NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

Router-Based Algorithms for Improving Internet Quality of Service

A dissertation
submitted for the degree of
Doctor of Philosophy

by

Rade Stanojević

Supervisor: Robert Shorten

Hamilton Institute
National University of Ireland, Maynooth

September 2007

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Abstract

We begin this thesis by generalizing some results related to a recently proposed positive system model of TCP congestion control algorithms. Then, motivated by a mean field analysis of the positive system model, a novel, stateless, queue management scheme is designed: Multi-Level Comparisons with index l (MLC(l)). In the limit, MLC(l) enforces max-min fairness in a network of TCP flows. We go further, showing that counting past drops at a congested link provides sufficient information to enforce max-min fairness among long-lived flows and to reduce the flow completion times of short-lived flows. Analytical models are presented, and the accuracy of predictions are validated by packet level *ns2* simulations.

We then move our attention to efficient measurement and monitoring techniques. A small active counter architecture is presented that addresses the problem of accurate approximation of statistics counter values at very-high speeds that can be both updated and estimated on a per-packet basis. These algorithms are necessary in the design of router-based flow control algorithms since on-chip Static RAM (SRAM) currently is a scarce resource, and being economical with its usage is an important task. A highly scalable method for heavy-hitter identification that uses our small active counters architecture is developed based on heuristic argument. Its performance is compared to several state-of-the-art algorithms and shown to out-perform them.

In the last part of the thesis we discuss the delay-utilization tradeoff in the congested Internet links. While several groups of authors have recently analyzed this tradeoff, the lack of realistic assumption in their models and the extreme complexity in estimation of model parameters, reduces their applicability at real Internet links. We propose an adaptive scheme that regulates the available queue space to keep utilization at desired, high, level. As a consequence, in large-number-of-users regimes, sacrificing 1-2% of bandwidth can result in queueing delays that are an order of magnitude smaller than in the standard BDP-buffering case. We go further and introduce an optimization framework for describing the problem of interest and propose an online algorithm for solving it.

CHAPTER 2

Introduction

The Internet is a large distributed network that has a huge impact on the modern world. A major technical step that allowed the building of a variety of services in the Internet was the development of the reliable, congestion-aware protocol for data delivery: the Transmission Control Protocol (TCP). The current implementation of TCP is based on the AIMD paradigm proposed in [49]. Since then, a number of enhancements have been proposed to improve network behavior in congested environments. These enhancements include Active Queue Management (AQM), Differentiated Services architecture (DiffServ), Explicit Congestion Notification (ECN), loss based and delay based TCP variants, and many others.

Router-based mechanisms for congestion control that do not require explicit cooperation with end-to-end users are particularly attractive since they potentially allow a diverse set of end-to-end congestion control policies to coexist in the network. An important requirement that these router-based congestion control schemes must posses is scalability. Line speeds in the Internet have increased at a rate that is much higher than that of hardware components (such as packet buffer, network processor, RAM, etc). Therefore, a scalable router-based congestion control algorithm should utilize existing hardware devices in a manner that would enable operation at very high line speeds. In the current Internet most very high-speed links are rarely congested, but with ever increasing demands for bandwidth, we may not enjoy an over-provisioned best-effort Internet indefinitely. However, at times of congestion, many existing router-based congestion control schemes that were designed for low line speeds would become infeasible, and some efficient and computationally-lighter algorithms would be needed. In this thesis several algorithms are designed to address this scenario by improving network performance at very high speeds.

The main issues we are concerned with are fairness, Flow Completion Time (FCT) and in the latter part of this thesis the interaction between utilization and queueing delays. The early algorithms for enforcing fairness and small FCT usually require state information that is significantly beyond the

abilities of the high-speed routers, which are built with very limited high-speed memory. In this thesis we show that a very small amount of information is enough for enforcing fairness and small FCT by designing the appropriate algorithms.

The interaction between queueing delays and link utilization has been an important research problem in the networking community. Although many models exist in the literature that attempt to capture this relationship, they are not directly applicable to real Internet links because of their complexity and the lack of realistic assumptions on which the models are derived. In the last part of this thesis we propose a measurement approach to the problem of interest rather than a modelling approach. The proposed algorithms for regulating delay-utilization tradeoff can operate at arbitrary line speeds, do not require expensive estimation techniques and are robust to the nature of traffic patterns.

2.1 Structure and Contributions

In Chapter 3 we review a recently proposed positive system model [99] of a network of TCP flows. We relax the assumptions made in [99] and characterize the asymptotic behavior of throughput of AIMD users. The main tool for obtaining these results is nonnegative matrix theory. Using it we relate the random matrix model to standard fixed-point theory.

A mean field analysis of random matrix model motivated us to consider a class of AQM algorithms, that we call Power-Drop (PD) schemes. In Chapter 4 we will present a particular implementation of PD scheme called Multi-Level Comparisons with index l (MLC(l)). A mean field analysis of the random matrix model gave us some intuition on resource allocation established by PD schemes. However, a flow-level Markov chain model is proposed to formalize the findings of the matrix-model based mean-field analysis. The main feature of the scheme is the fact that it can enforce max-min fairness on arbitrary network of TCP flows without any state information (other than packets that are already in the queue).

In Chapter 5 we develop a computationally light and memory efficient scheme for enforcing fairness called Markov Active Yield (MAY). The design of MAY is based on the following paradigm: *Drop proportionally to amount of past drops.* Surprisingly, for an arbitrary set of elastic or inelastic users, the resource allocation is max-min fair in a network of MAY queues. Keeping state only for dropped packets (instead of arrived packets) can reduce computational/memory requirements by order(s) of magnitude compared to existing fair-scheduling schemes.

Chapter 6 contains two randomized algorithms for efficient statistics-counters architecture that maintains counters that can be both estimated and updated at a per-packet basis. In Chapter 7 we propose a scalable algorithm called Real-time Heavy-hitter Estimator (RHE) for heavy-hitter identification. A small memory overhead, and only a few CPU cycles per packet, allow this algorithm to scale with line speeds of up to $40Gbps$. RHE exhibits superior performance compared to several state-of-the-art per-flow traffic measurement tools: Multistage filters [28], CATE [44] and Landmark

LRU [13]. Using RHE we design a FIFO queueing scheme called Heavy-hitter Hold (HH) that enforces max-min fair resource allocation by simple identification of the heaviest flows and dropping packets accordingly. The FIFO nature of HH is essential for very high speed (optical) routers where the usual scheduling algorithms are not feasible since in the optical case there is no equivalent to electronic buffering memory which is necessary for the implementation of fair schedulers.

The final problem of interest is the delay-utilization tradeoff. While several groups of authors have recently analyzed this tradeoff, the lack of realistic assumptions in their models and extreme complexity in estimation of model parameters reduces their applicability for real Internet links. In Chapter 8 we propose an adaptive scheme that regulates the available queue space to keep utilization at a desired, high level. As a consequence, in a large-number-of-users regime, sacrificing 1-2% of bandwidth can result in queueing delays an order of magnitude smaller than in standard BDP-buffering case. In Chapter 9 we go further and introduce an optimization framework to settle the problem of interest. By defining the relative importance between utilization and queueing delay, one can formulate the optimization problem of choosing the queueing-scheme parameter (available buffer space, drop probability, virtual queue capacity, etc.) that maximizes overall benefit.

The papers contributing to chapter 3 are listed below.

- A. Leizarowitz, R. Stanojević, R. Shorten, "Tools for the analysis and design of communication networks with Markovian dynamics". Proceedings of IEE, Control Theory and Applications, vol. 153(5), pp. 506-519, 2006.
- F. Wirth, R. Stanojević, R. Shorten, D. Leith. "Stochastic equilibria of AIMD communication networks". SIAM Journal on Matrix Analysis and Applications, vol. 28(3), pp. 703-723, 2006.

The paper contributing to chapter 4 is:

- R. Stanojević, R. Shorten. "Beyond CHOKe: Stateless fair queueing". Proceedings of NET-COOP 2007, Avignon, France, LNCS, Springer, vol.4465.

The paper contributing to chapter 5 is:

- R. Stanojević, R. Shorten. "Drop counters are enough". Proceedings of IEEE IWQoS, Chicago, USA, 2007.

The paper contributing to chapter 6 is:

- R. Stanojević. "Small active counters". Proceedings of IEEE Infocom, Anchorage, USA, 2007.

The following two papers are related to the material presented in Chapter 8. The first one develops the Active DropTail (ADT) algorithm while the second one uses ADT for performing the measurement study of the router buffers.

- R. Stanojević, R. Shorten, C. Kellet. "Adaptive tuning of Drop-Tail buffers for reducing queueing delays". IEEE Communications Letters, vol. 10(7), July, 2006.

- G. Vu-Brugier, R. Stanojević, D. Leith, R. Shorten. "A critique of recently proposed Buffer-Sizing strategies". ACM Computer Communications Review, vol. 37(1), January 2007.

The paper contributing to chapter 9 is:

- R. Stanojević, R. Shorten. "How expensive is link utilization?". Proceedings of NET-COOP 2007, Avignon, France, LNCS, Springer, vol.4465.

Part I

Matrix model

CHAPTER 3

Matrix model

Abstract - *In this chapter we review the random matrix model of the AIMD congestion control algorithm proposed in [99]. Using the nonnegative structure of the model and powerful machinery of Markov chain theory we extend the model from [99] in several directions.*

3.1 Positive system model of AIMD congestion control algorithms

Various types of models for AIMD networks have been developed by several authors, see for example [102] or [59] and the references therein for an overview of this work. We base our discussion on a recently developed random matrix model of AIMD dynamics that was first presented in [99]. This model uses a set of stochastic matrices to characterize the behaviour of a network of AIMD flows that compete for bandwidth via a single bottleneck router (as depicted in figure 3.1).

While other similar random matrix models have been proposed in the literature [8, 9], the model proposed in [99] has several attractive features. In particular, the authors use sets of nonnegative column stochastic matrices to model the evolution of communication networks and results from Frobenius-Perron theory to characterize the stochastic properties of such networks.

Suppose that the network under consideration has n flows, all of them operating an Additive Increase Multiplicative Decrease (AIMD) congestion control algorithm, competing for bandwidth over a bottleneck link which has a drop-tail queue. Then the current state of the network at times when a packet is dropped at the bottleneck router (referred to as the k'th congestion event) is given by the throughput of each network source at this time. We describe the network state at the k-th congestion event by an n-dimensional vector $W(k) = \{w_i(k)\}_{i=1}^n$ where $w_i(k)$ is the i-th component of $W(k)$, which is equal to the *throughput* of the i'th source when this source is informed of network congestion.

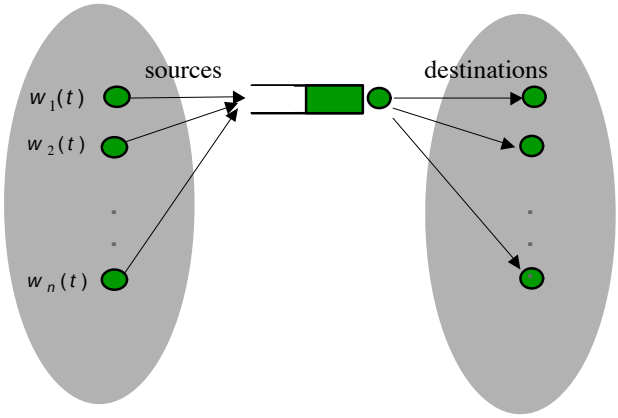

Figure 3.1: Network with single bottleneck router.

It has been shown in [99] that the sequence $\{W(k)\}_{k=0}^{\infty}$ satisfies:

$$W(k+1) = A(k)W(k), \tag{3.1}$$

where $W(k) = [w_1(k), \cdots, w_n(k)]^T$, and

$$A(k) = \begin{bmatrix} \beta_1(k) & 0 & \cdots & 0 \\ 0 & \beta_2(k) & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n(k) \end{bmatrix} + \frac{1}{\sum_{j=1}^{n} \alpha_j \gamma_j} \begin{bmatrix} \alpha_1\gamma_1 \\ \alpha_2\gamma_2 \\ \cdots \\ \alpha_n\gamma_n \end{bmatrix} \begin{bmatrix} 1-\beta_1(k), & \cdots & , 1-\beta_n(k) \end{bmatrix}. \tag{3.2}$$

For every $j \in \{1, 2, \ldots, n\}$, the constant $\alpha_j > 0$ in (3.2) is the Additive Increase parameter and $\gamma_j > 0$ is the constant $1/RTT_j^2$. Here RTT_j is the round-trip time for a packet from the j-th flow just before congestion, and either $\beta_j(k) = 1$, which holds if the j-th flow didn't lose any packet during the k-th congestion event, or $\beta_j(k)$ is equal to the Multiplicative Decrease parameter $\beta_j^0 \in (0, 1)$ if the j-th flow did lose some packet(s) in the k-th congestion event.

Comment 3.1 *We exclude the possibility that $\beta_1(k) = \beta_2(k) = \cdots = \beta_n(k) = 1$, since there is no congestion event without losing at least one packet.*

We denote by $\mathcal{M}$ the set of the possible values of the matrices $A(k)$, so that

$$\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$$

for some $m \leq 2^n - 1$ and for all k. Then $A(k) \in \mathcal{M}$ for every $k \geq 0$, and we remark that strict inequality $m < 2^n - 1$ may hold; namely that in the model which we consider, certain configurations of packets' loss cannot practically occur.

Let $I(k) = \{j : \ \beta_j(k) = \beta_j^0\}$ be the set of labels of flows which have experienced a loss of a packet during the k-th congestion event. Note that for each k the matrix $A(k)$ has a strictly positive j-th column if and only if $j \in I(k)$, and that for $j \notin I(k)$, the j-th column of $A(k)$ is equal to e_j, the j-th column of the identity $n \times n$ matrix I_n. We denote by Σ the $(n-1)$-dimensional simplex of all the n-dimensional stochastic vectors. Recall that a vector $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ is stochastic if each one of its coordinates v_i is nonnegative and $v_1 + \cdots + v_n = 1$. It turns out that the matrices M_i $(1 \leq i \leq m)$ which compose $\mathcal{M}$, are nonnegative and column-stochastic [11]. Therefore $M_i(\Sigma) \subset \Sigma$ holds for every $1 \leq i \leq m$. By normalizing $W(0)$ to belong to Σ we may therefore assume, with no loss of generality, that $W(k) \in \Sigma$ for every $k \geq 0$.

For networks with routers employing a drop-tail queueing discipline, it is often assumed in the networking community that congestion events may in some circumstances be modelled as sequences of independent events [2, 8, 91, 99]. In terms of the model described above, this means that for networks with a single bottleneck link and with a drop-tail queue the following holds:

Assumption (i) $\{A(k)\}_{k \in N}$ is a sequence of independent and identically distributed (i.i.d) random variables $A(k)$, and for every $j \in \{1, 2, \ldots, n\}$ the probability that the j-th flow detects a drop in each congestion event is positive.

As we have already mentioned, in designing rules which determine how the network will react to congestion, one can typically have two approaches. The first is the design of flow-based congestion control algorithms and the second is the design of a queueing discipline. Here we concentrate on the latter and propose two general queueing disciplines which we characterize by computing the stationary statistics for the vector $W(k)$ for each of these cases. We first consider queuing discipline with the property that packets are dropped from the queue in such a manner that the following assumption is valid:

Assumption (ii) $\{A(k)\}_{k \in N}$ is a *stationary* Markov chain on the finite set of matrices $\mathcal{M}$ with transition matrix $P \in \mathbb{R}^{m \times m}$. Moreover we assume that for each $j \in \{1, 2, \ldots, n\}$ there exists a matrix $M \in \mathcal{M}$ with positive j-th column.

We note that as in the i.i.d. case, we must have the latter assumption in order to ensure that each flow will see a drop at some point. We also remark that stationarity is assumed to avoid technical difficulties and it is not essential. An additional assumption in Sections 3.2 and 3.3 (which is relaxed in 3.4) is that the transition matrix P has strictly positive entries. Theorems 3.2 and 3.4 give the asymptotic values of $E(W(k))$ and $E[(W(k))(W(k))^T]$ in the limit where k tends to infinity, which we denote V^* and D^* respectively. Although we do not have explicit formulas for V^* and D^*, Theorems 3.1 and 3.3 provide iterative algorithms for computing them in a geometric convergence rate. In section 3.4 we extend the results of the previous two sections to the case where the matrix P is merely primitive and its entries are not necessarily strictly positive.

The second queueing discipline we propose here is the following: the probability that a certain set of flows will detect a drop during the k-th congestion event depends only on the vector $W(k)$.

Formally we assume that the router drops packets from the queue when it is full in such fashion that the following is true:

Assumption (iii) $\{W(k)\}_{k\in N}$ is a stochastic process in the set of stochastic vectors Σ, which has the following property:

For every $i \in \{1, 2, \ldots, m\}$ and $w \in \Sigma$, the conditional probability of $A(k)$ given $W(k)$ is expressed by

$$P[A(k) = M_i |\ W(k) = w] = p_i(w)$$

for some positive Dini-continuous functions $p_i : \Sigma \to \mathbb{R}^+$ which satisfy $\sum_{i=1}^m p_i(w) = 1$ for all $w \in \Sigma$. Again, for each $i \in \{1, 2, \ldots, n\}$ we require the existence of a matrix $M \in \mathcal{M}$ with positive i-th column.

In view of the relation $W(k+1) = A(k)W(k)$, Assumption (iii) implies that the distribution of $W(k+1)$ is completely determined by the distribution of $W(k)$. Section 3.5 is devoted to studying the behavior of $W(k)$ under Assumption (iii). It turns out that the study of the model under Assumption (iii) can be reduced to its study under Assumption (ii). This enables us to establish the analogous results concerning the asymptotic behavior of $E(W(k))$ and $Var(W(k))$ for this case. In particular, the latter can be computed by iterative methods producing schemes which converge at a geometric rate.

3.2 The asymptotic expectation of $W(N)$

In this section we compute the equilibrium expected value of the window size variable $W(N)$ under Assumption (ii), and supposing that the transition probabilities P_{ij} are positive:

$$P_{ij} > 0 \text{ for every } 1 \le i, j \le m. \tag{3.3}$$

Denoting by $\rho = (\rho_1, ..., \rho_m)$ the unique equilibrium distribution corresponding to P, we associate with P_{ij} the *backward transition probabilities* matrix $\tilde{P}$ (see [81], Chapter 1.9) given by

$$\tilde{P}_{ij} = \frac{\rho_i}{\rho_j} P_{ij} =$$
$$\frac{P[A(k-1) = M_i]}{P[A(k) = M_j]} P[A(k) = M_j | A(k-1) = M_i] =$$
$$= P[A(k-1) = M_i | A(k) = M_j]. \tag{3.4}$$

We interpret $\tilde{P}_{ij}$ as the conditional probability that the system occupied the state M_i at the previous instant of time given that it is presently at state M_j, for the *stationary* Markov chain $\{A_k\}$.

Let $\Phi : (\mathbb{R}^n)^m \to (\mathbb{R}^n)^m$ be the linear mapping given by:

$$\Phi(V) = \left(\sum_{i=1}^m \tilde{P}_{i1} M_i V_i, \ldots, \sum_{i=1}^m \tilde{P}_{im} M_i V_i \right) \tag{3.5}$$

where $V = (V_1, ..., V_m)$, $V_i \in \mathbb{R}^n$ and $M_i \in \mathcal{M}$ for every $1 \le i \le m$. We have the following result:

Proposition 3.1 *For an arbitrary $W(0) = s \in \Sigma$ and all $i = 1, 2, \ldots, m$, the following limits exist:*

$$V_i = \lim_{k \to \infty} E[W(k)|A(k) = M_i],\ i = 1, 2, ..., m. \tag{3.6}$$

Moreover, the vector $V = (V_1, ..., V_m) \in \Sigma^m$ whose components are defined in (3.6) *satisfies the fixed point equation*

$$V = \Phi(V). \tag{3.7}$$

Proof. The proof will be given after establishing Theorem 3.1.

Let $\mathcal{S}$ be the subspace of $\mathbb{R}^n$ defined by:

$$\mathcal{S} = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 0 \right\}.$$

It turns out that Φ has the following property:

Proposition 3.2 *The mapping Φ is linear from Σ^m into itself, and from $\mathcal{S}^m$ into itself.*

Proof: Both claims follow from the facts that if M is a column stochastic $n \times n$ matrix, then for every $x \in \mathbb{R}^n$ we have

$$\sum_{i=1}^{n} (Mx)_i = \sum_{i=1}^{n} x_i,$$

and that each M_i is a column stochastic matrix. ∎

We know that any matrix in $\mathcal{M}$ can be written in the form

$$M = \text{diag}(\beta_1, \beta_2, ..., \beta_n) + (\delta_1, \ldots, \delta_n)^T((1 - \beta_1), \ldots, (1 - \beta_n))$$

where $0 < \beta_k \leq 1$ for every $1 \leq k \leq n$ and not all of them are equal to 1. We denote $\beta = (\beta_1, ..., \beta_n)^T$, $\delta = (\delta_1, ..., \delta_n)^T$, and δ is a stochastic vector with positive entries. If the first q entries of β are equal to 1, namely $\beta_1 = \beta_2 = \cdots = \beta_q = 1$ for some $q < n$, and the last $n - q = r > 0$ entries are all smaller than 1, then our matrix M has the following form:

$$M = \begin{pmatrix} I_q & M' \\ 0 & M'' \end{pmatrix}. \tag{3.8}$$

The matrix I_q in (3.8) is the $q \times q$ identity matrix, all the entries in the last r columns are positive, and the sum of the entries in each of these columns is equal to 1.

The following result is the main technical tool that we employ in studying properties of Φ. We denote by $\|\cdot\|_1$ the L_1 norm of vectors in $\mathbb{R}^n$. We will prove it as Corollary 3.2 after having established Lemma 3.3. A direct proof can be found in [118].

Lemma 3.1 *Let $M \in \mathcal{M}$. Then for every $x \in \mathcal{S}$ we have*

$$Mx \neq x \Rightarrow \|Mx\|_1 < \|x\|_1. \tag{3.9}$$

Proof: See the proof of Corollary 3.2. ■

Comment 3.2 *The property which is established in Lemma* 3.1 *is referred to in the literature as the* paracontracting *property; see [45] chapter* 8 *or [26]. We have thus showed that the matrices M_i are paracontractive in $\mathcal{S}$ in L_1 norm. We will again use the notion of paracontractivity in Section 3.5.*

Lemma 3.2 *Suppose that $M \in \mathcal{M}$ is such that the columns which contain zeros are indexed by $i_1, i_2, \ldots, i_q$, and let $x \in \mathcal{S}$ be such that $Mx = x$. Then x belongs to the subspace spanned by the basic vectors $e_{i_1}, e_{i_2}, \ldots, e_{i_q}$.*

Proof: We suppose without loss of generality that the first q columns of M contain zeros and the last $r = n - q$ columns are positive, i.e. that M has the form given by (3.8). We will establish that the last r coordinates of x are equal to 0. If $r = 0$ then there is nothing to prove. If $r = n$ then M is a stochastic matrix with strictly positive entries, and therefore it is a contraction on $\mathcal{S}$, implying that $x = 0$.

We now suppose that $0 < r < n$, and let P be the $r \times r$ submatrix of M which is defined by the last r rows and last r columns. If we denote by x' the r-dimensional vector composed of the last r coordinates of x, then $Px' = x'$. But the sum of each column of P is smaller than 1 and P is nonnegative, hence $\|Px'\|_1 < \|x'\|_1$ whenever $x' \neq 0$, implying that x' must vanish. This concludes the proof of the lemma. ■

We define on $(\mathbb{R}^n)^m$ the norm

$$\|V\| = \|(V_1, \ldots, V_m)\| = \max_{1 \leq i \leq m} (\|V_i\|_1),$$

and consider the subspace $\mathcal{S}^m$ and subset Σ^m of $(\mathbb{R}^n)^m$ endowed with this norm. The next result establishes that Φ^2 is a contraction on the metric space Σ^m as well as on the normed space $\mathcal{S}^m$.

Proposition 3.3 *Let Φ be the mapping given by* (3.5)*. We assume that* (3.3) *holds, so that in view of* (3.4)*, all the backward probabilities $\tilde{P}_{ij}$ are positive as well. Then there exists a constant $\theta < 1$ such that*

$$\|\Phi^2(U) - \Phi^2(V)\| \leq \theta \|U - V\| \tag{3.10}$$

holds for all $U, V \in \Sigma$.

Proof: We will establish that for every pair $U \neq V$ in Σ^m, the inequality $\|\Phi^2(U) - \Phi^2(V)\| < \|U - V\|$ holds. This will imply the assertion of the proposition in view of the compactness of Σ^m.

Thus let $U = (U_1, \ldots, U_m)$ and $V = (V_1, \ldots, V_m)$ be any two different elements belonging to Σ^m.

We have

$$\|\Phi(U)-\Phi(V)\| = \max_j \|\sum_{i=1}^m \tilde{P}_{ij} M_i(U_i - V_i)\|_1 \tag{3.11}$$

$$\leq \max_j \sum_{i=1}^m \tilde{P}_{ij} \|M_i(U_i - V_i)\|_1 \tag{3.12}$$

$$\leq \max_j \sum_{i=1}^m \tilde{P}_{ij} \|U_i - V_i\|_1 \tag{3.13}$$

$$\leq \max_j \sum_{i=1}^m \tilde{P}_{ij} \|U - V\| = \|U - V\|. \tag{3.14}$$

We will next check under which conditions equality $\|\Phi^2(U) - \Phi^2(V)\| = \|U - V\|$ can hold. We thus assume that $U \neq V$ are such that $\|\Phi(\Phi(U)) - \Phi(\Phi(V))\| = \|U - V\|$. It follows from

$$\|U - V\| = \|\Phi(\Phi(U)) - \Phi(\Phi(V))\| \leq \|\Phi(U) - \Phi(V)\| \leq \|U - V\|$$

that $\|\Phi(U) - \Phi(V)\| = \|U - V\|$. Thus in this situation all the inequalities in (3.11)-(3.14) are actually equalities.

We now denote $W = U - V \in \mathcal{S}^m$ and we note that in view of (3.14), for some j,

$$\sum_{i=1}^m \tilde{P}_{ij} \|W_i\|_1 = \max_i(\|W_i\|_1) = \|W\|. \tag{3.15}$$

Since we suppose that all $\tilde{P}_{ij}$ are positive, (3.15) implies

$$\|W_i\|_1 = \|W\| \text{ for every } 1 \leq i \leq m. \tag{3.16}$$

It then follows from (3.13) that

$$\max_j \sum_{i=1}^m \tilde{P}_{ij} \|M_i(W_i)\|_1 = \max_j \sum_{i=1}^m \tilde{P}_{ij} \|W_i\|_1 = \|W\|,$$

which in view of $\|M_i(W_i)\|_1 \leq \|W_i\|_1$ and the positivity of all the P_{ij}, implies

$$\|M_i(W_i)\|_1 = \|W\| \tag{3.17}$$

for all i. It follows from (3.16), (3.17) and Lemma 3.1 that

$$M_i(W_i) = W_i \tag{3.18}$$

for all i. We thus conclude from (3.12), (3.17) and (3.18) that there exist some j such that

$$\|\sum_{i=1}^m \tilde{P}_{ij}(W_i)\|_1 = \sum_{i=1}^m \tilde{P}_{ij} \|W_i\|_1 = \|W\|.$$

However, this can happen if and only if for every $r \in \{1, 2, \ldots, n\}$, there does not exist $1 \leq i, j \leq m$ such that the r-th coordinates $(W_i)_r$ and $(W_j)_r$ are of opposite signs.

By employing the above argument, and also the conclusion (3.18) to the equality $\|\Phi(\Phi(W))\| = \|\Phi(W)\|$ rather than to $\|\Phi(W)\| = \|W\|$, we have that for all $k \in \{1, 2, \ldots, m\}$

$$M_k\left(\sum_{i=1}^m \tilde{P}_{ik} W_i\right) = \sum_{i=1}^m \tilde{P}_{ik} W_i. \tag{3.19}$$

Assumption (ii) of our model is such that for every $r \in \{1, 2, ..., n\}$ there exists a matrix $M_k \in \mathcal{M}$ with positive r-th column. It follows from Lemma 3.2 and 3.19 that the r-th coordinate of $\sum_{i=1}^{m} \tilde{P}_{ik} W_i$ must vanish. But we have that there are no two indices i_1 and i_2 such that the r-th coordinates of W_{i_1} and W_{i_2} have opposite signs. This fact implies that the r-th coordinate of the vector W_i must vanish, and this is true for every $1 \leq i \leq m$. Since r is arbitrary, we conclude that $W_i = 0$ for all i. We have thus established that if $U, V \in \Sigma^m$ are distinct then

$$\|\Phi^2(U) - \Phi^2(V)\| < \|U - V\|.$$

The proof of the proposition is thus complete. ■

Theorem 3.1 *There exists a unique solution $V^\star$ for equation* (3.7)*, and the iteration scheme*

$$V^{(k+1)} = \Phi(V^{(k)}), k = 0, 1, 2, ...$$

with any starting point $V^{(0)} = V_0$ in Σ^m satisfies

$$\lim_{k \to \infty} V^{(k)} = V^\star. \tag{3.20}$$

Proof: The existence and uniqueness of solutions of (3.7) follows directly from the contractive property of Φ^2 and generalized Banach fixed point theorem (see [85], Chapter C.6.3). ■

Proof of Proposition 3.1. Let $V_i(k) = E[W(k)|A(k) = M_i]$. The sequence of vectors $V(k) = (V_1(k), \ldots, V_m(k)) \in \Sigma^m$ satisfies

$$V(k+1) = \Phi(V(k)). \tag{3.21}$$

From Theorem (3.1), $\{V(k)\}_{k=0}^{\infty}$ converge and the existence of the limits in (3.6) follows. In view of (3.21) these limits satisfy the fixed point equation (3.7). □

We have the following result which is actually Theorem 3.1 from [99]:

Corollary 3.1 *Let Assumption (i) hold, so that the probability that $A(k) = M_i$ is equal to ρ_i for every $k \geq 0$ and $1 \leq i \leq m$. Then the asymptotic expected value of $W(k)$ is the unique stochastic eigenvector of $\sum_{i=1}^{m} \rho_i M_i$ which corresponds to the eigenvalue* 1*.*

Proof: The sequence $\{A(k)\}$ of i.i.d. random matrices can be seen as a Markov chain on the set $\mathcal{M} = \{M_i \ : \ \rho_i > 0\}$ with the $m \times m$ transition matrix P given by $P_{ij} = \rho_j$. Since P_{ij} is positive for every i and j we have that $\tilde{P}_{ij} = \rho_i P_{ij}/\rho_j = \rho_i > 0$. We look for a solution of equation (3.1) for which all the components V_i are the same, say equal to $\bar{V}$. This yields the equation

$$\bar{V} = \left(\sum_{i=1}^{m} \rho_i M_i \right) \bar{V},$$

which implies the assertion of the corollary. ■

Theorem 3.2 *Under Assumption (ii), and assuming that the transition matrix P has strictly positive entries, then the asymptotic behavior of the expectation of the random variable $W(N)$ is given by:*

$$\lim_{N\to\infty} E(W(N)) = \sum_{i=1}^{m} \rho_i V_i^\star, \tag{3.22}$$

where $V^\star = (V_1^\star, \ldots, V_m^\star) \in \Sigma^m$ is the unique solution of (3.7), and $\rho = (\rho_1, \ldots, \rho_m)$ is the Perron eigenvector of the transition probability matrix (P_{ij}).

Proof: The proof is immediate:

$$\lim_{N\to\infty} E(W(N)) =$$
$$\lim_{N\to\infty} \sum_{i=1}^{m} E[W(N) \mid A(N) = M_i]P[A(N) = M_i] = \sum_{i=1}^{m} \rho_i V_i^\star.$$

■

3.3 The asymptotic variance of $W(N)$

The goal of this section is to compute the asymptotic value of the second order[1] moment of $W(N)$ under Assumption (ii) and assuming a positive transition matrix P.

Define the linear mapping $\Psi : (\mathbb{R}^{n\times n})^m \to (\mathbb{R}^{n\times n})^m$ by:

$$\Psi(D_1, \ldots, D_m) = \left(\sum_{i=1}^{m} \tilde{P}_{i1} M_i D_i M_i^T, \ldots, \sum_{i=1}^{m} \tilde{P}_{im} M_i D_i M_i^T \right) \tag{3.23}$$

Suppose for a moment that for $W(0) = s \in \Sigma$ the following limits exist:

$$D_i = \lim_{k\to\infty} E[W(k)W(k)^T | A(k) = M_i].$$

Comment 3.3 *Note that each D_i must be a symmetric nonnegative definite matrix, that it has nonnegative entries and it satisfies*

$$D_i u = \lim_{k\to\infty} E[W(k)W(k)^T u \mid A(k) = M_i] =$$
$$\lim_{k\to\infty} E[W(k) \mid A(k) = M_i] = V_i, \tag{3.24}$$

where u is the n-dimensional vector which satisfies $u_i = 1$ for every $1 \le i \le n$.

In view of (3.24) let $\mathcal{D}$ be the set:

$$\mathcal{D} = \{(D_1, \ldots, D_m) \mid D_i \in \mathbb{R}^{n\times n},\ D_i = D_i^T,\ D_i u = V_i\}. \tag{3.25}$$

It turns out that Ψ maps $\mathcal{D}$ into itself. Indeed, $(\Psi(D))_j$ is symmetric whenever all D_i are such. Moreover, using (3.7) we obtain

$$\sum_{i=1}^{m} \tilde{P}_{ij} M_i D_i M_i^T u = \sum_{i=1}^{m} \tilde{P}_{ij} M_i D_i u = \sum_{i=1}^{m} \tilde{P}_{ij} M_i V_i = V_j,$$

implying that $\Psi(D) \in \mathcal{D}$ for every $D \in \mathcal{D}$. We have the following result:

[1] What we call variance, or second order moment is actually covariance matrix for the vector $W(N)$: $E[(W(N))_i((W(N))_j)] - E[(W(N))_i]E[(W(N))_j]$.

Proposition 3.4 *For arbitrary $W(0) = s \in \Sigma$ and all $i \in \{1, 2, \ldots, m\}$ the following limits exist:*

$$D_i = \lim_{k\to\infty} E[W(k)W(k)^T | A(k) = M_i]. \tag{3.26}$$

The m-tuple $D = (D_1, ..., D_m) \in \mathcal{D}$ defined by (3.26) *satisfies the fixed point equation*

$$D = \Psi(D). \tag{3.27}$$

The proof will be given after having established Theorem(3.3)

Let

$$\mathcal{B} = \{C \mid C \in \mathbb{R}^{n\times n},\ C = C^T,\ Cu = 0\}. \tag{3.28}$$

Thus $\mathcal{B}$ is the vector space of all $n \times n$ symmetric matrices C such that all the columns of B belong to $\mathcal{S}$. A computation similar to the one preceding Proposition 3.4 implies that $\Psi(\mathcal{B}^m) \subset \mathcal{B}^m$. Since the difference between any two elements from $\mathcal{D}$ belongs to $\mathcal{B}^m$, then fixing any norm on $(\mathbb{R}^{n\times n})^m$, it follows that the linear mapping Ψ is a contraction on the metric space $\mathcal{D}$ if it is a contraction on the vector space $\mathcal{B}^m$. We wish to establish the existence and uniqueness of solutions $D \in \mathcal{D}$ of equation (3.27). To this end it is enough to find a norm in which the mapping Ψ^2 is a contraction on the complete metric space $\mathcal{D}$.

Let $\|\cdot\|$ be the norm on $\mathbb{R}^{n\times n}$ defined by :

$$\|A\| = \sum_{i,j=1}^{n} |A_{ij}| \ \text{ for } A \in \mathbb{R}^{n\times n}.$$

The next result establishes a crucial connection between this norm and the mapping $C \mapsto MCM^T$ for $C \in \mathcal{B}$ and $M \in \mathcal{M}$. It is close in spirit to Lemma 3.1.

Lemma 3.3 *Let $M \in \mathcal{M}$. Then the following relation*

$$MC \neq C \Rightarrow \|MCM^T\| < \|C\| \tag{3.29}$$

holds for every $C \in \mathcal{B}$.

Proof: As in the proof of Lemma 3.2, we consider a matrix M which has the form (3.8) for some $0 \leq q < n$, and where the last $r = n - q$ columns are positive. We then have

$$\|MCM^T\| = \sum_{i,j} |\sum_{k,l} m_{ik}c_{kl}m_{jl}| \leq \sum_{i,j}\sum_{k,l} m_{ik}m_{jl}|c_{kl}| =$$

$$= \sum_{k,l} |c_{kl}| \sum_{i} m_{ik} \sum_{j} m_{jl} = \sum_{k,l} |c_{kl}| = \|C\| \tag{3.30}$$

since M is column stochastic. We have thus established

$$\|MCM^T\| \leq \|C\| \tag{3.31}$$

for any column stochastic matrix M and any matrix C. We will next prove that equality holds in (3.31) only if $MC = C$. We remark that if $MC = C$, then in view of $C = C^T$, we have

$$MCM^T = CM^T = (MC)^T = C^T = C,$$

implying that equality holds in (3.31) if $MC = C$.

We now suppose that M and C are such that $\|MCM^T\| = \|C\|$. This is possible if and only if for each pair $1 \leq i, j \leq n$, the only inequality which appears in (3.30) is actually an equality. This, however, holds if and only if for every $1 \leq i, j \leq n$ the following holds:

Property S. *There are no two pairs of indices (k, l) and (k', l') such that $m_{ik}m_{jl}$ and $m_{ik'}m_{jl'}$ are both positive while c_{kl} and $c_{k'l'}$ have opposite signs.*

For $1 \leq i \leq q$ let $\Lambda_i = \{c_{il} \mid q < l \leq n\} = \{c_{li} \mid q < l \leq n\}$, and denote $\Lambda_0 = \{c_{kl} \mid q < k \leq n,\ q < l \leq n\}$. Using Property S for a pair $i, j \in \{1, 2, \ldots, q\}$, and noting that for all $k, l \in \{q+1, q+2, \ldots, n\}$, we have $m_{ik}m_{jj} > 0$, $m_{ii}m_{jl} > 0$ and $m_{ik}m_{jl} > 0$, and further we conclude that there are no two elements in set $\Lambda_{ij} = \Lambda_i \cup \Lambda_j \cup \Lambda_0$ with opposite sign. Since for each pair of indices (k, l) and (k', l') with $\max\{k, l\} > q$ and $\max\{k', l'\} > q$ there is pair $i, j \in \{1, 2, \ldots, q\}$ such that both c_{kl} and $c_{k'l'}$ are contained in Λ_{ij}, we conclude that either

$$c_{kl} \geq 0 \text{ whenever } \max\{k, l\} > q, \tag{3.32}$$

or

$$c_{kl} \leq 0 \text{ whenever } \max\{k, l\} > q. \tag{3.33}$$

For any integer $q < l \leq n$ the sum of the entries in the l-th column (or row) have the same sign as the constant sign of its elements, namely nonnegative (non-positive) if (3.32) ((3.33)) holds. Since this sum is zero, we conclude that all the entries c_{kl} that are such that at least one of $k > q$ and $l > q$ holds must vanish. Thus C must have the form:

$$C = \begin{pmatrix} C' & 0 \\ 0 & 0 \end{pmatrix}$$

where $C' \in \mathbb{R}^{q \times q}$, and since $M = \begin{pmatrix} I_q & M' \\ 0 & M'' \end{pmatrix}$, it follows that $MC = C$, concluding the proof of the lemma. ■

Now we are able to present the following short proof of Lemma 3.1.

Corollary 3.2 *Let $M \in \mathcal{M}$. Then for every $x \in \mathcal{S}$ we have*

$$Mx \neq x \Rightarrow \|Mx\|_1 < \|x\|_1. \tag{3.34}$$

Proof: Note that for $x \in \mathcal{S}$, $C = xx^T \in \mathcal{B}$ we can conclude that

$$\|MCM^T\| = \|Mxx^TM^T\| = \|Mx\|_1^2 \leq \|C\| = \|x\|_1^2$$

with equality if and only if $MC = C$. Moreover from the proof of the previous lemma we conclude that for all j such that the j-th column of M is positive, the j-th column of C must be zero, which also means that $x_j = 0$. Thus $\|Mx\|_1 = \|x\|_1$ implies that $Mx = x$. ■

We wish to employ the Banach fixed point theorem to the mapping Ψ on the set $\mathcal{D}$, with the metric which is induced on $\mathcal{D}$ by the following norm

$$\|B\| = \|(B_1, \ldots, B_m)\| = \max_{1 \leq i \leq m} \|B_i\| \tag{3.35}$$

on $(\mathbb{R}^{n \times n})^m$. Given this, we are now ready to establish that Ψ^2 is a contraction on the metric space $\mathcal{D}$.

Proposition 3.5 *Let Ψ be the mapping given by* (3.23) *and we assume that the transition matrix P is positive. Then there exists a constant $\eta < 1$ such that*

$$\|\Psi^2(D) - \Psi^2(E)\| \leq \eta \|D - E\| \tag{3.36}$$

holds for all $D, E \in \mathcal{D}$.

Proof: We will establish that for every nonzero $B \in \mathcal{B}^m$ we have $\|\Psi^2(B)\| < \|B\|$, which implies that there exists a $0 < \eta < 1$ such that

$$\|\Psi^2(B)\| \leq \eta \|B\|, \tag{3.37}$$

since $\mathcal{B}$ is a normed linear space. Clearly (3.36) follows from (3.37) since $D - E \in \mathcal{D}$. We thus consider any $B = (B_1, \ldots, B_m) \neq 0$ such that $B_i \in \mathcal{B}$ and compute

$$\|\Psi(B)\| = \max_j \|\sum_{i=1}^m \tilde{P}_{ij} M_i B_i M_i^T\| \tag{3.38}$$

$$\leq \max_j \sum_{i=1}^m \tilde{P}_{ij} \|M_i B_i M_i^T\| \tag{3.39}$$

$$\leq \max_j \sum_{i=1}^m \tilde{P}_{ij} \|B_i\| \tag{3.40}$$

$$\leq \max_j \sum_{i=1}^m \tilde{P}_{ij} \|B\| = \|B\|. \tag{3.41}$$

We will next check under which conditions the equality $\|\Psi^2(B)\| = \|B\|$ can hold. We thus assume that $B \neq 0$ is such that $\|\Psi(\Psi(B))\| = \|B\|$. It follows from

$$\|B\| = \|\Psi(\Phi(B))\| \leq \|\Psi(B)\| \leq \|B\|$$

that $\|\Psi(B)\| = \|B\|$. Thus it follows in this situation that all the inequalities in (3.38)-(3.41) are actually equalities.

In view of (3.40) and (3.41), for some j

$$\sum_{i=1}^{m} \tilde{P}_{ij}\|B_i\| = \max_i(\|B_i\|) = \|B\|. \tag{3.42}$$

Since we assume that all the $\tilde{P}_{ij}$ are positive, (3.42) implies

$$\|B_i\| = \|B\| \text{ for every } 1 \leq i \leq m. \tag{3.43}$$

It then follows from (3.39) that

$$\max_j \sum_{i=1}^{m} \tilde{P}_{ij}\|M_iB_iM_i^T\| = \max_j \sum_{i=1}^{m} \tilde{P}_{ij}\|B_i\| = \|B\|,$$

which together with the positivity of all the $\tilde{P}_{ij}$ imply that

$$\|M_iB_iM_i^T\| = \|B\| \tag{3.44}$$

for all i. In view of Lemma 3.3 it therefore follows from (3.43) and (3.44) that

$$M_iB_iM_i^T = B_iM_i^T = (M_iB_i^T)^T = (M_iB_i)^T = B_i,$$

namely the equalities

$$M_iB_i = B_i \text{ and } M_iB_iM_i^T = B_i \tag{3.45}$$

hold for all i. It follows from (3.38), (3.44) and (3.45) that there exist some j such that

$$\|\sum_{i=1}^{m} P_{ij}B_i\| = \sum_{i=1}^{m} P_{ij}\|B_i\| = \|B\|.$$

However, this can happen if and only if the following property holds:

A sign condition. *For every $r, s \in \{1, 2, \ldots, n\}$, there don't exist $1 \leq i, j \leq m$ such that the (rs)-th coordinates $(B_i)_{rs}$ and $(B_j)_{rs}$ have opposite signs.*

Employing the above argument, and the conclusion (3.45), to the equality $\|\Psi(\Psi(B))\| = \|\Psi(B)\|$ it follows that for all $k \in \{1, 2, \ldots, m\}$

$$M_k\left(\sum_{i=1}^{m} P_{ik}B_i\right) = \sum_{i=1}^{m} P_{ik}B_i. \tag{3.46}$$

Now let $l \in \{1, 2, \ldots, n\}$ be arbitrary. Then by Assumption (ii) of our model, there exists some matrix $M_k \in \mathcal{M}$ with positive l-th column. From (3.46) we can conclude that the columns of $\sum_{i=1}^{m} P_{ik}B_i$ are eigenvectors of the matrix M_k, which correspond to the eigenvalue 1. Moreover, they are convex combination of vectors from $\mathcal{S}$, hence they also belong to $\mathcal{S}$. Using Lemma 3.2, we can therefore conclude that the l-th column of the matrix $\sum_{i=1}^{m} P_{ik}B_i$ must vanish, and by employing the above sign condition, it follows that corresponding entries of the various l-th columns of the matrices

M_i don't have opposite signs. This implies that all the entries in the l-th columns of the matrices $B_1, \ldots, B_m$ must vanish. Since l is arbitrary, we conclude that

$$B_1 = B_2 = \cdots = B_m = 0.$$

This contradiction concludes the proof of the lemma. ■

Theorem 3.3 *The spectral radius of the restriction of the mapping Ψ to $\mathcal{B}^m$ is smaller than 1. In particular there exists a unique solution $D^\star$ for equation (3.27), and the iteration scheme*

$$D^{(k+1)} = \Psi(D^{(k)}), k = 0, 1, 2, ...$$

with the starting point $D^{(0)} = D_0$, satisfies

$$\lim_{k\to\infty} D^{(k)} = D^\star$$

for every $D_0 \in \mathcal{D}$.

Proof: The proof of this theorem follows the same lines and uses the same arguments as those employed in the proof of Theorem 3.1. ■

Proof of Proposition 3.4. Similarly to the proof of Proposition 3.1, let $D_i(k) = E[W(k)W(k)^T | A(k) = M_i]$ and $D(k) = (D_1(k), \ldots, D_m(k)) \in (\mathbb{R}^{n\times n})^m$. Then $D_i(k)u = E[W(k)W(k)^T u | A(k) = M_i] = E[W(k)|A(k) = M_i] = V_i(k) \to V_i$ as $k \to \infty$ which means that

$$\lim_{k\to\infty} \mathrm{dist}(D(k), \mathcal{D}) = 0.$$

Let $\varepsilon > 0$ and let $k_0 \in N$ be such that $\|D(k_0) - \tilde{D}\| < \varepsilon$ for some $\tilde{D} \in \mathcal{D}$. Since the mapping Ψ is nonexpansive in the norm given by (3.35) on all $(\mathbb{R}^{n\times n})^m$ we have:

$$\|D(k_0 + r) - \Psi^r(\tilde{D})\| = \|\Psi^r(D(k_0) - \tilde{D})\| \le \|D(k_0) - \tilde{D}\| \le \varepsilon.$$

On the other hand since $\tilde{D}$ is in $\mathcal{D}$,

$$\lim_{r\to\infty} \Psi^r(\tilde{D}) = D'$$

exists by the previous theorem. This means that there is k_1 such that for all $r > k_1$,

$$\|\Psi^r(\tilde{D}) - D'\| \le \varepsilon.$$

Now we conclude that for all $r > k_0 + k_1$:

$$\|D(r) - D'\| \le \|D(r) - \Psi^{r-k_0}(\tilde{D})\| + \|\Psi^{r-k_0}(\tilde{D}) - D'\| \le 2\varepsilon.$$

The last relation means that the sequence $\{D(r)\}$ is a Cauchy, and therefore $\lim_{r\to\infty} D(r)$ exists and (3.26) follows. Having this, (3.27) follows from the continuity of the linear mapping Ψ. □

Theorem 3.4 *Under Assumption (ii) and the positivity of the transition matrix P, the asymptotic behavior of the variance*

$$Var(W(N)) = E[W(N)W(N)^T] - E[W(N)]E[W(N)]^T$$

is given by:

$$\lim_{N\to\infty} Var(W(N)) =$$
$$= \sum_{i=1}^{m} \rho_i D_i^\star - \left(\sum_{i=1}^{m} \rho_i V_i^\star\right)\left(\sum_{i=1}^{m} \rho_i V_i^\star\right)^T \tag{3.47}$$

where $D^\star = (D_1^\star, \ldots, D_m^\star) \in \mathcal{D}$ *is the unique solution of* (3.27)*, and* $\rho = (\rho_1, \ldots, \rho_m)$ *is the Perron eigenvector of the transition matrix* (P_{ij})*.*

Proof: We have the following equalities:

$$\lim_{N\to\infty} Var(W(N)) =$$
$$\lim_{N\to\infty} E[W(N)W(N)^T] - \lim_{N\to\infty} E[W(N)]E[W(N)]^T =$$
$$= \lim_{N\to\infty} \sum_{i=1}^{m} E[W(N)W(N)^T \mid A(N) = M_i]P[A(N) = M_i]$$
$$- \lim_{N\to\infty} E[W(N)]E[W(N)]^T =$$
$$= \sum_{i=1}^{m} \rho_i D_i^\star - \left(\sum_{i=1}^{m} \rho_i V_i^\star\right)\left(\sum_{i=1}^{m} \rho_i V_i^\star\right)^T.$$

■

Corollary 3.3 *Let Assumption* (i) *hold, so that the probability that* $A(k) = M_i$ *is equal to* ρ_i *for every* $k \geq 0$ *and* $1 \leq i \leq m$*. Then the asymptotic behavior of* $Var(W(k))$ *is given by*

$$\lim_{N\to\infty} Var(W(N)) = \bar{D} - \bar{V}\bar{V}^T, \tag{3.48}$$

where $\bar{V}$ *is the unique stochastic eigenvector of the matrix* $\sum_{i=1}^{m} \rho_i M_i$*, and* $\bar{D}$ *is the unique solution of the matrix equation*

$$\sum_{i=1}^{m} \rho_i M_i D M_i^T = D,$$

which satisfies $Du = \bar{V}$*. Moreover,* $\bar{D}$ *is the unique eigenvector corresponding to eigenvalue* 1 *and satisfying* $Du = \bar{V}$ *of the linear mapping*

$$D \mapsto \sum_{i=1}^{m} \rho_i M_i D M_i^T \tag{3.49}$$

defined on $\mathbb{R}^{n\times n}$*.*

Proof: The sequence $\{A(k)\}$ of i.i.d. random matrices can be seen as Markov chain on the set $\mathcal{M} = \{M_i \ : \ \rho_i > 0\}$ with the $m \times m$ transition matrix P given by $P_{ij} = \rho_j$. Since P is positive for every i and j, it follows that $\tilde{P}_{ij} = \rho_i P_{ij}/\rho_j = \rho_i > 0$. We look for a solution of equation (3.27) for which all the components D_i are the same, say equal to $\bar{D}$. This yields the equation

$$\bar{D} = \sum_{i=1}^{m} \rho_i M_i \bar{D} M_i^T, \tag{3.50}$$

which implies the first assertion of the corollary.

For the second assertion we represent the linear mapping in (3.49) by an $n^2 \times n^2$ matrix with nonnegative entries, call it T, and apply Perron Frobenius Theorem to T. By Theorem 3.3 the spectral radius of the restriction of T to $\mathcal{B}$ is smaller than 1, but by (3.50) we have that $\bar{D}$ is an eigenvector corresponding to the eigenvalue 1. Since the iterations of the mapping (3.49) converge to $\bar{D}$, it follows that 1 is the unique eigenvector satsfying $\bar{D}u = \bar{V}$. The second assertion follows. ■

Example 3.1 *In this example we illustrate how the previous results can be applied. We consider a network where the bottleneck router operates according to Assumption(ii). In particular, consider a network of* 5 *flows with additive increase parameters* $\alpha = [5, 4, 3, 2, 1]$*, multiplicative decrease parameters given by* $\beta = [1/3, 2/4, 3/5, 4/6, 5/7]$*, and with corresponding vector* γ *given by* $\gamma = [1/60, 1/70, 1/80, 1/90, 1/100]$*. We assume that at congestion events the router drops packets from only one flow. Thus the set* $\mathcal{M}$ *has* 5 *elements:*

$$M_1 = \begin{bmatrix} 0.5054 & 0 & 0 & 0 & 0 \\ 0.1475 & 1.0000 & 0 & 0 & 0 \\ 0.1291 & 0 & 1.0000 & 0 & 0 \\ 0.1147 & 0 & 0 & 1.0000 & 0 \\ 0.1033 & 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1.0000 & 0.1291 & 0 & 0 & 0 \\ 0 & 0.6106 & 0 & 0 & 0 \\ 0 & 0.0968 & 1.0000 & 0 & 0 \\ 0 & 0.0860 & 0 & 1.0000 & 0 \\ 0 & 0.0774 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 1.0000 & 0 & 0.1033 & 0 & 0 \\ 0 & 1.0000 & 0.0885 & 0 & 0 \\ 0 & 0 & 0.6774 & 0 & 0 \\ 0 & 0 & 0.0688 & 1.0000 & 0 \\ 0 & 0 & 0.0620 & 0 & 1.0000 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} 1.0000 & 0 & 0 & 0.0860 & 0 \\ 0 & 1.0000 & 0 & 0.0738 & 0 \\ 0 & 0 & 1.0000 & 0.0645 & 0 \\ 0 & 0 & 0 & 0.7240 & 0 \\ 0 & 0 & 0 & 0.0516 & 1.0000 \end{bmatrix}$$

$$M_5 = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0.0738 \\ 0 & 1.0000 & 0 & 0 & 0.0632 \\ 0 & 0 & 1.0000 & 0 & 0.0553 \\ 0 & 0 & 0 & 1.0000 & 0.0492 \\ 0 & 0 & 0 & 0 & 0.7585 \end{bmatrix}$$

Let the transition matrix P to be given by:

$$P = \begin{bmatrix} 0.2667 & 0.2467 & 0.2133 & 0.1667 & 0.1067 \\ 0.2606 & 0.2424 & 0.2121 & 0.1697 & 0.1152 \\ 0.2526 & 0.2368 & 0.2105 & 0.1737 & 0.1263 \\ 0.2444 & 0.2311 & 0.2089 & 0.1778 & 0.1378 \\ 0.2370 & 0.2259 & 0.2074 & 0.1815 & 0.1481 \end{bmatrix}$$

Then

$$lim_{k\to\infty} E(W(k)) = V^{\star} = \begin{bmatrix} 0.2359 \\ 0.2295 \\ 0.2124 \\ 0.1852 \\ 0.1370 \end{bmatrix}$$

The meaning of this result is that the first flow should expect to get 23.59% of bandwidth at a congestion event, while the fifth flow should expect to get 13.7% of the bandwidth at a congestion event over the bottleneck link. The asymptotic behavior of variance of $W(k)$ in this example is given by:

$$lim_{k\to\infty} Var(W(k)) =$$

$$\begin{bmatrix} 0.0144 & -0.0061 & -0.0042 & -0.0027 & -0.0013 \\ -0.0061 & 0.0118 & -0.0029 & -0.0019 & -0.0009 \\ -0.0042 & -0.0029 & 0.0089 & -0.0012 & -0.0005 \\ -0.0027 & -0.0019 & -0.0012 & 0.0060 & -0.0002 \\ -0.0013 & -0.0009 & -0.0005 & -0.0002 & 0.0030 \end{bmatrix}.$$

The rate at which of $V^{(k)}$ and $D^{(k)}$ converge to their equilibrium values is depicted graphically in Figures 3.2 *and* 3.3 *respectively.*

Comment 3.4 *Note that the result from Theorem 3.1 refers to the resource allocation over congestion events. Model presented here is not suitable for the analysis of the instantaneous (or time averaged) resource allocation. We refer reader to the models presented in [7] and [98] which allow the analysis of the time averaged resource allocation.*

Comment 3.5 *The paper [97] discusses the dynamics of higher order moments of the stochastic process $W(k)$ in the related matrix model.*

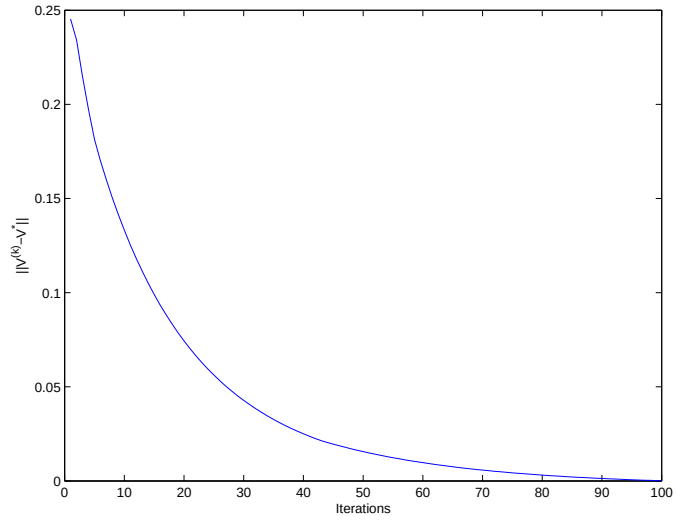

Figure 3.2: Evolution of $\|V^{(k)} - V^\star\|_1$.

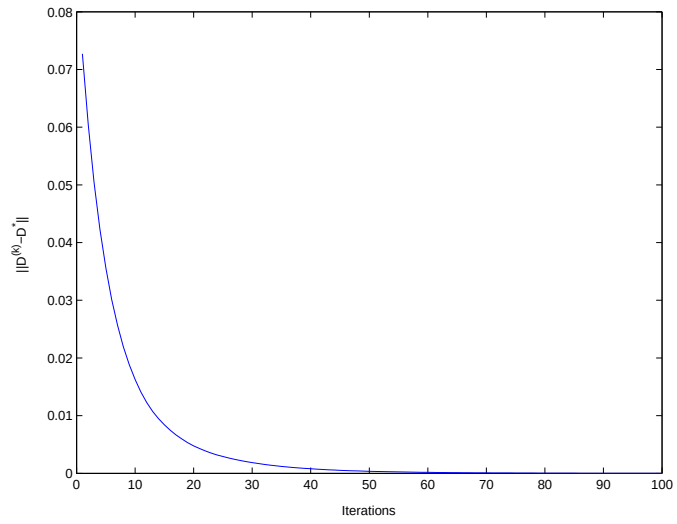

Figure 3.3: Evolution of $\|D^{(k)} - D^\star\|$.

3.4 A useful extension

In this section we will extend the results of the previous sections in the following sense. We will consider a transition probability matrix P which does not necessarily have positive entries, but is rather primitive; namely $P^s > 0$, for some integer $s \geq 1$. We note here that if P is primitive then $\tilde{P}$ is primitive too since they have the same zero-nonzero pattern.

Lemma 3.4 *If P is a primitive matrix such that $P^s > 0$ for some positive integer s, then Φ^{2s} is a contraction on Σ^m.*

Proof: We first note that for all $k, j, l \in \{1, 2, \ldots, m\}$, there is sequence $(i) = (i_1, i_2, \ldots, i_{2s-1})$ of indices which contains l, such that

$$\tilde{P}_{ki_{2s-1}} \tilde{P}_{i_{2s-1}i_{2s-2}} \cdots \tilde{P}_{i_2 i_1} \tilde{P}_{i_1 j} > 0.$$

Indeed, since $P^s > 0$ there must exist sequences $(i') = (i'_1, \ldots, i'_{s-1})$ and $(i'') = (i''_1, \ldots, i''_{s-1})$ such that

$$\tilde{P}_{ki'_{s-1}} \tilde{P}_{i'_{s-1}i'_{s-2}} \cdots \tilde{P}_{i'_2 i'_1} \tilde{P}_{i'_1 l} > 0$$

and

$$\tilde{P}_{li''_{s-1}} \tilde{P}_{i''_{s-1}i''_{s-2}} \cdots \tilde{P}_{i''_2 i''_1} \tilde{P}_{i''_1 j} > 0,$$

implying the existence of a sequence (i) with the desired property.

As we noted in section 3.2, proving that Φ^{2s} is a contraction on the metric space Σ^m is equivalent to proving that it is a contraction on the vector space $\mathcal{S}^m$, and here we establish the latter. Thus for an arbitrary $W \in \mathcal{S}^m$ the j component of $\Phi^{2s}(W)$ has the form

$$(\Phi^{2s}(W))_j =$$

$$= \sum_{(i)_{2s}} \tilde{P}_{i_{2s}i_{2s-1}} \tilde{P}_{i_{2s-1}i_{2s-2}} \cdots \tilde{P}_{i_1 j} M_{i_1} M_{i_2} \cdots M_{i_{2s}} W_{i_{2s}},$$

where we denote by $(i)_{2s}$ a sequence of indices which has length $2s$, and the summation is over all possible sequences $(i)_{2s}$. The sum in the last equality can be rewritten as follows:

$$(\Phi^{2s}(W))_j = \sum_{k=1}^{m} \sum_{(i)} \tilde{P}_{ki_{2s-1}} \cdots \tilde{P}_{i_1 j} M_{i_1} \cdots M_{i_{2s-1}} M_k W_k. \tag{3.51}$$

The inner sum in (3.51) is over all the sequences of indices (i) which have the length $2s - 1$.

Having this in mind we write:

$$\|\Phi^{2s}(W)\| =$$

$$= \max_j \| \sum_{k=1}^{m} \sum_{(i)} \tilde{P}_{ki_{2s-1}} \cdots \tilde{P}_{i_1 j} M_{i_1} \cdots M_{i_{2s-1}} M_k W_k \|_1 \tag{3.52}$$

$$\leq \max_j \sum_{k=1}^{m} \sum_{(i)} \tilde{P}_{ki_{2s-1}} \cdots \tilde{P}_{i_1 j} \| M_{i_1} \cdots M_{i_{2s-1}} M_k W_k \|_1 \tag{3.53}$$

$$\leq \max_j \sum_{k=1}^{m} \sum_{(i)} \tilde{P}_{ki_{2s-1}} \tilde{P}_{i_{2s-1} i_{2s-2}} \cdots \tilde{P}_{i_1 j} \| W_k \|_1 \tag{3.54}$$

$$\leq \max_j \sum_{k=1}^{m} \sum_{(i)} \tilde{P}_{ki_{2s-1}} \tilde{P}_{i_{2s-1} i_{2s-2}} \cdots \tilde{P}_{i_1 j} \| W \| = \| W \|. \tag{3.55}$$

We will next establish that $\|\Phi^{2s}(W)\| = \|W\|$ implies that $W = 0$, which will conclude the proof of the lemma. If $W \in \mathcal{S}^m$ satisfies $\|\Phi^{2s}(W)\| = \|W\|$ then all the previous inequalities (3.52), (3.53) and (3.54) are actually equalities. This means that there exists some $j \in \{1, 2, \ldots, m\}$ for which all the above maxima are attained at this j. For a $k \in \{1, 2, \ldots, m\}$ and a sequence (i) of indices we denote

$$Q_{kj}((i)) = \tilde{P}_{ki_{2s-1}} \tilde{P}_{i_{2s-1} i_{2s-2}} \cdots \tilde{P}_{i_1 j}.$$

It follows from $P^{2s} = (P^s)^2 > 0$ that for each $k \in \{1, 2, \ldots, m\}$ we have $\sum_{(i)} Q_{kj}((i)) > 0$. From this property together with

$$\sum_{k=1}^{m} \|W_k\|_1 \left(\sum_{(i)} Q_{kj}((i)) \right) = \max_k \|W_k\|_1 = \|W\|$$

and

$$\sum_{k=1}^{m} \sum_{(i)} Q_{kj}((i)) = 1$$

we can conclude that for every $k \in \{1, 2, \ldots, m\}$

$$\|W_k\|_1 = \|W\|.$$

It follows from the equality

$$\sum_{k=1}^{m} \sum_{(i)} Q_{kj}((i)) \|M_{i_1} M_{i_2} \cdots M_{i_{2s-1}} M_k W_k\|_1 =$$

$$= \sum_{k=1}^{m} \sum_{(i)} Q_{kj}((i)) \|W_k\|_1,$$

that for every sequence (i) such that $Q_{kj}((i)) > 0$ the following holds:

$$\|M_{i_1} M_{i_2} \cdots M_{i_{2s-1}} M_k W_k\|_1 = \|W_k\|_1,$$

which in turn implies

$$\|M_{i_1} \cdots M_{i_{2s-1}} M_k W_k\|_1 = \|M_{i_2} \cdots M_{i_{2s-1}} M_k W_k\|_1 \tag{3.56}$$

$$\cdots = \|M_{i_{2s-1}} M_k W_k\|_1 = \|M_k W_k\|_1 = \|W_k\|. \tag{3.57}$$

Employing Lemma 3.1, we conclude that for all sequences (i) with $Q_{kj}((i)) > 0$:

$$W_k = M_k W_k = M_{i_{2s-1}} W_k = \cdots = M_{i_1} W_k. \tag{3.58}$$

Recall now that for arbitrary k and l there exists a sequence (i) which contains l such that $Q_{kj}((i)) > 0$. Using (3.58) we conclude that

$$M_l W_k = W_k, \quad \forall k, l \in \{1, 2, \ldots, m\}, \tag{3.59}$$

and the relations (3.59) imply

$$W_1 = \cdots = W_m = 0. \tag{3.60}$$

Indeed, for each $h \in \{1, 2, \ldots, m\}$ there exists a matrix $M_l \in \mathcal{M}$ with positive h column(by Assumption (ii)) . Thus, Lemma 3.2 implies that the h coordinate of each W_k vanishes, and since h is arbitrary, (3.60) follows. The proof of the lemma is complete. ■

The following result may be established by using the same arguments as those employed in proving the previous lemma, and we will not repeat it here.

Lemma 3.5 *If P is a primitive matrix, such that $P^s > 0$ for some positive integer s, then Ψ^{2s} is a contraction on $\mathcal{D}$.*

Inspecting the proof of Theorem 3.1, we realize that we didn't use any special properties of the second power in deriving the proof while using the contractive property of Φ^2. Namely for any positive integer q, if Φ^q is contractive on Σ^m, then Theorem 3.1 follows. Similarly, if Ψ^q is contractive on $\mathcal{D}$ for some positive integer q then Theorem 3.3 follows. As a consequence of the previous two Lemmas, we have the following results.

Theorem 3.5 *Let Assumption (ii) hold and suppose that the transition matrix P is primitive, so that there exists an integer $s \geq 1$ such that P^s has positive entries. Then the spectral radius of the restriction of Φ to $\mathcal{S}^m$ is smaller than 1. In particular there exists a unique solution $V^\star$ for equation (3.7), and the iteration scheme*

$$V^{(k+1)} = \Phi(V^{(k)}), k = 0, 1, 2, ...$$

with any starting point $V^{(0)} = V_0$ in Σ^m

$$\lim_{k \to \infty} V^{(k)} = V^\star. \tag{3.61}$$

Moreover, the asymptotic behavior of the expectation of the random variable $W(N)$ is given by:

$$\lim_{N \to \infty} E(W(N)) = \sum_{i=1}^{m} \rho_i V_i^\star, \tag{3.62}$$

where $V^\star = (V_1^\star, \ldots, V_m^\star) \in \Sigma^m$ is the unique solution of (3.7), and $\rho = (\rho_1, \ldots, \rho_m)$ is the Perron eigenvector of the transition probability matrix (P_{ij}).

Theorem 3.6 *Let Assumption (ii) hold, and assume that the transition probability matrix P is primitive. Then the spectral radius of the restriction of the mapping Ψ to $\mathcal{B}^m$ is smaller than 1. In particular there exists a unique solution $D^\star$ for equation (3.27), and the iteration scheme*

$$D^{(k+1)} = \Psi(D^{(k)}), k = 0, 1, 2, ...$$

with the starting point $D^{(0)} = D_0$ satisfies

$$\lim_{k\to\infty} D^{(k)} = D^\star$$

for every $D_0 \in \mathcal{D}$. Moreover, the asymptotic behavior of the variance

$$Var(W(N)) = E[W(N)W(N)^T] - E[W(N)]E[W(N)]^T$$

is given by:

$$\lim_{N\to\infty} Var(W(N)) =$$
$$= \sum_{i=1}^{m} \rho_i D_i^\star - \left(\sum_{i=1}^{m} \rho_i V_i^\star\right)\left(\sum_{i=1}^{m} \rho_i V_i^\star\right)^T \tag{3.63}$$

where $D^\star = (D_1^\star, \ldots, D_m^\star) \in \mathcal{D}$ is the unique solution of (3.27), and $\rho = (\rho_1, \ldots, \rho_m)$ is the Perron eigenvector of the transition matrix (P_{ij}).

3.5 R-model

In the previous sections we considered the process $\{(A(k), W(k))\}_{k=0}^{\infty}$ under the assumption that $\{A(k)\}_{k=0}^{\infty}$ is a Markov process, and the distribution of $W(k+1)$ is determined by the value of $A(k)$ and the distribution of $W(k)$. In this model, the emphasis is put on the process $\{A(k)\}_{k=0}^{\infty}$, and $\{W(k)\}_{k=0}^{\infty}$ may be considered as a 'shadow' of it since the properties of $\{W(k)\}_{k=0}^{\infty}$ are derived from the distribution of $\{A(k)\}_{k=0}^{\infty}$.

However, one can construct a router such that the probability that a packet will be dropped at the k-th congestion event depends on the information provided by the vector $W(k)$, whose j-th coordinate is equal to the throughput of the j-th flow at the k-th congestion. We thus assume throughout this section that Assumption (iii) holds, and we will describe it again below.

When we consider the model under Assumption (iii), which we call *the R-model*, we assume that the value of $W(k)$ at the k-th congestion event, say $W(k) = w$, determines the distribution of $A(k)$. Namely, there exist Dini-continuous functions $w \mapsto p_i(w) \in \mathbb{R}^+$ on Σ such that

$$P[A(k) = M_i | W(k) = w] = p_i(w)(\forall 1 \le i \le m)(\forall w \in \Sigma) \tag{3.64}$$

and

$$\sum_{i=1}^{m} p_i(w) = 1 \text{ for every } w \in \Sigma.$$

In order to ensure that each flow has nonzero probability to detect a drop we assume that for each flow i there exists a matrix in $\mathcal{M}$ with positive i-th column.

We begin this Section by proving that for any initial distribution of $W(0)$ almost all products $\{A(k)\cdots A(0)\}_{k\in N}$ are weakly ergodic. Recall that a sequence $\{Q_k\}_{k\in N}$ of column-stochastic matrices is called weakly ergodic if

$$\lim_{k\to\infty} \text{dist}(Q_k, \mathcal{R}) = 0,$$

where we denote by $\mathcal{R}$ set of rank-1 column stochastic matrices. For any column stochastic matrix Q, we know that $Q(\mathcal{S}) \subset \mathcal{S}$ (see the proof of Proposition 3.2). Thus the restriction of Q to $\mathcal{S}$ is a mapping to itself, and we denote this map by $\tilde{Q}$. It follows from the definition of weak ergodicity given above that the sequence $\{Q_k\}_{k\in N}$ is weakly ergodic if and only if

$$\lim_{k\to\infty} \tilde{Q_k} = 0, \quad \text{(see [45])}.$$

Recall also that a linear operator on a vector space V is called paracontractive with respect to norm $\|\cdot\|$ if for all $x \in V$

$$Vx \neq x \Rightarrow \|Vx\| < \|x\|.$$

The main tool in establishing almost sure weak ergodicity will be the following result which is given in [26]:

Theorem 3.7 *Let $\|\cdot\|$ be a norm on $\mathbb{R}^m$ and let $\mathcal{F} \subset \mathbb{R}^{m\times m}$ be a finite set of linear operators which are paracontractive with respect to $\|\cdot\|$. Then for any sequence $\{A_k\}_{k\in N} \subset \mathcal{F}^N$, the sequence of left products $\{A_kA_{k-1}\ldots A_1\}_{k\in N}$ converges.*

Proposition 3.6 *Let the random variable $W(0)$ have arbitrary distribution on Σ. Under Assumption (iii), the sequence of products $\{A(k)A(k-1)\cdots A(0)\}_{k\in N}$ is weakly ergodic with probability 1, i.e.*

$$\lim_{k\to\infty} \tilde{A}(k)\tilde{A}(k-1)\cdots\tilde{A}(0) = 0 \text{ almost surely.} \tag{3.65}$$

Proof: From the assumption of positivity and the continuity of the mappings p_i on the compact set Σ it follows that

$$\eta = \inf\{p_i(s)|\ s \in \Sigma,\ i \in \{1,\ldots,m\}\} > 0.$$

This means that $p(A(k) \neq M_i) \leq 1 - \eta < 1$ for every $1 \leq i \leq m$. For every such i let T_i be a matrix with positive i-th column. Then with probability 1 the matrix T_i appears infinitely often in $\{A(k)\}_{k=0}^{\infty}$. We will next establish that weak ergodicity holds for left products $\{A_k\ldots A_1A_0\}_{k\in N}$ whenever for all i there are infinitely many appearances of T_i in $\{A_k\}_{k=0}^{\infty}$.

By Lemma 3.1, for all $M \in \mathcal{M}$, $\tilde{M}$ is paracontracting on $\mathcal{S}$ with respect to the L_1 norm. By Theorem 3.7 it follows that the sequence $\{\tilde{A}_k\tilde{A}_{k-1}\cdots\tilde{A}_0\}_{k\in N}$ is convergent, and we claim that the

limit is zero. To show this let $s \in \mathcal{S}$. Then there exist a $y \in \mathcal{S}$ such that $y = \lim_{k\to\infty} \tilde{A}_k \tilde{A}_{k-1} \cdots \tilde{A}_0 s$. For any fixed i let $\{A_{n_k}\}_{k\in N}$ be a subsequence of $\{A_k\}_{k\in N}$ with $A_{n_k} = T_i$. Then

$$y = \lim_{k\to\infty} \tilde{A}_{n_k} \tilde{A}_{n_k-1} \cdots \tilde{A}_0 s = T_i \lim_{k\to\infty} \tilde{A}_{n_k-1} \cdots \tilde{A}_0 s = T_i y,$$

hence $T_i y = y$. But by Lemma 3.2 i-th coordinate of y must be zero. Since i is arbitrary, it follows that $y = 0$. We have thus established that $\lim_{k\to\infty} \tilde{A}_k \tilde{A}_{k-1} \cdots \tilde{A}_0 = 0$, which implies the assertion of the proposition. ■

Comment 3.6 *The previous proposition is also established in [118], under Assumption (i), i.e. when all the functions p_i are constant.*

Note that in a sense, under Assumption (iii), the roles of $\{A(k)\}_{k=0}^{\infty}$ and $\{W(k)\}_{k=0}^{\infty}$ are interchanged compared to their roles in the model under Assumption (ii): the emphasis is put on $\{W(k)\}_{k=0}^{\infty}$, and $\{A(k)\}_{k=0}^{\infty}$ is considered as its shadow process.

We will henceforth restrict attention to stationary processes. The process $\{(A(k), W(k))\}_{k=0}^{\infty}$ is Markovian in the compact state space $\{1, 2, .., m\} \times \Sigma$, and we will next establish that it has a unique equilibrium distribution

$$\{\rho_1, ..., \rho_m\} \times (\lambda_1(dw), ..., \lambda_m(dw)).$$

Namely the probability that $A(k) = M_i$ and $W(k) \in U$ is equal, in the limit where $k \to \infty$, to $\rho_i \lambda_i(U)$.

We use the following result proved in [10] (see [103] for a nice survey of related results) that gives a sufficient condition for the existence of unique equilibrium measure.

Theorem 3.8 *(Barnsley et al. [10]) Let $p_i(x)$ be Dini-continuous and bounded away from zero. If*

$$\sup_{x\neq y} \sum_{i=1}^{m} p_i(x) \log \frac{\|M_i x - M_i y\|}{\|x - y\|} < 0 \tag{3.66}$$

then the Markov chain $W(k)$ has an unique stationary probability measure.

The following proposition shows that condition (3.66) is satisfied in the R-model.

Proposition 3.7 *In the R-model, (3.66) is true. Therefore the Markov chain $W(k)$ has an unique equilibrium measure.*

Proof: Note that from Jensen's inequality, for concave function $\log(t)$ we have:

$$\log\left(\sum_{i=1}^{m} p_i(x) \frac{\|M_i x - M_i y\|}{\|x - y\|}\right) \geq \sum_{i=1}^{m} p_i(x) \log \frac{\|M_i x - M_i y\|}{\|x - y\|}.$$

Now we prove that for any $x, y \in \Sigma.$, $x \neq y$ implies

$$\sum_{i=1}^{m} p_i(x) \frac{\|M_i x - M_i y\|}{\|x - y\|} < 1. \tag{3.67}$$

Since Σ is a compact set this proves (3.66). Denote by $w = x - y \in \mathcal{S}$. Then from Lemma 3.1 we know that $\|M_i w\| \leq \|w\|$ with equality if and only if $M_i w \leq w$. Thus

$$\sum_{i=1}^{m} p_i(x) \frac{\|M_i w\|}{\|w\|} \leq 1 \tag{3.68}$$

with equality if and only if $M_i w = w$ for all $i = 1, \dots m$. From Assumption (iii) we know that for any $j \in \{1, \dots, n\}$ there exist matrix $M_{i(j)}$ with positive j-th column, and from Lemma 3.2 $M_{i(j)} w = w$ is possible only if j-th component of vector w is zero. Since j is arbitrary, we conclude that $\sum_{i=1}^{m} p_i(x) \frac{\|M_i w\|}{\|w\|} = 1$ if and only if $w = 0$. ∎

The dynamics of $\{(A(k), W(k)\}_{k=0}^{\infty}$ can be described as follows. For a vector $W(k) = w$ at the instant of time k, choose a matrix $A(k)$ from $\mathcal{M}$ according to the distribution $\{p_i(w)\}_{i=1}^{m}$, and set $W(k+1) = A(k)w$. We then follow the steps

$$\begin{aligned} &W(0) \to A(0) \to W(1) = A(0)W(0) \to A(1) \to \cdots \\ &\to W(k) \to A(k) \to W(k+1) = A(k)W(k) \to \cdots \end{aligned} \tag{3.69}$$

We restrict attention only to the terms $W(k)$ in the chain of variables (3.69), and if $W(0) \sim (\lambda_1, ..., \lambda_m)$ then $\{W(k)\}_{k=0}^{\infty}$ turns out to be a stationary Markov process.

We now view the dynamics in a different manner, and this time we focus on the terms $A(k)$ in the above chain (3.69). If in the outset we restrict attention to stationary processes, then the distribution of each variable $W(k)$ is $(\lambda_1(dw), ..., \lambda_m(dw))$. Assuming this we restrict attention only to the variables $A(k)$ in (3.69), which turns out to be a stationary Markov chain in $\mathcal{M}$ provided that we take the initial distribution $A(0) \sim \rho$. We thus suppose that $W(k) \sim (\lambda_1, ..., \lambda_m)$, and that $A(k) = M_i$ for some $1 \leq i \leq m$. This determines the distribution of $W(k+1) = M_i W(k)$, as well as the distribution of $A(k+1)$. More explicitly we define $P_{ij} = E p_j(M_i W(k))$ where E denotes the expectation operation with respect to the distribution $\lambda(dw)$, namely

$$P_{ij} = \int p_j(M_i w) \lambda_i(dw). \tag{3.70}$$

Although we defined $P_{ij} = E p_j(M_i W(k))$, actually P_{ij} in (3.70) doesn't depend on k since all the variables $W(k)$ have the same distribution λ. However, we have to verify that our construction does yield this distribution to all $W(k)$. But indeed, since $\rho \times (\lambda_1, ..., \lambda_m)$ is an equilibrium distribution for the Markov process $\{(A(k), W(k))\}_{k=0}^{\infty}$, it follows that if we have $A(k) \sim \rho$, then the distribution of $W(k+1)$ is $(\lambda_1, ..., \lambda_m)$, and that of $A(k+1)$ is ρ.

We are interested in the asymptotic behavior of $W(N)$ where $N \to \infty$, which in view of $W(k+1) = A(k)W(k)$ reduces to the study of the asymptotic distribution of the products

$$\left(\prod_{k=0}^{N} A(k) \right) W(0) \tag{3.71}$$

when $N \to \infty$. Since weak ergodicity holds for the matrix products $\Pi_{k=0}^{N} A(k)$ it follows that the asymptotic behavior of the expressions in (3.71) doesn't depend on $W(0)$ there, and we consider these expressions with an arbitrary choice of $W(0) \in \Sigma$. We define

$$V_i^N = E\left[A(N-1)A(N-2)\cdots A(0)W(0)|A(N) = M_i\right], \tag{3.72}$$

where the expectation is with respect to the measure in which $\{A(k)\}_{k=0}^{\infty}$ is a stationary Markov chain with the transition probability matrix P in (3.70). Associated with this P is the matrix $\tilde{P}$ of backward probabilities, so that $\tilde{P}_{ij}$ is the probability of having $A(k) = M_i$ given that $A(k+1) = M_j$. Thus assuming $A(N+1) = M_j$, it follows from (3.72) that

$$\begin{aligned} V_j^{N+1} &= E\left[A(N)A(N-1)\cdots A(0)W(0)|A(N+1) = M_j\right] \\ &= \sum_{i=1}^{m} \tilde{P}_{ij} E\left[M_i A(N-1)\cdots A(0)W(0)|A(N) = M_i, A(N+1) = M_j\right] \\ &= \sum_{i=1}^{m} \tilde{P}_{ij} M_i E\left[A(N-1)\cdots A(0)W(0)|A(N) = M_i\right], \end{aligned}$$

where in the last equality we have used the Markov property. Equating the first and last terms and using (3.72) we obtain the relations

$$V_j^{N+1} = \sum_{i=1}^{m} \tilde{P}_{ij} M_i V_i^N, \ N \geq 0. \tag{3.73}$$

But we observe now that (3.73) is an iterations scheme for the fixed point equation (3.7). Thus the results of the previous sections imply that the following limits exist

$$\lim_{N\to\infty} V_i^N = V_i^\star \text{ for every } 1 \leq i \leq m, \tag{3.74}$$

where $V^\star = (V_1^\star, ..., V_m^\star)$ is the unique solution of (3.7). As a consequence of this discussion we have the following:

Theorem 3.9 *The conclusions of Theorems* 3.5 *and* 3.6 *hold true when we replace Assumption* (*ii*) *there by Assumption* (*iii*) *where* P_{ij} *is defined by (3.70).*

3.6 Summary

In this chapter we consider the dynamics of AIMD-networks that evolve according to Markovian dynamics. We have shown that such networks have well defined stochastic equilibria and provided tools that can be used to characterize these equilibria. In particular, for routers that operate according to Assumption (ii), we have developed tools for computing $\lim_{k\to\infty} E(W(k))$ and $\lim_{k\to\infty} Var(W(k))$. We then extended these results to the R-model given by Assumption (iii).

While developing these tools represent an important first step in studying such networks, much work remains to be done. The results derived in this chapter provide tools to address the problem of designing routers that achieve, in the long run, certain goals. By controlling the distribution of the random variable $A(0)$ in the i.i.d. case (i), or the transition matrix P in the Markov cases (ii) and (iii), one can guarantee that in the long run, the asymptotic expected value of $W(k)$ is close to a certain prescribed vector $V^\star$. A major objective of future work will be to investigate how this might in fact be achieved.

Another interesting designing problem that is of great practical interest, and which may be addressed in the setting provided by either Assumption (ii) or Assumption (iii), is the following. For a prescribed vector $V^\star$, consider all the transition matrices P for which

$$\lim_{k\to\infty} E(W(k)) = V^\star, \tag{3.75}$$

and among them pick one for which

$$\lim_{k\to\infty} \mathrm{Var}(W(k)) = T^\star \tag{3.76}$$

is the smallest possible in a certain sense. Minimizing the variance makes it more likely that the desired long-run behavior expressed by $\lim_{k\to\infty} E(W(k)) = V^\star$ will be realized faithfully (although the cost of this choice may be a slow network convergence or some other undesirable network behavior). This goal defines a constrained optimization problem which may be addressed either numerically or theoretically. We note that the minimization may be approached in various ways: either minimizing a certain functional, e.g. the trace of $T^\star$, or looking for a $T_0^\star$ such that

$$T_0^\star \leq T^\star$$

in the positive definite sense, where $T_0^\star$ and $T^\star$ correspond to certain matrices P_0 and P such that (3.75) and (3.76) hold.

Part II

Resource allocation

CHAPTER 4

Beyond CHOKe: Stateless fair queueing

Abstract - *Making drop decisions to enforce the max-min fair resource allocation in network of standard TCP flows without any explicit state information is a challenging problem. Here we propose a solution to this problem by developing a suite of stateless queue management schemes that we refer to as Multi-Level Comparison with index l (MLC(l)). We show analytically, using a Markov chain model, that for an arbitrary network topology of standard TCP flows and queues employing MLC(l), the resource allocation converges to max-min fair as l increases. The analytical findings are verified experimentally using packet level ns2 simulations.*

4.1 Introduction

Resource allocation in communication networks has been a topic of interest for some time. In the current Internet most traffic uses TCP as the transport protocol, and most Internet routers do not differentiate packets from different flows. In order to adjust the resource allocation amongst competing users, one can do the following: (1) design the new end-to-end protocol(s) and leave the network infrastructure (routers) unchanged [122, 58, 37]; (2) design the new end-to-end protocol(s) and network support that will allow cooperation between end-users and network (routers) [55, 3, 120]; (3) leave the end-to-end protocol(s) unchanged but design the network based scheme that determines desired resource allocation [110, 18, 100].

Most current proposals have as their performance objective a resource allocation that is max-min fair. In this chapter we propose a scheme that belongs to the third group listed above, and whose performance goal is enforcing a max-min fair resource allocation. Its main features are the following.

1. No changes to end-to-end transport protocols are required.
2. The decision to drop (mark) a packet is made locally by each router;
3. No multiple queues or per flow counters are used.

Thus, our goal is to design a stateless active queue management scheme that can enforce max-min fairness in the network of TCP users. While there exists a large amount of work related to analysis and design of distributed algorithms/architecture that enforce max-min fairness, to the best of our knowledge, our algorithm is the first that attempts a stateless active queue management scheme to enforces max-min fairness in the network of TCP users.

4.1.1 Chapter contributions

Why is reaching the goal stated above so hard? First, recall that in the max-min fair regime, a TCP flow f experiences drops at *one and only one* link l_f at its path (we say that f is bottlenecked at l_f), and therefore must be protected at other links (that can be congested) by receiving lossless service. Second, if two or more flows are bottlenecked at the same link they must receive nonuniform loss rates that are function of their aggressiveness (round-trip times, queuing delays, delayed acks option, etc). Assuming that the router has access to the individual flow rates, or the existence of multiple queues that are appropriately scheduled, a number of solutions to these two problems exist, and are described in previous works [110, 55, 88, 18, 100]. However, in our case (routers with no explicit state information) it is highly nontrivial to make the drop (mark) decision without any explicit information.

The main contributions of this chapter are the following:

• A stateless queue management scheme, Multi Level Comparisons with index l (MLC(l)), which makes the drop decision based on the structure of packets that are already in the queue (using simple comparisons only).

• A Markov chain analysis of the randomized algorithm MLC(l) that shows that the resource allocation of MLC(l) converge to the max-min fair, for arbitrary network topology and arbitrary set of TCP users, as the index l grows.

• Packet level simulations that support the analytical findings.

4.2 Power-drop AQM schemes

It has been noticed in many studies that both drop tail and RED routers have large bias against large-RTT flows. For example the authors of [71] have made the empirical observation that for a drop-tail router and two flows with round trip times RTT_1 and RTT_2, the ratio of asymptotic throughput of the first and the second flow is in the ratio $(RTT_2/RTT_1)^a$ for some $a \in (1, 2)$. Similarly, it has also been noticed in a number of studies, that oblivious (ones that do not differentiate packets from different flows) AQM schemes (RED [40], BLUE [36], etc.) which attempt to estimate the loss probability for a given traffic pattern and to drop packets according to this estimation, share bandwidth among

competing users with round trip times RTT_1 and RTT_2 in the ratio RTT_2/RTT_1, [38, 1]. In this section we will investigate RTT unfairness characteristics for more general AQM schemes we call power-drop AQM schemes.

Definition 4.1 *An AQM is power-drop if it drops a packet from a flow with current throughput*[1] *U with probability $\rho_0 U^{l-1}$, where ρ_0 is a variable controlled by router and l a positive integer called the index of the given power-drop AQM.*

Comment 4.1 *With $l = 1$ this corresponds to a router which drops packets with loss probability ρ_0. The case when $l = 2$ is similar to CHOKe [89] in the limit when the average queue size does not go below minimum threshold, and in addition, when there is neither a RED nor an overflow drop. Indeed, comparing a packet at the entrance of a queue with a packet from the queue and making drop-decision based on this comparison is actually dropping a packet with probability which is proportional to current throughput of the flow.*

In this section we will describe a class of power-drop AQM's called Multi Level Comparison (MLC) AQM's. In particular, we will describe and analyze the fairness characteristics of this queueing discipline for TCP flows competing for bandwidth. We will see that the MLC scheme with index l achieves $1/RTT^{1/(l+1)}$-fairness[2] under the assumption of low loss probability (Theorem 4.1). More generally, Theorem 4.2 shows that increasing l leads resource allocation among TCP users arbitrarily close to max-min fairness in general network topologies.

4.2.1 Description of MLC

The basic strategy in MLC(l) is to extend the core idea from CHOKe of comparing of a packet arriving at the queue with packets which are already in queue; these stored packets are measure of the proportion of bandwidth used by certain flow. MLC(l) maintains a variable h_M which is used to control the probability of dropping an arriving packet: at every packet arrival h_M dropping trials are executed.

> *Dropping trial:* Pick randomly $l - 1$ packets from the queue: if *all* l packets belong to the same flow, then drop the arriving packet (If $l = 1$ the arriving packet is dropped by default).

If the arriving packet is not dropped after the execution of h_M dropping trials then it is enqueued. If h_M is not an integer, the number of dropping trials is given as follows. For $h_M < 1$ we execute 1 dropping trial with probability h_M and 0 dropping trial with probability $1 - h_M$. Similarly, $h_M > 1$ we execute $\lfloor h_M \rfloor + 1$ dropping trials with probability $\{h_M\} = h_M - \lfloor h_M \rfloor$ and $\lfloor h_M \rfloor$ dropping trials with probability $1 - \{h_M\}$.

Proposition 4.1 *For a given h_M, MLC(l) is power-drop scheme with index l.*

[1] Throughput is measured in packets per unit of time.

[2] Two flows with round trip times RTT_1 and RTT_2 which have a single bottleneck operating with MLC, obtain bandwidth in ratio: $(RTT_1/RTT_2)^{-1/(l+1)}$.

Proof: Let U_f be the throughput of a flow f, and U_0 the aggregate throughput on the link. A packet is dropped at one dropping trial with probability

$$q_1 = \left(\frac{U_f}{U_0}\right)^{l-1}.$$

Probability that a packet is dropped after h_M trials is 1-$Prob$[packet is not dropped at any of h_M trials] which is given by

$$q = 1 - (1 - q_1)^{h_M} \approx q_1 \cdot h_M = U_f^{l-1} \frac{h_M}{U_0^{l-1}}.$$

Taking $\rho_0 = \frac{h_M}{U_0^{l-1}}$ we conclude that MLC(l) satisfies Definition 4.1. ■

The higher h_M the more frequent the losses are. Consequently, if the link is under-utilized, h_M should be decreased in order to decrease probability of dropping packets. On the other hand if the aggregate traffic on the link is greater than the link capacity then h_M should be increased to reduce traffic load.

Controlling the variable h_M : MLC uses a parameter Δ_0 to affect changes in the variable h_M (in our simulations Δ_0 is set to $100ms$). h_M is adjusted once per Δ_0 using a MIMD (Multiplicative Increase - Multiplicative Decrease) scheme. The performance goal is to keep the utilization at a certain level u_0. Namely, if within the previous Δ_0 the link utilization was less than desired u_0, h_M is set to h_M/γ for some $\gamma > 1$, otherwise h_M is adjusted as $h_M := h_M\gamma$.

At this point it is important to emphasize a few differences between MLC and CHOKe. First, note that CHOKe makes a comparison only when the average queue size becomes greater than min_{th} (RED minimum threshold), and therefore its performance (in terms of resource allocation between TCP users) depends mainly on the number of users: a small number of TCP flows will affect the synchronization of losses, while for large number of users, the number of CHOKe-drops will be much less than number of RED-drops and therefore the effect of CHOKe on TCP fairness would be negligible. Second, the design of CHOKe basically neglects the TCP fairness as a performance objective and concentrates on reducing throughput of unresponsive flow(s) [112], while MLC is designed to improve TCP fairness and neglects effects of unresponsive flows.

Our experiments indicate that the parameter Δ_0 should be in the range of round trip times of the connections using the link (in order to allow users to react to changes in h_M). The parameter γ controls the speed of adaptation to changes in network traffic and should be set such that, for a given Δ_0, h_M can be doubled/halved within a few seconds.

4.2.2 Model and analysis of power-drop AQM

In this section we present a model of power-drop AQM's servicing multiple TCP users. We present results that characterize this situation for both a single bottleneck and for general network topologies.

Single bottleneck case: We consider N TCP-flows with heterogenous round trip times RTT_i, $i = 1, \ldots, N$, traversing a single bottleneck link that employs power-drop AQM with an index l. If we assume that ρ_0 does not fluctuate much (so that we can model it as constant) and that the drop

probability for a packet is small, then our analysis shows that the asymptotic rates achieved by TCP users are proportional to $\frac{1}{RTT_i^{2/(l+1)}}$. This is the main result of this section and is given in Theorem 4.1.

Model : At the flow level, let Δ be the length of sampling interval over which we evaluate changes in throughput. If a flow with a round trip time RTT does not see a drop within interval of length Δ, then its throughput will be increased by Δ/RTT^2. If a flow registers a drop within this sampling interval then its throughput will be halved[3]. The probability that the first event will happen is equal to probability that each of ΔU packets from the flow are not dropped. This probability is given by:

$$\eta_1 = (1 - \rho_0 U^{l-1})^{\Delta U} \approx e^{-\Delta\rho_0 U^l}.$$

Clearly, the probability that a flow with current throughput U will see a drop within a sampling interval of length Δ is equal to

$$\eta_2 = 1 - (1 - \rho_0 U^{l-1})^{\Delta U} \approx 1 - e^{-\Delta\rho_0 U^l}.$$

The previous approximations are valid under the assumption of a small probability that a packet will be dropped : $\rho_0 U^{l-1} \ll 1$. This assumption seems reasonable, since if this probability is not small, a flow would suffer too many losses and therefore would not get a chance to enter the (AIMD) congestion avoidance phase.

Let $U_k^{(\rho)}$ be a stochastic process which describes the evolution of throughput of a TCP flow with round-trip time RTT traversing over link with a power-drop AQM scheme with index l. Here $\rho = \Delta\rho_0$. Since Δ is fixed we can assume that Δ is equal to one unit of time.

We model $U_k^{(\rho)}$ as a Markov chain on $[0, \infty)$ defined by $U_0^{(\rho)} = 0$ and:

$$U_{k+1}^{(\rho)} = U_k^{(\rho)} + \frac{1}{RTT^2} \quad \text{with probability} \quad e^{-\rho(U_k^{(\rho)})^l}$$

$$U_{k+1}^{(\rho)} = \frac{1}{2}U_k^{(\rho)} \quad \text{with probability} \quad 1 - e^{-\rho(U_k^{(\rho)})^l}.$$

The following theorem characterizes the time averaged throughput of a TCP flow with round trip time given by RTT, running over power-drop queue management scheme with index l and its proof is given in Appendix.

Theorem 4.1 *The time averaged throughput of the i'th flow:* $\frac{1}{M}\sum_{i=1}^{M} U_i^{(\rho)}$ *converges almost surely to:*

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} U_i^{(\rho)} =: \overline{U}^{(\rho)} = \frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}} D_{MLC}(l) + \frac{1}{\rho^{\frac{1}{l+1}}} S^{(\rho)}$$

where $D_{MLC}(l)$ is a constant that does not depend on ρ or RTT and $S^{(\rho)}$ converges to 0 as ρ goes to 0.

[3]Throughout this chapter, variations in round trip times are neglected.

Comment 4.2 *The previous theorem is a generalization of well known square root formula. Indeed, for $l = 1$, power-drop scheme is an oblivious AQM that drops packets with probability $\rho = h_M$ and Theorem 4.1 says that time averaged throughput converges to $\frac{1}{RTT\sqrt{\rho}} D_{MLC}(1) + o(\frac{1}{\sqrt{\rho}})$.*

To conclude this section we prove that for a given network with routers employing a power-drop AQM with index l, and assuming that the steady state throughput is given by the previous theorem, we can find large enough l such that bandwidth allocation is arbitrary close to max-min fairness. The following characterization of max-min fairness can be found in [102].

Lemma 4.1 *A set of rates x_r is max-min fair if and only if for every flow r there exists a link on its path, such that the rates of all flows which traverse through that link are less or equal than x_r.*

With this characterization of max-min fair allocation in mind, we shall prove that increasing the index of the MLC will result in allocation of bandwidth in such fashion that each flow will have link on its path such that its asymptotic rate is "almost" the largest among all flows using that link.

Theorem 4.2 *For any given network topology, and given $\varepsilon > 0$, there exists l such that if all queues employ MLC with index l and loss probabilities are small then for every flow r there exist a link on its path, such that the rates of all flows which traverse through that link are less than $(1+\varepsilon)x_r$ (here x_r is steady state rate of flow r).*

Proof: Let L be the number of links in the network and N the number of flows. We label flows by $i = 1, 2, \ldots, N$ and links by $s = 1, 2, \ldots, L$. By R we denote the routing matrix: $R_{is} = 1$ if flow i uses link s otherwise $R_{is} = 0$. On each link s, a router drops a packet from the flow with current throughput U with probability $\rho(s)U^{l-1}$. Let M be the length (in number of links) of the path of the flow with most links on its route and ν the ratio of the largest and the smallest round trip time in the network. Choose l such that

$$\nu^{\frac{2}{l+1}} M^{\frac{1}{l+1}} < 1 + \varepsilon.$$

For each flow r, let $s_1^{(r)}, \ldots, s_w^{(r)}$ be links used by it and let $s_{max}^{(r)}$ the most congested link on its route in the following sense:

$$\rho(s_{max}^{(r)}) = \max\{\rho(s_j^{(r)}) |\ j = 1, \ldots, w\}. \tag{4.1}$$

If the current rate of flow r is U, a packet from that flow will be dropped with probability $\lambda_r U^{l-1}$, where $\lambda_r = \sum_{j=1}^{w} \rho(s_j^{(r)})$, and therefore the steady state throughput for flow r is given by

$$x_r = \frac{1}{RTT_r^{\frac{2}{l+1}} \lambda_r^{\frac{1}{l+1}}} C_0.$$

For any other flow t which uses the link $s_{max}^{(r)}$ with $\lambda_t = \sum_{j:R_{jt}=1} \rho(l_j) \geq \rho(s_{max}^{(r)})$ the steady state throughput is given by:

$$x_t = \frac{1}{RTT_t^{\frac{2}{l+1}} \lambda_t^{\frac{1}{l+1}}} C_0.$$

Recall that we have defined the link $s_{max}^{(r)}$ as the most congested link on route of flow r in the sense of (4.1). This implies that $\lambda_r \leq M\rho(s_{max}^{(r)})$. Now

$$\frac{x_t}{x_r} = \left(\frac{RTT_r}{RTT_t}\right)^{\frac{2}{l+1}} \left(\frac{\lambda_r}{\lambda_t}\right) \leq \left(\frac{RTT_r}{RTT_t}\right)^{\frac{2}{l+1}} \left(\frac{M\rho(s_{max}^{(r)})}{\rho(s_{max}^{(r)})}\right)^{\frac{1}{l+1}} \leq \nu^{\frac{2}{l+1}} M^{\frac{1}{l+1}} < 1 + \varepsilon.$$

■

Comment 4.3 *Note that for a single bottleneck topologies resource allocation given by* $\frac{C}{RTT_i^{2/(l+1)}}$ *is* $((RTT_i^2), l+1)$ *proportionally fair [80, 102]. Indeed, for any resource allocation* (x_i)*, utility* $U(x) = \sum_{i=1}^{N} \frac{RTT_i^2}{x_i^l}$*, and link capacity* c_0 *we have (using Holder's inequality):*

$$(U(x))^{\frac{1}{l+1}} \cdot c_0^{\frac{l}{l+1}} = \left(\sum_{i=1}^{N} \frac{RTT_i^2}{x_i^l}\right) \cdot (\sum_{i=1}^{N} x_i) =$$

$$\left(\sum_{i=1}^{N} \left(\frac{RTT_i^{2/(l+1)}}{x_i^{\frac{l}{l+1}}}\right)^{l+1}\right)^{\frac{1}{l+1}} \cdot \left(\sum_{i=1}^{N} \left(x_i^{\frac{l}{l+1}}\right)^{\frac{l+1}{l}}\right)^{\frac{l}{l+1}} \leq$$

$$\leq \sum_{i=1}^{N} \frac{RTT_i^{2/(l+1)}}{x_i^{\frac{l}{l+1}}} \cdot x_i^{\frac{l}{l+1}} = \sum_{i=1}^{N} RTT_i^{2/(l+1)}$$

$U(x)$ is maximized if equality holds in the inequality above, which is equivalent to $x_i = \frac{C}{RTT_i^{2/(l+1)}}$ *for some constant C. Thus, while spectrum of delay-based end-to-end protocols [80] assume no cooperation from routers to converge to max-min fairness, MLC(l) does not require changes in end-to-end protocol to enforce max-min fairness.*

A similar interesting feature is shared between shuffling parameter $\gamma > 0$ of XCP [55] and index l of MLC. Namely, in the recent paper [74] the authors proved that "... *given any network topology, one can choose a shuffling parameter, γ, sufficiently small so that the resulting allocation is close to max-min fairness. For any fixed $\gamma > 0$, however, there are topologies in which some flow rates can be far away from their max-min allocations*". In the case of MLC, for given network topology we can chose sufficiently large index l so that resulting allocations are close to max-min, but for any fixed l there are topologies in which some flow rates can be far away from their max-min allocations.

4.3 Experimental results

In this section we briefly describe some *ns*2 simulations that demonstrate the behavior of proposed AQM schemes.

4.3.1 Single bottleneck

The first set of simulations are designed to demonstrate the fairness properties of the MLC in single bottleneck scenario. Specifically, we present results for a single link with service rate of $80Mbps$ that

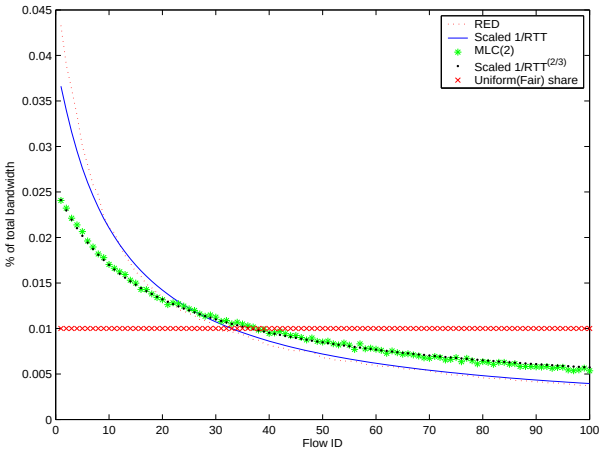

Figure 4.1: Scaled throughput for 100 flows over congested link employing RED, MLC(2).

services 100 long-lived TCP users with round trip times uniformly distributed in the range $40-440ms$. To provide baseline results, we include the performance of RED for the same scenario. Share of total throughput taken by each of 100 flows assuming the bottleneck queue is managed by MLC(2) is depicted in Figure 4.1.

It can be seen from Figure 4.1 that the fairness of RED is approximately proportional to the inverse of RTT. This is in accordance with observations made in [1, 38]. It can also be observed that the fairness of MLC with index 2 is proportional to $1/RTT^{2/3}$ as predicted by Theorem 4.1. The MLC parameters used in the simulation are: $l = 2$, $\Delta_0 = 100ms$, $\gamma = 1.01$, $u_0 = 0.98$.

4.3.2 Multiple bottleneck topologies

Our second set of simulations demonstrate Theorem 4.2. The network topology that we considered is given in Figure 5.7. Here, we consider a network of 24 nodes: $n1-n5, m1-m5, p1-p5, q1-q5$, and $c1$, $c2$, $c3$, $c4$ and 30 flows traversing the network as follows: $n(i) \rightarrow p(i); n(i) \rightarrow q(i), m(i) \rightarrow p(i); m(i) \rightarrow q(i); n(i) \rightarrow m(i); p(i) \rightarrow q(i)$ where $i = 1, 2, 3, 4, 5$.

The delays on each of the links in ms are defined as follows:

$$ni \rightarrow c1 \ : \ 40 \cdot i + 1; \quad pi \rightarrow c3 \ : \ 40 \cdot i + 1$$

$$mi \rightarrow c2 \ : \ 40 \cdot i + 1; \quad qi \rightarrow c4 \ : \ 40 \cdot i + 1$$

and the delays $c1-c2$, $c2-c3$, $c3-c4$ are $10ms$. The capacities of all links are $10Mbps$. With this topology, the max-min fair shares are $0.5Mbps$ for 20 flows that uses link $c2-c3$, and $1Mbps$ for other 10 flows ($n(i) \rightarrow m(i)$ and $p(i) \rightarrow q(i)$).

Each flow uses the standard TCP-SACK algorithm, with a packet size 1000B. The aggressiveness of each flow is mainly determined by its RTT. The behavior of the network is evaluated with each link $c1-c2$, $c2-c3$ and $c3-c4$ using: DropTail, RED, MLC(2), MLC(3), MLC(5), MLC(9), and MLC(17) with a queue size of 100 packets. MLC(a) parameters are: $l = a$, $\Delta_0 = 100ms$, $\gamma = 1.01$, $u_0 = 0.98$.

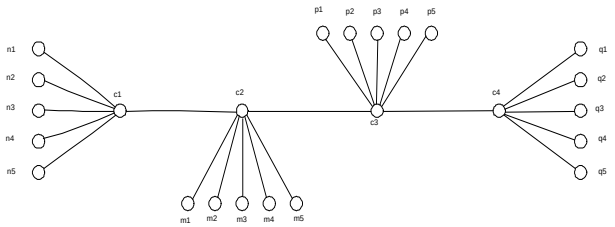

Figure 4.2: Network topology

The normalized Jain's fairness index for vector that represents resource allocations $U = (U_1, \ldots, U_N)$ in the network with max-min fair resource allocation given by vector $U_{mm} = (U_{1,mm}, \ldots, U_{N,mm})$ is given by

$$j(U) = \frac{\left(\sum_{i=1}^{N} \frac{U_i}{U_{i,mm}}\right)^2}{N \sum_{i=1}^{N} \left(\frac{U_i}{U_{i,mm}}\right)^2}.$$

Its values for 7 schemes of interest are following:

$$j(U_{DTail}) = 0.345,\ j(U_{RED}) = 0.731,\ j(U_{MLC(2)}) = 0.846,\ j(U_{MLC(3)}) = 0.884,$$

$$j(U_{MLC(5)}) = 0.956,\ j(U_{MLC(9)}) = 0.989,\ j(U_{MLC(17)}) = 0.997.$$

We can see significant unfairness in the oblivious schemes(DropTail, RED). As we increase index of MLC scheme, we obtain share of bandwidth very close to max-min share as expected by Theorem 4.2.

4.4 Summary

In this chapter we developed an AQM scheme for enforcing max-min fairness in TCP networks called MLC(l). MLC(l) is a stateless scheme and belongs to class of queue management schemes that we call power-drop. We showed analytically that by increasing index l, the resource allocation among TCP users using network of MLC(l) queues converge to max-min fair. Presented analytical findings are confirmed by packet level simulations.

CHAPTER 5

Drop history is enough

Abstract - *Small Flow Completion Time (FCT) of short-lived flows and fair bandwidth allocation of long-lived flows have been two major, sometimes concurrent goals in the design of resource allocation algorithms. In this chapter we present a framework that naturally unifies these two principles under a single umbrella; namely by proposing resource allocation algorithm Markov Active Yield (MAY). By exploiting a probabilistic strategy: "*drop proportional to the amount of past drops*", MAY achieves very small FCT among short-lived TCP flows as well as good fairness among long-lived TCP flows, using only a small amount of on-chip SRAM: roughly 1 bit per flow in Pareto-like flow size distributions. Analytical models are analyzed and results are verified experimentally using packet level ns*2 *simulations.*

5.1 Introduction

Resource allocation in communication networks has been a major topic of interest for some time. Many current proposals have as their performance objective a resource allocation that is max-min fair [110, 18, 100, 55, 3, 120]. However, from the user point of view, a major (and arguably the most important) performance metric is Flow Completion Time (FCT) [24]. A number of schemes that minimizes FCT exist in the literature [93, 4, 24, 42]; either requiring state for every arriving flow or requiring cooperation from end-to-end users. Note that max-min fair proposals can harm FCT of short-lived flows as their sending rate is limited by the fair share determined by the packet scheduler.

Our **goal** is the design of a queue management algorithm that will integrate the design principles stated above: (1) max-min fairness of long-lived TCP flows and (2) small FCT short-lived flows without use of explicit per-flow state.

Our design is based on the following idea *Drop proportional to the amount of past drops.* We propose an implementation called Markov Active Yield (MAY), that satisfies both of our design principles yet is simple to implement, uses first come first served queue, scales for speeds of over

$40Gbps$, and requires roughly one bit per flow of on-chip SRAM in internet-like flow size distributions.

5.1.1 Chapter contributions

Why is reaching the goal stated above hard? First, recall that in the max-min fair regime, a TCP flow f experiences drops at *one and only one* link l_f along its path (we say that f is bottlenecked at l_f), and therefore must be protected at other links (that can be congested) by receiving lossless service. Second, if two or more flows are bottlenecked at the same link they must receive nonuniform loss rates that are functions of their aggressiveness (round-trip times, queuing delays, etc). Assuming that the router has access to the individual flow rates, or the existence of multiple queues that are appropriately scheduled, a number of solutions to these two problems exist, and are described in previous works [110, 55, 88, 18, 100]. However, in our case it is highly nontrivial to make the drop (mark) decision without any explicit information. On the other hand, existing router-based algorithms for minimizing of FCT, require either cooperation from the TCP sender [4], or full state information [93, 42] for *every* arriving flow, making them not scalable at very high speeds.

The main contributions of this chapter are the following:

• A novel queue management scheme Markov Active Yield (MAY) that unifies two major design principles: small FCT of short-lived flows and max-min fair bandwidth sharing of long-lived flows. The *only* state information that is kept by MAY is a short history of *already dropped packets*.

• An analysis of the randomized algorithm MAY that shows that the resource allocation of the long-lived TCP users in the network of MAY queues is max-min fair. On the other hand, nonelastic CBR flows receive (virtually) lossless service if their sending rate is less than max-min fair share, and asymptotic denial of service otherwise. Packet level simulations are presented to support our analytical findings.

• Analysis and evaluation of the effects of MAY on the FCT of short flows. MAY can reduce the FCT of TCP flows significantly compared to oblivious schemes (like RED) and instantaneously-fair packet schedulers (like DRR), and is just slightly higher than those of LAS, a stateful scheme.

While the MAY algorithm is not completely stateless the only state that is kept is frequency of drops. This can significantly reduce memory requirements, in heavy-tailed flow-size/rate environments, which are typical of Internet links. Namely, as we shall see, MAY memory requirements are roughly one bit (of on-chip SRAM[1]) per flow in Pareto-like flow size distributions (see Section 5.3.4). Since the computational complexity of the scheme is extremely low, MAY scales well at speeds greater than 40Gbps and several dozens of millions of concurrent flows, with currently available hardware devices and state-of-the-art implementations of hash tables [101] and statistics counters [106].

[1]Static RAM. Current implementations have access times of 2-6ns.

5.1.2 Related work

Both of the problems we study in this chapter have been studied extensively in the recent literature. Here we classify the existing proposals in two categories: (1) end-to-end transport protocols require modifications to cooperate with the routers; (2) end-to-end transport protocols do not require any modifications (routers are fully responsible for the resource allocation).

(1A) Resource allocation of long-lived end-to-end users that cooperate with intermediate routers has attracted a large attention in the last decade [102]. Examples include XCP [55], REM [3], MaxNet [120], etc.

(1B) The small FCT of short-lived flow has been a subject of interest of several recent studies. In [24] the authors propose an end-to-end protocol for reducing the FCT of short-lived flows. On the other hand the authors of RuN2C [4] propose a two-queue strategy, that cleverly exploits the 32-bit TCP-sequence-number space to reduce the FCT, by requiring a slight change in the TCP stack (the initial TCP-sequence-number must be set to an appropriate value in order to be prioritized by RuN2C).

(2A) Similarly to (1A), enforcing fairness "in the middle" without cooperation of end-to-end users has been a very active research area. Various *stateful* algorithms exist: DRR [100], FQ [18], FRED [73], CSFQ [110], AFD [88], PD-RED [77] etc.

(2B) Size-aware scheduling policies like Shortest Job First (SJF), Shortest Remaining Processing Time (SRPT) or Least Attained Service (LAS) [60] have been used extensively in the area of operating systems to favor short jobs. However, in Internet links, strategies that require knowledge of flow (job) sizes (such as SJF or SRPT) are not suitable, since that information is generally unavailable. On the other hand, stateful approaches have been shown to significantly improve FCT of short lived flows [93, 42].

5.2 Markov Active Yield (MAY)

In order to enforce fairness among long-lived users with different levels of aggressiveness, and responsiveness, it is clear that more aggressive and less responsive users must be punished more than others. The question is "by how much" and how this strategy should be implemented in an efficient manner. We would also like to have some kind of protection for "short-lived" flows. Before we proceed we need to define the notion of short-lived flow:

Definition 5.1 *A flow is called* short-lived *if it contains not more than S_0 packets.*

Ideally, for a given S_0, all short-lived flows should be protected at a congested link, by receiving a lossless service. On the other hand, flows with more than S_0 packets should receive nonuniform loss rates that will result in fair bandwidth share. However, this is hardly feasible at high speeds ($\geq 10Gbps$) since it would require keeping state information for each flow (strictly). The basis for our

discussion is the following idea that can reduce state information needed for by more than two orders of magnitude, with minor performance degradation (in terms of FCT and max-min fairness):

(*) Drop proportionally to the amount of past drops

A queue management scheme that follows (*) is a positive feedback system: the more drops a flow experiences the higher will be the drop probability of that flow in the future. Sometimes, positive feedback systems can be unstable, but self-regulatory nature of TCP makes systems based on (*) stable (in terms of drop probabilities); see Section 5.3.

We will call a flow f new if it has not experienced any loss until time t; otherwise we will say that f is old. A natural question that arises is what happens with new flows under strategy (*). As we said, ideally we would like that all flows with size smaller than S_0 receive a lossless service, and that all flows with size greater than S_0 join the "battle" for fair bandwidth share. However, this requires full-per-flow state information, and we use a probabilistic argument to keep short-lived flows virtually lossless and enforce flows with length greater than S_0 packets to receive most of the losses in congested environments.

To this end, we keep track of all old flows, and drop a new-flow's packets with probability $q_0 = \frac{1}{S_0}$. This will be the input to the positive feedback system defined by (*). As we will see, this stateless strategy for dropping a new-flow's packets, enforces very few losses among flows with size less than S_0. On the other hand, an easily implementable strategy (*) makes bandwidth allocations among arbitrary set of long-lived AIMD flows asymptotically independent of additive increase or multiplicative decrease parameters.

Now we proceed with a detailed description of MAY. As we already said the basic idea is to keep information of recently dropped packets and to use that statistics to regulate the drop probability for each of the flows. Pseudo-code is given in Figure 5.1.

The data structure used is a hash table H that stores quadruples $(FlowID, \delta(FlowID), TS(FlowID), ND(FlowID))$ for flows that have experienced some drops. Here $\delta(FlowID)$ is the drop frequency of the flow given by the identifier $FlowID$, $TS(FlowID)$ is the time stamp that tracks the time of last update of the given hash table entry and $ND(FlowID)$ is the number of dropped packets from the flow $FlowID$ during the current update interval of length Δ. On each packet arrival its $FlowID$ is calculated. If there exists an entry in H that corresponds to $FlowID$, the arriving packet is dropped with probability proportional to the frequency of past drops: $\nu\delta(FlowID)$. The frequency of losses is calculated using weighted averaging (see line 24) with weight q_w. Finally the control variable ν determines the size of drop probabilities and therefore the utilization: if the current utilization is less than a desired level (u_0) ν should be decreased to allow lower drop probabilities and increase utilization, while if current utilization is greater than u_0, then ν should be increased to allow higher drop probabilities and decrease utilization.

The following synthetic example can be helpful in understanding the motivation for MAY in terms of bandwidth allocation.

```
1   OnPacketArrival(pkt, FlowID)
2     if FlowID ∈ H
3       with probability νδ(FlowID) do
4         drop(pkt);
5         ND(FlowID) + +;
6         TS(FlowID) = now;
7       enddo
8     else with probability q_0 do
9       create(FlowID, H)
10      p(FlowID) = 1;
11      ND(FlowID) = 1;
12      TS(FlowID) = now;
13      if CurrentUtilization > u_0;
14        drop(pkt);
15      endif
16    enddo

17  Update(H)
18    if now − LastUpdate > Δ
19      ν = ν + κ(CurrentUtilization − u_0);
20      for all FID in H
21        if now − TS(FID) > T_0
22          remove(FID, H)
23        else
24          δ(FID) = (1 − q_w)δ(FID) + q_w ND(FID);
25          ND(FID) = 0;
26        endelse
27      endfor
28      LastUpdate = now;
29    endif
```

Figure 5.1: Pseudocode of MAY.

u_0	desired utilization
Δ	length of update period
q_w	weighted average parameter
κ	controller gain
T_0	Timeout value

Table 5.1: Parameters of MAY.

Consider a single bottleneck topology, where the congested link services N TCP users with round-trip times $RTT_1, \ldots, RTT_N$. For an AQM scheme Γ, denote by $\sigma(\Gamma)$ the set of serviced packets during the unit of time, and by $\delta(\Gamma)$ the set of dropped packets during the unit of time. If we denote by U_i and p_i the throughput and the loss rate of flow i, then the proportion σ_i of packets from flow i in $\sigma(\Gamma)$ is equal to $U_i \cdot a_1$ and the proportion δ_i of packets from flow i in $\delta(\Gamma)$ is equal to $U_i \cdot p_i \cdot a_2$, for some constants a_1 and a_2 independent of i. For the TCP flow i from the square root formula $U_i = \frac{C_0}{RTT_i \cdot \sqrt{p_i}}$ we conclude that

$$\sigma_i \cdot \delta_i = \frac{A}{RTT_i^2}, \tag{5.1}$$

for some constant A that does not depend on i. For example if Γ is oblivious[2], the loss rate for each flow is constant ($p_i = p$) and $\sigma_i = \frac{A'}{RTT_i}$ and $\delta_i = \frac{A'}{RTT_i}$, for $A' = \frac{1}{\sum_{j=1}^{N} \frac{1}{RTT_j}}$. Consider now the positive feedback system ($\delta(k)$) depicted in Figure 5.2: the dropping probability for the flow i during the time step $t+1$ is proportional to the number of dropped packets during the time step t. If we start with vector $\delta(1) = A'(\frac{1}{RTT_1}, \ldots, \frac{1}{RTT_N})$, at the end of time step 1, $\delta(2)$ will be equal to $(\frac{B(1)}{RTT_i^{3/2}}, \ldots, \frac{B(1)}{RTT_N^{3/2}})$. Actually from the relation (5.1) we can deduce that $\delta(t) = (\frac{B(t)}{RTT_i^{e(t)}}, \ldots, \frac{B(t)}{RTT_N^{e(t)}})$ where sequence $e(t)$ satisfies $e(1) = 1$, and the following recurrence equation

$$e(t+1) = 1 + \frac{e(t)}{2}. \tag{5.2}$$

By taking $\hat{e}(t) = 2 - e(t)$, (5.2) is equivalent to $\hat{e}(t+1) = \frac{\hat{e}(t)}{2}$. Thus $e(t) = 2 - 2^{-(t-1)}$, and from (5.1), $U_i(t) = \frac{C(t)}{(RTT_i)^{2^{-(t-1)}}}$ does not depend on RTT_i for for large t, meaning that asymptotically all flows achieve the same throughput.

5.2.1 Implementation issues

State-of-the-art algorithms for high-speed hash table implementations presented in [101] allow implementations that run on line speeds of 40Gbps (assuming that hash table is stored in on-chip SRAM). In order to keep the whole hash table size small, we use the implementation of statistics counters architecture proposed in [106]. The size of MAY hash table entry can be made to be 8 bytes: 12 bits

[2] An AQM scheme is oblivious if it does not distinguish the packets from different flows. RED, BLUE or PI are examples of oblivious schemes.

```
1   UpdateLossRates()
2     t = 1;
3     for i = 1 : N
4       δ_i(t) = A' / RTT_i;
5     endfor;
6     while forever
7       p_i(t) = ρ δ_i(t);
8       Drop packet from flow i with prob. p_i(t);
9       Control ρ such that utilization is u_0
10      On the end of t-th time step do
11        for i = 1:N
12          δ_i(t + 1) = Number_of_Drops_i(t);
13        endfor;
14        t = t + 1;
15      enddo;
16    endwhile;
```

Figure 5.2: Example: Positive feedback loop for calculating drop probabilities p_i.

per counters $\delta(FlowID)$, $TS(FlowID)$ and $ND(FlowID)$); 12 bits for $FlowID$ fingerprint, and 16 bits for hash-table pointer. This would allow 16K hash-table entries on 1Mbit on-chip SRAM and 128K hash-table entries on 8Mbit SRAM.

5.3 Analysis of MAY

5.3.1 Resource allocation of long-lived elastic AIMD flows

We now prove that in steady-state an arbitrary set of AIMD flows [122] (flows with arbitrary linear increase parameter and arbitrary multiplicative decrease parameter), asymptotically obtain equal amount of throughput in a single bottleneck scenario. Formally, throughout this subsection we assume the following.

Assumption 5.1 *There are N long-lived flows $f_1, \ldots, f_N$ that use a congested link and all of them employ AIMD congestion control algorithms with an additive increase parameter $\gamma_i > 0$ and a multiplicative decrease $\beta_i \in (0,1)$, $i = 1, 2, \ldots, N$.*

Under the previous assumption we can prove that MAY stabilizes ν and $\delta_i = \delta(f_i)$ (frequency of drops of flow f_i) in the model presented below.

Let $U_k^{(i)}$ be the throughput of flow f_i measured after k units of time Δ_0 (say a millisecond; We assume that $\Delta_0 \ll \Delta$). Denote the loss probability for a packet from flow (i) in time step k, between t-th and $t+1$-st hash-table update ($k\Delta_0 \in (t\Delta, (t+1)\Delta)$), by $\mu_i(k) = \delta_i(t)\nu(t)$. Having this, we can consider $U_k^{(i)}$ as a Markov chain on R^+ given by the transition:

$$U_{k+1}^{(i)} = U_k^{(i)} + \gamma_i \quad \text{with probability} \quad e^{-\delta_i(t)\nu(t)U_k^{(i)}}$$
$$U_{k+1}^{(i)} = \beta_i U_k^{(i)} \quad \text{with probability} \quad 1 - e^{-\delta_i(t)\nu(t)U_k^{(i)}}.$$

In the rest of the section we will use these tools to prove convergence of $\delta_i(k)$ and $\nu(k)$ as well as the asymptotic independence of throughput of flow f_i from AIMD parameters γ_i, β_i.

From [25] we have that expected number of sent packets of flow i in time interval $(t\Delta, (t+1)\Delta)$ is

$$\bar{U}^{(i)}(t) = \frac{\beta_i}{1-\beta_i} \frac{\sqrt{\gamma_i}}{\sqrt{\delta_i(t)\nu(t)}} \cdot A \tag{5.3}$$

for a constant A, that does not depend on i ((5.3) is a generalized version of square root formula). Denote by $\bar{\delta}_i(t)$ the number of dropped packets of flow f_i during time interval $(t\Delta, (t+1)\Delta)$. Each of $\bar{U}^{(i)}(t)$ packets is dropped with the same probability $\delta_i(t)\nu(t)$. We use the mean field approximation to estimate $\bar{\delta}_i(t)$:

$$\bar{\delta}_i(t) = \bar{U}^{(i)}(t)\delta_i(t)\nu(t) = \frac{\beta_i}{1-\beta_i} \frac{\sqrt{\gamma_i}}{\sqrt{\delta_i(t)\nu(t)}} \cdot A\delta_i(t)\nu(t) = \frac{\beta_i}{1-\beta_i}\sqrt{\gamma_i} A \sqrt{\delta_i(t)\nu(t)} = a_i\sqrt{\delta_i(t)\nu(t)},$$

(5.4)

where we denoted by $a_i = \frac{\beta_i}{1-\beta_i}\sqrt{\gamma_i}A$. On the other hand, we know that $\nu(t)$ is regulated to achieve utilization u_0C, where C is the link capacity. Thus we have:

$$\sum_{i=1}^{N} \bar{U}^{(i)}(t) = \frac{1}{\sqrt{\nu(t)}} \sum_{i=i}^{N} a_i \frac{1}{\sqrt{\delta_i(t)}} = u_0 C. \tag{5.5}$$

From (5.4) and (5.5), we obtain

$$\bar{\delta}_i(t) = a_i \sqrt{\delta_i(t)} \sum_{i=1}^{N} a_i \frac{1}{\sqrt{\delta_i(t)}} \frac{1}{u_0 C}.$$

Since at the $t+1$-th update we use the weighted averaging: $\delta(t+1) = (1-q_w)\delta(t) + q_w\bar{\delta}_i(t)$, we conclude that the evolution of $\delta(t)$ is given by:

$$\delta_i(t+1) = (1-q_w)\delta_i(t) + q_w a_i \sqrt{\delta_i(t)} \sum_{j=1}^{N} \frac{a_j}{\sqrt{\delta_j(t)}} \frac{1}{u_0 C}. \tag{5.6}$$

Theorem 5.1 *Suppose initially that $\delta_i(1) > 0$ for all $i = 1, \ldots, N$. Then in the mean field model, presented above, the system (5.5)-(5.6) is stable.*

Before we proceed, note that by setting $z_i(t) = \delta_i(t)\frac{u_0 C}{a_i^2}$ system (5.6) becomes equivalent to

$$z_i(t+1) = (1-q_w)z_i(t) + q_w\sqrt{z_i(t)} \sum_{j=1}^{N} \frac{1}{\sqrt{z_j(t)}} \tag{5.7}$$

Without loss of generality, we can order z_i at time $t = 1$: $z_1(1) \le z_2(1) \le \cdots \le z_N(1)$. Directly from (5.7) we get that for all $t \ge 1$

$$z_1(t) \le z_2(t) \le \cdots \le z_N(t).$$

Denote by $r(t) = \frac{z_1(t)}{z_N(t)}$, the ratio between the smallest and the largest component of vector $z(t)$, and by $s(t) = \sum_{j=1}^{N} \frac{1}{\sqrt{z_j(t)}}$. Our first technical lemma shows that $r(t)$ is a nondecreasing function of t.

Lemma 5.1 *Ratio $r(t) = \frac{z_1(t)}{z_N(t)}$ is a nondecreasing function of t.*

Proof: Since the order of z_i is preserved we have that

$$r(t+1) = \frac{z_1(t+1)}{z_N(t+1)} = \frac{(1-q_w)z_1(t) + q_w\sqrt{z_1(t)}s(t)}{(1-q_w)z_N(t) + q_w\sqrt{z_N(t)}s(t)}$$

$$= \frac{z_1(t)}{z_N(t)} \left(\frac{(1-q_w) + q_w\frac{1}{\sqrt{z_1(t)}}s(t)}{(1-q_w) + q_w\frac{1}{\sqrt{z_N(t)}}s(t)} \right) \ge \frac{z_1(t)}{z_N(t)} = r(t)$$

■

Lemma 5.2 *The sequence $z_N(t)$ is bounded from above:*

$$z_N(t) \leq z_N(1) + \frac{N}{\sqrt{r(1)}} =: D.$$

Proof: First, note that $\sqrt{z_N(t)}\sum_{j=1}^{N}\frac{1}{\sqrt{z_j(t)}} < \sqrt{z_N(t)}\frac{N}{\sqrt{z_1(t)}} = \frac{N}{\sqrt{r(t)}} < \frac{N}{\sqrt{r(1)}}$. Now we prove the lemma by mathematical induction. For $t = 1$, statement is clearly true. Suppose that it is valid for $t = m$, then for $t = m + 1$:

$$z_N(m+1) = (1-q_w)z_N(m+1) + \sqrt{z_N(m)}\sum_{j=1}^{N}\frac{1}{\sqrt{z_j(m)}} \leq$$

$$(1-q_w)(z_N(1) + \frac{N}{\sqrt{r(1)}}) + q_w\frac{N}{\sqrt{r(1)}} < z_N(1) + \frac{N}{\sqrt{r(1)}} = D.$$

■

Having that $r(t) \leq 1$ and that $r(t)$ is monotone, nondecreasing, we know that $r(t)$ converges to $r^* \leq 1$. Our next lemma shows that r^* is indeed 1.

Lemma 5.3 *The sequence $r(t)$ converges to 1.*

Proof: Suppose it is not true. Then there exist $\delta > 0$, such that for all t, $r(t) < 1 - \delta$. From the definition of $r(t)$, we have:

$$\begin{aligned}
r(t+1) &= r(t)\left(1 + q_w\frac{(\frac{1}{\sqrt{z_1(t)}} - \frac{1}{\sqrt{z_N(t)}})s(t)}{1 - q_w + q_w\frac{1}{\sqrt{z_N(t)}}s(t)}\right) \\
&= r(t)\left(1 + q_w\frac{\frac{1}{\sqrt{z_N(t)}}(\frac{1}{\sqrt{r(t)}} - 1)s(t)}{1 - q_w + q_w\frac{1}{\sqrt{z_N(t)}}s(t)}\right) \\
&= r(t)\left(1 + q_w\frac{(\frac{1}{\sqrt{r(t)}} - 1)}{\frac{(1-q_w)\sqrt{z_N(t)}}{s(t)} + q_w}\right) \\
&> r(t)\left(1 + q_w\frac{(\frac{1}{\sqrt{r(t)}} - 1)}{(1-q_w)z_N(t) + q_w}\right) \\
&> r(t)\left(1 + q_w\frac{(\frac{1}{\sqrt{1-\delta}} - 1)}{(1-q_w)D + q_w}\right) = r(t)E,
\end{aligned}$$

as the constant E is strictly greater than 1. Thus, assuming that $r(t) < 1 - \delta$, for all t, implies $r(t) > r(1)E^{t-1} \to \infty$ which is in turn a contradiction with $r(t) \leq 1$. ■

Proof: (Theorem 5.1) Since $z_1(t)/z_N(t)$ converges to 1, we have that for every i, $\sqrt{z_i(t)}s(t)$ converges to N, and therefore, from (5.7) $z_i(t)$ converges to N. Now, from the definition of $z_i(t)$ we obtain that

$$\lim_{t\to\infty}\delta_i(t) = N\frac{a_i^2}{u_0 C} = \delta_i^*.$$

From (5.5), we conclude that

$$\lim_{t\to\infty}\nu(t) = \lim_{t\to\infty}\left(\frac{1}{u_0 C}\sum_{i=i}^{N} a_i\frac{1}{\sqrt{\delta_i(t)}}\right)^2 = \frac{N}{u_0 C} = \nu^*.$$

■

Corollary 5.1 *In steady state, the expected throughput of flow f_i does not depend on AIMD parameters, β_i and γ_i.*

Proof: From (5.3),

$$\bar{U}^{(i)}(t) = \frac{\beta_i}{1-\beta_i}\frac{\gamma_i}{\sqrt{\delta_i^*\nu^*}} \cdot A = \frac{a_i}{\sqrt{N\frac{a_i^2}{u_0C}\frac{N}{u_0C}}} = \frac{u_0C}{N}$$

■

5.3.2 Resource allocation of non-responsive CBR flows

In the previous subsection we have seen that a set of N long-lived elastic AIMD flows bottlenecked at link with throughput u_0C with MAY queue receive asymptotically the same throughput u_0C/N. In this subsection we are going to explore the effect of MAY on the throughput of non-responsive CBR flows. Suppose that a link is shared by N elastic AIMD flows with AIMD parameters $\gamma_i > 0$ and $\beta_i \in (0,1)$ and M non-responsive CBR flows with constant sending rate of x_j packets per δ.

Theorem 5.2 *In steady-state, the throughput of all AIMD flows is*

$$U_i^* = \frac{1}{\nu^*}. \tag{5.8}$$

If $\delta_j(1)\nu^* \geq 1$ the throughput of CBR flow T_j^* is zero; otherwise:

$$T_j^* = x_j \quad if \quad x_j \leq \frac{1}{\nu^*}, \tag{5.9}$$

$$T_j^* = 0 \quad if \quad x_j > \frac{1}{\nu^*}. \tag{5.10}$$

Here ν^* is the steady-state value of ν.

Proof: From the proof of the Theorem 5.1 we have (5.8). On the other hand denote by $\delta_j(t)$ the weighted average of frequencies of drops from CBR flow c_j, and by $\bar{\delta}_j(t)$ the amount of drops from the same flow during the time step t. Then if $\delta_j(t_0)\nu^* \geq 1$ for some t_0, all packets starting after period t_0 will be dropped and $T_j^* = 0$. If $\delta_j(t)\nu^* < 1$ then

$$\delta_j(t+1) = (1-q_w)\delta_j(t) + q_w\bar{\delta}_j(t) =$$

$$= (1-q_w)\delta_j(t) + q_w x_j \delta_j(t)\nu^* = \delta_j(t)((1-q_w) + q_w x_j \nu^*)$$

Thus, for $x_j > \frac{1}{\nu^*}$, we have that $((1-q_w) + q_w x_j \nu^*) > 1$ and $\delta_j(t+1)$ is exponentially increasing until it becomes greater than $\frac{1}{\nu^*}$, and after that all packets from that flow are dropped. If $x_j < \frac{1}{\nu^*}$, $\delta_j(t+1) \to 0$ as $t \to \infty$, and therefore CBR flow g_j receives asymptotically lossless service, and $T_j^* = x_j$.

■

5.3.3 Max-min fairness of the elastic AIMD flows

The following lemma is a characterization of max-min fairness [102].

Lemma 5.4 *A set of rates x_r is max-min fair if and only if for every flow r there exists a link l on its path, such that the rates of all flows which traverse through l are less or equal than x_r.*

The next theorem proves that in steady-state resource allocations of elastic AIMD users is max-min fair.

Theorem 5.3 *Let G be a network topology, with queues employing MAY and end users employing AIMD congestion control. For every flow r with steady-state rate x_r^* there exists a link on its path, such that the steady-state rates of all flows which traverse through that link are less then x_r^*.*

Proof: Let L be the number of links in the network and N be the number of flows. We label flows by $i = 1, 2, \ldots, N$ and links by $s = 1, 2, \ldots, L$. By R we denote the routing matrix: $R_{is} = 1$ if flow i uses link s otherwise $R_{is} = 0$. By $\delta_i^{(s)*}$ denote the frequency of drops of flow i at link s, and by ν_s^* steady-state value of ν at link s. Then for an arbitrary flow r there must exist a link l for which $\delta_r^{(l)*} > 0$. On the other hand, from the definition of $\delta_r^{(s)*}$ we have that it satisfies the following equation.

$$\delta_r^{(s)*} = (1 - q_w)\delta_r^{(s)*} + q_w x_r^* \nu_s^* \delta_r^{(s)*} = \delta_r^{(s)*}(1 - q_w + q_w x_r^* \nu_s^*). \tag{5.11}$$

Since we picked l such that $\delta_r^{(l)*} > 0$, from (5.11) we conclude that $1 - q_w + q_w x_r^* \nu_l^* = 1$, which implies $x_r^* = \frac{1}{\nu_l^*}$. Now, for any other flow r_1 that uses link l, $\delta_{r_1}^{(s)*} > 0$ implies $x_{r_1}^* = \frac{1}{\nu_l^*} = x_r^*$ while $\delta_{r_1}^{(s)*} = 0$ implies $x_{r_1}^* < \frac{1}{\nu_l^*}$ (see proof of Theorem 5.2). ■

5.3.4 Memory consumption

Let the sizes of flows be distributed according to the distribution $G(x)$:

$$P[|f| \geq x] = G(x) = \int_x^\infty g(y)dy.$$

Then the number of hash-table entries needed for a population of n flows: $(f_i)_{1 \leq i \leq k}$ is given by the random variable

$$Z(n) = \sum_{i=1}^{n} z(f_i),$$

where:

$z(f_i) = 1$ if flow f_i has been hashed (experienced at least one loss)

$z(f_i) = 0$ if flow f_i has not been hashed (had no losses)

Denote by l_i the length of flow f_i. The probability that $z(f_i) = 0$ is:

$$P[z(f_i) = 0] = (1 - q_0)^{l_i} = (1 - \frac{1}{S_0})^{l_i} \approx e^{-\frac{l_i}{S_0}}.$$

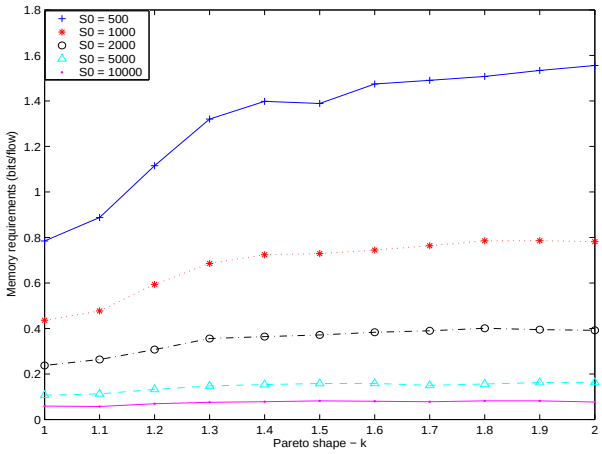

Figure 5.3: Expected memory consumption (in bits per flow). Pareto shape $k \in [1,2]$, mean $\mu = 10$, $500 \leq S_0 \leq 10000$

The expected memory consumption is then given by

$$E(Z(n)) = nE(z(f)) = n \int_0^\infty g(y)(1 - e^{-\frac{y}{S_0}})dy,$$

and the variance of $Z(k)$ is:

$$Var(Z(n)) = nVar(z(f)) = n \int_0^\infty g(y)(1 - e^{-\frac{y}{S_0}})e^{-\frac{y}{S_0}}\, dy.$$

The expected memory consumption (EMC) in bits per flow is given by $M(G) = 64E(Z(n))/n$ (we assume that each hash-table entry have $8bytes = 64bits$). Figure 5.3 contains EMC for Pareto distributions, with Pareto shape in $[1,2]$ and mean 10 packets.

5.3.5 FCT of short-lived TCP flows

A major factor that determines the FCT of a flow f is the congestion control algorithm (and its implementation) used by f. For example in standard TCP, the parameters that directly determines FCT are: the slow start threshold ($sstresh\_$), the advertised window, the maximum congestion window ($maxcwnd\_$), the delayed acknowledgements option, etc. In the simplest case (without $sstresh\_$ and $maxcwnd\_$ limitations), most of short flows will complete quickly (in slow start phase), if they do not experience loss. In Pareto flow size environments, the proportion of short-lived flows that experience loss is very small. Namely, in a population of k flows with Pareto distributed sizes, with shape $k \in [1,2]$, and mean size μ the proportion of short-lived flows that do not experience drops is

$$L(k, S_0) = Prob[z(f) = 0| \quad |f| < S_0].$$

The value of $L(k, S_0)$ is depicted in Figure 5.4.

To see how the FCT of TCP flows is affected by MAY one can consider a simple model of TCP with delayed acknowledgements (one ack per two packets) and without (upper) limitation in $cwnd\_$.

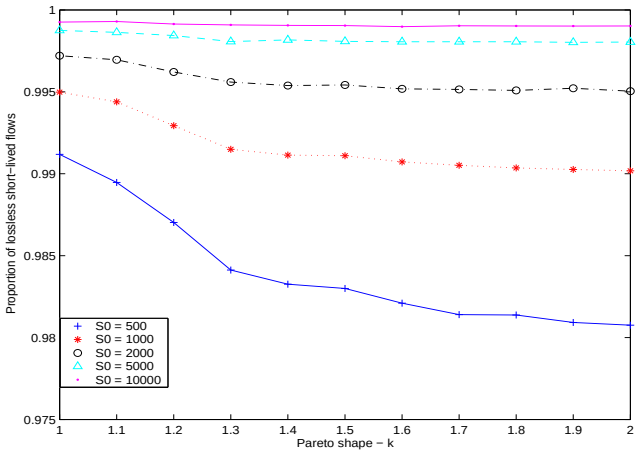

Figure 5.4: Proportion of lossless short-lived flows. Pareto shape $k \in [1, 2]$, mean $\mu = 10$, $500 \leq S_0 \leq 10000$.

Congestion window's ($cwnd\_$) evolution is given by the following equations, where updates are taken once per RTT.

If $rand(0, 1) < e^{-q_0 cwnd\_}$

$cwnd\_ = \frac{3}{2} cwnd\_$ if $dropped == false$ and $cwnd\_ < ssthresh\_$

$cwnd\_ = cwnd\_ + \frac{1}{2}$ otherwise

otherwise

$dropped = true$

$cwnd\_ = \frac{cwnd\_}{2}$

Figure 5.5 depicts the mean FCT of short-lived TCP flows ($S_0 = 1000$) in three cases LAS, MAY and FIFO with drop rate $p_0 = 0.01$. Both $ssthresh\_ = 2$ and $ssthresh\_ = \infty$ are included. We can observe a slight increase in FCT between MAY and LAS, which is the consequence of the (stateless) probabilistic nature of MAY and price that must be payed for not keeping per-flow packet counters as in LAS.

5.4 Experimental results

In this section we briefly describe results of packet level $ns2$ simulations that demonstrate the behavior of MAY. We look at two issues:

- fairness of long-lived flows
- flow completion times of short-lived flows

To compare how far bandwidth allocations $U = (U_1, \ldots, U_N)$ deviate from max-min fair bandwidth allocations, $U_{mm} = (U_{1,mm}, \ldots, U_{N,mm})$, we use Jain's fairness index [50] given by:

$$j(U) = \frac{\left(\sum_{i=1}^{N} \frac{U_i}{U_{i,mm}}\right)^2}{N \sum_{i=1}^{N} \left(\frac{U_i}{U_{i,mm}}\right)^2}. \tag{5.12}$$

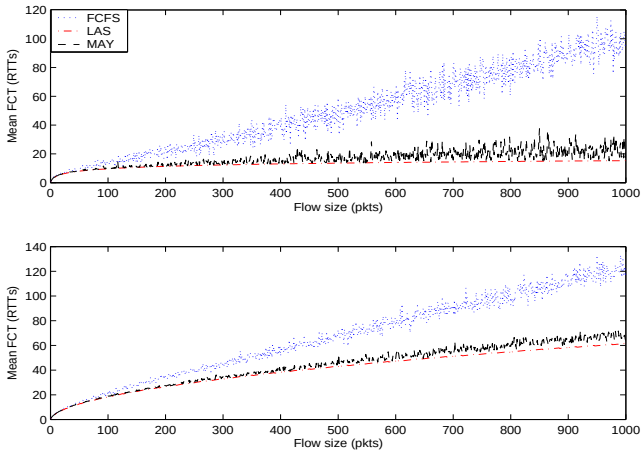

Figure 5.5: Flow sizes versus FCT (measured in RTTs) in FIFO (with drop rate $p_0 = 0.01$), MAY ($S_0 = 1000$) and LAS cases. Top plot has no slow start limitations, while the bottom plot corresponds to $sstrhesh\_ = 2$.

Clearly, $j(U)$ has a global maximum 1 that is attained at $U = U_{mm}$ and since it is a continuous function, by measuring how far the index is from 1, one can get some intuition as to how far the vector U is from U_{mm}. The MAY parameters used in the experiments are: $\Delta = 1s$, $u_0 = 0.98$, $\kappa = 0.1$, $q_w = 0.05$, $T_0 = 64sec$. The DRR parameters are $No\_buckets\_ = 100$, $blimit\_ = 100Kbytes$, $quantum\_ = 1000bytes$. The self configuring Adaptive RED [39] is used to determine the RED parameters.

5.4.1 Fairness - Single bottleneck

The first set of simulations are designed to demonstrate the fairness properties of the proposed AQM schemes in single bottleneck scenario. Specifically, we present results for a single link with service rate of $80Mbps$ that services 100 long-lived TCP users with round trip times uniformly distributed in range $40 - 440ms$, with a packet size $1000bytes$. To provide baseline results, we include the performance of RED and DDR for the same scenario. The share of the total throughput taken by each of 100 flows in these 3 schemes is depicted in Figure 5.6.

Jain's fairness indices for these three schemes are:

$$j(U_{RED}) = 0.606, \quad j(U_{DRR}) = 0.997, \quad j(U_{MAY}) = 0.993.$$

It can be seen from Figure 5.6 that the fairness of RED is approximately proportional to the inverse of RTT. This is in accordance with square root formula and observations made by others [1, 38].

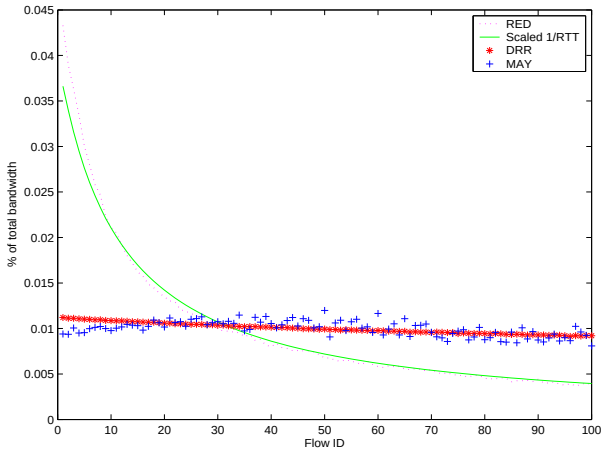

Figure 5.6: Scaled throughput for 100 long-lived TCP flows over congested link employing RED, DRR and MAY.

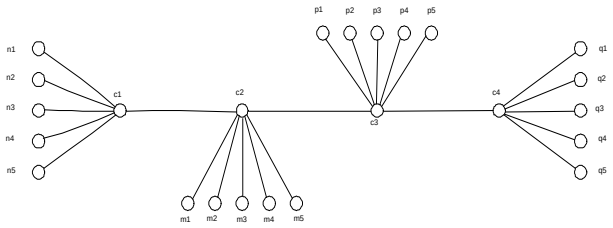

Figure 5.7: Network topology

5.4.2 Fairness - Multiple bottleneck topology

Our second set of simulations demonstrate the fairness properties of MAY in network with multiple bottlenecks. The network topology that we considered is given in Figure 5.7. Here, we consider a network of 24 nodes: $n1-n5, m1-m5, p1-p5, q1-q5$, and $c1$, $c2$, $c3$, $c4$ and 30 flows traversing the network as follows: $n(i) \rightarrow p(i); n(i) \rightarrow q(i), m(i) \rightarrow p(i); m(i) \rightarrow q(i); n(i) \rightarrow m(i); p(i) \rightarrow q(i)$ where $i = 1, 2, 3, 4, 5$.

The delays on each of the links in ms are defined as follows:

$$ni \rightarrow c1 \ : \ 40 \cdot i + 1; \quad pi \rightarrow c3 \ : \ 40 \cdot i + 1$$

$$mi \rightarrow c2 \ : \ 40 \cdot i + 1; \quad qi \rightarrow c4 \ : \ 40 \cdot i + 1$$

and the delays $c1 - c2$, $c2 - c3$, $c3 - c4$ are $10ms$. The capacities of all links are $10Mbps$. With this topology, the max-min fair shares are $0.5Mbps$ for the 20 flows that use link $c2 - c3$, and $1Mbps$ for the other 10 flows ($n(i) \rightarrow m(i)$ and $p(i) \rightarrow q(i)$).

Each flow uses the standard TCP-SACK algorithm, with a packet size 1000B. The aggressiveness of each flow is mainly determined by its RTT. The behavior of the network is evaluated with each link $c1 - c2$, $c2 - c3$ and $c3 - c4$ using: RED, DRR and MAY with a queue size of 100 packets. The MAY parameters used in the experiment are same as in previous section.

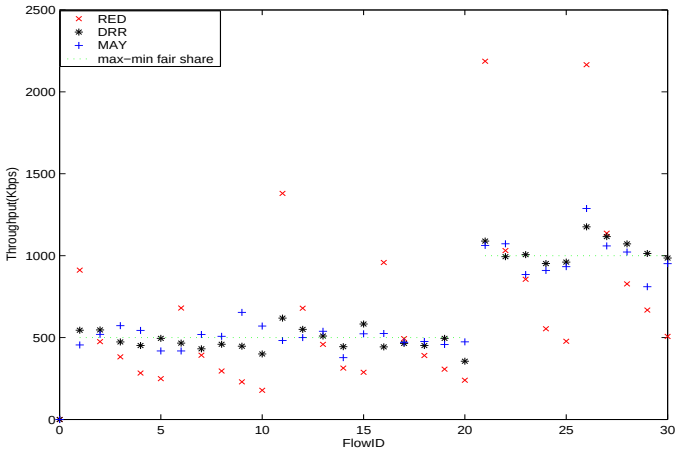

Figure 5.8: Bandwidth taken by each of 30 flows in multiple bottleneck topology. Congested links use RED, DRR, MAY.

Figure 5.8 depicts 3 scenarios, one for each dropping scheme used. We plot the amount of throughput taken by each of 30 flows. The doted line represents the max-min fair share of bandwidth. We can see significant unfairness in oblivious (RED) scheme and close-to-max-min-fair resource allocation in DRR and MAY case. Jain's indices, defined by (9.17) are:

$$j(U_{RED}) = 0.731,\ j(U_{DRR}) = 0.987,\ j(U_{MAY}) = 0.985$$

5.4.3 Throughput of a nonelastic flow

In this subsection we present simulations that support analytical findings from Theorem 5.2. The basic setup is the following: $N = 19$ TCP flows with RTTs uniformly distributed in $[20ms, 220ms]$ share a MAY (the MAY parameters used are the same as in previous subsection) link with capacity $40Mbps$ with a single CBR flow with sending rate x_1. We varied x_1 in the interval $[0.5Mbps, 8Mbps]$ and evaluate the throughput in each case. The results are presented in Figure 5.9. Each simulation lasted for 5 minutes, with first minute neglected. In this context, fair share is simply $x_f = 40/20Mbps = 2Mbps$. From Figure 5.9, we can see that for x_1 less than $x_f = 2Mbps$ throughout is identical to the sending rate, while for $x_1 > x_f$ CBR flow is shut down, by receiving (almost) zero throughput.

5.4.4 Flow sizes versus FCT of short-lived flows

In this subsection we present results that show how the FCT of short-lived flows (here we use $S_0 = 1000$) is affected by the queue management algorithm used at a congested link. The basic setup is the following: 250 short-lived TCP flows share a $40Mbps$ bottleneck link with 50 long-lived TCP flows. Each connection has no $sthresh\_$ nor $cwnd\_$ limitations, delayed acknowledgements are switched on and packet sizes are $1000bytes$. RTT's of short-lived flows are $100ms$ (we use the same value in order to compare the FCTs of different flows, which are RTT-dependent), and RTT's of long-lived

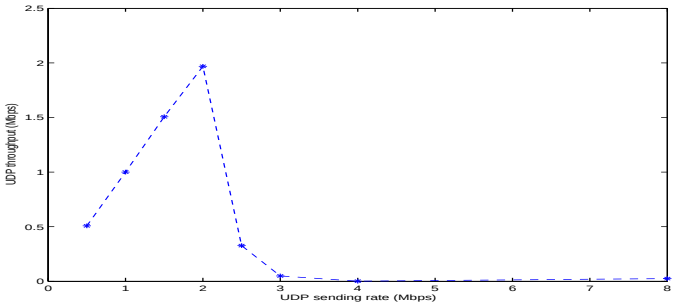

Figure 5.9: Sending rate versus Throughput of the nonelastic CBR flow.

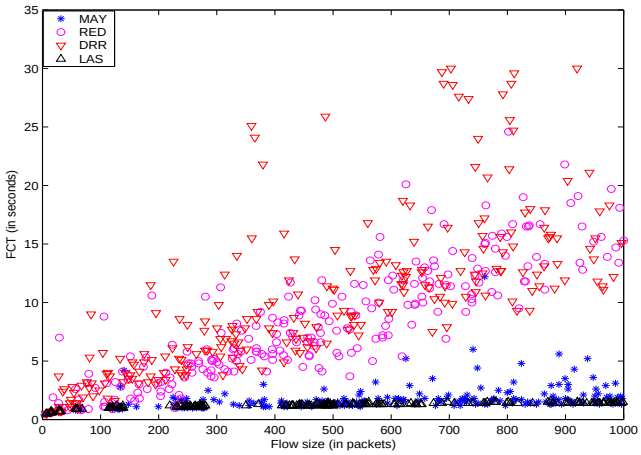

Figure 5.10: Flow sizes versus FCT (measured in seconds) in RED, DRR, MAY and LAS cases.

flows are uniformly distributed in $[20ms, 200ms]$. The sizes of short-lived flows are picked randomly with uniform distribution in the interval [1,1000] packets. We evaluated the FCT of short lived flows obtained by four different queue management algorithms used by the bottleneck link: RED, DRR, LAS and MAY. The results are depicted in figure 5.10. Numerically, the average FCT (AFCT) for 250 short-lived flows in this four cases are:

$$AFCT_{RED} = 8.14s, \;\; AFCT_{DRR} = 10.34s,$$

$$AFCT_{LAS} = 1.26s, \;\; AFCT_{MAY} = 1.72s.$$

Note the slight increase of FCT in MAY compared to stateful scheme - LAS. The oblivious scheme (RED) and instantaneously fair packet scheduler (DRR) achieve significantly larger FCT. In RED no packet is prioritized while in DRR the instantaneous sending rate is limited and most of short-lived flows experience loss during the slow-start phase.

5.5 Discussion of MAY

Isolating the high-rate nonelastic flow. Nonelastic flows that have sending rate higher than responsive flows bottlenecked at a given link would asymptotically get full denial of service by MAY. One may

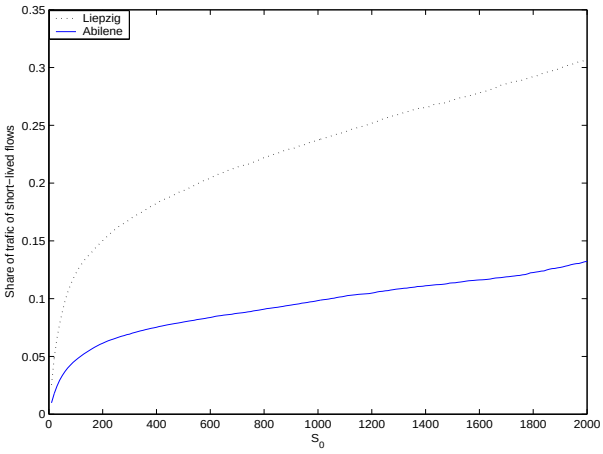

Figure 5.11: Share of traffic produced by short-lived flows (those with less than S_0 packets) as function of S_0. Real Internet traces: Leipzig and Abilene.

argue that this is not desirable feature of MAY, but since the nonelastic flows (with high sending rate) have few orders of magnitude higher drop rates compared to elastic flows they can be easily identified and can receive additional processing by the router if required.

Estimating the fair share and the number of bottlenecked flows. From the analysis in section 5.3 we know that max-min fair share x^* can be estimated with

$$x^* = \frac{1}{\nu^*}.$$

From the proof of Theorem 5.3, we also know that a flow f is bottlenecked at the link l if and only if δ_f^* is asymptotically positive. In practice, we can estimate the number of bottlenecked flows at link l at time t as

$$NB(l) = \#\{f \in H : \delta_f(t) > \epsilon\}$$

for some $\epsilon > 0$.

Variable packet size. Throughout this chapter we assumed uniform packet sizes among the flows to ease the exposition. The problem of variable packet size can be easily solved by employing byte-based variable instead packet-based variables.

Control of ν. In the present implementation the control of ν is rate based, rather than queue-based. This means that congestion indicator is arrival rate at the queue, rather than queue length. Queue based control of ν is possible as well, without affecting the main features of MAY.

Definition of short-lived flow. In our terminology a flow is short-lived if it has length not greater than S_0 packets. The quantity S_0 determines memory consumption as well as the amount of traffic prioritized. The Figure 5.11 depicts the proportion of short-lived-flow traffic as function of S_0 at two real internet traces [84]: Leipzig and Abilene. We can see, for example, that picking $S_0 = 1000$ corresponds to 10% of traffic in the Abilene trace and 24% in the Leipzig trace.

Parameter calibration. Initial tests show that MAY is highly robust to the choice of parameters. The update interval Δ should be taken to cover several "typical" RTT, thus to belong to interval

$[500, 5000]ms$. The weighted average q_w should be chosen to allow averaging over several update intervals: $q_w \in [0.01, 0.1]$. Timeout T_0 depends on the definition of persistence of a flow. The standard value used in the Internet measurements if $T_0 = 64sec$ and here we use the same value. Self tuning of parameters is possible but is out of scope of the present chapter.

5.6 Summary

In this chapter we proposed a randomized framework for unifying two major principles for the design of resource allocation algorithms at Internet links: small Flow Completion Time of short-lived flows and max-min fair bandwidth sharing of long-lived flows. While both problems have been extensively studied, over last two decades, no stateless solution exist for either of them. A powerful paradigm "Drop proportionally to the amount of past drops", allows us to develop a virtually stateless scheme, called MAY which unifies two design principles using around 1 bit per flow of on-chip SRAM in Internet-like flow size distributions. The light computational complexity allows implementation at speeds greater than 40Gbps.

Our analytical work, shows that for elastic AIMD flows, the amount of bandwidth obtained by each long-lived flow asymptotically does not depend on its aggressiveness (additive increase factor) nor responsiveness (multiplicative decrease factor). The fairness level of MAY, evaluated using Jain's fairness index, is almost identical to one of DRR. On the other hand, increase of FCT for short-lived flows compared to LAS is minimal ($< 10\%$) and is a consequence of probabilistic nature of our scheme, and price payed for avoidance of state information.

CHAPTER 6

Small active counters

Abstract - *The need for efficient counter architectures has arisen for the following two reasons. Firstly, a number of data streaming algorithms and network management applications require a large number of counters in order to identify important traffic characteristics. And secondly, at high speeds, current memory devices have significant limitations in terms of speed (DRAM) and size (SRAM). For some applications no information on counters is needed on a per-packet basis, and several methods have been proposed to handle this problem with low SRAM memory requirements. However, for a number of applications it is essential to have the counter information on every packet arrival. In this chapter we propose two, computationally and memory efficient, randomized algorithms for approximating the counter values. We prove that the proposed estimators are unbiased and also derive variance bounds. A case study of Multistage Filters (MSF) on real Internet traces shows a significant improvement by using the active counters architecture.*

6.1 Introduction

Statistics counters are an essential element of most data streaming or sampling algorithms. Maintaining them at high line speeds is a challenging research problem. Briefly, we need an architecture (1) to store a large number of counters (say millions) and (2) to update a large number (say tens of millions) of counters per second. While chip and large Dynamic RAM (DRAM) can easily satisfy condition (1), DRAM access times of $50-100ns$ cannot accommodate the large number of counter-updates needed[1]. On the other hand, expensive Static RAM (SRAM) with access times of $2-6ns$ allow much more counter increments per second, but are not economical for large number of counters. Recently, several solutions [94, 127, 95, 96] have been proposed that use a hybrid SRAM/DRAM architecture: for each counter m-bit counter ($m \ll 64$) is stored in SRAM, while a full counter is stored in DRAM. SRAM

[1]Moreover one cannot expect in the near future that the speeds of DRAM (which increase approximately 7% per year) will reach the speeds of backbone links (which roughly double every 18 months)[30, 2].

counters are updated on a per-packet basis and when some of them come close to overflow different Counter Management Algorithms (CMA) decide which counters should be flushed to DRAM. These approaches share a common feature: it is not possible to estimate a counter's value without accessing DRAM. We use the term *Passive Counters* to refer to such schemes. For a number of applications it is essential to have an estimate of the counter on every packet arrival. For example, in Multistage Filters (MSF) [30], the conservative update step requires knowledge of counters at each stage, and is essential for good performance of MSF: [30] reports up to 100 times higher false-positive ratio without conservative updates. Similarly, in Smart Sampling [22], the sampling probability depends on the current estimate of the flow size. The algorithm for building a Spectral Bloom Filter [15], the algorithm for online hierarchical-heavy-hitter identification [124], as well as efficient implementations of hash tables [101] also require volume estimates on per-packet basis. We will use term *Active Counters* for schemes that allow estimate of full counters on per-packet basis without DRAM access. Thus active counters have to store enough information in fast memory. Storing the full-width counter in SRAM is an example of an active counter. At high-speeds these counters would require up to 64 bits to prevent overflow. The main objective of this chapter is the design of a small active counter architecture with small errors between the small-counter-based estimate and the real volume. Having efficient small active counters would reduce SRAM space needed for a number of applications; or equivalently, for given SRAM space one can fit more small active counters than regular 32(or 64)-bit counters, which in turn can significantly improve performance of data streaming algorithms with limited memory.

6.1.1 Related work

The design of an efficient statistics counter architecture has been identified as an important problem by Shah et al. [96]. They propose a hybrid SRAM/DRAM architecture in which DRAM is used to store the statistics counter while a small SRAM is used to enable counter updates at line rate. In other words, suppose that we need to track n counters of size M bits, then n counters of full size M bits are stored in DRAM and n counters of a smaller size $m < M$ bits are cached in SRAM. The counters in SRAM are updated on every packet arrival to keep track of the recent history of counter updates and are periodically flushed to corresponding DRAM. The decision as to which counter should be flushed to DRAM is made by a Counter Management Algorithm (CMA). By maintaining smaller SRAM counters required SRAM space is reduced, while larger access times of DRAM are compromised by accessing DRAM not too often. The size of SRAM counters, as well as the frequency of DRAM access is determined by CMA. CMA decides which counters should be updated first. Different proposals have different CMAs. The following features are shared by previous proposals:

- *full counters are stored in slow DRAM.*

- *counters stored in DRAM are exact.*

In order to provide the exact values of DRAM counters CMA must ensure that no SRAM counter overflows before updating to DRAM.

Shah et al. [96] propose a CMA called Largest Counter First (LCF). In LCF, an SRAM counter with the largest value is flushed to DRAM. LCF is difficult to implement at high speeds for the following two reasons. First, LCF requires a large amount of control memory: as LCF SRAM counters are 9 bits large control memory can take up to 20 bits per counter. Second, LCF requires finding the largest of n counter values which is the main performance bottleneck of LCF.

In order to reduce the control memory consumption of LCF and expensive searching for the largest counter value, Ramabhadran and Varghese [94] proposed a new CMA called Largest Recent with threshold T, $LR(T)$. $LR(T)$ tracks the list of counters with values greater than the threshold T and makes the decision as to which SRAM counter should be flushed to DRAM using only this list. By doing this, the computational complexity of LCF is significantly reduced, while the usage of the efficient data structure called aggregated bitmap requires only 2 bits per counter devoted for the control memory. Compared to LCF (that consumes 29 bits per counter) $LR(T)$ requires only 11 bits per counter.

Roeder and Lin [95] developed an extension to LCF and $LR(T)$ to reduce the memory requirements for nonuniform traffic patterns. Namely, if the frequency of updating fast memory counters is not uniform over counters, then allocating more bits to SRAM counters with more frequent updates would reduce the number of DRAM accesses; therefore the amount of fast memory space needed for storing recent history counters can be reduced. They propose multiple levels of fast memory instead of one: one level of SRAM and one or more levels of CAM (Content Addressable Memory) for storing the partial counter values. Their enhancement can reduce the amount of equivalent fast memory storage up to 28%.

A recent paper by Zhao et al. [127] removed the basic design objective of previous approaches that does not allow SRAM-counter overflows. The key observation is, assuming that SRAM counters are large enough, counter overflows occur rarely enough so that a small buffer for storing this overflowed counters would be sufficient to allow the exact tracking of DRAM counters with extremely low probability of data loss[2]. The implementation of their proposal requires $l(\mu) = \lceil \log_2(\frac{1}{\mu}) \rceil$ bits for SRAM counters, and $\epsilon < 1$ (say 0.01) bits per counter for buffer that keep track of overflowed counters. Here μ represents the ratio between the access times of SRAM and DRAM. For currently available SRAM/DRAM devices, μ is in the range $[1/50, 1/10]$ which implies $l(\mu) \in \{4, 5, 6\}$. We believe that the multilevel SRAM/CAM/DRAM architecture, similar to those proposed in [95], can further reduce equivalent fast memory storage.

[2]To quote [127] "However, in practice there is no need to worry about this probability, since it can be made so small that even if router operates continuously for billions of years (say from Big Bang to now)), the probability that a single loss of increment happens is less than 1 over a billion".

6.1.2 Chapter contributions

The current proposals for efficient counter statistics architectures belong to class that we call passive counters: full counter information is stored in DRAM, and it is not possible to estimate the full counter value without accessing DRAM. While for many applications passive counters are good enough [63, 66], for a number of applications full counter value, or its estimate, is required on per-packet basis [30, 15, 22, 124]. In these cases, at high speed links (say 10Gbps or more), it is necessary to have enough information for estimation of full counter based on information stored in the fast memory. The design of efficient active counter architecture is the main concern of this chapter.

Briefly, the main contributions of this chapter are the following:

• Two active counter architecture schemes, SAC and HAC, are proposed that have small per counter memory requirements, as well as a small expected error.

• A proof of unbiasedness of presented estimators and the analysis of the expected error.

• A framework for the resource control of SAC and HAC, and a demonstration of the potential benefit of proposed solutions is presented in a case study done with Multistage filters over real Internet traces.

Both of our schemes are randomized and therefore give the estimate of total traffic rather than exact values. Our first scheme called SAC uses only SRAM memory, with m bits per counter devoted for tracking values in range $[0, 2^M]$ where $M > m$. The standard error of this scheme is proportional to $\frac{1}{\sqrt{2^k+2^{k-r}}}$, for some $r < k < m$. In order to reduce the error we developed the second scheme called HAC which uses hybrid SRAM/DRAM architecture. In HAC the main counter information is stored in SRAM, while DRAM is used for storing augmenting counters which are used to reduce the standard error. Our analysis shows that if η is the relative frequency of accessing of DRAM, and $\hat{c}$ the estimate of the counter, then the standard error of HAC is proportional to $\frac{\eta}{\sqrt{\hat{c}}}$ (which can be substantially smaller for large counter values).

We argue that in applications of statistics counters for firewall support, intrusion detection, performance monitoring, flow control, packet schedulers, load balancing or traffic engineering, no need for exact counters exist. We believe that small errors (of say 1%) are acceptable, so that all reasons for design of existing passive counters apply to active counter architecture as well. We also believe that storing counter information in on-chip SRAM can be beneficial for remote monitoring/control as it would reduce latency of SMNP queries at heavily utilized network processors that usually have low priority (and therefore high latency).

Throughout this chapter we use the following notation. For a real number x: by $\lfloor x \rfloor$ we denote the largest integer not greater than x by $\lceil x \rceil$ we denote the smallest integer not less than x and by $\{x\} = x - \lfloor x \rfloor$ we denote the residuum of x modulo 1.

6.2 Simple Active Counters (SAC) scheme

Suppose that we have $q \cdot n$ bits of SRAM allocated for n counters. Let V_i be real value of counter i. Our goal is to develop scheme that estimates V_i using only q bits of SRAM memory. Our scheme works as follows. We divide the available q bits of the i-th SRAM counter in two parts: l-bit exponent part that we call $mode[i]$, and the k-bit estimation part $A[i]$. We use the estimator

$$\hat{V}_i = A[i] \cdot 2^{r \cdot mode[i]} \tag{6.1}$$

Here r is the global parameter, and is determined by scale over which this set of counters run. To update $A[i]$ we use a randomized scheme. Suppose that we need to increment the counter i by value inc. If $inc \geq 2^{r \cdot mode[i]}$ then we first increment $A[i]$ by $\lfloor \frac{inc}{2^{r \cdot mode[i]}} \rfloor$ and then increment $A[i]$ by 1 with probability given by normalized value of the residue: $\frac{1}{2^{r \cdot mode[i]}}(inc - \lfloor \frac{inc}{2^{r \cdot mode[i]}} \rfloor \cdot 2^{r \cdot mode[i]})$. Similarly, for $inc < 2^{r \cdot mode[i]}$ we increment $A[i]$ by 1 with probability $\frac{inc}{2^{r \cdot mode[i]}}$.

The pseudocode of the scheme is given in Figure 6.1. A renormalization step moves counters to a higher scale: $r = r + 1$.

Initially, we divide the available memory of q bits in two parts of size l and k such that $l + k = q$. How to chose "right" allocation pair (k, l) depends on various factors, and we will discuss this in detail in Section 6.5.

The analysis from Section 6.4 shows that the standard error of the scheme is approximately proportional to $\frac{1}{\sqrt{2^k + 2^{k-r}}}$. This error can be substantial for very small k. Intuitively, for large counters, the frequency of updates is very small, which implies the large variance of the background estimator. In the next section we present the architecture that will use augmenting DRAM counters to reduce the variance of estimators.

6.3 Hybrid Active Counters (HAC) scheme

Suppose again that we have q bits per counter in SRAM, divided in two parts with sizes of k ($A[i]$) and l ($mode[i]$) bits. Assuming knowledge of the full counter value V_i, setting $A[i]$ to $round(\frac{V_i}{2^{r \cdot mode[i]}})$ would imply the lowest errors under this circumstances. However, knowledge of the exact value V_i does not exist, so we have to find other ways to capture the top k bits of V_i. As we noticed, purely SRAM solution exhibits $O(\frac{1}{\sqrt{2^k}})$ errors. This is due to high variance of the background estimators. To illustrate this, consider the following example: scale $r = 2$, $mode[1] = 5$, with all increments equal to 1. In this setting, $A[i]$ is incremented by 1 with probability 2^{-10}. Suppose that the counter V_i updates come with a rate of 1 per unit of time, this means that the time between the two consecutive $A[i]$ increments is given by the geometric random variable $G(p_0)$, with $p_0 = 2^{-10}$. The expected value of $G(p_0)$ is $1/p_0 = 1024$, while the variance of $G(p_0)$ is $\frac{1-p_0}{p_0^2} \approx 2^{20}$. Suppose now that we have available additional memory which can be accessed only occasionally, say once in 16 time units on average. Let $B[i]$ be the additional memory space and in the every time unit increment $B[i]$ by 1 with

```
1   UpdateCounter(i, inc)
2     A[i] = A[i] + ⌊ inc / 2^{r·mode[i]} ⌋
3     With probability { inc / 2^{r·mode[i]} }: A[i] = A[i] + 1
4     if A[i] > 2^k
5       mode[i] = mode[i] + 1;
6       p = { A[i] / 2^r };
7       A[i] = ⌊ A[i] / 2^r ⌋;
8       With probability p: A[i] = A[i] + 1;
9     endif
10    if mode[i] == 2^l
11      RenormalizeCounters(r)
12      r = r + 1;
13    endif
14  Initialize()
15    r = 1;
16    mode[i] = 0;
17    A[i] = 0;
18  RenormalizeCounters(r)
19    for s = 1 : n
20      old_mode = mode[i];
21      mode[i] = ⌈ mode[i] · r / (r+1) ⌉;
22      r_1 = mode[i] · (r + 1) − old_mode · r;
23      p = { A[i] / 2^{r_1} };
24      A[i] = ⌊ A[i] / 2^{r_1} ⌋;
25      With probability p: A[i] = A[i] + 1;
26    endfor
```

Figure 6.1: The pseudocode of SAC.

```
1  UpdateCounter(i, inc)
2   if inc / 2^{r·mode[i]} > 1/η
3     A[i] = A[i] + ⌊inc / 2^{r·mode[i]}⌋
4     With probability {inc / 2^{r·mode[i]}}: A[i] = A[i] + 1
5   else
6     With probability 1/η do
7       B[i] = B[i] + inc;
8       if B[i] ≥ 2^{r·mode[i]} / η;
9         A[i] = A[i] + 1;
10        B[i] = B[i] − 2^{r·mode[i]} / η;
11      endif
12    enddo
13  endelse
14  if A[i] > 2^k
15    mode[i] = mode[i] + 1;
16    B[i] = RandomUniform[0, 2^{r·mode[i]} / η]
17    p = {A[i] / 2^r};
18    A[i] = ⌊A[i] / 2^r⌋;
19    With probability p: A[i] = A[i] + 1;
20  endif
21  if mode[i] == 2^l
22    RenormalizeCounters(r)
23    r = r + 1;
24  endif
25 Initialize()
26  r = 1;
27  mode[i] = 0;
28  A[i] = 0;
29  B[i] = 0;
30 RenormalizeCounters(r)
31  for s = 1 : n
32    old_mode = mode[i];
33    mode[i] = ⌈mode[i] · r / (r+1)⌉;
34    r_1 = mode[i] · (r + 1) − old_mode · r;
35    p = {A[i] / 2^{r_1}};
36    A[i] = ⌊A[i] / 2^{r_1}⌋;
37    With probability p: A[i] = A[i] + 1;
38    B[i] = RandomUniform[0, 2^{(r+1)·mode[i]} / η];
39  endfor
```

Figure 6.2: The pseudocode of HAC.

probability $p_1 = 1/16$. By incrementing $A[i]$ when $B[i]$ reaches 64, the time between two consecutive $A[i]$ increments is given by the random variable Z_1 which is sum of 64 i.i.d. geometric random variables $G(p_1)$. The expected value of Z_1 is $64 \cdot \frac{1}{p_1} = 1024$, while the variance of Z_1 is $64 \cdot \frac{1-p_1}{p_1^2} \approx 2^{14}$, which is approximately 2^6 times smaller than in the case without additional memory. We can think of $B[i]$ as of additional counters stored in the slower DRAM with rare accesses.

Our second scheme HAC formalizes the reasoning from the simple example presented above. The basic SRAM architecture is the same as in SAC: a memory of $n \cdot q$ bits; each q bits counter is divided in k bits estimate space $A[i]$ and l bits $mode[i]$. The difference between SAC and HAC is in the way they increment $A[i]$. In HAC, each of n counters have a shadow counter in DRAM, $B[i]$. Let η be the allowable relative DRAM frequency: every η SRAM accesses we can have one DRAM access. We update $B[i]$ with probability $\frac{1}{\eta}$, and increment $A[i]$ when $B[i]$ "overflows".

Figure 6.2 contains the pseudocode of HAC. For low r and $mode[i]$ (if statement in line 2 is positive) there is no need for DRAM, which become beneficial only when $r \cdot mode[i]$ is larger.

At low speeds, η can be equal to 1, which means that we can store full counters in DRAM. However, in our case of interest, $\eta > 1$, for the moment we will assume that is manually configurable constant. In Section 6.5 we will present a possible control strategy for η, that will take into consideration the available system bus and DRAM bandwidth. In the case of constant η no need for the renormalization of DRAM counters $B[i]$ exists. However, in the case of dynamic η, the renormalization of $B[i]$ is necessary.

Compared to other existing counter architectures HAC does not have exact counter value, but stores only its estimate. Note a paradigm shift: *the main counter information is stored in SRAM while DRAM is used to store information only from the recent history.*

6.4 Analysis

In this section we provide an analysis of the proposed schemes: SAC and HAC. Throughout this section we concentrate on a single counter, say $(A[1], mode[1])$, and denote it with $(A, mode)$. Let $u_1, u_2, \ldots$ be the sequence of increments of counter $(A, mode)$. Denote $V_j = \sum_{s=1}^{j} u_s$. For each $j = 1, 2, \ldots$, denote by A_j the k-bit random variable A after j updates, by $mode_j$ we denote the random variable $mode$ after j updates, and by r_j the random variable that determines scale r after j updates. In the next Theorem we show that the following estimator of V_j is unbiased:

$$\hat{V}_j = A_j \cdot 2^{mode_j \cdot r_j}.$$

Theorem 6.1 *The SAC estimator $\hat{V}_j$ is unbiased:*

$$E[\hat{V}_j] = V_j.$$

Proof: Denote by $(A'_j, mode'_j, r'_j)$,$(A''_j, mode''_j, r''_j)$ and $(A'''_j, mode'''_j, r'''_j)$ the values of $(A, mode, r)$ after execution of the lines 3, 9 and 13 of the SAC algorithm given in Figure 6.1 (Recall that index i is omitted, and that $inc = u_{j+1}$). We have that

$$E[A'_j \cdot 2^{r'_j \cdot mode'_j} | (A_j, mode_j, r_j)] = \left(A_j + \lfloor \frac{u_{j+1}}{2^{r_j \cdot mode_j}} \rfloor\right) \cdot 2^{r_j \cdot mode_j} + \{\frac{u_{j+1}}{2^{r_j \cdot mode_j}}\} \cdot 2^{r_j \cdot mode_j} =$$

$$A_j \cdot 2^{r_j \cdot mode_j} + u_{j+1}.$$

Thus

$$E[A'_j \cdot 2^{r'_j \cdot mode'_j}] = E[A_j \cdot 2^{r_j \cdot mode_j}] + u_{j+1}. \tag{6.2}$$

After the line 9 is executed, we have that either $(A''_j, mode''_j, r''_j)$ is equal to $(A'_j, mode'_j, r'_j)$ or $mode''_j = mode'_j + 1$ and

$$E[A''_j \cdot 2^{r''_j \cdot mode''_j} | (A'_j, mode'_j, r'_j)] = \left\lfloor \frac{A'_j}{2^{r'_j}} \right\rfloor \cdot 2^{r'_j \cdot (mode'_j + 1)} + \left\{ \frac{A'_j}{2^{r'_j}} \right\} \cdot 2^{r'_j \cdot (mode'_j + 1)} = A'_j \cdot 2^{r'_j \cdot mode'_j}.$$

Thus, in both subcases

$$E[A''_j \cdot 2^{r''_j \cdot mode''_j}] = E[A'_j \cdot 2^{r'_j \cdot mode'_j}]. \tag{6.3}$$

Similarly

$$E[A'''_j \cdot 2^{r'''_j \cdot mode'''_j}] = E[A''_j \cdot 2^{r''_j \cdot mode''_j}]. \tag{6.4}$$

Since $(A'''_j, mode'''_j, r'''_j)$ is equal to $(A_{j+1}, mode_{j+1}, r_{j+1})$, from (6.2), (6.3) and (6.4) we conclude that

$$E[\hat{V}_{j+1}] = E[A_{j+1} \cdot 2^{r_{j+1} \cdot mode_{j+1}}] = E[A_j \cdot 2^{r_j \cdot mode_j}] + u_{j+1} = E[\hat{V}_j] + u_{j+1}.$$

Now by a straightforward mathematical induction argument the assertion of the theorem follows. ■

The following Theorem establishes the same result for the HAC.

Theorem 6.2 *The HAC estimator $\hat{V}_j$ is unbiased:*

$$E[\hat{V}_j] = V_j.$$

Proof: Similarly, as in proof of Theorem 6.1 we denote by $(A'_j, mode'_j, r'_j)$,$(A''_j, mode''_j, r''_j)$ and $(A'''_j, mode'''_j, r'''_j)$ values of $(A, mode, r)$ after the execution of the lines 13, 20 and 24 of the HAC algorithm (given in Figure 6.2) respectively; $inc = u_{j+1}$.

From the proof of the Theorem 6.1 we have that the relations (6.3) and (6.4) are satisfied. Now we prove that relation (6.2) is satisfied as well. We distinguish two cases:

1st Case: $\frac{u_{j+1}}{2^{r \cdot mode[i]}} > \frac{1}{\eta}$. Then HAC is identical to SAC and (6.2) follows.

2nd Case: $\frac{u_{j+1}}{2^{r \cdot mode[i]}} \leq \frac{1}{\eta}$. First, note that if B is a random variable uniformly distributed in the segment $[0, a]$ then for any real number x, the random variable $mod(B+x, a) = B+x-a \cdot \lfloor \frac{B+x}{a} \rfloor \in [0, a]$ is also uniformly distributed in $[0, a]$. This means that if we initialize the DRAM counter B with the uniform distribution, it will remain uniform (in the appropriate segment $[0, \frac{2^{r_j \cdot mode_j}}{\eta}]$) after the updates

(line 10). This implies that the probability of $B+u_{j+1} > \frac{2^{r_j \cdot mode_j}}{\eta}$ is given by $\frac{u_{j+1}}{\frac{2^{r_j \cdot mode_j}}{\eta}}$. Thus counter A_j is incremented by one (line 9) with probability

$$q_0 = \frac{1}{\eta} \cdot \frac{u_{j+1}}{\frac{2^{r_j \cdot mode_j}}{\eta}} = \frac{u_{j+1}}{2^{r_j \cdot mode_j}}.$$

Formally

$$E[A'_j \cdot 2^{r'_j \cdot mode'_j} | (A_j, mode_j, r_j)] = (A_j + q_0) \cdot 2^{r_j \cdot mode_j} = A_j \cdot 2^{r_j \cdot mode_j} + u_{j+1}.$$

This implies

$$E[A'_j \cdot 2^{r'_j \cdot mode'_j}] = E[A_j \cdot 2^{r_j \cdot mode_j}] + u_{j+1}.$$

■

6.4.1 Variance estimation

Suppose that a counter runs in scale r and mode $m > 0$. Then by definition the value A of k-bit counter is in the range $[2^{k-r}, 2^k - 1]$. Denote by $T(A, m)$ the random variable that represents the amount of total traffic needed from the start of counting until the SAC (HAC) counter reaches the state (A, m). In this section we evaluate the variance of $T(A, m)$, assuming an uniform increment size θ.

Theorem 6.3 *Suppose that increments to the SAC counter are uniform and given by $\theta > 0$. Then, for the random variable $T(A, m)$ defined above we have:*

$$E[T(A, m)] = A \cdot 2^{rm} \tag{6.5}$$

and

$$Var[T(A, m)] = 2^{k-r} \left(1 + \frac{2^{2rm} - 1}{2^r + 1} - \theta(2^{rm} - 1)\right) + (A - 2^{k-r}) \left(2^{2rm} - \theta 2^{rm}\right). \tag{6.6}$$

Proof: Recall that for a geometric random variable[3] $G(p)$: $E[G(p)] = \frac{1}{p}$ and $Var[G(p)] = \frac{1-p}{p^2}$. For each s denote by $W(r, s)$ the random variable that represents the amount of traffic needed to make one increment of the counter $(A, mode)$ when $mode = s$. The probability of incrementing the counter $(A, mode)$ in a single trial is $p_s = \frac{\theta}{2^{rs}}$. Therefore, the number of trials before the increment of $(A, mode)$ is $G(p_s)$ and since each trial corresponds to θ of the total traffic we have

$$W(s) = \theta G(p_s).$$

The time from the beginning of counting can be divided in $m + 1$ intervals, corresponding to each mode $s = 0, 1 \ldots, m$. In mode $s = 0$ A is incremented $q_0 = 2^k$ times, for modes $s = 1, 2 \ldots, m - 1$, we have $q_s = 2^k - 2^{k-r}$ increments, while in mode $s = m$ until the k-bit counter is A the number of its

[3] A geometric random variable represents the number of Bernoulli trials needed for the first success, when the success of each trial has probability p.

increments is $q_m = A - 2^{k-r}$. Denote by $Q(s)$ the total arrived traffic in each of these $m+1$ intervals. Now for any $s = 0, 1, \ldots, m$

$$Q(s) = \sum_{j=1}^{q_s} W_j(s),$$

where $W_i(s)$ is the set of independent identically distributed (i.i.d.) random variables, with distribution given by $W(s)$. Therefore:

$$E[Q(s)] = \sum_{j=1}^{q_s} E[W_j(s)] = q_s \cdot \theta \frac{1}{p_s} = q_s \cdot 2^{rs}.$$

and

$$Var[Q(s)] = \sum_{j=1}^{q_s} Var[W_j(s)] = q_s \cdot \theta^2 \frac{1-p_s}{p_s^2} = q_s \cdot 2^{2rs}(1-p_s).$$

Now $T(A, m)$ is given by:

$$T(A, m) = \sum_{s=0}^{m} Q(s).$$

The expected value of $T(A, m)$ is now:

$$E[T(A,m)] = \sum_{s=0}^{m} E[Q(s)] = q_0 + \sum_{s=1}^{m-1} q_s \cdot 2^{rs} + q_m \cdot 2^{rm} = 2^k + (2^k - 2^{k-r}) \sum_{s=1}^{m-1} \cdot 2^{rs} + (A - 2^{k-r}) \cdot 2^{rm} =$$

$$2^{k-r} + (2^k - 2^{k-r}) \sum_{s=0}^{m-1} \cdot 2^{rs} + (A - 2^{k-r}) \cdot 2^{rm} = 2^{k-r} + (2^k - 2^{k-r}) \frac{2^{mr}-1}{2^r - 1} + (A - 2^{k-r}) \cdot 2^{rm} = A \cdot 2^{rm}.$$

And the variance

$$Var[T(A,m)] = \sum_{s=0}^{m} Var[Q(s)] = q_0 + \sum_{s=1}^{m-1} q_s \cdot 2^{2rs}(1-p_s) + q_m \cdot 2^{2rm}(1-p_m) =$$

$$2^k + (2^k - 2^{k-r}) \sum_{s=1}^{m-1} \cdot 2^{2rs}(1 - \frac{\theta}{2^{rs}}) + (A - 2^{k-r}) \cdot 2^{2rm}(1 - \frac{\theta}{2^{rm}}) =$$

$$2^{k-r} + (2^k - 2^{k-r}) \sum_{s=0}^{m-1} \cdot (2^{2rs} - \theta 2^{rs}) + (A - 2^{k-r}) \cdot (2^{2rm} - \theta 2^{rm}) =$$

$$2^{k-r} + 2^{k-r}(2^r - 1)(\frac{2^{2mr}-1}{2^{2r}-1} - \theta \frac{2^{mr}-1}{2^r - 1}) + (A - 2^{k-r}) \cdot (2^{2rm} - \theta 2^{rm}),$$

and (6.6) follows. ■

With $\delta(T(A,m))$ we denote the *Coefficient of variation*:

$$\delta(T(A,m)) = \sqrt{E\left[\left(\frac{T(A,m) - A2^{rm}}{A2^{rm}}\right)^2\right]} = \sqrt{\frac{Var[T(A,m)]}{(E[T(A,m)])^2}}$$

The following corollary characterizes the asymptotic behavior of $\delta_{SAC}(T(A,m))$ at the beginning of mode m: $A = 2^{k-r}$.

Corollary 6.1 *For SAC counters at the beginning of mode m ($A = 2^{k-r}$):*

$$\delta^2_{SAC}(T(A,m)) = \frac{1}{2^{k-r+2rm}} \left(1 + \frac{2^{2rm}-1}{2^r + 1} - \theta(2^{rm} - 1)\right). \tag{6.7}$$

When $r \cdot m$ is large:

$$\delta_{SAC}(T(A,m)) \approx \sqrt{\frac{1}{2^k + 2^{k-r}}}.$$

Proof: The equality (6.7) follows directly from (6.5) and (6.6). When $r \cdot m$ is large

$$\delta_{SAC}(T(A,m)) \approx \sqrt{\frac{1}{2^{k-r+2rm}} \frac{2^{2rm}}{2^r+1}} = \sqrt{\frac{1}{2^k + 2^{k-r}}}.$$

■

In HAC case we have the following

Theorem 6.4 *Suppose that the increments to the HAC counter are uniform and given by $\theta > 0$ with relative DRAM access frequency η. Then, for the random variable $T(A,m)$ defined above it holds:*

$$E[T(A,m)] = A \cdot 2^{rm} \tag{6.8}$$

and

$$Var[T(A,m)] = A \cdot 2^{rm} \cdot \theta(\eta - 1). \tag{6.9}$$

Proof: As in the proof of Theorem 6.3, we divide the time into $m+1$ intervals, each corresponding to one mode $s = 0, 1, \ldots, m$ and denote by $W(s)$ the amount of traffic needed to make one increment of the HAC counter at mode s. Thus, $W(s)$ is given by the amount of traffic needed for the DRAM counter B to overflow (ie. become larger than $\frac{2^{rs}}{\eta}$) which is equal to the sum of $b = \frac{2^{rs}}{\eta\theta}$) i.i.d. random variables $Z_i \equiv \theta G(\frac{1}{\eta})$. Therefore

$$W(s) = \sum_{i=1}^{b} Z_i.$$

For $W(s)$ we have

$$E[W(s)] = \sum_{i=1}^{b} E[Z_i] = b \cdot \theta \cdot \frac{1}{\frac{1}{\eta}} = 2^{rs}.$$

and

$$Var[W(s)] = \sum_{i=1}^{b} Var[Z_i] = b \cdot \theta^2 \cdot \frac{1 - \frac{1}{\eta}}{\frac{1}{\eta^2}} = 2^{rs} \cdot \theta(\eta - 1).$$

Now, the amount of traffic $Q(s)$ that corresponds to mode s is given by the sum of q_s i.i.d. random variables $W_j(s)$ identically distributed as $W(s)$: $Q(s) = \sum_{j=1}^{q_s} W_j(s)$. Note that $Var[W(s)] = \theta(\eta - 1) \cdot E[W(s)]$ which implies that

$$E[Q(s)] = q_s \cdot 2^{rs}$$

and

$$Var[Q(s)] = \theta(\eta - 1) \cdot q_s \cdot 2^{rs} = \theta(\eta - 1) \cdot E[Q(s)]. \tag{6.10}$$

Thus, for $T(A,m) = \sum_{s=0}^{m} Q(s)$, from the proof of Theorem 6.3 we conclude that (6.8) is satisfied. Now (6.9) follows from (6.8) and (6.10). ■

From the previous theorem we get:

$$\delta_{HAC}(T(A,m)) = \sqrt{\frac{A \cdot 2^{rm} \cdot \theta(\eta - 1)}{(A \cdot 2^{rm})^2}} = \sqrt{\frac{\theta(\eta - 1)}{A \cdot 2^{rm}}}.$$

Comment 6.1 *The variance calculated above neglected the initial period when* $\frac{inc}{2^{r \cdot mode[i]}} > \frac{1}{\eta}$*, during which the HAC is identical to the SAC. Since the SAC error is smaller than the HAC error in this regime, we do not use the hybrid scheme until the end of this period. Thus, the real HAC error is strictly less than the HAC error presented above. However, the difference is small since the initial period takes a relatively short period of time.*

Comment 6.2 *Note that from the Cauchy-Schwartz inequality, for any real valued random variable* X*:* $\sqrt{E(X^2)} \geq E(|X|)$*. Taking* X *to be the relative error of* $T(A,m)$*,* $X = \frac{T(A,m)-A2^{rm}}{A2^{rm}}$*, the previous inequality implies that the expected relative error is not greater than* $\delta(T(A,m))$*:*

$$E\left[\left|\frac{T(A,m) - A2^{rm}}{A2^{rm}}\right|\right] \leq \delta(T(A,m)).$$

6.5 Resource control

In this section we discuss possible approaches for controlling the variables k and η.

6.5.1 Allocating q bits in two parts.

The basic question is the following. Given q bits of available memory for a counter $(A, mode)$, how many bits should we allocate to A and *mode*? Allocating more bits to A gives less space for *mode* and therefore implies a potentially larger scale r. A larger r implies larger errors for fixed k. Figure 6.3 depicts the graphs of $\delta_{SAC}(T(A,m))$ for $k = 8$ allocating bits for A, in three modes $r = 1, 2, 3$, in the SAC case. On the other hand, allocating too much space for storing *mode* might waste precious space for A. Thus, for a single counter (of size q) one possible way to choose the values k and l is to pick k and $l = q - k$ for which $\delta(T(A(k), mode(k)))$ is minimized. Thus, assuming that the performance objective is minimizing the value of $\delta(T(A(k), mode(k)))$ then k should be dynamically updated to a value that minimizes $\delta(T(A(k), mode(k)))$.

In the case of multiple counters we should pick a common k for all counters. The performance objective can depend on various factors. For example some counters might require higher precision then others; for various applications counters with very low or very high values might be more important than others. In general, for n counters: $(A_i, mode_i)$, picking the optimal k is formally equivalent to the optimization problem:

$$\min \sum_{i=1}^{n} f_i(A_i(k), mode(k), \delta(T(A_i(k), mode(k)))). \tag{6.11}$$

Here f_i are cost functions. If none of the counters is prioritized by default, f_i is independent of i. One should be careful since changing k might cause change of scale r and all counters should be re-normalized to take into account the new conditions. This re-normalization can be done in a straightforward manner and will not be discussed here.

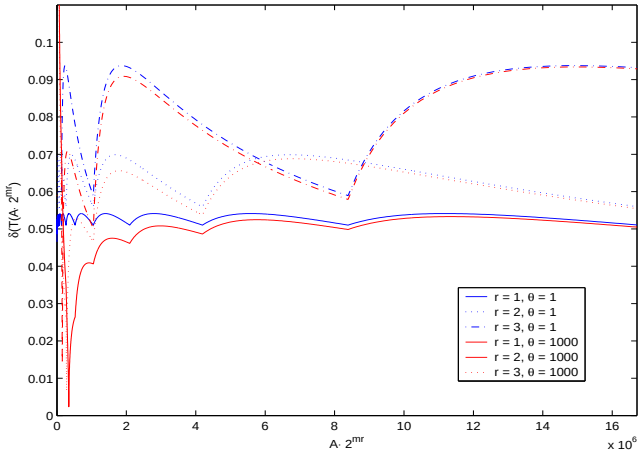

Figure 6.3: Coefficient of variation $\delta_{SAC}(T(A,m))$ for $r = 1, 2, 3$ and $\theta = 1, 1000$; $k = 8$.

6.5.2 Controlling the η

In the HAC case an important parameter that determines accuracy of active counters is relative DRAM access frequency η. A lower η corresponds to better accuracy of the HAC counters. However, there exist a tradeoff between the usage of DRAM and the accuracy of the HAC scheme. The performance bottleneck in many network processors is the system bus and DRAM bandwidth. From the point of exploitation of this resource we can distinguish two cases:

1. The devoted number of DRAM accesses used by HAC in a unit of time Δ (say 1 *sec*) is constant and is given by α.

2. DRAM bandwidth is shared between HAC and other applications.

In both cases we can isolate the buffer for storing the DRAM access queries required by HAC. The rate by which this buffer is serviced depends on the level of utilization and the DRAM bandwidth resource allocation algorithm. Bearing in mind that η is the parameter that controls the arrival rate to this buffer, we can exploit ideas from the large spectrum of AQM algorithms. For example we can control η based on queue length; by tracking the low pass filter of the queue length as in RED like schemes [40, 35, 39] we can dynamically update η to keep the queueing delay low. Virtual queue techniques can be used as well [41, 70], as well as other control strategies. For the stability of such control schemes, it is beneficial that there does not exist feedback delay unlike the case of TCP/AQM where feedback delays can be significant and dangerous from the stability point of view.

The HAC algorithm defined in Section 6.3 assumes a constant sampling rate η. In the context of variable η the augmenting DRAM counters $B[i]$ should take into account the sampling rate. To do this, we can track $\eta[i]$ - the sampling rate at the last update of i-th DRAM counter $B[i]$. At every update of the counter $B[i]$ we first re-normalize $B[i]$ taking into account the current η:

$$B[i] := B[i] \cdot \frac{\eta[i]}{\eta},$$

then reset $\eta[i]$ to the current value of η

$$\eta[i] := \eta.$$

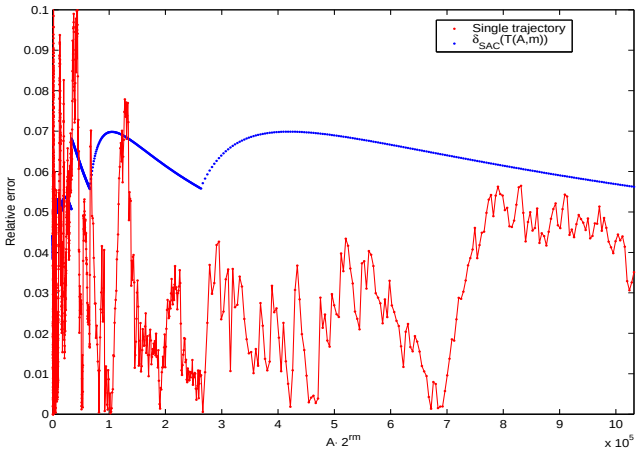

Figure 6.4: Relative error for a single SAC with unit increments; $q = 12, k = 8$.

After this, continue with updating the counter $B[i]$. By doing this unbiasedness of the appropriate counters is preserved. In normal conditions, measurements [83] show that, on typical 150Mbps+ links, basic IP parameters (such as the number of active connections, proportion of TCP traffic, aggregate IP traffic, etc.) do not change dramatically in short time periods which protects η from large oscillations.

Comment 6.3 *At highly loaded links, DRAM bandwidth and network processors are also highly utilized and therefore necessitate higher η. While the accuracy of HAC scheme decreases as η increases, at highly loaded links counter values grow as well which is beneficial for HAC accuracy. However, we are unable to quantify this tradeoff between link (DRAM bandwidth) utilization and HAC accuracy since it is a function of other network processor parameters which are hard to characterize.*

Comment 6.4 *A relatively small queue for storing DRAM updates can (in conjunction with appropriate queue management algorithm) ensure virtually zero loss of data (see [127]).*

6.6 Evaluation

In the first experiment[4] we evaluate the behavior of a single SAC (and HAC) counter over a stream of unit increments. The stream length is $L = 1000000$. Figures 6.4 and 6.5 depict observed relative errors of SAC and HAC respectively, together with the corresponding coefficient of variation. The counter size is $q = 12$ and $k = 8$.

In the following experiment we evaluate the behavior of a single SAC (and HAC) counter over a stream of nonuniform increments. We used first $L = 1000000$ packet arrivals of a NLANR unidirectional trace that we refer to as MRA [84]. Increments are given by the packet size of each of the first million packets. Figures 6.4 and 6.5 depict observed errors of SAC and HAC respectively, together with the corresponding coefficient of variation. For the evaluation of $\delta_{SAC}(T(A, m))$ and

[4] All MATLAB simulations used in this section can be downloaded from http://www.hamilton.ie/person/rade/AC/

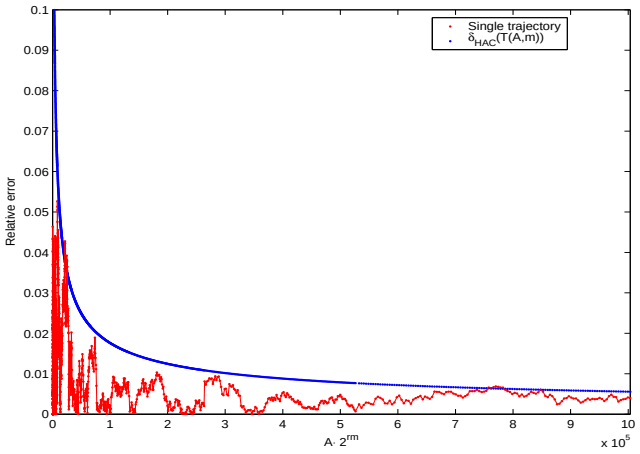

Figure 6.5: Relative error for a single HAC with unit increments; $q = 12, k = 8$.

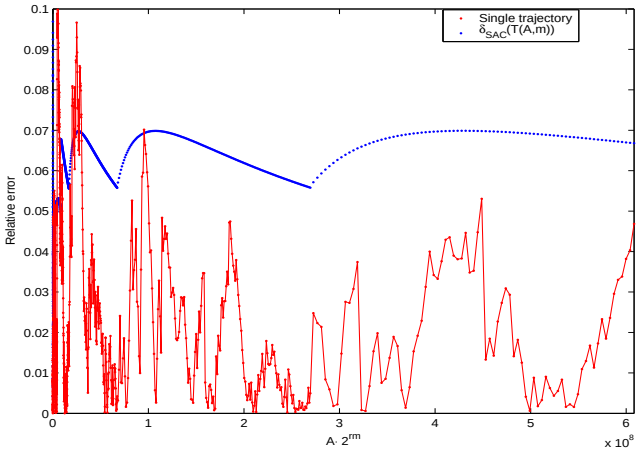

Figure 6.6: Relative error for a single SAC with nonuniform increments; $q = 12, k = 8$.

$\delta_{HAC}(T(A, m))$ we used the average increment size for θ. The counter size is the same as in previous case $q = 12$ and $k = 8$.

6.6.1 Errors versus counter size q

In this experiment we evaluate the average errors for SAC and HAC as a function of counter size q. We use the cost function $f(A(k), mode(k), \delta(T(A(k), mode(k)))) = \delta(A(k), mode(k))$ for dividing q bits in two parts.

Figure 6.8 depicts the results based on a stream with unit increments in the SAC case. Averages are based on 10 independent runs. Figure 6.8 also contains the $\delta_{SAC}(T(A(k), mode(k)))$ for k that minimizes the cost function. As we already noticed (Comment 6.2, Section 6.4.1) the coefficient of variation δ is greater than the expected relative error.

In the HAC case the comparison between errors and counter size is not that straightforward as it depend on several other factors: η and the value of the counter $A \cdot 2^{r \cdot mode}$. In the SAC case the

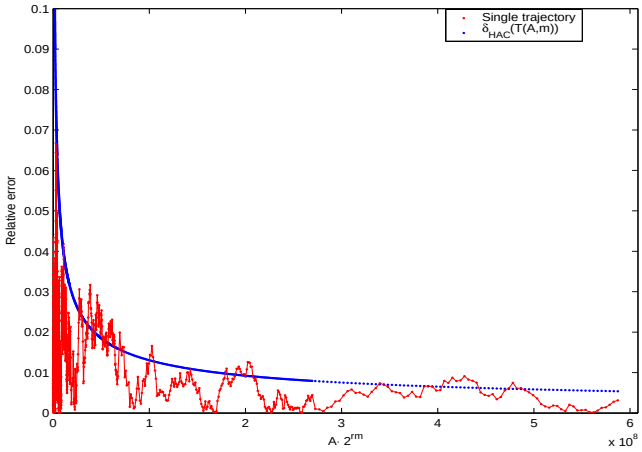

Figure 6.7: Relative error for a single HAC with nonuniform increments; $q = 12, k = 8$.

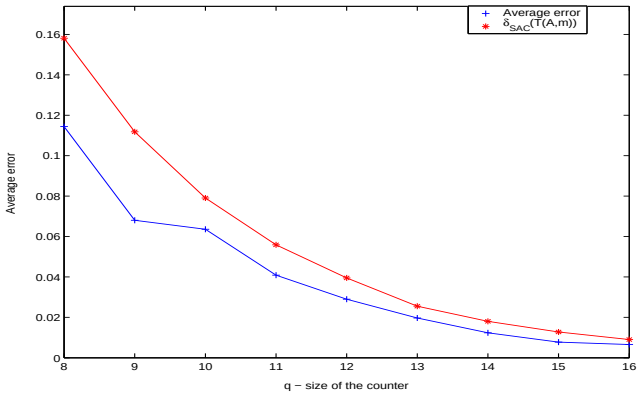

Figure 6.8: SAC. Average errors for different values of q.

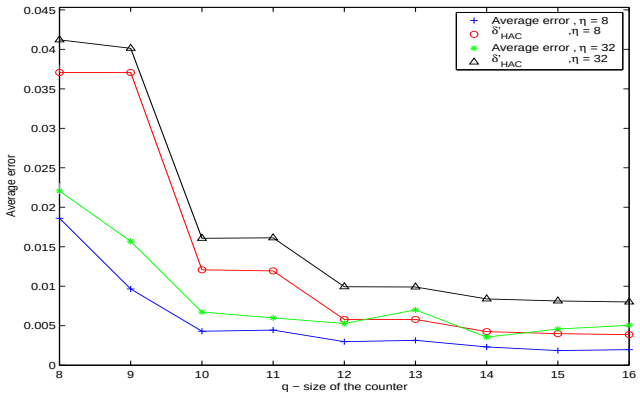

Figure 6.9: HAC. Average errors for different values of q. Stream length 500000 unit increments

MSF	q	S_1	S_2
Standard	24	40	24
With AC	9	16	16

Table 6.1: Parameters of MSF counter size and flow entry size.

rounding error which is of order[5] $O(\frac{1}{2^k})$ is negligible compared to expected δ_{SAC} and we neglected it in computing the expected error δ_{SAC}. However, in HAC case δ_{HAC} can be much smaller and for estimating the error we use $\delta'_{HAC}(T(A, mode)) = \delta_{HAC}(T(A, mode)) + \frac{1}{A}$. Figure 6.9 depicts the errors of the HAC estimates after 500000 unit increments averaged over 10 independent runs, and two different values of η: 8 and 32.

6.6.2 Case study: MSF

In this section we investigate the effects that active counters could have on one of the state-of-the-art algorithm for identification of heavy hitters: Multistage Filters (MSF) [30]. As we said, previously proposed passive counters [96, 94, 95, 127] cannot be used since they store full counter information in DRAM, which is too slow to track the conservative-update step of MSF. MSF uses memory divided in two parts. The first part contains d levels with b counters (of size of q bits) on each level, while the second part contains the hash table that keeps information on heavy hitters; we denote by FC the number of available flow entries and by S the size of a flow entry, in bits. We use S bits of a flow entry to track information on the flow size (S_1 bits) and the flow identifier + hash overhead ($S_2 = S - S_1$ bits). Table 6.1 determines the parameters for two cases: one without active counter enhancements (standard), and another with active counters (with AC).

Smaller sizes of counters allow more counters per stage and more flow entries. The price that must be paid is the lower accuracy caused by smaller counters. Here we show that the gain obtained by more counters outweight the inaccuracy of small counters. We run MSF over two unidirectional NLANR traces: MRA and FRG [84]. The average number of packets per second is 60K (MRA) and 69K (FRG). The average throughput is 285Mbps (MRA) and 428Mbps (FRG). The available

[5] We track the top k bits($A \sim 2^k$).

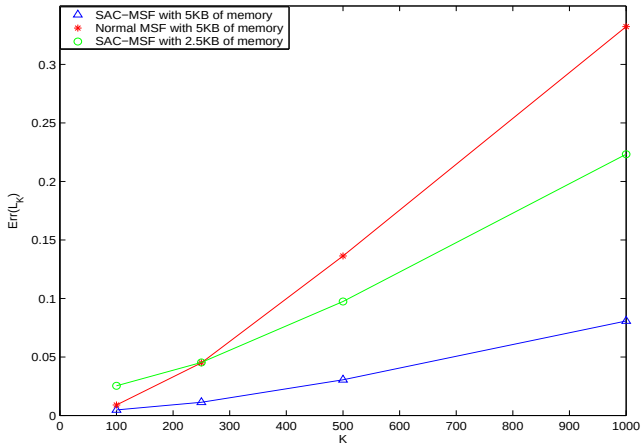

Figure 6.10: MRA trace. The errors in estimation of the top K flows. MSF without any AC (with 40Kbit of memory), MSF + SAC (with 40Kbit of memory) and MSF + SAC (with 20Kbit of memory).

SRAM memory is 40Kbit. We use the recommendation from [30] to determine other parameters of MSF (b,d and FC) in both cases. Since in both cases counters are not large enough (the largest is of order of magnitude of 10^7) to exploit the benefit of HAC for simplicity we use only SAC[6]. By using the cost function $f_i(A_i(k), mode_i(k), \delta_{SAC}(T(A_i(k), mode_i(k)))) = \delta_{SAC}(T(A_i(k), mode_i(k)))$ the optimization criterion stabilizes k at value $k_1 = 6$ for the basic 9-bit counters. For 16-bit (flow container) counters that track flow sizes the value of k that solves the same optimization problem is $k_2 = 12$.

The metric of interest is the average error for the set of flows indexed by the set I defined as in [30] by:

$$Err(I) = \frac{\sum_{i \in I} |est(f_i) - r(f_i)|}{\sum_{i \in I} r(f_i)} \tag{6.12}$$

Here $r(f_i)$ is the exact size of flow f_i and $est(f_i)$ is the MSF estimate. By L_K we denote the set of top-K flows. We evaluate the average errors for the top-100, top-250, top-500 and top-1000 flows. The results are presented in Figures 6.10 and 6.11. Observe that the errors of top-100 flows are approximately the same both with and without SAC, while the ratio between the errors $Err(L_K)$ for MSF with standard counters and MSF with SAC grows as K grows: MSF with SAC gives up to 6 times smaller errors in measuring the top-1000 flows. In parallel we show the errors of MSF with SAC using half as much memory (20Kbit).

6.7 Summary

In this chapter we propose two randomized active counters algorithms, SAC and HAC, prove their unbiasedness and evaluate the variance of the corresponding random variables. While active counters

[6]Using HAC would give errors that are strictly less then using only SAC, but the difference is not significant in the present experiment, so we concentrate on SAC only.

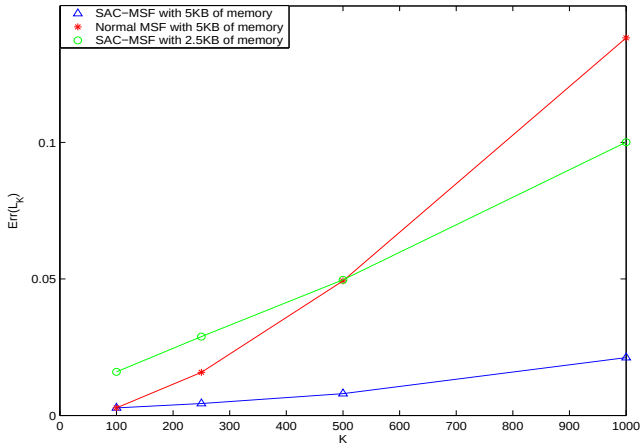

Figure 6.11: FRG trace. The errors in estimation of the top K flows. MSF without any AC (with 40Kbit of memory), MSF + SAC (with 40Kbit of memory) and MSF + SAC (with 20Kbit of memory).

are essential for a number of recently proposed data streaming and/or packet sampling algorithms [30, 15, 22, 124], they can also be beneficial for other applications like remote monitoring at heavily loaded network processors (See Section 6.1.2).

SAC uses only SRAM but exhibits larger errors than HAC which uses augmenting DRAM for storing information on the recent history to reduce the variance. Storing full counter information in SRAM and recent history information in DRAM is a paradigm shift from the previous passive counters proposals [94, 127, 95, 96] that use DRAM for full counter information and SRAM for recent history details. Having in mind the possible high load of DRAM bandwidth caused by other applications we suggest dynamic control of the DRAM access frequency. This enhancement can be directly exploited by other SRAM/DRAM proposals that usually neglect the variability of the available DRAM bandwidth.

Finally we conclude with a list of open problems that we plan to investigate in the future.

Open problem 1. Let Λ be a data streaming algorithm that uses limited memory (given by S bytes) and active counters architecture with q-bits per counter. Pick q such that cost $C(\Lambda)$ (errors, false positives, false negatives, or something else) is minimized.

Open problem 2. Characterize available DRAM bandwidth at (heavily loaded) network processors and investigate (adaptive) hybrid SRAM/DRAM schemes under variable DRAM access frequency using rich AQM theory.

Open problem 3. The proposed solutions can be immediately extended to multi-operations arithmetics (present solutions consider only additive counters). Could the theory be extended? Are there applications that need it?

CHAPTER 7

Scalable HH identification

Abstract - *In this chapter we address the problem of real-time identification of rate-heavy-hitters on very high speed links, using small amounts of SRAM memory. Assuming a nonuniform distribution of flow rates, runs (where two randomly selected bytes belong to the same flow) occur predominantly in the heaviest flows. Based on this observation we present RHE (Realtime Heavy-hitter Estimator), a measurement tool which uses a small amount of memory for extracting the heaviest flows. Using this tool a queue management algorithm called HH (Heavy-hitter Hold) is presented that approximates fair bandwidth allocations. RHE possesses very low memory requirements, scales with line speeds of several tens of Gbps and covers a wide range of flow rate distributions. Measurements over real Internet traces show the high efficiency of RHE, achieving a high accuracy with a very small amount of SRAM memory. Compared to Estan-Varghese's Multistage Filters and Lucent's CATE estimator, RHE achieves up to 10 times smaller errors in measuring high-rate flows on a number of synthetic traces. Packet level ns2 simulations in a synthetic heavy-tailed environment are presented to illustrate efficacy of HH.*

7.1 Introduction

Measurement of traffic on high speed links is essential for appropriate network management. In order to optimize performance, network operators need traffic measurements that provide insights at both quantitative and qualitative level. Long-term measurements can be used for traffic engineering in developing network architecture or rerouting traffic. Short-time measurements that monitor traffic on a scale of few seconds to few hours are needed for intrusion detection. Real time measurements can be used for managing flow control and providing QoS. Also, practical knowledge of various aspects of network traffic provides necessary feedback to researchers that work on analysis of IP networks.

Having its importance in mind, traffic measurement, as a research area, has attracted significant attention in last few years. The basic problem behind this research lies in the inability of large DRAM

memory devices to process packets quickly enough and fast SRAM memories to allow enough space for full per-flow state. Namely, the number of concurrent flows on high speed links ($\geq 1Gbps$) can be more than a million. A naive approach, of keeping per flow state for each flow cannot be implemented on large DRAM since it is not possible to manage per-flow counters at speeds of current backbone links; moreover one cannot expect in the near future that the speeds of DRAMs (which increase approximately 7% per year) will reach speeds of backbone links (which roughly double every year) [30, 2]. On the other hand, using fast SRAM is not feasible since the size of available SRAM chips is significantly smaller than needed for keeping per-flow state for all flows at very high speeds [2, 94, 127].

Based on the observation that on most internet links a small number of flows account for large amount of bandwidth, the authors of the seminal paper [30], have developed 2 efficient algorithms for measuring of these heavy-hitters – flows that account for most of bandwidth. However, these algorithms are passive in sense that they measure size of flows rather than their *rates*. Efficient, real-time, measurements of flow rates rather than flow sizes would be essential in QoS per-flow management and for enforcing fairness in networks in which users use heterogenous congestion control algorithms, or do not react on congestion signals at all. In environments with flows that have a fixed sending rate, measuring the flow size and the flow rate over an interval is equivalent. In environments with TCP-like flows, where the rate may increase and decrease, one should be much more careful when measuring flow rates.

In this chapter we develop a novel technique, called RHE, for real-time measuring of rates of flows with the highest rates on very high speed links. We then apply this to develop an Active Queue Management(AQM) scheme called HH that controls fairness using information on measured rates.

For the purpose of measuring flows with highest rates we introduce a data structure that is built of memory cells. Each memory cell corresponds to certain set of flows that are mapped to it. Assuming nonuniform distribution of flow rates, information kept in a memory cell is enough to extract (with high probability) the flow with the highest rate that maps to that cell.

Measurements over synthetic streams show that RHE achieves up to 10 times smaller errors in measuring heavy flows, compared to state of the art methods: Multistage Filters (MSF) [30] and Coincidence bAsed Rate Estimator[1] (CATE) [44]. On the other hand, a number of measurements performed on the real packet traces show high accuracy of RHE with extremely small amount of memory.

The chapter is structured as follows. In Section 7.2 we describe previous work and define the problem of interest. Section 7.3 will present the technique RHE for dynamic tracking of flows with highest rates. Analysis of RHE will be given in Section 7.4. In Section 7.5 we apply RHE to introduce an AQM scheme HH that will control fairness[2]. We will experimentally evaluate both RHE and HH in Section 7.6.

[1] CATE is the final version of runs based estimators developed by Lucent group and its performance is strictly better than the previous incarnations RATE [61] and ACCEL-RATE [43].

[2] Throughout this chapter we accept max-min definition of fairness [102].

7.2 Preliminaries

7.2.1 Previous work

The most widely used tool for traffic measurements is Cisco NetFlow [82]. NetFlow keeps per-flow state for each user in a large DRAM memory. At very high speeds NetFlow deals with the following two issues: (a) packet inter-arrival times can be significantly smaller than time needed for processing of a packet; (b) the amount of traffic measurement data can be enormous. To overcome these problems NetFlow uses sampling, with a sampling rate as a parameter that is set manually. However, it is clear that in order to optimize memory usage and accuracy, traffic conditions must be considered when choosing the sampling rate. The paper [32] deals with this problem by introducing self-tuning NetFlow, where the sampling rate adapts to current traffic measurements. See [46] or [23] for review of sampling methods in traffic measurements.

In order to move from slow[3] DRAM to fast SRAM the authors of [30] suggest a method called Multistage Filtering (MSF) that attempts to monitor only flows with volume greater than a certain threshold, T, in a certain measurement interval, and to neglect all smaller flows. In order to meet this goal, MSF contains two components: hash filters and a flow container. On every packet arrival the flow identifier is used as an index for computing hash in several stages. Each stage uses an independent hash function and each counter at every stage contains the aggregate amount of traffic for all flows that map to that counter. The volume of a flow is then estimated as the least value of all counters that this flow maps to. If the estimated volume of a flow is greater than the threshold T the flow is added to the flow container. It is obvious that if a certain flow has a volume greater than T then the estimated volume will be at least T and it will be added to the flow container, assuming that there is space for it. In order to prevent flows with volume smaller than T being wrongly identified as large, two enhancements are introduced: shielding and conservative updates. Assuming that the sending rates do not vary over the measurement intervals and that the objective is finding the total volume over the measurement interval rather than the sending rate of a flow, MSF together with conservative updates and shielding achieve this goal with high accuracy for heavy hitters in environments with heavy-tailed flow size distributions. For flow control one needs rate, rather than the volume of a flow over long measurement intervals and a quick response to this information. The main contribution of this chapter is RHE which will meet these requirements using only small and fast SRAM memory.

In [61], in order to save memory, the authors propose a mechanism, called RATE, for estimating elephant flows by counting *runs*: events that correspond to the situation where an arriving packet belongs to the same flow as some sampled packet that has recently arrived at the link. However, to give an accurate estimate of elephant flows on links with large number of users RATE needs a long time. In order to solve this scalability problem, the same authors propose ACCEL-RATE [43] and CATE [44] which require more per packet processing and memory. As it is shown in [44] these algorithms "*...tend to slightly over-estimate flow rates.*" over real IP traces. This problem is caused

[3]Current access times are in range $50 - 100ns$ for DRAM, and $2 - 6ns$ for SRAM [127].

by two factors: (a) the bursty nature of internet traffic, and (b) the fact that many flows do not have a constant rate but rather a variable rate, or even more drastically on/off patterns. Factor (a) can be resolved by introducing buffer for counting runs. Factor (b) appears to be much harder to resolve and is the main cause of inaccuracy of RATE-like algorithms that try to use a run counter as the main source of measurements. However, counting runs can give some important insights on flow rates. We are going to exploit this idea, but in a different way, as we will see in later sections.

Another interesting approach in traffic measurements is the accurate estimation of the number of very small flows as they would indicate potential DDoS or Internet worm attacks. In [66] the authors use a Bayesian statistical method, Expectation Maximization, together with the efficient implementation of sequence of counters proposed in previous chapter to give a very accurate estimation of the flow size distribution[4]. A potential weakness of this work is a vulnerability in the number of counters. Namely, it is shown in [66] that accuracy of the proposed algorithm drops considerably when number of flows is larger than 2*(number of available counters). See [65] for another memory efficient scheme that utilizes a multiresolution-Bloom-filter technique to capture the (heavy) flow rates. For caching techniques used in heavy-hitter identification see [13] and references therein.

Fair queueing has been an important research area in networking for the past two decades, motivated by the idea that in heterogenous network conditions, the service that a user gets should not depend on the aggressiveness of its congestion control algorithm. Various approaches include fair scheduling [18, 100], keeping state information on overly aggressive users [73, 76, 88] or change of IP infrastructure and exploiting state information written in IP header for controlling fairness as in [110, 55]. The way information is extracted about highly aggressive flows in [73, 76, 88] is by straightforward sampling and requires relatively large amounts of memory on very high speed links. HH goes further in this direction and by exploiting efficacy of RHE we aim to control flow rates even on very high speed links with low memory usage.

7.2.2 Problem definition

A flow can be defined as set of packets that satisfy a certain condition that is called the flow key. In most cases the flow key is defined as some function of the following information kept from a packet header: IP address of source and destination, port numbers, protocol ID, etc.

Our goal is to do real-time accurate measurement of the rates of flows with the highest rates with the finite amount of SRAM memory. In order to do so, the most important question is "What flows have the biggest rate?". Thus the problem of interest for us is following:

For a given positive integer M identify and monitor as many as possible among the M flows with the highest sending rate using approximately $10 \cdot M$ bytes of SRAM memory.

An important assumption that has to be taken in consideration is that per-packet processing should be less than few hundreds of ns, or equivalently less than the time of processing an average-

[4]Note that problem of estimation of flow size/rate distribution is different from the problem of heavy-hitter identification since the first problem ignores identities of the flows.

sized packet at given high-speed link. Thus, although we are concerned with elephant flows, our objective is to monitor the M "biggest" flows rather than flows with size greater than some threshold T.

The notion of flow rate is not strictly defined in networking literature. However, there is consensus that for applications related to flow control, flow rates should involve some kind of weighted averaging in order to smooth out possible large variations that can occurs as a consequence of bursty traffic and on/off traffic patterns. For example, the authors of [110] define flow rate as a weighted average that depends on inter-arrival time T between the previous and the current packet of size l:

$$A_{new} = (1 - e^{-T/K})\frac{l}{T} + e^{-T/K}A_{old},$$

where A_{new} and A_{old} denote estimated rates for the current and previous packet respectively and K is a smoothing parameter. Implementing a strategy like this would require keeping per-flow time-stamps as well as using a "CPU-expensive" exponential function. Since our goal is the estimation of rates with as small amount of memory usage as possible that runs at line speed for high speed links, we will try to avoid keeping the time-stamps and use functions with lower computational cost in our definition of flow rate.

We define the sending rate of flow f in the following way: we divide the time from beginning of a measurement into sub-measurement intervals of length δ. Let $vol^{(f)}(a, b)$ be the volume of data in the interval (a, b) in bytes. At time $t \in (s\delta, (s+1)\delta]$, the rate of flow f in bytes per second is defined as:

$$r_f(t) = \frac{1}{q\delta + (1-q)(t - s\delta)}(q\delta \cdot r_f(s\delta) + (1-q) \cdot vol^{(f)}(s\delta, t)) \tag{7.1}$$

This can be seen as weighted averaging[5] of rates over sub-measurement intervals with weighting parameter q: packets that arrive in the last sub-measurement interval have weight $1-q$, packets that arrive in the previous sub-measurement interval are accounted with weight $q(1-q)$,..., packets that arrive in the k-th previous sub-measurement interval are accounted with weight $q^k(1-q)$. Throughout this chapter we will calibrate parameters q and δ in such fashion that weights of bytes decreases approximately exponentially with factor e per second. The weight of a byte that arrived at time $t = \tau_0 - 1$ is $q^{-1/\delta}$ times[6] smaller than weight of a byte arrived at time $t = \tau_0$. Thus, our calibration rule is $q^{1/\delta} \approx e^{-1}$. For q close to 1, we can approximate:

$$e^{-1} \approx q^{1/\delta} = \left((1 - (1-q))^{\frac{1}{1-q}}\right)^{\frac{1-q}{\delta}} \approx e^{-\frac{1-q}{\delta}}.$$

Therefore, choosing q and δ to satisfy a simple rule $\delta = 1 - q$, implies approximately exponential decrease of weights with rate of e per second.

We will now show how one can effectively compute the rate of a flow f defined by (7.1). Let co be a counter that is incremented by the packet size of each arriving packet that belongs to flow f. At

[5]Weighted averaging (or low pass filtering) is a technique widely used for the de-noising of noisy data. While other more subtle signal-processing methods are possible (see for example [20]), in our case the weighted averaging performs good in the de-noising flow rates and can be effectively implemented.

[6]We assume that δ is measured in seconds; then $1/\delta$ is the number of sub-measurement intervals in one second.

the time $t_s = s\delta$, the value of the counter co is updated by $co \leftarrow q \cdot co$. Now, at time $t \in (s\delta, (s+1)\delta]$, the rate of flow f is given by

$$r_f(t) = co \cdot \frac{1-q}{q\delta + (1-q)(t - s\delta)}$$

The RHE measurement tool described in next section, is motivated by applications in flow control and therefore is designed for measuring flow rates. However, for the purpose of measuring flow sizes one can take $\delta = 1$ and discard the scaling factor. With this minor change one can adapt RHE for flow sizes measurement (but throughout this chapter we assume that $q < 1$ and that RHE measures rates rather than sizes).

7.2.3 How to check if an element of the set is maximal?

Here we present the argument that can help to build the intuition behind RHE.

Suppose that we have a set A of positive real numbers, that all sum up to 1: $\sum_{x \in A} x = 1$. How can we check whether one of them, say $x_1 \in A$, is the maximal element of the set A using a small amount of memory? In some extreme cases no additional memory is needed to answer this question[7]. Suppose now that value of $m_d = \sum_{x \in A} x^d$ is available as well, for some $d > 1$. Then:

$$\lim_{d \to \infty} \frac{m_d}{x_1^d} = \lim_{d \to \infty} \sum_{x \in A} \left(\frac{x}{x_1}\right)^d = 1 \quad \text{if} \quad x_1 = \max(A),$$

and

$$\lim_{d \to \infty} \frac{m_d}{x_1^d} = \lim_{d \to \infty} \sum_{x \in A} \left(\frac{x}{x_1}\right)^d = +\infty \quad \text{if} \quad x_1 < \max(A).$$

In practice, for a finite d, one can chose a threshold M such that x_1 is declared as maximal if and only if $\frac{m_d}{x_1^d} < M_d$. By choosing the appropriate M, the confidence of the above decision can be very high as we will see in next section.

In our context (of heavy-hitter identification) the set A corresponds to the set of flow rates (typically few dozens of flows) that map to a cell. By using the idea from RATE, i.e. by counting runs, one can obtain an estimate of m_2 with very small memory overhead and small computational complexity. Using higher powers ($d > 2$) is possible also, but it would increase memory overhead needed for calculating m_d, and our initial experiments show no benefit by choosing higher d so we concentrate on $d = 2$ in the rest of the chapter.

7.3 RHE

In this section we present tool for measuring flows with the highest rates. The basic idea is as follows: at each packet arrival a hash of a flow is computed and the packet is mapped to a memory cell. Since many flows can map to same memory cell, our goal is to identify the one with highest sending rate. In order to do it we introduce data structure sketched on Figure 7.1.

[7] For example, if $x_1 > 0.5$ then it is clearly the largest element of A.

$cRate$ (2B)	$eqPairs$ (2B)
$rateTyp$ (2B)	$SampledB$ (2B)
$typPkt$ (2B)	LF (1bit)

Figure 7.1: Layout of a memory cell.

A memory cell contains $cRate$, information on aggregate rate of all flows that map to this cell and $rateTyp$, information on rate of a flow that is a candidate for the highest rate among flows that maps to this cell that is further identified by $typPkt$: another 16bit hash value of its flow key. Counters $cRate$, $rateTyp$ and $eqPairs$ (which will be described shortly) mimics the technique introduced in Section 7.2.2, for tracking the rate with one counter without time-stamps. Thus, whenever a packet from a flow that maps to the memory cell arrives, its size is added to counter $cRate$; if it belongs to flow with hash value equal to $typPkt$, counter $rateTyp$ is incremented by size of arriving packet. Finally at the end of each sub-measurement interval of length δ we set

$$(cRate, rateTyp, eqPairs) \leftarrow q(cRate, rateTyp, eqPairs),$$

with $q \in (0, 1)$ as a weighted averaging parameter. By doing this we follow the definition of the flow rate given by (7.1), so the aggregate flow rate at time $t \in (s\delta, (s+1)\delta]$ in bytes per second is given by $cRate \cdot \frac{1-q}{q\delta+(1-q)(t-s\delta)}$ and similarly the rate of "candidate" flow is given by $rateTyp \cdot \frac{1-q}{q\delta+(1-q)(t-s\delta)}$. The question is "how do we know if the candidate is the one with highest sending rate?". To answer this question we exploit the heavy tailed nature of the Internet flow rate distribution, and introduce an idea that is at the core of our method and applies the reasoning discussed in Section 7.2.3.

Each memory cell contains $SampledB$: a 16bit hash value of flow ID for some uniformly sampled byte (rather than packet) that arrived in this cell. On each arrival $SampledB$ is compared with the arriving packet's 16bit hash value and if they match then the size of arriving packet is added to counter $eqPairs$ [8]. As we already said, at the end of each sub-measurement interval of length δ, the counter $eqPairs$[9] is set to $q \cdot eqPairs$. We claim that once we have this information then we can say with high confidence if the flow represented by $typPkt$ has highest rate (or one of the highest rates) among flows that map to this cell. To support this statement we will have to understand the meaning of the counter $eqPairs$. Let $r_1, r_2 \ldots, r_n$ be the rates of all flows that map into same cell and let R be the aggregate rate: $R = r_1 + \cdots + r_n = cRate\frac{1-q}{q\delta+(1-q)(t-s\delta)}$. Sampling $SampledB$ byte-wise and assuming no hash collision we have that

$$SampledB = i \quad \text{with probability} \quad z_i = \frac{r_i}{R}.$$

[8]The idea of the comparing the arriving packet's flow ID with the flow ID of some previously arrived packet has been exploited in AQM design in Stabilized RED [86] and CHOKe [89] as well as in recently proposed traffic estimators [61, 44].

[9]Following the terminology of RATE [61], counter $eqPairs$ can be seen as the total number of runs weighted by packet sizes.

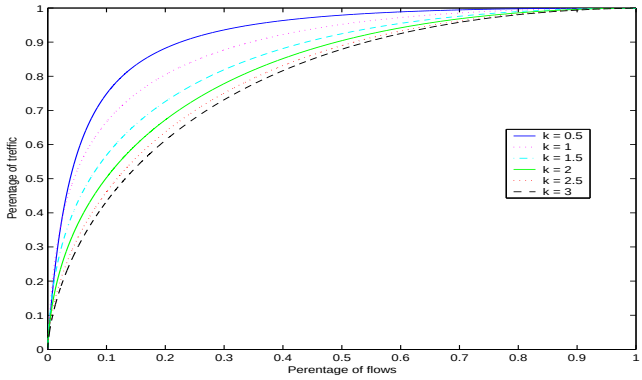

Figure 7.2: CDF for Pareto distribution for $k = 0.5, 1, 1.5, 2, 2.5, 3$, $m = 40$.

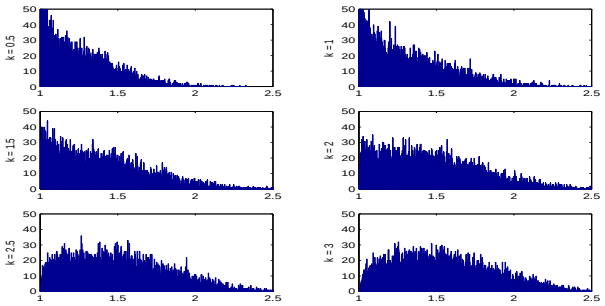

Figure 7.3: Histograms of $\sqrt{\sum_i^n z_i^2}/\max(z_i)$, $n = 20$.

Therefore, the expected value of $eqPairs$ can be estimated by:

$$E(eqPairs) = \frac{q\delta + (1-q)(t - s\delta)}{1-q} \sum_{i=1}^{n} r_i P(SampledB = i) =$$

$$= \frac{q\delta + (1-q)(t - s\delta)}{1-q} \cdot \sum_{i=1}^{n} (R \cdot z_i) \cdot z_i = cRate \sum_{i=1}^{n} z_i^2 \tag{7.2}$$

In the large number of packets-per-second regime variance of $eqPairs$ is small and $\sum_i^n z_i^2$ can be accurately approximated with $eqPairs/cRate$. On the other hand, for the Internet-like (heavy-tailed) flow rate distributions $\sum_i^n z_i^2$ is dominated by the maximal value of z_i.

To see this we plot histograms of $\sqrt{\sum_i^n z_i^2}/\max(z_i)$ for six different cumulative distribution of z_i given in Figure 7.2. These six CDF's, correspond to Pareto distribution with $k = 0.5, 1, 1.5, 2, 2.5, 3$ and $m = 40$ ($P(X_k > m + i) \sim (\frac{1}{m+i})^k$). Figure 7.3 contain histograms for $n = 20$; Figure 7.4 contain histograms for $n = 100$.

As we can see from Figures 7.3 and 7.4, with very high confidence one can declare that $\sqrt{\sum_i^n z_i^2}/\max(z_i)$ is less than 2.5 (in the first case) or 3.5 (in the second case). We use this to define a test for deciding if the candidate flow, represented by its hash value $typPkt$ and with rate $rateTyp$ is the "high-rate" flow. We know that $rateTyp = z_i \cdot cRate$ for some $i \in \{1, \ldots, n\}$. By using the estimate (7.2), we conclude that if $\frac{\sqrt{\frac{eqPairs}{cRate}}}{\frac{typRate}{cRate}}$ is large, then this implies that the candidate flow is not one with maximal rate and we should continue with the search for maximal one. Formally, at the end of each

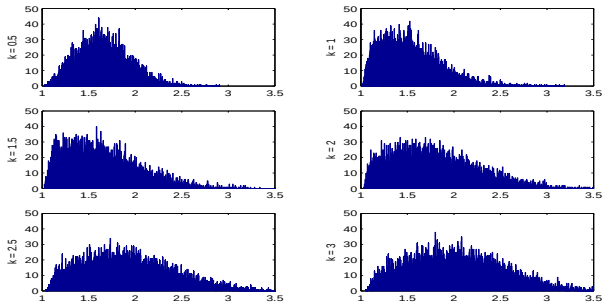

Figure 7.4: Histograms of $\sqrt{\sum_i^n z_i^2}/\max(z_i)$, $n = 100$.

sub-measurement interval of length δ, we do the test described in Figure 7.5. One can notice that we use the CPU-expensive $\sqrt{\cdot}$ function in defining the test. However one can ask (equivalently) if $eqPairs \cdot cRate > (c_0 \cdot typRate)^2$. We emphasize again that this test is done only once in a sub-measurement interval of length δ and therefore its expense is not essential.

. if $\sqrt{\frac{eqPairs}{cRate}} > c_0 \frac{typRate}{cRate}$

. REPLACE $typPkt$ WITH $SampledB$.

. end if

Figure 7.5: Testing of a candidate for high-rate flow.

Here $c_0 \in [1, 4]$ is a parameter that should be chosen to reflect traffic conditions. In the section 7.4.2 we will show how to calibrate c_0.

Thus we have described an algorithm that will eventually give us the hash of the flow ID for the flow that satisfy $\sqrt{\frac{eqPairs}{cRate}} < c_0 \frac{typRate}{cRate}$. This flow might not be the largest among flows that map to the same cell, but in that case,

$$(\frac{typRate}{cRate})^2 + \max(\frac{r_i}{cRate})^2 \leq \sum_i^n z_i^2 < c_0^2 (\frac{typRate}{cRate})^2.$$

This means that

$$typRate \geq \sqrt{\frac{1}{c_0^2 - 1}} \max(r_i).$$

Having some kind of heavy-tailed flow rate distribution, we can chose c_0 to be in the interval $(2, 4)$ and therefore the presented algorithm will give us a flow with a rate that is either the flow with largest rate or a flow whose rate is within an order of magnitude of the largest rate among flows that map to that cell.

Sometimes two or more heavy flows can map to the same cell. In order to identify as many large-rate flows as possible, we divide the available memory of M memory cells into several stages. The first stage with size $\lfloor M(1-\rho) \rfloor$, the second with size $\lfloor M(1-\rho)\rho \rfloor$, the third with size $\lfloor M(1-\rho)\rho^2 \rfloor$ and so on. Since we want to reduce per-packet processing we will use small ρ, say $\rho \leq 1/2$. At every packet arrival, if the packet belongs to the flow represented by $typPkt$ its rate is estimated by $rateTyp \cdot \frac{1-q}{q\delta + (1-q)(t-s\delta)}$. If the packet does not belong to the flow represented by $typPkt$ and if the

. if $\frac{1-q}{q\delta+(1-q)(t-s\delta)} cRate\sqrt{\frac{eqPairs}{cRate} - (\frac{typRate}{cRate})^2} > T$

. $LF \leftarrow 1$

. else

. $LF \leftarrow 0$

. end if.

Figure 7.6: LF bit update rule

bit LF (bit LF is described in the next paragraph and indicates existence of some Large Flow among non-$typPkt$ flows that map to the cell) is set to 1, the packet is hashed in one of cells in the next stage. If the arriving packet does not belong to the flow represented by $typPkt$ and LF bit is 0, the flow the arriving packet belongs to is identified as small, and its rate is estimated by $T/2$, where T is threshold that will be introduced in next paragraph.

In order to allow as few small flows to pass this stage as possible, the bit LF bit is updated once, or several times (depending on abilities of hardware) per sub-measurement interval, in the way sketched on Figure 7.6.

By doing this we ensure that if there is a flow that sends at a rate greater than threshold T, it will be allowed to go to next stage. Indeed, suppose that LF is set to 0 then

$$T \geq \frac{1-q}{q\delta+(1-q)(t-s\delta)} cRate\sqrt{\frac{eqPairs}{cRate} - (\frac{typRate}{cRate})^2} =$$

$$\frac{1-q}{q\delta+(1-q)(t-s\delta)} cRate\sqrt{\sum_{i=1}^{n} z_i^2 - (\frac{typRate}{cRate})^2} =$$

$$\frac{1-q}{q\delta+(1-q)(t-s\delta)} cRate\sqrt{\sum_{i\neq typPkt}^{n} z_i^2} \geq$$

$$\frac{1-q}{q\delta+(1-q)(t-s\delta)} cRate \max(z_i \mid i \neq typPkt) =$$

$$\max(r_i \mid i \neq typPkt).$$

Thus, if there is at least one flow that maps to the cell that is not represented by $typPkt$ and with rate greater than T, then the LF bit will not be set to 0, and therefore it will be allowed to pass to next stage (together with all other non-$typPkt$ flows). Note that although the LF bit set to 1 does not necessary mean that there is a flow with rate greater than T among the flows that pass from the cell to the next stage, it will indicate the existence of a flow with rate that is within an order of magnitude of T in this set of flows, assuming heavy-tailed flow rate distribution. In other words, if there does not exist a flow whose rate is within an order of magnitude of T, then the LF bit will be set to 0 and these flows will not be processed in the next stage.

In Subsection 7.3.2 we will show how to adapt the value of threshold T in order to ensure effective usage of memory. Until then we will assume that T is a parameter that is chosen manually.

. INPUT - PACKET: pkt, INTEGER: s

. $CELL \leftarrow f_s(ID(pkt))$ * HASH ON FLOW ID OF pkt

. $cRate(CELL) \leftarrow cRate(CELL) + size(pkt)$

. $h \leftarrow f_0(ID(pkt))$ * SECOND HASH

. if $(h == typPkt(CELL))$

. $rateTyp(CELL) \leftarrow rateTyp(CELL) + size(pkt)$

. end if

. if$(h == SampledB(CELL))$

. $eqPairs(CELL) \leftarrow eqPairs(CELL) + size(pkt)$

. end if

. WITH PROBABILITY $p = \frac{size(pkt)}{MaxPktSize}$ do

. $SampledB(CELL) \leftarrow h$

. end do

Figure 7.7: Per-packet update of memory cell at stage s.

7.3.1 Implementation issues

The data structure used in RHE is composed of M memory cells (with structure sketched at Figure 7.1) divided into the number of stages whose sizes decrease roughly geometrically with coefficient ρ. At the beginning of the measurement all values of all memory cells are initialized to 0. Whenever a packet arrives, the hash function of its flow key is computed. This maps the packet to one memory cell, and the counter $cRate$ is incremented by the size of arriving packet in bytes. Another hash function is computed on the flow identifier of the arriving packet that maps each flow to a 16bit integer h. If h is equal to value $typPkt$ then the counter $rateTyp$ is incremented for the size of the arriving packet in bytes. Also, if h is equal to $SampledB$, then the counter $eqPairs$ is incremented by the size of the arriving packet. Finally with probability $p = \frac{PktSize}{MaxPktSize}$ the $SampledB$ field is updated with h. Here the parameter $MaxPktSize$ is the maximal possible size of a packet that can be seen at the link. Since more than 99.99% of all packets in the Internet are 9000B or less we set $MaxPktSize = 9000$[10].

Pseudo code for the processing of a packet at stage s is given in Figure 7.7. The complete sequence of actions for an arriving packet is shown on Figure 7.8

In order to avoid random number generation (that might consume a lot of CPU time), we suggest the following: one register should contain 16bit integer RN. Instead of calling a random number generator to update $SampledB$ we add the packet size to RN: $RN \leftarrow RN + size(pkt)$. If $RN \geq MaxPktSize$ then update $SampledB$ and set $RN \leftarrow ((RN + 1)\ mod\ MaxPktSize)$, otherwise do nothing. By doing this, the value RN will mimic a random integer uniformly distributed on $[0, MaxPktSize - 1]$ and the probability that $SampledB$ is updated is proportional to the size of the arriving packet. Histograms of RN using the first million arrivals of MRA trace [84] shows uniform

[10]Once MTU increases for a significant amount of traffic, then $MaxPktSize$ can be increased to follow this change.

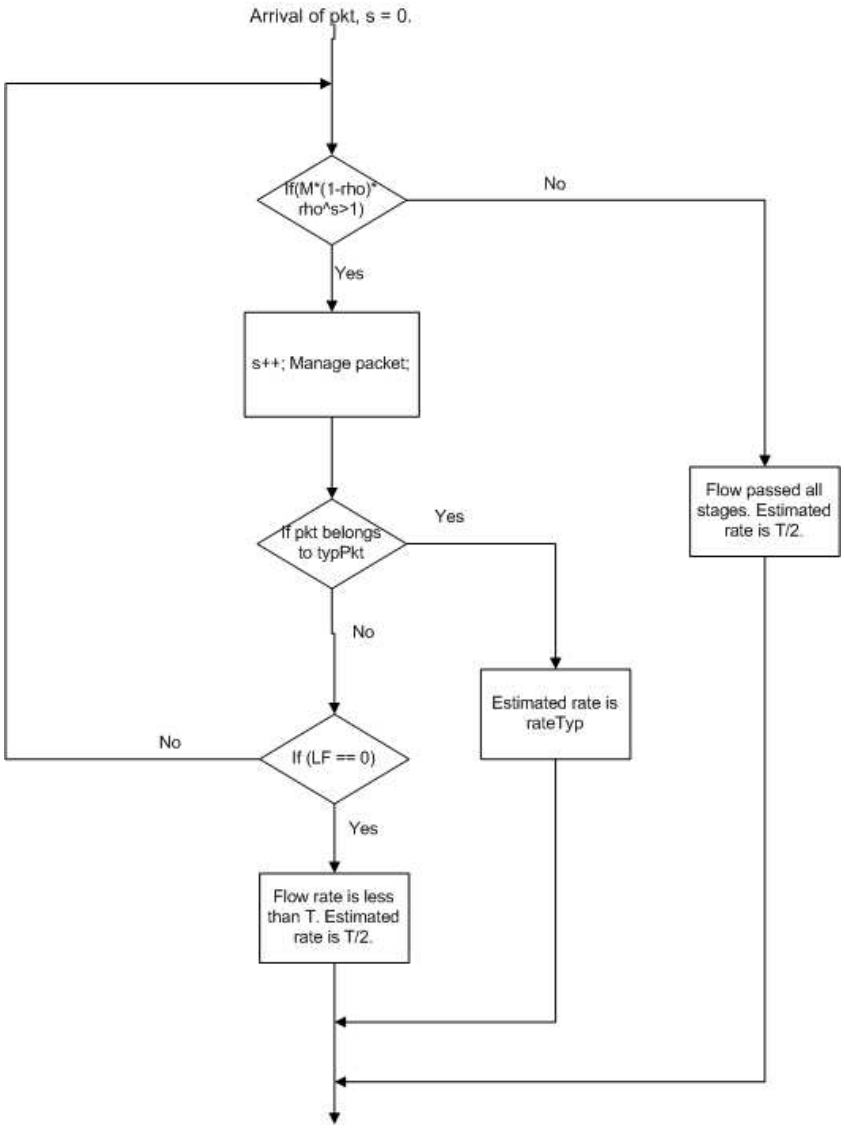

Figure 7.8: Flow chart

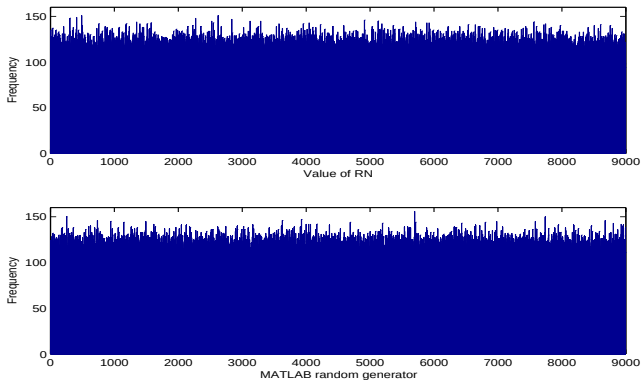

Figure 7.9: Histogram of RN using trace MRA(top) and appropriate histogram of random numbers generated by MATLAB random generator.

distribution of RN on $[0, MaxPktSize - 1]$.

If we do not require 100% accuracy, we can reduce the number of bits of a counter co using the active counters architecture proposed in [106].

7.3.2 Self tuning of threshold T

The threshold T is used by RHE for reducing the amount of traffic that passes from one stage of memory cells to another. Using too high value of T might cause underutilization of memory cell space in higher level stages, and a loss of accuracy for flows with rate less than T, since RHE does not care about accuracy of flows with sending rate less than T. On the other hand, setting T too low results in a large amount of traffic passing from one stage to another. This will imply the following undesirable features. Firstly, since the number of cells per stage decreases geometrically, the number of flows that will map to the same cell grows implying a loss of confidence in our test (Figure 7.5) as well as increasing the time to search for a flow with one of the highest rates. Secondly, a large number of packets passing this "threshold test" would imply larger average per-packet processing times. Knowing the flow rate distribution, one can manually set a threshold that would be close to optimal for a given amount of memory cells[11].

However we propose a simple adaptation algorithm that will control amount of traffic that passes from one stage to another. The basic idea is the following: if the number of cells with the LF bit set to 1 is less then half of number of cells at the first stage, then the filtering should be amplified and so T should be increased, otherwise T should be decreased. Our algorithm uses simple MIMD[12] strategy with parameters $\alpha_1 = 1.1$, $\alpha_2 = 0.9$ that updates T once per sub-measurement interval. Pseudo code is given on Figure 7.10.

[11] For example, with M memory cells one possible approach is to chose a T value around $M/2$-th highest flow rate.

[12] Multiplicative-Increase, Multiplicative Decrease. Other control strategies are possible as well, but in our case MIMD is satisfactory.

ONCE PER SUB-MEASUREMENT INTERVAL OF LENGTH δ

. if $((\sum LF$ *on stage 1*$)>0.5\cdot($*Size of stage 1*$))$

. $\quad T \leftarrow T \cdot 1.1$

. else

. $\quad T \leftarrow T \cdot 0.9$

. end if

Figure 7.10: MIMD algorithm for adaptation of threshold T.

7.4 Calibration of parameters

7.4.1 Effect of ρ

In this section we evaluate how the choice of ρ can affect the number of identified top flows. Going back to the question stated in Section 7.2.2, our goal was monitoring of as much as possible of the largest M flows with the memory equivalent to the M memory cells. A perfect algorithm would identify and monitor all M largest flows. However, using RHE there might exist "bad" cells that no one of the largest M flows map to. Since the total number of cells is M, the number of the largest M flows that can be monitored is at most (M - number of "bad" cells). By estimating the number of "bad" cells we will give bounds on the number of large flows that can be tracked by M memory cells. We will see that these bounds are not far from simulation results presented in Section 7.6.

Questions of interest are as follows:

(1) Estimate G - the number of cells that contain at least one of the largest M flows. We denote the set of the largest M flows by L_M.

(2) Estimate VG - the number of cells that contain at least one of the largest $M/2$ flows. We denote the set of the largest $M/2$ flows by $L_{M/2}$.

The following analysis will estimate G and VG as function of ρ. Since we are interested in the upper bounds for G and VG, we will assume, for simplicity, that the threshold is such that all L_M flows passes from one stage to another. This will slightly decrease the accuracy of our bound, but the main objective of this section is to give some intuition on how much we can expect from RHE.

Let a_k be the average number of flows that map to one cell at k-th stage. Denoting by N_k the total number of flows at k-th stage, we have that

$$a_k = \frac{N_k}{M\rho^{k-1}(1-\rho)}.$$

The probability that among a_1 flows at first stage there is no one from set L_M is

$$P((f_1 \notin L_M) \wedge \cdots \wedge (f_{a_1} \notin L_M)) = P(f_1 \notin L_M)^{a_1} =$$

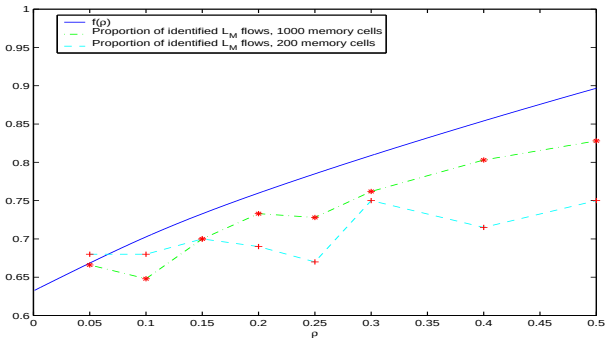

Figure 7.11: $f(\rho)$ - Bound on the proportion of identified flows from L_M for $\rho \in (0, 0.5)$.

$$\left(1 - \frac{M}{N_1}\right)^{a_1} = \left(\left(1 - \frac{1}{a_1(1-\rho)}\right)^{a_1(1-\rho)}\right)^{\frac{1}{1-\rho}} \approx e^{-\frac{1}{1-\rho}}$$

Therefore at first stage, the number of "bad" cells one can expect is roughly:

$$B_1 = M(1-\rho)e^{-\frac{1}{1-\rho}}.$$

Thus, at the first stage one can expect roughly $G_1 = M(1-\rho) - B_1$ "good" cells. In the perfect case (where each cell extract the flow with the highest rate) at the second stage one can expect approximately $M - G_1 = M(\rho + (1-\rho)(1 - e^{-\frac{1}{1-\rho}}))$ flows from L_M. Therefore, the probability that a cell from the second stage is "bad" is

$$P((f_1^{(2)} \notin L_M) \wedge \cdots \wedge (f_{a_2}^{(2)} \notin L_M)) = \left(P(f_1^{(2)} \notin L_M)\right)^{a_2} =$$

$$\left(1 - \frac{M - G_1}{N_2}\right)^{a_2} = \left(1 - \frac{\rho + (1-\rho)e^{-\frac{1}{1-\rho}}}{a_2\rho(1-\rho)}\right)^{a_2} \approx e^{-\frac{\rho+(1-\rho)e^{-\frac{1}{1-\rho}}}{\rho(1-\rho)}}.$$

Thus the number of "bad" cells on the second stage is roughly

$$B_2 = M\rho(1-\rho)e^{-\frac{\rho+(1-\rho)e^{-\frac{1}{1-\rho}}}{\rho(1-\rho)}}.$$

Similarly, at the k-th stage the number of bad cells can be approximated with

$$B_k = M\rho^{k-1}(1-\rho)e^{-\frac{\rho^{k-1}+(B_1+\cdots+B_{k-1})/M}{\rho^{k-1}(1-\rho)}}.$$

Having this value the proportion of identified L_M flows is upper-bounded by

$$f(\rho) = \frac{M - \sum_{k=1}^{\infty} B_k}{M}. \tag{7.3}$$

On Figure 7.11 we plot the quantity $f(\rho)$, for $\rho \in (0, 0.5)$.

From Figure 7.11 we can see that by choosing a higher ρ we can enforce a lower number of "bad" cells. The price for this would be a larger number of stages and therefore a larger per-packet processing time.

The bound given by (7.3) gives a rough approximation to G - the number of "good" cells as function of ρ. The number of identified L_M flows is not larger than

$$G = f(\rho) \cdot M. \tag{7.4}$$

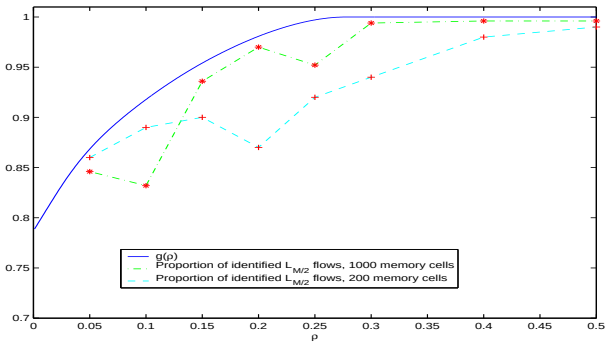

Figure 7.12: $g(\rho)$ - Bound on the proportion of identified flows from $L_{M/2}$ for $\rho \in (0, 0.5)$.

Using similar arguments we obtain a bound for VG:

$$VG = g(\rho)\frac{M}{2}$$

where

$$g(\rho) = \frac{\sum_{k=0}^{\infty} A_k}{\frac{M}{2}}. \tag{7.5}$$

The sequence A_k satisfies: $A_0 = 0$ and

$$A_k = (1-\rho)\rho^{k-1}M(1 - e^{-\frac{\frac{1}{2}-(A_0+\cdots+A_{k-1})/M}{\rho^{k-1}(1-\rho)}}).$$

The plots in Figures 7.11 and 7.12 also depicts simulation results on the proportion of identified L_M and $L_{M/2}$ flows with RHE for various values of ρ. Two sets of simulations were performed over 10000 flows with a cumulative distribution of flow rates given by Pareto distribution with $k = 0.5, m = 40$, as depicted in Figure 7.2: one with 1000 memory cells another with 200 memory cells. Since the number of flows per memory cell in the second case is larger than in the first case, the test for identifying the highest flow has lower confidence in the second than in the first case. Therefore, we see that the proportion of identified L_M and $L_{M/2}$ flows with 200 memory cells is slightly lower than with 1000 memory cells (although we see for $\rho \leq 0.1$ the opposite effect; this effect is consequence of threshold adaptation that is neglected in our analysis).

7.4.2 Choosing c_0

We saw in the Section 7.3 that if n numbers, $z_1, \ldots, z_n$, are drawn with some heavy tailed distribution $\mathcal{D}$ then the distribution $\sqrt{\sum_i^n z_i^2}/\max(z_i)$ is concentrated close to 1; see Figures 7.3 and 7.4. The parameter c_0, that defines test given on Figure 7.5, should be chosen is such fashion that following probability is minimized:

$$p_0(\mathcal{D}, n, c_0) = Prob((z_1, \ldots, z_n)|\ \frac{\sqrt{\sum_i^n z_i^2}}{\max(z_i)} > c_0).$$

The value p_0 represents the probability that the test will not eventually finish its search once

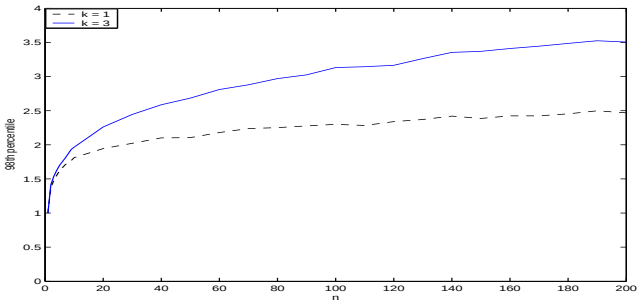

Figure 7.13: 98% percentile of the distribution of $\sqrt{\sum_i^n z_i^2}/\max(z_i)$, for $n = 1, ..., 200$. z_i are drawn with Pareto distribution with $k = 1$ (dashed line) and $k = 3$ (full line).

the heaviest flow is found[13]. Note that even in that case (test does not finish), the maximal flow will be hashed to the next stage(s) (and will be given chance for identifying there) as long as its rate is greater than the threshold T.

Figure 7.13 depicts (empirically obtained) values of c_0 for which $p_0 = 0.02$ when z_i are drawn with Pareto distribution for two different values of parameter k and $m = 40$. Formally, it depicts graphs of $\bar{c}_0(Pareto_1, n)$ and $\bar{c}_0(Pareto_3, n)$ for the distributions $Pareto_1$ $(k = 1, m = 40)$ and $Pareto_3$ $(k = 3, m = 40)$, where $\bar{c}_0$ is defined by:

$$\bar{c}_0(\mathcal{D}, n) = x_0 \quad \text{for which} \quad p_0(\mathcal{D}, n, x_0) = 0.02$$

Throughout this chapter we assume that for the distribution of flow rates $\mathcal{D}$ holds following

Assumption 7.1

$$\bar{c}_0(\mathcal{D}, n) \leq \bar{c}_0(Pareto_3, n) \tag{7.6}$$

Informally, it says that the distribution $\mathcal{D}$ is "more" heavy-tailed than $Pareto_3$. We performed an analysis on a number of real Internet traces and we have not found a single trace that does not satisfy Assumption 9.1. Assuming (7.6), for choosing c_0, we first estimate the number of flows of a single memory cell using bitmap technique [31](b is the size of the bitmap, we use $b = 200$; z is the number of zero bits):

$$\hat{n} = b \ln(\frac{b}{z}).$$

Then we use

$$c_0 = \bar{c}_0(Pareto_3, \hat{n}). \tag{7.7}$$

[13]Recall that choosing too large c_0 might cause the algorithm to stop searching for the heaviest flow by finding a flow (with rate *typRate*) that satisfies $\sqrt{\frac{eqPairs}{cRate}} \leq c_0 \frac{typRate}{cRate}$.

7.5 Application: Heavy-hitter Hold - HH

In this section we present a queue management scheme called HH, that will be built on RHE. The main objective of HH is enforcing max-min fair bandwidth allocation. The reasons for employing "fair queueing" and its importance can be found in [110], [88] and [33]. In particular, a queue scheme that enforces fair bandwidth allocation would protect responsive and not too aggressive flows from non-responsive and aggressive ones, and could be essential for providing QoS requirements once the number of users that use some of the high-speed [58, 37] or loss-insensitive transport protocols becomes large (compared to the total number of users).

The pseoudocode of HH is given in Figure 7.14. We use a MIMD (Multiplicative Increase-Multiplicative Decrease) algorithm for controlling the variable $FAIR$ (representing the maximal allowable rate): whenever the estimated rate, *estc*, of the flow the arriving packet belongs to, is greater than $FAIR$, the packet should be dropped with probability $(estc - FAIR)/estc$. We chose to update the variable $FAIR$ once per interval of length δ, with same δ as in RHE (but of course the frequency of updating $FAIR$ and RHE parameters do not have to be same). Our performance goal is to keep utilization of the link at the prescribed value *util*. On each update of $FAIR$ we ask whether the utilization is less then *util* or not and update $FAIR$ with the MIMD parameters $t_1 > 1$ and $t_2 > 1$.

Having information on $FAIR$, RHE will use threshold $T = FAIR/2$ rather than manually setting the threshold or self-tuning of the threshold. Setting the threshold on value $T = FAIR/2$ allows the monitoring of a long-lasting TCP flow after it responds to packet loss by halving its congestion window.

In the pseudocode given in Figure 7.14, RHEEstRate(*pkt*) and EstCELL(*pkt*) represent the estimated rate of the flow packet *pkt* belongs to and the memory cell that has given this estimate.

In order to prevent multiple losses for TCP (or more generally loss-responsive) flows, we introduced a 2bit variable *marked* attached to each cell. For the memory cell $CELL$, $marked(CELL)$ is 0 if no packet that corresponds to CELL has been dropped in both the current and previous update interval; $marked(CELL)$ is 1 if some packet that corresponds to CELL has been dropped in the current update interval but no packet that corresponds to CELL is dropped in the previous update interval; $marked(CELL)$ is 2 if some packet that corresponds to CELL has been dropped in the previous update interval but no packet that corresponds to CELL is dropped in current update interval and $marked(CELL)$ is 3 if some packets that correspond to CELL have been dropped in both the previous and the current update interval. If $marked(CELL)$ is equal to 1 no packets with EstCELL(*pkt*) equal to $CELL$ will be dropped until the next $FAIR$ update when $marked(CELL)$ is updated to reflect the definition we have already given.

```
On arrival of packet pkt
. [estc, CELL] = [RHEEstRate(pkt) EstCELL(pkt)]
.   if (estc < FAIR)
.     ShouldDrop ← 0;
.   else
.     rand = Random :: uniform(0, 1);
.     if (rand < (estc − FAIR)/estc)
.       ShouldDrop ← 1;
.     end if
.   end if
.   if marked(CELL) == 1
.     ShouldDrop ← 0;
.   end if
.   if ShouldDrop == 1
.     Drop(pkt);
.     if (marked(CELL) mod 2 == 0);
.       marked(CELL) ← marked(CELL) + 1;
.     end if
.   end if
.   if (now − LastUpdate > δ)
.     if (CurrentThroughput/Capacity < util)
.       FAIR = FAIR ∗ t1;
.     else
.       FAIR = FAIR/t2;
.     end if
.     LastUpdate = now;
.     for CELL = 1 : memorysize
.       if (marked(CELL) mod 2 == 1);
.         marked(CELL) ← 2
.       else if (marked(CELL) == 2)
.         marked(CELL) ← 0
.       end if
.     end for
.   end if
```

Figure 7.14: Pseudocode of HH

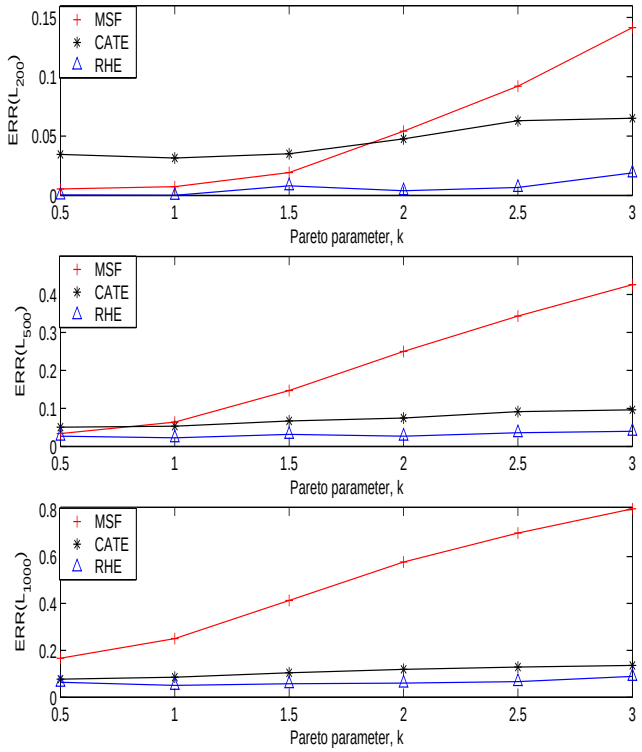

Figure 7.15: MSF vs CATE vs RHE. Average errors $ERR(L_{200})$, $ERR(L_{500})$ and $ERR(L_{1000})$ for different values of Pareto parameter.

7.6 Evaluation

Following the analysis from section 7.4.1 we set $\rho = 0.3$ in the following experiments[14]. Setting $\rho = 0.3$ implies $g(\rho) \approx 1$, meaning that number of cells that contain at least one of the largest $M/2$ flows is approximately equal to $M/2$, allowing high probability of identifying this top-$M/2$ flows. The parameter c_0 is tuned according to rule (7.7).

7.6.1 Comparison between Multistage Filters, CATE and RHE

We will compare MSF [30], CATE [44], and RHE over 6 different synthetic streams where flow size distribution follows Pareto distribution with parameter $k = 0.5, 1, 1.5, 2, 2.5, 3$, and $m = 40$. Cumulative Distribution Functions (CDF) of this six distributions are shown in Figure 7.2.

For each k, the length of stream is $L = 500000$, and the total number of flows is $NF = 10000$, each member of the stream has unit size and belongs to flow $i \in \{1, 2 \ldots, NF\}$ with probability w_i where w_i are chosen with Pareto distribution with parameter k whose CDF is depicted in Figure 7.2. The parameters for the weighted averaging of flow rates are $\delta = 0.5sec$, $q = 0.5$. The length of measurement interval for MSF is $\delta_1 = 2sec$. The number of arrivals per second is 50000, giving total length of single test 10 seconds. The available memory for RHE is $M = 1000$ memory cells. We

[14]Code used in this section can be found at http://www.hamilton.ie/person/rade/RHE/.

assume that the relative size of counters is 20% of a memory cell, and that the relative size of flow entry (flow identifier + counter + pointer needed for hash table) is 40% of the size of a memory cell. For the given memory of 1000 memory cells, the values for the number of stages (d), the number of counters per stage (b) and the number of flow entries (FC) are derived from the recommendations given in Section 6.1 in [30]: $d = 4$, $b = 615$, $FC = 1270$. For CATE, we use $k = 50$ comparisons per packet[15], as recommended in [44].

The metric of interest is the *average error between estimated traffic share and real traffic share* defined as in [30] by the ratio between the sum of the absolute values of the differences between the estimated and real traffic share divided by total sum of the real traffic share[16]. Thus, for a set of flows indexed by set I, the error is defined as:

$$ERR(I) = \frac{\sum_{i \in I} |est(f_i) - r(f_i)|}{\sum_{i \in I} r(f_i)} \tag{7.8}$$

We evaluated the average errors for the top 200, top 500 and top 1000 flows, indexed by sets L_{200}, L_{500} and L_{1000}. Our findings are presented in Figure 7.15. We observe that the ratio between the average errors of MSF and RHE grows as k grows, which shows a much higher sensitivity of MSF on the "heavy-tail assumption". We also note the low insensitivity of CATE to the "heavy-tail assumption": errors grow slowly with Pareto parameter k. However, the high variance of CATE implies relatively large errors.

Comment 7.1 *Note that the basic Multistage filtering algorithm (without shielding and conservative updates) identifies a flow as being large based on the information on* $\min_i(AggrVol(C_i) \mid i = 1 : d)$. *In the case of large number of flows per counter(say* > 10*) the quantity* $AggrVol(C_i)$ *will be (with high probability) within order of magnitude with volume of the heaviest flow that maps to that counter* C_i *only in the very heavy-tailed environments. Indeed, the ratio* $AggrVol(C_i) \,/ \max(Vol(f) \mid f \in C_i) = (\sum_{f \in C_i} z_f) / \max_{f \in C_i}(z_f)$ *has a probability distribution that has a much larger mean than* $\sqrt{(\sum_{f \in C_i}(z_f)^2)} / \max_{f \in C_i}(z_f)$ *and therefore the quantity* $\max_{f \in C_i}(Vol(f))$ *is not dominant in* $AggrVol(C_i)$ *if either number of flows per counter is large or size distribution does not have* very *heavy tail. As we already noted in Section 7.2.1, the authors of MSF introduced multiple stages, shielding and conservative updates to improve confidence in finding heavy-hitters, while RHE uses additional information given in the memory cell to solve same problem.*

Comment 7.2 *Multiple stages in MSF are an integral part of MSF algorithm, and a packet have to pass all stages twice (once for the counter increments and once for the conservative-update correction). However, RHE attempts to identify the heaviest flow at single stage and additional stages (with sizes that decrease geometrically) are introduced only because of possible collision when two heavy flows*

[15] Using more comparisons would be computationally expensive for high speed links($> 1Gbps$).

[16] RHE measures flow rates while CATE and MSF measure flow sizes so we use generic term "traffic share" to refer to flow rates in RHE case and flow sizes in MSF and CATE case.

mapping to same cell. The following table shows average number of stages (ANS) per packet for RHE, for six different values of parameter k:

k	*0.5*	*1*	*1.5*	*2*	*2.5*	*3*
ANS	*1.457*	*1.351*	*1.372*	*1.352*	*1.360*	*1.356*

Comment 7.3 *In [29] MSF is compared with a number of sketch based approaches (see [16, 63] and references therein) for identifying the heavy-hitters. It is claimed that MSF needs less memory to achieve given task. To quote [29]:* "...the conclusion is that multistage filters as I use them identify heavy hitters with less memory than sketches...".

7.6.2 Measurements on real packet traces.

In this section we present experimental results over 3 NLANR unidirectional traces [84] that we refer to as MRA, FRG and ALB. In all our experiments we use the definition of a flow at the granularity of TCP connections, i.e. a flow is defined by the 5-tuple of source and destination IP address, and port and protocol numbers. We choose to compare the measured and real rates at the moment $t_1 = 15sec$ from the start of measurement $t_0 = 0sec$. The number of flows, the number of packets, and the average (total) throughput of these three traces in the interval $[t_0, t_1]$ are given in the following table:

Trace	# of flows	# of packets	Avg throughput
MRA	70826	905312	285.7 *Mbps*
FRG	27393	1030953	428.7 *Mbps*
ALB	181394	2501702	919.1 *Mbps*

The parameters used are: $\delta = 0.1sec$, $q = 0.9$, and the number of memory cells in MRA and FRG measurements is $M = 500$ and for larger ALB we set $M = 1000$.

In Table 7.1 we present our results. The metrics of interest are: the proportion of identified $L_{M/5}$(top $M/5$ flows), $L_{M/2}$ and L_M flows(we denote proportion of identified L_k flows by $PropIden(k)$), the average error for $L_{M/5}$, $L_{M/2}$ and L_M flows and the average number of stages(ANS) per-packet. We also provide the average errors for L_M flows obtained by CATE (with $k = 50$ comparisons) and MSF with memory space equivalent to M memory cells(see previous subsection) on these three packet traces. Large errors with CATE are due to relatively small sample (15 seconds of trace) and therefore high variance.

The average number of flows per memory cell on the first stage is around 200 on MRA and ALB traces and around 80 on FRG trace. In spite of this large number of flows per memory cell, RHE is able to extract almost all the top-$M/5$ flows and about 90% of $M/2$ flows with highest rates. The difference between the number of identified $L_{M/2}$ and L_M flows, and the upper bounds given in Section 7.4, is mainly caused by the fact that the threshold T often stabilizes around a value that does not allow for some non-identified flows from L_M and $L_{M/2}$ to pass from one stage to another. We should also observe results for the average number of stages per cell. On the most challenging trace

Metrics	MRA	FRG	ALB
$PropIden(M/5)$	0.99	1.00	1.00
$PropIden(M/2)$	0.904	0.896	0.960
$PropIden(M)$	0.648	0.694	0.730
$ERR(L_{M/5})$	0.0073	0.0122	0.0094
$ERR(L_{M/2})$	0.0277	0.0312	0.0241
$ERR(L_M)$	0.0409	0.0408	0.0412
ANS	1.5593	1.4057	1.5909
$ERR(L_M)$(MSF)	0.2837	0.1368	0.0625
$ERR(L_M)$(CATE)	0.3850	0.4780	0.4967

Table 7.1: The proportion of identified $L_{M/5}$, $L_{M/2}$ and L_M flows; the average error for flows from $L_{M/5}$, $L_{M/2}$ and L_M; the average number of stages(ANS) per-packet; MSF and CATE errors for L_M flows.

ALB, with aggregate throughput of more than $900Mbps$, the average number of packets per second is around 166000. With 1.5909 stages per-packet, this gives more than $3\mu s$ per-packet update on one stage, defined on Figure 7.7. Assuming fast SRAM with access time of $4ns$[17], a packet update would take less than $100ns$, which is roughly 30 times less than that needed on trace ALB with aggregate throughput of about $919Mbps$.

For the trace A, let $k(A)$ be the largest integer such that all top-$k(A)$ flows are identified; we will denote by $TT(A)$ the proportion of traffic taken by top-$k(A)$ flows. On all three traces $TT(MRA), TT(FRG), TT(ALB)$ are greater than 60%. Thus, RHE is able to *fully* identify top flows responsible for more than 60% of traffic on the gigabit links with only several kilobytes of SRAM memory. Figure 7.16 graphically illustrates the accuracy of RHE for all 3 traces.

7.6.3 Performance of HH.

In this subsection we will present $ns2$ packet level simulation results that will illustrate the behavior of HH. To do this we initiated 1000 TCP flows divided in three groups[18], whose TCP parameters are described in the Table 7.6.3 that share 40 Mbit/s link. We ran two experiments with these 1000 flows: one over the link with a Drop Tail queue and another over a HH queue. The queue size in both cases is 300Kbytes which is $60ms$ of buffering on $40Mbit/s$ link.

The HH parameters are $util = 0.98$, $t1 = 1.003$, $t2 = 1.003$. The weighting-average, parameters are the same as in the previous subsection $\delta = 0.1sec$, $q = 0.9$ and the number of memory cells is $M = 40$. Flows from the first group do not have maximal congestion window limitation and they account for 67.36% of bandwidth in the DropTail case. In the HH case, the total share of bandwidth for flows from the first group is reduced to 41.03%, while the total share of bandwidth for other two

[17]Current SRAM access time varies between $2ns$ and $5ns$[127].

[18]The first group mimics elephant flows, the second kangaroos and the third mice flows.

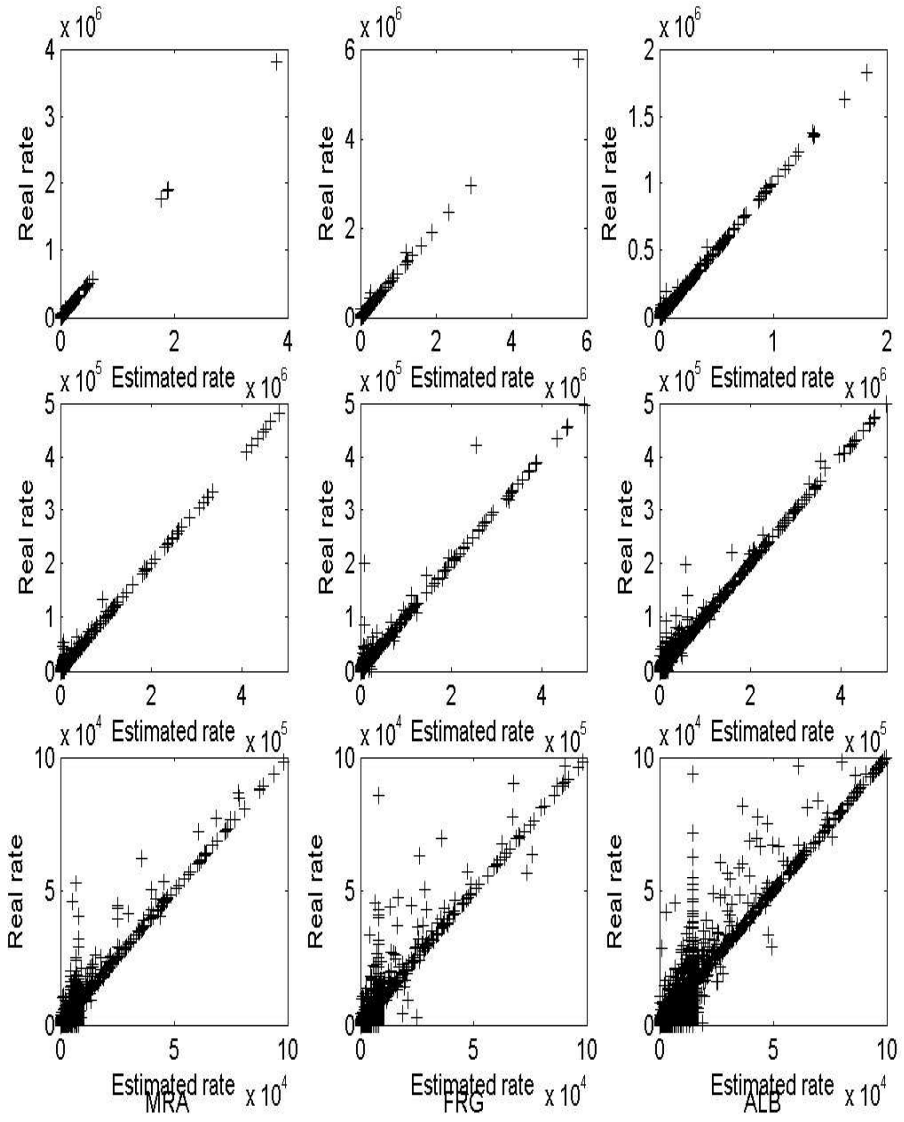

Figure 7.16: The estimated vs. real rates on MRA (left), FRG (middle) and ALB (right) trace. The top figures depicts all flows, the middle figures zoom in on flows with rates $\leq 500KB/sec$ and the bottom zoom in on flows with rates $\leq 100KB/sec$.

Flow ID (u)	$maxcwnd$	RTT (ms)	$packetSize$(B)
$u \in 1..20$	∞	$20 + 20u$	1500
$u \in 21..120$	30	$8u$	$650 - 5u$
$u \in 121..1000$	3	$8u + 40$	$60000/u$

Table 7.2: Characteristics of 1000 TCP connections.

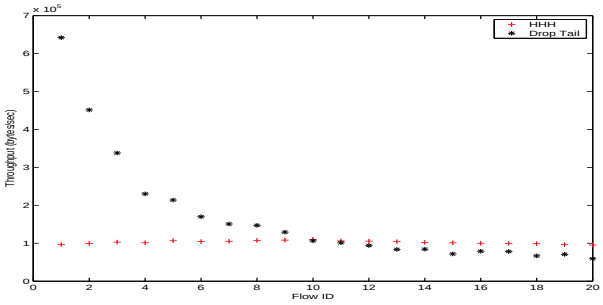

Figure 7.17: Throughput for flows from the first group under DropTail and HH.

groups increased significantly as is sketched in Table 7.3 (here $TOT(a..b)$ denotes share of bandwidth taken by flows $a, a+1, \ldots, b$). The utilization in the DropTail case is slightly higher than in the HH case which is almost equal to desired $util = 0.98$.

In the networking literature, the standard metric used for evaluating the level of fairness for certain bandwidth allocations is Jain's fairness index(JFI) [50]. For a set of k numbers $a_1, \ldots, a_k$, the JFI is the square of the ratio between arithmetic and quadratic mean of these k numbers:

$$JFI(a_1, \ldots, a_k) = \frac{(a_1 + \cdots + a_k)^2}{k(a_1^2 + \cdots + a_k{}^2)}$$

The last row in Table 7.3 contains the JFI for share of bandwidth for flows from the first group in both the DropTail and HH case. Figure 7.17 graphically illustrates the share of bandwidth for flows from the first group in both scenarios.

Metrics	DropTail	HH
$TOT(1..20)$	0.6736	0.4103
$TOT(21..120)$	0.2933	0.5539
$TOT(121..1000)$	0.0330	0.0356
$Utilization$	0.9916	0.9799
Average queueing delay(ms)	43.7	10.9
$JFI(1..20)$	0.5706	0.9984

Table 7.3: Experimental results for DropTail and HH queue.

7.7 Summary

In this chapter we present a heuristic method named RHE for realtime identification of flows that account for most of the bandwidth. If positive numbers $z_1, \ldots, z_n$ are taken with some heavy tailed probability distribution then $\sum_{i=1}^{n} z_i^2$ is dominated by the maximum of z_i. Assuming that z_i are Internet flow rates, efficient estimation of $\sum_{i=1}^{n} z_i^2$ allows us to design an algorithm that with high confidence identifies the highest rate flows. The presented method is highly scalable in the following sense(s)

- *Computationally.* Per packet processing requires only 3 comparisons, one hashing and up to 4 additions per stage. The design of RHE allows only a very few packets to higher levels and therefore the average number of stages per packet is just slightly above 1 (in all experiments presented here it is in range [1.3,1.6]). As a result of this, RHE can scale with line speeds of up to several tens of Gbps.

- *Memory space.* RHE needs a very small amount of memory. Experiments in Section 7.6.2 show high efficacy with just several kilobytes of SRAM memory on Gbps-traces. Low memory requirements of our scheme allow usage comparatively scarce on-chip SRAM which is highly desirable at highly loaded network processors.

- *Heavy-tail assumption.* RHE covers a wide range of flow rate distributions and is much more robust to heavy-tailed assumption, compared to MSF.

- *Number of active flows.*

Our main motivation for developing RHE was flow control. The proposed queue management scheme HH exploits the efficacy of RHE and controls fairness using a very small amount of memory. An important, highly nontrivial and open question that arises in the study of the "fair queueing" algorithms is:

(*) *What would the Internet flow rate distribution look like if routers employed fair queueing?*

In the context of a queueing scheme whose efficacy relies on the assumption of heavy-tailed traffic[19], the change in the flow rate distribution caused by fair queueing is an important metric that affects the performance of these algorithms. Our algorithms depend on the flow rate distribution only through the constant c_0 that defines the test given in Figure 7.5. Initial measurements indicate that Assumption 9.1 used for tuning c_0 is satisfied for all tested traces. However, analytical research is necessary to answer the question (*) en route to an efficient, scalable and robust "fair queueing" algorithm for the current and future Internet.

[19]The efficacy of most of Fair-queueing schemes depends on the flow rate distribution, see [62].

Part III

Utilization versus Queueing delays tradeoff

CHAPTER 8

Adaptive tuning of DropTail queues

Abstract - *Internet router buffers are used to accommodate packets that arrive in bursts and to maintain high utilization of the egress link. Such buffers can lead to large queueing delays. We propose a simple algorithm, Active Drop-Tail (ADT), which regulates the queue size, based on prevailing traffic conditions, to a minimum size that still allows for a desired (high) level of utilization. Packet level* ns2 *simulations are provided to show that Adaptive Drop-Tail achieves significantly smaller queues than current approaches at the expense of 1-2% of the link utilization.*

8.1 Introduction

Traditionally, router buffers have been provisioned according to the delay-bandwidth product (DBP) rule: namely, one chooses the buffer size as $q = B \times T$, where B is the rate of the link served by the router, and T is the "typical" [1] round trip time (RTT) experienced by connections utilizing the buffer. This amount of buffering allows for 100% utilization of the egress link under all traffic conditions. Following this rule, most router buffers are designed to have 100-250ms of buffering. This, together with the TCP mechanism of congestion avoidance, ensures high link utilization. In the last few years, several studies related to buffer sizing at a congested router have occurred [19, 2, 119, 5]. For example, it is claimed in [2] that the amount of buffer space needed for high utilization of a link is highly dependent on the number of active flows on the link. Namely, they claim that, if N is the number of active TCP flows, the buffer space required for 99% utilization of the link capacity is $B_{AKM} = \frac{B \times T}{\sqrt{N}}$.

Having small buffers is attractive as it reduces the amount of memory, required physical space, energy consumption, and price of the router. From our point of view, however, the main advantage

[1] In [19], it is suggested that in order to ensure full utilization of the link one should use $B \times T_h$ of buffering, where T_h is the harmonic mean of round trip times for all bottlenecked connections passing through the link. In this chapter we use the term "typical" RTT for harmonic mean T_h

of having small buffers is the reduction in queueing delays and jitter. In the current Internet the average number of hops on a random path is about 13 [113]. For a single flow with that many hops it is possible to expect several congested links on the path. Thus buffering of several hundreds ms at each router would imply very large queueing delays.

The $\frac{1}{\sqrt{N}}$ bound in [2] depends on the number of active users accessing a bottleneck link and the distribution of RTTs. Since, for any congested link, these quantities vary and are difficult to estimate, a network operator must provide as much buffer space as is necessary for the least possible number of active users in order to keep utilization high at all times. Assuming that a router uses a drop-tail queue of sufficient size to ensure high utilization in the low number of users regime, queueing delays will be much larger than necessary in regimes with a large number of users. Motivated by this observation we develop a simple algorithm, called Active Drop-Tail (ADT), which keeps the available buffer space as small as possible while maintaining a certain level of utilization. Thus, for a large number of users the available queue size will be low, and for a low number of users will be as large as necessary to achieve high utilization.

8.2 Active Drop-Tail algorithm

We propose a discrete-time queue management algorithm, called Active Drop-Tail (ADT), that tries to "find" the smallest queue size that allows the outgoing link to operate at a given desired utilization u. ADT has two stages: (i) estimating the (average) throughput of the link and (ii) adjusting the available buffer space once per sampling period based on this measurement. In order to estimate the throughput, at each sampling time we first compute the current throughput as the number of bytes enqueued normalized by the length of the sampling interval. We then compute a weighted average of the throughput. We will control the available buffer space based on this weighted average.

ADT maintains an internal variable q_{ADT} that corresponds to the number of packets that can be accommodated in the buffer. The basic idea is to modify q_{ADT} based on the estimated average throughput. If the average throughput is less than the desired utilization, q_{ADT} is increased to allow more buffering, yielding higher utilization. On the other hand, if the average throughput is greater than the desired utilization, we decrease q_{ADT} to regulate utilization to the desired level. For the purposes of adjusting q_{ADT} we use a Multiplicative Increase - Multiplicative Decrease (MIMD) strategy. While other control strategies are possible, our simulations show that a simple MIMD approach works well to reduce the queue size needed to maintain a certain level of utilization. Assuming that network conditions do not vary quickly[2], the MIMD parameter should be chosen to allow q_{ADT} to be halved/doubled in a few seconds up to one minute. ADT deals with arriving packets in the same fashion as drop-tail with q_{ADT} as the queue limit, i.e. if, at the moment of packet arrival, the queue has size less than $q_{ADT} - 1$, then the packet is enqueued. Otherwise it is dropped[3]. Pseudo-code

[2]Measurements [83] show that, on typical 150Mbps+ links, basic IP parameters (such as the number of active connections, proportion of TCP traffic, aggregate IP traffic, etc.) do not change dramatically in short time periods.

[3]We assume that the buffer is configured in packets. However one can easily change the algorithm to track q_{ADT}

describing the ADT algorithm is given in the table below. The parameters of ADT are as follows. ρ is an averageing parameter in $(0, 1)$ and is used to calculate the weighted average of the throughput. $c > 1$ is the MIMD parameter for controlling q_{ADT}. $SamplePeriod$ and u are the sampling period and desired utilization respectively.

ON EVERY PACKET ARRIVAL:
IF $(now - LastUpdate) > SamplePeriod$
. CURTHR := $\frac{NmbBEnq(now) - NmbBEnq(LastUpdate)}{now - LastUpdate}$
. THR := ρ CURTHR + (1-ρ)THR
. IF ($\frac{\text{THR}}{ServiceRate} < u$)
. $q_{ADT} := q_{ADT} \cdot c$
. OTHERWISE
. $q_{ADT} := \frac{q_{ADT}}{c}$
. END
. $q_{ADT} := min(q_{ADT}, SizeOfBuffer)$
. $LastUpdate := now$
END

In the above table $NmbBEnq(t)$ denotes the number of enqueued bytes in the interval $[0, t]$, and now is the current sample time. The parameter ρ is used to filter possible transient effects, such as a flash crowd or a sudden decrease in arrival traffic, and should be chosen such that weighted averaging is performed over several congestion epochs[4]. While in most cases congestion epochs last less than a few seconds, some extreme situations (e.g., high bandwidth-low number of users) might require a lower choice of ρ. Such situations can be easily identified and dealt with by adapting ρ online: due to space restrictions we do not describe the adaptation of ρ here other than to note that it can be done in a straightforward manner.

Comment 8.1 *In the case of a noncongested link, as long as the average arrival rate is less than the desired utilization u, the ADT algorithm will keep q_{ADT} at $SizeOfBuffer$; i.e. at the actual buffer size. However, in noncongested links queueing delays are negligible, and there is no need for ADT.*

Comment 8.2 *ADT is designed for loss based TCP. Delay-based TCP proposals, like Vegas or FAST, require a certain level of queueing delay as a congestion signal. Adaptation of available buffer space in such circumstances is not beneficial.*

and queue size in bytes instead packets.

[4]A congestion epoch is the time between two consecutive packet losses.

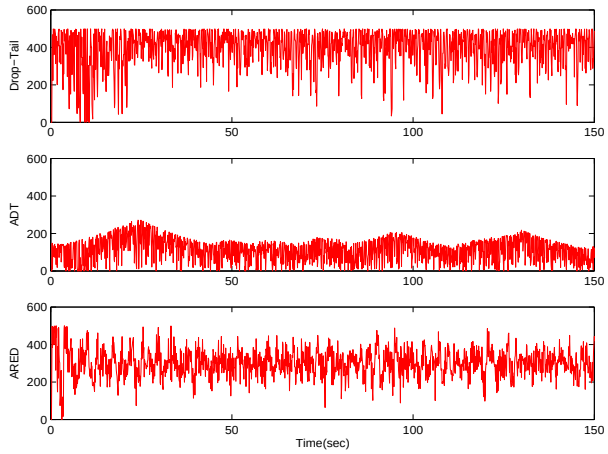

Figure 8.1: Queue sizes for Drop-Tail, ADT-0.99, and Adaptive RED

8.3 Evaluation

In this section we present *ns2* packet level simulation results on the performance of ADT[5]. Throughout this section we use the following set of ADT parameters: $\rho = 0.1$, $SamplePeriod = 0.3sec$, $c = 1.01$, and desired utilization $u = 0.99$.

A. ADT, Drop-Tail, and Adaptive RED : Our first *ns2* simulations are for 1000 TCP users with RTTs uniformly distributed in the interval 40 to $440ms$, and a packet size 1500B, all competing through a single bottleneck link with capacity $200Mb/sec$. Figure 8.1 shows the queue occupancy for a Drop-Tail (DT) queue with a buffer size of 500 packets ($30msec$ of buffering), an ADT queue, and an Adaptive RED [39] queue. The following table contains the average utilization (AU), loss rate (LR), average queueing delay (aQd), and Jain's fairness index (JFI) [50] for each of these three disciplines. In order to show how ADT adapts the available buffer space in the case of a congested reverse path we performed a simulation with 1000 TCP connections in both directions competing over ADT queues.

.	AU(%)	LR(%)	aQd(ms)	JFI
DT	99.99	4.73	23.78	0.58
ADT	99.03	4.87	6.66	0.56
ARED	100.00	4.90	17.99	0.79
ADT(R)	99.04	4.73	17.14	0.70

As we can see from the previous table, the utilization of Drop-Tail and Adaptive RED is 100% while the utilization of ADT is 99.03%. The loss rates of all three queueing disciplines are similar and are mainly driven by the congestion control algorithms of the users. A stated secondary goal of most RED-like schemes is to keep queueing delays small. However, having designed ADT specifically to minimize queueing delay, we see that the average queueing delay of ADT, in this example, is three times less than that of Adaptive RED. We note that even with reverse path traffic and heavy conges-

[5]ns2 software used in these simulations can be found at http://www.hamilton.ie/person/rade/ADT/.

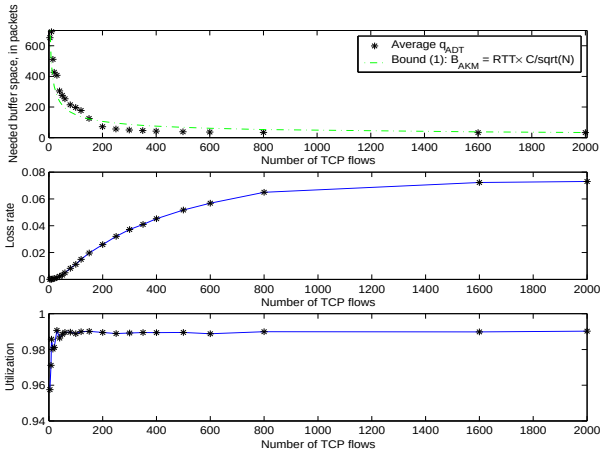

Figure 8.2: Average q_{ADT} (top), loss rates (middle), and utilization (bottom).

tion in both directions (i.e., ACK accumulation and larger bursts) ADT is able to adapt the available queue size in order to achieve the desired utilization (although because of ACK losses goodput is significantly reduced).

B. Queue size, number of users, and loss rate : Here we investigate the relationship between queue size, number of users, and loss rate. In order to compare our results with the $\frac{1}{\sqrt{N}}$ bound from [2] we consider N TCP flows with RTTs uniformly distributed in $[80, 100]ms$[6] with a packet size of $1500B$. All flows compete via a bottleneck link of $200Mb/sec$ with an ADT queue. We vary N from 4 to 2000 and evaluate loss rate and average q_{ADT}. In Figure 8.2 (top) we compare the average queue size for a fixed number of users with the estimate given in [2]. We observed close fit to the theoretical bound from [2]. As we can see from Figure 8.2 (middle) the loss rate increases slowly with N. This is a consequence of the time-out mechanism of TCP.

C. ADT and a varying number of users : We now allow the number of users to vary with time[7]. We consider a bottleneck link with capacity $100Mb/s$ where the number of active TCP users varies from 10 to 500 and back to 10 again, with one user becoming active or inactive every two seconds as depicted in Figure 8.3 (top). The RTT's are randomly chosen uniformly from the interval 10 to $300ms$. As we can see in Figure 8.3 (middle and bottom), q_{ADT} falls below 40 packets for large numbers of users, while for a small number of active users the required queue size is over 400 packets.

D. Selection of parameters : ADT is highly robust to the choice of its parameters. To illustrate this we ran a set of N TCP connections, over a $100Mb/s$ link and varied the ADT parameters as

[6] In deriving their $\frac{1}{\sqrt{N}}$ bound the authors of [2] assume uniform RTTs. For this reason we chose the RTTs in the range $[80, 100]ms$, to emulate the condition of "almost the same" RTT.

[7] As we deal with slowly varying environments, in our simulation we vary the number of active users in a continuous manner rather than abruptly.

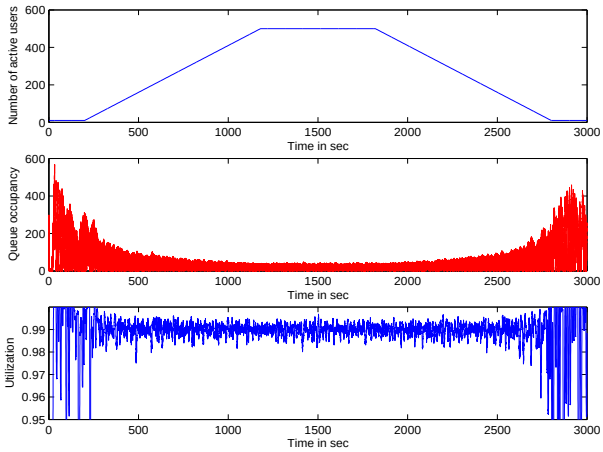

Figure 8.3: Utilization and q_{ADT} with changing number of active TCP users.

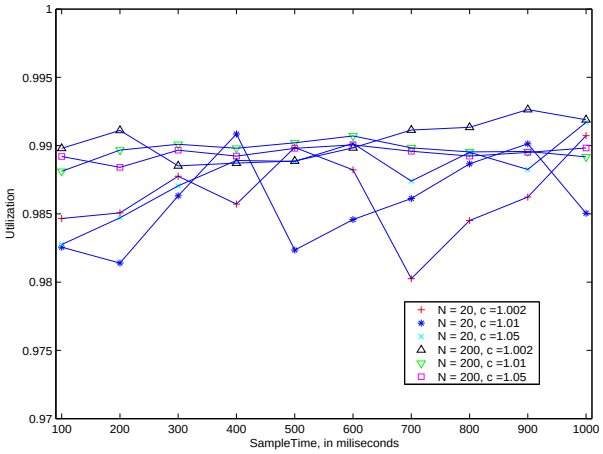

Figure 8.4: Average utilization (5 minutes) for different choices of parameters.

follows: $SampleTime$ in range $[100, 1000]msec$; $c \in \{1.002, 1.01, 1.05\}$. We fixed the weighted average factor $\rho = 0.1$. We evaluated two cases, low number of users: $N = 20$, and high number of users: $N = 200$. Recall, that the performance goal of ADT is to regulate utilization at a prescribed level; in our case $u = 0.99$. The ability of ADT to achieve this objective for different choices of parameters is depicted in Figure 8.4.

8.4 Summary

In this chapter we have presented a simple algorithm, ADT, for keeping queueing delays small, while maintaining a certain desired utilization. Via packet level simulations, we have shown that, for networks serving a large number of TCP flows, by allowing a 1-2% underutilization of a bottleneck link, we can realise smaller average queueing delays than in other queueing disciplines. We point out that this algorithm is easy to implement in current routers, requiring a minimum amount of processing power. Finally, we note that ADT strives to adjust the available buffer space to accommodate bursts

through the network buffers irrespective of their generating mechanisms.

CHAPTER 9

How expensive is link utilization

Abstract - *Understanding the relationship between queueing delays and link utilization for general traffic conditions is an important open problem in networking research. Difficulties in understanding this relationship stem from the fact that it depends on the complex nature of arriving traffic and the problems associated with modelling such traffic. Existing AQM schemes achieve a "low delay" and "high utilization" by responding early to congestion without considering the exact relationship between delay and utilization. However, in the context of exploiting the delay/utilization tradeoff, the optimal choice of a queueing scheme's control parameter depends on the cost associated with the relative importance of queueing delay and utilization. The optimal choice of control parameter is the one that maximizes a benefit that can be defined as the difference between utilization and cost associated with queuing delay. We present two practical algorithms, Optimal Drop-Tail (ODT) and Optimal BLUE (OB), that are designed with a common performance goal: namely, maximizing this benefit. Their novelty lies in fact that they maximize the benefit in an online manner, without requiring knowledge of the traffic conditions, specific delay-utilization models, nor do they require complex parameter estimation. Packet level ns2 simulations are given to demonstrate the efficacy of the proposed algorithms and the framework in which they are designed.*

9.1 Introduction

Measurements from a number of sources suggest that traffic generated by TCP users accounts for 85-95% of the Internet traffic [30, 34]. Van Jacobson's congestion control algorithm [49] is one of the main reasons for the robustness of the Internet and the prevention of the congestion collapse over last two decades. Given the success of TCP and the stability of the current internet it is unlikely that other, conceptually different, transport protocols will replace TCP in the near future.

Most Internet routers use FIFO Drop-Tail buffers. Current router buffers are generally sized by the rule-of-thumb given in the Villamizar&Song paper [115]: router buffers require approximately

space for $B = \overline{RTT} \times C$ packets, where $\overline{RTT}$ is the "average" round trip time for connections that use the link and C is capacity of the link. Following this rule, most router buffers are designed in such a fashion that they result in up to $100ms$ to $250ms$ of queueing. This, together with TCP's mechanism of congestion avoidance, serves to ensure a high link utilization. In the last few years several studies related to buffer sizing for congested routers have appeared. It is claimed in [2] that the amount of buffer space required for high link utilization can in some circumstances be far less than that suggested by the Bandwidth-Delay-Product rule. However, it is also shown in this paper that the required buffering highly depends on the number of active flows using the link. In particular, briefly, assuming a single congested link topology, and N homogenous, unsynchronized, long TCP flows, with a "typical"[1] round trip time $\overline{RTT}$, then the buffer space required for a link utilization of $u \cdot C$ is given by:

$$B_{AKM}(u) = A(u)\frac{\overline{RTT} \times C}{\sqrt{N}}. \tag{9.1}$$

Here, $A : (0,1) \mapsto (0,\infty)$ is a real function for which $A(0.99) \approx 1$ and $A(0.9999997) \approx 2$. One should note that, although the bound (9.1) is derived in the context of drop-tail queues, it is also applicable to other AQM schemes as well. Namely, in order to ensure utilization of $u \cdot C$, one needs a physical buffer space for accommodating packets of N unsynchronized TCP flows, given by (9.1).

Although the bound (9.1) yields important theoretical insights into the relation between link utilization and the required buffering it is not immediately applicable to size buffers in the real Internet routers for a number of reasons. Firstly, the bound (9.1) depends on the number of active users that are bottlenecked at the link, as well as their RTT distribution. These quantities vary, and are also usually hard to estimate. Secondly, the mathematical assumptions used in deriving (9.1) are not realistic and do not take into account the various and variable traffic mixes possible, the level of synchronization, the existence of non-TCP traffic, etc. Most importantly, while it is useful to know that delay and utilization are related in some manner, it is not immediately clear how to utilize this relationship in a meaningful manner. Specifically, an important practical question that arises in the queue management design is:

(*) "What is more important: low queueing delays or high utilization?"

To illustrate this question, say that for a given traffic mix, the relation between average queueing delays (aQd) and utilization (u) is given in Table 3:

In this table, t is some parameter that defines the queue management scheme. For example, t can be interpreted as available buffer space, or the per packet drop probability. Now, the important practical question is, which t should a network operator chose under this traffic mix? The answer to

[1] It is suggested in [19] that in environments without synchronization, one should use harmonic mean of the RTTs of the active connections as "typical" RTT.

t	t_1	t_2	t_3	t_4
$aQd(t)(sec)$	0.1	0.02	0.005	0.001
$u(t)$	1.00	0.98	0.90	0.60

Table 9.1: Synthetic example of $aQd(t)$ and $u(t)$ for 4 different possible choices of parameter t.

this question depends on the relative importance of utilization and queueing delays. To formalize this notion we can define the benefit $B(t)$ as the difference between utilization and cost a network operator is willing to pay that is an increasing function of queueing delays. Having defined this cost function, which specifies formally the tradeoff between utilization and delays, the problem then becomes that of choosing the optimal queueing scheme parameter t. This is a problem of maximizing the benefit $B(t)$ and can be solved in an optimization framework. In the Section 9.3 we will formalize the framework described above.

As we already noted, although there exist a number of mathematical models [2, 5, 6, 19, 27] that can give us some insight into the delay-utilization relationship, it appears extremely hard to evaluate this relationship for general traffic environments. Moreover, even if one has reasonably accurate theoretical predictions between delay and utilization for a given traffic mix, these predictions would certainly be a function of traffic parameters such as the number of active flows, the number of active TCP flows, the proportion of TCP traffic, per flow responsiveness, the distribution of round trip times, the level of loss synchronization, the level of congestion on other links in the network, etc. From a measurement point of view, estimation of these quantities is very demanding and requires significant amount of computational and physical resources [31, 52, 51, 114].

(**) "It is highly nontrivial to predict or estimate in real-time, relation between queueing delays and utilization, for congested high-speed Internet links."

Having (**) as the starting point, we will try to maximize overall the benefit $B(t)$ online rather than estimating the delay-utilization relationship. Here we propose two practical queue management schemes that have the same common goal: namely, maximizing the benefit $B(t)$, by controlling the parameter t online. In the first scheme, called Optimal Drop-Tail (ODT), t is the maximum available buffer space in the FIFO Drop-Tail queue, while in the second scheme, called Optimal BLUE (OB), the parameter t is the probability that an arriving packet is dropped.

The rest of the chapter is organized as follows. Existing models of the delay-utilization relationship and AQM schemes are disscused in the Section 9.2. In Section 9.3 we define the optimization problem to be addressed and provide a theoretical analysis of the possible approaches to solving it. The queue management schemes Optimal Drop-Tail and Optimal BLUE are introduced in Sections 9.4 and 9.5 respectively. In Section 9.6 we provide detailed packet level ns2 simulations to show behavior of both ODT and OB. Finally we summarize our findings and discuss open issues in Section 9.7.

9.2 Previous work

Within this section we discuss existing models for the delay-utilization relationship as well as Active Queue Management schemes that aim to reduce queueing delays.

Models. Over last few years, a number of models have been proposed to estimate the relationship between queueing delays and utilization. Most of these consider the problem of sizing FIFO Drop-Tail buffers for achieving a certain level of link utilization under the assumption of a single bottleneck link servicing N long TCP transfers. In [2] the authors give a $O(\frac{1}{\sqrt{n}})$ bound (9.1). Another bound of this type is given in [5]. It is showed there (under the assumption of N unsynchronised homogenous TCP users with the same round trip time $\overline{RTT}$) that to achieve 100% of utilization one needs a buffer size of :

$$B_{AAP} = \frac{(\overline{RTT} \times C)^2}{2N(4N-1)^2} \approx \frac{(\overline{RTT} \times C)^2}{32N^3}. \tag{9.2}$$

The bound (9.2) is derived from a fluid model which assumes full unsynchronization (ie. only one source loses packets per congestion event). The authors [27] are even more optimistic and claim that if TCP users have bounded the maximal congestion window $maxcwnd\_$, then the needed queue size for achieving high throughput is of form of $O(log(maxcwnd\_))$. Such an approach assumes a low maximum $cwnd\_$, or equivalently a high loss rate, and does not allow high speed connections [37, 58].

Another important problem that could result from extra small buffers is the problem of extreme unfairness between flows. To see this, consider a congested link, without per-flow management, with a loss rate p and an average queueing delay d_0. From the square root formula [87], the sending rate (in packets per second) of flow with base RTT given by RTT_b is equal to

$$r = \theta \frac{1}{\sqrt{p}(RTT_b + d_0)}. \tag{9.3}$$

Bearing in mind that a typical range of base RTT's spans from a few microseconds to several seconds, and assuming that d_0 is in the range of few microseconds (as suggested in all-optical framework in [119, 27]) then this would imply unfairness between competing connections in the range $1 : 10^5$. In such situations, long-RTT connections would suffer heavily, and will not have any benefit in reducing queueing delays. Allowing a few milliseconds of queueing delays would decrease unfairness level to range $1 : 100$, which is several order of magnitude more acceptable.

A very different approach has been presented in [19]. Namely, they suggest that an important performance metric in dimensioning router buffers is loss rate. By exploiting the window based nature of the TCP congestion control algorithm, they provide bound for buffer sizes that ensure loss rates lower than certain, prescribed, value.

As we already have noted, the insight obtained by theoretical analysis of the proposed models is highly valuable for understanding the problem of interest, but there is a significant gap between them and their application in real Internet links.

AQMs. Active queue management generally stands for a mechanism that has, as its ultimate objective, of keeping utilization as high as possible without incurring "large queueing delays". In TCP environments, the main cause of low utilization is synchronization. By breaking synchronization and responding early (but not "too early"), AQM schemes like RED, BLUE, PI and AVQ, aims to achieve high throughput together with low delays. However, no existing AQM takes in account the interaction between queueing delays and utilization. Low queueing delays and high utilization are mainly an ad hoc consequence of early response, rather than a formal performance goal.

9.3 Optimization framework

Lets go back to the example from the introduction illustrated in Table 9.1. For simplicity, assume for the moment that the parameter t is the available buffer space on the congested FIFO Drop-Tail queue; for buffer size equal to t_1 the average queue delay is $100ms$ and the utilization is 100%, for buffer size equal to t_2 the average queue delay is $20ms$ and the utilization is 98%, and so on. Which choice of t is optimal (among 4 possible in this example), depends on the "importance" of low queueing delays. To formalize this, one can identify the "importance" by the relative price between utilization and queueing delays. Let $P : [0, \infty) \mapsto [0, \infty)$ be a function that specifies relative price between utilization and delays. In other words, a queueing delay of d seconds has same price as utilization of $P(d)$. Formally, a price function is any function that satisfies the following definition.

Definition 9.1 *The function $P : [0, \infty) \mapsto [0, \infty)$ is a price function if it is twice differentiable, increasing and convex. In other words if:*
(a) $\forall d \in [0, \infty)\ \exists P''(d)$
(b) $\forall d \in [0, \infty)\ P'(d) \geq 0$
(c) $\forall d \in [0, \infty)\ P''(d) \geq 0$

The following simple technical lemma will be useful in later discussion.

Lemma 9.1 *Let $E \subset R$ be a segment ($E = [a_1, a_2]$ for some real a_1 and a_2). If $f : E \mapsto [0, \infty)$ is a twice differentiable, convex function and P an arbitrary price function, then $P \circ f : E \to [0, \infty)$ is convex as well.*

Proof. The proof is straightforward. It is enough to prove that $(P(f(t))'' \geq 0$ for all $t \in E$.

$$(P(f(t))'' = (P'(f(t)) \cdot f'(t))' =$$

$$P''(f(t)) \cdot (f'(t)))^2 + P'(f(t)) \cdot f''(t) \geq 0.$$

□

Having defined a price function, the overall benefit, in the case given by the parameter t, can be written in the form:

$$B(t) = u(t) - P(aQd(t)). \tag{9.4}$$

Comment 9.1 *Notion similar to the benefit $B(t)$ is introduced in [5] for t representing the available buffer size.*

The definition of benefit allows us to define a notion of optimal choice, as the value of t that maximizes the benefit. Formally:

Definition 9.2 *For a given price function P and set $\mathcal{T}$ of possible choices of t, an optimal Delay-Utilization(D-U) choice is any t_0 such that*

$$B(t_0) = \max\{B(t) \mid t \in \mathcal{T}\}, \tag{9.5}$$

if the maximum on the right hand side exist.

Comment 9.2 *In the rest of the chapter we will consider exclusively the following two cases: (1) the set $\mathcal{T}$ is finite; then the maximum in (9.5) clearly exists; (2) the set $\mathcal{T}$ is closed and bounded in metric space[2] R, and $B : \mathcal{T} \mapsto \mathcal{R}$ is continuous function - in this case the maximum in (9.5) exists as well.*

In the example given in Table 9.1, if we completely ignore the importance of low queueing delays, by setting $P(d) \equiv 0$ for all d, then the optimal D-U choice is given by t_1, as this maximizes the benefit $B(t) = u(t) - P(aQd(t)) = u(t)$ on the set $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$. For the price function $P(d) = 5 \cdot d$, the optimal D-U choice is t_2, and for the price function $P(d) = 100 \cdot d$, the optimal D-U choice is t_4. Linear price functions $P(d) = \gamma \cdot d$, are a simple way of specifying the relative price of queueing delays in sense that $a\%$ of utilization is equivalent with $\frac{a \cdot 0.01}{\gamma}$ $sec = \frac{10 \cdot a}{\gamma} msec$ of queueing delays. Thus, a high γ gives high importance of low queueing delays, while a low γ gives priority to high utilization.

Throughout this chapter we assume:

Assumption 9.1 *Under static traffic conditions the overall benefit given by (9.4) is a concave function of t.*

In the previous discussion, we have referred to t as a parameter which defines a queueing scheme. In later sections we will be concerned with the following two cases:

Case A. t_S defines the available buffer space, and packets are queued in Drop-Tail queue of size t_S.

Case B. t_p defines the drop probability, and each packet is dropped on arrival with probability t_p.

[2] With euclidian distance.

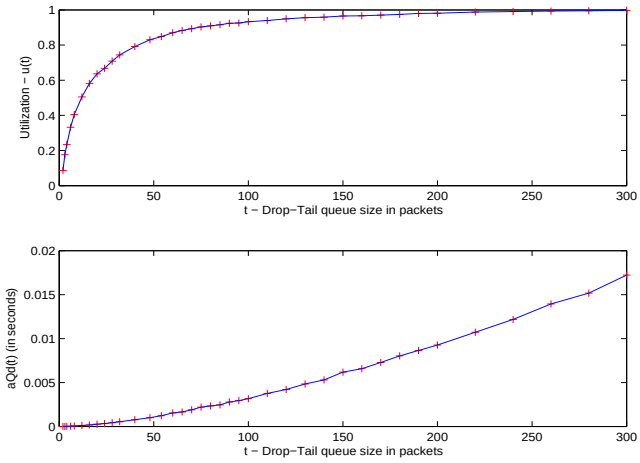

Figure 9.1: t_S : Drop - Tail queue size vs. $u(t_S)$ (top) and $aQd(t_S)$ (bottom).

In the next two sections we present two queue management algorithms: Optimal Drop-Tail (ODT) and Optimal BLUE (OB). Both of these have a common performance goal: to maximize the overall benefit given by (9.4). ODT achieves this goal by adaptation of the available buffer space, while OB tunes the drop probability t_p in order to maximize $B(t_p)$.

Comment 9.3 *Other approaches might be possible as well. For example, in the framework of Virtual Queue (VQ) schemes [41], t can be seen as virtual queue capacity. Following our optimization methodology, one can design an virtual queue management algorithm by adapting the virtual queue capacity subject to the performance goal that maximize the benefit $B(t)$ rather than keeping the utilization at certain level γ as it has been done in the Kunniyur and Srikant's AVQ [70] algorithm.*

In the rest of this section we discuss the validity of the Assumption 9.1, the existance/uniqueness of optimal D-U choice and possible strategies for online solving optimization problem (9.5).

Assumption 9.1 is very hard to formally check. In a theoretical framework, this would require accurate models of various traffic mixes, and as we already noted, modelling such complex environments is highly nontrivial. Some results related to the convex relationship between utilization and buffer size in non-elastic traffic environments are developed in [67, 68]. However, our empirical observations suggest that for the traffic mix that is consisted from the static number of TCP and UDP flows, Assumption 9.1 holds in both Case A and Case B. To illustrate this we run two sets of packet level ns2 simulations, and evaluate the utilization and the average queueing delay.

Simulation A. In the first set of simulations, we consider a FIFO Drop-Tail queue with a service rate of 10MBps and size of t_S packets. This queue is shared by 50 TCP connections with round trip times uniformly distributed in range $[20, 220]ms$ and with packet size of 1000 bytes. We varied t_S from 1 to 300 packets, and plotted $u(t_S)$ and $aQd(t_S)$ in Figure 9.1. The convexity of $aQd(t_S)$, by Lemma 9.1, implies the convexity of $P(aQd(t_S))$, and this together with concavity of $u(t_S)$ implies concavity of the benefit $B(t_S)$, for arbitrary price function P.

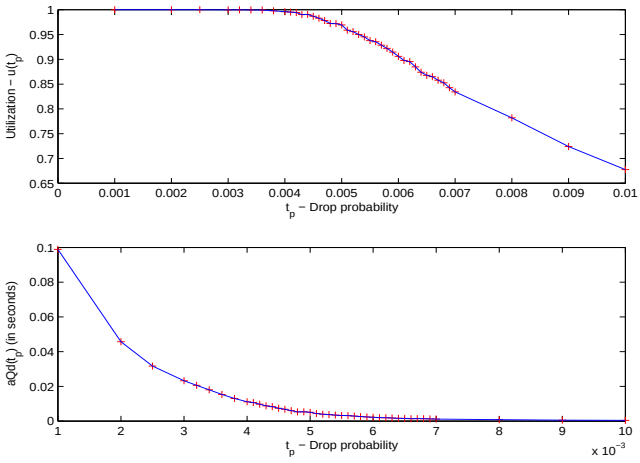

Figure 9.2: t_p : Drop probability vs. $u(t_p)$ (top) and $aQd(t_p)$ (bottom).

Simulation B. In the second set of simulations, we consider a queue with service rate of 10MBps, of size of 10000 packets (so that no packet is dropped because of overflow), such that every packet is dropped with probability t_p on the arrival to the queue. This queue is shared by the same set of 50 TCP connections as in first simulation. We varied t_p in the range $[10^{-3}, 10^{-2}]$, and plotted $u(t_p)$ and $aQd(t_p)$ in Figure 9.2. As in the previous case, convexity of $aQd(t_p)$, together with concavity of $u(t_p)$ implies concavity of the benefit $B(t_p)$, for arbitrary price function P.

Convex optimization has been widely employed in the networking community. For example, optimization methods are essential in the analysis and design of distributed congestion control algorithms [102, 57]. In our case, we need efficient algorithms for solving (9.5). A standard control strategy for solving (9.5) is given by

$$\dot{t} = g(t) \cdot B'(t), \;\; g(t) \geq \epsilon > 0, \tag{9.6}$$

or its discrete version:

$$t(k+1) = t(k) \left(1 + g(k) \frac{B(t(k)) - B(t(k-1))}{t(k) - t(k-1)} \right), \;\; g(k) \geq \epsilon > 0, \tag{9.7}$$

The problem with employing one of these strategies in the present case is twofold. First, as we do not have explicit relationship between t and $B(t)$, we can not instantly compute the derivative $B'(t)$. Second, the signal to noise[3] ratio in measuring of both queueing delays and utilization can be very large especially in the neighborhood of the solution of (9.5). This would potentially imply low confidence in the estimation of $B'(t)$ in the neighborhood of the solution of (9.5). One approach to this problem is the use of larger sampling times and low pass filter for smoothing out the results. To illustrate the level of noise one can expect in queue measurements we ran the following ns2 simulation.

[3]By definition $B(t)$ is function of average utilization $u(t)$ and average queueing delay $aQd(t)$. Instantaneous utilization (queueing delay) can be seen as random variable that is sum of $u(t)$ ($aQd(t)$) and appropriate zero mean random variable, that we refer to as noise.

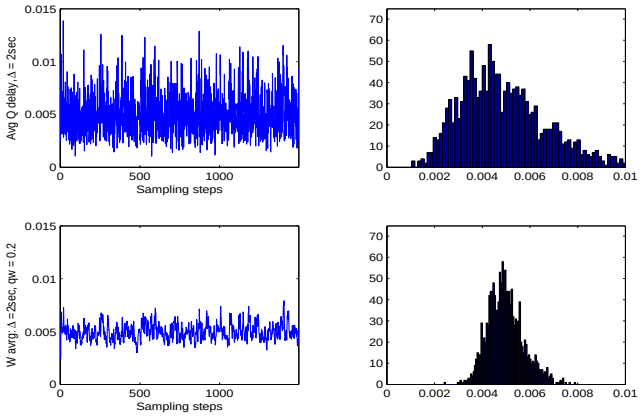

Figure 9.3: $d_\Delta(k)$ (top), and $\overline{d}_\Delta(k)$ (bottom); $\Delta = 2sec$, $qw = 0.2$.

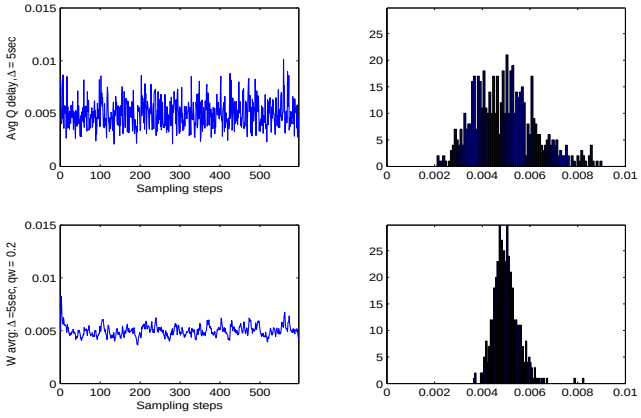

Figure 9.4: $d_\Delta(k)$ (top), and $\overline{d}_\Delta(k)$ (bottom); $\Delta = 5sec$, $qw = 0.2$.

Simulation C. We consider the same setup of 50 TCP flows competing over a link with service rate of 10MBps, as in Simulation B. We drop each packet on arrival with constant probability $t_p = 0.005$. The quantities of interest are the following: $d_\Delta(k)$ is the average queueing delay in the k-th sampling period $[(k-1)\Delta, k\Delta]$, and the weighted average $\overline{d}_\Delta(k)$, with weighting factor qw, given by:

$$\overline{d}_\Delta(k) = (1 - qw) \cdot \overline{d}_\Delta(k-1) + qw \cdot d_\Delta(k).$$

Figures 9.3 and 9.4, depict measured values of $d_\Delta(k)$ (left top) and $\overline{d}_\Delta(k)$ (left bottom) from the simulation, for $\Delta = 2sec$ and $\Delta = 5sec$. The right hand side plots show the histograms of quantities depicted on left plots.

We observed a similar level of noise in measurements of link utilization. We also ran the same simulation over Drop-Tail instead of constant drop rate queues, and the level of noise in queueing delay and utilization measurements are approximately of the same order of magnitude as in Simulation C.

In the next two sections we will present two novel queue management algorithms for which the performance goal is given by maximization of the benefit defined by (9.4). The first one, Optimal Drop-Tail(ODT), achieves this goal, by adaptation of t_S - the available Drop-Tail queue space, while the second one, Optimal BLUE(OB), adapts t_p - the per packet drop probability. Both ODT and OB

use a form of MIMD[4] algorithm, where at the end of each sampling period control variable t (that is t_S or t_p) is updated by the rule:

$$t(k+1) = t(k) \cdot m(k). \tag{9.8}$$

where $m(k)$ is either α or $1/\alpha$, for some α greater 1, and $m(k)$ determines direction in which t should go. Algorithms of this type cannot settle to constant value, but rather continuously search for the point on the grid $\mathcal{T}_\alpha = \{t(0) \cdot \alpha^n,\ n \in Z\}$, that maximizes $B(t), t \in \mathcal{T}_\alpha$. However, choosing α to be close to 1, the point on the grid $\mathcal{T}_\alpha$ that maximizes $B(t)$ will be close to the global optimal value.

9.4 Optimal Drop-Tail

Drop-Tail queueing is the scheme that is employed by the majority of the current Internet routers. Drop-Tail queues have a single parameter S that determines the available size[5] of the queue, and is usually manually configured by a network operator. The following simple observation is the basis for the spectrum of algorithms presented here. That, by controlling the value t_S - available queue size[6], the objective of achieving certain performance goals, given in terms of utilization and queueing delays, can be met.

By controlling t_S, one can control both utilization and queueing delays.

For example, if the performance goal is given by keeping the average utilization at a certain level λ, one can design a strategy for achieving that goal by controlling t_S. Similarly, if the performance objective is keeping the average queueing delay (at the times of congestion) at a prescribed level d_0, another control strategy can be designed for solving that problem. At this point we should note that by controlling t_S one can control not only utilization and queueing delays, but also other (important) performance metrics such as jitter and loss rate. Embedding them into an optimization framework could be done in straightforward manner.

Following the delay-utilization optimization framework developed in the previous section, the performance goal of interest will be the maximization of the benefit $B(t_S)$. We proceed by presenting an ODT algorithm, a strategy with that performance goal.

The ODT algorithm controls the variable t_S that represent available queue size. In other words, on every packet arrival, a packet is dropped, if by its enqueuing to the existing queue, the queue length would be greater than t_S, otherwise the packet is enqueued. The value t_S is updated once per sample time period (Δ) in the following manner:

[4] Multiplicative Increase Multiplicative Decrease.

[5] Size can be configured in either bytes or packets.

[6] We write t_S instead S to distinguish cases between variable queue size (for which we use t_S) and constant queue size (for which we use S).

$$t_S(k+1) = t_S(k) \cdot m(k), \tag{9.9}$$

where $m(k)$ is defined by:

$$m(k) = \alpha, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_S(k) - t_S(k-1)} \geq 0,$$

$$m(k) = \frac{1}{\alpha}, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_S(k) - t_S(k-1)} < 0.$$

Here, $\alpha > 1$ is a constant parameter, close to 1. The choice of α determines the responsiveness of the algorithm. Since t_S is either multiplied with α or divided by α, in each step k, $t_S(k) = t_S(0) \cdot \alpha^{l(k)}$, for some integer $l(k)$. By $\hat{B}(l(k))$ we denote the estimated value of $B(x)$ at the point $x = t_S(k) = t_S(0) \cdot \alpha^{l(k)}$. Algorithms of this type can be seen as a version of (9.7) that do not allow arbitrarily small steps. Strategies of the form of (9.7) are inappropriate in our problem for the following two reasons. First, any algorithm of type (9.7) that allows very small changes in the parameter t_S would suffer from a high noise to signal ratio around global maximum of $B(t_S)$, and would require long time for accurate estimation of B in the neighborhood of the global maximum. Second, it has been proved in [90], using information-theoretical techniques, that any algorithm for finding an optimum using noisy observations of a benefit function has slow expected convergence. Namely, $O(\epsilon^{-4})$ queries have to be made before one can ensure ϵ-accuracy in the estimation of the optimum x^*. Under dynamic, Internet-like traffic conditions, frequent (small) changes of the traffic patterns might not allow such algorithms to converge, and can potentially cause undesirable large oscillations.

Algorithms of the form of (9.9) that do not converge to the certain value, but rather continuously search for the optimal value have been extensively used in the networking literature. Examples of such algorithms are AIMD[7] $cwnd_-$ control in TCP [49], AIAD algorithm for controlling the drop probability in BLUE [36] as well as MIMD algorithm for the adaptation of RED parameters in Self-Configuring RED [35].

Now we proceed by describing the technique for estimation of $\hat{B}(l(k))$.

In every sampling interval, we have that $t_S(k) = t(0)\alpha^{l(k)}$ for some integer $l(k)$, and this integer is uniquely determined. In other words the mapping between the set of all possible values of t_S, $\mathcal{T}_\alpha$, and set of integers given by $t(0)\alpha^m \mapsto m$ is bijective. This allows us to use the history at each possible value of $t_S(k)$ independently for computing the estimate $\hat{B}(l(k))$. For each possible integer m we will keep the value: $n_-(m)$ which is the number of sample intervals within previous W_0 sampling intervals for which $l(k)$ was equal to m. Thus at sampling interval k we have:

$$n_-(m) = \#\{k_1 : \ k_1 \in (k - W_0, k], \ l(k_1) = m\}.$$

Denote by $\tilde{B(k)}$ the instantaneous benefit in the k-th sampling interval : $[(k-1)\Delta, k\Delta]$, i.e.:

[7] Additive Increase Multiplicative Decrease.

$$\tilde{B(k)} = \tilde{u}(k) - P(\tilde{d}(k)),$$

where $\tilde{u}(k)$ and $\tilde{d}(k)$ are the instantaneous utilization and queueing delay in the k-th sampling interval respectively, we will estimate $\hat{B}(l(k))$ using the following weighted average:

$$\hat{B}(l(k)) = \hat{u}(k) - P(\hat{d}(k)), \tag{9.10}$$

where:

$$\hat{u}(l(k)) = \frac{1}{n_-(l(k))}\hat{u}(l(k)) + (1 - \frac{1}{n_-(l(k))})\tilde{u}(k),$$

and

$$\hat{d}(l(k)) = \frac{1}{n_-(l(k))}\hat{d}(l(k)) + (1 - \frac{1}{n_-(l(k))})\tilde{d}(k).$$

The rationale for the estimation technique given by (9.10) is the following. If some m has a large number of occupancies in the recent history of $l(k)$ then this indicates that the corresponding $t_S = t_S(0)\alpha^m$ is close to the optimal value and a finer estimation of the benefit is required. If m has a small number of appearances in the previous W_0 sampling periods we need less accuracy, but a faster response to changes in the traffic conditions. If m had no appearances in the previous W_0 sampling periods, all history will be forgotten in the future estimation of $\hat{B}(t_S(0)\alpha^m)$.

We use one additional correction step, that is rarely needed under static conditions but is important for improving accuracy for new values t_S that have had very few visits in the recent history. Namely, we do not allow non-monotonicity in the estimation of $\hat{u}$ and $\hat{d}$. Set $i = l(k)$. Then:

if $\hat{u}(i) > \hat{u}(i+1)$ or $\hat{u}(i) < \hat{u}(i-1)$ then

$$\hat{u}(i) = \frac{n_-(i-1)\hat{u}(i-1) + n_-(i)\hat{u}(i) + n_-(i+1)\hat{u}(i+1)}{n_-(i-1) + n_-(i) + n_-(i+1)}, \tag{9.11}$$

if $\hat{d}(i) > \hat{d}(i+1)$ or $\hat{d}(i) < \hat{d}(i-1)$ then

$$\hat{d}(i) = \frac{n_-(i-1)\hat{d}(i-1) + n_-(i)\hat{d}(i) + n_-(i+1)\hat{d}(i+1)}{n_-(i-1) + n_-(i) + n_-(i+1)}. \tag{9.12}$$

Recall that $n_-(m)$ represents the number of sampling periods k, in the recent history (driven by sliding window of size W_0), in which $l(k)$ was equal to m. We have a confidence in the estimate of $\hat{u}$ and $\hat{d}$ that is roughly proportional to n_-. This is exploited in (9.11) and (9.12).

Note that we need to store four sequences, *aug* (augmenting sequence of length W_0 for computing n_-), n_-, $\hat{u}$ and $\hat{d}$, for implementing this estimator. The size of the sequence *aug* is W_0, while the size of sequences n_-, $\hat{d}$ and $\hat{u}$ depend on the choice of α in a logarithmic fashion: if the physical buffer space is S_0 bytes, then the size of sequences $\hat{d}$, $\hat{u}$ and n_- should be $\log_\alpha S_0$ to cover the range from 1 to S_0 bytes. For example, with $\alpha = 1.01$, a sequence of size 2000 will cover the range from 1 to $S_0 = 1.01^{2000} \approx 439286205$ bytes with a granularity of 1%.

From the computational point of view, ODT is a very light scheme. Namely, for the computation of the instantaneous utilization $\tilde{u}(k)$, and the instantaneous queueing delay $\tilde{d}(k)$, we use three counters:

$P(d)$	price function
Δ	length of sampling period
α	MIMD parameter
W_0	history window

Table 9.2: Parameters of ODT, OB.

$NmbBytes$ (number of processed bytes since last update), $NmbArrivals$ (number of packet arrivals since last update) and $TotQDelay$ (sum of potential queueing delays for all $NmbArrivals$ packets since last update)[8]. All of these three counters are updated once per packet and these updates are the only per packet operations required.

The input parameters for ODT are given in the Table 2. While in general $P(d)$ can be an arbitrary function that satisfies Definition 9.1, throughout this chapter we will mainly use functions that are linear in d:

$$P_\gamma(d) = \gamma d,\ \gamma > 0. \tag{9.13}$$

If we restrict ourselves to price functions of this form then the parameter $P_\gamma(d)$ can be specified by a single scalar γ. A higher value of γ assigns more importance to low delays and vise versa. The sampling period time Δ should be chosen to cover several "typical" round trip times, in order to allow traffic to respond to change of t_S. Choosing Δ in range $[1sec, 5sec]$ usually satisfies this condition. The parameter α determines the responsiveness of ODT, and should be selected such that it allows doubling/halving of t_S within several seconds (up to one minute). The parameter W_0, is the one that determines importance of old measurements in the current estimation, a large W_0 is appropriate for static or very slowly varying environments, while a small W_0 is necessary for more dynamic conditions. In our experiments we use W_0 such that ΔW_0 is within an order of magnitude of one minute.

At this point we discuss the notion of variability in the traffic conditions. Measurements from [83] show that on typical 150Mbps+ links, basic IP parameters such as the number of active connections, the proportion of TCP traffic, the aggregate IP traffic, etc., do not change dramatically. Although we do not exclude the possibility that there can be drastic changes in the traffic mixes, our basic presumption in the design of ODT is that such events are rare enough to be considered as exception rather than rule. Thus, ODT is designed to search for an optimal solution in the "regular" intervals, during which traffic conditions vary slowly. In the cases of dynamic traffic conditions, one can perform self tuning of the parameters W_0 and Δ depending on the level of changes in the traffic conditions. However, for the reasons discussed above, present ODT algorithm does not incorporate this adaptation of the parameters.

The following theorem shows that, assuming that estimators $\hat{B}$ preserves order of B on the grid $\mathcal{T}_\alpha = \{t(0) \cdot \alpha^n,\ n \in Z\}$ the controller (9.9) runs system to the state that is close to global optima.

[8]Having this information $\tilde{u}(k)$ is computed as ratio $NmbBytes/\Delta$, while $\tilde{d}(k) = TotQDelay/NmbArrivals$, at the end of the k-th sampling period.

Theorem 9.1 *Let t^* be the point where global maximum of B is attained. Suppose that estimator $\hat{B}$ preserves the order on the grid $\mathcal{T}_\alpha$, ie. for all $m_1, m_2 \in Z$:*

$$\hat{B}(m_1) \geq \hat{B}(m_2) \Leftrightarrow B(t(0)\alpha^{m_1}) \geq B(t(0)\alpha^{m_2}). \tag{9.14}$$

Then there exist m_0 such that for all positive integers r:

$$t(m_0 + 2r) = t(m_0 + 2r + 2) = \bar{t}$$

$$t(m_0 + 4r + 1) = \bar{t}\alpha, \quad \textit{and} \quad t(m_0 + 4r - 1) = \frac{\bar{t}}{\alpha},$$

and the relative error between $\bar{t}$ and t^* satisfies:

$$\frac{\bar{t} - t^*}{t^*} \leq \alpha - 1. \tag{9.15}$$

Proof. Suppose without loss of generality that $t(0) < t^*$. Then by Assumption 9.1 and (9.14) there exist positive integer k_0 such that for all $k < k_0$:

$$t(k+1) = \alpha t(k) > t(k)$$

and

$$t(k_0) > t^*.$$

Now, we can distinguish two cases:

1st Case: $B(t(k_0)) \geq B(t(k_0-1))$. Then $t(k_0+1) = \alpha t(k_0)$. By concavity $B(t(k_0+1)) < B(t(k_0))$ and therefore $t(k_0+2) = \frac{t(k_0+1)}{\alpha} = t(k_0)$, and $t(k_0+3) = \frac{t(k_0+1)}{\alpha}$.

2nd Case: $B(t(k_0)) < B(t(k_0-1))$. Then $t(k_0+1) = \frac{t(k_0)}{\alpha} = t(k_0-1)$ and $t(k_0+2) = \frac{t(k_0-1)}{\alpha}$. From concavity of B we get $t(k_0+3) = t(k_0-1)$ and $t(k_0+4) = \alpha t(k_0-1)$.

By taking $\bar{t} = t(k_0)$ and $m_0 = k_0$ in the first case, or $\bar{t} = t(k_0-1)$ and $m_0 = k_0 - 1$ in the second case, we conclude the first part of the Theorem. To obtain the inequality (9.15), note that $\bar{t} \in (\frac{t^*}{\alpha}, t^*\alpha)$ which is equivalent to

$$\frac{\bar{t} - t^*}{t^*} \in (\frac{1-\alpha}{\alpha}, \alpha - 1) \subset (-(\alpha-1), \alpha - 1).$$

□

9.5 Optimal BLUE

BLUE is an active queue management algorithm that maintains a single internal variable p_m which is used for calculating the drop probability of arriving packets: each packet is dropped with probability p_m on arrival. Roughly speaking, the performance goal of BLUE is to keep the loss probability as

low as possible such that no buffer overflow occurs. The variable p_m is updated once per interval of length $freeze\_time$, in an Additive Increase - Additive Decrease (AIAD) fashion with parameters δ_1 and δ_2: if during the previous $freeze\_time$ interval no buffer-overflow losses has occurred then p_m is reduced by δ_2, otherwise p_m is increased by δ_1. By using a strategy of this type, BLUE searches for the "correct" rate at which it should drop packets. As we can see, there is no formal objective in terms of utilization or delays. However, we use the idea of controlling the drop probability t_p (that is same as p_m in the original BLUE) in order to maximize overall benefit $B(t_p)$. The first step in the design of such scheme is the following observation.

By controlling drop probability, one can control both utilization and queueing delays.

Low drop probabilities keep both queueing delays and utilization large, and high drop probabilities keep both queueing delays and utilization low; see Simulation B in the Section 9.3. By specifying a price function $P(d)$ that defines a relative price between delays and utilization, our performance goal in the design of Optimal BLUE is maximization of the benefit:

$$B(t_p) = u(t_p) - P(aQd(t_p)).$$

The quantities of interest for the calculation of the benefit are the average values of utilization and the queueing delays, and can be seen as the expected values of appropriate random variables. Because of large noise in the estimation of these quantities, it is helpful to use filtering in conjunction with an algorithm that continuously searches for the optimum. OB will use same strategy for controlling of t_p as ODT uses for controlling the available queue size t_S. We update t_p once per sample period of length Δ in an MIMD fashion:

$$t_p(k+1) = t_p(k) \cdot m(k), \tag{9.16}$$

where $m(k)$ is defined by:

$$m(k) = \alpha, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_p(k) - t_p(k-1)} \geq 0$$

$$m(k) = \frac{1}{\alpha}, \quad \text{if} \quad \frac{\hat{B}(l(k)) - \hat{B}(l(k-1))}{t_p(k) - t_p(k-1)} < 0.$$

Here $l(k)$ denotes an integer such that $t_p(k) = t_p(0) \cdot \alpha^{l(k)}$, and $\hat{B}(l(k))$ is the estimated value of the benefit $B(x)$ at the point $x = t_p(k) = t_p(0) \cdot \alpha^{l(k)}$. The method for estimating $\hat{B}(l(k))$ is same as in ODT and is given by (9.10). We also exclude non-monotonic estimation of $\hat{u}$ and $\hat{d}$. Recall that $u(t_S)$ and $aQd(t_S)$ are increasing functions of the available queue size t_S, while $u(t_p)$ and $aQd(t_p)$ are decreasing functions of t_p(see Figures 9.1 and 9.2). Because of this (9.11) is executed if $\hat{u}(i) < \hat{u}(i+1)$ or $\hat{u}(i) > \hat{u}(i-1)$, and (9.12) is executed if $\hat{d}(i) < \hat{d}(i+1)$ or $\hat{d}(i) > \hat{d}(i-1)$; $i = l(k)$.

We use an MIMD strategy for the control of the drop probability t_p rather than the original BLUE-like AIAD, because of scalability reasons. Namely, with an MIMD strategy, if algorithm runs in a low drop probability regime, absolute changes in t_p per step are smaller than absolute changes in t_p per step in higher drop probability regimes. On the other hand, in AIAD schemes, absolute changes per step are constant and are given by AI and AD parameters.

The memory requirements of OB are same as ODT. Storing four sequences, $aug, n_-, \hat{u}$ and $\hat{d}$, requires just several kilobytes. Since all these quantities are used just once per Δ, then they can be stored off-chip and their size is not important. From the computational point, OB requires one more operation per packet: random drop. This, makes OB roughly equivalent to RED from the level of the computational resources they require. The parameters of OB are the same as in ODT and are given in the Table 2. Comments related to the selection of the parameters of ODT applies to the OB as well.

9.6 Simulations

Simulation D. Our first set of simulations illustrate the dynamics of t_S and t_p under static conditions of 50 TCP flows with RTT's uniformly distributed in range $[20, 220]msec$ and with packet sizes of 1000 bytes. We run ODT and OB with parameters $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The price function used in both cases is $P_{10}(d) = 10 \cdot d$. Initially: $t_S(0) = 100Kbytes$, $t_p = 0.002$. The off-line (see Simulation E) optimal values are approximately $t_S^* \approx 130Kbytes$ and $t_p^* \approx 0.0048$.

Simulation E. The second set of simulations shows how close the average queueing delays and average utilization are to the optimal values, in static conditions with a constant number TCP flows, for both ODT and OB. We ran the same set of 50 TCP flows, with RTT's uniformly distributed in range $[20, 220]ms$ and packet sizes of 1000 bytes, as in Simulations A and B (Section 9.3). By running a sequence of long simulations in Section 9.3 we obtained plots given in Figures 9.1 and 9.2, that represent the dependance between t_S, (t_p respectively), the average utilization, and the queueing delays. Having these graphs, we can, in an off-line manner, find the value of t_S (t_p) that maximizes benefit $B(t_S)$ (resp $B(t_p)$). The red stars on Figures 9.7 (ODT) and 9.8 (OB) show these off-line optimal values for this optimization problem (9.5) (defined by price functions $P_\gamma(d) = \gamma \cdot d$) for $\gamma = 2, 10, 20$. Having $P_\gamma(d)$ as a parameter of the scheme, we ran ODT and OB. The other simulation parameters were: $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The blue crosses on Figures 9.7 and 9.8 represent the long-run (5 minutes) averages of queueing delays and utilization for $\gamma = 2, 10, 20$. The numerical results for this simulation are shown in the Table 3.

It is important to notice here that in TCP environments, OB outperforms ODT in terms of maximization the benefit B. This is because a Drop-Tail queue that achieves average queueing delay d_0 achieves less throughput than a queue with on-arrival random dropping, with the same average queueing delay d_0. The reason for this lies in the phenomenon of (partial) synchronization of losses that is a feature of Drop-Tail queues. Figure 9.9 illustrates this difference for queues servicing 50 TCP

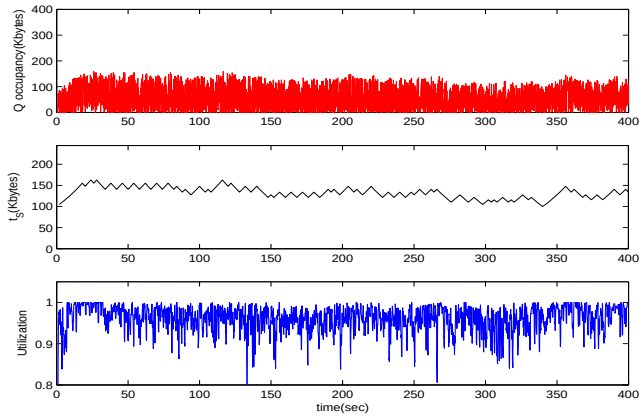

Figure 9.5: Simulation D. Queue occupancy, available buffer space(t_S), and utilization for ODT servicing 50 TCP flows.

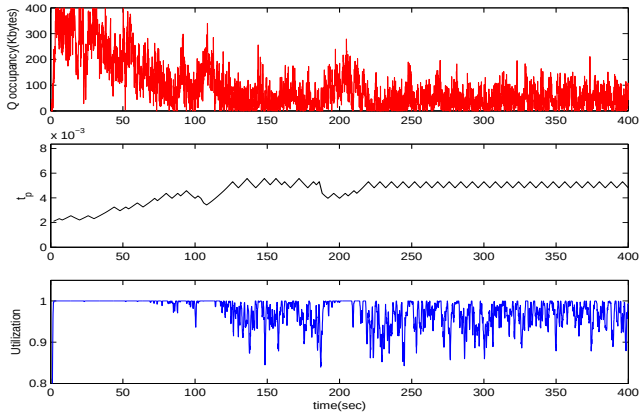

Figure 9.6: Simulation D. Queue occupancy, drop probability(t_p), and utilization for OB servicing 50 TCP flows.

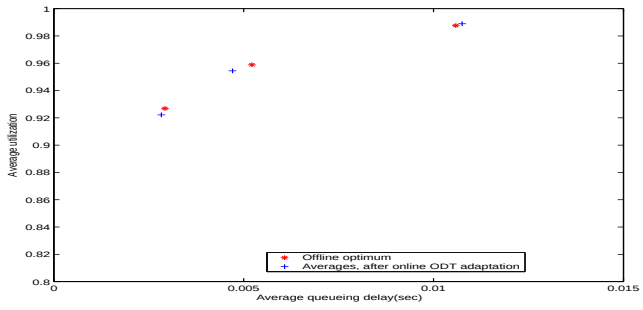

Figure 9.7: ODT: Off-line vs. online average queueing delays and utilization. Price functions $P_\gamma(d)$, for $\gamma = 2, 10, 20$.

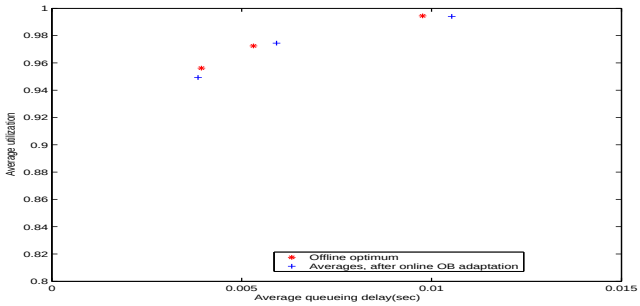

Figure 9.8: OB: Off-line vs. online average queueing delays and utilization. Price functions $P_\gamma(d)$, for $\gamma = 2, 10, 20$.

flows, and is based on data from the simulations A and B.

Scheme, γ	$aQd(sec)$	u	B_γ
ODT, $\gamma = 2$, online	0.01075	0.9890	0.9675
DT, $\gamma = 2$, off-line	0.01058	0.9876	0.9664
ODT, $\gamma = 10$, online	0.00471	0.9544	0.9074
DT, $\gamma = 10$, off-line	0.00521	0.9589	0.9067
ODT, $\gamma = 20$, online	0.00283	0.9222	0.8655
DT, $\gamma = 20$, off-line	0.00293	0.9269	0.8683
OB, $\gamma = 2$, online	0.00975	0.9941	0.9730
CDP, $\gamma = 2$, off-line	0.00973	0.9945	0.9749
OB, $\gamma = 10$, online	0.00591	0.9745	0.9153
CDP, $\gamma = 10$, off-line	0.00531	0.9725	0.9193
OB, $\gamma = 20$, online	0.00385	0.9493	0.8723
CDP, $\gamma = 20$, off-line	0.00394	0.9562	0.8774

Table 9.3: Numerical results: off-line optima and online ODT and OB averages. The last column represents $B_\gamma(t) = u(t) - \gamma \cdot aQd(t)$. (CDP - stands for Constant Drop Probability queuing)

Remark. At this point we note an important practical issue related to applicability of our algorithms in the existing Internet routers. The only two pieces of information they require are the "achieved utilization" and the "average queueing delay" experienced during one sampling period. At the end of each sampling period we set new values of the available buffer space (t_S in ODT) or drop probability (t_p in OB). While most routers allow configuration of the available buffer space, we are not aware of option for configurability of the drop probability.

Simulation F. In this simulation we present the behavior of ODT and OB in the case of mixtures of TCP and UDP traffic. In this simulation, the same set of 50 TCP flows that were defined previously compete for a bandwidth on $10Mbyte/sec$ link, with 50 UDP flows that have exponentially distributed on and off periods. The on-periods have a mean of $1000ms$, and the off-periods have mean of $3000ms$.

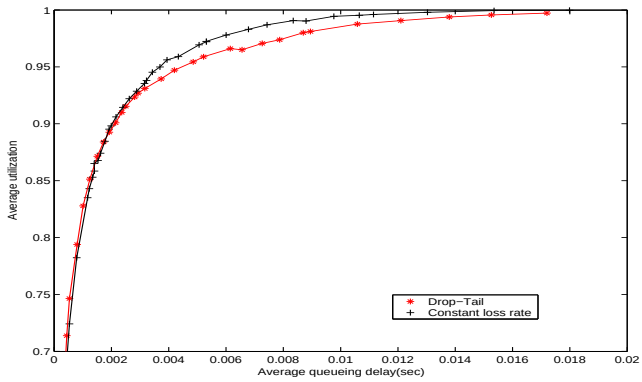

Figure 9.9: Average queueing delays vs. average utilization for drop tail and constant loss rate queues.

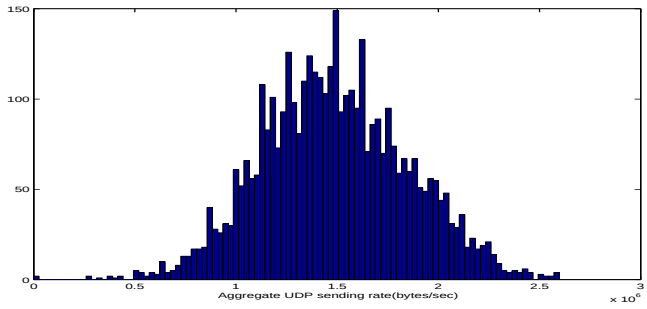

Figure 9.10: Simulation F. Histogram of aggregate UDP sending rate; sampling intervals $100ms$.

The sending rate for the on-periods is $1000Kbit/sec$. The aggregate UDP arrival rate has a mean of $1.4867Mbyte/sec$ which is approximately 14.9% of the link's service rate. A histogram, given in Figure 9.10, shows the distribution of the aggregate UDP sending rate sampled on $100ms$ intervals.

The ODT and OB parameters are the same as in previous simulations: $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The price function used in both cases is $P_{10}(d) = 10 \cdot d$. Initially: $t_S(0) = 100Kbytes$ and $t_p(0) = 0.002$.

Figures 9.11 and 9.12 depict plots of utilization, queue occupancy and t_S(t_p resp.). We observed a slightly larger fluctuation in t_S and t_p compared to Simulation D. This is consequence of the stochastic nature of the non-responsive traffic.

Simulation G. In the following simulation we show the behavior of ODT and OB in the case of sudden changes in the traffic conditions. Two sets of 50 TCP flows in this simulation compete for bandwidth on a $10Mbyte/sec$ link. The first set is active during whole simulation in the interval $[0, 900]$ seconds; connections from the second set are active only in the interval from $[300, 600]$ seconds. Thus, at time $t_1 = 300sec$, the number of TCP connections is doubled while at time $t_2 = 600sec$, the number of connections is halved, as is depicted in the Figure 9.13. The ODT and OB parameters are the same as in previous simulations: $\Delta = 2sec$, $\alpha = 1.05$, $W_0 = 30$. The price function used in both cases is $P_{10}(d) = 10 \cdot d$. Initially: $t_S(0) = 50Kbytes$ and $t_p(0) = 0.002$.

Figures 9.14 and 9.15 illustrate the behavior of ODT and OB in this case. Although ODT and OB are not designed for environments with sudden changes we see that they adjust their parameters

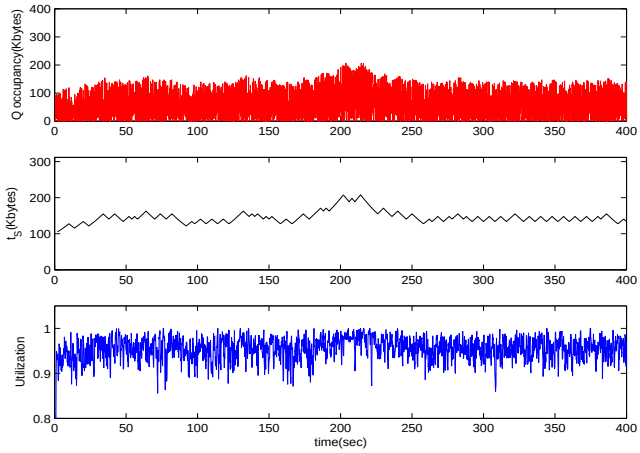

Figure 9.11: Simulation F. Queue occupancy, available buffer space(t_S), and utilization for ODT servicing 50 TCP flows and 50 on-off UDP flows.

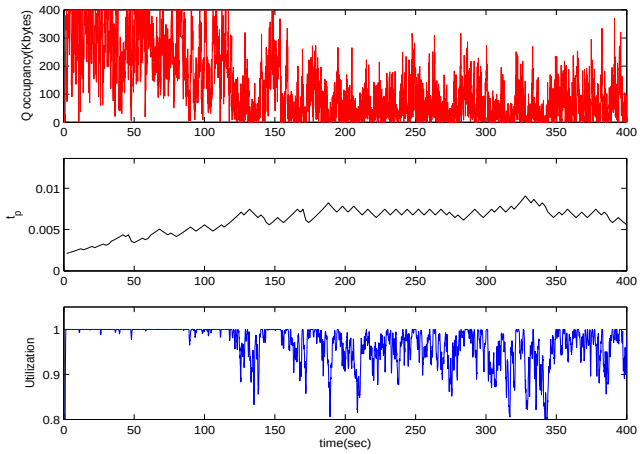

Figure 9.12: Simulation F. Queue occupancy, drop probability(t_p), and utilization for OB servicing 50 TCP flows and 50 on-off UDP flows.

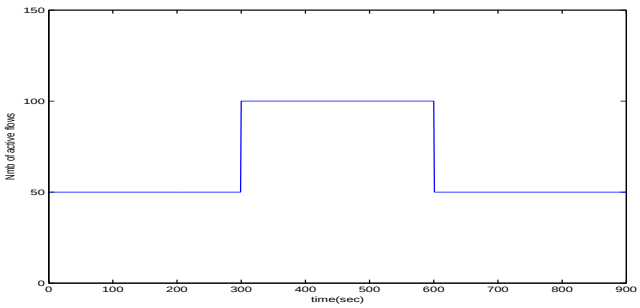

Figure 9.13: Simulation G. Number of active TCP flows in time.

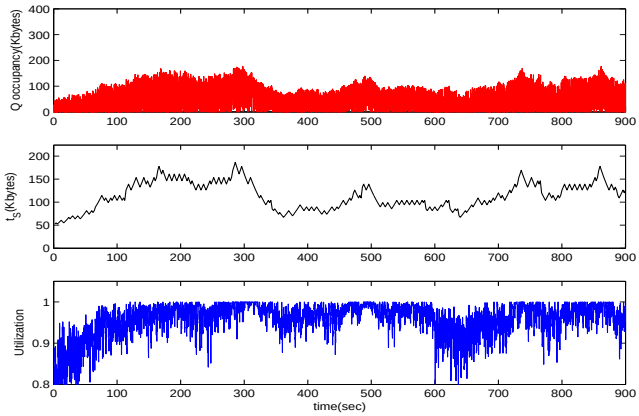

Figure 9.14: Simulation G. Queue occupancy, available buffer space(t_S), and utilization for ODT servicing a varying number of TCP flows.

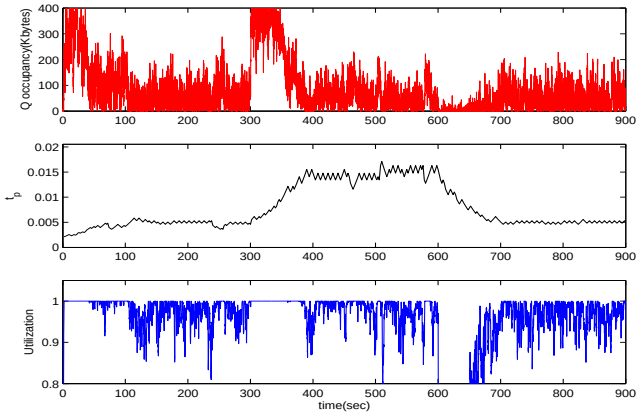

Figure 9.15: Simulation G. Queue occupancy, drop probability(t_p), and utilization for OB servicing a varying number of TCP flows.

to the sudden traffic changes. However, "convergence" to the optimal region takes 1-2 minutes in the present simulation.

Simulation H. Here we demonstrate how other performance metrics are impacted by changes in queueing delay. We concentrate on fairness and loss rate. We use Jain's Fairness Index (JFI) [50] as a fairness indicator and is defined as follows. For set of users $u_1, \ldots, u_k$ let $r = (r_1, \ldots, r_k)$ be vector of their achieved average rates during the measurement interval. Then

$$JFI(r) = \frac{\left(\sum_{i=1}^{N} r_i\right)^2}{N \sum_{i=1}^{N} r_i^2}. \tag{9.17}$$

The simulation setup is same as in Simulations A and B and consists of 50 TCP flows serviced by 10MBps link with RTT's uniformly distributed in $[20, 200]ms$. A basic observation is that the performance of TCP-like congestion control algorithms, whose dynamics depend on round-trip time, is significantly affected by queueing delays. By increasing the queueing delay, the aggressiveness of

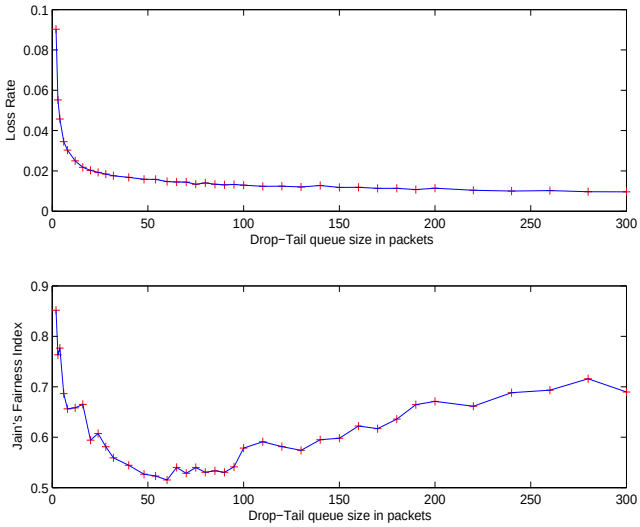

Figure 9.16: Simulation H. Loss rates (top) and JFI (bottom) for 50 TCP flows serviced by Drop-Tail queues of different sizes.

TCP senders is increased, implying lower loss rates. From a fairness perspective, larger queueing delays decrease bias against long-RTT connections. Indeed, for two TCP connections, with round trip times $RTT_1, RTT_2,\ RTT_1 < RTT_2$, bottlenecked at a single link with queueing delay d_0, the ratio of their expected rates[9] is $\frac{RTT_1+d_0}{RTT_2+d_0}$. Increasing, d_0 leads this ratio to a value closer to one. Figure 9.16 presents the dependance between available space in FIFO Drop-Tail queue and loss rate and JFI. Each '+' corresponds to a 5 minute average. We note that for very small queue sizes (< 50 packets), loss rates are large and TCP dynamics is dominated by timeouts. In this regime the square root formula is not valid and fairness is impacted mainly by timeout mechanism. Corresponding average queueing delays and utilization are depicted in Figure 9.1.

Figure 9.17 depicts the JFI for the same set of 50 flows, on link where each packet is dropped on arrival with constant probability t_p. The corresponding average queueing delays and utilization are depicted in Figure 9.2. Note that for t_p in range $[0.005, 0.01]$, queueing delays are small ($< 5ms$) compared to the RTT's and that the loss rate is small enough implying good accuracy of the square root formula. As result of this, the JFI is roughly constant in this range.

9.7 Summary

In this chapter we have addressed the problem of utilizing the tradeoff between queueing delays and link utilization. By specifying the relative importance of queueing delays and utilization, an optimal choice of a queue management parameter is the one that maximizes the overall benefit defined by (9.4). Another view on the same problem could be minimization of the total cost where the total cost is defined by the sum of the proportion of idle time, $1 - u$, and price of the queueing delays. There could

[9] This follows from the square root formula (9.3).

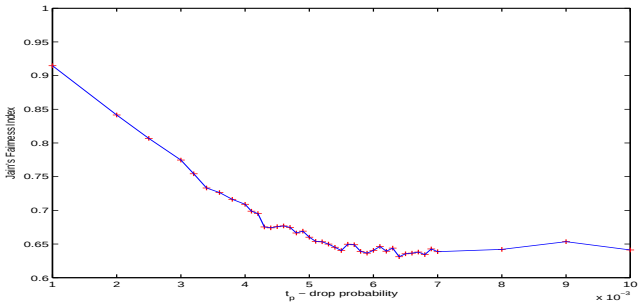

Figure 9.17: Simulation H. JFI for 50 TCP flows serviced by queue with constant drop probability.

be two possible approaches for solving this problem. First, suppose that one can, by accurate modelling and effective estimation, predict the delay/utilization dependance from the control parameter. Then, by an off-line solving of the underlying optimization problem we can set the parameter that controls queue scheme to the optimal value; see [5] for one strategy of this type. And second, where one can adapt the control parameter such that on average the overall benefit is maximized. We argue, that the first approach is not feasible because of both nonexistence of accurate and tractable enough models for the delay/utilization dependance, and the highly nontrivial estimation techniques that such an approach would require. We thus follow the second approach and design two schemes Optimal Drop-Tail and Optimal BLUE which aim to solve the underlying optimization problem by online adaptation of available buffer space (ODT) or drop probability (OB).

The optimization problem (9.5) assumes a linear dependance between utilization and benefit, and completely neglected other important performance metrics such as jitter, loss probability, and fairness. In fact, one can define the general overall benefit of the queueing scheme controlled by parameter t as:

$$B_G(t) = V(u(t)) - P_1(aQd(t)) - P_2(j(t)) - P_3(L(t)) - P_4(f(t)), \tag{9.18}$$

where $j(t)$ is jitter, $L(t)$ is the loss rate, $f(t)$ is a fairness indicator, $V(u(t))$ is the value of the utilization $u(t)$, and $P_i, i = 1, 2, 3, 4$ are appropriate price functions. We again emphasize the importance of fairness in TCP environments where long-RTT connections could heavily suffer from low queueing delays at the congested links. The embedding of jitter and loss rate into current framework can be done in straightforward manner. However, including fairness into the optimization framework, would be much more challenging as we are not aware of any, computationally light, estimation technique that would faithfully indicate level of fairness. One possible approach to estimate level of the fairness could be by counting runs[10] as suggested in [61]. However, a runs counter would give estimate of Jain's fairness index [50] only for flows that have experienced runs and Jain's fairness index is just one possible fairness indicator.

In this chapter we implemented two strategies for settling the underlying optimization problem:

[10]Run is event where arriving packet belongs to the same flow as some, previously arrived packet.

in one, the control variable is the available queue space, while another controls the drop probability. Queueing schemes that use different control parameters are possible as well. For example, virtual queue capacity could be powerful in the control of delay/utilization, and therefore could be basis for ODT/OB-like scheme.

From the theoretical point of view, an important open issue is convexity (concavity) of the average utilization/Q-delays/ loss-rates as function of control parameter t (available buffer space, random drop probability, virtual queue capacity, etc). While some results exist for the nonelastic traffic [67, 68], in the case of elastic traffic, arrival process depends on the control parameter, which makes modelling of the corresponding tradeoff curve much more challenging.

Other AQM schemes could be seen in the optimization framework as well. For example an Adaptive Virtual Queue algorithm described in [70], has the performance goal of keeping utilization at some prescribed level λ, by controlling $\tilde{C}$ (the service rate of the virtual queue). This can be translated in the problem of maximizing the benefit $B(\tilde{C}) = V(u(\tilde{C}))$, for a concave "value" function $V : [0, 1] \mapsto R$, that achieves global maximum at the point $x_{max} = \lambda$. Similarly, for a PI controller [47] that has the performance goal of keeping queue occupancy at level q_{ref}, one can define the optimization problem of minimization of the cost $C(p) = P_1(q(p))$, where $q(p)$ represents average queue occupancy for the parameter p, and a concave "cost" function $P_1 : [0, q_{max}] \mapsto R$ have global minimum at the point $x_{min} = q_{ref}$.

The MIMD based ODT and OB algorithms introduced here are just one possible approach for solving the optimization problem (9.5). In Section 9.4 we discussed the rationale for choosing MIMD algorithm that continuously searches for optimal value instead of an algorithm that will search for an exact optimal value under noisy measurements. It will be interesting to investigate other control strategies with better performance[11] than ODT and OB as part of future work, along the lines presented here.

[11] Performance metrics of interest can be benefit $B(t)$, speed of convergence to the optimal region, robustness to noise, etc.

CHAPTER 10

Conclusions

We started this thesis by studying TCP resource allocation based on the random matrix model proposed in [99]. A main feature of this model is the fact that it models the interaction between TCP flows and is relatively simple and tractable. We relaxed the IID assumption from [99], gave a fixed-point-based iteration schemes for calculating the TCP resource allocation and its variance in the Markovian model. In rest of the thesis we have studied several problems that arise in the design of high-speed routers.

In Chapter 4 we developed a stateless scheme for enforcing the max-min fairness of TCP flows: MLC(l). However, there are two important open issues related to practical deployment of MLC(l). Firstly, the design of MLC(l) is based on the assumption that all the flows use TCP congestion control; therefore it does not have effects on non-elastic flows and some extension to MLC(l) is needed to capture the non-elastic flows. Secondly, MLC(l) uses virtually no memory but it could require a large computational resources under some circumstances. Investigating this tradeoff between computational resources and memory requirements needed for enforcing (max-min) fairness is an important unresolved problem.

Markov Active Yield (MAY), the queue management scheme designed in the chapter 5 to resolve two issues related to MLC(l): (1) controlling the fairness of both elastic and non-elastic flows and (2) computational expensiveness. The memory requirements in Pareto-like flow size distributions are equivalent to roughly one-bit per flow which together with currently available devices and state-of-the-art implementation of hash-tables and statistics-counters allows implementations at line speeds of over $40Gbps$.

The need for efficient statistics-counter architectures is evident as Internet line-speeds are growing with rate which is significantly greater than rate of growth of the speeds of the appropriate memory-devices. While several proposals exist for passive applications (counter value information is not required on per-packet basis), for many active applications such as flow-control, flow-scheduling, remote control, etc. the counter value information is required on per-packet basis. In Chapter 6 we

propose a randomized active counters architecture that have small memory requirements as well as small expected errors.

Heavy-hitter identification has been extensively studied in the recent past. The algorithm presented in Chapter 7 posses several attractive features: (1) it tracks flow rates rather than flow sizes (which can significantly differ in the context of Internet flows) (2) it has very low memory requirements (3) it is relatively robust to heavy-tailed assumption on flow sizes and (4) it is computationally light and highly scalable in terms of number of active flows.

In the last part of the thesis we discuss the delay-utilization tradeoff in the congested Internet links. While the first part(s) of this thesis is mainly based on the modelling as the main tool for the design of the appropriate algorithms, in the last part of the thesis we use models (developed by others) as motivation and use the measurement-based control approach for settling the problems of interest. The algorithms presented in Chapters 8 and 9 are surprisingly simple yet effective in searching for optimal queue-scheme parameter subject to delay-utilization tradeoff. The main issue that remains unsolved is concavity of the delay-utilization tradeoff for various queue-scheme parameters such as DropTail queue size, loss rate, virtual-queue capacity, etc. While a number of results exist in open-loop case, closed-loop nature of TCP traffic makes arrival process dependent on the control parameter, which makes modelling of the corresponding tradeoff curve much more challenging.

The proposed AQM schemes for enforcing max-min fairness are MLC(l), MAY and HH. While MLC(l) have mainly theoretical value, MAY and HH possess the potential practical value. While HH is designed to exploit limited amount of fast SRAM memory it is a rather ad-hoc algorithm without strict performance bounds. On the other hand, the MAY design is built on the rigorous theoretical analysis that allows explicit performance bounds and highly efficient use of the SRAM resources which makes it a better candidate for use in real Internet links with complex traffic mix. Adaptive algorithms discussed in the last part of the thesis for control of queueing delay vs. throughput tradeoff are easy to implement since they do not require neither hardware support nor per-packet processing. We note here, that because of RTT dependance of the TCP throughput ADT-like algorithms for reducing packet delays might require some mechanism for enforcement of fairness that can be harmed by extremely small queueing delays.

Finally we discuss one idea that could unify two designs from this thesis: low state consumption and low buffering. Assume that M bytes of SRAM are available that can be used for buffering and state information. Let M_1 bytes are used for buffering and the rest $M_2 = M - M_1$ bytes of SRAM are used for the state information. Suppose that performance goal is simultaneous achieving low queuing delays and max-min fair resource allocation. The key observation is that we need: large M_1 and small M_2 in small-number-of-users regime or small M_1 and large M_2 in large-number-of-users regime. The question of efficient and meaningful allocation of (M_1, M_2) is an interesting open question left unanswered in the thesis.

APPENDIX A

Random matrix model of AIMD congestion control

A.1 Synchronised communication networks

We begin our discussion by considering communication networks for which the following assumptions are valid: (i) at congestion every source experiences a packet drop; and (ii) each source has the same round-trip-time (RTT)[1]. In this case an exact model of the network dynamics may be found as follows [92].

Let $w_i(k)$ denote the congestion window size of source i immediately before the k'th network congestion event is detected by the source. Over the k'th congestion epoch three important events can be discerned: $t_a(k), t_b(k)$ and $t_c(k)$; as depicted in Figure A.1. The time $t_a(k)$ denotes the instant at which the number of unacknowledged packets in flight equals $\beta_i w_i(k)$; $t_b(k)$ is the time at which the bottleneck queue is full; and $t_c(k)$ is the time at which packet drop is detected by the sources, where time is measured in units of RTT[2]. It follows from the definition of the *AIMD* algorithm that

[1] One RTT is the time between sending a packet and receiving the corresponding acknowledgement when there are no packet drops.

[2] Note that measuring time in units of RTT results in a linear rate of increase for each of the congestion window variables between congestion events.

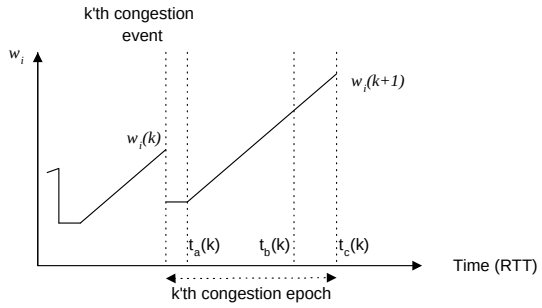

Figure A.1: Evolution of window size

the window evolution is completely defined over all time instants by knowledge of the $w_i(k)$ and the event times $t_a(k), t_b(k)$ and $t_c(k)$ of each congestion epoch. We therefore only need to investigate the behavior of these quantities.

We assume that each source is informed of congestion one RTT after the queue at the bottleneck link becomes full; that is $t_c(k) - t_b(k) = 1$. Also,

$$w_i(k) \geq 0, \sum_{i=1}^{n} w_i(k) = P + \sum_{i=1}^{n} \alpha_i, \ \forall k > 0, \tag{A.1}$$

where P is the maximum number of packets which can be in transit in the network at any time; P is usually equal to $q_{max} + BT_d$ where q_{max} is the maximum queue length of the congested link, B is the service rate of the congested link in packets per second and T_d is the round-trip time when the queue is empty. At the $(k+1)$th congestion event

$$w_i(k+1) \quad = \quad \beta_i w_i(k) + \alpha_i [t_c(k) - t_a(k)]. \tag{A.2}$$

and

$$t_c(k) - t_a(k) \quad = \quad \frac{1}{\sum_{i=1}^{n} \alpha_i} [P - \sum_{i=1}^{n} \beta_i w_i(k)] + 1. \tag{A.3}$$

Hence, it follows that

$$w_i(k+1) \quad = \quad \beta_i w_i(k) + \frac{\alpha_i}{\sum_{j=1}^{n} \alpha_j} [\sum_{i=1}^{n} (1 - \beta_i) w_i(k)] \tag{A.4}$$

and that the dynamics an entire network of such sources is given by

$$W(k+1) = AW(k), \tag{A.5}$$

where $W^T(k) = [w_1(k), \cdots, w_n(k)]$, and

$$\begin{aligned} A \quad &= \quad \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n \end{bmatrix} \\ &+ \quad \frac{1}{\sum_{j=1}^{n} \alpha_j} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} \begin{bmatrix} 1-\beta_1 & 1-\beta_2 & \cdots & 1-\beta_n \end{bmatrix}. \end{aligned} \tag{A.6}$$

The matrix A is a positive matrix (all the entries are positive real numbers) and it follows that the synchronized network (A.5) is a positive linear system [11].

A.2 Models of unsynchronized network

The preceding discussion illustrates the relationship between important network properties and the eigensystem of a positive matrix. Unfortunately, the assumptions under which these results are derived, namely of source synchronization and uniform RTT, are quite restrictive (although they may,

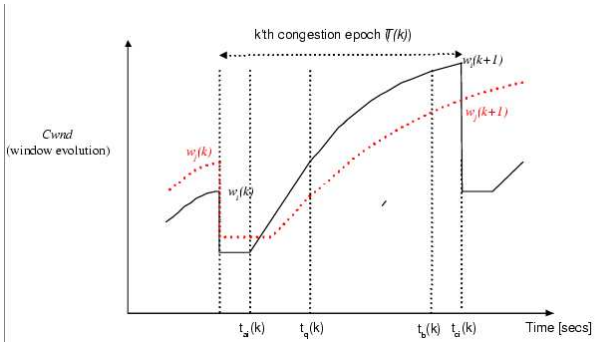

Figure A.2: Evolution of window size over a congestion epoch. $T(k)$ is the length of the congestion epoch in seconds.

for example, be valid in many long-distance networks [121]). It is therefore of great interest to extend our approach to more general network conditions. As we will see the model that we obtain shares many structural and qualitative properties of the synchronized model described above. To distinguish variables, we will from now on denote the nominal parameters of the sources used in the previous section by α_i^s, β_i^s, $i = 1, \ldots, n$. Here the index s may remind the reader that these parameters describe the *synchronized case*, as well as that these are the parameters that are chosen by each *source*.

Consider the general case of a number of sources competing for shared bandwidth in a generic dumbbell topology (where sources may have different round-trip times and drops need not be synchronized). The evolution of the *cwnd* (w_i) of a typical source as a function of time, over the $k'th$ congestion epoch, is depicted in Figure A.2. As before a number of important events may be discerned, where we now measure time in seconds, rather than units of RTT. Denote by $t_{ai}(k)$ the time at which the number of packets in flight belonging to source i is equal to $\beta_i^s w_i(k)$; $t_q(k)$ is the time at which the bottleneck queue begins to fill; $t_b(k)$ is the time at which the bottleneck queue is full; and $t_{ci}(k)$ is the time at which the i'th source is informed of congestion. In this case the evolution of the i'th congestion window variable does not evolve linearly with time after t_q seconds due to the effect of the bottleneck queue filling and the resulting variation in RTT; namely, the RTT of the i'th source increases according to $RTT_i(t) = T_{d_i} + q(t)/B$ after t_q, where T_{d_i} is the RTT of source i when the bottleneck queue is empty and $0 \leq q(t) \leq q_{\max}$ denotes the number of packets in the queue. Note also that we do not assume that every source experiences a drop when congestion occurs. For example, a situation is depicted in Figure A.2 where the i'th source experiences congestion at the end of the epoch whereas the j'th source does not.

Given these general features it is clear that the modelling task is more involved than in the synchronized case. Nonetheless, it is possible to relate $w_i(k)$ and $w_i(k+1)$ using a similar approach to the synchronized case by accounting for the effect of non-uniform RTT's and unsynchronized packet drops as follows.

(i) **(i) Unsynchronized source drops :** Consider again the situation depicted in Figure A.2.

Here, the i'th source experiences congestion at the end of the epoch whereas the j'th source does not. This corresponds to the i'th source reducing its congestion window by the factor β_i^s after the $k+1$'th congestion event, and the j'th source not adjusting its window size at the congestion event. We therefore allow the back-off factor of the i'th source to take one of two values at the k'th congestion event.

$$\beta_i(k) \quad \in \quad \{\beta_i^s, 1\}\,, \tag{A.7}$$

corresponding to whether the source experiences a packet loss or not.

(ii) **Non-uniform RTT :** Due to the variation in round trip time, the congestion window of a flow does not evolve linearly with time over a congestion epoch. Nevertheless, we may relate $w_i(k)$ and $w_i(k+1)$ linearly by defining an average rate $\alpha_i(k)$ depending on the k'th congestion epoch:

$$\alpha_i(k) := \frac{w_i(k+1) - \beta_i(k)w(k)}{T(k)}, \tag{A.8}$$

where $T(k)$ is the duration of the k'th epoch measured in seconds. Equivalently we have

$$w_i(k+1) = \beta_i(k)w_i(k) + \alpha_i(k)T(k)\,. \tag{A.9}$$

In the case when $q_{max} << BT_{d_i}\,, i = 1, \ldots, n$, the average α_i are (almost) independent of k and given by $\alpha_i(k) \approx \alpha_i^s/T_{d_i}$ for all $k \in N, i = 1, \ldots, n$. The situation when

$$\alpha_i \approx \frac{\alpha_i^s}{T_{d_i}}, \quad i = 1, \ldots, n \tag{A.10}$$

is of considerable practical importance and such networks are the principal concern of this paper. This corresponds to the case of a network whose bottleneck buffer is small compared with the delay-bandwidth product for all sources utilizing the congested link. Such conditions prevail on a variety of networks; for example networks with large delay-bandwidth products, and networks where large jitter and/or latency cannot be tolerated. In view of (A.7) and (A.9) a convenient representation of the network dynamics is obtained as follows. At congestion the bottleneck link is operating at its capacity B, i.e.,

$$\sum_{i=1}^{n} \frac{w_i(k) - \alpha_i}{RTT_{i,max}} = B, \tag{A.11}$$

where $RTT_{i,max}$ is the RTT experienced by the i'th flow when the bottleneck queue is full. Note, that $RTT_{i,max}$ is independent of k. Setting $\gamma_i := (RTT_{i,max})^{-1}$ we have that

$$\sum_{i=1}^{n} \gamma_i w_i(k) = B + \sum_{i=1}^{n} \gamma_i \alpha_i\,. \tag{A.12}$$

By interpreting (A.12) at $k+1$ and inserting (A.9) for $w_i(k+1)$ it follows furthermore that

$$\sum_{i=1}^{n} \gamma_i \beta_i(k) w_i(k) + \gamma_i \alpha_i T(k) = B + \sum_{i=1}^{n} \gamma_i \alpha_i\,. \tag{A.13}$$

Using (A.12) again it follows that

$$T(k) = \frac{1}{\sum_{i=1}^{n} \gamma_i \alpha_i} \left(\sum_{i=1}^{n} \gamma_i (1 - \beta_i(k)) w_i(k) \right). \tag{A.14}$$

Inserting this expression into (A.9) we finally obtain

$$\begin{aligned} w_i(k+1) &= \beta_i(k) w_i(k) \\ &+ \frac{\alpha_i}{\sum_{j=1}^n \gamma_j \alpha_j} \left(\sum_{j=1}^n \gamma_j (1-\beta_j(k)) w_j(k) \right) \end{aligned} \tag{A.15}$$

and the dynamics of the entire network of sources at the k-th congestion event are described by

$$W(k+1) = A(k)W(k),\ A(k) \in \{A_1, ..., A_m\}. \tag{A.16}$$

where

$$A(k) = \begin{bmatrix} \beta_1(k) & 0 & \cdots & 0 \\ 0 & \beta_2(k) & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n(k) \end{bmatrix} \tag{A.17}$$

$$+\frac{1}{\sum_{j=1}^n \gamma_j \alpha_j} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} [\gamma_1(1-\beta_1(k)), \ldots, \gamma_n(1-\beta_n(k))]$$

and where $\beta_i(k)$ is either 1 or β_i^s. The non-negative matrices $A_2, .., A_m$ are constructed by taking the matrix A_1,

$$A_1 = \begin{bmatrix} \beta_1^s & 0 & \cdots & 0 \\ 0 & \beta_2^s & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n^s \end{bmatrix} \tag{A.18}$$

$$+\frac{1}{\sum_{j=1}^n \gamma_j \alpha_j} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} \begin{bmatrix} \gamma_1(1-\beta_1^s),\ \ldots,\ \gamma_n(1-\beta_n^s) \end{bmatrix}$$

and setting some, but not all, of the β_i to 1. This gives rise to $m = 2^n - 1$ matrices associated with the system (A.16) that correspond to the different combinations of source drops that are possible. We denote the set of these matrices by $\mathcal{A}$.

Finally we note that another, sometimes very useful, representation of the network dynamics can be obtained by considering the evolution of scaled window sizes at congestion; namely, by considering the evolution of $W_\gamma^T(k) = [\gamma_1 w_1(k)\ , \gamma_2 w_2(k), ..., \gamma_n w_n(k)]$. Here one obtains the following description of the network dynamics:

$$W_\gamma(k+1) = \bar{A}(k) W_\gamma(k) \tag{A.19}$$

with $\bar{A}(k) \in \bar{\mathcal{A}} = \{\bar{A}_1, ..., \bar{A}_m\}, \quad m = 2^n - 1$ and where the $\bar{A}_i$ are obtained by the similarity transformation associated with the change of variables. As before the non-negative matrices $\bar{A}_2, .., \bar{A}_m$ are constructed by taking the matrix $\bar{A}_1$ and setting some, but not all, of the β_i^s to 1. All of the matrices in the set $\bar{\mathcal{A}}$ are now column stochastic.

APPENDIX B

Proof of Theorem 4.1

The proof is heavily based on work [25] which deals with case $l = 1$ (constant drop probability). Here we show that for $l > 1$ appropriate scaling is possible and that same arguments can be employed as well.

Let s_k be the time of k-th loss, ie: s_k is the k-th smallest m which satisfies $U_m^{(\rho)} = \frac{1}{2}U_{m-1}^{(\rho)}$. Define $Z_0^{(\rho)} = 0$ and $Z_k^{(\rho)} = U_{s_k}^{(\rho)}$. It follows that $Z_k^{(\rho)}$ is also a Markov chain and the following theorem provides insight into the behavior of $Z_k^{(\rho)}$ for small values of ρ.

Proposition B.1 *There exist a Markov chain $\overline{V}_k$, such that for $\rho > 0$,*

$$Z_k^{(\rho)} = \frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}}\overline{V}_k + \frac{1}{\rho^{\frac{1}{l+1}}}R_k^{(\rho)},$$

where the stochastic process $R_k^{(\rho)}$ converge to zero in distribution as $\rho \to 0$.

Preamble to Proof of Proposition B.1

The Markov chain $U_k^{(\rho)}$ can be written as $U_k^{(\rho)} = RTT^2 W_k^{(\alpha)}$, where $\alpha = \rho/RTT^{2l}$ and

$$W_{k+1}^{(\alpha)} = W_k^{(\alpha)} + 1 \quad \text{with probability} \quad e^{-\alpha(W_k^{(\alpha)})^l}$$
$$W_{k+1}^{(\alpha)} = \frac{1}{2}W_k^{(\alpha)} \quad \text{with probability} \quad 1 - e^{-\alpha(W_k^{(\alpha)})^l}.$$

Analogously, we can define another associated Markov chain with states of $W_k^{(\alpha)}$ just after congestion: $V_k^{(\alpha)} = Z_k^{(\rho)}/RTT^2$. Its transition is given by

$$V_{k+1}^{(\alpha)} = \frac{1}{2}(V_k^{(\alpha)} + G_{V_k^{(\alpha)}}^{(\alpha)}).$$

Here $G_n^{(\alpha)}$ is random variable determined by distribution:

$$P(G_x^{(\alpha)} \geq y) = \prod_{k=x}^{x+\lfloor y \rfloor} e^{-\alpha k^l}.$$

Lemma B.1 *Let $x \geq 0$. Then as α goes to 0, the random variable $\alpha^{\frac{1}{l+1}} G^{(\alpha)}_{x/\alpha^{\frac{1}{l+1}}}$ converges in distribution to a random variable $\overline{G}_x$ such that for $y \geq 0$*

$$P(\overline{G}_x \geq y) = e^{-\frac{1}{l+1}((x+y)^{l+1} - x^{l+1})}. \tag{B.1}$$

Proof: Let $x, y \geq 0$ and $\alpha < 1$. Then

$$P(\alpha^{\frac{1}{l+1}} G^{(\alpha)}_{x/\alpha^{\frac{1}{l+1}}} > y) = P(G^{(\alpha)}_{x/\alpha^{\frac{1}{l+1}}} > y/\alpha^{\frac{1}{l+1}})$$

$$= \prod_{k=x/\alpha^{\frac{1}{l+1}}}^{\lfloor (x+y)/\alpha^{\frac{1}{l+1}} \rfloor} e^{-\alpha k^l}.$$

Taking logarithms we conclude:

$$ln(P(\alpha^{\frac{1}{l+1}} G^{(\alpha)}_{x/\alpha^{\frac{1}{l+1}}} > y)) = -\alpha \sum_{k=x/\alpha^{\frac{1}{l+1}}}^{\lfloor (x+y)/\alpha^{\frac{1}{l+1}} \rfloor} k^l \longrightarrow$$

$$\longrightarrow -\alpha \int_{x/\alpha^{\frac{1}{l+1}}}^{(x+y)/\alpha^{\frac{1}{l+1}}} t^l dt = \frac{1}{l+1}((x+y)^{l+1} - x^{l+1}).$$

■

Definition B.1 *The sequence $\overline{V}_n$ denotes a Markov chain given by $\overline{V}_0 = 0$ and transitions:*

$$\overline{V}_{n+1} = \frac{1}{2}(\overline{V}_n + \overline{G}_{\overline{V}_n}).$$

Here $\{\overline{G}_x : x \geq 0\}$ is a family of random variables independent of $\overline{V}_n$ such that the distribution of $\overline{G}_x$ is given by (B.1).

Lemma B.2 *The Markov chain $\alpha^{1/(l+1)} V^{(\alpha)}_n$ converges in distribution to the Markov chain $\overline{V}_n$.*

Proof: The proof of this fact follows same lines as proof of Corollary 1 in paper [25]. Namely, using uniform continuity of the mapping $t \to t^l$, and techniques from the proof of Proposition 1 from [25], it follows that for any $K > 1$

$$\lim_{\alpha \to 0} \sup_{\alpha^{1/(l+1)} < x,y < K} |P(\overline{G}_x \geq y) - P(\alpha^{\frac{1}{l+1}} G^{(\alpha)}_{x/\alpha^{\frac{1}{l+1}}} > y)| = 0.$$

Then, using the previously established uniform convergence, it follows that for any continuous function f on R^+ with compact support:

$$\lim_{\alpha \to 0} \sup_{x > \alpha^{1/(l+1)}} |E(f(\overline{G}_x)) - E(f(\alpha^{\frac{1}{l+1}} G^{(\alpha)}_{x/\alpha^{\frac{1}{l+1}}}))| = 0.$$

Finally, again following the same lines as in proof of Proposition 2 from [25] we can conclude desired convergence in distribution. ■

The previous lemma allows us to approximate the Markov chain $V_k^{(\alpha)}$ with $\overline{V}_k$ with initial value $\overline{V}_0 = \alpha^{1/(l+1)} V_0^{(\alpha)} = 0$. We are ready to give:

Proof of Proposition B.1. Recall that $Z_k^{(\rho)}$ represents throughput just after packet loss and that $\alpha = \rho / RTT^{2l}$. Then for small ρ we can write[1]

$$Z_k^{(\rho)} = \frac{1}{RTT^2} V_k^{(\alpha)} = \frac{1}{RTT^2} \left(\frac{1}{\alpha^{1/(l+1)}} \overline{V}_k + o\left(\frac{1}{\alpha^{\frac{1}{l+1}}} \right) \right) =$$

$$= \frac{RTT^{2l/(l+1)}}{RTT^2} \frac{1}{\rho^{1/(l+1)}} \overline{V}_k + o\left(\frac{1}{\rho^{\frac{1}{l+1}}} \right) =$$

$$= \frac{1}{RTT^{\frac{2}{l+1}} \rho^{\frac{1}{l+1}}} \overline{V}_k + o\left(\frac{1}{\rho^{\frac{1}{l+1}}} \right)$$

■

Thus the Markov chain $Z_k^{(\rho)}$ of throughput after congestion events can be approximated with the Markov chain $\frac{1}{RTT^{\frac{2}{l+1}} \rho^{\frac{1}{l+1}}} \overline{V}_k$ for small values of ρ as was claimed.

At this point we are concentrating on $\overline{V}_k$ and following the basic ideas from [25]. We shall prove that $(\overline{V}_k)^{l+1}$ is autoregressive(AR) process with unique invariant distribution that can be explicitly written.

Theorem B.1 *For the Markov chain $\overline{V}_k$, the process $\overline{V}_k^{l+1}$ is autoregressive with following representation:*

$$\overline{V}_{k+1}^{l+1} = \frac{1}{2^{l+1}} (\overline{V}_k^{l+1} - (l+1) E_n)$$

where E_n is an IID sequence of exponentially distributed random variables with parameter 1: $P(E_n \geq y) = e^{-y}$. Moreover the Markov chain $\overline{V}_k$ has unique invariant distribution represented by the random variable $\overline{V}_\infty$ which satisfy:

$$\overline{V}_\infty = \sqrt[l+1]{(l+1) \sum_{n=1}^{\infty} \frac{1}{2^{n(l+1)}} E_n}, \tag{B.2}$$

and the process defined with this invariant distribution as initial distribution for $\overline{V}_0$ is ergodic.

Proof: By definition we have

$$\overline{V}_{n+1} = \frac{1}{2} (\overline{V}_n + \overline{G}_{\overline{V}_n}).$$

Let $E_n = \frac{1}{l+1} (2^{l+1} \overline{V}_{n+1}^{l+1} - \overline{V}_n^{l+1})$. Then:

$$P(E_n \geq y) = P\left(\frac{1}{l+1} (2^{l+1} \overline{V}_{n+1}^{l+1} - \overline{V}_n^{l+1}) \geq y \right) =$$

$$= P((\overline{V}_n + \overline{G}_{\overline{V}_n})^{l+1} \geq (l+1) y + \overline{V}_n^{l+1})$$

$$= P(\overline{G}_{\overline{V}_n} \geq ((l+1) y + \overline{V}_n^{l+1})^{1/(l+1)} - \overline{V}_n) =$$

$$= e^{-\frac{1}{l+1} ((l+1) y + \overline{V}_n^{l+1} - \overline{V}_n^{l+1})} = e^{-y}$$

[1] Here, $o(\frac{1}{\alpha^{\frac{1}{l+1}}})$ represent random variable $R(\alpha)$ such that $R(\alpha)/\alpha^{\frac{1}{l+1}} \to 0$ as $\alpha \to 0$

Thus, the first part of the Theorem is proved. To prove the second part notice that random variable given by (B.2) represents an invariant probability. To prove its uniqueness we use Theorem 7.16 from [12]. It is enough to prove that there are no two disjoint sets $A_1, A_2 \subset R^+$ such that for $i = 1, 2$ and all $x \in A_i$, $P(\overline{V}_2 \in A_i | \ \overline{V}_1 = x) = 1$. Suppose that there exist two such sets A_1, A_2. Let $x_1 \in A_1$. Then since G_{x_1} has positive density we conclude that $[\frac{x_1}{2}, \infty) \setminus A_1$ has Lebesgue-measure 0. Similarly for $x_2 \in A_2$, $[\frac{x_2}{2}, \infty) \setminus A_2$ has Lebesgue-measure 0 which means that A_1 and A_2 cannot be disjoint. ■

Now we are ready to prove that for small values of ρ_0, the steady state throughput can be approximated by $\frac{1}{RTT^{\frac{2}{l+1}} \rho_0^{\frac{1}{l+1}}} D_{CL}$, where D_{CL} does not depend on ρ_0 nor l.

Proof: (of the Theorem 4.1)If we write $U_i^{(\rho)} = RTT^2 W_k^{(\alpha)}$, where $\alpha = \rho / RTT^{2l}$, the Theorem is an immediate consequence of the Theorem B.1 and Proposition 9 from [25] for a multiplicative decrease factor $\delta = \frac{1}{2}$.[2] ■

[2]Here constant $\frac{3}{4} = \frac{1+\delta}{2\delta(1-\delta)}$, for $\delta = 0.5$.

BIBLIOGRAPHY

[1] A. Abouzeid, S. Roy. "Analytic understanding of RED gateways with. multiple competing TCP flows". Proc. of IEEE GLOBECOM, 2000.

[2] G. Appenzeller, I. Keslassy N. McKeown. "Sizing router buffers". Proceedings of ACM SIGCOMM '04, Portland, OR, USA, August/September 2004.

[3] S. Athuraliya, D. Lapsley, S. Low. "Enhanced Random Early Marking algorithm for Internet flow control". Proc. of IEEE INFOCOM, Tel Aviv, Israel, 2000.

[4] K. Avrachenkov, U. Ayesta, P. Brown, E. Nyberg. "Differentiation between short and long TCP flows: Predictability of the response time". Proc. of IEEE INFOCOM, Hong Kong, 2004.

[5] K. Avrachenkov, U. Ayesta, A. Piunovskiy. "Optimal choice of the buffer size in the Internet routers". Proceedings of the IEEE CDC, pp. 1143-1148, Seville, Spain, 2005.

[6] K. Avrachenkov, U. Ayesta, A. Piunovskiy. "Convergence and Optimal Buffer Sizing for Window Based AIMD Congestion Control". Preprint 2007. Available online: http://arxiv.org/abs/cs/0703063.

[7] E. Altman, K. Avrachenkov, C. Barakat. "A stochastic model of TCP/IP with stationary random losses". In Proceedings of ACM SIGCOMM, August 2000.

[8] F. Bacelli, D. Hong. "AIMD Fairness and Fractal Scaling of TCP traffic". Proceedings of IEEE INFOCOM, New York, NY, USA, June, 2002.

[9] F. Bacelli, D. Hong. "Interaction of TCP flows as billiards". INRIA Rocquencourt, INRIA Technical Report 4437, 2002.

[10] M.F Barnsley, S.G. Demko, J.H. Elton, J.S. Geronimo "Invariant measures for Markov processes arising from iterated function systems with place-dependent probabilities". Ann. Inst. Henri Poincare, vol. 24 no. 3, 1988.

[11] A. Berman, R. Plemmons. "Nonnegative matrices in the mathematical sciences". SIAM, 1979.

[12] L. Breiman. "Probability". SIAM classics in Applied Math. 1992.

[13] L. Che, B. Qiu, "Landmark LRU: an efficient scheme for the detection of elephant flows at internet routers". IEEE Communications Letters, vol. 10 (7), July, 2006.

[14] Y. Chait, C. V. Hollot, V. Misra, H. H., Y. Halevi. "Dynamic analysis of congested TCP networks". Proceedings of American Control Conference, San Diego, CA, USA, June 1999.

[15] S. Cohen, Y. Matias. "Spectral bloom filters" Proc. of the ACM SIGMOD '03, San Diego, CA, USA, 2003.

[16] G. Cormode, S. Muthukrishnan. "What's hot and what's not: tracking most frequent items dynamically". ACM Transactions on Database Systems, Vol 30 , Issue 1, March 2005.

[17] A. Das, D. Dutta, A. Goel, A. Helmy, J. Heidemann. "Low State Fairness: Lower Bounds and Practical Enforcement". Proc. of the IEEE INFOCOM, Miami, FL, USA, March 2005.

[18] A. Demers, S. Keshav, S. Shenker. "Analysis and simulation of a fair queueing algorithm". Proc. of ACM SIGCOMM, Austin, TX, 1989.

[19] A. Dhamdhere, H. Jiang, C. Dovrolis. "Buffer sizing for congested internet links". Proceedings of IEEE INFOCOM, Miami, FL, USA, 2005.

[20] D.L. Donoho. "De-Noising by Soft-Thresholding". IEEE Transactions on Information Theory, Vol. 41, No. 3, May 1995.

[21] N. Duffield. "Sampling for Passive Internet Measurement: A Review". Statistical Science, Volume 19, no. 3, 2004, Pages 472-498.

[22] N. Duffield, C. Lund, M. Thorup. "Charging from sampled network usage". Proc. of the 1st ACM SIGCOMM Workshop on Internet Measurements, San Francisco, California, USA, 2001.

[23] N. Duffield. "Sampling for Passive Internet Measurement: A Review". Statistical Science, Volume 19, no. 3, 2004, Pages 472-498.

[24] N. Dukkipati, M Kobayashi, R. Zhang-Shen, N. McKeown. "Processor Sharing Flows in the Internet". Proc. of IWQoS 2005, Passau, Germany 2005.

[25] V. Dumas, F. Guillemin, P. Robert. "A Markovian analysis of additive-increase multiplicative-decrease algorithms". Adv. in Appl. Probab. 34 (2002), no. 1, 85-111.

[26] L. Elsner, I. Koltracht, M. Neumann. "On the convergence of asynchronous paracontractions with applications to tomographic reconstrucion from incomplete data". Linear Algebra Appl., vol. 130, pp. 65-82, 1990.

[27] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown,T. Roughgarden. “Part III: routers with very small buffers”. ACM Computer Communication Review 35(3): 83-90 (2005).

[28] C. Estan, G. Varghese. “New Directions in Traffic Measurement and Accounting”. Proc. SIGCOMM, August 2002

[29] C. Estan. “Comparison between multistage filters and sketches for finding heavy hitters”. UCSD technical report CS2004-0784, April 2004

[30] C. Estan, G. Varghese. “New directions in traffic measurement and accounting”. ACM Transactions on Computer Systems, Vol 21 , (3), 2003.

[31] C. Estan, G. Varghese, M. Fisk. “Bitmap algorithms for counting active flows on high speed links”. Proc. of 3rd ACM SIGCOMM conference on Internet measurement. Miami Beach, Florida, USA, 2003.

[32] C. Estan, G. Varghese. “Building a better NetFlow”. Proc. of ACM SIGCOMM ’04, Portland, Oregon, USA, 2004.

[33] K. Fall, S. Floyd. “Router mechanisms to support end-to-end congestion control”. [online] ftp://ftp.ee.lbl.gov/papers/collapse.ps.

[34] W. Fang, L. Peterson. “Inter-as traffic patterns and their implications”. Proc. of IEEE GLOBECOM’99, Rio de Janeiro, Brazil, 1999.

[35] W. Feng, D. D. Kandlur, D. Saha, K. G. Shin. “A Self-Configuring RED Gateway”. Proceedings of INFOCOM, New York, NY, USA, 1999.

[36] W. Feng, K.G. Shin, D.D. Kandlur, D. Saha. “The BLUE active queue management algorithms”. *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, 513-528, August 2002.

[37] S. Floyd. “HighSpeed TCP for Large Congestion Windows”. RFC 3649, Experimental, December 2003.

[38] S. Floyd. “Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: Oneway Traffic”. ACM Computer Communication Review, 30 - 47, Vol. 21 , Issue 5, October 1991.

[39] S. Floyd, R. Gummadi, S. Shenker. “Adaptive RED: An algorithm for increasing the robustness of RED’s active queue management”. August 2001, online: http://www.icir.org/floyd/papers/adaptiveRed.pdf.

[40] S. Floyd, V. Jacobson. “Random early detection gateways for congestion avoidance”. IEEE/ACM Transactions on Networking, vol. 1, pp. 397 -413, Aug. 1993.

[41] R. J. Gibbens, F. P. Kelly. “Distributed Connection Acceptance Control for a Connectionless Network”, 16th International Teletraffic Conference, Edimburgh, June 1999, pp. 397-413.

[42] L. Guo, I. Matta. The War between Mice and Elephants. Proc. of IEEE ICNP, Riverside, USA, 2001.

[43] F. Hao, M. S. Kodialam, T. V. Lakshman. "ACCEL-RATE: a faster mechanism for memory efficient per-flow traffic estimation". ACM SIGMETRICS '04, New York, NY, USA, 2004.

[44] F. Hao, M. Kodialam, T. V. Lakshman, H. Zhang. "Fast, Memory-Efficient Traffic Estimation by Coincidence Counting". Proc. of IEEE INFOCOM 2005, Miami, Florida, USA, 2005.

[45] D. Hartfiel. "Nonhomogeneous matrix products", World Scientific, 2002.

[46] N. Hohn, D, Veitch. "Inverting sampled traffic". Proc. of 3rd ACM SIGCOMM conference on Internet measurement. Miami Beach, Florida, USA, 2003.

[47] C. Hollot, V. Misra, D. Towsley, W.B. Gong. "Analysis and design of controllers for AQM routers supporting TCP flows". *IEEE Transactions on Automatic Control*, pp. 945-959 June, 2002.

[48] C. V. Hollot Y. Chait. "Non-linear stability analysis of a class of TCP/AQM networks". Proceedings of IEEE Conference on Decision and Control, Orlando, FL, USA, December 2001.

[49] V. Jacobson. "Congestion avoidance and control". Proceedings of ACM SIGCOMM '88, Stanford, CA, USA, August 1988.

[50] R. Jain. "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling". John Wiley and Sons, INC., 1991.

[51] S. Jaiswal, G. Iannaccone, C. Diot, D. F. Towsley. "Inferring TCP connection characteristics through passive measurements". Proceedings of INFOCOM, Hong Kong, China, March 2004.

[52] H. Jiang , C. Dovrolis. "Passive estimation of TCP round-trip times". ACM SIGCOMM Computer Communication Review, v.32 n.3, p.75-88, July 2002.

[53] S.Jin, L. Guo, I. Matta, A. Bestavros. "A spectrum of TCP-friendly window-based congestion control algorithms". *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, 341-355, 2003.

[54] R. Johari D. Tan. "End-to end congestion control for the internet: delays and stability". *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 818–832, 2001.

[55] D. Katabi, M. Handley, C. Rohr. "Internet Congestion Control for Future High Bandwidth-Delay Product Environments". Proc. of ACM SIGCOMM, Pittsburgh, PA, USA, 2002.

[56] F. P. Kelly. "Mathematical modelling of the internet". *Proceedings of ICIAM 99, 4th International Congress of Industrial Applied Mathematics*, Edinburgh, UK, July 1999.

[57] F.P. Kelly, A.K. Maulloo, D.K.H. Tan. "Rate control for communication networks: shadow. prices, proportional fairness and stability". Journal of the Operational Research Society, Vol. 49 (3), pp. 237-252, March 1998.

[58] T. Kelly. “Scalable TCP: Improving Performance in Highspeed Wide Area Networks”. ACM SIGCOMM Computer Communication Review Volume 33, Issue 2, April 2003.

[59] I. Khalifa, L. Trajkovic. ”An overview and comparison of analytical TCP models”. Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS ’04, Volume: 5, pp. :469–472, 23-26 May 2004.

[60] L. Kleinrock. “Queuing Systems Vol. II: Comp. App.”. Wiley, 1976.

[61] M. Kodialam, T. V. Lakshman, S. Mohanty. “Runs based Traffic Estimator (RATE): A Simple, Memory Efficient Scheme for Per-Flow Rate Estimation”. Proc. of IEEE INFOCOM, Hong Kong, March 2004.

[62] A. Kortebi, L. Muscariello, S. Oueslati, J. W. Roberts. “Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing”. ACM SIGMETRICS ’05, Banff, Canada, 2005.

[63] B. Krishnamurthy, S. Sen, Y. Zhang, Y. Chen. “Sketch-based change detection: methods, evaluation, and applications”. Proc. of the ACM SIGCOMM IMC. Miami Beach, Florida, USA, 2003.

[64] A. Kumar, J. Xu, L. Li, J. Wang. “Data streaming algorithms for efficient and accurate estimation of flow size distribution”. Proc. of ACM SIGMETRICS ’04, New York, NY, USA, 2004.

[65] A. Kumar, J. Xu, L. Li, J. Wang. “Space-code bloom filter for efficient traffic flow measurement”. Proc. of IEEE INFOCOM 2004, Hong Kong, China, 2004.

[66] A. Kumar, J. Xu, L. Li, J. Wang. “Data streaming algorithms for efficient and accurate estimation of flow size distribution”. ACM SIGMETRICS ’04, New York, NY, USA, 2004.

[67] K. Kumaran, M. Mandjes. “The buffer-bandwidth trade-off curve is convex”. Queueing Systems, 38 (2001), no. 4, 471–483.

[68] K. Kumaran, M. Mandjes, A. Stolyar. “Convexity properties of loss and overflow functions”. Operations Research Letters, 31 (2003), no. 2, 95–100.

[69] S. S. Kunniyur R. Srikant. ”Stable, Scalable, Fair Congestion Control and AQM schemes that achieve high utilisation in the internet”. *IEEE Transactions on Automatic Control*, vol. 48, no. 11, pp. 2024–2029, 2003.

[70] S. Kunniyur, R. Srikant. “Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management”. *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, 286-299, April 2004.

[71] T. V. Lakshman, U. Madhow. “The performance of TCP/IP for networks with high bandwidth-delay products and random loss”. *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, 336–350, June 1997.

[72] A. Leizarowitz, R. Stanojević, R. Shorten, “Tools for the analysis and design of communication networks with Markovian dynamics”. Proceedings of IEE, Control Theory and Applications, vol. 153(5), pp. 506-519, 2006.

[73] D. Lin, R. Morris. “Dynamics of random early detection”. Proc. of ACM SIGCOMM, Cannes, France, 1997.

[74] S. Low, L. Andrew, B. Wydrowski. “Understanding XCP: Equilibrium and Fairness”. Proc. of IEEE Infocom, Miami, FL, March 2005.

[75] S. Low, F. Paganini, J. Doyle. ”Internet congestion control”. *IEEE Control Systems Magazine*, vol. 32, no. 1, pp. 28–43, 2002.

[76] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker. “Controlling high bandwidth aggregates in the network”. ACM SIGCOMM Computer Communication Review Volume 32, Issue 3, Pages: 62 - 73, July 2002.

[77] R. Mahajan, S. Floyd, D. Wetherall. “Controlling high-bandwidth flows at the congested router”. Proc. of IEEE ICNP, Riverside, CA, USA, 2001.

[78] S. Mascolo. ”Congestion control in high speed communication networks using the Smith principle”. *Automatica*, vol. 35, pp. 1921–1935, 1999.

[79] L. Massoulie. ”Stability of distributed congestion control with heterogeneous feedback delays”, *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 895–902, 2002.

[80] J. Mo, J. Walrand. “Fair end-to-end window-based congestion control”. IEEE/ACM Transactions on Networking, Vol. 8, No. 5 October, 2000.

[81] J.R. Norris. ”Markov Chains”. Cambridge University Press, 1997.

[82] NetFlow documentation, available online: http://www.cisco.com/warp/public/732/Tech/nmp/netflow/

[83] NLANR IP traces: http://pma.nlanr.net/Special/.

[84] NLANR IP traces, online, MRA (ftp://pma.nlanr.net/traces/daily/20050416/MRA-1113621669-1.tsh.gz), FRG (http://pma.nlanr.net/Traces/Traces/old/ 20011015/FRG-1114614867-1.tsh.gz), ALB(ftp://pma.nlanr.net/traces/long/ipls/1/IPLS-CLEV-20020814-103000-0.gz).

[85] E.A. Ok. “Real Analysis with Economic Applications”. Princeton University Press, 2007.

[86] T. J. Ott, T. V. Lakshman, L. H. Wong. “SRED: Stabilized RED”. Proc. IEEE INFOCOM, New York, March 1999.

[87] J. Padhye, V. Firoiu, D. F. Towsley, J. F. Kurose. "Modeling TCP Reno performance: a simple model and its empirical validation". *IEEE/ACM Transactions on Networking*, Volume 8 , Issue 2 ,April 2000.

[88] R. Pan, L. Breslau, B. Prabhakar, S. Shenker. "Approximate Fairness through Differential Dropping". ACM SIGCOMM Computer Communication Review Volume 33 , Issue 2, April 2003

[89] R. Pan, B. Prabhakar, K. Psounis. "CHOKe: A stateless AQM scheme for approximating fair bandwidth allocation". Proc. of IEEE INFOCOM, Tel Aviv, Israel, 2000.

[90] B. Pearlmutter. "Bounds on query convergence". Preprint 2005. Online: http://arxiv.org/abs/cs.LG/0511088.

[91] L. Qiu, Y. Zhang , S. Keshav. "Understanding the performance of many TCP flows". Computer Networks: Volume 37, Issue 3-4, pp. 277–306, 2001

[92] R.Shorten, D. Leith, J. Foy R. Kilduff. "Analysis and design of synchronised communication networks". Automatica, vol. 41, pp. 725–730. 2005.

[93] I.A. Rai, E.W. Biersack, G. Urvoy-Keller. "Size-based scheduling to improve the performance of short TCP flows". IEEE Network, 2005.

[94] S. Ramabhadran, G. Varghese. "Efficient implementation of a statistics counter architecture". ACM SIGMETRICS '03, San Diego, CA, USA, 2003.

[95] M. Roeder, B. Lin. "Maintaining exact statistics counters with a multi-level counter memory". Proc. of IEEE Globecom, Dalas, USA, 2004.

[96] D. Shah, S. Iyer, B. Prabhakar, N. McKeown. "Analysis of a statistics counter architecture". IEEE Micro, 22(1):76-81, 2002.

[97] R. Shorten, C. Kellett, D. Leith. "On the Dynamics of TCP's Higher Moments". IEEE Communications Letters, vol 11(2), 2007.

[98] Shorten, R., King. C., Wirth, F. and Leith, D. "Modelling TCP in drop-tail and other environments". Accepted for publication in Automatica, 2007.

[99] R. Shorten, F. Wirth, D. Leith. "A positive systems model of TCP-like congestion control: Asymptotic results". IEEE/ACM Transactions on Networking 14(3): 616-629 (2006).

[100] M. Shreedhar, G. Varghese. "Efficient fair queueing using deficit round-robin". *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, June 1996.

[101] H. Song, S. Dharmapurikar, J. Turner, J. Lockwood. "Fast hash table lookup using extended bloom filter: an aid to network processing". Proc. of SIGCOMM, Philadelphia, USA, 2005.

[102] R. Srikant. "Internet congestion control". Control theory, 14, Birkhäuser Boston Inc., Boston, MA, 2004.

[103] O. Stenflo. "Uniqueness of invariant measures for place-dependent random iterations of functions". IMA Vol. Math. Appl., 132 (2002), 13-32.

[104] R. Stanojević. "Scalable heavy-hitter identification". Preprint, 2006.

[105] R. Stanojević, R. Shorten. "Drop counters are enough". Proceedings of IEEE IWQoS, Chicago, USA, 2007.

[106] R. Stanojević. "Small active counters". Proceedings of IEEE Infocom, Anchorage, USA, 2007.

[107] R. Stanojević, R. Shorten. "Beyond CHOKe: Stateless fair queueing". Proceedings of NET-COOP 2007, LNCS, Springer, vol.4465.

[108] R. Stanojević, R. Shorten. "How expensive is link utilization?". Proceedings of NET-COOP 2007, LNCS, Springer, vol.4465.

[109] R. Stanojević, R. Shorten, C. Kellet. "Adaptive tuning of Drop-Tail buffers for reducing queueing delays". IEEE Communications Letters, vol. 10(7), July, 2006.

[110] I. Stoica, S. Shenker, H. Zhang. "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks". *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 33-46, February 2003.

[111] B. Suter, T.V. Lakshman, D. Stiliadis, A.K. Choudhury. "Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing". IEEE Journal on Selected Areas of Communications 17 (6) (1999) 1159-1170.

[112] A. Tang, J. Wang, S. H. Low. "Understanding CHOKe: throughput and spatial characteristics". IEEE/ACM Transactions on Networking. Vol 12 (4),2004.

[113] Traffic measurements. [Online]. Available: http://www.caida.org/tools/measurement/skitter/RSSAC/.

[114] B. Veal, K. Li, D. Lowenthal. "New Methods for Passive Estimation of TCP Round-Trip Times". Proceedings of PAM, Boston, MA, USA, March 2005.

[115] C. Villamizar, C. Song. "High Performance TCP in ANSNET". ACM Computer Communication Review, 24(5), 1994

[116] G. Vinnicombe. "On the stability of networks operating TCP-like congestion control". Cambridge Univ. Statistical Laboratory Research Report, 2000-398., Tech. Rep., 2000.

[117] G. Vu-Brugier, R. Stanojević, D. Leith, R. Shorten. "A critique of recently proposed Buffer-Sizing strategies". ACM Computer Communications Review, vol. 37(1), January 2007.

[118] F. Wirth, R. Stanojević, R. Shorten, D. Leith. "Stochastic equilibria of AIMD communication networks". SIAM Journal on Matrix Analysis and Applications, vol. 28(3), pp. 703-723, 2006.

[119] D. Wischik N. McKeown. "Part I: Buffer sizes for core routers". ACM Computer Communication Review 35(3): 75-78 (2005).

[120] B. Wydrowski, M. Zukerman. "MaxNet: A congestion control architecture for maxmin fairness". IEEE Communications Letters, vol. 6, no. 11, Nov. 2002, pp.512-514.

[121] L. Xu, K. Harfoush, I. Rhee. "Binary increase congestion control for fast long-distance networks". Proceedings of IEEE INFOCOM, Hong Kong, March 2004.

[122] Y. Yang, S. Lam. "General AIMD Congestion Control". Proceedings of the IEEE International Conference on Network Protocols, Osaka, Japan, November 2000

[123] Y. Zhang, L. Breslau, V. Paxson, S. Shenker. "On the Characteristics and Origins of Internet Flow Rates". Proceedings of ACM SIGCOMM, Aug. 2002, Pittsburgh, PA.

[124] Y. Zhang, S. Singh, S. Sen, N. Duffield, C. Lund. "Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications". Proc. ACM IMC '04. Taormina, Sicily, Italy, 2004.

[125] Q. Zhao, J. Xu, Z. Liu. "Design of a Novel Statistics Counter Architecture with Optimal Space and Time Efficiency". Proc. of ACM SIGMETRICS '06, France, 2006.

[126] Z. Zhao, S. Darbha, A.L.N. Reddy. "A method for estimating the proportion of nonresponsive traffic at a router". *IEEE/ACM Transactions on Networking*, vol.12, no. 4, 708- 718, 2004.

[127] Q. Zhao, J. Xu, Z. Liu. "Design of a Novel Statistics Counter Architecture with Optimal Space and Time Efficiency". ACM SIGMETRICS '06, France, 2006.