DYNAMIC THRESHOLD-BASED ALGORITHMS FOR COMMUNICATION NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Mehmet Altan Toksöz

August 2009

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Nail Akar(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ezhan Karaşan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. İbrahim Körpeoğlu

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

DYNAMIC THRESHOLD-BASED ALGORITHMS FOR COMMUNICATION NETWORKS

Mehmet Altan Toksöz
M.S. in Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Nail Akar
August 2009

A need to use dynamic thresholds arises in various communication networking scenarios under varying traffic conditions. In this thesis, we propose novel dynamic threshold-based algorithms for two different networking problems, namely the problem of burst assembly in Optical Burst Switching (OBS) networks and of bandwidth reservation in connection-oriented networks. Regarding the first problem, we present dynamic threshold-based burst assembly algorithms that attempt to minimize the average burst assembly delay due to burstification process while taking the burst rate constraints into consideration. Using synthetic and real traffic traces, we show that the proposed algorithms perform significantly better than the conventional timer-based schemes. In the second problem, we propose a model-free adaptive hysteresis algorithm for dynamic bandwidth reservation in a connection-oriented network subject to update frequency constraints. The simulation results in various traffic scenarios show that the proposed technique considerably outperforms the existing schemes without requiring any prior traffic information.

Keywords: Burst assembly algorithms, optical burst switching, dynamic bandwidth reservation, adaptive hysteresis

ÖZET

İLETİŞİM AĞLARİ İÇİN DİNAMİK EŞİK-TABANLI ALGORİTMALAR

Mehmet Altan Toksöz
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Nail Akar
Ağustos 2009

Eşikleme mekanizmaları iletişim ağlarının çeşitli alanlarında kullanılmaktadır. Bu tezde, iletişim ağlarının iki temel probleminde kolayca uygulanabilen çeşitli dinamik eşikleme algoritmaları önerildi. Bunlar Optik Çoğuşum Anahtarlama (OBS) ağlarında çoğuşum birleştirme ve bağlantı odaklı ağlarda bant genişliği rezervasyonudur. İlk problemle ilgili olarak, çoğuşum birleştirme işleminden dolayı oluşan ortalama çoğuşum birleştirme gecikmesini minimuma indirmeye çalışan aynı zamanda çoğuşum oluşturma frekansı kısıtlamalarını hesaba katan iki tane dinamik çoğuşum birleştirme algoritması sunuldu. Sentetik ve gerçek trafik izleri kullanılarak, önerilen algoritmaların performansının geleneksel algoritmalarınkinden daha iyi olduğu gösterildi. İkinci problemde, güncelleme frekansına uyan bağlantı-tabanlı ağlarda dinamik bant genişliği rezervasyonu için modele gereksinimi olmayan ve telefon görüşme bazlı uyarlanabilir histerez algoritması önerildi. Çeşitli trafik senaryolarında, önerilen tekniğin herhangi bir ön trafik bilgisi gerektirmeden geleneksel metotlardan daha iyi çalıştığı simülasyon sonuçlarıyla gösterildi.

Anahtar Kelimeler: Çoğuşum birleştirme algoritmaları, optik çoğuşum anahtarlama, dinamik band genişliği rezervasyonu, uyarlanabilir histerez

ACKNOWLEDGMENTS

I am sincerely grateful to Assoc. Prof. Dr. Nail Akar for his supervision, guidance, insights and support throughout the development of this work. His broad vision and profound experiences in engineering has been an invaluable source of inspiration for me.

I would like to thank to the members of my thesis jury, Assoc. Prof. Dr. Ezhan Karaşan and Asst. Prof. Dr. İbrahim Körpeoğlu for reviewing this dissertation and providing helpful feedback.

I would like to especially thank to my family for their help and supports to my whole life.

Many thanks to Alper Kabasakal, Fazlı Kaybal, Osman Günay, Vahdettin Taş, Ahmet E. Sezgin, Ahmet Güngör, Ömür Arslan, Osman Gürlevik, Fırat Karataş, Mehmet Köseoğlu and Mehmet Akif Yazıcı for their kind friendship.

Financial support of The Scientific and Technological Research Council of Turkey (TUBITAK) for the Graduate Study Scholarship Program(2210) is gratefully acknowledged.

This work is also supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK) under project No. EEEAG106E046.

Contents

List of Figures

List of Tables

Dedicated to my parents, my sister, and my brother...

Chapter 1

INTRODUCTION

1.1 Introduction to the Burst Assembly Problem

Optical Burst Switching (OBS) has been receiving increasing attention as an alternative transport architecture for the next-generation optical Internet in academia and also in industry [2],[3],[4]. There are several features of OBS that make it a viable technology. Firstly, in OBS, data travels through the network in the form of relatively long bursts and all-optically. A number of client packets are assembled into a data burst at the edge of an OBS network while the followings are taken into consideration: (i) increasing burst lengths helps relax optical switching-speed requirements, (ii) reducing burst lengths also reduces delays stemming from burst assembly. A second principle of OBS is the separation of the control and data planes where the data plane is all-optical but the control plane can be optical-electronic in the sense that control packets are processed electronically at the core nodes. Once a data burst is formed at the edge device, the ingress node prepares a control message on behalf of the data burst and transmits it in the form of a Burst Control Packet (BCP) over the control plane

towards the egress node. The BCP carries information about the data burst, such as its length, destination, arrival time, etc. A receipt of a BCP by a core node initiates a configuration of the node by means of reserving resources for the burst when available. On the other hand, the data burst is transmitted over the data plane after an offset time which has to be at least as long as the sum of the per-hop processing times that the corresponding BCP will encounter. In a typical OBS network with no buffers, the end-to-end delay of a single packet is then written as the sum of the offset time and the burst assembly delay, the latter forming the scope of our study. In this part of the thesis, we propose dynamic threshold-based burst assembly algorithms that attempt to minimize the average burst assembly delay due to burstification process while taking the burst rate constraints into consideration. The proposed algorithms minimize either the average packet or byte delay and their performance are comparatively studied against timer- and size-based conventional burst assembly mechanisms. Using synthetic and real traffic traces, we show that the proposed algorithms perform significantly better than the existing schemes.

1.2 Introduction to the Dynamic Bandwidth Reservation Problem

In order to solve the problem of frequently setting up and tearing down a huge number of connections in large networks, a number of connection-oriented network technologies have been deployed like Asynchronous Transfer Mode (ATM) [5], Multiprotocol Label Switching (MPLS) [6], or a single aggregate Resource ReserVation Protocol (RSVP) reservation [7]. In these technologies, connections belonging to the same class can be grouped on a virtual tunnel to be treated in the same way as a group (Fig. 1.1). In ATM, the bandwidth of the physical link is logically divided into separate Virtual Paths (VPs) by using the Virtual Path

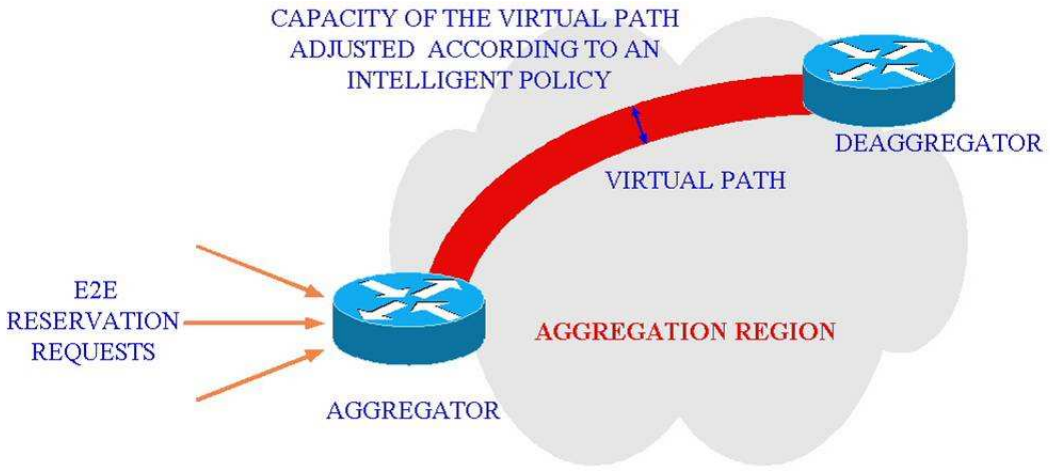

Figure 1.1: A Generic Virtual Path

Identifier (VPI) of the corresponding path. Also each VP in a link is divided into Virtual Circuits (VCs) by the Virtual Channel Identifier (VCI) of each VC. The bandwidth of a VP can be dynamically adjusted by controlling the number of VCs included in that VP. MPLS technology presents efficient engineering granularity by configurable virtual tunnels which are called Label Switched Paths (LSPs). By MPLS Traffic Engineering (MPLS TE), the capacity of these LSPs can be adjusted without tearing down and reestablishing the current connection.

1.3 Outline

In Chapter 2, we first give the basics of an OBS network. Then, we describe the burst assembly process and summarize the conventional as well as the proposed burst assembly algorithms. Both by analysis and simulations, we compare the performances of these various algorithms. Chapter 3 addresses the problem of dynamic bandwidth reservation. First, we describe the problem and present a number of scenarios in which this problem arises. We then present a number of existing schemes for this purpose as well as our proposed technique. At the end of this chapter, we present numerical examples to validate the proposed approach. Finally, Chapter 4 concludes this thesis.

Chapter 2

OBS BURST ASSEMBLY ALGORITHMS SUBJECT to BURST RATE CONSTRAINTS

2.1 Motivation and Related Work

An OBS network basically contains two kinds of nodes namely edge and core nodes as shown in Fig. 2.1. The burst assembly process is performed in the ingress edge nodes by receiving the incoming IP packets from an outside access network into bursts by aggregating them (Fig. 2.2). When a burst data packet (BDP) is created, first a burst control packet (BCP), which contains the knowledge of burst arrival time, burst length, and routing information, is sent out. Between a BDP and BCP, there is an offset time which is used by the intermediate node to configure the switch for wavelength allocation.

Various burst assembly algorithms have been proposed to aggregate a number of client packets (such as IP packets) into data bursts. Typically, an ingress node maintains per-destination queues to store client packets awaiting burstification

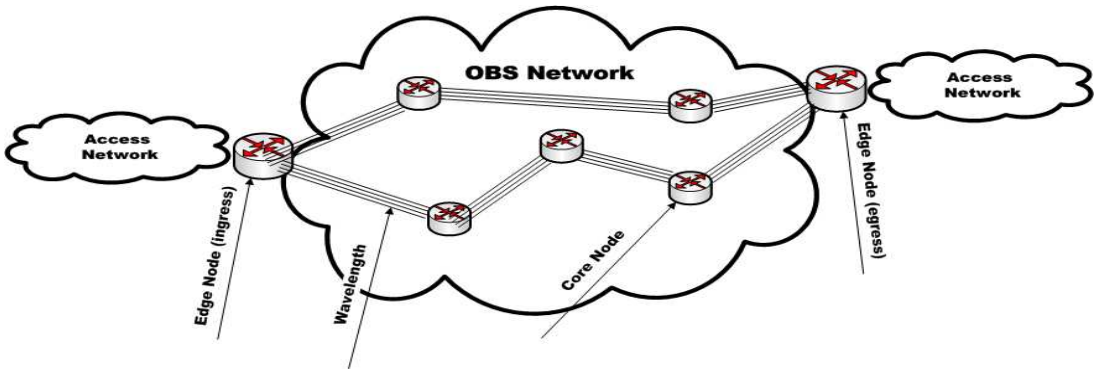

Figure 2.1: An OBS Network

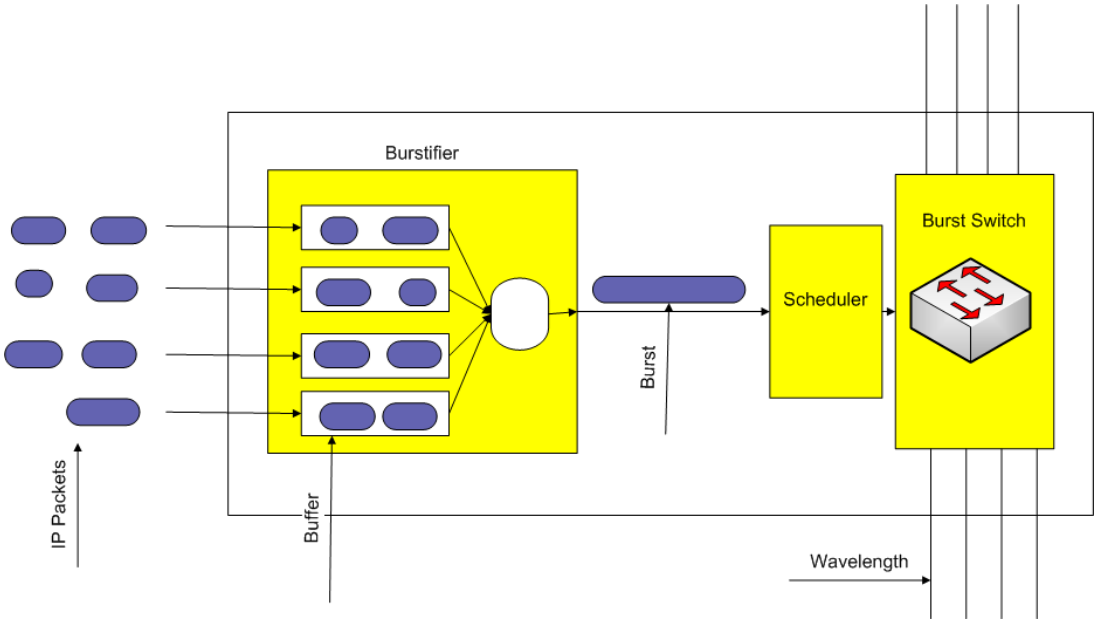

Figure 2.2: Structure of an Edge Node

that are destined for a specific destination. Multiple instances of a burst assembly algorithm are run for each of these queues which decide when the packets in the queue should be aggregated into a burst and sent out. Other variations are also possible in which multiple queues are maintained for each destination, one for each QoS-class and different burst assembly algorithms may be run for each of these queues. Such scenarios are left outside the scope of our study.

Four classes of burst assembly algorithms are available in the literature, namely timer-based, size-based, hybrid (timer- and size-based), and dynamic threshold-based algorithms. In timer-based burst assembly [8], a timer is started

once a client packet arrives at an empty burst assembly buffer. This timer expires after T (in units of seconds) by which time all packets awaiting in the burst assembly buffer are aggregated into a burst and sent out. The timer parameter T is chosen as the largest allowable delay due to burstification. Moreover, a lower burst length parameter B_{min} (in units of bytes) is used along with timers to keep the load on the control channel at reasonable levels. For this purpose, padding is used if the number of bytes awaiting in the buffer upon timer expiration is less than B_{min}. The second class of algorithms are size-based and when the assembly buffer size reaches or exceeds a size parameter B then all packets in the buffer are aggregated into a burst [9]. Clearly, B should be set to a value larger than the lower limit B_{min}. However, these two classes of burst assembly algorithms have their problems of their own. Size-based algorithms suffer from excessive delays especially when the traffic load is light. On the other hand, under heavy traffic load, timer-based algorithms experience a longer average delay than size-based algorithms. The third class of algorithms, namely hybrid timer- and size-based algorithms, keep track of the assembly buffer occupancy, as well as the time since the arrival of the first packet into the assembly buffer. A representative algorithm in this class is proposed in [10] in which an upper burst length limit B_{max} (in units of bytes) is imposed on the pure timer-based scheme. In this proposal, if the buffer occupancy is to exceed B_{max} before the timer expires, a portion of the awaiting packets are aggregated into a burst immediately without having to wait for the timer to expire. The final class of algorithms are based on the use of dynamic thresholds, where either the timer parameter T or the size parameter B or both are adjusted dynamically [11],[12]. Recently, various methods using dynamic thresholds have been proposed in [13],[14],[15].

The statistical characteristics of input IP traffic and the generated burst traffic significantly affects performance of an optic network [10], [9], [8], [16]. It has been shown that today's IP traffic is statistically self-similar [17]. Several works have been done to investigate if the self-similarity or long range dependency of

input ip traffic can really affects the performance of the core of an optic network after the assembly process [8], [18], [19]. Reference [8] claims that assembly algorithms reduces the self-similarity of the input IP traffic and it increases the performance. On the other hand, [16] and [18] report that the long range dependency is not reduced after assembly process. However, long range dependency in the assembled traffic does not have any impact on burst loss performance at the core nodes. Only short range characteristics smooth the traffic which increase the loss performance. Several other studies have supported the results in [16] and [18].

The assumptions we have for the burst assembly problem studied in this chapter are given below:

a) We focus on burst assembly algorithms whose average burst generation rates (both short- and long-term rates) are upper bounded by a desired burst rate parameter called β (in units of bursts/sec). We have two main goals with this approach. Firstly, β determines the frequency of BCPs traveling on the control channel and by adjusting β, one can control the control plane load in the system and thus limit BCP queueing delays due to processing. Secondly, a fair comparison of two burst assembly algorithms is only meaningful when their average burst rates are the same since algorithms with higher burst generation rates are to naturally outperform others in terms of burstification delays.

b) We impose lower and upper burst length limits B_{min} and B_{max} in units of bytes as in [10].

c) Given the above two constraints, our goal is to devise a burst assembly scheme that minimizes

   - the average packet delay D_P which is defined as the average of all packet delays in the assembly buffer, or

- the average byte delay D_B which is defined as the weighted average of all packet delays where the weights are taken to be packet lengths in units of bytes. A burst assembly algorithm that attempts to minimize D_B needs to keep track of packet lengths as well.

d) Finally, we seek a model-free algorithm which is also simple to implement. If the traffic statistics were known, one can obtain an analytical solution as in [20] but generally burstifiers do not have a good understanding of the statistical properties of the traffic streams they need to process. Moreover, traffic is generally unpredictable which leads us to use traffic-adaptive assembly algorithms.

In our study, we mainly focus on the reduction of the delays D_P and D_B that are caused by the assembly process and we develop two dynamic threshold-based algorithms that attempt to minimize one of these two delay parameters under a burst rate constraint β. We then compare our results to those obtained with conventional timer-based schemes under realistic traffic and packet length distribution scenarios. The remainder of this chapter is organized as follows. In Section 2.2, we present an overview of existing timer-based and size-based algorithms. The two algorithms we propose are presented in Section 2.3. Section 2.4 provides numerical results concerning the performance evaluation of existing and proposed algorithms under different traffic scenarios.

2.2 Burst Assembly Algorithms

In this section, we will first present three conventional burst assembly algorithms, the first two being timer-based, and the third one being size-based. We will then present the two algorithms we propose.

2.2.1 Timer-based Min-Length Burst Assembly

This basic algorithm is given as Algorithm 1. It is called Timer-based Min-Length Burst Assembly algorithm, or in short *timer-min* since the algorithm is timer-based and also the minimum burst length limit is enforced. In this algorithm, the inter-burst time is fixed to the timer threshold T which will be set to $1/\beta$. The worst case delay then equals T and assuming packet arrivals for burst i occur uniformly in the interval $((i-1)T, iT)$, the average packet delay is $T/2 = 1/(2\beta)$. This algorithm does not employ an upper limit B_{max} on burst lengths. The next algorithm attempts to modify the current one by imposing an upper burst length limit.

Algorithm 1 *timer-min*

PARAMETERS:
t: time counter
T: assembly time window
i: burst index
$p_i(t)$: data accumulated for the i-th burst at time t (bytes)
B_{min}: lower burst length limit (bytes)

THE ALGORITHM

```
t ⇐ 0 {start the time counter at t = 0}
if t = T then
  if p_i(t) ≥ B_min then
    p_i(t) ⇐ 0 {send p_i(t) as burst i immediately}
    i ⇐ i + 1 {increase burst counter}
    t ⇐ 0 {reset time counter}
  else
    p_i(t) ⇐ B_min {increase the data size to B_min with padding}
    p_i(t) ⇐ 0 {send p_i(t) as burst i immediately}
    i ⇐ i + 1 {increase burst counter}
    t ⇐ 0 {reset time counter}
  end if
end if
```

2.2.2 Timer-based Min-Max-Length Burst Assembly

This modified algorithm is given as Algorithm 2. It is called Timer-based Min-Max-Length Burst Assembly algorithm, or in short *timer-min-max*, since the upper burst length limit B_{max} is also imposed. In this algorithm, when the data accumulated for the i-th burst at time t, denoted by $p_i(t)$, at the epoch of timer expiration exceeds B_{max} then a maximum number of packets whose packet length sum does not exceed B_{max} is sent out as burst i. The remaining packets in the burst assembly buffer wait for the next opportunity. In both timer-based algorithms, a decision to assemble is made synchronously without paying attention to the assembly buffer content. Worst case delays are bounded when $B_{max} \rightarrow \infty$ and the burst rate requirement β is inherently taken care of by setting $T = 1/\beta$. One of the main goals of this study is to explore alternative methods that would potentially benefit from asynchronous burst assembly in terms of either average packet or byte delays.

2.2.3 Fixed Threshold-based Burst Assembly

Assume that the average packet arrival rate to the assembly buffer is known and is denoted by λ. Let us assume $b = \lambda/\beta$ is an integer. We can then use a burst assembly algorithm that generates a burst every time b packets are accumulated in the buffer. This strategy ensures a burst generation rate of β. This assembly method will be referred to as *fixed-threshold*. It is then crucial to know whether this policy is optimal. Let us assume renewal inter-packet arrival times with mean α. Let us use an arbitrary probabilistic policy that assembles when b_i packets are present with probability $p_i, 1 \leq i \leq N$. To enforce a β burst generation rate, we should have $\sum_{i=1}^{N} b_i p_i = b$. An arbitrary packet will then belong to a burst with length b_i with probability $\frac{b_i p_i}{b}, 1 \leq i \leq N$. The average packet delay then

Algorithm 2 *timer-min-max*

PARAMETERS:
t: time counter
T: assembly time window
i: burst index
$p_i(t)$: data accumulated for the i-th burst at time t (bytes)
B_{min}: lower burst length limit (bytes)
B_{max}: upper burst length limit (bytes)

THE ALGORITHM

```
t ⇐ 0 {start the time counter at t = 0}
if t = T then
  if p_i(t) < B_min then
    p_i(t) ⇐ B_min {increase the data size to b with padding}
    p_i(t) ⇐ 0 {send p_i(t) as burst i immediately}
    i ⇐ i + 1 {increase burst counter}
    t ⇐ 0 {reset time counter}
  else if p_i(t) ≥ B_min and p_i(t) < B_max then
    p_i(t) ⇐ 0 {send p_i(t) as burst i immediately}
    i ⇐ i + 1 {increase burst counter}
    t ⇐ 0 {reset time counter}
  else
    p_i(t) ⇐ p_i(t) − B_max {send subtracted p_i(t) as burst i immediately}
    i ⇐ i + 1 {increase burst counter}
    t ⇐ 0 {reset time counter}
  end if
end if
```

becomes

$$D_P = \frac{\alpha}{2b} \sum_{i=1}^{N} p_i b_i (b_i - 1) \tag{2.1}$$

It is obvious that the average delay is minimized with a deterministic policy $N = 1$ that generates a burst every time b packets are accumulated in the buffer. In this case

$$D_P = \frac{\alpha(b-1)}{2} \tag{2.2}$$

which provides an expression for the optimum average packet delay. For instance, if λ is 50000 packets/second and β is 1000 bursts/second, then an optimal burst assembly policy will be to wait for 50 packets to arrive for burst assembly. It is very likely that the value $b = \lambda/\beta$ may not be an integer. Say the value b is in the form $x + y$ where x is the integer part of b and y is the fractional part where $0 < y < 1$. The optimal policy in this case is one which assembles packets when x packets are accumulated with probability $1 - y$, or when $x + 1$ packets are accumulated with probability y. There are several drawbacks of this dynamic threshold-based burst assembly mechanism described above:

- The method is very sensitive to the average packet arrival rate λ; a deviation of the estimate from the actual value will lead to burst generation rates that differ from β.

- When the packet arrival process is a non-renewal process, using a fixed threshold of b packets for burst assembly would generate bursts at a long-term rate of β but over relatively shorter terms, the burst rate constraints can be violated leading to occasional problems on the control plane. For this scenario, a need arises to employ a dynamic-threshold algorithm to keep track of changes in the arrival process so as to maintain the short-term burst rate averages at a desired rate of β as well. This situation appears to worsen with non-stationary traffic.

- When b packets are accumulated, most of these packets can turn out to be relatively large packets making the total length exceed B_{max}. It appears to

be very difficult to enforce in this algorithm the upper limit B_{max} which is in units of bytes. The lower limit can be enforced by padding.

- Since the algorithm keeps track of only the number of packets and not their lengths, this algorithm can not differentiate between packet and byte delays. If the focus is the minimization of the byte delays, then we should resort to a modified algorithm.

Although a fixed-threshold-based burst assembly algorithm has nice theoretical properties, we still seek a method that is model-free, which is simple to implement, and which keeps track of bytes for the purposes of enforcing the lower and upper bandwidth limits as well as the minimization of average byte delay in addition to average packet delay.

2.3 Proposed Burst Assembly Algorithms

The proposed algorithms we propose do not require any prior information such as the average packet arrival rate or average bit rate. Another strength of the proposed algorithms is their simplicity as compared to other dynamic-threshold algorithms. Next, we present these two algorithms.

2.3.1 Packet-based Dynamic-Threshold Algorithm for Burst Assembly

This algorithm (given as Algorithm 3) is an entirely packet-based algorithm and it is referred to as *dyn-threshold-packet* in short. In this algorithm, we keep track of the packet count in the assembly buffer and we aim to minimize the average packet delay due to burstification. The lower and upper burst length limits are given in units of packets and they are denoted by L_{min} and L_{max}, respectively.

We also maintain a counter called *bucket* to indicate the dynamic threshold used in our burst assembly algorithm. Each time a packet, say packet k, arrives at the assembly buffer, the bucket is decremented by β times the inter-arrival time between packets $k-1$ and k. A decision for burst assembly is made only when the current packet count exceeds the bucket value. When an assembly decision is made, the bucket is incremented by one. To enforce lower and upper burst length limits, the bucket is allowed to take values in the interval $[L_{min}, L_{max}-1]$. We have also added an expiration time T_{max} for a burst to meet the worst case delay requirement. Even if the conditions for a burst are not met in low traffic load, the expiration time mechanism would force the generation of the burst.

2.3.2 Byte-based Dynamic Threshold Algorithm for Burst Assembly

This algorithm (given as Algorithm 4) is a byte-based algorithm and it is referred to as *dyn-threshold-byte* in short. In this algorithm, we keep track of the byte count in the assembly buffer and we aim to minimize the average byte delay due to burstification. The reason for this is that client packet lengths are variable; short and long packets are to be treated differently since they contribute differently to the overall byte delay. The lower and upper burst length limits are given in units of bytes and they are denoted by B_{min} and B_{max}, respectively. Similar to the *dyn-threshold-packet* algorithm, we maintain a *bucket* to indicate the dynamic threshold used in our burst assembly algorithm. Each time a packet, say packet k, arrives at the assembly buffer, the bucket is decremented by an amount in direct proportion with the inter-arrival time between packets $k-1$ and k with the constant of proportionality set to $\kappa\beta$. A decision for burst assembly is made only when the current byte count exceeds the bucket value. When an assembly decision is made, the bucket is incremented by κ. The parameter κ is the learning parameter of the system. A large value of κ indicates an algorithm

Algorithm 3 dyn-threshold-packet

PARAMETERS:
i: packet index
j: burst index
β: desired burst rate (bursts/sec)
$L(i,j)$: data accumulated for the j-th burst at the arrival epoch of i-th packet (in units of packets)
L_{min}: lower burst length limit (in units of packets)
L_{max}: upper burst length limit (in units of packets)
bucket: dynamic threshold
t: time counter
T_{max}: burst expiration time
t_i: inter-arrival time between $(i-1)$st and ith packets

THE ALGORITHM

if $L(i,j) = 1$ **then**
  $t \Leftarrow 0$ {if the assembler queue contains 1 packet, start the time counter}
end if
$bucket \Leftarrow bucket - t_i\beta$ {leak the bucket}
$bucket \Leftarrow \max(L_{min}, bucket)$ {enforce lower burst length limit}
if $L(i,j) \geq bucket$ **then**
  $L(i,j) \Leftarrow 0$ {send $L(i,j)$ as burst j immediately}
  $bucket \Leftarrow \min(bucket + 1, L_{max} - 1)$ {update bucket and enforce upper burst length limit}
  $j \Leftarrow j + 1$ {increase burst counter}
  $t \Leftarrow 0$ {reset time counter}
else if $t \geq T_{max}$ **then**
  $L(i,j) \Leftarrow L_{min}$ {increase the data size to L_{min} with padding}
  $L(i,j) \Leftarrow 0$ {send $L(i,j)$ as burst j immediately}
  $j \Leftarrow j + 1$ {increase burst counter}
  $t \Leftarrow 0$ {reset time counter}
end if

that rapidly tracks changes in incoming traffic. However, when κ is large, it is possible to occasionally deviate from the desired burst rate β. The parameter κ should be chosen by taking into consideration of these two effects. Unless otherwise stated, we use $\kappa = 1000$ in our numerical examples. To enforce lower and upper burst length limits, the bucket is allowed to take values in the interval $[B_{min}, B_{max} - P_{max}]$ where P_{max} denotes the length of the maximum-sized packet. The expiration time T_{max} is again used.

Algorithm 4 dyn-threshold-byte

PARAMETERS:
i: packet index
j: burst index
β: burst rate (bursts/sec)
$D(i,j)$: data accumulated for the j-th burst at the arrival of i-th packet (bytes)
B_{min}: lower burst length limit (bytes)
B_{max}: upper burst length limit (bytes)
P_{max}: maximum packet length (bytes)
κ: learning parameter
bucket: dynamic threshold
t: time counter
T_{max}: burst expiration time
t_i: inter-packet time between $(i-1)$st and ith packets

THE ALGORITHM

if $D(i,j)$ contains 1 packet **then**
  $t \Leftarrow 0$ {start the time counter}
end if
$bucket \Leftarrow bucket - t_i \beta \kappa$ {leak the bucket}
$bucket \Leftarrow \max(B_{min}, bucket)$ {enforce lower burst length limit}
if $D(i,j) \geq bucket$ **then**
  $D(i,j) \Leftarrow 0$ {send $D(i,j)$ as burst j immediately}
  $bucket \Leftarrow \min(bucket + \kappa, B_{max} - P_{max})$ {update bucket and enforce upper burst length limit}
  $j \Leftarrow j + 1$ {increase burst counter}
  $t \Leftarrow 0$ {reset time counter}
else if $t \geq T_{max}$ **then**
  $D(i,j) \Leftarrow B_{min}$ {increase the data size to B_{min} with padding}
  $D(i,j) \Leftarrow 0$ {send $D(i,j)$ as burst j immediately}
  $j \Leftarrow j + 1$ {increase burst counter}
  $t \Leftarrow 0$ {reset time counter}
end if

2.4 Numerical Results

We will present our numerical results basically for two different types of traffic scenarios (i) synthetic traffic (ii) real traffic traces. We will use synthetic traffic mainly to show several theoretical properties of the burst assembly algorithms mentioned above.

2.4.1 Synthetic Traffic

We study in this section two synthetic traffic models, the first one being the Poisson traffic model, and the second one being the MMPP (Markov Modulated Poisson Process) model [21]. MMPP is not a renewal process but instead a Markov renewal process in which the successive inter-arrival times depend on each other. MMPP-based traffic models capture auto-correlation and they are quite common in the modeling of Internet traffic [22].

Poisson Traffic Scenario

We first assume that the input packet traffic is stationary Poisson with arrival rate λ (in units of packets/sec). Under the burst rate constraint dictated by β, we can calculate the threshold and average packet delay for the threshold-based algorithms, and the average packet delay for the timer-based algorithms. As stated before, under these assumptions, the fixed threshold which minimizes the average packet delay for the *fixed-threshold* algorithm is given by $b = \lambda/\beta$. Recall that the average packet delay of *fixed-threshold* is given by $D_P = (b-1)/(2\lambda) = 1/(2\beta) - 1/(2\lambda)$. On the other hand, the average packet delay for the *timer-min* algorithm is $1/(2\beta)$ as we mentioned earlier. The term $1/(2\lambda)$ is the reduction in packet delays using a size-based algorithm that has a-priori information on λ.

Table 2.1: Packet Size Distribution from [1]

Size Range (bytes)	# Packets	Probability
32-64	2171017	0.2955
64-128	2519797	0.2621
128-256	574504	0.0598
256-512	297002	0.0309
512-1024	251686	0.0262
1024-2048	3800020	0.3953

In order to verify the results obtained above and to compare them against the algorithms we propose, we have designed a simulation scenario as given below:

- Packet arrival process is stationary Poisson with rate λ that is varied from 5000 to 50000.
- Desired burst rate β is set to 1000.
- Packet size distribution is taken from Table 2.1 which uses the traffic traces from [1]. To clarify, the first row of Table 2.1 suggests that 29.55 % of all the packets have lengths (in units of bytes) in the interval $[32, 64)$ and 2171017 such packets are observed. For convenience, in our simulations, we assume that with probability 0.2955, an incoming packet has a discrete uniform distribution in the interval $[32, 64)$, with probability 0.2621, it has a discrete uniform distribution in the interval $[64, 128)$, and so on. We believe that our synthetic method of generating packet lengths matches quite well with real traffic traces. Unless otherwise stated, this packet size distribution method will be used throughout the numerical examples used in this paper.
- Simulation length is 1000 seconds.
- Lower and upper burst length limits are not enforced.

Fig. 2.3 compares the average packet delay of the three algorithms *timer-min*, *fixed-threshold*, and *dyn-threshold-packet* as a function of the arrival rate λ. As

$\lambda \to \infty$, the average packet delay of *fixed-threshold* approaches to that of *timer-min* validating the closed-form expressions stated before. The average packet delay obtained by *dyn-threshold-packet* follows very closely the curve of *fixed-threshold* for all arrival rates. Note that *dyn-threshold-packet* does not assume an a-priori knowledge of the arrival rate λ as *fixed-threshold*. In Fig. 2.4, we also observe that *dyn-threshold-packet* achieves a burst rate which is very close to β validating the burst rate conformance of bucket-based algorithms.

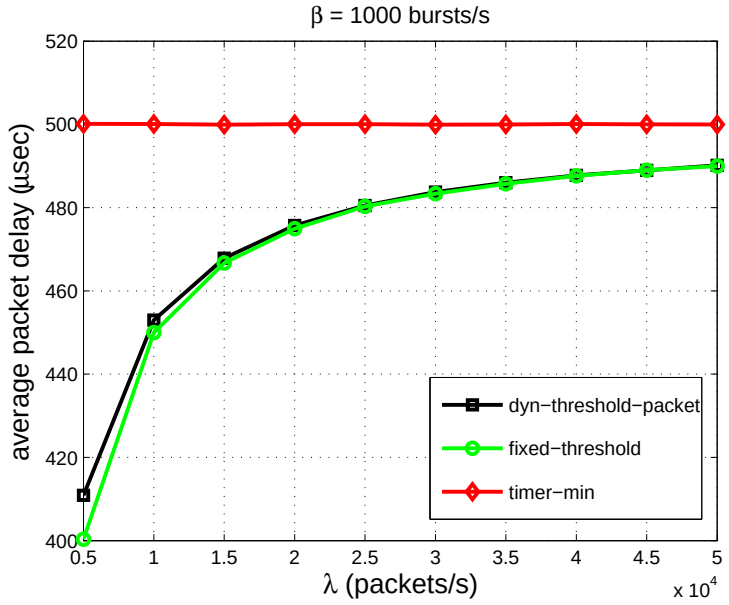

Figure 2.3: Average packet delay of the three assembly algorithms as a function of arrival rate λ

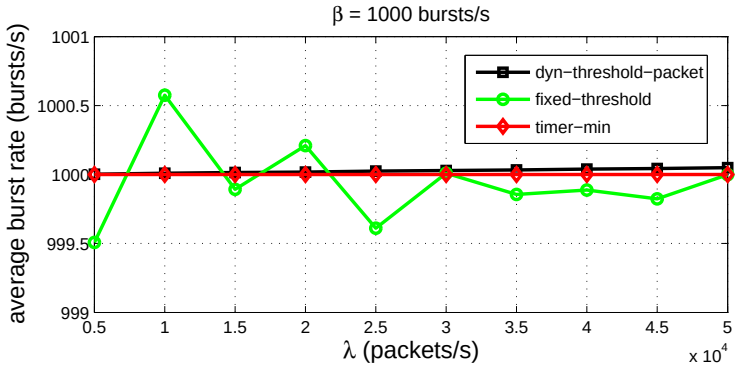

Figure 2.4: Average burst rate obtained using the three assembly algorithms as a function of arrival rate λ

We propose *dyn-threshold-byte* for the purpose of reducing average byte delays instead of packet delays. Average packet and byte delays (D_P and D_B) for

the two algorithms *dyn-threshold-packet* and *dyn-threshold-byte* as a function of arrival rate λ are given in Fig. 2.5 which shows that the algorithm *dyn-threshold-packet* generates identical byte and packet delays since this algorithm is not aware of packet lengths. On the other hand, the length-aware algorithm *dyn-threshold-byte* substantially reduces D_B. We are led to believe that one should use *dyn-threshold-byte* if the minimization of byte delays are sought.

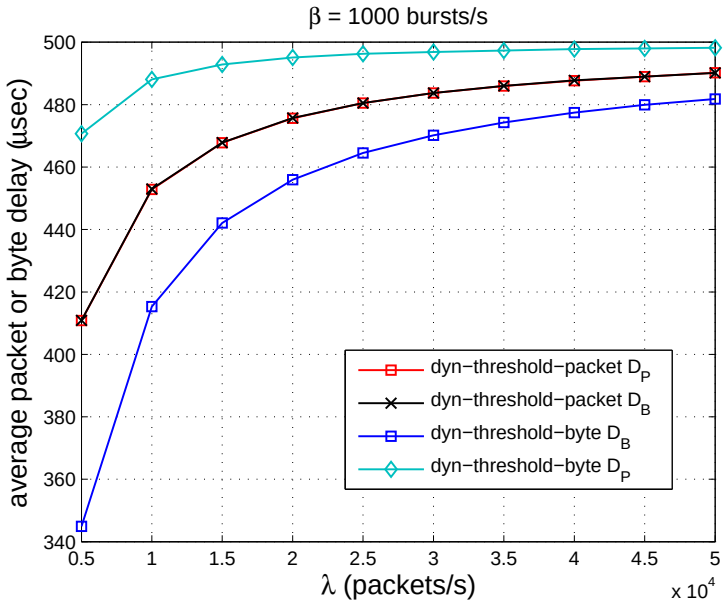

Figure 2.5: Average packet and byte delays (D_P and D_B) for the two algorithms *dyn-threshold-packet* and *dyn-threshold-byte* as a function of arrival rate λ

MMPP Traffic Scenario

We experiment a non-renewal inter-arrival scenario using synthetic traffic. For this purpose, we use a two-state MMPP to model client packet arrivals to the assembly buffer as shown in Fig 2.6. In this model, $\lambda_i, i = 1, 2$ denotes the arrival rate at state i. The average state holding time in state i is denoted by T_i. Therefore, the transition rate from state 1 to state 2 (from state 2 to state 1) in Fig. 2.6 is $1/T_1$ $(1/T_2)$. The average packet arrival rate is denoted by $\lambda = (\lambda_1 T_1 + \lambda_2 T_2)/(T_1 + T_2)$.

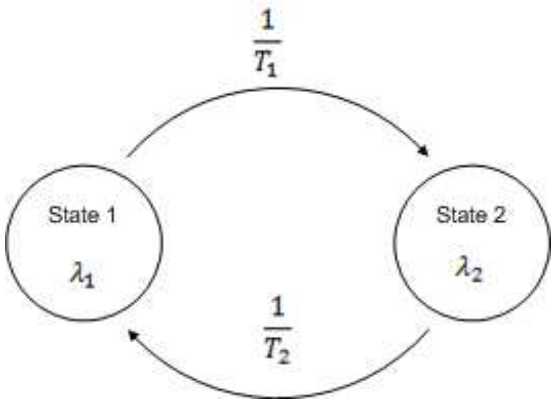

Figure 2.6: State diagram of the input traffic modeled with two-state MMPP

The *timer-min* algorithm produces $D_P = 1/(2\beta)$ irrespective of incoming packet traffic characteristics. The *fixed-threshold* algorithm assumes a-priori information on average arrival rate λ and generates bursts each time $b = \lambda/\beta$ packets are accumulated assuming integer b. The average packet delay for the *fixed-threshold* algorithm can then be written as:

$$D_P = \frac{\left(\frac{\lambda}{\beta} - 1\right)\frac{1}{2\lambda_1}\lambda_1 T_1 + \left(\frac{\lambda}{\beta} - 1\right)\frac{1}{2\lambda_2}\lambda_2 T_2}{\lambda_1 T_1 + \lambda_2 T_2} \tag{2.3}$$

Let us now use another scheme called *optimum* that is aware of the state which the MMPP is visiting. For the purposes of optimal performance, this scheme generates bursts in state 1 (in state 2) when $b_1 = \lambda_1/\beta$ ($b_2 = \lambda_2/\beta$) packets are accumulated. Here, we again assume b_1 and b_2 are integers. The burst rate of the *optimum* scheme is then equal to β irrespective of which state of MMPP is being visited. The average packet delay for the *optimum* scheme is easy to write:

$$D_P = \frac{\left(\frac{\lambda_1}{\beta} - 1\right)\frac{1}{2\lambda_1}\lambda_1 T_1 + \left(\frac{\lambda_2}{\beta} - 1\right)\frac{1}{2\lambda_2}\lambda_2 T_2}{\lambda_1 T_1 + \lambda_2 T_2} \tag{2.4}$$

It is not difficult to show that the two expressions in (2.3) and (2.4) lead to identical average packet delay D_P which can further be simplified to

$$D_P = \frac{1}{2\beta} - \frac{1}{2\lambda} \tag{2.5}$$

The second term above characterizes the reduction in average packet delay by using a size-based algorithm as opposed to a timer-based algorithm. Note that

this term is identical to that of the Poisson traffic scenario. We therefore conclude that the *fixed-threshold* algorithm provides optimum average packet delay but it suffers from fluctuations in the burst rate. When the actual traffic rate exceeds the mean rate, the burst rate of the *fixed-threshold* method exceeds the desired burst rate β. Similarly, when the actual rate is lower than the mean rate, burst rates are lower than β. On the other hand, the *optimum* scheme produces optimal D_P while maintaining the burst rate at β at all times. However, it is very hard to implement the *optimum* scheme since in this scheme, the traffic model should be entirely available to the burst assembly unit which should also accurately estimate the instantaneous state of the MMPP. In order to study how the proposed algorithms compare to these three algorithms, we experiment a scenario where $T_1 = \gamma t$ and $T_2 = (1-\gamma)t$ where $t = 10$ seconds, $0 < \gamma < 1$ and $\lambda_1 = 5000$ and $\lambda_2 = 50000$ packets/sec. The lower and upper burst length limits are not enforced in this experiment. We have tested the algorithms for three different values of $\gamma = 0.3, 0.5, 0.7$ for each algorithm. Let b_i^* and $\beta_i^*, i = 1, 2$ denote the average threshold value (in units of packets) and average burst generation rate (in units of bursts/sec) while at state i. We provide b_i^* and $\beta_i^*, i = 1, 2$ as well as the average packet delay D_P using the *fixed-threshold*, *optimum*, and *dyn-threshold-packet* algorithms as a function of γ in Table 2.2. Note that the *timer-min* algorithm average delay is fixed at 500 μs for all examples. In the *fixed-threshold* algorithm, the thresholds are fixed irrespective of the state of the MMPP and therefore the burst rates in each state deviate substantially from the desired burst rate although the long-term burst rate is kept approximately at β. The *optimum* scheme employs two separate burst assembly thresholds depending on the MMPP state and burst generation rate can therefore be set to β irrespective of the MMPP state. The average packet delays for these two algorithms are very close to each other as expected (see expression (2.5)). The proposed *dyn-threshold-packet* algorithm performs very close to the *optimum* method by

Table 2.2: The values b_i^* and $\beta_i^*, i = 1, 2$ and D_P using the *fixed-threshold*, *optimum*, and *dyn-threshold-packet* algorithms as a function of γ

Algorithm	γ	b_1^*	b_2^*	β_1^*	β_2^*	D_P (μs)
fixed-threshold	0.3	36.10	36.12	138.49	1384.04	486.26
	0.5	28.13	28.12	177.74	1777.88	482.29
	0.7	21.23	21.22	235.49	2356.44	476.71
optimum	0.3	5	50	1000.53	999.96	486.53
	0.5	5	50	999.95	999.86	481.70
	0.7	5	50	1000.12	999.69	472.67
dyn-threshold-packet	0.3	5.08	49.69	985.07	1006.20	486.94
	0.5	5.04	49.56	991.09	1008.89	482.87
	0.7	5.03	49.30	993.67	1014.27	475.46

adjusting properly the assembly thresholds at each state so that the burst generation rate settles at β and its delay performance is very close to the size-based algorithms. Despite the difficulty in implementing the *optimum* method, our proposed method is model-free and is very easy to implement.

For *dyn-threshold-byte* algorithm, in order to see the effects of the choice of the learning parameter κ, we plotted the dynamic thresholds as a function of time for various values of κ when γ is set to 0.5 in the previous example. As we see in Fig. 2.7, for $\kappa = 10$, the dynamic threshold changes slowly despite the abrupt change in the traffic and the algorithm comes short of tracking the thresholds of the *optimum* scheme. For $\kappa = 10000$, on the other hand, change in traffic is captured but at the expense of large-scale fluctuations in the dynamic threshold. We also provide Table 2.3 which presents the quantities b_i^*, $\beta_i^*, i = 1, 2$ and D_B using the *dyn-threshold-byte* algorithm as a function of κ. It is clear that large-scale fluctuations in the dynamic threshold result in increases in the average byte delay D_B. We conclude that the choice of the learning parameter $\kappa = 1000$ is a reasonable choice since in this case κ is large enough to track rapid changes in traffic and κ is small enough to make sure that fluctuations in the dynamic thresold are reasonably small. We set κ to 1000 in the remaining numerical studies of the current article.

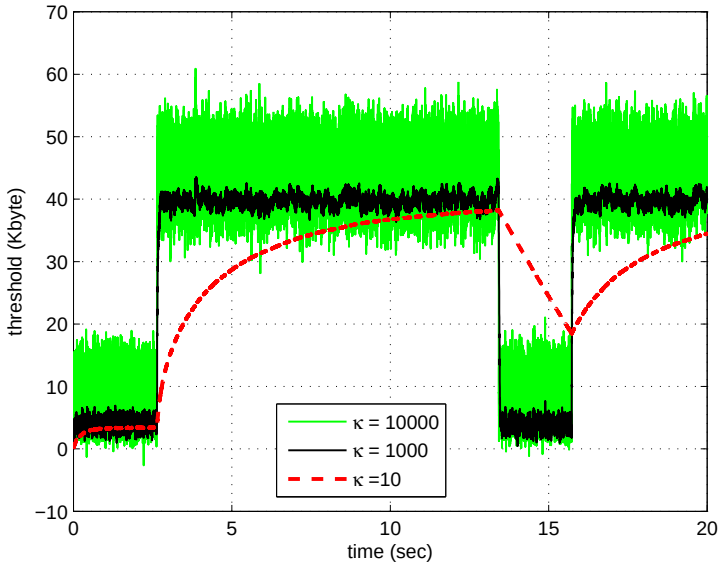

Figure 2.7: A twenty-second snapshot of the dynamic thresholds of the *dyn-threshold-byte* algorithm with respect to time for different values of κ

Table 2.3: The values b_i^* and $\beta_i^*, i = 1, 2$ and D_B using the *dyn-threshold-byte* algorithm as a function of κ

κ	β_1^*	β_2^*	β	D_B (μs)
1	224.01	1733.19	1002.88	466.72
10	611.88	1364.66	1000.19	467.86
100	935.17	1063.37	1000.00	467.76
1000	992.61	1007.24	1000.00	469.55
10000	999.25	1000.73	1000.00	478.09
30000	999.73	1000.26	1000.00	484.41

2.4.2 Assembled Burst Statistics

In order to investigate the statistical characteristics of output burst traffic, first we have simulated each algorithm with a stationary Poisson traffic with rate $\lambda = 30000$ packets/second. Then, we have designed a simulation scenario with two-state MMPP having the following parameters;

- Packet arrival process is two-state MMPP.
- $T_1 = \alpha t$ and $T_2 = (1 - \alpha)t$ where $t = 10$ seconds, $\alpha = 0.5$
- $\lambda_1 = 5000$ packets/second, $\lambda_2 = 50000$ packets/second.

- Average burst rate (β) is 1000 bursts/second.
- Packet size distribution is taken from Table 2.1.
- Simulation length is 1000 seconds.
- The algorithms are not bounded with B_{max} and B_{min}.

Short Term Statistics

As pointed out in [10] and [18], for the fixed threshold based algorithms, distribution of the inter-burst time converges to Gaussian distribution. Similarly, in fixed period timer based algorithms, distribution of the burst length converges to Gaussian distribution as well. Small variance with Gaussian distribution in short term is acceptable both for inter-burst time and burst length in a burst assembly queue. Since the proposed algorithm is adaptive and it dynamically changes its threshold, both inter-burst time and burst length is variable in our case. In Fig. 2.8 and Fig. 2.9, we see that the distribution of the inter-burst time converges to Gaussian distribution for stationary Poisson and two-state mmpp traffic. On the other hand, since the threshold of the proposed method dynamically changes with the time, the distribution of the burst length is mostly determined by the shape of the input traffic. We have also calculated the Squared Coefficient of Variations (SCV) for all algorithms in order to see variances of the burst length and inter-burst times. In Table 2.4, we see that SCV of the burst length for *dyn-threshold-packet* is very close to that of *optimum* in both traffic scenarios. Table 2.5 shows the SCV values for the inter-burst times for all algorithms. Here again we see that SCV for the inter-burst times has similar characteristics in both *dyn-threshold-packet* and *optimum* scheme.

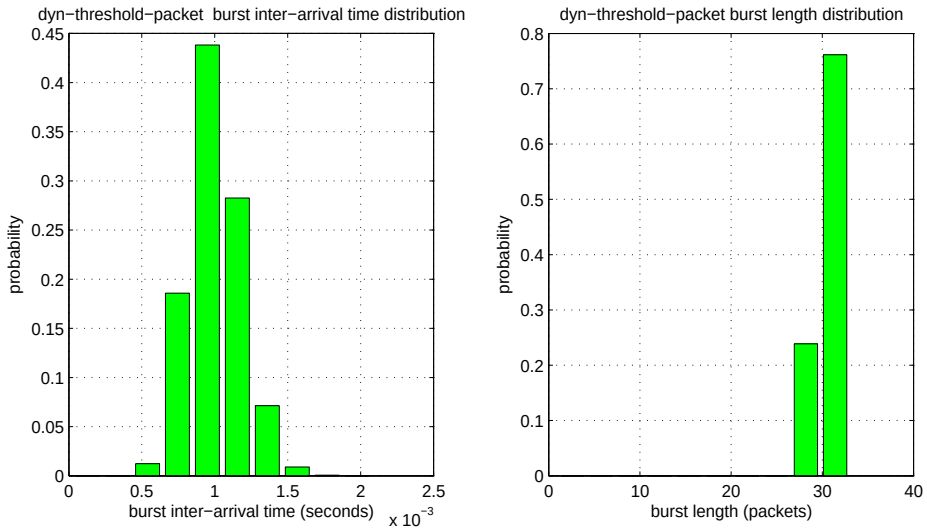

Figure 2.8: Inter-Burst Time and Burst Length Distribution in Stationary Poisson Traffic

Table 2.4: SCV Test for Burst Length

algorithm	SCV (two-state MMPP)	SCV (stationary Poisson)
dyn-threshold-packet	0.6758	0.0007
optimum	0.6028	0.0000
timer-min	0.8144	0.0333
fixed-threshold	0.2616	0.000

Long Term Statistics

As pointed out and mathematically proved in [16], the long range dependence in the assembled burst traffic does not affect the loss performance of the bufferless core network. Thus, the degree of the self-similarity of the assembled traffic has no effect and could be ignored. To see the effect of the proposed assembler algorithm, We have used the an input traffic having hurst parameter of 0.87. After estimating the hurst parameter of the assembled burst traffic, we have observed that it remains the same as the original value.

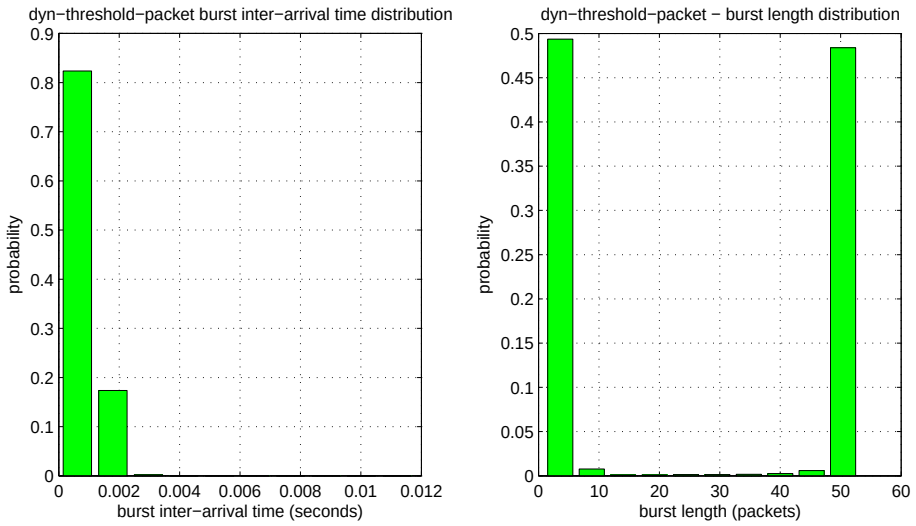

Figure 2.9: Inter-Burst Time and Burst Length Distribution in Two-state MMPP Traffic

Table 2.5: SCV Test for Inter-Burst Time

algorithm	SCV (two-state MMPP)	SCV (stationary Poisson)
dyn-threshold-packet	0.1104	0.0318
optimum	0.1043	0.0333
timer-min	0.0000	0.0000
fixed-threshold	2.0847	0.0333

2.4.3 Real Traffic Traces

In the previous scenarios driven with synthetic traffic, we have shown the basic properties of various burst assembly methods. However, it is also crucial to study the delay performance of the proposed algorithms in case of more realistic traffic scenarios. In this numerical experiment, we focused on only byte delays and not packet delays. For this purpose, we use two different traces taken from a traffic data repository maintained by the MAWI (Measurement and Analysis on the WIDE Internet) Working Group of the WIDE Project [1]. We also scale down the inter-arrival times in these traces to generate varying incoming bit rates. While the first trace has a low standard deviation (STD), the latter is quite bursty. For each traffic trace, we use three different values of $\beta = 1000, 2000, 3000$. The lower and upper burst length limits have been enforced in this experiment, i.e., $B_{min} = 1$ Kbytes and $B_{max} = 70$ Kbytes. We have studied the performance of

the *dyn-threshold-byte* algorithm against the *timer-min* and the *timer-min-max* algorithms. The learning parameter κ is set to 1000 for *dyn-threshold-byte* and T_{max} is set to ∞. We have not tested the *fixed-threshold* algorithm in this scenario due to its highly variable burst rates that may not be desirable.

The first trace was obtained from the WIDE backbone at Sample Point B on Jan 1, 2006 at 14:00:00 for a trans-Pacific line with 100 Mbps link speed [1]. The original trace has a duration of 899.76 seconds, mean rate = 22.33 Mbps, and STD = 1.53M. Feeding the trace to the burst assembly unit with varying bit rates (by scaling down the inter-arrival times), we have simulated the performance of various burst assembly algorithms. The average byte delays for the three algorithms are given in figures 2.10a-c for three different values of β. Fig. 2.10d gives a minute-long snapshot of the incoming bit rate (scaled 14 times) as a function of time. The trace is pretty smooth similar to a Poisson traffic stream and therefore *timer-min* and *timer-min-max* performed very similarly since the probability that the accumulated number of bytes within a timer expiration period exceeding B_{max} was negligibly small for this smooth traffic. The results clearly show that the proposed *dyn-threshold-byte* significantly reduces the average byte delay compared to timer-based algorithms especially for lower bit rates. The percentage gain in using our proposed algorithm also increases with β.

We then study the second trace which was obtained again from the WIDE backbone at Sample Point F on Sat Jan 5, 2008 at 14:00:00 for a trans-Pacific line with 150 Mbps link speed [1]. The original trace has a duration of 900.29 seconds, mean rate = 61.56Mbps, and STD = 11.67M. The average byte delays for the three algorithms are given in figures 2.11a-c for three different values of β. Fig. 2.11d gives a two minute-long snapshot of the incoming bit rate (scaled 7 times) as a function of time. The trace is not as smooth as the previous one and is quite bursty. Therefore, when enforcing the upper burst length limit, there were quite a few occasions at which the accumulated number of bytes within

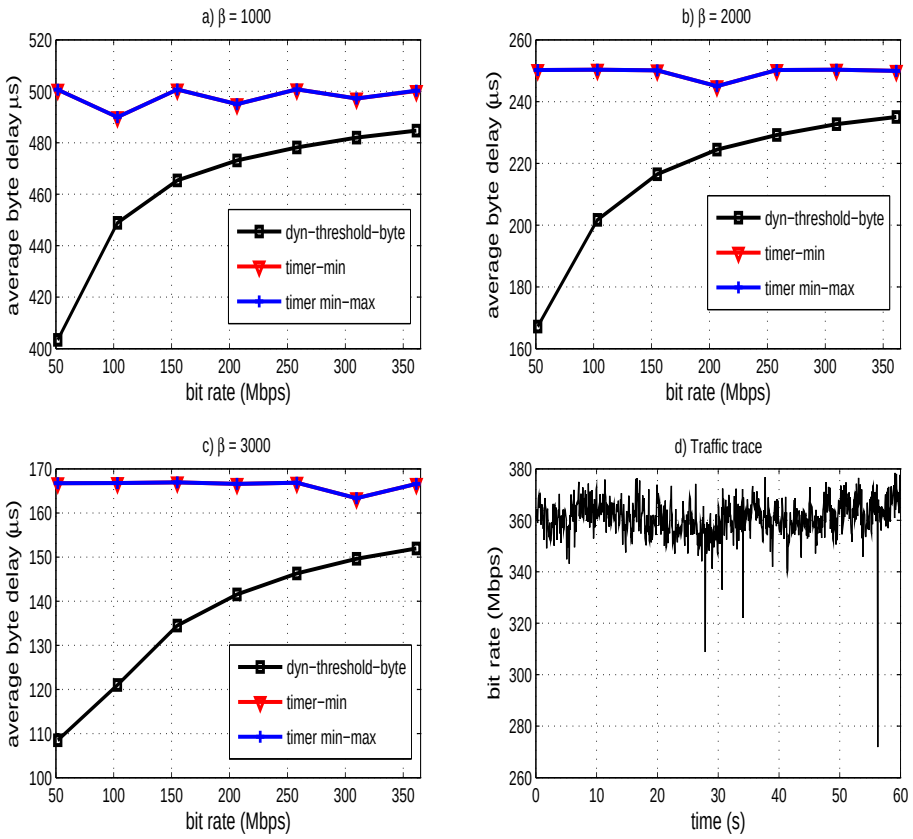

Figure 2.10: Average byte delay for the cases a) $\beta = 1000$ b) $\beta = 2000$ c) $\beta = 3000$ using various algorithms for the trace from Sample Point B (2006) whose one-minute snapshot is given in d)

a timer expiration period exceeded B_{max} and some packets had to wait for the next timer expiration epoch when using *timer-min-max*. In this case, the *timer-min-max* performed very poorly compared to the *timer-min* algorithm for which there was no enforcement of B_{max}. As expected, this situation is more evident for relatively lower β. The proposed *dyn-threshold-byte* is shown to significantly reduce the average byte delay D_B compared to both timer-based algorithms especially for lower bit rates and higher β. We also note that *dyn-threshold-byte* not only reduces D_B but also properly enforces the lower and upper burst length limits.

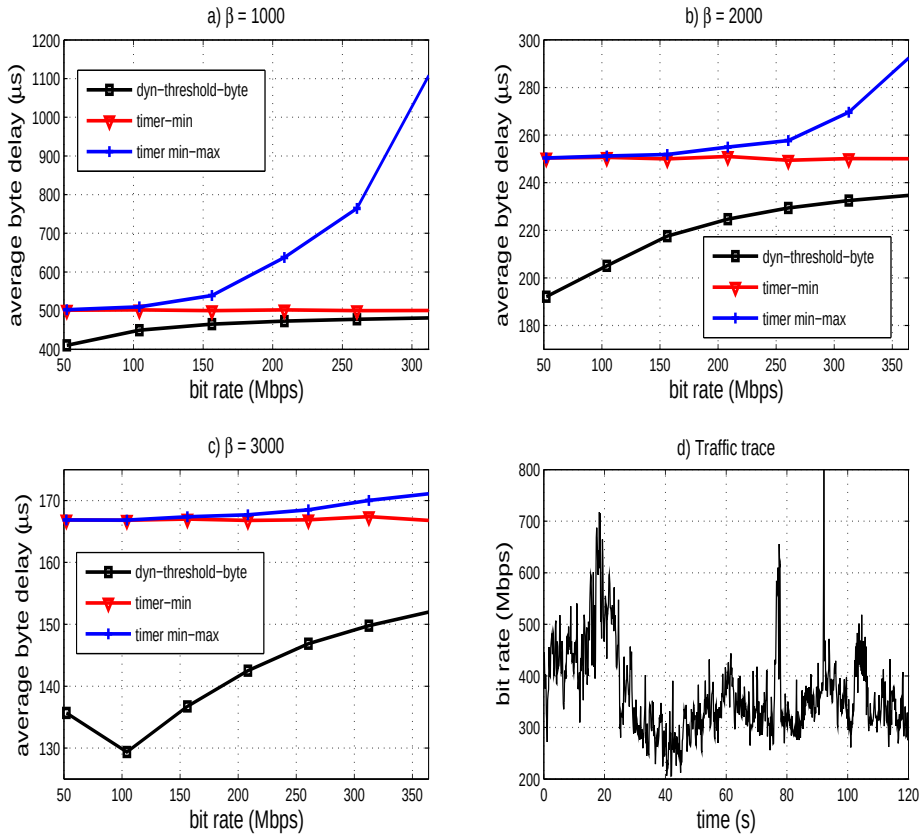

Figure 2.11: Average byte delay for the cases a) $\beta = 1000$ b) $\beta = 2000$ c) $\beta = 3000$ using various algorithms for the trace from Sample Point F (2008) whose two-minute snapshot is given in d)

2.4.4 Loss Performance

In the previous numerical studies, we have shown the reductions in average packet or byte delays in the burst assembly buffer using the proposed dynamic-threshold algorithms while enforcing lower and upper burst length limits. However, it is also vital to address the traffic statistics of the bursts fed into the OBS network and their impact on burst loss performance in the OBS network. Recall that the *timer-min* or *timer-min-max* algorithms produce deterministic burst inter-arrival times with variable burst lengths whereas the *fixed-threshold* algorithm generates bursts that have fixed number of packets in them but variable inter-burst times. The proposed algorithms in this article produce both variable inter-burst times and burst lengths. In this section, we address the question of whether such

modified traffic characteristics have any impact on loss performance in the OBS network. In order to study the loss performance of the proposed and existing algorithms in an OBS network, we have chosen the topology given in Fig. 2.12 in which n access networks feed IP packets into a burst assembly buffer located at an OBS edge router which is connected to OBS core router using four wavelengths for data (bandwidth of each wavelength is set to 10 Gbps) and one wavelength for control. Packet arrivals from each access network is assumed to be Pareto on-off [23] with Hurst parameter $H = 0.8$, on-time $t_{on} = 5\ 10^{-8}$, off-time $t_{off} = 5\ 10^{-9}$ seconds with mean bit rate set to 0.8 Gbps. Packet size distribution is based on Table 2.1. We set $B_{min} = 10$ Kbytes and $B_{max} = 70$ Kbytes. The size of the burst header is assumed to be 125 bytes, the offset time is set to 40μs and simulation run-time is set to 20 seconds. When a burst assembly decision is to be made by the burst assembly unit and if all the wavelength channels are occupied after the offset time, this particular burst is assumed to be lost. We are interested in the probability of loss using various burst assembly methods. In Fig. 2.12, we increase the number of access networks (denoted by n) from 42 to 46 and we have set β to $3000n$. Under these conditions, we have compared the loss rates of various burst assemblers. Although the measured average burst size is about 35 Kbytes for each assembly algorithm, we have observed that the *dyn-threshold-byte* algorithm significantly reduces the probability of loss in the bufferless core network as we see in Fig. 2.13. From this example, we conclude that the proposed algorithms not only reduce average packet or byte delays but the traffic they generate do not appear to have any adverse impact on the loss performance in the OBS network.

To see the effects of multiple assemblers, we have chosen another topology in Fig. 2.14 having n edge nodes and assemblers. In this scenario, we have varied the number of sources from 20 to 40. Simulation results show that each assembler method has similar loss performance. See Fig. 2.15. This is because sufficiently

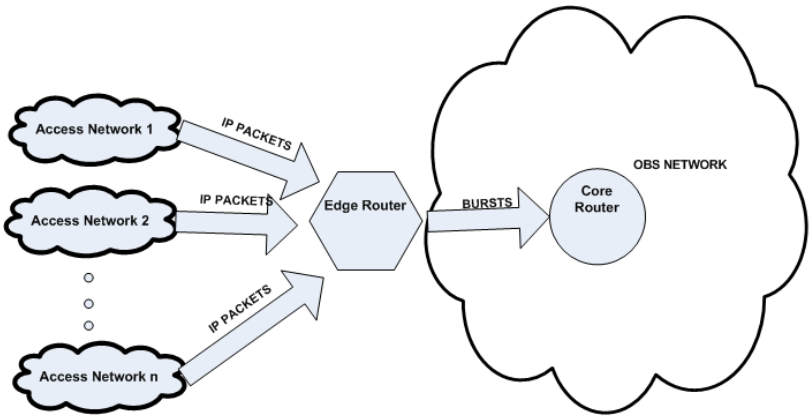

Figure 2.12: Burst assembly scenario to study the probability of loss

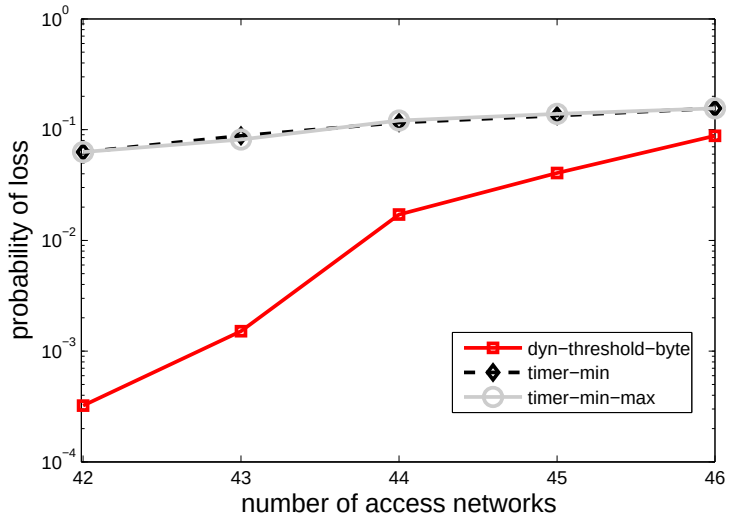

Figure 2.13: Probability of loss as a function of the number of access network n

large number of sources having the same statistical characteristics produce an output which converges to Gaussian distribution by central limit theorem.

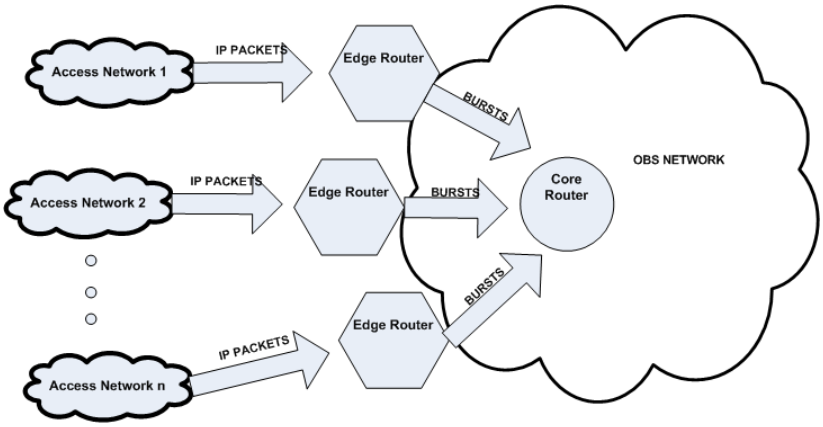

Figure 2.14: Topology 2

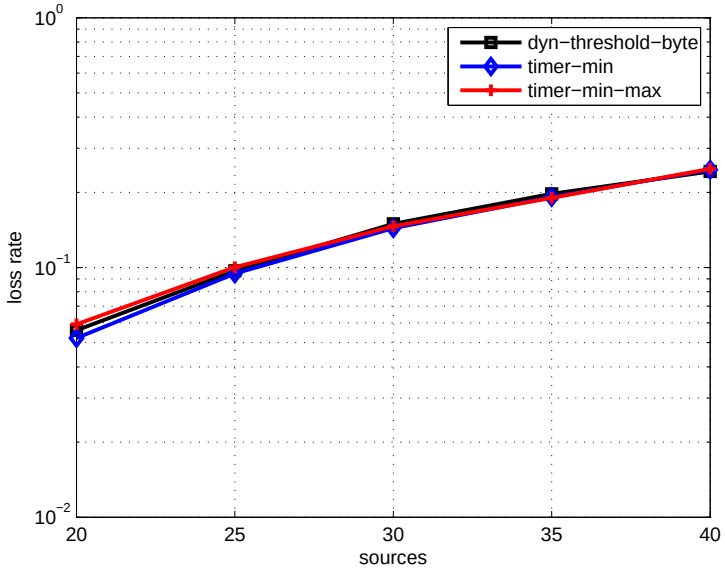

Figure 2.15: Loss Ratio

Chapter 3

ADAPTIVE HYSTERESIS for DYNAMIC BANDWIDTH RESERVATION

3.1 Motivation and Related Work

Some basic techniques exist in the literature to perform the reservation of network resources to a virtual path or tunnel. Consider a scenario in which end-to-end reservation requests initiated by Public Switched Telephone Network (PSTN) voice calls arrive at a virtual path to be destined to a particular voice over packet gateway (Fig. 1.1). One method for reservation is that whenever a bandwidth need for a call is requested or an existing call is terminated, the bandwidth of the VP is adjusted simultaneously which provides optimal usage of the available bandwidth by tracking the actual call traffic. This method is called Switched Virtual Circuit (SVC). On the other hand, the main drawback of this approach is too much signalling and message processing burden on the system. Another simple technique is Permanent Virtual Path (PVP) approach. According to this

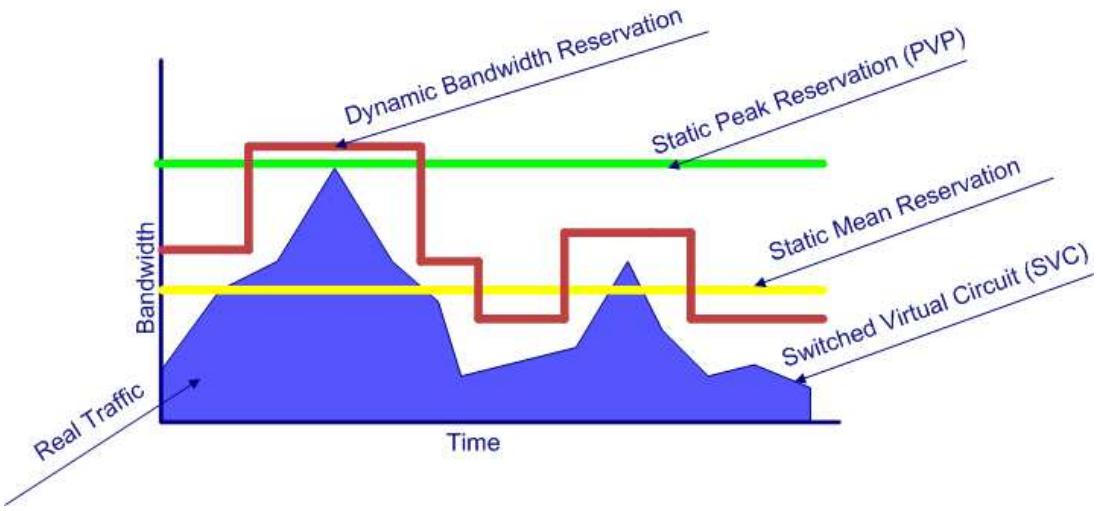

Figure 3.1: Bandwidth Reservation Mechanisms

approach the reservation is done based on largest bandwidth demand over a long time demand (24 hours). However, in this case the network bandwidth will be under utilized. By eliminating those problems, up to now several Dynamic Bandwidth Allocation (DBR) or Reservation mechanisms have been proposed to solve the intelligent bandwidth allocation problem of a VP. Fig. 3.1 shows several reservation mechanisms.

A state dependent dynamic bandwidth control algorithm has been proposed in [24] for the virtual paths of an ATM network. According to this approach, upon arrival of a call if there is insufficient bandwidth in the current virtual path, the bandwidth of that virtual path is increased by a fixed step S. With the same step S, depending on the virtual path utilization condition, a bandwidth decrement is carried out. One main contribution of this approach is that with small sized S and large number of VPs, the transmission efficiency of the current link is high in terms of processing overhead and bandwidth wastage. On the other hand, oscillations around a threshold may increase signalling burden and also in high traffic conditions, a large amount of bandwidth waste occurs due to fixed size S.Another virtual path allocation policy has been proposed in [25] which eliminates the potential problems of [24] by applying two thresholds, namely upper and lower ones. By these thresholds, it introduces the concept hysteresis,

by which it reduces the possibility of oscillations. On the other hand, since the computation of the thresholds require construction of an auxiliary Markov chain with known arrival rates, it is a model-based policy. Reference [26] proposes a periodic capacity management policy which assigns virtual path capacities according to information of the offered traffic intensity and link occupancy based on capacity assignment tables in order to reduce on-line operations and achieve a desired call admission rate. In [27], a simple operational rule has been proposed to assign capacities to virtual paths based on processing and bandwidth utilization constraints. At link level, an optimal solution is obtained. A similar problem has been considered in [28]. This approach uses an ARIMA model to forecast the traffic and does synchronous bandwidth reservations. However, the forecast is done at packet level, and it does not consider the sessions and flows in application level. A layered bandwidth allocation scheme has been proposed in [29] using a cost factor which consists of the linear combination of the reserved bandwidth on each link. Reference [30] uses a Discrete Kalman Filter to estimate number of flows in the aggregate traffic in the first step. In the second step, a reservation is carried out based on deriving the transient probabilities of the possible system states. Similar to [30], [31] proposes an approximate Kalman-Bucy Filter to predict the number of active connections for an LSP. Based on this estimation, solving some optimization problems on-line, the best reservation of bandwidth and the time interval over which this reserved bandwidth holds are calculated. A commercially available synchronous approach for dynamic bandwidth allocation is Auto-Bandwidth allocator by [32]. It automatically adjusts the bandwidth of an MPLS tunnel based on the local maximum approach. This allocator monitors the bandwidth periodically with X minutes (default X = 5 min) and keeping track of the maximum bandwidth over an interval Y hours (default Y = 24 hours), it re-adjusts the tunnel bandwidth for next Y interval based on the tracked maximum bandwidth. One main drawback of this approach is when the traffic load is higher or lower than the allocated bandwidth, a waste of bandwidth

and losses may occur. In [33], an adaptive bandwidth allocation technique has been proposed for wide area networks (WANs) based on static and dynamic traffic matrices which are calculated by using busy hours and time zones of the border routers of a WAN. A distributed approach has been proposed in [34], which suggests the benefits of the dynamic traffic engineering for dynamic bandwidth reservation. Basically, using dynamic resizing mechanism, the route of each LSP is optimized periodically in a decentralized manner, which results in better network utilization. Recently, a trend based bandwidth provisioning mechanism has been suggested in [35]. It basically uses a slope estimator and memory moderator unit to estimate the traffic trend to be used to adjust an LSP bandwidth while reducing the signaling overhead.

In our proposal, we assume that every call has an identical bandwidth for the voice traffic. The proposed method is model-free and does not require any traffic model. However, to be able to compare the performance of the proposed algorithm with those in the existing literature, we assume that individual calls arrive at the connection oriented network according to a non-stationary Poisson process with rate $\lambda(t)$ and call holding times are exponentially distributed with mean $1/\mu$. Basically, we have two versions of our proposed algorithm, namely Adaptive Hysteresis for Single-Class (Single-Virtual Path) Case and Adaptive Hysteresis for Multi-Class (Multiple-Virtual Path) Case. In the first version, VP capacity allocation is performed locally with considering the maximum bandwidth, C_m, without knowing the allocated bandwidths of the other VPs in the current physical link. On the other hand, in the Multi-Class version, the dynamic bandwidth reservation is done locally for every VP in the link with knowing the bandwidths of the other VPs. For each version we introduce a desired update rate parameter, β (updates per hour), which is a tradeoff between message processing and bandwidth efficiency costs. Our goal is then to allocate and minimize the reserved bandwidth dynamically subject to that the reserved bandwidth is larger than the actual traffic bandwidth and the actual average update rate is less

than β. We also propose the same method for Internet data traffic. In this case, each flow in a VP has variable size capacities instead of identical call capacities and also flow lengths may have different statistical characteristics than those of voice calls. In this scenario, the events which make the proposed algorithm work are defined as periodically monitored bandwidth values instead of call arrivals or departures in the previous scenario.

The rest of this chapter is organized as follows. In Section 3.2, we present the details of an existing synchronous approach. Section 3.3 presents a model-based asynchronous approach which gives the optimal solution of the problem by using Relative Value Iteration (RVI) algorithm. In Section 3.4, we demonstrate two versions of the proposed method. Finally, Section 3.5 concludes this chapter by giving the performance evaluation of the proposed and existing techniques.

3.2 Synchronous Dynamic Bandwidth Reservation

In this approach, the bandwidth update is performed periodically with period T. At each decision epoch, kT where $k = 0, 1, 2, ...$, the mechanism chooses the minimum bandwidth allocation, R_k, based on the number of calls, N_k, and the utilization factor, $\rho_k = \lambda_k/\mu$ where λ_k is the estimate of the call arrival rate and μ is the service rate, for which the average blocking probability, $P(\rho_k, R, N_k, T)$, in the current time interval, $[kT, (k + 1)T)$, stays below the desired blocking probability, P_b. Here the average blocking probability is in an interval of T is calculated by the equation $P(\rho_k, R, N_k, T) = 1/T \int_0^T P_{R|N_k}(t)dt$ where $P_{R|N_k}(t)$ is the probability of finding the system in state R at time t, which can be calculated by numerical transient solutions of continuous-time Markov chains as demonstrated in [36]. In cases when R_k is larger than the physical link capacity, C_m, R_k is set to C_m. Finally, one can create lookup tables off-line with indices

of R_k and N_k. On the other hand, since the blocking probabilities depend on the arrival and departure processes, this approach is model-based and has some drawbacks:

- If the traffic is non-stationary, large lookup tables have to be performed to estimate the traffic parameters.
- Solving large systems could be cumbersome.
- Periodic decision epochs may not be the most effective strategy compared to the asynchronous approaches.

3.3 Model-Based Optimal Solution

In the previous approach, since the decisions are made only at fixed epochs, the problem is formulated as the subject of discrete-time Markov decision model. However, as we declared previously the most effective strategy could be an asynchronous approach. The dynamic bandwidth reservation problem with random decision epochs could be solved by a semi-Markov decision model [37]. The problem satisfies the following Markovian properties: the time until the next decision epoch depends only on the present state, thus the decision made is independent of the past history of the system. Also, the cost incurred until the next decision epoch depends on the present state and the action chosen at that state. Reference [37] proposes a data-transformation method by which a semi-Markov decision model can be converted to a discrete-time Markov decision model in order to reduce the calculation costs. This transformation technique provides us to use the recursive Relative Value Iteration (RVI) algorithm for the semi-Markov decision model. The model and the parameters are given as follows:

- The set of possible states is denoted by I .

- For each state $i \in I$, a set $A(i)$ of possible actions is available.
- It is assumed that I and $A(i)$ are finite.
- $P_{ij}(a)$ = the probability that at the next decision epoch the system will be in state j if action a is chosen in the present state i
- $\tau_i(a)$ = the expected time until the next decision epoch if action a is chosen in the present state i
- $c_i(a)$ = the expected costs incurred until the next decision epoch if action a is chosen in the present state i
- It is assumed that $\tau_i(a) > 0$ for all $i \in I$ and $a \in A(i)$

3.3.1 The Data-transformation Method

As pointed out in [37], a semi-Markov decision model can be converted to a discrete-time Markov decision equivalent by assuming the following transformations:

- $\overline{I} = \mathrm{I}$
- $\overline{A}(i) = A(i)$, $i \in \overline{I}$
- $\overline{c}_i(a) = c_i(a)/\tau_i(a)$, $i \in \overline{I}$ and $a \in \overline{A}(i)$
- $\overline{P}_{ij} = \begin{cases} (\tau/\tau_i(a))P_{ij}(a) & \text{if } j \neq i,\ i \in \overline{I} \text{ and } a \in \overline{A}(i) \\ (\tau/\tau_i(a))P_{ij}(a) & +[1-(\tau/\tau_i(a))] \text{ if } j = i,\ i \in \overline{I} \text{ and } a \in \overline{A}(i) \end{cases}$

3.3.2 Relative Value Iteration Algorithm

Since the discrete-time Markov decision model has the same class of stationary policies as the original semi-Markov decision model, we can say that a value-iteration algorithm for the original semi-Markov decision model is implied by

the value iteration algorithm for the transformed discrete-time Markov decision model. The recursive method for the semi-Markov decision model given as follows:

- Step 0: Choose $V_0(i)$ such that $0 \leq V_0(i) \leq min_a\{c_i(a)/\tau_i(a)\}$ for all i. Choose a number τ with $0 < \tau \leq min_{i,a}\tau_i(a)$. Let $n := 1$

- Step 1: Compute the function $V_n(i)$, $i \in I$ from

$$V_n(i) = \min_{\mathrm{a\in A(i)}} \left[\frac{c_i(a)}{\tau_i(a)} + \frac{\tau}{\tau_i(a)} \sum_{\mathrm{j\in I}} P_{ij}(a)V_{n-1}(j) + (1 - \frac{\tau}{\tau_i(a)})V_{n-1}(i) \right].$$

  Let $R(n)$ be a stationary policy whose actions minimize the right-hand side of the equation above.

- Step 2: Compute the bounds

$$m_n = \min_{j\in I}\{V_n(j) - V_{n-1}(j)\}, \qquad M_n = \max_{j\in I}\{V_n(j) - V_{n-1}(j)\}$$

  The algorithm is stopped with policy R(n) when $0 \leq (M_n - m_n) \leq \varepsilon m_n$, where ε is a pre-specified accuracy number. Otherwise, go to step 3.

- Step 3: $n := n + 1$ and go to step 1.

For the choice of τ, if the Markov chains of the semi-Markov decision model are aperiodic, it is reasonable to take $\tau = min_{i,a}\tau_i(a)$; otherwise $\tau = \frac{1}{2}min_{i,a}\tau_i(a)$.

3.3.3 Formulation with the Dynamic Bandwidth Allocation Problem

In this model, we assume that the connection oriented network has C_m identical channels and individual calls arrive at this connection oriented network according to stationary Poisson process with rate λ. The service time of each individual call is exponentially distributed with mean $1/\mu$. We also assume that the call

arrival rate, λ is less than the maximum service rate, μC_m. A channel can handle only one call request at any time. When the channels turned on from a to b, a non-negative switching cost is incurred with the function, $K|a-b|$. There are also an operating cost, r, for the channels being turned on and a holding cost, h, for the calls in progress. Under these assumptions, we re-define the system as follows:

- The state of the system is described by the pair (N, R) at any time instant, where N is the number of calls in progress and R is the number of channels allocated.
- Whenever a bandwidth need for a call is requested or an existing call is terminated, a decision is made.
- Normally this model has infinite states, but we truncate the model to maximum C_m channels since the system has finite capacity.
- State space can be described as $I = \{(N, R)|0 \leq N \leq C_m, 0 \leq R \leq C_m\}$.
- Action space can be described as

$$A(N,R) = \begin{cases} \{R'|R' = 0, ..., C_m\}, & 0 \leq N \leq C_m - 1, 0 \leq R \leq C_m \\ \{C_m\}, & N = C_m, 0 \leq R \leq C_m \end{cases}$$

  Here action R' in state (N, R) means that the reservation is adjusted from R to R' at any decision epoch.

- The time until the next decision epoch can be specified as;

$$\tau_{(N,R)}(R') = \frac{1}{\lambda + \min(N, R')\mu}, \quad 0 \leq N \leq C_m - 1, 0 \leq R' \leq C_m$$

- The immediate cost is;

$$c_{(N,R)}(R') = K|R - R'| + \frac{hN + rR'}{\lambda + \min(N, R')\mu}, \quad 0 \leq N \leq C_m - 1, 0 \leq R' \leq C_m$$

- Step 1 in the RVI formula becomes;

$$V_n((N,R)) = \min_{0\leq R'\leq C_m}\{\lambda + \min(N,R')\mu\}K|R-R'|$$
$$+hN + rR' + \frac{\lambda}{\lambda + C_m\mu}V_{n-1}((N+1,R'))$$
$$+\frac{\min(N,R')\mu}{\lambda + C_m\mu}V_{n-1}((N-1,R')) + \{1 - \frac{\lambda + \min(N,R'}{\lambda + C_m\mu}\}V_{n-1}((N,R'))\}$$

for the states (N,R) with $0 \leq N \leq C_m - 1$, $0 \leq R \leq C_m$. And for states (C_m,R);

$$V_n((C_m,R)) = \frac{1}{C_m\mu}(\lambda + C_m\mu)(C_m\mu - \lambda)K|R - C_m|$$
$$+\frac{h\lambda}{C_m\mu - \lambda} + hC_m + rCm + \frac{C_m\mu - \lambda}{\lambda + C_m\mu}V_{n-1}((C_m - 1, C_m))$$
$$+\frac{\lambda(C_m\mu - \lambda)}{C_m\mu(\lambda + C_m\mu)}V_{n-1}((C_m,C_m)) + 1 - \frac{C_m\mu - \lambda}{C_m\mu}V_{n-1}((C_m,R))$$

3.4 Adaptive Hysteresis for DBR

Fig. 3.2 shows a static hysteresis-based binary control system which has two actions, namely 0 and 1, a controlled variable x, a threshold parameter T_x on the controlled variable, and a hysteresis band parameter d. It is clear from the figure that, when x drops below $T_x - d$, action 1 is taken and when it exceeds $T_x + d$, in this case action 0 is taken. Otherwise no action is taken. For the DBR problem, we propose an adaptive hysteresis algorithm with hysteresis-based binary control in which the threshold and band parameters vary within the time. This is performed by a leaky bucket mechanism. The details are given in the following section.

3.4.1 Algorithm for Single-Class Case

In this version, we assume that we have a single virtual path with the maximum allocated capacity C_m and the DBR process is done locally without knowing the

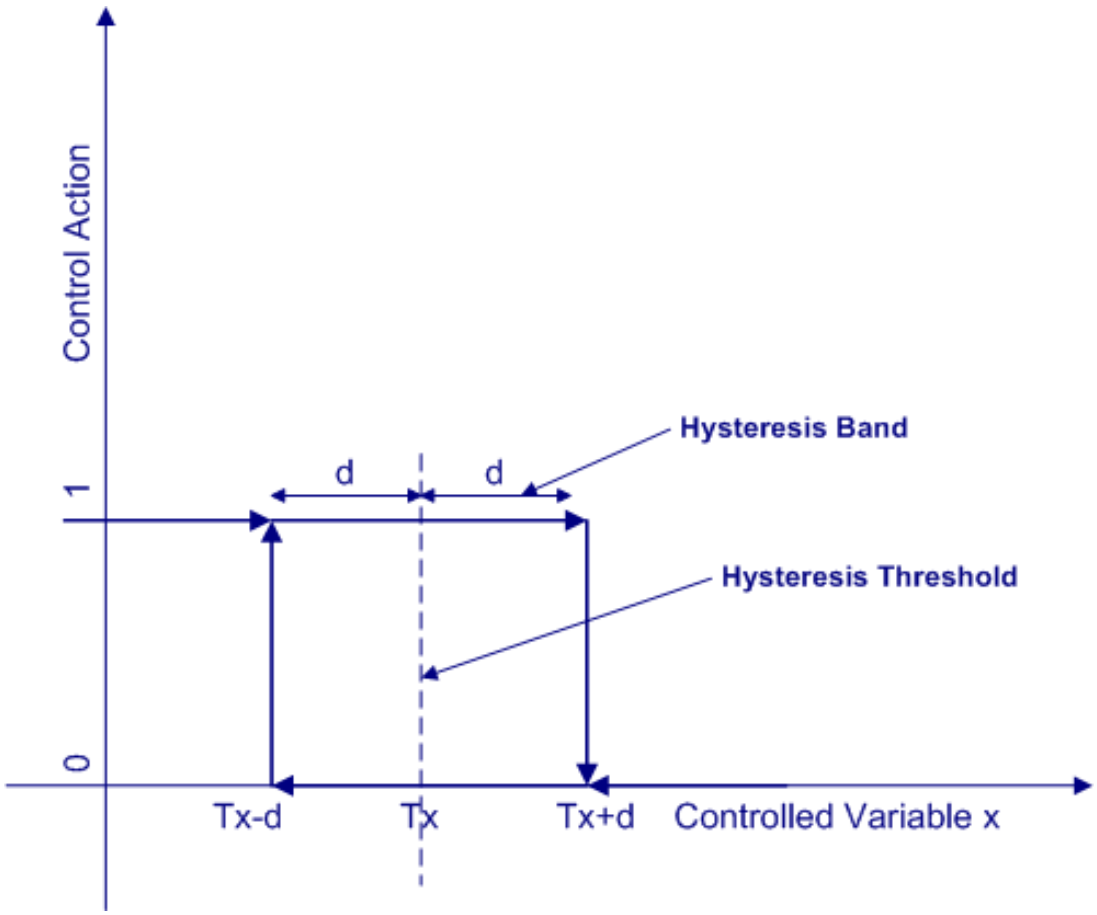

Figure 3.2: A binary control system using static hysteresis

bandwidth of the other VPs in the link. Algorithm 5 shows the details of the proposed method.

3.4.2 Algorithm for Multi-Class Case

In this version, the dynamic bandwidth reservation is done locally for every VP in a physical link with knowing the allocated bandwidths of the other VPs. Algorithm 6 shows the details of the proposed method.

Algorithm 5 Adaptive Hysteresis for Single-Class Case

PARAMETERS:

i: event (call arrival or departure) index

β: desired update rate(updates/hour)

$N(i)$: number of calls in progress after the i-th event

N_L: number of calls when the last update occurred

$R(i)$: reserved bandwidth after the i-th event

R_L: reserved bandwidth when the last update occurred

C_m: maximum allocated capacity

bucket: leaky bucket parameter

d: hysteresis band barameter

B_m: maximum leaky bucket size

t_i: inter-event time between ith and $i-1$th events

THE ALGORITHM

$bucket \Leftarrow bucket - t_i\beta/3600$\{leak the bucket\}

$bucket \Leftarrow \max(0, bucket)$\{guarantee the min bucket size\}

$d \Leftarrow \frac{C_m}{B_m} bucket$ \{adjust the hysteresis band\}

if $N(i) \notin \{N_L - d, N_L + d\} or N(i) > R_L$ **then**

  if $d = 0$ **then**

    $R(i) \Leftarrow N(i)$\{make a decision\}

  else

    $R(i) \Leftarrow \min(C_m, N(i) + \lceil d \rceil)$\{make a decision\}

  end if

  if $R(i) \neq R_L$ **then**

    $bucket \Leftarrow \min(B_m, bucket + 1)$\{update the bucket\}

    $d \Leftarrow \frac{C_m}{B_m} bucket$ \{adjust the hysteresis band\}

    $R_L \Leftarrow R(i)$ \{Update the last reserved bandwidth\}

    $N_L \Leftarrow N(i)$ \{Update the last number of calls\}

  end if

end if

Algorithm 6 Adaptive Hysteresis for Multi-Class Case

PARAMETERS:

n: number of classes.
i^j: event (call arrival or departure) index of the j-th class
β^j: desired update rate(updates/hour) of the j-th class
$N^j(i^j)$: number of calls in progress after the i-th event in the j-th class
N_L^j: number of calls when the last update occurred in the j-th class
$R^j(i^j)$: reserved bandwidth after the i-th event in the j-th class
R_L^j: reserved bandwidth when the last update occurred in the j-th class
C_m: maximum physical link capacity
$bucket^j$: leaky bucket parameter of the j-th class
d^j: hysteresis band parameter of the j-th class
B_m^j: maximum leaky bucket size of the j-th class
t_i^j: inter-event time between i^jth and $i^j - 1$th of the j-th class

THE ALGORITHM

$bucket^j \Leftarrow bucket^j - t_i^j \beta^j / 3600$ {leak the bucket}
$bucket^j \Leftarrow \max(0, bucket^j)$ {guarantee the min bucket size}
$d^j \Leftarrow \dfrac{C_m}{nB_m^j} bucket^j$ {adjust the hysteresis band}
if $N^j(i^j) \notin \{N_L^j - d^j, N_L^j + d^j\} or N^j(i^j) > R_L^j$ **then**
  if $d^j = 0$ **then**
    $R^j(i^j) \Leftarrow N^j(i^j)$ {make a decision}
  else
    $R^j(i^j) \Leftarrow \min(C_m - \sum_{k=0,k\neq j}^{n-1} R^k(i^k), N^j(i^j) + \lceil d^j \rceil)$ {make a decision}
  end if
  if $R^j(i^j) \neq R_L^j$ **then**
    $bucket^j \Leftarrow \min(B_m^j, bucket^j + 1)$ {update the bucket}
    $d^j \Leftarrow \dfrac{C_m}{nB_m^j} bucket^j$ {adjust the hysteresis band}
    $R_L^j \Leftarrow R^j(i^j)$ {Update the last reserved bandwidth}
    $N_L^j \Leftarrow N^j(i^j)$ {Update the last number of calls}
  end if
end if

3.5 Performance Evaluation

3.5.1 Single-Class Case with Stationary Poisson Voice Traffic

We have compared the performances of the adaptive hysteresis (the proposed method), RVI, and synchronous approach under a stationary Poisson traffic in terms of reserved bandwidth and the gain with respect to SVC approach. We note that average reserved bandwidth by PVP and SVC approaches are included as reference. For each simulation, the traffic parameters and assumptions are given as follows;

- Maximum physical link capacity $C_m = 16$ identical channels
- Average service time $1/\mu = 180$ seconds
- Call blocking probability $P_b = 0.01$
- The call arrival rate $\lambda = 0.0493055$ calls/sec which can be calculated by Erlang B formula using C_m, P_b, and μ.
- Maximum bucket size $B_m = 16$ for the proposed approach.
- Average update rate β is varied from 2 to 350 updates/hour for each method.
- For RVI, we have chosen r such that the average update rate becomes the same as those of others and also $K = 1000$, $h = 1$.
- We have taken the average of 5 simulations each having 10^7 calls.

Average Reserved Bandwidth

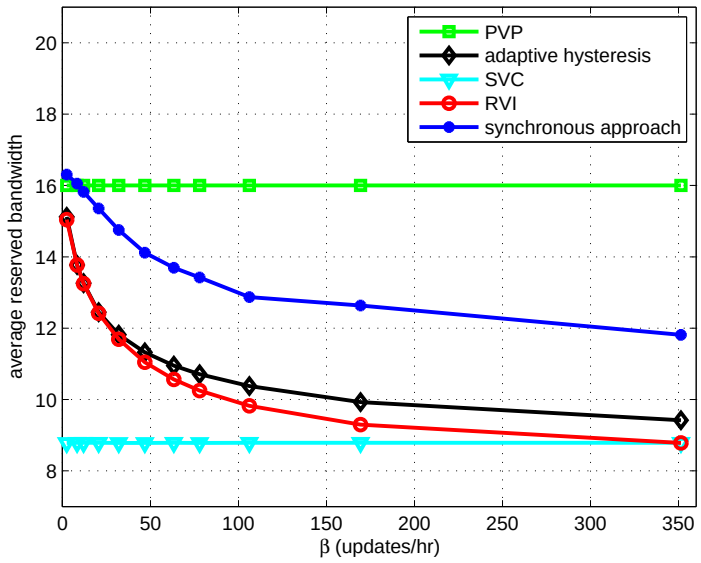

Figure 3.3: Average Reserved Bandwidth

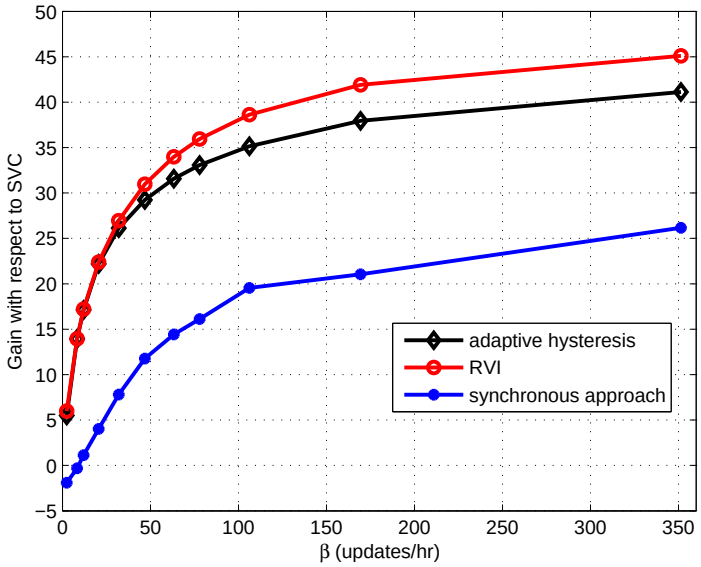

Figure 3.4: Gain with respect to SVC

In Fig. 3.3, we see the average bandwidth reserved by all methods. For PVP approach, the reserved bandwidth is always equal to C_m which is 16 for this example. For SVC approach, since the switching cost is not considered, it has the

minimum reservation which can be calculated by $\lambda(1-P_b)/\mu$ and is particularly 8.785 for this scenario. Between SVC and PVP bounds, all other approaches have some reservations as a function of β. We observe that the proposed method and RVI significantly outperform the synchronous approach since they take the advantage of asynchronous update epochs. When we compare the proposed method with the RVI approach we see that up to 40 updates/hour, they have nearly the same performance. After that point, we see that RVI slightly outperforms the proposed method. On the other hand, we know that RVI and synchronous approach assume a model to be available and they run only under stationary Poisson traffic with known arrival and departure rates. However, adaptive hysteresis algorithm is model-free and it works without knowing the traffic rate and process. Fig. 3.4 shows the gains attainable by all methods with respect to SVC approach. For the synchronous approach, the achievable gains are varied from 0% to 25% with β. Again the proposed method and RVI outperform the synchronous approach in terms of gain. The achievable gains are varied from 5% to 45% for RVI and from 5% to 40% for adaptive hysteresis. Here we see that the maximum achieved gain by the proposed method is only 5% less than the optimum value.

In Fig. 3.5, we see the bandwidth reservation behavior of the proposed method for different values of beta for 1800 seconds in the current scenario. Note that the desired update rate β could be considered as available credits for bandwidth updates. Our observations show that the proposed method in high updates rates, uses only a portion of the available credits.

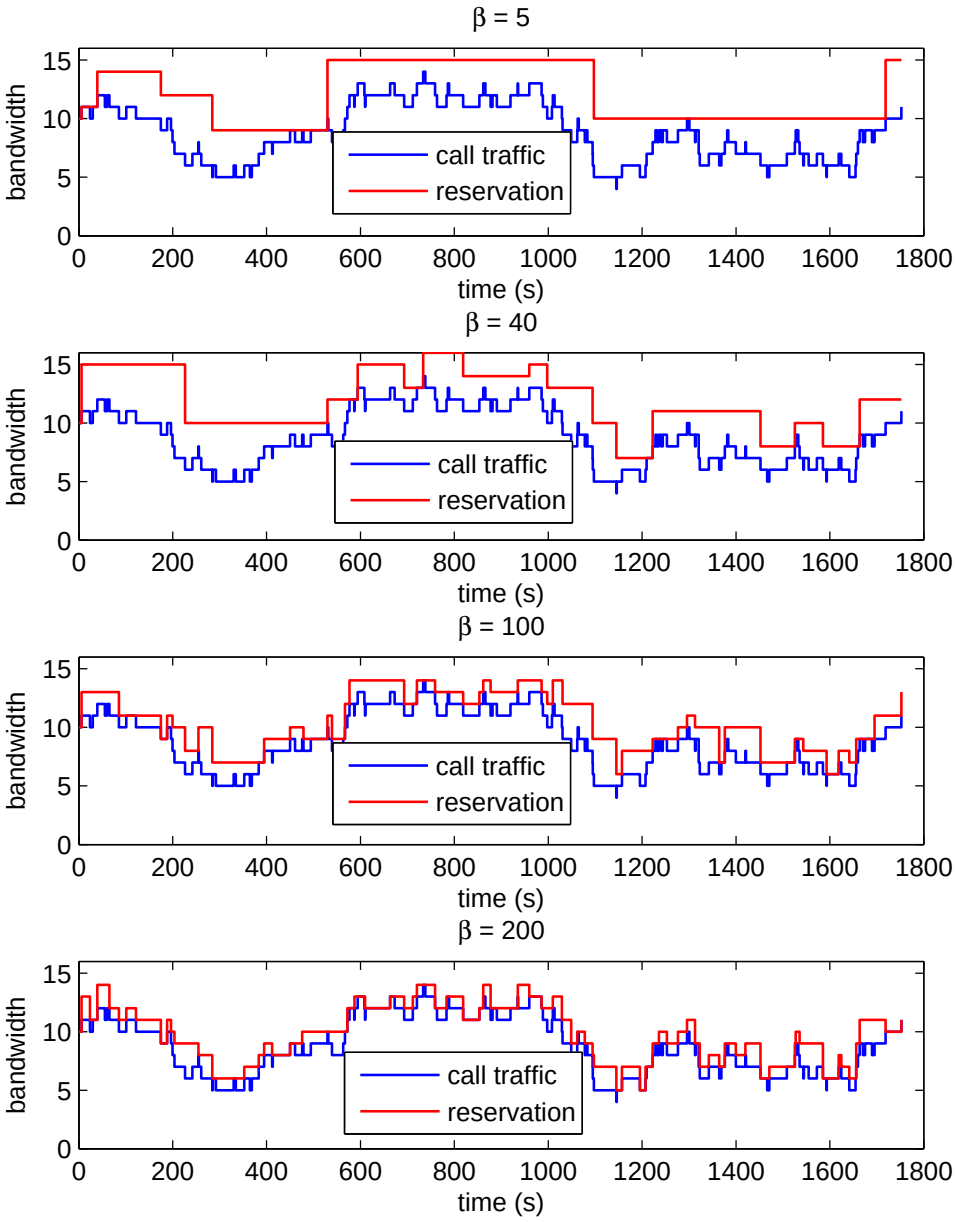

Figure 3.5: Reserved Bandwidth by Adaptive Hysteresis for Different Values of β

In order to describe how the proposed algorithm works, we construct an example system that starts at $t = 0$ and for which $C_m = B_m = 10$, $N(0^+) = 5$, $R(0^+) = 6$, $B(0^+) = 2$ and $\beta = 1/4$ updates/min [38]. We assume at $t = 0^+$,

a bandwidth update has just occurred. Note that with this choice of β we have 15 update opportunities per hour. Instead of a tele-traffic model, we introduce arrivals and departures at pre-specified instances for this system. The evolution of $N(t)$, $R(t)$, and the lower and upper hysteresis thresholds are illustrated in Fig. 3.6.

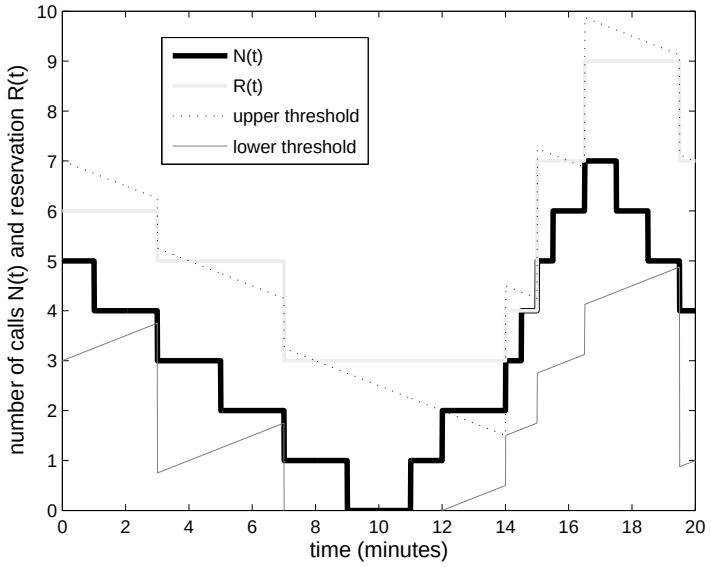

Figure 3.6: The evolution of number of ongoing calls $N(t)$ and the reservation $R(t)$ as a function of t for a sample scenario for which $C_m = B_m = 10$, $N(0) = 5$, $R(0) = 6$, $B(0) = 2$ and $\beta = 1/4$ updates/min.

Convergence to Desired Update Rate (β)

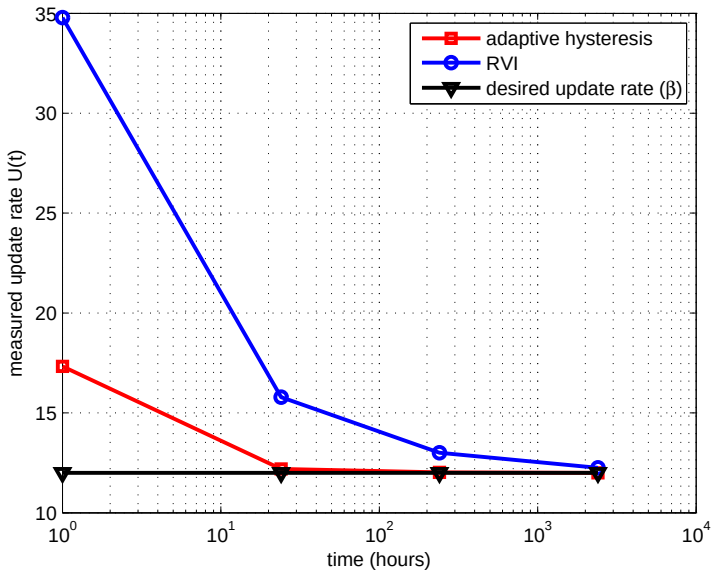

Figure 3.7: Convergence to β

We define the measured update rate $U(t)$ as a function of t as the average update rate measured in the interval $[0, t]$.Fig. 3.7 shows the convergence of $U(t)$ to the desired update rate as $t \rightarrow \infty$ for the proposed technique and RVI. We see that adaptive hysteresis converges faster than RVI. We know that real traffic conditions have a non-stationary characteristics. Based on this characteristics we can conclude that adaptive hysteresis can easily adapt the variable traffic conditions in terms of β.

Comparison with Non-Linear Versions

In the adaptive hysteresis algorithm, when defining the hysteresis band parameter d, we have used $\frac{C_m}{B_m} bucket$ which is the linear version of the band control. On the other hand, in order to investigate the performances of the non-linear versions of the band control, we have proposed adaptive hysteresis-square approach with band $\frac{C_m}{B_m^2} bucket^2$ and adaptive hysteresis-square-root approach with

band $\frac{C_m}{\sqrt{B_m}}\sqrt{bucket}$. In Fig. 3.8, we see the performances of the linear and non-linear versions of the proposed algorithm. Adaptive hysteresis-square approach performs a little bit better than the linear version. On the other hand, linear approach significantly outperforms adaptive hysteresis-square-root approach.

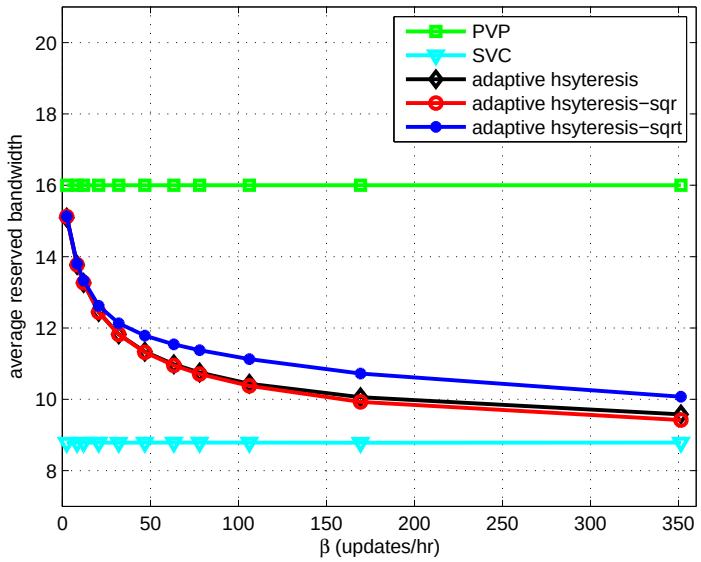

Figure 3.8: Average Reserved Bandwidth

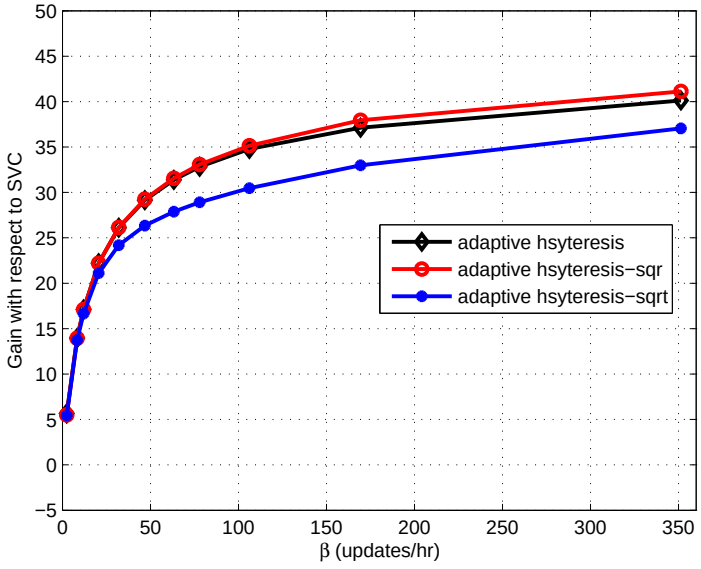

Figure 3.9: Gain with respect to SVC

Fig. 3.9 shows the gains achievable by all versions with respect to SVC approach. Up to 100 updates/hour, linear and square versions perform nearly the same. After that rate, square version exceeds the linear one about 1% percent. Here we see that the square-root version is 5% outperformed by the others.

Varying Maximum Bucket Size (B_m)

In the same simulation scenario, we have varied maximum bucket size B_m for several βs in order to see the effects of it to the gain with respect to SVC. Fig. 3.10 shows the attainable gains for several values of β. For each simulation except $\beta = 2.43$, the maximum gains have been obtained when $B_m = 16$ which is equal to C_m for the current simulation scenario. As a result, we can say that a suitable choice of B_m could be C_m for single-class case and C_m/n for multi-class case.

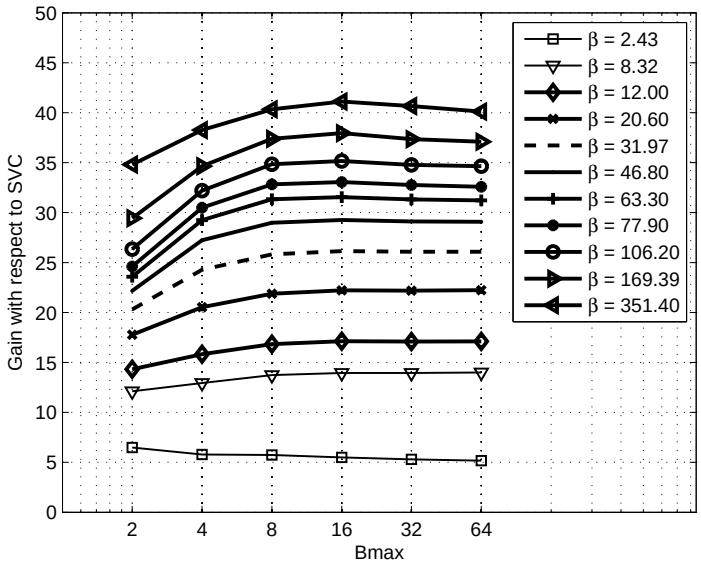

Figure 3.10: Gains with respect to SVC by varying B_m

Varying Maximum Physical Link Capacity (C_m)

In order to study the effect of C_m on the proposed approach, we have designed three simulation scenarios. For each scenario we have used the same service rate which is 1/180. On the other hand, for each different C_m in each simulation, we have chosen different arrival rates such that the loss probabilities to be 0.01 for each scenario. Then we have plotted the gain with respect to SVC for each scenario as shown in Fig. 3.11. For this example, we see that the maximum achievable gains for $C_m = 8, 16, 32$ are $60\%, 45\%, 30\%$ respectively. From the figure, it is clear that the systems with low capacity have more significant gains than the high ones.

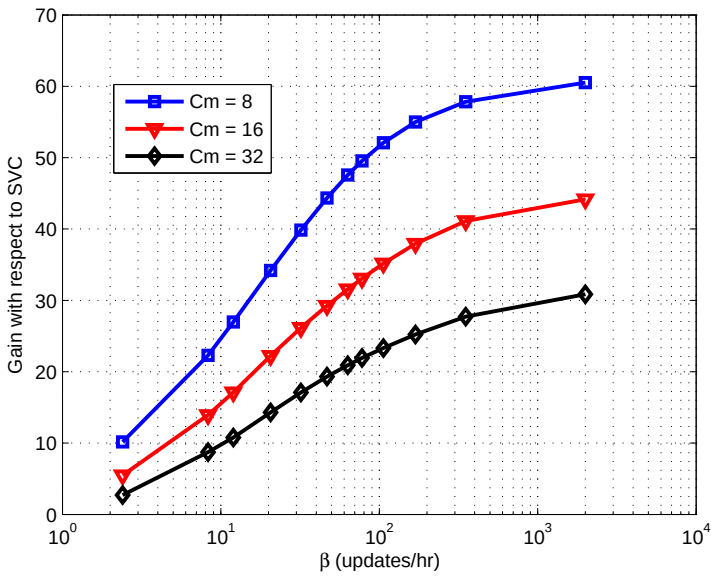

Figure 3.11: Gains with respect to SVC by varying C_m

3.5.2 Multi-Class Case with Stationary Poisson Voice Traffic

Up to now, we have investigated the effects and properties of the proposed algorithm in a single VP for a single class stream. However, a physical link can

consists of various VPs and traffic classes streaming in those VPs. For this reason, we have designed an experiment which has the following parameters and assumptions;

- Number of VPs = 6.
- Maximum physical link capacity $C_m = 96$ identical channels
- Average service time $1/\mu = 180$ seconds for each class
- Maximum call blocking probability $P_b = 0.01$ for each class
- The call arrival rate to each class, $\lambda = 0.0493055$ calls/sec which can be calculated by Erlang B formula using C_m, P_b, and μ.
- Maximum bucket size $B_m = 16$.
- Average update rate β is varied from 2 to 350 updates/hour.
- We have taken the average of 5 simulations each having 2.10^7 calls.

Loss Reduction

While varying β, we have plotted the loss rates of any VP in the link. Unlike single stream case for which the loss probability P_b stays the same for each β, the loss probability for any VP in the link in multi-class case significantly reduces as β increases. We can see from Fig. 3.12 that the loss probability reduces approximately from 10^0 to 10^{-4} as β increases up to 350.

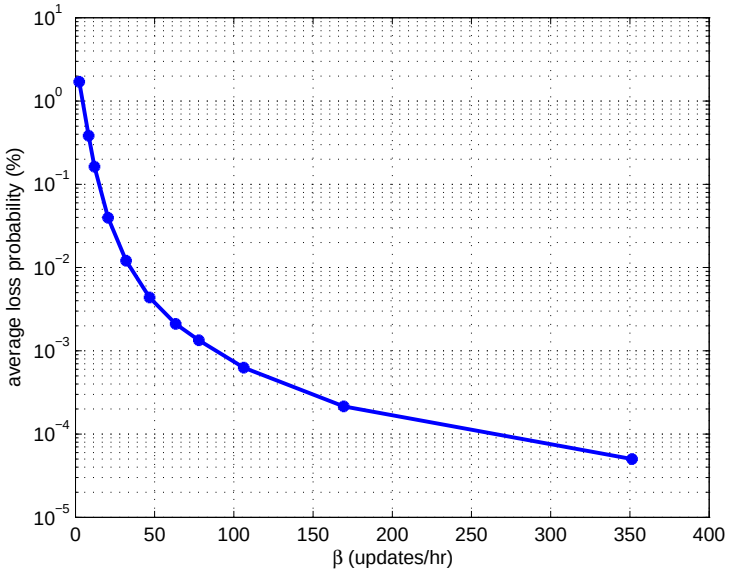

Figure 3.12: Loss probability for any VP in the physical link

Varying Number of Paths

In the same example, we have also varied the number of the VPs in order to see the effects of the variation on the gain. We have studied three choices for the number of streams $n = 3, 6, 12$. For each scenario, we have adjusted the maximum physical capacity accordingly. Fig. 3.13 shows that as n increases the gain increases as well.

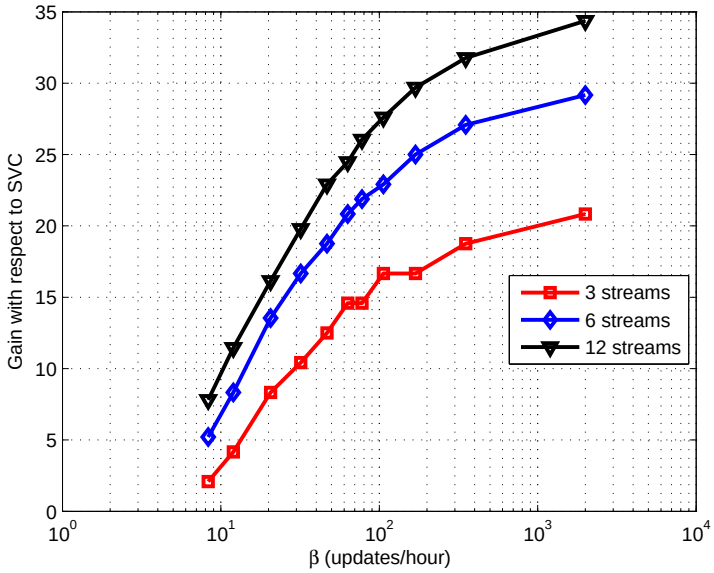

Figure 3.13: Gains with respect to SVC by varying n

3.5.3 Non-Stationary Poisson Voice Traffic Case

In the previous parts, we have assumed that the call traffic is stationary with rate $\lambda_m = \max_t \lambda(t)$ which is worst case scenario. On the other hand, in realistic networks, the call arrival rate varies over the time. In order to see the performance of the proposed algorithm in a non-stationary traffic, we have set up an experiment which has the wide area network (WAN) topology as shown in fig 3.14. In this topology, we have considered the traffic between the nodes 2 and 3. The activity model has been taken from [33] and $\lambda(t)$ has been calculated for 24 hours as shown in fig 3.15.

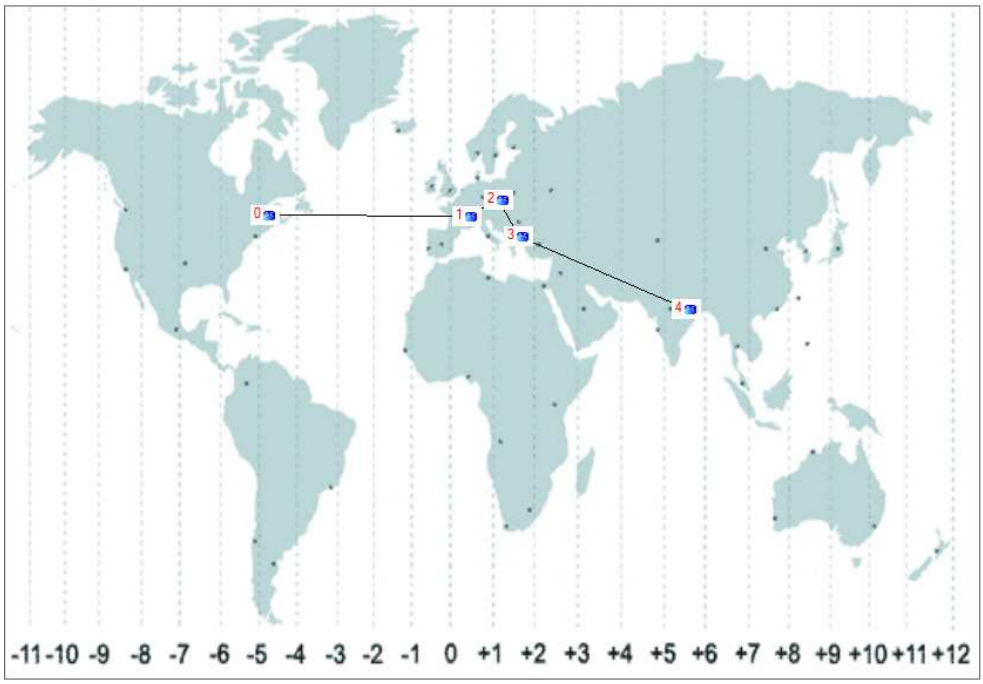

Figure 3.14: A 5-node wide area network topology

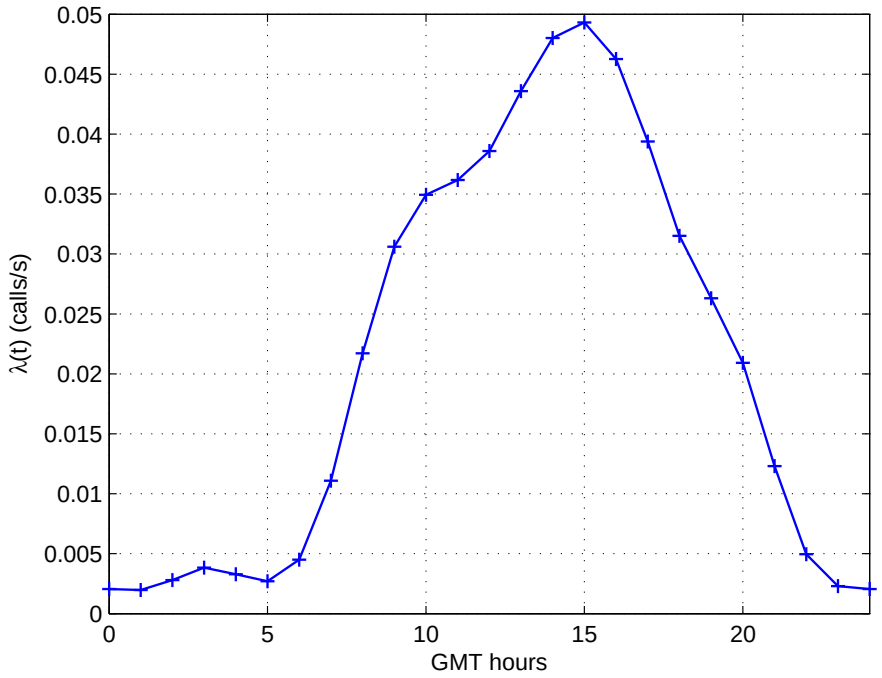

Figure 3.15: $\lambda(t)$ between the nodes 2 and 3

Under this traffic, we have evaluated the reserved bandwidth and gain with respect to SVC for a single VP. The other traffic parameters kept the same as those of the stationary Poisson scenario. In fig 3.16, we can see the reserved bandwidth in the current traffic scenario. As compared to the stationary case, it is relatively smaller and closer to SVC. Also, fig 3.17 shows that the maximum achievable gain varies from 44% to 74% as β increases. Note that the loss probability stayed about 0.29% for each β. From these results, we can conclude that the maximum achievable gains in non-stationary traffics significantly increase with the proposed algorithm.

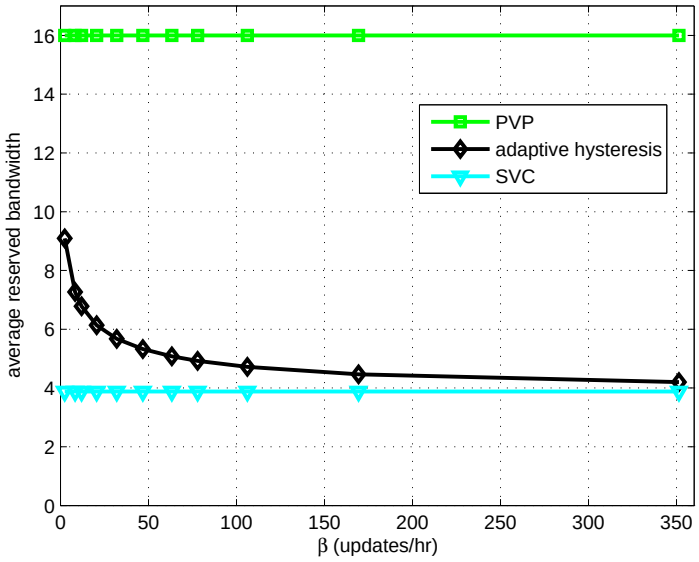

Figure 3.16: Average Reserved Bandwidth

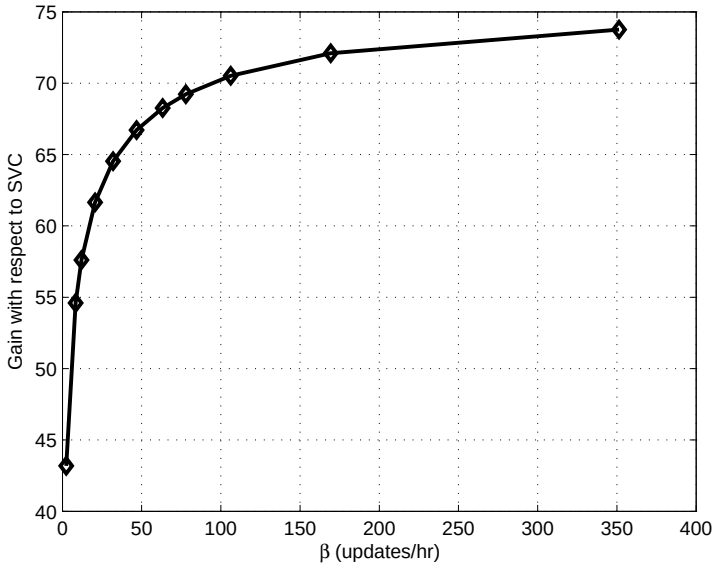

Figure 3.17: Gain with respect to SVC

3.5.4 Single-Class Case with Self-Similar Internet Data Traffic

In the original algorithm, which has been used for the DBR problem in voice traffic, the system state is defined by the number of calls at any time instant. On the other hand, in the self-similar Internet traffic, we define the system state as the current used bandwidth in a path. This bandwidth value is periodically measured and at the end of each measurement a decision is made. We have tested the proposed method under several synthetic traces and a one-day trace taken from a traffic data repository maintained by the MAWI (Measurement and Analysis on the WIDE Internet) Working Group of the WIDE Project [1]. We have also compared the results with CISCO's auto-bandwidth allocator (referred to as *cisco-aba* in short) since it is measurement based and used for data traffic [32]. For the synthetic traffic generation, we have used the following parameters;

- Maximum physical link capacity $C_m = 10$ Mbps

- Simulation length is set to 30 days
- Flow arrival process is Poisson with $\lambda(t)$ as shown in fig 3.15.
- Flow lengths are pareto distributed with hurst parameter 0.8
- Packet size distribution is taken from Table 2.1 which uses the traffic traces from [1]
- Monitoring period is set to 300 seconds (5 Minutes)

Under these conditions, we have varied the parameters B_m and β, and obtained the gain with respect to maximum link capacity. Fig. 3.18 shows the attainable gains for several values of B_m and β. As expected, when we increase β, the achievable gains increase as well. We have also observed that when B_m is set to C_m or $5C_m$, we can approximately obtain the maximum achievable gains for each β.

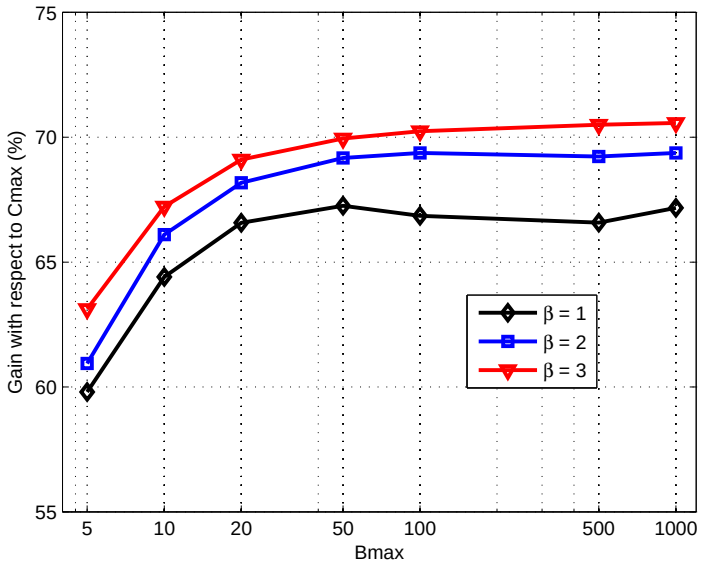

Figure 3.18: Gains with respect to C_m by varying B_m and β

By setting B_m to $5C_m$, we have compared the losses (%) and gains (%) of *cisco-aba* and *adp-hys* for different values of β as shown in Table 3.1. As we

Table 3.1: Performance Comparison

β	loss (*cisco-aba*)	loss (*adp-hys*)	gain (*cisco-aba*)	gain (*adp-hys*)
0.300	12.111	0.084	63.023	58.696
1.000	5.218	0.389	69.234	67.257
2.000	2.898	0.600	70.588	69.172
3.000	2.178	0.796	71.260	69.951

see in the table, *cisco-aba* has a little bit higher gains (about 5% to 1%) than *adp-hys*. On the other hand, *cisco-aba* permits huge loss rates while *adp-hys* allows very small losses. For the visualization, we have also 2-day snapshot of the bandwidth reservations done by *cisco-aba* and *adp-hys* for *beta* set to 0.3 and 1 updates/hour as shown in Fig. 3.19 and Fig. 3.20 respectively.

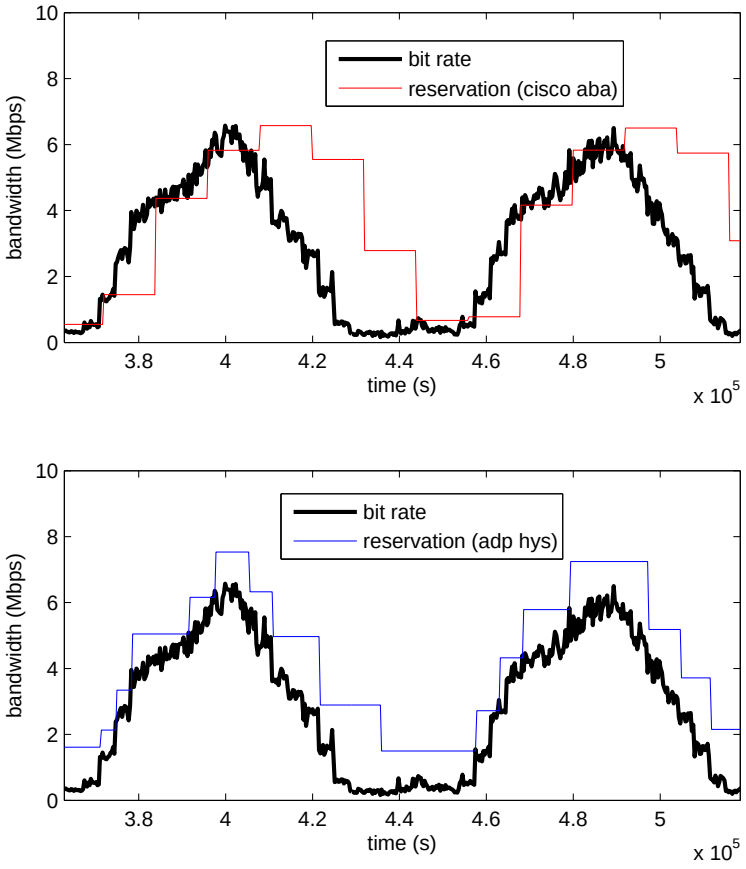

Figure 3.19: Bandwidth reservation with $\beta = 0.3$

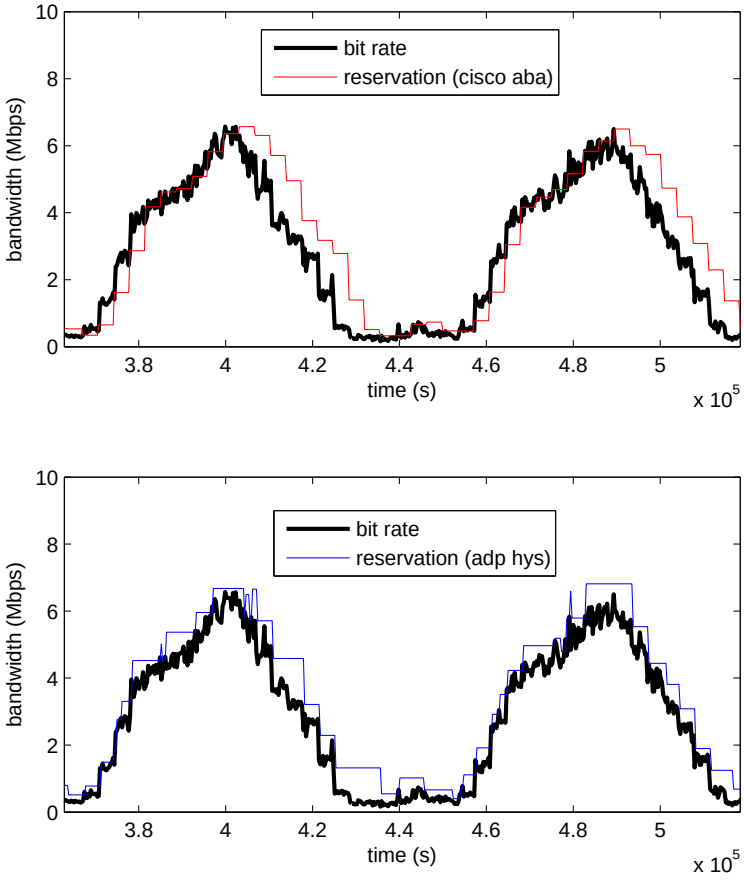

Figure 3.20: Bandwidth reservation with $\beta = 1$

We have also compared the bandwidth reservations under a real traffic trace obtained from the WIDE backbone at Sample Point B on May 14, 1999 for US-Japan link with 10 Mbps link speed [1]. Since the trace is not long enough to compare the gains and losses, we have only shown the reserved bandwidths in Fig. 3.21 and Fig. 3.22 done by *cisco-aba* and *adp-hys* for *beta* set to 0.3 and 1 updates/hour respectively.

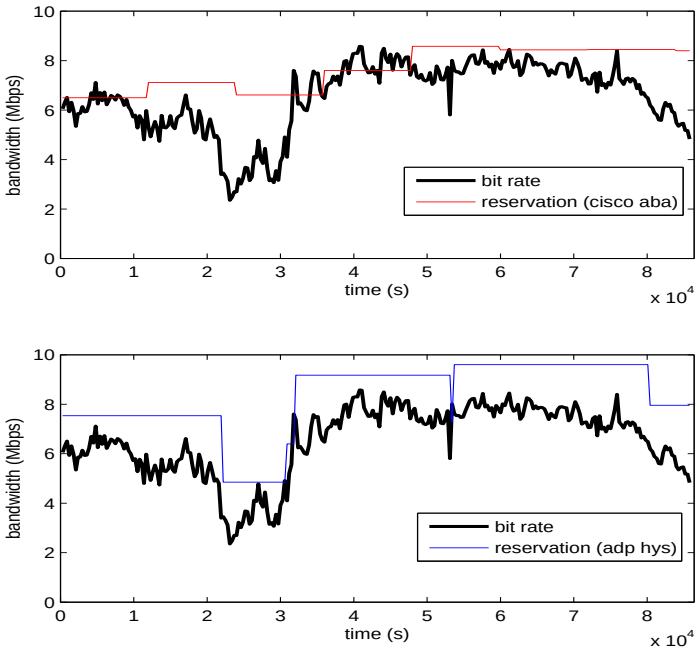

Figure 3.21: Bandwidth reservation with $\beta = 0.3$

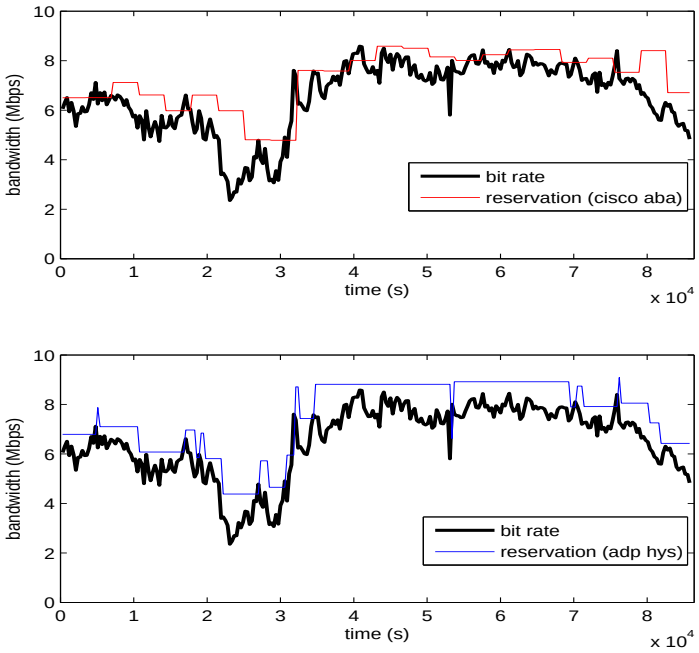

Figure 3.22: Bandwidth reservation with $\beta = 1$

Chapter 4

CONCLUSIONS

In this thesis, we have studied two problems arising in communication networks. In the first problem, we have proposed two dynamic-threshold based algorithms that aim at the reduction of average assembly delays (packet or byte delays) at the burst assembly buffers located at the edge of an OBS network while conforming to a desired burst rate. Moreover, enforcement of lower and upper burst length limits is embedded in these algorithms. The major contribution of this part is the significant reduction of average assembly delays while keeping the short- and long-term burst rates close to the desired burst rate by means of dynamically adjusting the assembly threshold in case of changing traffic conditions. The benefits of the proposed algorithms are demonstrated with both synthetic traffic and actual traffic traces. Moreover, the algorithms are model-free and simple to implement making them viable alternatives for the design and implementation of burst assembly units in next-generation OBS systems.

In the second problem, we have presented an adaptive hysteresis algorithm for dynamic bandwidth reservations focusing on the reduction of the reserved bandwidth in a connection-oriented network while limiting the average update rate to a desired update rate in both long and short terms. We have observed that

the major contribution of the algorithm is notable bandwidth gains with certain desired update rates without requiring any traffic model and prior information. Moreover, using some optimization techniques in which the traffic model is assumed to be known, we have shown that the performance of the proposed scheme is very close to the optimum. In addition to potential enhancements in bandwidth use, the simplicity of the proposed algorithm offers a promising solution to network administrators for engineering next-generation connection-oriented networks.

Bibliography

[1] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository maintained by the MAWI Working Group of the WIDE Project, http://mawi.wide.ad.jp/mawi."

[2] Matisse Networks, http://www.matissenetworks.com/.

[3] C. Qiao and M. Yoo, "Optical burst switching (OBS) - a new paradigm for an optical Internet," *J. High Speed Networks*, vol. 8, no. 1, pp. 69–84, 1999.

[4] S. Verma, H. Chaskar, and R. Ravikanth, "Optical burst switching: a viable solution for terabit IP backbone," *IEEE Network Mag.*, vol. 14, no. 6, pp. 48–53, 2000.

[5] M. de Prycker, *Asynchronous transfer mode: Solution for Broadband ISDN*. Prentice Hall PTR, 1995.

[6] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann Publishers, 2000.

[7] F. Baker, C. Iturralde, F. Le Faucheur, and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 reservations," *Work in Progress*.

[8] A. Ge, F. Callegati, and L. Tamil, "On optical burst switching and self-similar traffic," *Communications Letters, IEEE*, vol. 4, pp. 98–100, Mar 2000.

[9] V. M. Vokkarane, K. Haridoss, and J. P. Jue, "Threshold-based burst assembly policies for QoS support in optical burst-switched networks," in *in Proc. SPIE OptiComm 2002*, pp. 125–136, 2002.

[10] X. Yu, Y. Chen, and C. Qiao, "Study of traffic statistics of assembled burst traffic in optical burst switched networks," in *Proc. Opticomm*, pp. 149–159, 2002.

[11] X. Cao, J. Li, Y. Chen, and C. Qiao, "Assembling TCP/IP packets in optical burst switched networks," in *IEEE GLOBECOM*, vol. 3, pp. 2808–2812, Nov. 2002.

[12] S.-Y. Oh, H. H. Hong, and M. Kang, "A data burst assembly algorithm in optical burst switching networks," *ETRI Journal*, vol. 24, no. 4, pp. 311–322, 2002.

[13] P. Du and S. Abe, "Burst assembly method with traffic shaping for the optical burst switching network," in *IEEE GLOBECOM*, (San Francisco, CA, USA), 27 2006-Dec. 1 2006.

[14] N. Korkakakis and K. Vlachos, "An adaptive burst assembly scheme for OBS-GRID networks," in *Communication Systems, Networks and Digital Signal Processing, 2008. CNSDSP 2008. 6th International Symposium on*, pp. 414–417, July 2008.

[15] J. N. T. Sanghapi, H. Elbiaze, and M. Zhani, "Adaptive burst assembly mechanism for OBS networks using control channel availability," in *Transparent Optical Networks, 2007. ICTON '07. 9th International Conference on*, vol. 3, pp. 96–100, July 2007.

[16] X. Yu, Y. Chen, and C. Qiao, "Performance evaluation of optical burst switching with assembled burst traffic input," in *IEEE Global Telecommunications Conference, 2002. GLOBECOM'02*, vol. 3, 2002.

[17] W. Leland, M. Taqqu, W. Willinger, D. Wilson, and M. Bellcore, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on networking*, vol. 2, no. 1, pp. 1–15, 1994.

[18] X. Yu, J. Li, X. Cao, Y. Chen, and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *Lightwave Technology, Journal of*, vol. 22, no. 12, pp. 2722–2738, 2004.

[19] G. Hu, K. Dolzer, and C. Gauger, "Does burst assembly really reduce the self-similarity?," in *Optical Fiber Communications Conference, 2003. OFC 2003*, pp. 124–126, 2003.

[20] J. H. Hong and K. Sohraby, "On the asymptotic analysis of packet aggregation systems," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS '07. 15th International Symposium on*, pp. 353–359, Oct. 2007.

[21] H. Heffes and D. Lucantoni, "A Markov modulated characterization of packetized voice and date traffic and related statistical multiplexer performance," *JSAC*, vol. 4, no. 6, pp. 856–868, 1986.

[22] L. Muscariello, M. Meillia, M. Meo, M. Marsan, and R. Cigno, "An mmpp-based hierarchical model of internet traffic," in *Communications, 2004 IEEE International Conference on*, vol. 4, pp. 2143–2147, June 2004.

[23] T. Bohnert and E. Monteiro, "A comment on simulating LRD traffic with pareto ON/OFF sources," in *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pp. 228–229, ACM New York, NY, USA, 2005.

[24] S. Ohta and K. Sato, "Dynamic bandwidth control of the virtual path in an asynchronous transfer mode network," *IEEE Transactions on Communications*, vol. 40, no. 7, pp. 1239–1247, 1992.

[25] A. Orda, G. Pacifici, and D. Pendarakis, "An adaptive virtual path allocation policy for broadband networks," in *Proceedings IEEE INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation*, vol. 1, 1996.

[26] U. Mocci, P. Pannunzi, and C. Scoglio, "Adaptive capacity management of virtual path networks," in *Global Telecommunications Conference, 1996. GLOBECOM'96.'Communications: The Key to Global Prosperity*, vol. 1, 1996.

[27] H. Levy, T. Mendelson, and G. Goren, "Dynamic allocation of resources to virtual path agents," *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 4, pp. 746–758, 2004.

[28] B. Krithikaivasan, K. Deka, and D. Medhi, "Adaptive bandwidth provisioning envelope based on discrete temporal network measurements," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2004.

[29] M. Gursoy, J. Hui, N. Moayeri, and R. Yates, "A layered broadband switching architecture with physical or virtualpath configurations," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 9, pp. 1416–1426, 1991.

[30] T. Anjali, C. Scoglio, and G. Uhl, "A new scheme for traffic estimation and resource allocation for bandwidth brokers," *Computer Networks*, vol. 41, no. 6, pp. 761–777, 2003.

[31] T. Anjali, C. Bruni, D. Iacoviello, and C. Scoglio, "Dynamic bandwidth reservation for label switched paths: An on-line predictive approach," *Computer Communications*, vol. 29, no. 16, pp. 3265–3276, 2006.

[32] M. Cisco, "AutoBandwidth Allocator for MPLS Traffic Engineering: A Unique New Feature of Cisco IOS Software," *White Paper, Cisco Systems.*

[33] J. Milbrandt, M. Menth, and S. Kopf, "Adaptive bandwidth allocation for wide area networks," in *COST-279 Management Committee Meeting, Antalya, Turkey*, 2005.

[34] S. Dasgupta, J. de Oliveira, and J. Vasseur, "A new distributed dynamic bandwidth reservation mechanism to improve resource utilization," in *Proceedings of IEEE INFOCOM*, 2006.

[35] S. Dasgupta, J. de Oliveira, and J. Vasseur, "Trend based bandwidth provisioning: An online approach for traffic engineered tunnels," *Next Generation Internet Networks, 2008. NGI 2008*, pp. 53–60, 2008.

[36] A. Reibman and K. Trivedi, "Numerical transient analysis of Markov models," *Computers and Operations Research*, vol. 15, no. 1, pp. 19–36, 1988.

[37] H. Tijms, *A First Course in Stochastic Models*. John Wiley and Sons, 2003.

[38] N. Akar, "Model-free adaptive hysteresis for dynamic bandwidth reservation," in *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS'07. 15th*, pp. 331–336, 2007.