcome. This is in part due to the fact that termination control is largely a one step decision and is completely insensitive to the specific flow arrival pattern, contrary to the properties in the case of admission. The nature of the termination control make it resilient to errors in approximation and make it possible to use a simple aggregate-level, expected-traffic-based fluid model and still achieve bound accuracy.

[6] In fact, we computed the difference statistics with an increment of 500 experiments at a time, and found that the statistics converge to the results we presented here.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this chapter, we conclude the dissertation by 1) summarizing our contributions and 2) outlining several directions for future work.

6.1 Contribution

Pre-Congestion Notification (PCN) is an framework for flow admission and termination based on (pre-)congestion information in order to protect the quality of service of flows within a DiffServ domain. PCN encompasses two functions - an admission control that limits the amount of PCN traffic under "normal" operating condition, and a complementary termination control that terminates some admitted flows to allow the network to recover from sudden unexpected surges of PCN-traffic, and hence ensuring the QoS of the remaining flows. Given the PCN's overall framework and its objective, several questions need to be addressed. First, given its simple structure and ad-hoc measurement-based nature, is there a way of implementing the PCN framework that can adequately achieve its objective? Second, under the PCN framework, can differences in implementation significantly change the performance?

In an attempt to answer these two questions, we conducted a comprehensive performance evaluation of two proposals that implement PCN, CL-PHB and Single-Marking in this dissertation. Our main contributions are twofold: First, we showed that CL-PHB works reliably for both admission and termination control, even in the presence of packet loss, which in turn demonstrates that the PCN framework is a viable architecture to provide QoS to inelastic real-time traffic. Second, we show that there is a performance difference between different implementations. In particular, the benefits of Single-Marking such as saving one marking codepoint and simplifying the complexity of the interior router implementations come at a price. With respect to admission control, the performance of the admission algorithms of CL and SM appears to be comparable at a sufficient level of aggregation; at a low aggregation SM performs worse than CL. The termination algorithm of CL shows substantial performance degradation compared to CL. At the same time, our results imply that as long as sufficient levels of aggregation are present, and if a larger termination error is acceptable (thus implying the necessity of larger separation between admission and termination thresholds), SM may nevertheless be a viable practical solution.

The thorough PCN performance evaluation requires a large scale simulation effort. To that end, we developed two separate simulation methods - one is a discrete-event packet-based simulator, and the other is a novel fluid model that is based on both theoretical formulations and approximations. We used the two types of simulators in conjunction with each other and validated the accuracy of the fluid simulator against its packet counterpart. Our fluid simulator not only enabled us to run large scale PCN experiments, but also provided some insight on the applicability of fluid models in general. In particular, we found that for admission control, the fluid simulator works remarkably well in modeling the various properties of bottleneck behavior, and yet it fails to predict ingress-egress aggregates' outcomes with bounded accuracy. The insufficiency stems from the fact that admission control is a sequence of making decisions and the inaccuracy in each step of the approximation (no matter how small) is accumulated. This property leads us to hypothesize that an admission fluid model is fundamentally incapable of predicting the complete per-ie-aggregate outcome of its packet counterpart in the long run. In contrast, termination control, being largely a one-step decision process, was adequately predicted by our fluid model.

6.2 Future Work

There are several issues that we did not address in the dissertation. First, in the work we reported, we looked only at a single rerouting event, resulting in a sudden spike of traffic on a link. This scenario roughly corresponds to traffic behavior on a fast reroute. We have not seriously investigated the performance of the system when rerouted traffic arrives on the link over a longer period of time, or in several batches separated in time. For both of CL-PHB and Single-Marking, the termination control reaction time is on the order of RTT between ingress and ingress (plus the length of the measurement interval). Hence, we believe that if the inter-reroute-arrival time is larger than the reaction time, then each event can be treated independently. If multiple rerouting events occur within a single reaction time interval, we believe multiple events will just prolong the reaction time, and bear no consequence on the magnitude of the termination error. We did perform a small set of experiments validating this argument, but a more thorough validation remains to be done.

Second, we ignore the effect of Equal-Cost-Multi-Path routing (ECMP). ECMP is a common practice in today's network, mainly for the benefit of loading balancing. If there are multiple paths for a given ingress-egress pair, then when a certain action is carried out (e.g. stop admitting or terminating flows), the ingress/egress may not know exactly which path the action will impact. It is easy to construct cases where CL and SM will over- or under-terminate in presence of ECMP. It is therefore important to evaluate its impact across a wide range of networks.

We are also interested in investigating the effect of signaling delays and probing on both admission and termination algorithms. Finally, we would like to investigate in more depth the applicability of the admission fluid model. This will range from implementing a more detailed fluid model similar to the one described in [19], to designing a packet-level simulation to test if there exists any kind of packet-level variation that can cause substantial differences in system outcomes (which will directly prove our conjecture of the inapplicability of any fluid model).

APPENDIX A

PROOF OF SYNCHRONIZATION CONDITION

Notations:

P **:** packet (token) size

R **:** token-bucket-rate

$T_{X,Y}$ **:** the time interval between the arrival of packet X and Y

S_Z **:** the size of the token-bucket at the time packet Z arrives and before Z is processed. We sometime write $S_Z = N_Z * P + E_Z$, where N_Z is the number of available tokens, E_Z is the residual bucket size with $E_Z < P$

M_1 **:** the first marked packet

M_2 **:** the packet after M_1 in the same flow. Note, T_{M_2,M_1} = inter-packet-arrival = 20ms

M_3 **:** the packet after M_2 in the same flow. Note, T_{M_3,M_2} = inter-packet-arrival = 20ms

L_1 **:** the last packet which upon arrival counters a full token-bucket. In other words none of the packets arriving after L_1 will see a full token-bucket.

L_2 **:** the packet after L_1 in the same flow. Note, T_{L_2,L_1} = inter-packet-arrival = 20ms

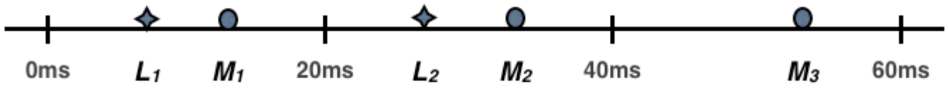

Figure A.1: The timeline of the packet arrivals

For simplicity, we assume:

1. The token-bucket never gets full again after packet L_1[1]
2. No flow joins or leaves during the time of study

[1] The intuition is that we segment the time into periods where the token-bucket gets full and it gets full again, and show our proof for just one of the periods. It's easy to see that the same argument applies to all periods.

Proof:

Given that we have an aggregation of CBR flows, and no flow joins or leaves during the time of study, it's easy to see that the packet arrival pattern in the interval $[M_1, M_2]$ is the same as in interval $[M_2, M_3]$ (and in the subsequent intervals). With this in mind, if we can show the token-bucket sizes at M_1 and M_2 are the same, $S_{M_1} = S_{M_2}$, then it follows trivially, that in all the subsequent intervals, the packets from the same flow will be marked exactly as the corresponding ones in interval $[M_1, M_2]$.

We can write $S_{M_1} = N_{M_1} * P + E_{M_1}$ and $S_{M_2} = N_{M_2} * P + E_{M_2}$. Recall, M_1 is the first marked packet, hence $N_{M_1} = 0$. To show $S_{M_1} = S_{M_2}$ is the same as showing: $N_{M_2} = 0$ and $E_{M_1} = E_{M_2}$. We claim if the bucket size at L_2 is an exact multiple of packet size, then it follows $S_{M_1} = S_{M_2}$.

Claim:

$$\text{if} \quad S_{L_2} = N_{L_2} * P + 0 \quad \text{and} \quad S_{M_1} = N_{M_1} * P + E_{M_1} \quad \text{where} \quad N_{M_1} = 0$$

$$\text{then} \quad S_{M_2} = N_{M_2} * P + E_{M_2} \quad \text{where} \quad N_{M_2} = 0 \quad \text{and} \quad E_{M_2} = E_{M_1}$$

Proof of the Claim: Recall, L_1 is the last packet that sees a full token-bucket, $S_{L_1} = N_{L_1} * P$ where N_{L_1} is the largest number of tokens the bucket can hold. If $N_{L_2} = N_{L_1}$, then the claim holds trivially. Otherwise it must be the case that $N_{L_2} < N_{L_1}$. Note, the bucket size at M_1 can be written as $S_{M_1} = S_{L_1} + T_{M_1,L_1} * R - A_1 * P$, where A_1 is the number of packets between L_1 and M_1 (This is because in our assumption the token-bucket never gets full after L_1). Similarly, $S_{M_2} = S_{L_2} + T_{M_2,L_2} * R - A_2 * P$ where A_2 is the number of unmarked packets between L_2 and M_2. It's easy to see that S_{M_1} and S_{M_2} always differ by an exact multiple of the packet size. It follows then that $E_{M_1} = E_{M_2}$. In addition, since the bucket size at L_1 is larger than L_2, given the same sequence of packet arrivals, it must follow that M_2 can never end up with more tokens in the bucket than M_1, that is, $N_{M_2} <= N_{M_1}$. And since $N_{M_1} = 0$, then $N_{M_2} = 0$. (End of Proof)

We have shown that if S_{L_2} is a multiple of the packet size, then we have synchronization. To satisfy this condition, note, $S_{L_2} = S_{L_1} + T_{L_2,L2_1} * R - A3 * P$, which means inter-packet-arrival (20ms) * *R* must be a multiple of a packet size.

APPENDIX B

TRANSIENT-STATE SOLUTION FOR M/M/∞ QUEUE

Consider a birth-death process, M/M/∞ in particular, starts from state 0 at time 0. We are interested in the probability that the process is in state k at time t, denoted by $P_k(t)$.

For general formula for birth-death process, we have:

$$\begin{aligned} \frac{\mathrm{d}P_k(t)}{\mathrm{d}t} &= -(\lambda_k + \mu_k)P_k(t) + \lambda_{k-1}P_{k-1}(t) + \mu_{k+1}P_{k+1}(t) && k \geqslant 1 \\ \frac{\mathrm{d}P_0(t)}{\mathrm{d}t} &= -\lambda_0 P_0(t) + \mu_1 P_1(t) && k = 0 \end{aligned}$$

For M/M/∞ queue, we have $\lambda_k = \lambda$ and $\mu_k = k\mu$, so the above equations become:

$$\begin{aligned} \frac{\mathrm{d}P_k(t)}{\mathrm{d}t} &= -(\lambda + k\mu)P_k(t) + \lambda P_{k-1}(t) + (k+1)\mu P_{k+1}(t) && k \geqslant 1 \quad \text{(B.1)} \\ \frac{\mathrm{d}P_0(t)}{\mathrm{d}t} &= -\lambda P_0(t) + \mu P_1(t) && k = 0 \quad \text{(B.2)} \end{aligned}$$

To solve for $P_k(t)$, we first apply a z-transform, let

$$P(z,t) \triangleq \sum_{k=0}^{\infty} P_k(t) \cdot z^k$$

Multiply Equation (B.1) by z^k and sum over the range of k, we have:

$$\sum_{k=1}^{\infty} \frac{\mathrm{d}P_k(t)}{\mathrm{d}t} z^k = \sum_{k=1}^{\infty} -(\lambda + k\mu)P_k(t)z^k + \sum_{k=1}^{\infty} \lambda P_{k-1}(t)z^k + \sum_{k=1}^{\infty} (k+1)\mu P_{k+1}(t)z^k \quad \text{(B.3)}$$

Taking the partial derivative of Equation (B.3) with respect to t gives:

$$\frac{\partial \sum_{k=1}^{\infty} P_k(t)z^k}{\partial t} = -\lambda \sum_{k=1}^{\infty} P_k(t)z^k - \mu \sum_{k=1}^{\infty} kP_k(t)z^k + \lambda \sum_{k=1}^{\infty} P_{k-1}(t)z^k + \mu \sum_{k=1}^{\infty} (k+1)P_{k+1}(t)z^k \quad \text{(B.4)}$$

Note, each component in Equation (B.4) with a summation can be written as follows:

$$\begin{aligned}
\sum_{k=1}^{\infty} P_k(t)z^k &= \sum_{k=0}^{\infty} P_k(t)z^k - P_0(t) = P(z,t) - P_0(t) \\
\sum_{k=1}^{\infty} kP_k(t)z^k &= z\sum_{k=1}^{\infty} kP_k(t)z^{k-1} = z\frac{\partial P(z,t)}{\partial z} \\
\sum_{k=1}^{\infty} P_{k-1}(t)z^k &= z\sum_{k=1}^{\infty} P_{k-1}(t)z^{k-1} = z\,P(z,t) \\
\sum_{k=1}^{\infty}(k+1)P_{k+1}(t)z^k &= \frac{\partial P(z,t)}{\partial z} - P_1(t)
\end{aligned}$$

So Equation (B.4) becomes:

$$\begin{aligned}
\frac{\partial[P(z,t) - P_0(t)]}{\partial t} &= -\lambda[P(z,t) - P_0(t)] - \mu z\frac{\partial P(z,t)}{\partial z} + \lambda zP(z,t) + \mu[\frac{\partial P(z,t)}{\partial z} - P_1(t)] \\
\frac{\partial P(z,t)}{\partial t} &= -\lambda(1-z)P(z,t) + \mu(1-z)\frac{\partial P(z,t)}{\partial z} + \lambda P_0(t) - \mu P_1(t) + \frac{\mathrm{d}P_0(t)}{\mathrm{d}t}
\end{aligned}$$

Using Equation (B.2), the last three terms of the above equation add to 0, the equation reduces to:

$$\frac{\partial P(z,t)}{\partial t} = -\lambda(1-z)P(z,t) + \mu(1-z)\frac{\partial P(z,t)}{\partial z} \tag{B.5}$$

The $P(z,t)$ that satisfies above partial differential equation in (B.5) should be in an exponential family. Suppose $P(z,t) = e^U$, where U is a function of z, t, then Equation (B.5) becomes:

$$\frac{\partial U}{\partial t} = -\lambda(1-z) + \mu(1-z)\frac{\partial U}{\partial z} \tag{B.6}$$

Apply the separation of variable method to Equation (B.6), let $U = f(z) \cdot g(t)$, we have:

$$\begin{aligned}
f(z)g'(t) &= -\lambda(1-z) + \mu(1-z)f'(z)g(t) \\
f(z)g''(t) &= \mu(1-z)f'(z)g'(t) \\
\frac{g''(t)}{g'(t)} &= \frac{f'(z)}{f(z)}\mu(1-z) = K
\end{aligned}$$

where K is some constant. It's easy to see that $g(t)$ must be an exponential function and $f(z)$ is a polynomial function. Let

$$g(t) = c_1 + c_2 e^{c_3 t} \text{ and } f(z) = c_4(1-z)^{c_5}$$

Plug $g(t)$ and $f(z)$ into Equation (B.6), to solve for the constants:

$$\begin{aligned} c_4(1-z)^{c_5} c_2 c_3 e^{c_3 t} &= -\lambda(1-z) + \mu(1-z)(c_1 + c_2 e^{c_3 t}) c_4 c_5 (1-z)^{c_5 - 1}(-1) \\ &= -\lambda(1-z) - \mu c_1 c_4 c_5 (1-z)^{c_5} - \mu c_2 e^{c_3 t} c_4 c_5 (1-z)^{c_5} \end{aligned}$$

Equating the terms, we get:

$$c_5 = 1 \text{ and } c_3 = -\mu \text{ and } c_1 c_4 = -\frac{\lambda}{\mu}$$

$$U = f(z) \cdot g(t) = (c_1 c_4 + c_2 c_4 e^{c_3 t})(1-z)^{c_5} = (-\frac{\lambda}{\mu} + c_2 c_4 e^{-\mu t})(1-z)$$

Z-transform property gives $P(0,t) = P_0(t)$ and the initial condition says the process is at state 0 when $t = 0$, which translates to $P_0(0) = 1$. Combine the two:

$$\begin{aligned} P(0,0) = 1 \ &\Rightarrow \ e^{U(0,0)} = 1 \\ &\Rightarrow \ (-\frac{\lambda}{\mu} + c_2 c_4 e^{-\mu t})(1-z)\Big|_{t=0,z=0} = 0 \\ &\Rightarrow \ c_2 c_4 = \frac{\lambda}{\mu} \end{aligned}$$

Thus, we have completely solved $P(z,t)$, plug in the solved constants, we get:

$$P(z,t) = \exp\left((1 - e^{-\mu t})(z-1)\frac{\lambda}{\mu}\right)$$

Now inverse the z-transform using series expansion to obtain $P_k(t)$

$$\begin{aligned} P_k(t) &= \frac{1}{k!}\frac{\partial^k P(z,t)}{\partial z^k}\Big|_{z=0} \\ &= \boxed{\frac{1}{k!}\exp\left(-(1 - e^{-\mu t})\frac{\lambda}{\mu}\right)\left((1 - e^{-\mu t})\frac{\lambda}{\mu}\right)^k} \end{aligned}$$

BIBLIOGRAPHY

[1] A. Charny, et al. Pre-congestion notification using single marking for admission and pre-emption. Internet-Draft, work in progress.

[2] Adrian S-W. Tam, Dah-Ming Chiu, John C. S. Lui, and Y. C. Tay . A case for tcp-friendly admission control. *IWQoS*, 2006.

[3] Anlu Yan and Wei-Bo Gong. Time-driven fluid simulation for high-speed networks. *IEEE Transactions on Information Theory*, 1999.

[4] Ayalvadi J. Ganesh, Peter B. Key, Damien Polis, and R. Srikant. Congestion notification and probing mechanisms for endpoint admission control. *IEEE/ACM Transactions on Networking (TON)*, 2006.

[5] B. Briscoe, et al. Pre-congestion notification marking. Internet-Draft, work in progress.

[6] B. Davie, B. Briscoe and J. Tay . Explicit congestion marking in mpls. RFC 5129.

[7] D. J. Songhurst, P. L. Eardley, B. Briscoe, C Di Cairano Gilfedder, and J. Tay. Guaranteed qos synthesis for admission control with shared capacity. BT Technical Report TR-CXR9-2006-001.

[8] David Nicol, Michael Goldsby, and Michael Johnson. Fluid-based simulation of communication networks using ssf. *European Simulation Symposium*, 1999.

[9] Edward W. Knightly and Jingyu Qiu. Measurement-based admission control with aggregate traffic envelopes. *IEEE/ACM Transactions on Networking (TON)*, 2001.

[10] Benjamin Epstein and Ishay Weissman. *Mathematical Models for Systems Reliability*. Chapman and Hall/CRC, 2008.

[11] John Evans and Clarence Filsfils. *Deploying IP and MPLS QoS for Multi-service Networks*. Morgan Kaufmann, 2007.

[12] F. Baker, C. Iturralde, F. Le Faucheur and B. Davie. Aggregation of rsvp for ipv4 and ipv6 reservations. RFC 3175.

[13] F. Kelly, P. Key, and S. Zachary. Distributed admission control. *IEEE Journal on Selected Areas in Communications*, 2000.

[14] G. Bianchi, F. Borgonovo, A. Capone, L. Fratta, and C. Petrioli. Endpoint admission control with delay variation measurements for qos in ip networks. *ACM SIGCOMM Computer Communication Review*, 2002.

[15] G. Kesidis , A. Singh, D. Cheung, and W. Kwok. Feasibility of fluid event-driven simulation for atm networks. *IEEE Globecom*, 1996.

[16] J . M. Pitts. Cell-rate modelling for accelerated simulation of atm at the burst level. *Communications, IEE Proceedings*, 1995.

[17] L. Breslau, S. Jamin, and S. Shenker. Measurement-based admission control: what is the research agenda? *Proc. IEEE IWQoS*, 1999.

[18] L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. *Proc. IEEE INFOCOM*, 2000.

[19] Lasse Jansen and Ivan Gojmerac and Michael Menth and Peter Reichl and Phuoc Tran-Gia. Discrete-time algorithms for ow-level simulation of data networks. *International Teletrafc Congress (ITC)*, 2007.

[20] Lasse Jansen, et al. An algorithmic framework for discrete-time flow-level simulation of data networks. *International TeletrafcCongress (ITC)*, 2007.

[21] Lee Breslau, Edward W. Knightly, Scott Shenker, Ion Stoica, and Hui Zhang. Endpoint. admission control; architectural issues and performance. *Proc. ACM SIGCOMM*, 2000.

[22] M. Ajmone Marsan, M. Garetto, P. Giaccone, E. Leonardi, E. Schiattarella, and A. Tarello. Using partial differential equations to model tcp mice and elephants in large ip networks. *IEEE Infocom*, 2004.

[23] M. Bahrands, et al. On rate-based simulation of communication networks. *Design, Analysis, and Simulation of Distributed Systems (DASD)*, 2003.

[24] M. Grossglauser and D.N.C. Tse . A framework for robust measurement-based admission control. *IEEE/ACM Transactions on Networking (TON)*, 1999.

[25] M. Karsten and J. Schmitt. Admission control based on packet marking and feedback signalling - mechanisms, implementation and experiments. KOM Technical Report 03/2002.

[26] M. Menth. Pcn-based resilient network admission control: The impact of a single bit. Technical Report.

[27] P. Eardley, J. Babiarz, K. Chan, A. Charny, R. Geib, G. Karagiannis, M. Menth, and T. Tsou. Pre-congestion notification architecture. Internet-Draft, work in progress.

[28] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture. RFC 1633.

[29] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp). RFC 2005.

[30] R. C. F. Tucker. Accurate method for analysis of a packet-speech multiplexer with limited delay. *IEEE Transactions on Communications, vol. 36*, 1988.

[31] R. J. Gibbens and F. P. Kelly . Resource pricing and the evolution of congestion control. *Automatica, vol. 35,*, 1999.

[32] R. Mortier, I. Pratt, C. Clark, and S. Crosby. Implicit admission control. *IEEE Journal on Selected Areas in Communications*, 2000.

[33] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475.

[34] S. Georgoulas, P. Trimintzios, and G. Pavlou. Joint measurement- and traffic descriptor-based admission control at real-time traffic aggregation points. *IEEE International Conference on Communications*, 2004.

[35] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking (TON)*, 1993.

[36] Tom Kelly. An ecn probe-based connection acceptance control. *ACM SIGCOMM Computer Communication Review*, 2001.

[37] Vishal Misra, Wei-bo Gong, and Don Towsley. Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. *ACM SIGCOMM*, 2000.

[38] 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. RFC 2998.

[39] Yong Liu, Francesco Lo Presti, Vishal Misra, Don Towsley, and Yu Gu. Fluid models and solutions for large-scale ip networks. *ACM SIGMETRICS*, 2003.