CONNECTIONLESS TRAFFIC AND VARIABLE PACKET SIZE SUPPORT IN HIGH SPEED NETWORK SWITCHES: IMPROVEMENTS FOR THE DELAY-LIMITER SWITCH

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİCAN AKÇASOY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2008

Approval of the thesis:

CONNECTIONLESS TRAFFIC AND VARIABLE PACKET SIZE SUPPORT IN HIGH SPEED NETWORK SWITCHES: IMPROVEMENTS FOR THE DELAY-LIMITER SWITCH

Submitted by **ALİCAN AKÇASOY** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. İsmet Erkmen
Head of Department, **Electrical and Electronics Eng.** _______________

Assist. Prof. Dr. Şenan Ece Güran Schmidt
Supervisor, **Electrical and Electronics Eng.** _______________

Examining Committee Members

Prof. Dr. Semih Bilgen
Electrical and Electronics Eng, METU _______________

Assist. Prof. Dr. Şenan Ece Güran Schmidt
Electrical and Electronics Eng, METU _______________

Associate Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Eng, METU _______________

Assist. Prof. Dr. Cüneyt Bazlamaçcı
Electrical and Electronics Eng, METU _______________

Özgül Boyacı, M.Sc.
Production Engineering, ASELSAN A.Ş. _______________

Date: 06.06.2008

ABSTRACT

CONNECTIONLESS TRAFFIC AND VARIABLE PACKET SIZE SUPPORT IN HIGH SPEED NETWORK SWITCHES: IMPROVEMENTS FOR THE DELAY-LIMITER SWITCH

Akçasoy, Alican
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Şenan Ece Güran Schmidt

June 2008, 146 pages

Quality of Service (QoS) support for real-time traffic is a critical issue in high-speed networks. The previously proposed Delay-Limiter Switch working with the Framed-Deadline Scheduler (FDS) is a combined input-output queuing (CIOQ) packet switch that can provide end-to-end bandwidth and delay guarantees for connection-oriented traffic. The Delay-Limiter Switch works with fixed-size packets. It has a scalable architecture and can provide QoS support for connection-oriented real-time traffic in a low-complexity fashion. The Delay-Limiter Switch serves connectionless traffic by using the remaining resources from the connection-oriented traffic. In this case, efficient management of the residual resources plays an important role on the performance of the connectionless traffic. This thesis work integrates new methods to the Delay-Limiter Switch that can improve the performance of the connectionless traffic while still serving the connection-oriented traffic with the promised QoS guarantees. A new method that makes it possible for the Delay-Limiter Switch to support variable-sized packets is also proposed.

Keywords: high-speed router, QoS, scheduler, real-time, best-effort

ÖZ

YÜKSEK HIZLI AĞ ANAHTARLARINDA BAĞLANTISIZ TRAFİK VE DEĞİŞKEN PAKET BOYU DESTEĞİ: GECİKME SINIRLAYICI ANAHTAR İÇİN GELİŞTİRMELER

Akçasoy, Alican
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Yard. Doç. Dr. Şenan Ece Güran Schmidt

Haziran 2008, 146 sayfa

Yüksek hızlı ağlarda gerçek zamanlı trafik için hizmet kalitesi desteği sağlamak kritik bir konudur. Daha önce önerilen Çerçeveli-Bitiş Noktası Çizelgeleyici ile çalışan Gecikme Sınırlayıcı Anahtar, bağlantı kipindeki trafik için uçtan uca bant genişliği ve gecikme garantisi sağlayabilen birleşik giriş-çıkış sıralı bir paket anahtarıdır. Gecikme Sınırlayıcı Anahtar sabit boydaki paketlerle çalışır. Ölçeklenebilir bir mimarisi vardır ve bağlantı kipindeki gerçek zamanlı trafik için düşük karmaşıklığa sahip bir usul ile hizmet kalitesi desteği sağlayabilir. Gecikme Sınırlayıcı Anahtar, bağlantısız trafiği bağlantı kipindeki trafikten artan kaynakları kullanarak yönlendirir. Bu durumda arta kalan kaynakların verimli kullanılması bağlantısız trafik performansı üzerinde önemli rol oynar. Bu tez çalışması Gecikme Sınırlayıcı Anahtar'a, bağlantı kipindeki trafiğe söz verilen hizmet kalitesi garantilerini sağlarken bağlantısız trafik performansını arttıran yeni yöntemler eklemektedir. Ayrıca Gecikme Sınırlayıcı Anahtar'ın değişken boydaki paketlerle çalışmasını mümkün kılan yeni bir yöntem önermektedir.

Anahtar Kelimeler: yüksek hızlı yönlendirici, hizmet kalitesi, çizelgeleyici, gerçek zamanlı, azami gayret

To Yeşer and To My Family

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis supervisor Assist. Prof. Dr. Şenan Ece Güran Schmidt for her invaluable guidance during my studies.

I would like to thank ASELSAN INC. for giving me the opportunity to carry on my graduate studies while pursuing a career in electronics. I would like to render thanks to Olcay Demirörs for encouraging me at the very early stages of my studies, Hür Topçuoğlu for his understanding and my friends, especially Ferhat Can Gözcü and Alper Akca, for being by my side.

I would like to thank my dear mother and my whole family for their endless love and generous support.

I would like to thank my fiancée Yeşer Güler for her supporting love and precious patience.

Finally, I should like to apologize to all people I like for my nonchalance and to myself for having to stay away many people and many things I like during my journey.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ATM: Asynchronous Transfer Mode

BWRR: Budgeted-Weighted-Round-Robin

CIOQ: Combined Input Output Queue

D-EDD: Delay-Earliest Due-Date

DLS: Delay-Limiter Switch

DLS/VPS: Delay-Limiter Switch with Variable Packet Size

DRR: Dual Round Robin

FDS: Framed-Deadline Scheduler

FIFO: First In First Out

FVATS: Framed Virtual Arrival Time Scheduler

GPS: Generalized Processor Sharing

HOL: head of line

HRR: Hierarchical Round Robin

I/O: input/output

IP: Internet Protocol

IQ: Input Queue

ISD: inter-switching delay

iDRR: Iterative Deficit-Round-Robin

iPDRR: Port-based Iterative Deficit-Round-Robin

iSLIP: Iterative SLIP

iTFF: Iterative “Time to leave” and “Full or not” First

LPF: Longest Port First

OQ: Output Queue

PIM: Parallel Iterative Matching

QoS: Quality of Service

RCSP: Rate Controlled Static Priority

RR: round-robin

SD: switching delay

S&G: Stop and Go

SCFQ: Self-Clocked Fair Queuing

SRA: Single Round-Robin Arbitration

Su: speed-up

TM: Traffic Matrix

TPC: Total Packet Count Algorithm

TSA: Time Slot Assignment

tvint: virtual interarrival time

VAT: Virtual Arrival Time

VND: Virtual Node Delay

VOQ: Virtual Output Queue

WFQ: Weighted Fair Queuing

WPIM: Weighted Probabilistic Iterative Matching

CHAPTER 1

INTRODUCTION

Data networks are composed of two main parts: network edge and network core. Network edge involves various applications and the hosts where these applications run while network core consists of high-speed transmission lines and routers that connect multiple transmission lines to each other. The main factors that affect the performance of the network core operation are the communication protocols employed and the architecture of the routers.

Ideally, data should pass through a high-speed router at the same speed with the line rate so that when the transmission line is observed it seems as if there were no routers on the path of the data. Moreover, because the routers are the contention points where many transmission lines intersect, the possible congestion problems must be handled in the most effective way. These imply that the routers on the network backbone must operate at very high speeds and very efficiently. The data networks must be able to support various quality of service (QoS) requirements for various applications although the routers in the network may have different sizes, port numbers and operating speeds. Hence, the router architectures and algorithms must be scalable. As a result of these facts, core network routers must be fast, efficient and scalable.

1.1 MOTIVATION OF THE THESIS

In this thesis, the previous work in [1] and [2] are extended and a novel high-speed switch architecture is proposed. [1] and [2] present a switch architecture called the Delay-Limiter Switch which provides end-to-end delay guarantees with a fixed amount of buffer requirement. The Delay-Limiter Switch operates with fixed-size frames in a connection oriented fashion with a constant time complexity of O(1).

Serving best-effort traffic with the best service quality possible is a significant issue in the-state-of-the-art networks since this type of traffic may carry time-sensitive data. Although the previous work on the delay-limiter switch, [1], proposes a fast and scalable router architecture to serve connection-oriented traffic, it does not give an answer to the question of how to efficiently transport connectionless best-effort traffic through the network. The improvement on the connectionless traffic performance was left as a future work.

The first contribution of this thesis is to elaborate the support for the best effort traffic in the Delay Limiter Switch by using the remaining resources after the connection oriented traffic with service guarantees.

The delay-limiter switch works with fixed-size packets. This requires the forming of large packets which may include several smaller packets at the network ingress. This situation introduces extra delay to the traffic. Furthermore, Internet Protocol (IP) which employs variable size packets is the dominating standard in the networks of today. The second contribution of this thesis is introducing a new switch architecture that combines the low time complexity property of the delay-limiter switch with the capability of operating on variable-sized packets. The new end-to-end delay guarantees achieved with the variable packet size support are derived which shows that the same order QoS guarantees are achieved as of the connection-oriented traffic which can be offered by the original delay-limiter switch.

The average end-to-end delay performance for the best effort traffic and the variable packet size support for the delay-limiter switch are demonstrated with simulation studies.

1.2 ORGANIZATION OF THE THESIS

The rest of this thesis is organized as follows. Chapter 2 presents a background on high-speed routers and their main components. Chapter 3 introduces the delay-limiter switch and its basic operation principles. Chapter 4 presents the add-ons to the delay-limiter switch architecture with a new method that provide the connectionless best-effort traffic with low delay values. Chapter 5 proposes a novel switch architecture that supports variable-size packets with end-to-end delay guarantees and operates with a constant time complexity. We show that this architecture requires a fixed amount of buffer size. Chapter 6 summarizes the conclusions and discusses future work.

CHAPTER 2

BACKGROUND

2.1 GENERAL HIGH-SPEED ROUTER ARCHITECTURE

Routers in a network are the junction points which connect many different transmission lines and provide data flows to be switched among these lines. They can be resembled to the junctions on the highways in such a way that while they serve to connect distinct lines to each other, they form contention points where the traffic flows must be handled in a controlled manner.

A generic high-speed router consists of the following parts which are shown in Figure 1.1.

Input Ports: Input Ports are the interfaces of a router to the incoming transmission lines. Data packets coming to a router arrive at an input port where they may be classified according to their service classes, formatted (fragmented into smaller sizes, stamped for unique properties, etc.) for proper operation in the router and stored in the input buffers if necessary.

Output Ports: Output Ports are the interfaces of a router to the outgoing transmission lines. Data packets which may have been formatted at the input ports are restored at the output ports. Data packets may also be stored in the output port buffers and scheduled for service order before they are transmitted on to the outgoing lines.

Switch Fabric: Switch fabric is the electronic circuitry which interconnects the input ports to the output ports to transport the incoming data packets to their destined outgoing lines.

Switch Fabric Scheduler: Switch fabric scheduler manages the arbitration of the input ports and configures the switch fabric to connect the input ports to the proper output ports according to a certain scheduling algorithm. An efficient fabric scheduler algorithm provides high throughput in the router which affects the overall performance of the whole network.

QoS Scheduler / Traffic Scheduler: QoS scheduler determines the order of the packets at an output port to be transmitted out from the router to the transmission line according to a certain scheduling algorithm. QoS scheduler algorithms play a very important role in providing service quality such as bandwidth, delay and jitter performance.

Routing Processor / Switch Processor: The routing processor runs the routing protocols and manages the overall system.

The router hardware should operate in certain timing whereas the arriving data packets can have variable sizes. It is more convenient to switch the data packets through the switch fabric within discrete time periods. Therefore, many routers adopt the concept of slotted operation. In this concept, the arriving data packets are segmented into fixed size units called *cells*. The time required for the operations of a single cell in the router is defined to be one *slot*. The arriving packets are fragmented into fixed size cells at the input ports, the cells are switched to the corresponding output ports through the switch fabric and the cells are reassembled into the original data packets at their output ports before they are sent out from the router to the transmission lines.

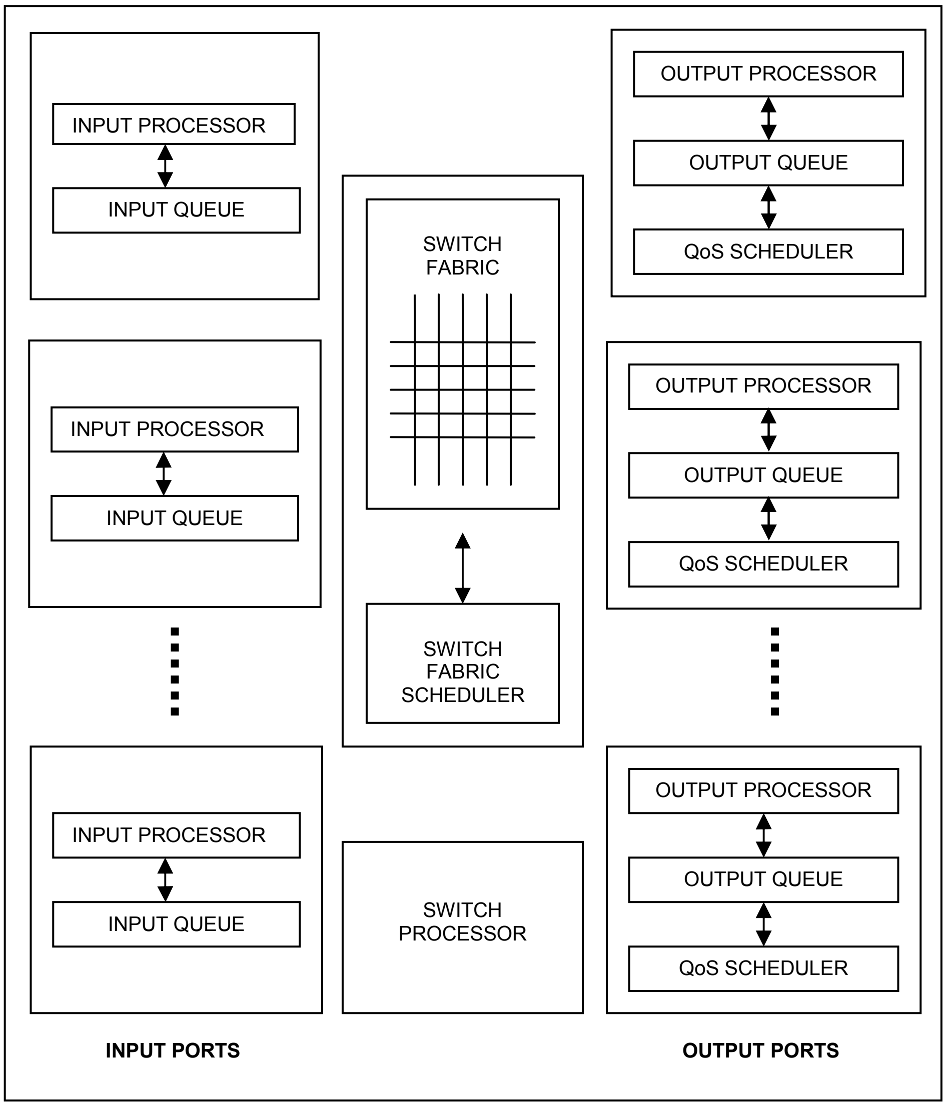


Figure 1.1 A generic high-speed router architecture.

Switch fabrics work with various restrictions and advantages depending on their implementation approaches. As [3] states, a crossbar fabric provides multiple simultaneous data paths between input and output ports, but it requires a centralized scheduler to decide the matching configuration of the ports and turn on and off cross-points to establish the data paths. Various switch fabric scheduling algorithms have been proposed which aim to find the maximum number of input-output port matches to obtain high throughput. Another method of increasing the throughput of the switch fabric is to increase its operation speed so that an output port can be connected to more than one input ports and can receive more than one cell in one slot time one after the other. The maximum number of cells that can be switched to an output port in a single slot is defined to be the *speed-up factor*. Another definition, made by [3], for the speed-up is the operating clock rate ratio of that at the switching elements core to that over external links.

When the speed-up factor is 1, the switch fabric operates at the line rate and only one cell can be switched to an output port in one slot. Since the output port can send one cell out to the transmission lines in a slot, all arriving cells are immediately transmitted from the output port. Thus, the output ports do not need to have any buffers to store the packets. If more than one input ports have packets that are to be switched to the same output port in the same slot, only one of them can be connected to the output port in that slot when the speed-up is 1. The other input port has to store its packet at least until the next slot. Hence, the input ports must have buffers (queues) to store the cells. As a result, the switch is called an *Input Queue* (IQ) *switch*, when the speed-up is 1.

In a switch with N input ports and N output ports (NxN switch), all N input ports can have packet arrivals destined for the same output port in the same slot. If the speed-up factor is N, each of the N input ports can be connected to the same output port in the same slot one after the other and all packets can be immediately switched. Since only one cell can arrive at an input port in one slot and it can be immediately switched to the corresponding output port, there is no need to have buffers at the input ports. This time, the output ports must have queues because N cells can be switched to an output port in one slot while only one of them can be transmitted out of the node in one slot. Therefore, the switch is called an *Output Queue* (OQ) switch when the speed-up is N. OQ switches can provide 100% throughput. On the other hand, OQ is impractical since it requires that the switch fabric and memory operate as fast as N times the line rate for an NxN switch [4].

When the speed-up factor is greater than 1 but less than N in an NxN switch, there can be output contentions and not all input ports can switch their packets immediately to the corresponding output ports since the speed-up is less than N. Thus, there must be buffers in the input ports to store the packets. Since the speed-up is greater than 1, more than one cells can be switched to an output port in the same slot. Hence, the output ports must also have buffers. Therefore, the switch is called a *Combined Input Output Queue* (CIOQ) switch when the speed-up factor is greater than 1 but less than N.

When the buffers in the input ports are first-in-first-out (FIFO) queues, only the cell which is at the head of the queue can be switched to its output port. Even though the output ports which the second or later cells in the FIFO queue of an input port are destined for are available to match with this input port, they cannot be connected to this input port because those cells cannot get out of the FIFO queue before the first cell leaves the queue. The cell at the head of the FIFO queue blocks the all other cells behind it. This phenomenon is known as the *head-of-line (HOL) blocking*. It is proven in [5] that the maximum throughput of an NxN IQ switch with FIFO queues is limited to 58.6% for large *N*. To solve the HOL blocking problem, the cells in an input port destined for distinct output ports are stored in separate queues. Thus, there is a separate queue for each output port in each input port of a switch. Each of these separate queues is called a *virtual output queue* (VOQ).

When the speed-up *Su* in a switch is greater than 1, the cells from input ports can arrive at the output ports *Su* times the line rate of the outgoing transmission lines. In other words, an output port can receive *Su* distinct cells in a slot whereas it can only send one of them in the same slot. Hence, a maximum of $Su - 1$ cells can accumulate at an output port in each slot. These accumulating cells are stored in the buffers at the output ports. If an output port has a single FIFO queue to store these packets, the output port can only send the cell at the head of the queue in a slot and there can be no service discrimination among the cells. Real-time traffic, for example, cannot be supported in a switch with such output ports because a real-time packet which is located at the back of the queue may not be sent within its delay bound due to the packets of other flows. On the other hand, if an output port possesses different queues for different traffic classes or for each individual connection, the next cell to be sent can be decided according to requirements of the traffic flows and the flows can be provided with a certain service quality. QoS schedulers determine the service order and time of the packets at the output ports and manage the available resources. Thus, they can improve the average performance of the traffic or provide service guarantees for certain traffic flows depending on their operation principles.

2.2 SWITCH FABRIC SCHEDULERS

There are various types of switch fabric architectures such as memory, bus and interconnection networks. In a crossbar interconnection fabric, each output port connects to one input port at a time and the input ports have to contend for the output ports if the speed-up less than *N* for an *N*x*N* switch. If the input and output ports are connected in a proper configuration, the throughput of the fabric can approach 100%. An a*rbiter* or *fabric scheduler* is used to manage this contention of the input ports and schedule the fabric for an appropriate switching configuration. A fabric scheduling algorithm aims to find the input-output port matching configuration that leads to the switching of the data packets from the input ports to the output ports in the most efficient way regarding metrics such as throughput, queuing delay and fairness. Since the speed-up is less than *N* in IQ and CIOQ switches, they need to employ a fabric scheduler for the arbitration of input ports. When the speed-up is equal to *N*, all arriving cells can be switched to their output ports immediately and there is no need for fabric schedulers in OQ switches.

A *maximum matching* is a matching of pairs that has the maximum number of matched pairs among all possible matching configurations. A *maximal matching* is a matching to which no more new pairs can be added without disturbing the previously established matches. [6] shows that an IQ switch with VOQs can achieve 100% throughput by using a maximum matching algorithm as its fabric scheduling algorithm. However, [7] states that the complexity of the best known maximum matching algorithm is $O(N^{2.5})$ for an *N*x*N* switch [8]. Therefore, many maximal matching algorithms have been proposed to achieve high throughput such as PIM [9], iSLIP [10], DRR [11, 12] and LPF [13]. PIM is a randomized iterative matching algorithm whereas iSLIP and DRR employ round-robin pointers to determine the priority of the ports. PIM and iSLIP find a maximal matching in $O(\log N)$ iterations. Many other fabric scheduling algorithms are proposed that add enhancements to these algorithms in order to decrease complexity, improve fairness or decrease maximum input queuing delay such as SRA [14], WPIM [15], iDRR/iPDRR [16] and iTFF [17,18]. There are also some fabric scheduling algorithms that try to emulate output queuing in input queue switches but they require communications and computations of high complexity.

"In input-queued scheduling, iSLIP algorithm is a milestone for its high throughput and low implementation complexity" [17]. iSLIP provides a very important basis for other scheduling algorithms that assign priorities to ports for arbitration since it is open for various enhancements. iSLIP can also be applied in CIOQ switches in which case it can provide higher throughput due to the increased fabric speed-up.

2.2.1 iSLIP SCHEDULING ALGORITHM

The iSLIP algorithm assigns priorities to each input and output port in a round-robin fashion. Each input port sends a request to each output port which it has a cell destined for in its VOQs. Each output port chooses the input port with the highest priority among the input ports from which it receives a request and grants it. Then, each input port selects the output port with the highest priority among the output ports from which it receives a grant and accepts it. The input ports are marked to be matched with the output ports which they accept at the end of the iSLIP cycle.

iSLIP is an iterative algorithm. The above procedure of iSLIP can be repeated more than once in a cycle with only unmatched input and unmatched output ports participating in the next iteration. When iSLIP operates with multiple iterations, each new iteration attempts to add new matches not made by the previous iterations and the performance improves as the number of iterations increases [10].

The cells being scheduled by round-robin arbiters at each input port and at each output port, iSLIP overcomes two problems present in schedulers operating with randomization. First, round-robin arbiters are faster and simpler to implement than random arbiters. Second, the round-robin arbiters provide fairness since they guarantee that each port is given the highest priority in a round-robin fashion.

Each input port maintains a pointer that indicates the output port to which this input port gives the highest priority in the current cycle. This pointer is called an *accept pointer* and is denoted by a_i for the i^{th} input port.

Similarly, each output port maintains a pointer that indicates the input port to which this output port gives the highest priority in the current. This pointer is called a *grant pointer* and is denoted by g_j for the j^{th} output port.

iSLIP algorithm consists of three steps:

1) Each unmatched input port sends a request to each output port for which it has a cell in its VOQs.
2) Each unmatched output port chooses the input port with the highest priority among the input ports from which it receives a request. The output port traces the fixed round-robin order of input ports starting from the input port indicated by g_j and grants it if there is a request from this input port. If there is no request from this input port, the

output port continues tracing the fixed, round-robin schedule of input ports until it finds an input port from which there is a request or reaches the input port indicated by g_j. If it finds an input port that has a request, the output port grants it.

3) If an input port receives a grant, the input port traces the fixed round-robin order of output ports starting from the output port indicated by a_i and accepts it if there is a grant from this input port. Otherwise, the input port continues tracing the fixed, round-robin schedule of output ports until it finds an output port from which there is a grant or reaches the output port indicated by a_i. If it finds an output port that has a grant, the input port accepts it. If this is the first iteration in the current cycle, then the input port increments its accept pointer a_i modulo N to one location beyond the accepted output and the output port that receives an accept increments its grant pointer g_j modulo N to one location beyond the accepting input port.

At the end of these three steps, the input ports are matched with the output ports to which they send an accept signal in step three. The unmatched input and output ports can participate in the next iteration. When all iterations are complete, the switch fabric is configured according to the obtained input-output matches so that every matched input port is connected to its matched output port and the cells are switched to the corresponding output ports.

With the mentioned updating method of the accept and the grant pointers, lowest priority is given to the most recently made connection which protects the connections from starvation. Since an input port continues to request an output port until they are matched and the output can serve at most $N - 1$ other input ports first, waiting at most N cycles to be accepted by each input, a requesting input is always served in less than N^2 cycles [10].

iSLIP requires control lines between the input ports and the output ports for the transmission of request, accept and grant signals. Each input port can send a request signal to each output port, so there must be one control line between each input port and output port pair summing up to *N* lines for each input port in an *N*x*N* switch. Since there are *N* input ports, a total of N^2 control lines are required between all input and output ports in the switch. The same control lines are used for request, grant and accept signal in the respective steps of iSLIP.

2.3 QoS SCHEDULERS

The QoS schedulers that support guaranteed-service traffic require the traffic arrivals to be in a certain shape in order to provide delay bounds. Therefore, they operate with specific traffic models. Three of the common traffic models are the (X_{min}, X_{ave}, I, S_{max}) model [19], (σ, ρ) traffic model [20] and the (r, T) traffic model [21].

According to the (X_{min}, X_{ave}, I, S_{max}) traffic model, the interarrival time between the packets must not be less than X_{min}, the average packet interarrival time during any time interval of length I must be greater than X_{ave} and the maximum packet size must be less than S_{max}.

The (σ, ρ) traffic model which is also called leaky bucket model requires that during any time interval of length τ the number of bits transmitted must be less than ($\sigma + \rho\tau$) where σ corresponds to the maximum burst size and ρ to the average rate.

In the (r, T) traffic model, traffic flows are not allowed to transmit more than rT bits in any time interval of length T where r corresponds to the average rate.

In one perspective, the QoS schedulers are classified into two groups: work-conserving and non-work-conserving [22]. According to the work-conserving schedulers, an output port never stays idle if there is a packet to send. However, non-work-conserving schedulers put an eligibility constraint for each packet and let the output port send only eligible packets. Packets that are not yet eligible are not scheduled until they become eligible. Therefore, non-work-conserving schedulers shape the traffic at each output port so that they can provide strict end-to-end delay bounds with more controlled jitter and prevent accumulation of packets in the downstream switches which decreases the required buffers.

According to another perspective, the QoS schedulers are categorized into two as sorted-priority and frame-based [23]. Sorted-priority schedulers assign a timestamp for each packet and sort the packets according to their timestamps. Sorted-priority schedulers can provide tight delay bounds but timestamp computation can be very complex to be performed at the line rate. Frame-based schedulers divide the time into intervals called frame and limit the amount of traffic that can be served in each frame. They can provide delay bounds but may introduce extra delay for fitting the incoming traffic into certain frames.

Next, we present some service disciplines that provide service guarantees to certain traffic and the traffic shapes they operate with.

Weighted Fair Queuing (WFQ) and Self-Clocked Fair Queuing (SCFQ) are sorted-priority schedulers and they try to emulate the hypothetical generalized processor sharing (GPS) method which serves infinitesimal amount of data out of each queue in each turn. They operate with the *(σ, ρ)* traffic model. They both compute the time a packet would complete its departure if it were served with GPS but SCFQ uses some approximations. The complexity of the WFQ grows linearly with the number of flows through the output port and is $O(V)$ where V is the number of flows through the output port. The complexity of SCFQ grows logarithmically with the number of flows and is $O(\log V)$ where V is again the number of flows. They both provide a delay bound that depends on the allocated rate of the flow.

Rate Controlled Static Priority (RCSP) [24] is also a sorted-priority scheduler which operates with the *(σ, ρ)* traffic model. RCSP maintains a regulator for each connection through the switch and shapes the traffic. It computes an eligibility time for each packet. Its complexity is $O(n)$ where n is the number of offered link delay bounds.

Delay-Earliest Due-Date (D-EDD) is also a sorted-priority scheduler. It assigns an expected arrival time to every packet of each connection and computes a deadline according to the expected arrival time for each packet. The complexity of the D-EDD grows linearly with the number of connections through the output port and is $O(\log V)$ where V is the number of connections through the output port. It operates with the (X_{min}, X_{ave}, I, S_{max}) tarffic model and provides a delay bound depending on the number of hops on the end-to-end path of each connection.

Stop and Go (S&G) is a frame-based scheduler and operates with the (r, T) traffic model. S&G divides time into fixed length frames and the packets that arrive in different frames are sent in different frames in order not to allow the packets to close up and form bursts. It has a constant complexity of $O(1)$ and provides an end-to-end delay bound of $2HT$ where H is the number of nodes on the end-to-end path and T is the frame length. S&G works on fixed-size packets.

Hierarchical Round Robin (HRR) is another frame-based scheduler that operates with the (r, T) traffic model. HRR works on fixed-size packets and serves a limited number of packets for each connection per frame visiting each connection in a round-robin order. Similar to S&G, HRR has a constant complexity of $O(1)$ and provides an end-to-end delay bound of $2HT$ where H is the number of nodes on the end-to-end path and T is the frame length.

Budgeted-Weighted-Round-Robin (BWRR) is also a frame-based scheduler. BWRR works on fixed-size packets with periodic arrivals. It does not require the synchronization of frames. BWRR has a constant complexity of $O(1)$ and provides an end-to-end delay bound of $2HT$ where H is the number of hops on the end-to-end path and T is the frame length.

Framed-Deadline Scheduler (FDS) is a both sorted-priority and frame-based scheduler. FDS works on fixed-size packets. It operates with a traffic model which limits the number of packets per frame but allows the packets to arrive at any time within a frame without the need of periodicity. This traffic model is equivalent to the (r, T) traffic model with fixed frame origins. FDS has a constant complexity of $O(1)$ and provides an end-to-end delay bound of $(H+1)T$ where H is the number of hops on the end-to-end path and T is the frame length.

The above mentioned frame-based schedulers all have better complexity values than the sorted-priority schedulers above have. The one that works with the lowest complexity among these sorted-priority schedulers has a logarithmic scheduling complexity while the frame-based schedulers have constant scheduling complexity. These framed-based schedulers also provide delay bounds that do not depend on the allocated rate. On the other hand, all these framed-based schedulers work on fixed-size packets.

CHAPTER 3

DELAY-LIMITER SWITCH

This chapter describes the delay-limiter switch with the framed-deadline scheduler and its operation. The whole content of this section is gathered from the original works [1] and [2]. The information presented in this chapter summarizes the original work of the delay-limiter switch in [1] and [2] and aims to give a general background on the delay-limiter switch.

Employing combined input output queued (CIOQ) switches with lower fabric speed-ups than N (N is the number of ports) is more economical than using switch fabrics with pure output queuing at high data rates. Relying on this fact, the delay-limiter switch is a combined input-output queuing switch that works with any fabric speed-up $Su\geq 2$. The delay-limiter switch uses a deadline-based QoS scheduler which is called *framed-deadline scheduler* (FDS).

Since the delay-limiter switch is a CIOQ switch, arriving packets may not be switched to the output ports immediately. The arriving packets may have to be buffered at the input ports until they can be switched to their output ports. After arriving at the output ports, the packets are stored in per connection queues at the output ports. The packets are scheduled and sent out from the switch in an order according to the QoS algorithm at the output ports.

The delay-limiter switch works in a network in which the packets have fixed size and the packet times are synchronized. The fixed-size packets are called *sub-frames*. Because the packet size is fixed throughout the network, the time required to transmit one packet at any node is constant and it is referred to as a *slot*. Slot is the basic time unit in the operation of the delay-limiter switch. There is also a second fixed and larger time unit used by all nodes in the network. This larger time unit is called a *frame* and is utilized for scheduling and traffic shaping purposes in the network. The duration of a frame in any node is T slots where T also denotes the number of sub-frames per frame.

The traffic through the network is categorized into two: *guaranteed-service traffic* and *best-effort traffic*.

The guaranteed-service traffic is always carried in a connection-oriented fashion. Guaranteed-service connections are provided with guaranteed bandwidth and end-to-end delay bounds. When a new connection is to be established, the connection admission control checks the state of the network. If the resources on all of the links on the end-to-end path of the connection are sufficient to provide the promised guarantees, then the new guaranteed-service connection can be established.

The best-effort traffic can be carried in connection-oriented or connectionless fashion. Best-effort connection-oriented traffic and best-effort connectionless traffic utilize the remaining bandwidth from the guaranteed-service connections. As the name implies, the best-effort traffic is not provided by guaranteed bandwidth or end-to-end delay bounds.

3.1 TRAFFIC CHARACTERISTICS

When a connection is to be established on an end-to-end path, the connection reserves a portion of the total link bandwidth based on its peak rate. The allocated rate for a connection is expressed with the unit of *slots per frame*.

Let connection A with the allocated rate of B_A bits per second (bps) be established on an end-to-end path in which the link rate is C bps. Then, A has an allocated rate of ρ_A sub-frames/frame computed by taking the ceiling of the multiplication of the frame time by the ratio of the allocated rate for A to the link rate and hence,

$$\rho_A = \left\lceil T\frac{B_A}{C} \right\rceil \qquad \text{(Equation 3.1)}$$

The connection admission control ensures that the line rates are not oversubscribed by the connections. Let K be the set of connections established on link l and let $k \in K$ be a connection at the rate of ρ_k sub-frames/frame. The connection admission control ensures that the following condition holds for all links l:

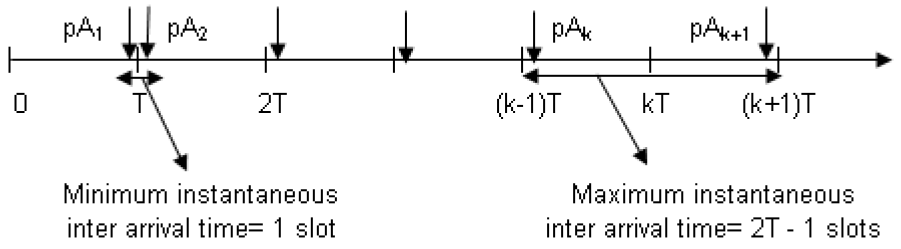
$$\sum_{k \in K} \rho_k \leq T \qquad \text{(Equation 3.2)}$$

The guaranteed-service connections reserve link rates by reserving a certain number of sub-frames over a frame. The symbol ρ_k is used to denote the number of sub-frames allocated for the k^{th} connection per frame. The guaranteed-service connections can have rates starting from ρ=1 sub-frame per frame up to the maximum rate of ρ=T sub-frames per frame. Thus, the network is a *multi-rate network*.

The *unit rate* is defined to be ρ_0 which is one sub-frame per frame. A *base-rate connection* has a rate that is an integer multiple of ρ_0 and a *unit connection* is a connection that has the unit rate. All connections established on a link connecting two nodes can be expressed as the superposition of unit connections. Let A be a guaranteed-service connection at the rate of ρ_A sub-frames/frame. Connection A can be decomposed into ρ_A unit connections. The network operates as a *unit-rate network* when all connections are decomposed into unit connections. All connections are assumed to be decomposed into unit connections unless stated otherwise.

The framed-deadline scheduler in the delay-limiter switch works with a certain traffic shape to provide end-to-end QoS guarantees. The guaranteed-service traffic is shaped at the network ingress which ensures that the connections do not exceed their allocated rates and the packet arrivals exhibit a certain behavior. The required incoming traffic shape for the end-to-end QoS support is as follows:

Let connection A be a unit-rate connection. Suppose that the time origin is set arbitrarily to 0 and periodic arrival frames of T slots are defined for connection A (Figure 3.1, with the time origin set arbitrarily to 0) Let pA_k denote the k^{th} sub-frame arriving from connection A. Suppose that the first sub-frame of connection A, pA_1, completes its arrival at the network before slot T. The second sub-frame completes its arrival after and including slot T and before $2T$. Then, the k^{th} sub-frame, pA_k, completes its arrival after (and including) slot $(k - 1)T$ and before slot kT.


pA_1
pA_2
pA_k
pA_{k+1}
0
T
2T
(k-1)T
kT
(k+1)T
Minimum instantaneous inter arrival time= 1 slot
Maximum instantaneous inter arrival time= 2T - 1 slots
Figure 3.1 Regulated traffic arrival shape.

The sub-frames for a connection do not have to arrive periodically. There is no constraint on the minimum interarrival time between consecutive sub-frames. Thus, they can arrive any time within a frame and this makes the traffic shape more flexible than the periodic traffic shapes. According to this traffic shape, two sub-frames can arrive back-to-back with an interarrival time of 1 slot or there can be two sub-frame arrivals with an interarrival time of $(2T - 1)$ slots (see Figure 3.1, maximum and minimum instantaneous interarrival times). This traffic shaper requires fewer buffers and introduces less delay than a traffic shaper whose output is periodic traffic.

3.2 CONDITION FOR END-TO-END DELAY BOUND

The end-to-end delay of a sub-frame is defined to be the total time the sub-frame spends in all of the nodes on its path to the destination. When a sub-frame leaves a node within the same slot in which it arrives at that node, the sub-frame experiences 0 delay in that node. Lemma 3.1 below which is proved in [1] gives a condition for the end-to-end delay bound of the sub-frames.

Lemma 3.1. Let A be a unit-rate connection established on an H-hop path. Let the k^{th} sub-frame of connection A, pA_k, arrive at the network after $(k - 1)T$ and before kT complying with the traffic shape described in Section 3.1. Let SW_h be the h^{th} switch on connection A's path, where (h = 1, 2, ..., H).

Condition: Assume that each switch SW_h can guarantee that if the k^{th} sub-frame arrives before $(h + k - 1)T$ then it is served before $(h + k)T$. Then on an H-hop path, the end-to-end delay bound for connection A is $(H + 1)T$.

Lemma 3.1 considers the arrival and the departure times of sub-frames within specified time intervals. The exact arrival and departure instants affect the time that pA_k spends at node SW_h. The (+1) term in the $(H + 1)T$ expression is a result of the relative displacement of the sub-frame in the arrival and departure frames.

Figure 3.2 demonstrates the sub-frame arrival and departure times that comply with Lemma 3.1. pA_1, pA_2, and pA_3 arrive at node SW_1 according to the traffic shape defined above. pA_2 arrives at node SW_1 after time T and before time $2T$. pA_2 leaves the node at an instant after $2T$ and before $3T$. Hence, pA_2 complies with Lemma 3.1 although it is delayed more than T slots at node $SW1$. pA_1, and pA_3 leave the node within a period of T slots after their arrivals.

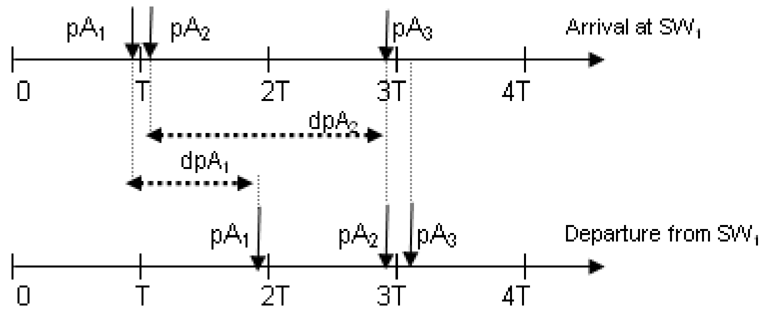


Figure 3.2 Node delays and traffic shape

3.3 OPERATION

There is a separate buffer for each established guaranteed-service connection at the input ports of the delay-limiter switch. The guaranteed-service connection sub-frames arriving at the delay-limiter switch are first stored in these input buffers. Then, the guaranteed-service sub-frames are switched to the corresponding output ports through the switch fabric which is configured according to the scheduler that uses the concept of time slot assignment (TSA) from TDM switches. The interval from the time a sub-frame arrives at its input port to the time it is switched to its output port is called *input delay*.

Similar to the input ports, there is a separate buffer for each established guaranteed-service connection at the output ports. The sub-frames that are switched to the output ports are stored in these buffers until they are transmitted out from the node to the transmission lines. At each output port, the next sub-frame to be served is chosen by the framed-deadline scheduler and sent out from the node.

3.4 SWITCH FABRIC SCHEDULER

The fabric schedule of a node is computed when a new connection is to be established through the node and the same schedule is used until another connection is to be established. The schedule is computed by using the reservation information of the new connections and any communication between input ports and output ports is not required.

Traffic Matrix (*D*) shows the traffic load for the guaranteed-service connections between the input and output port pairs in a switch. *D* is an *N*x*N* matrix for a switch with N input ports and N output ports. The element d_{ij} of D is equal to the number of reserved sub-frames per frame to be switched from input i to output j in a node. The traffic matrix D is computed and updated whenever a change in the state of any guaranteed-service connection occurs, that is when either a new connection is to be set up or an existing one is to be terminated.

The traffic matrix D can be expressed as follows:

$$D = \sum_{k=1}^{L} Tx_k \qquad \text{(Equation 3.3)}$$

where, L is the frame length and Tx_k is the *Transmission Matrix* (Tx_k) which shows the configuration of the switch fabric in a specific slot. The elements of Tx_k, t^k_{ij} are 1 if input port i is connected to output port j, 0 otherwise. Since an input (output) port can be connected to a single output (input) port at a time instant, there cannot be more than one 1 entry in a column or in a row. Hence, the maximum number of 1's in the transmission matrix is N for an NxN switch.

Any proper TSA can be used to compute the T transmission matrices from the traffic matrix. [25] states that the complexity of optimal TSA computation for an NxN switch is $O(N^2 \log N)$ but the computation of transmission matrices is only done when a change in the state of the connections occurs.

It is shown in [1] that the condition in Lemma 3.1 is satisfied with an input queuing switch with a static connection set-up. If the connection set-up is not static and new connections are established, the switch fabric schedule has to be computed and rearranged when new connections are established. If the fabric speed-up is 1, the same guaranteed-service connection may not have a chance to switch its sub-frames from its input port to its output port for $(2T - 1)$ slots during the transition between the old schedule and the new schedule. In order to reduce this period of $(2T - 1)$ slots below T slots, the fabric speed-up must be greater than or equal to 2. When it is guaranteed that the fabric schedule always scans every connection within a period of T slots, Lemma 3.1 and the end-to-end delay bounds can be satisfied. Therefore, the switch fabric speed-up must be greater than one and the switch must be a CIOQ switch to satisfy the end-to-end delay bounds in any case.

If the switch fabric speed-up (Su) is increased to twice the line speed ($Su = 2$), the longest time interval between two consecutive switching events for the existing connections is decreased to $(T - 1)$ slots. Thus, the switch can guarantee a transmission rate equal to the allocated bandwidth for any connection. If the connection set-up does not change, the worst-case input delay for any connection is $(T/2 - 1)$ because each connection is scanned twice within each frame time of T.

When the switch fabric works with a speed-up of 2, the sub-frames can accumulate at the output ports. As a result, the sub-frames may have to be buffered both at the input ports and at the output ports which requires a CIOQ switch. Hence, a traffic scheduler must be utilized to determine the departure order of the sub-frames from the output ports.

The connections through a switch are scanned in a round-robin (RR) fashion with fabric speed-up of Su and the entire set of T transmission matrices is scanned Su times for each T slot period. The local frame boundaries of a node can be marked according to an arbitrary starting point in the transmission matrix schedule. The frame boundaries are determined in each link independently and they are not necessarily synchronized with each other.

3.5 FRAMED-DEADLINE SCHEDULER

Framed-Deadline Scheduler (FDS) is both a frame-based and a sorted-priority service discipline. The same traffic shape which is allowed into the network is preserved throughout the network. The sub-frames are not allowed to get service until a certain eligibility time in order to preserve the traffic shape and FDS ensures that they are sent out from the node within a frame time of T slots after their eligibility time. Therefore, each sub-frame is assigned a deadline which shows the latest time the corresponding sub-frame can leave the node. The sub-frames are served in the order of their deadlines and FDS does not carry out any sort or search operations to pick the next sub-frame to get service. The execution time complexity of FDS is O(1) which is very low.

3.5.1 OPERATION

When we consider a unit-rate connection, each sub-frame pA_k arriving at a node defines the k^{th} service frame time for connection A denoted by FA_k. The beginning and end of the corresponding service frame is computed when a sub-frame arrives at the node. The arrival time of pA_k is denoted by $tarrA_k$. This computation is carried out on only the arrival time of the current sub-frame and the deadline of the previous sub-frame of the same connection in the current node. The sub-frame can only be served after the beginning of its service frame and must be served before the end of its service frame. Therefore, the beginning instant of FA_k is called the *activation time* and is denoted by $tactA_k$. The ending instant of FA_k is called the *deadline* and is denoted by dlA_k. The activation time and deadline of pA_k are computed as follows:

$$tactA_k = \max(tarrA_k, dlA_{(k-1)}) \qquad \text{(Equation 3.4)}$$

$dlA_k = tactA_k + T$ (Equation 3.5)

As the first slot of FA_k is $tactA_k$ and the last slot of FA_k is $dlA_{(k-1)}$, pA_k gets service within FA_k before dlA_k. Figure 3.3 shows an example traffic arrival sequence with the respective frames for each sub-frame where T is 4 slots. pA_3 which arrives only 3 slots after pA_2 has to wait for one slot until it becomes eligible because the frame length is 4 slots and a sub-frame cannot be eligible before the deadline of the previous sub-frame of the same connection.

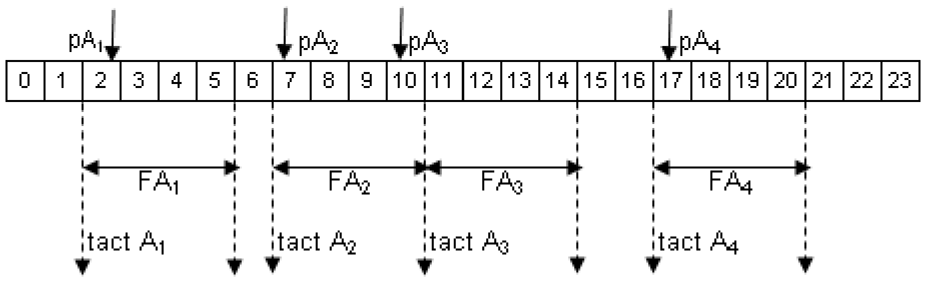


Figure 3.3 Frame times for FDS

FDS serves the eligible sub-frames at the output ports in the order of their deadlines without the need of a search or sort operation. If more than one sub-frames have the same deadline at an output port, any one of them can be served arbitrarily. The same traffic shape is preserved throughout the network due to the fact that a sub-frame cannot be eligible before the deadline of the previous sub-frame of the same connection and two consecutive service frames cannot overlap. The computations for the frame borders in a node require only the local time in that node and hence no time information is needed from other nodes. The consecutive service frames for a connection may not immediately follow each other if there are more than T slots between the arrivals of the corresponding sub-frames

Corollary 3.1: At most T unit-rate connections can be established per output port of a switch. Hence, the total number of sub-frames queued at a switch with N input ports and N output ports does not exceed $2TN$.

Corollary 3.2: The maximum delay a sub-frame has in a node is bounded by $(2T - 1)$.

Note that although the maximum node delay for a sub-frame is $(2T - 1)$, the end-to-end delay bound for FDS is $(H + 1)T$. Thus, a sub-frame cannot have $(2T - 1)$ delay in every node on its end-to-end path.

3.5.2 EFFECT OF RATE DECOMPOSITION ON DEADLINES

Suppose that connection A has allocated rate of $\rho_A = m$ such that $1 < m \leq T$. Connection A can be decomposed into ρ_A unit-rate components as $A_1, A_2, \ldots, A_m$. In each switch, the consecutive sub-frames that arrive from connection A are logically distributed to the components one by one and treated as sub-frames from distinct unit-rate connections.

The previously defined traffic arrival shape can be extended to multi-rate connections such that the sub-frames of connection A that belong to the each component A_i, i =1... m, comply with the traffic shape individually. The first sub-frame to arrive for this component A_i is pA_{1i}, the second sub-frame is pA_{2i}, and k^{th} sub-frame is pA_{ki}. In the extended traffic shape, suppose that pA_{1i}, completes its arrival at the network before slot T. pA_{2i} completes its arrival after and including T and before $2T$. Then, the k^{th} sub-frame, pA_{ki}, completes its arrival after and including slot $(k-1)T$ and before slot kT.

The components of connection A have their own state variables which are updated individually. The computation of the sub-frame frame borders is also performed per component. The generalized expressions for deadline and activation time for pA_k are as follows:

$$tactA_k = \max(tarrA_k, dlA_{(k-m)}) \quad \text{(Equation 3.6)}$$

$$\begin{aligned} dlA_k &= tactA_k + T \\ &= \max(tarrA_k, dlA_{(k-m)}) + T \end{aligned} \quad \text{(Equation 3.7)}$$

Note that, Equation 3.6 and Equation 3.7 become identical to Equation 3.4 and Equation 3.5 if $\rho_A = 1$

3.5.3 QoS GUARANTEES PROVIDED BY FDS

The switch fabric must work with a speed-up of 2 or more and the rearrangement in the traffic matrix can only occur once within a frame time so that FDS can provide its QoS guarantees. In addition, the traffic arrivals must obey the pre-described traffic shape. With these conditions, FDS provides the following QoS guarantees on an H-hop path:

(1) The end-to-end delay bound for any sub-frame is $(H+1)T$ where T is the frame duration.
(2) For an NxN switch the total number of sub-frames queued in the switch does not exceed $2TN$.

CHAPTER 4

SUPPORTING BEST-EFFORT TRAFFIC IN THE DELAY-LIMITER SWITCH

[2] states that best-effort traffic can be carried at low priority with the guaranteed-service traffic in connection oriented or connectionless fashion in the delay-limiter switch. However, neither [1] nor [2] propose any solutions that can be applied to switch the best-effort t traffic efficiently by using the remaining resources after the guaranteed-service traffic. They do not define any QoS schedulers for best-effort traffic or any modifications on FDS to improve the best-effort traffic performance. This chapter studies how to support connectionless best-effort traffic in the delay-limiter switch.

4.1 INPUT AND OUTPUT PORT QUEUES

The delay-limiter switch possesses a separate input port queue and a separate output port queue for each guaranteed-service connection established. Similarly, there is a VOQ for each input/output port pair at each input port to buffer the connectionless best-effort traffic packets. The output ports have only a single FIFO queue to store the connectionless best-effort traffic packets.

4.2 SWITCH FABRIC SCHEDULER

The delay-limiter switch employs a static switch fabric scheduling method for the guaranteed-service traffic. In this method, the switch fabric is configured according to a predetermined traffic matrix which is computed only when a new guaranteed-service connection is to be set up or an existing guaranteed-service connection terminates.

Static switch fabric scheduling can also be applied for connection-oriented best-effort traffic. When a new connection for best-effort traffic is to be established, the traffic matrix can be recomputed just like it is done when a new connection for guaranteed-service traffic is established. Since the input and output ports of a best-effort connection through each node are declared during the connection set-up procedure, the traffic matrix can be computed in such a way that the fabric schedule can satisfy the demands of the connection-oriented best-effort traffic as long as the those ports are not overloaded by other best-effort connections.

On the other hand, connectionless best-effort traffic does not reserve any resources before-hand. The data streams can behave unexpectedly and they do not promise a certain input traffic shape into the network. There can be sudden bursts or long silence periods from time to time. In addition to these, some ports may be overloaded by connectionless best-effort traffic while the request for other ports may be very low in different switches. Therefore, static switch fabric scheduling may not satisfy the connectionless best-effort traffic demands very efficiently.

In order for the delay-limiter switch to offer a better throughput and delay performance to the connectionless best-effort traffic, we integrate a dynamic fabric scheduler alongside the static fabric scheduler used for the connection-oriented traffic. Any high performance dynamic fabric scheduler that operates fast enough can be adopted. We choose iSLIP as the dynamic scheduler since iSLIP can provide high throughput and fairness [10]. Another important reason is the requirement of iSLIP for having a control line between each input port and each output port which overlaps with the same demand of the QoS scheduler for best-effort traffic used in the delay-limiter switch. iSLIP adapts to a fair scheduling policy under non-uniform traffic and prevents starvation of the input queues. [10] states that prototype and commercial implementations of iSLIP exist in systems with aggregate bandwidths ranging from 50 to 500 Gb/s.

iSLIP and the static fabric scheduler described in [1] are used together to support connection-oriented guaranteed-service and connectionless best-effort traffic in the delay-limiter switch simultaneously. At the beginning of an arbitration cycle, the input/output port pairs that are to be connected according to the transmission matrices computed upon the guaranteed-service traffic reservations are scanned. If an input port has a guaranteed-service traffic sub-frame destined to an output port to which this input port is to be connected according to the current transmission matrix, this input-output port pair is marked as matched. If an input-output port pair indicated by the current transmission matrix does not have a guaranteed-service traffic sub-frame in the input queue at this moment, then these input and output ports are marked as unmatched. After this step, the unmatched input ports that have at least one best-effort traffic sub-frame in their buffers and the unmatched output ports participate in the dynamic fabric arbitration process, iSLIP in our case. Finally, when the dynamic scheduling is completed, the input and output port pairs which are matched either statically or dynamically are connected through the switch fabric and the corresponding sub-frames are switched from the input ports to the output ports.

4.3 QoS SCHEDULER

The sub-frames belonging to guaranteed-service traffic must have precedence over the best-effort traffic sub-frames at the output ports in order to hold their delay guarantees. According to this scheme the best-effort traffic uses the remaining resources after the guaranteed-service traffic. The simulation results in [2] show that at 100% background connection utilization, the mean end-to-end delay observed when FDS is used with a CIOQ switch of speed-up of 2 is less than 60% of the theoretical bound. This fact tells us that the guaranteed-service traffic sub-frames are mostly served within the first half of the maximum allowable time that they can spend in a switch to satisfy the end-to-end delay bounds. When the best-effort traffic uses the resources after the guaranteed-service traffic in a rudimentary fashion, an output port sends best-effort traffic sub-frames only if there is no eligible guaranteed-service traffic sub-frames in its per connection queues. From here on, we refer to this kind of operation of the QoS scheduler as *rudimentary method*.

On the other hand, since the sub-frame size is fixed in the system, modifying only the QoS scheduler for the best-effort traffic to interchange the service order of the best-effort traffic sub-frames among themselves would not improve the average delay performance of the best-effort traffic. The reason is that changing the service order of the packets within a group does not reduce the average delay of the group when all packets in the group are of the same size. Hence, the average best-effort traffic delay can be decreased by interchanging the service order of both guaranteed and best-effort traffic packets. The best-effort traffic packets may have precedence over the guaranteed-service traffic packets in some cases so that the switches serve the guaranteed-service traffic packets near the end of their local delay bounds but still not violating the local delay guarantees. Such a method increases the average delay of the guaranteed-service traffic but can still satisfy the promised guarantees. Indeed, increasing the delay of the guaranteed-service traffic within the delay bounds may have benefits. [22] states that packets arriving too early may not be desirable in guaranteed-service class, because the earlier a packet arrives before its delay bound, the longer it may need to occupy the buffer in the destination.

4.4 TOTAL PACKET COUNT ALGORITHM

During the connection set-up process, all computations are carried out on the worst case assumptions and peak rates of the guaranteed-service traffic and the resources are reserved accordingly. Even under the worst case situations, the delay-limiter switch can provide its service guarantees for the guaranteed-service traffic. The simulations on the delay-limiter switch with original FDS in [2] show that the average end-to-end delay for the guaranteed-service traffic is around 40% of the maximum delay bound. If the average end-to-end delay for the guaranteed-service traffic is shifted near the maximum delay bound, the average end-to-end delay of the best-effort traffic can be improved while still preserving the guarantees for the guaranteed-service traffic. The *Total Packet Count* (TPC) Algorithm applies this idea to improve the best-effort traffic performance.

The Total Packet Count Algorithm delays the guaranteed-service traffic sub-frames as much as possible if needed to decrease the average delay and increase throughput of the best-effort sub-frames. Since the guaranteed-service traffic arrivals fit into a specific traffic shape, the busiest arrival pattern that can happen in the next T (T being the number of slots in a frame) slots is limited. This busiest pattern occurs when all T connections send two sub-frames back-to-back and a total of $2T$ sub-frames arrive at the node within a frame time. Suppose that a guaranteed-service packet is served before its deadline. As described in Section 3.5.1, the FDS scheduler at the output ports of the delay-limiter switch serves the next packet of the same connection within T slots after the deadline of the former packet. It is also a known fact that one output port can serve at most T number of packets in T slots. These imply that if there are more than T packets in a node at any time, only T of them must and will be served in the next T slots and the rest of the packets cannot and will not be served within the next T slots starting from the current slot. In other words, a maximum number of T packets destined for the same output port can be eligible at any time instant in a node. Therefore, if there are a total number of T eligible guaranteed-service traffic packets destined for the same output port in a node either in the input queues or in the output queues, then the output port has to serve one of them in the current slot in order to be able to serve all T packets within the next T slots.

Lemma 4.1: Suppose there are a total of N_{total} eligible guaranteed-service packets destined for the same output port both in the input port queues or in the output port per connection queues in a node at any time instant where $N_{total} < T$, T being the number of slots in a frame. Suppose also that none of these packets have a deadline equal to the current slot.

Condition 4.1: If N_{total} is less than the number of slots between the current time tc and the nearest deadline, dl_{min}, of these packets, then this output port does not have to serve any guaranteed-service traffic packets in the current slot and can serve a best-effort traffic packet. The output port can send a best-effort traffic packet as long as the following inequality holds:

$N_{total} < dl_{min} - tc$ (Equation 4.1)

Proof:

Let N_i denote the total number of eligible guaranteed-service packets present in the switch in the i^{th} slot. Suppose there are more than N_{total} best-effort packets and a total of N_{total} eligible guaranteed-service packets destined for the same output port in the switch at the current time tc and only one of them is at the output port while $N_{total} - 1$ of them are still at their input ports. Also suppose that all packets in the switch have the same deadline dl_{min} which is equal to $dl_{min} = tc + N_{total} + 1$ which is the worst case as dl_{min} cannot not decrease with the arrival of any of the eligible packets at the output port. Also, any packet that arrives after tc cannot have a smaller deadline than dl_{min} by Equation 3.4 and Equation 3.5.

Since there is only one packet at the output port, this packet is taken as the eligible packet with the nearest deadline. Let this packet be denoted by P_{min} and its deadline by dl_{min}. Since $N_i = N_{total} \leq dl_{min} - tc$, P_{min} is not served in the current slot i and a best-effort packet is served by the output port.

Suppose in each of the following slots, two or more eligible guaranteed-service packets are switched to this output port as the speed-up Su is greater than or equal to 2. Since all packets in the switch have the same deadline, dl_{min} does not change. If the time when the first guaranteed-service packet is served is denoted by ts, ts trivially satisfies the following equation:

$$N_j = N_{total} = dl_{min} - ts \qquad \text{(Equation 4.2)}$$

Since $ts = j$ in this slot:

$$N_j = N_{total} = dl_{min} - j \qquad \text{(Equation 4.3)}$$

In this slot, $tc = ts = j$ and the output port sends one guaranteed-service packet. Therefore in the next slot ($j + 1$), dl_{min} does not change and

$N_{(j+1)} = N_{total} - 1$ (Equation 4.4)

$tc = j + 1$ (Equation 4.5)

when we substitute Equation 4.4 and Equation 4.5 into Equation 4.3 we obtain,

$N_{total} - 1 = dl_{min} - (j + 1)$ (Equation 4.6)

which yields

$N_{total} - 1 = dl_{min} - j - 1$ (Equation 4.7)

$N_{total} = dl_{min} - j$ (Equation 4.8)

Since Equation 4.8 which represents the state in the $(j+1)^{th}$ slot does not satisfy the inequality in Equation 4.1, the output port again has to send a guaranteed-service packet $(j+1)^{th}$ slot. The same situation continues for N_{total} consecutive slots until Nj reaches 0.

Nj reaches 0 when tc equals the smallest deadline dl_{min} since Equation 4.2 holds in the slot when the first guaranteed-service packet is sent. Consequently, every guaranteed-service packet is sent from the output port before or at its deadline.

4.4.1 TOTAL PACKET COUNT ALGORITHM STEPS

The operation of the TPC algorithm is described as follows:

For each output port in each cycle:

1. If there is at least one best-effort traffic packet at this output port, calculate the total number of eligible guaranteed-service traffic packets destined for this output port currently present in this node and go to step 2. Else go to step 4.

2. If there are T (T is the number of slots in a frame.) or more (sum of the packets in the corresponding input queues and output queues) eligible guaranteed-service traffic packets destined for this output port, go to step 4. Else go to step 3.

3. If the number of eligible guaranteed-service traffic packets destined for this output port is greater than or equal to the smallest deadline of these packets, then go to step 4. Else go to step 5.

4. This output port serves the eligible guaranteed-service traffic packet with the smallest deadline if there is at least one eligible guaranteed-service traffic packet at this output port. Go to step 6.

5. This output port serves the next best-effort traffic packet if there is at least one at this output port. Go to step 6.

6. End cycle.

4.4.2 IMPLEMENTATION CONCEPTS

According to the Total Packet Count Algorithm, each output port is informed of the total number of eligible guaranteed-service traffic packets present at each input port. TPC Algorithm requires the input ports to send the corresponding packet numbers to every output port in each cycle. Since the iSLIP [10] fabric scheduler also requires some signaling as described in Section 2.2.1 between every input-output port pair, there are already N wires from each input port to every one of the N output ports in an $N \times N$ switch. These N^2 wires can be used to transmit the packet numbers from the input ports to the output ports for the TPC algorithm.

The delay-limiter switch can support T^2 connections simultaneously while there can be T connections at most set up through a single input/output port pair. Every one of these T connections through an input port may have eligible packets at the input queues not yet switched to the output port. Since a packet cannot be eligible before the deadline of the previous packet belonging to the same connection and the previous packet must have been transmitted out from the switch before its deadline, there can be at most one eligible packet for each connection in a node [2]. As the worst case, there can be at most one eligible packet in the input queue of each connection through the same input port, which means that there can be at most T eligible packets at a single input port in any slot. Thus, in the worst case $\lceil \log_2 T \rceil$ bits to encode the number of eligible guaranteed-service packets must be sent from an input port to the corresponding output port in each sub-frame time for the TPC algorithm if the binary logic is used. Thus, the total communication bandwidth between input and output ports required per sub-frame time for TPC is $N^2 \lceil \log_2 T \rceil$ for an $N \times N$ switch.

The output ports must add all the packet numbers coming from *N* different input ports for an $N \times N$ switch. As stated above, the maximum value that the packet number from a single input port to a single output port can take is *T*, which can be expressed by $\lceil \log_2 T \rceil$ bits in binary logic. Therefore, there must be an *N*-input $\lceil \log_2 T \rceil$-bit adder in every output port to compute the total number of eligible packets which are still at their corresponding input ports. In addition to that, the result from this adder must also be summed with the number of eligible packets currently at this output port.

The total packet count computed as described above must then be compared with *T*, the number of slots in a frame. If the total packet count turns out to be less than *T*, it must also be compared to the difference of the smallest deadline of the eligible packets and the current time.

The required computation load is constant and hence the TPC algorithm has a constant time complexity of O(1).

4.5 TOTAL PACKET COUNT ALGORITHM PERFORMANCE

This section presents the simulation results to demonstrate the average performance of the delay-limiter switch with the Total Packet Count algorithm. The simulation results reflect the performance regarding the switching and scheduling of the packets.

For the experiments, we use our own simulator which is built on the same code architecture used to perform the experiments in [2]. The simulation program is coded in C++.

In our experiments, connection-oriented guaranteed-service traffic performance bounds are those provided by the original delay-limiter switch with FDS. Best-effort traffic is carried through connectionless traffic streams. We observe the end-to-end delay performance improvement of the connectionless best-effort traffic after TPC is applied compared to the utilization of the remaining resources in a rudimentary fashion after the guaranteed-service traffic is served by FDS. We also observe the end-to-end delay performance changes of the guaranteed-service traffic.

iSLIP, which is applied as the dynamic fabric scheduler of our switch, is an iterative matching algorithm. "Intuitively, the size of the match increases with the number of iterations; each new iteration potentially adds connections not made by earlier iterations [10]." As stated in [10], it is expected to take about $\log_2 N$ iterations for iSLIP to converge for an $N \times N$ switch. In the operation of TPC, the input/output port pairs which are matched for guaranteed-service traffic according to the transmission matrices do not participate in iSLIP in the current slot. Therefore, it takes less iterations for iSLIP to converge in this case. Also, taking into account the speed considerations of the simulated switch which restricts the time for fabric scheduling, we use iSLIP with only two iterations as the dynamic scheduler for our simulations. We still use the transmission matrices computed upon the connection reservations as in the original delay-limiter switch as the static scheduler for the guaranteed-service traffic.

We adopt the same experiment set-up used in [2] which is described as follows:

There are 10 nodes connected in a linear fashion (H = 10) in our network topology. All links are assumed to have the same speed. Hence, a frame time is T slots on all of the links. There is a unit rate connection set up for each input/output port pair of each node. Hence, each output port receives packets from T distinct input ports. In this case, the number of packets that can arrive at an output port from distinct input ports at the same time maximizes the queuing delay and generates a worst case scenario for our experiments. Traffic sources for both guaranteed-service and best-effort traffic are attached to the input ports of the nodes in parking-lot fashion through these nodes (Figure 4.1). We observe the performance of the guaranteed-service and connectionless best-effort traffic which go through from source node 1 to destination node 10. This end-to-end traffic competes with the background traffic which starts at each node and terminates at the following node.

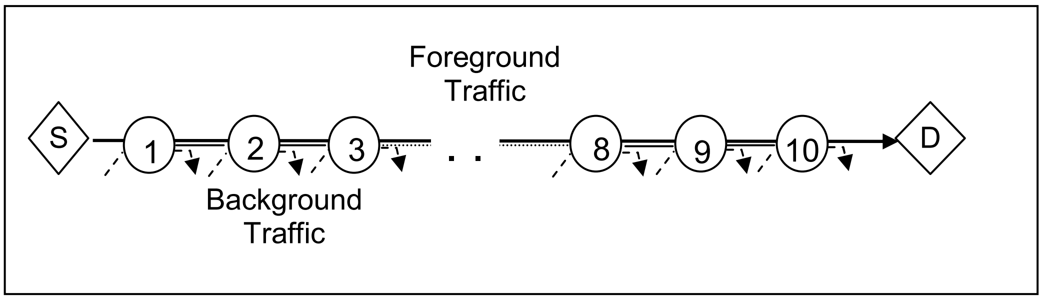


Figure 4.1: The parking-lot topology used in experiments [1].

The guaranteed-service traffic in our experiment is generated according to the traffic shape defined in Section 3.1. The probability of generating a packet in a frame is defined by the given utilization of the allocated rate for the respective connection. The consecutive packets for the same connection are generated in consecutive frames but in random positions in the frame.

The connectionless best-effort traffic is generated according to two different models. The first traffic model used is Poisson arrivals whereas the second traffic model is bursty traffic model. The simulations were run with average burst sizes of 5, 10 and 20 for the bursty arrivals.

The simulations are carried out with different connectionless best-effort traffic utilizations starting from 10% up to 60%. For each utilization level of the connectionless traffic, the guaranteed-service traffic load is varied between 10% and the maximum level which causes 100% utilization on the network together with the connectionless traffic.

We examine the end-to-end delay of the end-to-end traffic on the 10-hop path when the TPC is applied compared with the utilization of the remaining resources for best-effort traffic in a rudimentary fashion after the guaranteed-service traffic. The experiments are performed with enough number of packets so that the confidence interval for the mean delay is 5% with 95% confidence.

We test the performance of TPC with two different frame lengths of 50 and 100 slots using 50x50 and 100x100 switches, respectively. The end-to-end delay bound for the guaranteed-service traffic guaranteed by FDS in the experiment setup described above is $(H+1)T$ which is equal to 550 and 1100 slots for the two frame sizes that are tested.

[2] states that the delay-limiter switch with FDS can provide end-to-end delay guarantees for the guaranteed-service traffic with a fabric speed-up of 2 in the most economical implementation. Therefore, we test the performance of the delay-limiter switch with FDS/TPC with a fabric speed-up of 2.

4.5.1 EXPERIMENTS AND RESULTS

Figures 4.2 to 4.7 shows the ratio of the end-to-end delay of connectionless best-effort traffic when TPC is applied to the end-to-end delay of connectionless best-effort traffic when the remaining resources are used in a rudimentary fashion after the guaranteed-service traffic. The utilization factor for the connectionless traffic is held at 10%, 20%, 30%, 40%, 50%, 60%, 70% and 80% in the respective figures. The utilization factor for the guaranteed-service traffic changes between 0% and the maximum value which adds up to 100% together with best-effort load in each figure. Each figure illustrates the results for connectionless best-effort traffic with Poisson and bursty arrivals with average burst lengths of 5, 10 and 20. The frame length is 50 slots and the switch size is 50x50.

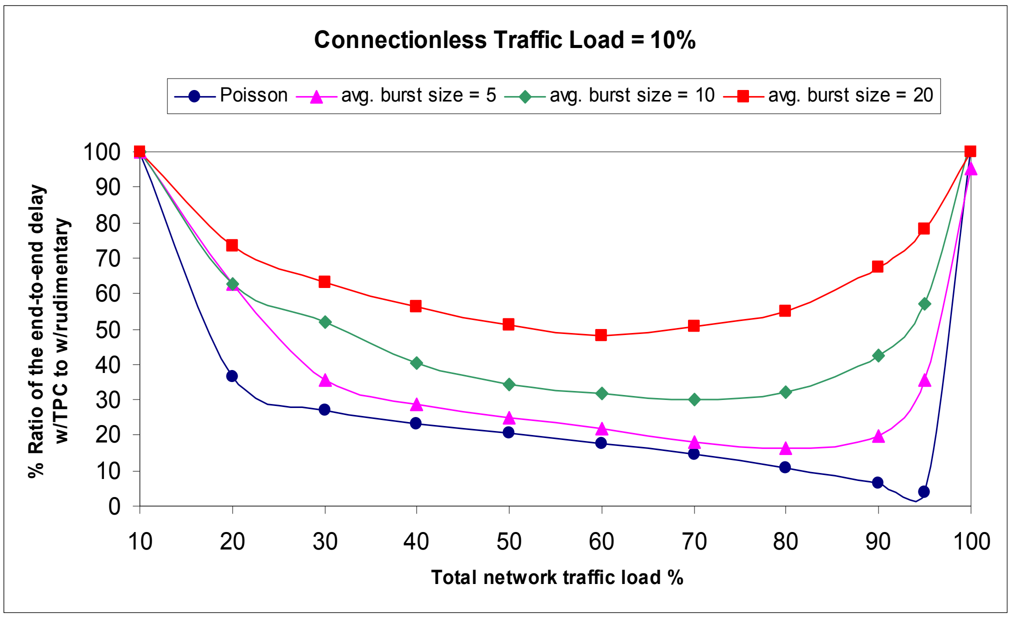


Figure 4.2 Ratio of the end-to-end delay of the connectionless traffic with TPC to without TPC when the connectionless traffic load is 10%.

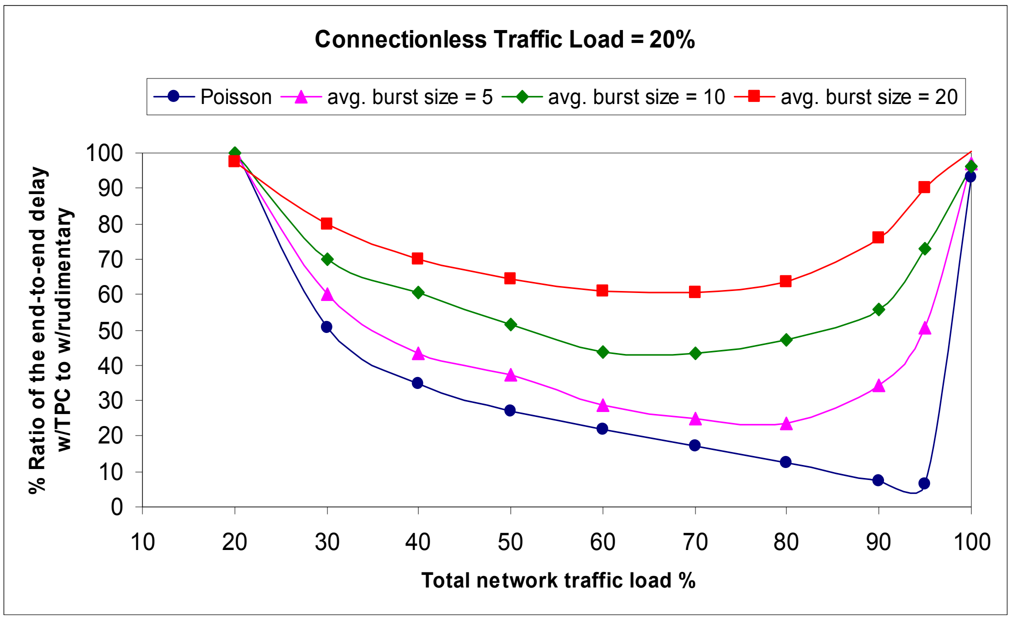


Figure 4.3 Ratio of the end-to-end delay of the connectionless traffic with TPC to without TPC when the connectionless traffic load is 20%.

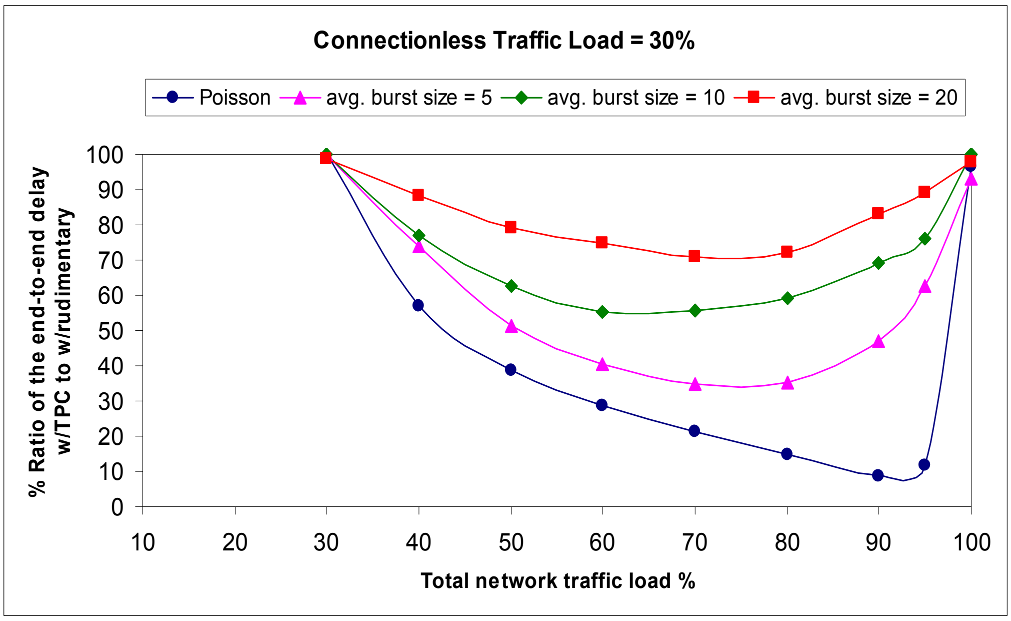


Figure 4.4 Ratio of the end-to-end delay of the connectionless traffic with TPC to without TPC when the connectionless traffic load is 30%.

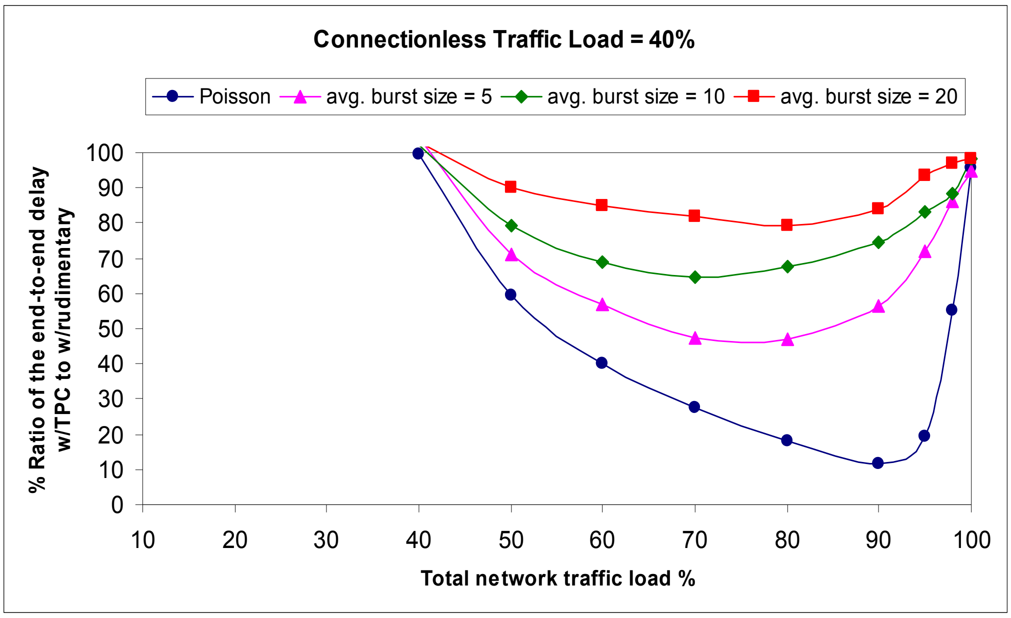


Figure 4.5 Ratio of the end-to-end delay of the connectionless traffic with TPC to without TPC when the connectionless traffic load is 40%.

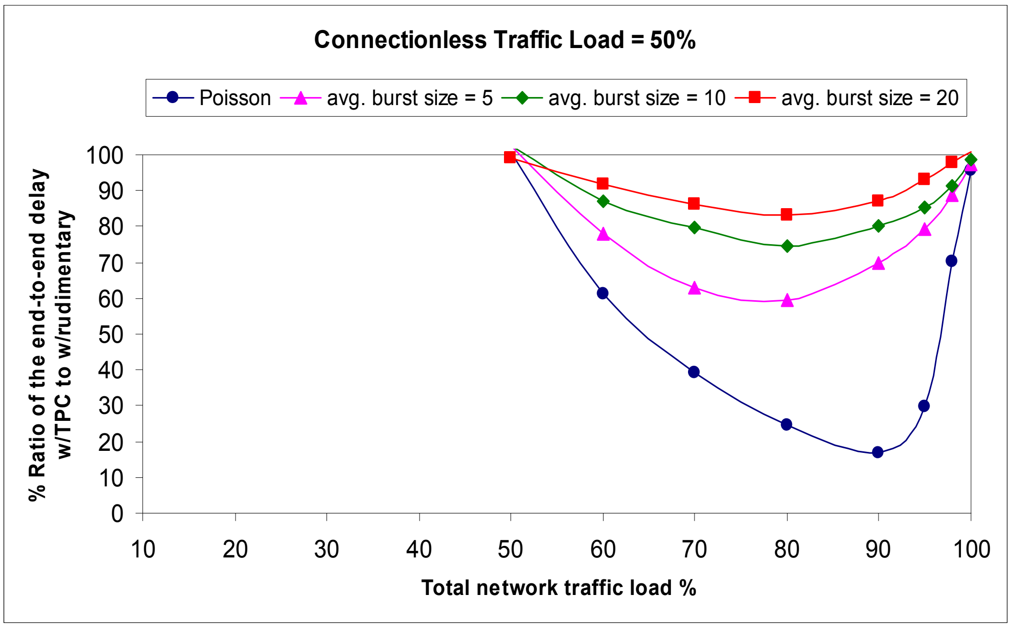


Figure 4.6 Ratio of the end-to-end delay of the connectionless traffic with TPC to without TPC when the connectionless traffic load is 50%.

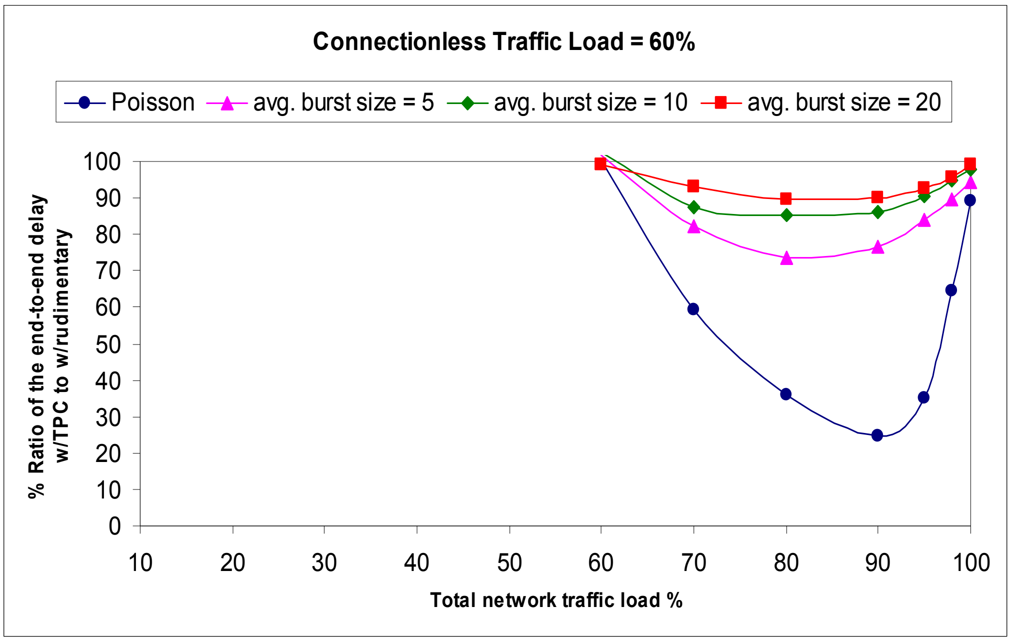


Figure 4.7 Ratio of the end-to-end delay of the connectionless traffic with TPC to without TPC when the connectionless traffic load is 60%.

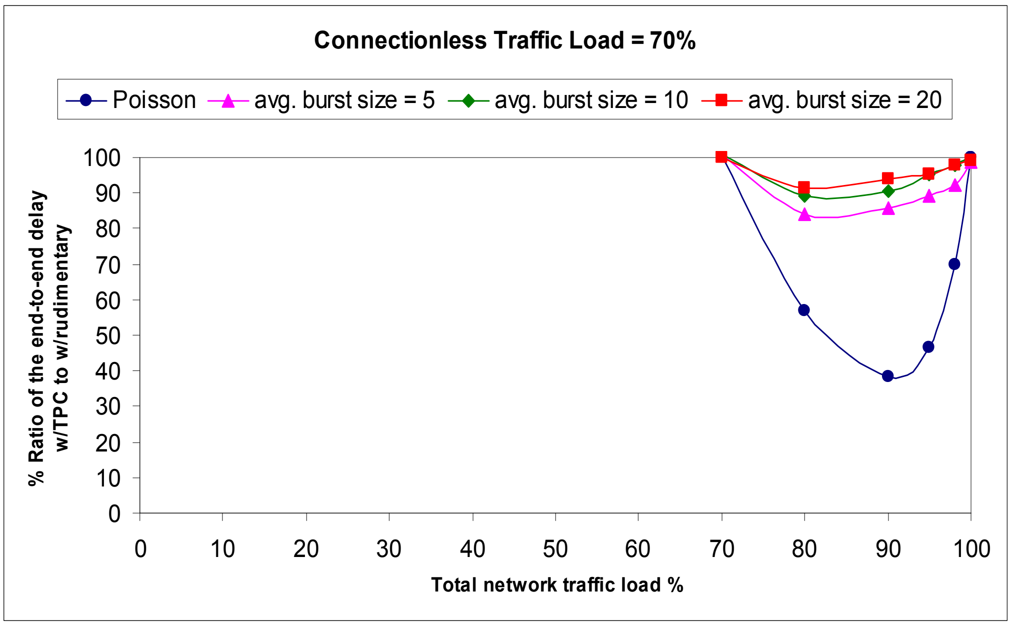


Figure 4.8 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 70%.

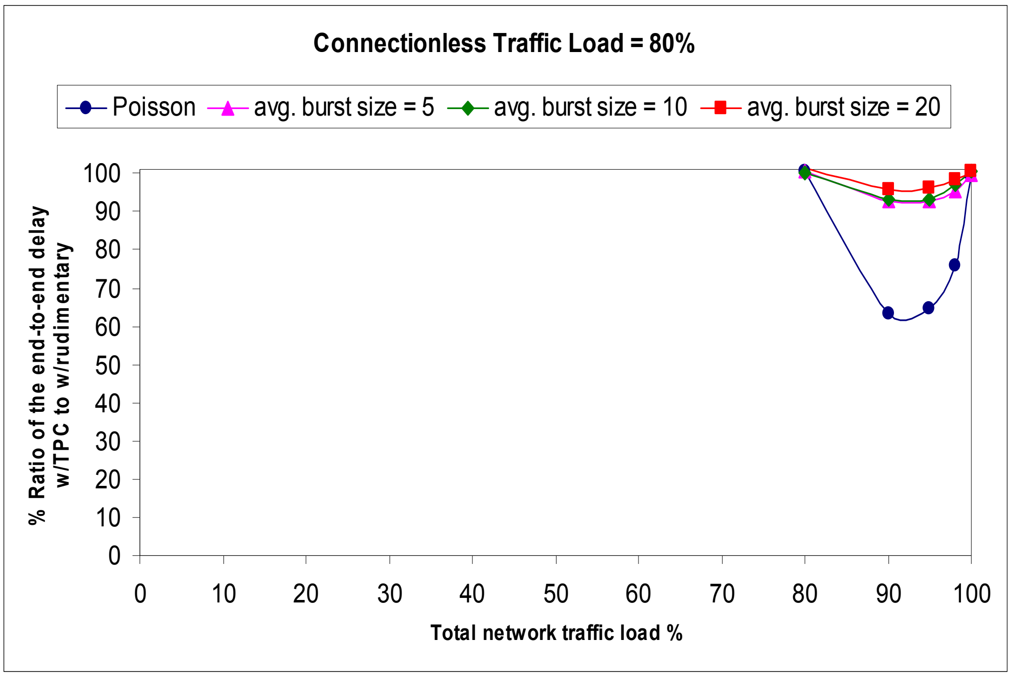


Figure 4.9 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 80%.

Figures 4.10 to 4.17 shows the ratio of the end-to-end delay of connectionless best-effort traffic when TPC is applied to the end-to-end delay of connectionless best-effort traffic when the remaining resources are used in a rudimentary fashion after the guaranteed-service traffic. The utilization factor for the connectionless traffic is held at 10%, 20%, 30%, 40%, 50%, 60%, 70% and 80% in the respective figures. The utilization factor for the guaranteed-service traffic changes between 0% and the maximum value which adds up to 100% together with best-effort load in each figure. Each figure illustrates the results for connectionless best-effort traffic with Poisson and bursty arrivals with average burst lengths of 5, 10 and 20. The frame length is 100 slots and the switch size is 100x100.

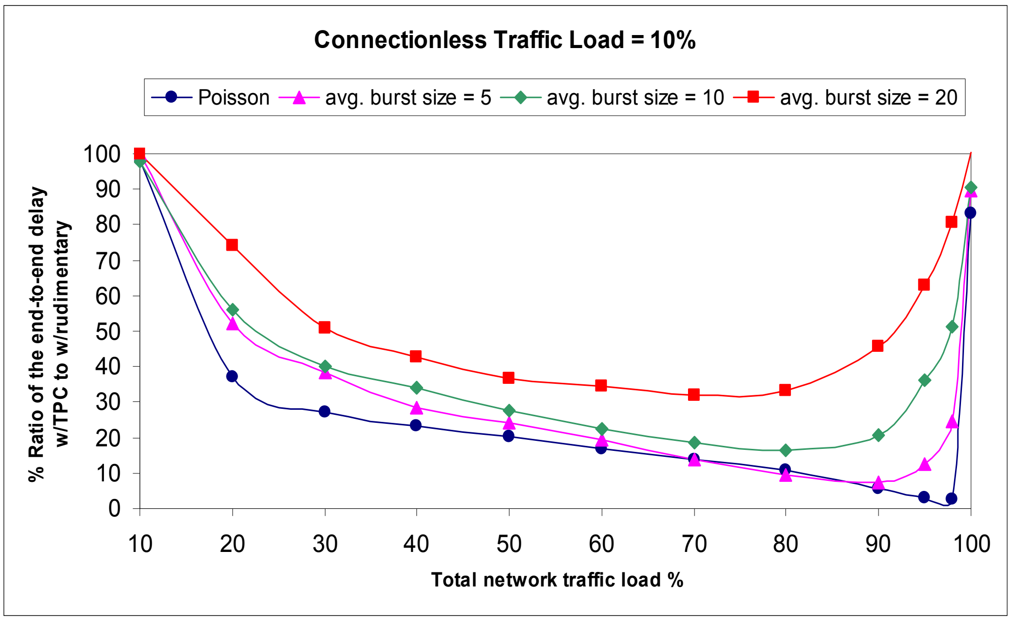


Figure 4.10 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 10%.

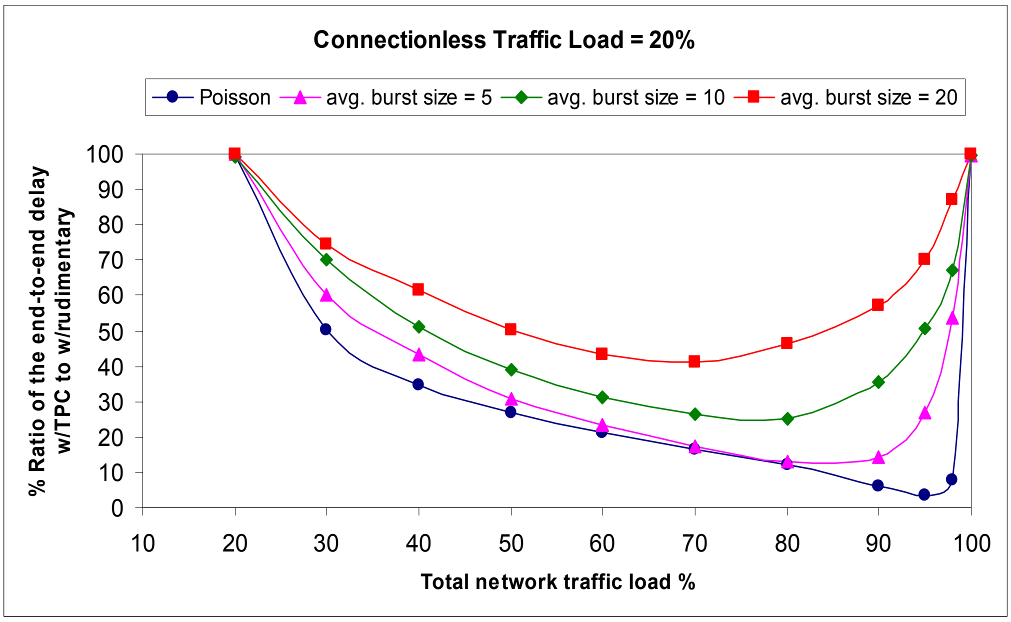


Figure 4.11 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 20%.

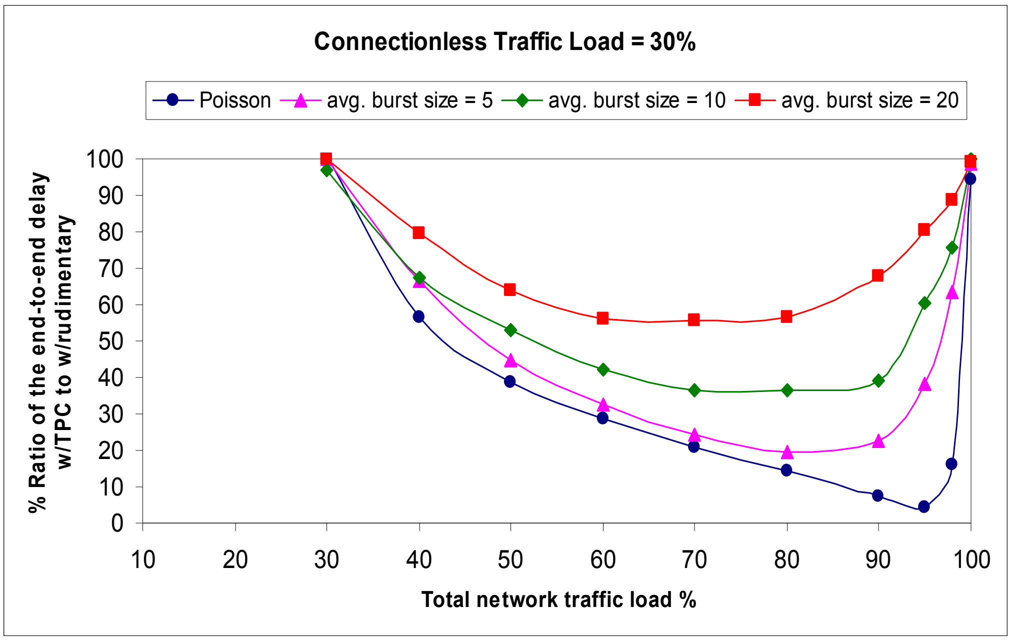


Figure 4.12 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 30%.

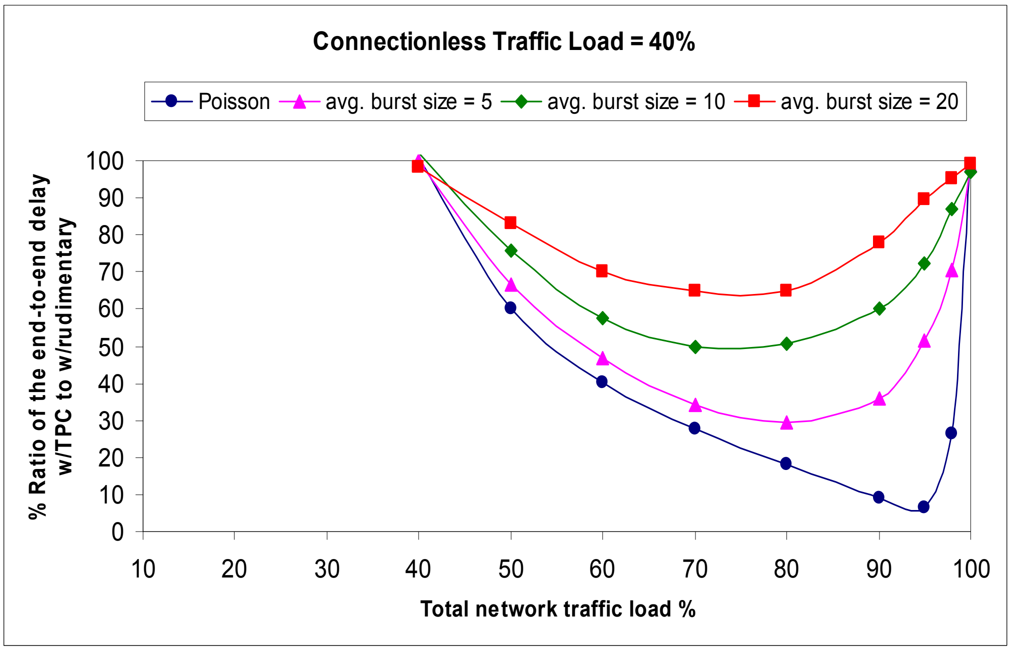


Figure 4.13 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 40%.

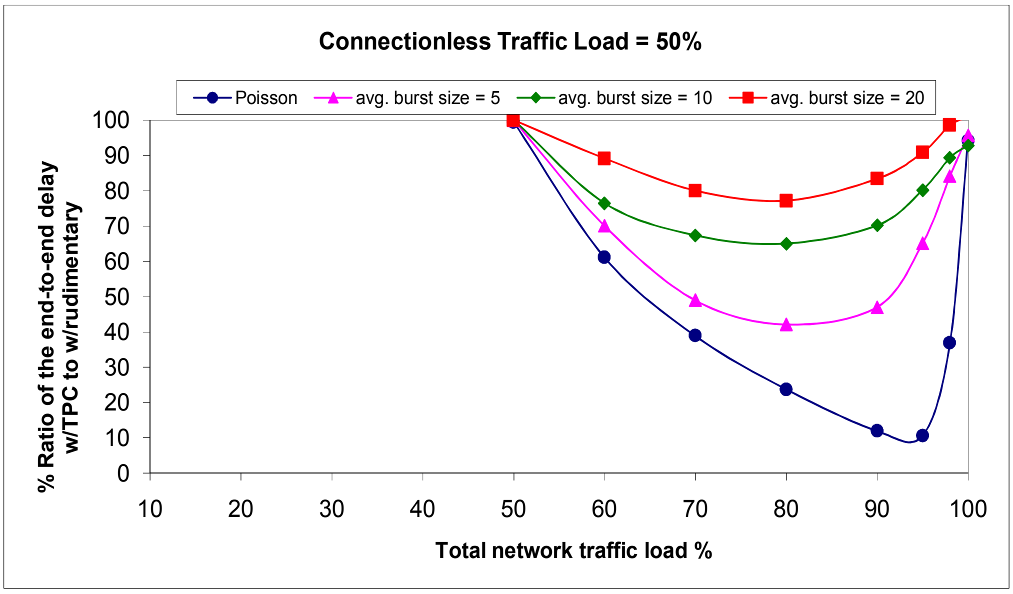


Figure 4.14 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 50%.

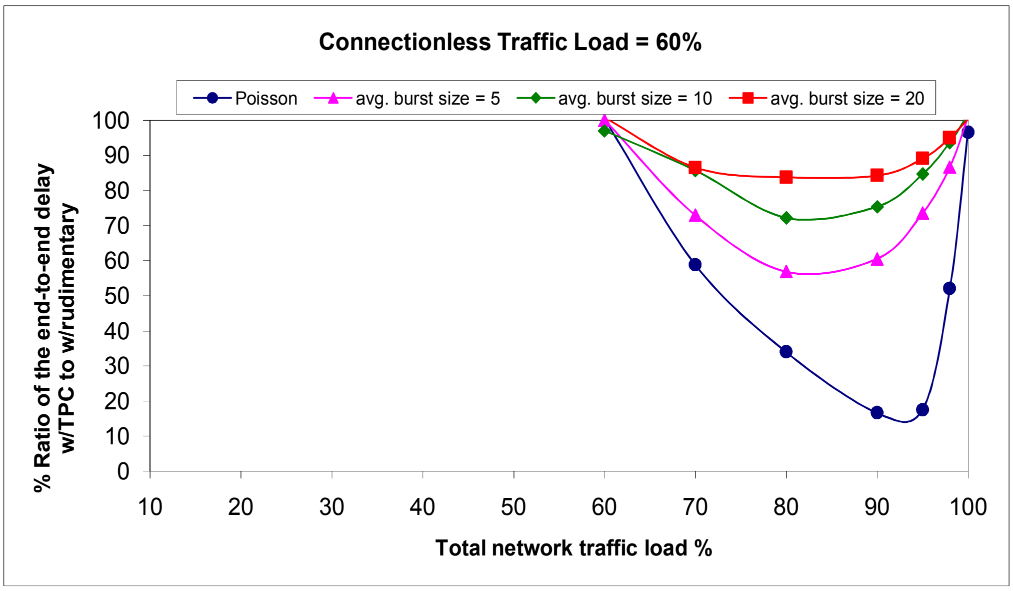


Figure 4.15 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 60%.

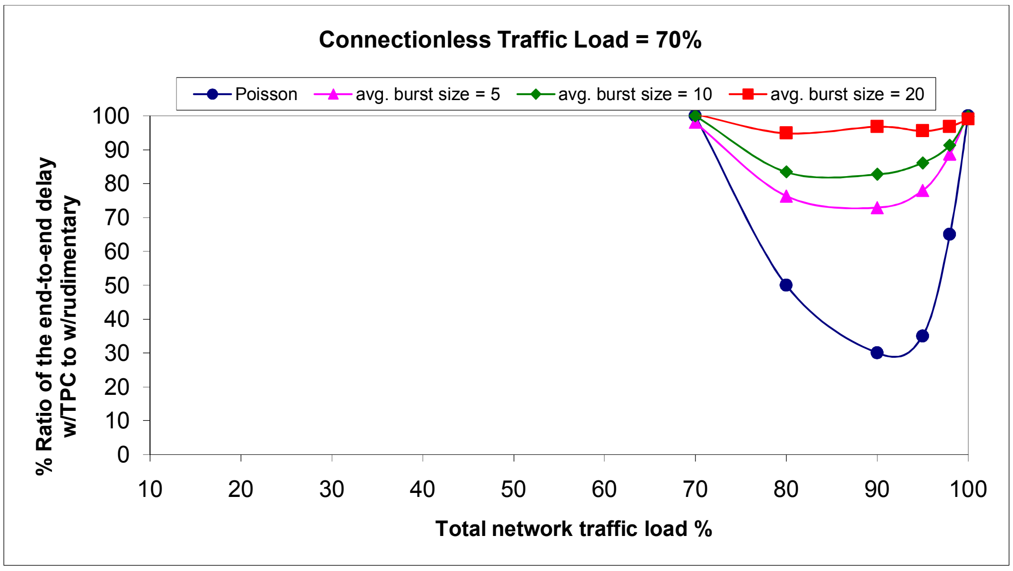


Figure 4.16 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 70%.

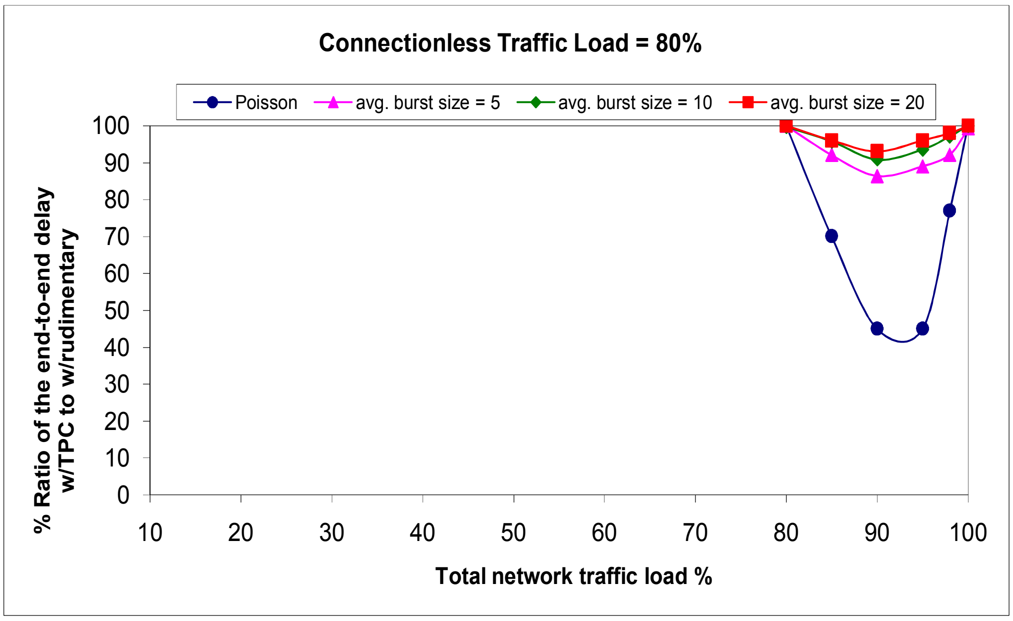


Figure 4.17 Ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method when the connectionless traffic load is 80%.

The simulation results show that TPC algorithm has a significant impact on the end-to-end delay of the best-effort traffic.

These results indicate that TPC algorithm performance decreases as the burstiness of the traffic arrivals increases. The guaranteed-service traffic obeys a strict and regular traffic shape. Thus, it leaves the output ports idle, i.e. leaves room for shifting their service times near their deadlines, in a well-spread fashion within frame times. When the best-effort traffic behaves as regular as possible, i.e. exhibits Poisson arrivals or low burst sizes, the shifting opportunities can be utilized most of the time .On the other hand, when the burst sizes increase, many best-effort packets arrive one after the other and compete for the same shifting opportunities. Therefore, the gain drops with irregular best-effort traffic arrivals.

The gain of the TPC algorithm also decreases as the ratio of the connectionless traffic load to the guaranteed-service traffic load increases. When the best-effort traffic dominates the network load, guaranteed-service packets have little effect on the best-effort traffic delay. The best-effort traffic packets obstruct each other and the main reason for their high delay values is their competence among themselves. Also the profit of shifting the service times of the guaranteed-service packets near their deadlines drops when their share on the network load decreases. As can be observed from the figures, TPC algorithm is most efficient when the total network load is around 70-80% as the average of all type of traffic arrivals.

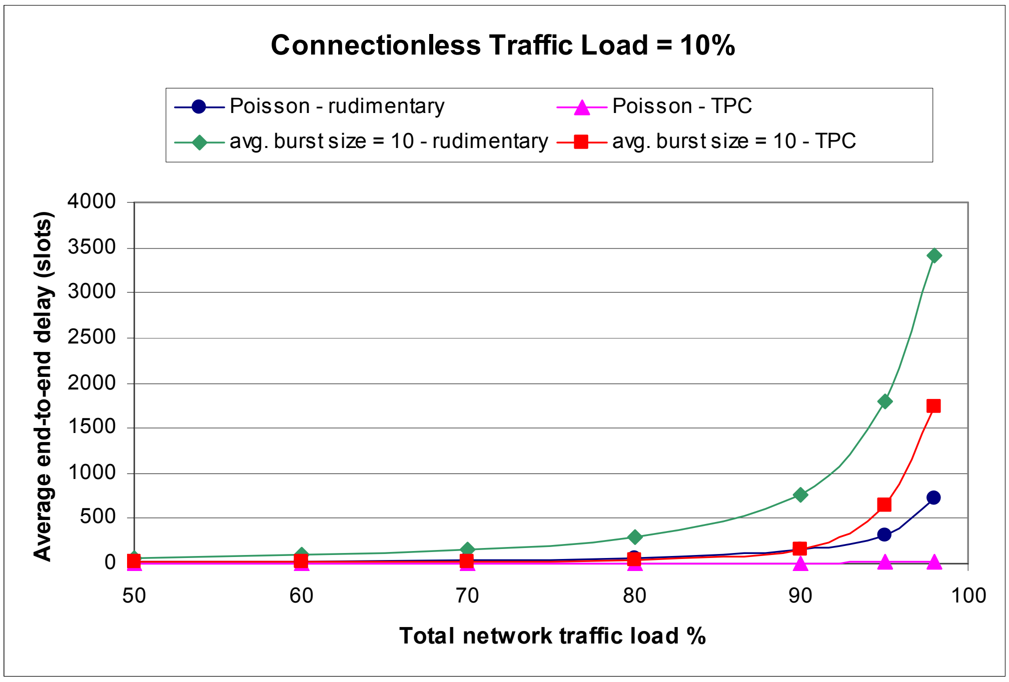


Figure 4.18 The end-to-end delay of the connectionless traffic with TPC and with rudimentary method when the connectionless traffic load is 10% for Poisson arrivals and bursty arrivals with average burst size of 10 packets.

The end-to-end delay of the connectionless best-effort traffic increases as the total network load increases both with the rudimentary method and the TPC algorithm as shown in Figure 4.18. In all figures that illustrate the ratio of the end-to-end delay of the connectionless traffic with TPC to with rudimentary method, the gain increases with the increasing guaranteed-service traffic load until some point. The reason is that as shown in Figure 4.18, the end-to-end delay values with rudimentary method increase more rapidly. The end-to-end delay values with TPC increase less rapidly because as the number of guaranteed-service packets in an output port increases, the gain obtained by shifting the transmission time of the guaranteed-service packets near their deadlines is more beneficial. At some point, the gain reaches maximum and then begins to drop. The reason for the drop in gain is that the number of guaranteed-service packets in an output port become so high that the shifting operation loses its flexibility and it becomes harder to shift the transmission time of the guaranteed-service packets near their deadlines. In summary, with low guaranteed-service traffic loads, there are not many guaranteed-service packets that should be delayed near their deadlines while with high guaranteed-service traffic loads there is not enough room to shift the guaranteed-service packets. Hence, with very low and very high network loads the effect of both algorithms on the end-to-end delay become very close to each other and become equal when the total network load is 100%.

The gain of the TPC algorithm also increases when the frame size in the switches increases. When the frame size increases, this gives more flexibility for the shifting operation and since the burst size / frame size ratio decreases the bursty traffic packets can fit better in the spaces between the guaranteed-service packets.

Appendix A.1 presents the figures that illustrate the actual end-to-end delay values of connectionless best-effort traffic with rudimentary method corresponding to the experiments shown in Figures 4.2 to 4.9. Appendix A.2 presents the figures that illustrate the normalized end-to-end delay values of the guaranteed-service traffic corresponding to the experiments shown in Figures 4.2 to 4.7

CHAPTER 5

DELAY LIMITER SWITCH WITH VARIABLE PACKET SIZE

The original delay-limiter switch operates on fixed-size sub-frames. “The sub-frames are relatively large units unlike the small size ATM cells. Thus, the data packets from the access networks are aggregated into sub-frames. [1]” One or more data packets from an access network are packed into one sub-frame and this sub-frame is carried across the network through delay-limiter switches. The original data packets are unpacked from the sub-frame at the egress of the core network before passing on to the access network.

In order to utilize the bandwidth through the network efficiently, the sub-frames should be filled up to their full capacity before entering the core network. This situation has some consequences.

First, when the utilization load of a guaranteed-service connection is low, the consecutive data packets for this connection arrive less frequently which causes the sub-frames at the network ingress to be delayed for some time until they are filled up. This delay is addressed as *packetization delay* in [1]. [1] proposes to set a threshold time before sending a partially full sub-frame into the network to put a bound to the packetization delay and make a trade-off between packetization delay and bandwidth utilization.

Second, the last data packet placed at the end of a sub-frame may not completely fit into the sub-frame. If the number of necessary bits to fill a sub-frame is less than the size of the next data packet arriving, then this data packet is split into two. The first part of this data packet is placed in the present sub-frame and the second part is placed in the following sub-frame. At that time the first sub-frame is immediately sent into the network and the second sub-frame may have to wait to be filled up with other packets. Even if the second sub-frame is sent into the network right after the first sub-frame without any delay between these two sub-frames, the delay-limiter switches in the network separate the two back-to-back arriving sub-frames by a frame time of T on their end-to-end path. Before a packet which is carried in two consecutive sub-frames in the core network can pass to the access network, it can be reassembled after the second sub-frame arrives at the network egress. Therefore, the end-to-end delay of that packet is equal to the end-to-end delay of the first sub-frame plus the time between the arrivals of the two sub-frames at the network egress. As a result, packetization of the data packets into larger sub-frames may increase the end-to-end delay of the packets.

On the other hand, if the delay-limiter switch works on variable-size packets instead of fixed-size sub-frames, the data packets can be sent into the core network without the need of packetization and the delay components introduced at the network ingress and egress can be avoided. Therefore, the *Delay-Limiter Switch with Variable Packet Size* (DLS/VPS) introduced in this chapter prevents the packetization delay while utilizing the bandwidth efficiently. DLS/VPS also allows the data packets to be carried as a whole between consecutive nodes through the network. In addition, it provides bandwidth guarantee and end-to-end delay bound for guaranteed-service traffic under a certain traffic arrival shape.

5.1 BASIC DEFINITIONS

Just like the state-of-the-art IP routers, the delay-limiter switch with variable packet size works in a slotted and synchronized fashion with fixed data units. Each arriving packet is fragmented into fixed-sized data units called *cells* in the DLS/VPS. All processing in the router is performed on these cells and the original packets are reassembled before they are sent out from the output ports to the transmission lines.

The cell size must be as small as necessary to process the smallest packet in the system as a whole in the DLS/VPS since there is no need to fragment the smallest packet. The time needed to process a cell in the DLS/VPS is called a *slot* and slot is the basic time unit.

A *sub-frame* is defined to be the time necessary to process the largest packet in the DLS/VPS. In other words, when the largest packet in the system is split up into unit cells in the DLS/VPS, the number of the resulting cells is equal to the number of slots in a sub-frame.

A *frame* is the time that it takes to serve all guaranteed-service connections that the system can simultaneously support once if they send packets into the network at their peak reserved rates.

A *unit-rate connection* is a connection set up through the DLS/VPS's whose peek data rate is one sub-frame per frame.

A *multi-rate connection* is a connection set up through the DLS/VPS's whose peek data rate is an integer multiple of one sub-frame per frame.

Suppose N is the number of cells obtained as a result of fragmenting the largest packet in the system. Then, there will be exactly N slots in each sub-frame in the DLS/VPS. If the number of the unit-rate connections that a DLS/VPS can simultaneously support with the promised service guarantees is S, then a frame involves S number of sub-frames. A frame consists of S sub-frames whereas a sub-frame consists of N slots. Hence, a frame consists of T slots where

$T = S \times N$ (Equation 5.1)

Figure 5.1 illustrates the relationship among slot, sub-frame and frame in a system where the ratio of the sizes of the largest packet and the smallest packet in the system is 6. This ratio is also equal to the ratio of the length of a sub-frame to the length of a slot and $N = 6$. The DLS/VPS can support 4 unit-rate connections simultaneously, so a frame time involves 4 sub-frames and $S = 4$. Thus, by Equation 5.1 $T = 24$.

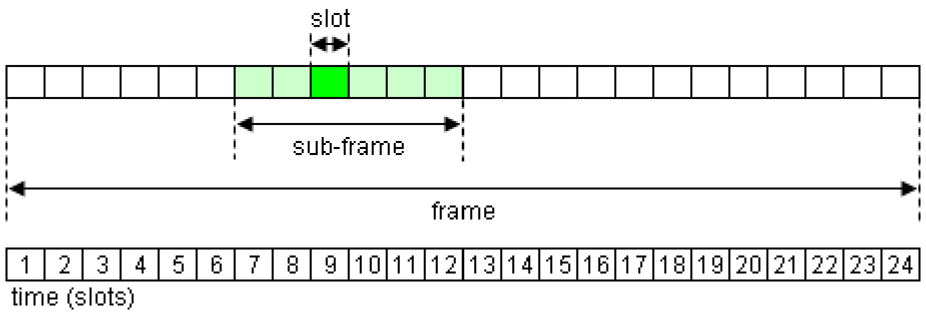


Figure 5.1 Slot, sub-frame and frame relationship.

If the cell size in the DLS/VPS is selected to be 48 Bytes, a slot corresponds to 38.4 nanoseconds at a line rate of 10 Gbps. If the largest packet in the system is assumed to be 1500 Bytes, a sub-frame consists of 32 slots and lasts for 1.2288 microseconds. If *S*, the maximum number of guaranteed-service unit-rate connections that a DLS/VPS can simultaneously support, is assigned to be 100, then a frame involves 100 sub-frames and endures 122.88 microseconds.

5.2 CONNECTION ADMISSION AND RESERVATION

Connection admission is controlled in the same way as the original delay-limiter switch. Each connection is assigned a fraction of the total link bandwidth. The capacity is allocated to connections based on their peak rates. The allocated rate for a connection is expressed with the unit of sub-frames per frame. Hence, the reservation granularity is one sub-frame per frame. Since the number of slots in a sub-frame is equal to the number of cells that the largest packet in the system involves, the reservation granularity is such that a packet of any length can be accommodated in a single frame. If the reservation granularity were less than a sub-frame per frame, then the largest packet in the system would have to be transmitted in two consecutive frames. In order to transmit a large packet in two consecutive frames, either it has to be carried as a whole requiring more complex calculations than DLS/VPS does or the packet has to be split up into two at the network ingress and reassembled at the network egress resulting in the same problems the original delay-limiter switch experiences..

The connections are not allowed to oversubscribe the line rates. The connection admission control ensures that the links are not overloaded by the connections and the sum of the rates of all guaranteed-service connections on a link does not exceed *S* sub-frames per frame. This condition satisfies Equation 5.1 and the fact that at most *S* unit-rate connections can be established on a link.

The symbol ρ is used to denote the number of sub-frames per frame allocated for a connection. A multi-rate connection can have rates starting from ρ=1 sub-frame per frame up to the maximum rate of ρ=*S* sub-frames per frame.

Reservations are done on a sub-frame basis, which means that a guaranteed-service connection can reserve integer multiples of a sub-frame per frame. This is the same as the original delay limiter switch. But the traffic sources are allowed to transmit packets with variable sizes into the network as they are generated. Thus, the variable-length packets do not suffer any extra delay for forming sub-frame-sized packets at the network ingress as is the case with the original delay-limiter switch.

5.3 INPUT TRAFFIC SHAPE

The packet arrivals into the network for connection-oriented traffic comply with a certain traffic shape. There is a traffic shaper for each connection source. The traffic shaper controls the transmission rate of a connection into the network.

5.3.1 THE TRAFFIC SHAPER

The traffic shaper uses a new concept called *service frame*. A service frame is the period of time that cannot be shorter than *T* slots and can end when the corresponding connection transmits enough cells of data to use up its entire reserved rate per frame. The service frames may have various sizes depending on the packet transmission patterns while the length of the shortest service frame is equal to the length of the system frame. Consider a unit-rate connection with an allocated rate of 1 sub-frame per frame which is equal to N slots per frame. Then, the length of the k^{th} service frame for the traffic shaper of this connection Lsf_k is computed as follows:

$$Lsf_k = \max(T, (tlast_k - tfirst_k + 1)) \quad \text{(Equation 5.2)}$$

where $tlast_k$ is the time in slots in which the kN^{th} cell of data is sent and $tfirst_k$ is the time in slots in which the $((k-1)N+1)^{\text{th}}$ cell of data for this connection is sent out of the shaper into the network.

From this point on, the word frame in the text refers to the system frame which is exactly equal to T slots whereas service frame refers to the concept adopted by the traffic shaper.

Example 5.1: Consider a system (Figure 5.2) in which the largest packet size is 3 cells and the number of unit-rate connections that can be simultaneously supported is 4. Thus, a frame in the system consists of 4 sub-frames and 12 slots where,

$N = 3$ (number of slots per sub-frame)
$S = 4$ (number of sub-frames per frame)
$T = 12$ (number of slots per frame)

Therefore, the reservation granularity of the network is 3 slots per frame.

Suppose a unit-rate connection A whose packets are of the size of the smallest packet in the system. Therefore, connection A sends only packets that are one cell long. The first service frame of the unit-rate connection *A* starts just when the shaper transmits the first cell of data of connection *A* into the network.

In Figure 5.2-a, connection *A* sends three packets back-to-back at the beginning of its first service frame and consumes its entire quota for the current frame. Connection *A* cannot send any more packets until the current system frame ends and the next one begins because connections are not allowed to oversubscribe the line rates. When a period of 12 slots passes after the beginning of the first service frame, the first service frame for connection *A* ends and the second service frame starts, since connection *A* has already used up its entire quota by that time.

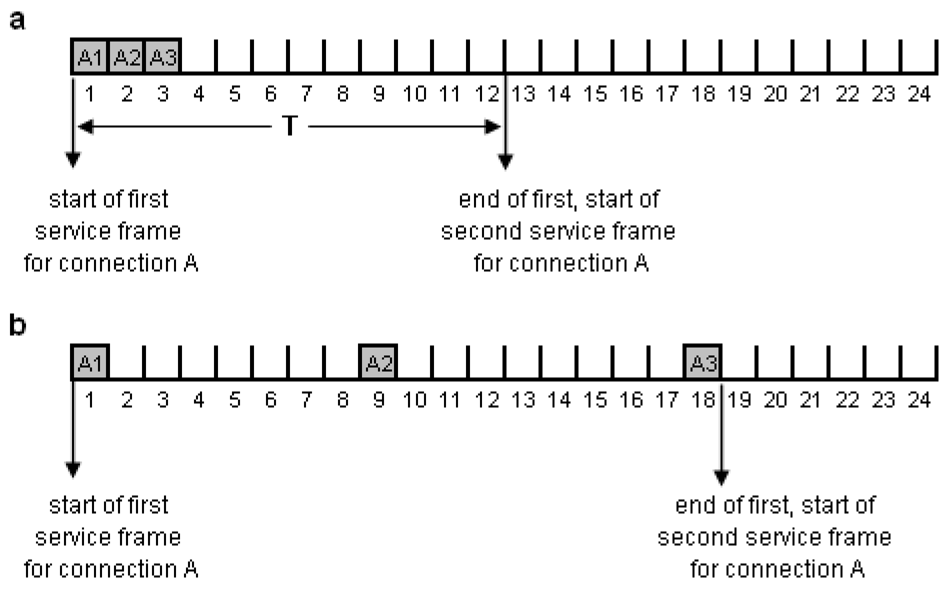

a
A1
A2
A3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
T
start of first service frame for connection A
end of first, start of second service frame for connection A
b
A1
A2
A3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
start of first service frame for connection A
end of first, start of second service frame for connection A

Figure 5.2 Different service frame lengths in a shaper for different packet arrivals. (a) The length of the first service frame for connection *A* is equal to the system frame length. (b) The first service frame for connection *A* lasts longer than a system frame.

In Figure 5.2-b, connection *A* sends three packets separated by a number of idle slots between them. It sends the first two packets within a system frame time after the beginning of its first service frame. Connection *A* sends its third packet in the 18th slot after the beginning of its first service frame while a system frame lasts only 12 slots. The first service frame does not end after slot 12 because connection *A* has not yet consumed its entire quota by that time. Instead, the first service frame continues until the third packet is sent and the entire quota of connection *A* finishes in slot 18. The second service frame for connection *A* immediately starts at slot 19 and connection *A* can send new packets using its refreshed quota by the beginning of its second service frame.

The traffic shaper utilizes two counters to measure the cells of data transmitted into the network per service frame time by the corresponding connection.

The first counter, *quota counter*, counts the number of slots transmitted by the corresponding connection in a service frame. At the beginning of each service frame, the quota counter is initialized to the number of reserved slots per frame for this connection, which is the peak rate of this connection per frame time. Every time the shaper sends out a cell of data, the first counter decrements by one.

The second counter, *frame counter*, counts the slots in each service frame and keeps track of the service frame length. When the quota counter is restored to the number of reserved slots for a connection, i.e. at the beginning of each service frame, the corresponding frame counter is initialized to the number of slots in a system frame. Then, the frame counter decrements by 1 at the beginning of every slot if it is greater than 0.

5.3.1.1 OPERATION OF THE TRAFFIC SHAPER

The operation of the traffic shaper using the dedicated counters can be described as follows:

Counter operations:

- ***Cq*****:** quota counter
- ***Cf*****:** frame counter
- Quota counter decrements by one if a cell of data is transmitted out from the shaper.
- Frame counter decrements by one in each slot unless it is 0.

1. *Cf* = 0 & *Cq* = 0: restore *Cf* and *Cq*, start decrementing *Cf* and *Cq*

2. *Cf* = 0 & *Cq* > 0: do not restore *Cf* or *Cq*, stop decrementing *Cf*, continue decrementing *Cq*

3. *Cf* > 0 & *Cq* = 0: do not restore *Cf* or *Cq*, continue decrementing *Cf*, stop decrementing *Cq*

4. *Cf* > 0 & *Cq* > 0: do not restore *Cf* or *Cq*, continue decrementing *Cf* and *Cq*

Eligibility check:

1. (there is a packet, Pa) & (size of Pa <= *Cq*): start sending the packet out

2. (there is a packet, Pa) & (size of Pa > *Cq*) & (*Cq* >= *Cf*): start sending the packet out

3. (there is a packet, Pa) & (size of Pa > *Cq*) & (*Cq* < *Cf*): do not start sending the packet out

4. (there is no packet): do nothing

The following paragraphs state the detailed operation of the shaper.

Restoration of the counters:

When the frame counter for a connection reaches zero, if the quota counter is also zero, the quota counter and the frame counter are restored to their initial values and the next service frame of this connection starts. The restriction of the both counters to be zero ensures that the service frame is at least as long as a system frame and the entire quota is consumed before the current service frame ends.

When the frame counter for a connection reaches zero, if the quota counter is not zero yet, the frame counter stops decrementing until the beginning of the next service frame. This indicates that the first condition for this service frame to end is satisfied and this service frame can end when the reserved quota is used up.

Start of the service frames in the shaper:

After a new connection is established through the network, the first service frame for a shaper begins with the transmission of the first cell of data by the corresponding connection. The shaper does not track the frame duration and the frame counter does not change until the first service frame starts by the transmission of the first bit of data for the corresponding connection.

The following service frames start one after the other just as the previous service frame ends.

At the beginning of the first service frame, with the transmission of the first cell of data, the quota counter is decremented by one from its maximum (initial) value for the connection. At the same time, the frame counter starts decrementing and decrements by one every next slot until it reaches 0. In the following slots, the shaper transmits one cell of data and the quota counter decrements by one if there are any cells of data in the shaper, the quota counter is greater than zero and the packet that this cell of data belongs to is *eligible*.

Eligibility:
If there is a packet in the shaper and the quota counter is greater than or equal to the number of cells this packet consists of, then this packet is *eligible* to be served by the shaper. If the quota counter is zero, this packet is not eligible to be transmitted into the network in the current service frame and will be eligible by the beginning of the next service frame when the quota counter is refreshed. If the quota counter is less than the number of cells of this packet and greater than zero, this packet becomes eligible when the frame counter becomes equal to the quota counter. These rules guarantee that any connection is not allowed to send packets into the network more than its reserved rate in any frame time.

Example 5.2: Consider the system in Example 5.1 where $N = 3$ (number of slots per sub-frame), $S = 4$ (number of sub-frames per frame) and $T = 12$ (number of slots per frame). This time connection A is still a unit-rate connection but can send packets of size 1, 2 or 3 cells.

In Figure 5.3 below, connection A sends three packets into the shaper. At the beginning of the first service frame, the quota counter is initialized to 3. The length of the first packet that arrives at the shaper is 2 cells. Since the packet length is less than the quota counter, the first packet is eligible and it is transmitted out from the shaper into the network immediately. With the transmission of each cell of data, the quota counter is decremented by one. When the second packet arrives at the shaper, the quota counter value is 1. The size of the second packet is also 1. Since the packet size is equal to value of the quota counter, the second packet is eligible and it is transmitted out from the shaper into the network immediately. With the transmission of the second packet, the quota counter reaches 0. This indicates that connection *A* cannot send any more data into the network until the next service frame starts. When the minimum service frame length is reached after slot 12 with the quota counter value equal to zero, the first service frame ends and the second service frame begins. The third packet arrives at the shaper in slots 14 and 15 and it is eligible since the quota for connection *A* is refreshed by the beginning of the second service frame.

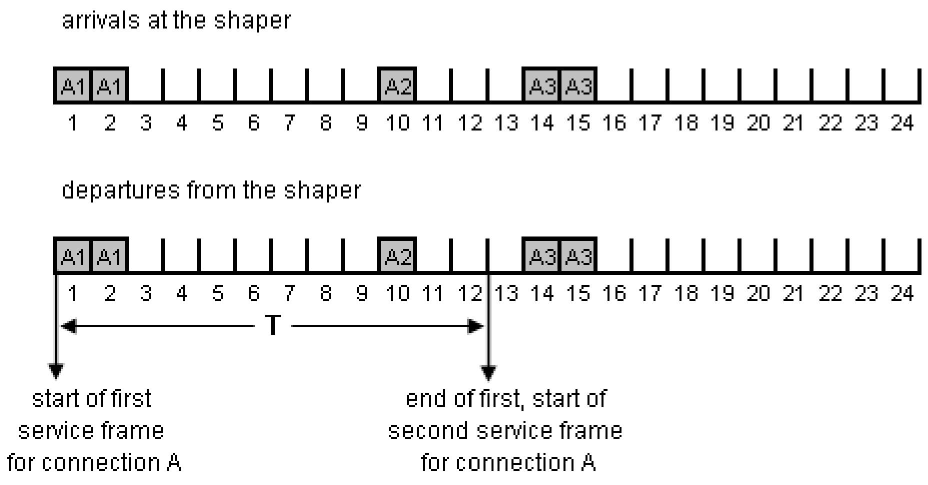


Figure 5.3 Traffic shaper operation with proper packet arrivals

Figure 5.4 below shows connection *A* sending three packets into the shaper. At the beginning of the first service frame, the quota counter is initialized to 3. Since the length of the first packet, which is 2, is less than the quota counter, the first packet is eligible and it is transmitted out from the shaper into the network as soon as it arrives. After the transmission of the first packet, the quota counter value drops to 1. When the second packet arrives at the shaper, the quota counter value is 1 whereas the size of the second packet is 2. Since the packet size is greater than the value of the quota counter, the second packet is not eligible by the time it arrives. Due to the fact that the frame counter decrements by one in every slot after the beginning of the service frame unless it is 0, its value reaches 1 at the beginning of slot 12. At that time, since the quota counter and the frame counter values are equal, the second packet becomes eligible and the shaper starts transmitting it into the network. When the first cell of the second packet is sent out from the shaper, both quota counter and the frame counter reach 0. Thus, the first service frame ends and the second service frame begins. The counters are restored to their initial values. In the first slot of the second service frame, the second cell of the second packet is sent out from the shaper. Hence, the second packet is transmitted into the network as a whole without connection *A* exceeding its reserved rate, *N* slots per frame. The third packet arrives at the shaper in slots 14 and 15 and it is eligible as it arrives since the quota for connection *A* is refreshed by the beginning of the second service frame and the quota counter value is equal to the packet size.

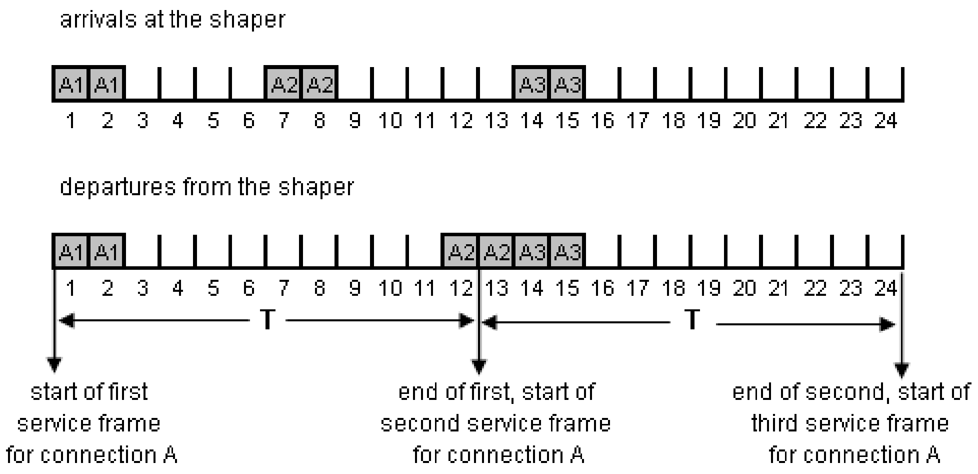

arrivals at the shaper
A1 A1 A2 A2 A3 A3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
departures from the shaper
A1 A1 A2 A2 A3 A3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
T
T
start of first service frame for connection A
end of first, start of second service frame for connection A
end of second, start of third service frame for connection A

Figure 5.4 Traffic shaper operation with frequent packet arrivals

The second packet of connection *A* is transmitted into the network in two different service fames. The first part of the second packet is sent at the end of the first service frame while the remaining part of it is sent in the second service frame. Such packets which are transmitted in two consecutive service frames are called *in-between packets*.

The operation of the traffic shaper described above allows back-to-back packet arrivals into the network. Packets A_2 and A_3 in Figure 5.4 are transmitted into the network one after the other. The sum of their sizes seems to exceed the reserved rate but actually the sum of the three packets A_1, A_2 and A_3 conform to the reserved rate. Such an operation of the shaper provides flexibility to the packet arrivals while still obeying the reserved transmission rates.

5.4 SWITCH FABRIC SCHEDULER

DLS/VPS uses the same static fabric scheduling method as the original delay-limiter switch. The fabric scheduler works with a speed-up of at least two in order to provide end-to-end delay guarantees even in case of traffic matrix rearrangements which occur when a new connection is established or an existing connection terminates. The input-output port match durations for the connections in the switch fabric are constant for every port and a match lasts for a period of time it takes to switch the longest packet from its input port to its output port. When the speed-up is two, each match is kept for half a sub-frame time which is equal to $N/2$ slots where a sub-frame time is the time necessary to transmit the longest packet out from an output port to the transmission line. Regardless of the number of cells to be switched at any time for any connection, the switch fabric is reconfigured as many times as the speed-up value per sub-frame time. Therefore, the switch fabric is reconfigured twice according to two different transmission matrices in each sub-frame time when the speed-up is two.

As stated in [1], *Traffic Matrix* (D) describes the traffic load for the connections between I/O port pairs. D is an SxS matrix for a switch with S input ports and S output ports. The element d_{ij} of D is the number of sub-frames of data per frame reserved to be switched from input i to output j. The traffic matrix D is computed and updated whenever a change in the state of any connection occurs, that is when either a new connection is to be set up or an existing one is to be terminated [1].

The *Transmission Matrix* (Tx_k) shows the configuration of the switch fabric in a specific slot [1] as in the original delay-limiter switch. But the difference from the original delay-limiter switch is that each configuration of the switch fabric is kept for N / Su consecutive slots where Su is the speed-up factor and N is the number of slots in a sub-frame. Thus, the switch fabric configuration changes Su times every sub-frame time similar to the original delay-limiter switch. The switch fabric is configured according to Su different transmission matrices in each sub-frame time and the whole set of transmission matrices is applied Su times per frame time.

[26] states that a switch configuration is an interconnection pattern of the switch such that at most one channel can be switched in a conflict-free manner between input-output port pairs. The elements of Tx_k, t^k_{ij} are 1 if input port i is connected to output port j, 0 otherwise. There can be only one 1 entry in a column or in a row. Hence, the maximum number of 1's in the transmission matrix is S for an SxS switch [1].

In a system that can simultaneously support *S* guaranteed-service connections at most, a frame consists of *S* sub-frames and hence *S* distinct transmission matrices are computed from the traffic matrix for the connections. Just like the original delay-limiter switch, the computed transmission matrices are scheduled in a round-robin (RR) fashion with fabric speed-up of *Su* and the entire set of *T* transmission matrices is scanned *Su* times for each *T* slot period. When the speed-up is two, the switch fabric is configured once according to each of *S* transmission matrices in half a frame time and the same transmission matrix pattern is applied for a second time in the second half of the same frame time.

Condition 5.1: Traffic matrix can be rearranged only once in a frame time. After a new rearrangement of the traffic matrix, each of the resulting transmission matrices must be applied *Su* times in order to provide fairness to all connections before the next rearrangement can take place.

5.4.1 INTER-SWITCHING DELAY

The *switching delay* (SD) for a packet is defined as the time between the arrival of the last cell of the packet at the input port and the arrival of the last cell of the packet at the output port.

The *inter-switching delay* (ISD) for a packet is the time between the arrival of the last cell of the packet at the input port and the arrival of the last cell of the packet at the output port in case there is no other packet belonging to the same connection waiting to be switched to the output port when this packet arrives at the node.

When the fabric speed-up is *Su*, each connection has the opportunity to switch its packets from the input ports to the output ports *Su* times per frame where each turn lasts *N* / *Su* slots for each connection.

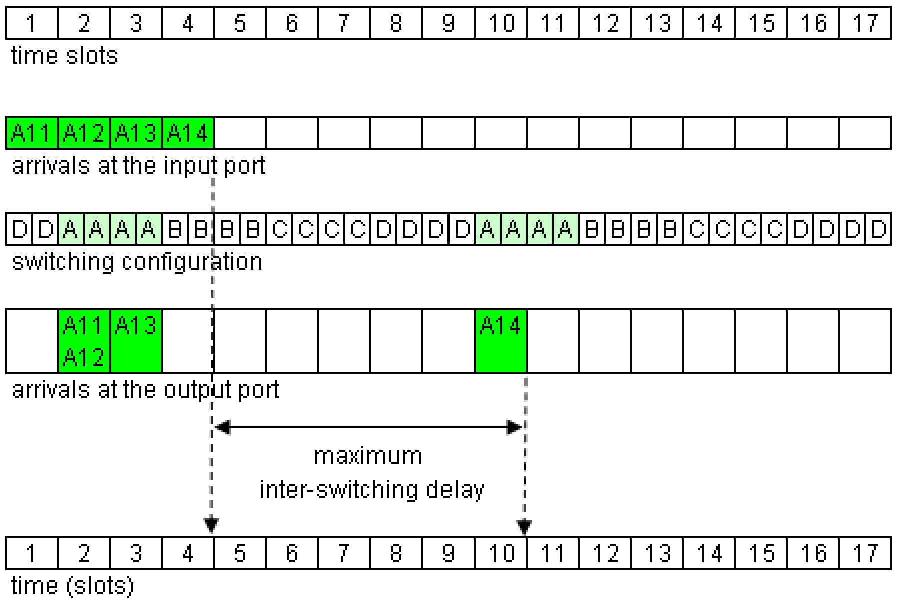


Figure 5.5 Maximum inter-switching delay without TM rearrangement

The maximum inter-switching delay for a multi-cell packet occurs when the last cell of the packet arrives at the input port just after its input port is disconnected from its output port, which were connected for the corresponding connection, through the switch fabric as in Figure 5.5. In the system in Figure 5.5, the sub-frame size *N* is 4 and the number of sub-frames per frame *S* is also 4. Thus, the frame length *T* is 16 slots. The speed-up *Su* is 2. There are 4 unit-rate connections set up through the switch in the figure, connection *A*, *B*, *C* and *D*. A_{ij} indicates the cells of the corresponding packet of connection *A* where *i* denotes the packet number and *j* denotes the cell number in that packet. Therefore, A_{ij} denotes the j^{th} cell of the i^{th} packet of connection *A*. The cells of connection *A* arrive at the corresponding input port of the switch during the slots 1 to 4. According to the fabric schedule this input port is connected to the output port of connection *A* in the slots 2 and 3; a total of four times consecutively within two slots since the sub-frame size is 4 and the fabric speed-up is 2. The last cell A_{14} arrives after the connection between the required ports is resolved and this cell has to wait until the next time that its input-output port pair is connected for its connection which is shown in the switching configuration in Figure 5.5. In this case, A_{14} has to wait for the switching of all other connections.

There are *S* connections including *A* and (S – 1) connections excluding *A* through the switch. The switch fabric configuration is kept constant for each connection for N / Su slots. Therefore, the inter-switching delay *ISD* for any unit-rate connection without TM rearrangement is

$$ISD = (S - 1) \times (N / Su) \qquad \text{(Equation 5.3)}$$

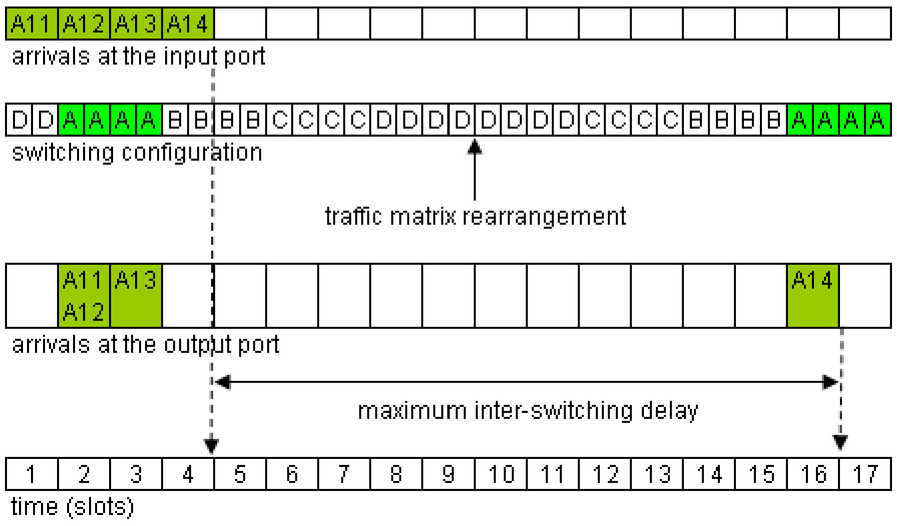
which can be expressed, by using Equation 5.1, as

$$ISD = (T - N) / Su \qquad \text{(Equation 5.4)}$$

Hence, the inter-switching delay ISD of A_1 in this case is:

$$\begin{aligned} \mathrm{ISDA}_1 &= (T - N) / Su \\ &= (16 - 4) / 2 = 6 \text{ slots} \end{aligned}$$

where ISDA_1 is the inter-switching delay of packet A_1.



A11
A12
A13
A14
arrivals at the input port
D D A A A A B B B B C C C C D D D D D D D D C C C C B B B B A A A A
switching configuration
traffic matrix rearrangement
A11
A12
A13
A14
arrivals at the output port
maximum inter-switching delay
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
time (slots)

Figure 5.6 Maximum inter-switching delay in case of TM rearrangement

When there are rearrangements in the traffic matrix, the switching delay of the connections change. Figure 5.6 shows the same arrival pattern for connection *A* as in Figure 5.5. In this case, the TM is rearranged after slot 9 which would be the end of the inter-switching delay of connection *A* if the TM rearrangement did not occur. Therefore, A_{14} misses its turn to get switched to the output port by one slot. The new switching configuration is arranged in such a way that connection *A* gets the last order in the fabric schedule after all other connections. This represents the worst case situation for connection *A* in which it has to wait for the switching of all other connections twice until its next turn in the fabric schedule. Therefore, the maximum inter-switching delay for any unit-rate connection with TM rearrangement ISD^{max} is

$$ISD^{max} = 2 \text{ x } (S - 1) \text{ x } (N / Su) \qquad \text{(Equation 5.5)}$$

which can be expressed, by using Equation 5.1, as

$ISD^{max} = 2 \text{ x } (T - N) / Su$ (Equation 5.6)

When the speed-up of the switch fabric is 2, Equation 5.6 becomes

$ISD^{max} = (T - N)$ (Equation 5.7)

Hence, the inter-switching delay ISD of A_1 in this case is:

$$\begin{aligned} ISDA_1 &= 2 \text{ x } (T - N) / Su \\ &= 2 \text{ x } (16 - 4) / 2 = 12 \text{ slots} \end{aligned}$$

5.5 QoS SCHEDULER: THE FRAMED VIRTUAL ARRIVAL TIME SCHEDULER

The delay-limiter switch with variable packet size employs a variation on the framed-deadline scheduler as its QoS scheduler. We call this scheduler as the *framed virtual arrival time scheduler* (FVATS).

5.5.1 FVATS OPERATION PRINCIPLES

FVATS assigns a *virtual arrival time* (VAT) to each cell of an arriving packet and serves the packets in the order of their virtual arrival times. Just like FDS stamps the packets with their deadlines, FVATS stamps the packets with their virtual arrival times at the input ports. The VAT of a packet determines the service priority of the packet among other packets so that it does not block more urgent packets and in return it is not delayed more than allowed by the other packets.

Each cell of an arriving packet is assigned a virtual arrival time at the input port and the virtual arrival time of the last cell in a packet is taken as the virtual arrival time of the packet.

When an output port is idle, it selects one of the packets in its per connection queues according to the rules of FVATS and starts transmitting that packet. An output port serves the *eligible* packet with the smallest VAT value. There are two conditions for a packet to be eligible:

1) All cells of a packet must have arrived at the output port before this packet can start being transmitted from the output port. In other words, a packet must be completely switched from its input port to its output port before being eligible for transmission.
2) The difference between the virtual arrival time of a packet and the current system time must be less than T slots. Thus, a packet which is at its output port T slots earlier than its VAT is not eligible.

The first condition assures that only reassembled, complete packets are sent to the transmission lines by the output ports and packets are transmitted as a whole among the nodes through the network. The second condition, together with the traffic arrival shape described in Section 5.3, guarantees that any node does not send cells of data more than 3 times the reserved rate per frame to the downstream node in a frame time of T slots. If a connection does not obey the traffic shape and sends more than its reserved rate, its packets may be dropped due to buffers getting full in the switches on the end-to-end path because of the eligibility concept. This operation punishes the overloading connections and protects the well-behaving connections while providing a bound on buffer size per node to support the guaranteed-service connections with no packet loss provided that they obey the promised traffic shape.

FVATS allows unit-rate connections to send more than one sub-frames of data to the downstream node during a time window of *T*. But in the next node the virtual arrival times of the packets are computed as if the upstream node allowed each connection to send at most one sub-frame in a time window of *T* and a strict traffic arrival shape was preserved throughout the network.

Figure 5.7 shows the virtual arrival time assignment to the packets of connection *A*, which is a unit-rate connection. The sub-frame size *N* is 4 and the number of unit-rate connections simultaneously supported *S* is 3, hence the system frame size *T* is 12 slots. In Figure 5.7-a, the packets arrive at the input port in a proper shape and rate; no more than *N* cells of data arrive in a period of *T* slots. The first frame for connection *A* begins when the first cell of data of connection *A* arrives at the node. The first *N* cells are assigned virtual arrival times which would be their actual arrival times if they arrived at the end of the first frame. The second *N* cells are assigned virtual arrival times which would be their actual arrival times if they arrived at the end of the second frame. Hence, the packets are considered as if they arrive in a strict order and the virtual arrival times are assigned in this way. In Figure 5.7-b, the packets arrive at the input port in an overloaded fashion; more than *N* cells of data arrive in a period of *T* slots. The first frame for connection *A* begins when the first cell of data of connection *A* arrives at the node. During the first 12 slots, the node receives 8 cells of data from connection *A*. Again, the first *N* cells are assigned virtual arrival times which would be their actual arrival times if they arrived at the end of the first frame and the second *N* cells are assigned virtual arrival times which would be their actual arrival times if they arrived at the end of the second frame. Therefore, no matter how frequent the packet arrivals are, the virtual arrival times are assigned in such a way that the service orders of the packets are determined as if they arrive in a proper shape.

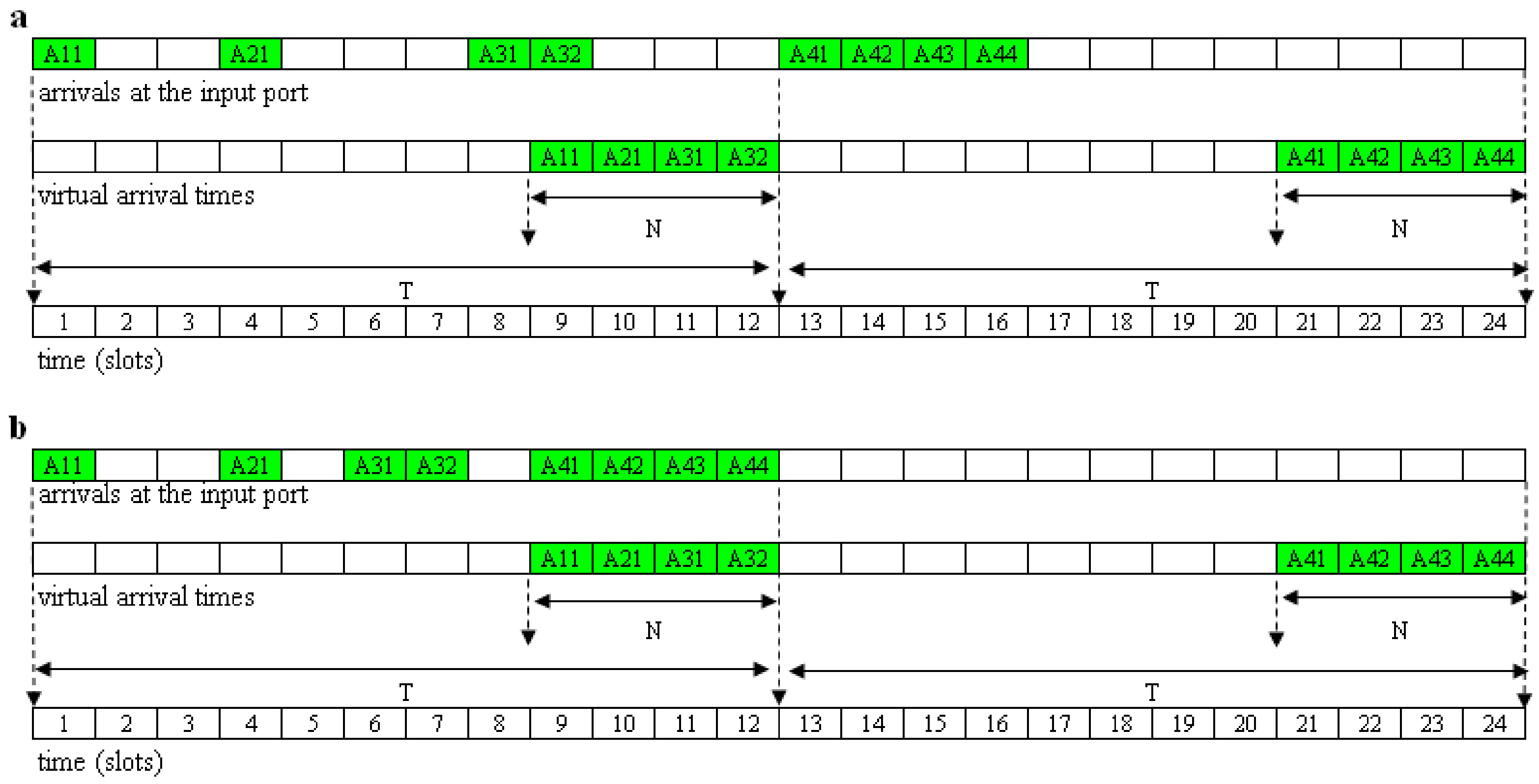


Figure 5.7 Virtual arrival time assignments to packets. (a) Proper arrivals. (b) Overloaded arrivals.

5.5.2 VIRTUAL ARRIVAL TIME COMPUTATION

FVATS assigns a virtual arrival time to each cell of the arriving packets for every connection. The virtual arrival time of a packet is the virtual arrival time of its last cell. To compute the virtual arrival time for a cell of data belonging to a connection, FVATS uses the VAT of the previous cell of data belonging to this connection, the number of cells of data that arrives for the connection and the system constants *N*, *S* and *T*.

FVATS calculates the virtual interarrival times between consecutive packets. The *virtual interarrival time* between two consecutive packets is defined as the number of idle slots to which no cell arrival is assigned according to the VAT computation procedure after the VAT of a packet and before the VAT of the first cell of the next packet. When a new packet arrives, the virtual interarrival time *tvint* is computed by taking the difference of the VAT of the first cell of the new packet and the VAT of the last cell of the previous packet. Actually, these two cells are consecutive cells. Hence, the virtual interarrival time is calculated when the first cell of a packet arrives by using the virtual arrival times of the two consecutive cells without taking the packet number into account as shown below:

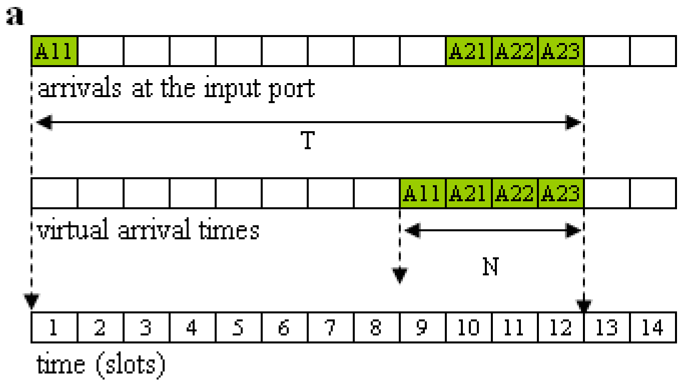
$$tvint_i = VATA_i - VATA_{(i-1)} - 1 \qquad \text{(Equation 5.8)}$$

where $VATA_i$ is the virtual arrival time of the i^{th} cell of data for connection *A*.

There is a *quota counter* C_q for each established connection. C_q counts the arriving cells of data for the corresponding connection. When a new connection is established, its C_q is initialized to 0. With the arrival of one cell of data for the corresponding connection, C_q is incremented by 1. If C_q for a unit-rate connection exceeds N, its value is restored to 1 and this indicates that the corresponding connection has used its entire quota reserved per frame time.

C_q is also used to determine if a packet is in-between or not. When the last cell of a packet arrives at the input port, the size of the packet is compared to the value of C_q. If C_q is greater than or equal to the packet size, this indicates that the packet is not an in-between packet. The maximum value of C_q for a unit-rate connection and the maximum packet size in the system are both equal to N. Therefore, C_q being greater than or equal to the packet size indicates that the packet consumes only a part of the reserved quota per frame time and it fully lies in a single frame. Similarly, if C_q is less than the packet size, this indicates that the packet is an in-between packet. C_q being less than the packet size indicates that this packet involves more cells than C_q and it must have started by using the quota of the previous frame. Thus, the packet lies in two consecutive frames.

Figure 5.8 shows a system where N is 4, S is 3 and T is 12 slots. The first frame for connection A begins when the first cell of the first packet, A_{11}, arrives at the node. There are two packet arrivals in Figure 5.8-a. The total size of these two packets is equal to N, so their virtual arrival times are assigned to be the last N slots of the first frame. They occupy exactly N slots in the output port during their transmission and connection A does not exceed its share of the link rate. Therefore, the packets of connection A do not obstruct the packets of other connections. There are again two packet arrivals in Figure 5.8-b but this time the sum of their sizes exceeds the sub-frame size N. The second packet of connection A is an in-between packet which lies in two consecutive frames. The first packet of connection A is not an in-between packet, so the second packet of connection A is an in-between packet that arrives after a non-in-between packet. The VAT of the first packet A_{11} is 9 as it is in Figure 5.8-a.



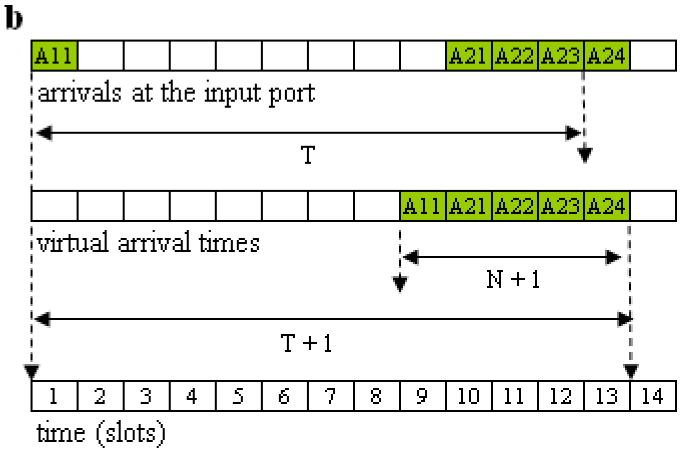


Figure 5.8 Two different packet arrivals. (a) Correct VAT assignment with all packets fully lying in a single frame. (b) Wrong VAT assignment with the second packet lying in two consecutive frames.

If the four cells of the second packet are assigned their virtual arrival times as 10, 11, 12 and 13, the fifth cell of connection *A*, A_{24}, is assigned an earlier VAT than it would have if it was not part of the second packet but was the first cell of a third packet. This may cause connection *A* to exceed its rate, which is *N* slots per *T* slots, and send $(N + 1)$ cells of data in $(T + 1)$ slots. Figure 5.8-b shows wrong VAT assignment to the second packet of connection *A* which is an in-between packet. Although FVATS allows a connection to send more than its reserved rate if there is no other connection to serve, it does not allow the packets to contend with early virtual arrival times in order not to cause unfairness. Therefore, FVATS adds an extra component in the virtual arrival time computation for in-between packets. The details of computing the extra delay for in-between packets are explained in Appendix C.

FVATS keeps the information of whether the previous packet was in-between or not in order to calculate the virtual arrival time of an in-between packet. If the current packet is an in-between packet, then FVATS adds an extra component to the VAT of the last cell of the packet to compute the VAT of the packet. The extra component *Cext* for an in-between packet A_i is computed as follows:

If the previous packet $A_{(i-1)}$ was not an in-between packet,

$$Cext = \max(0, (T - S - 3N + \lceil 3N / Su \rceil + 1) - tvint_i - (N - SA_i)) \quad \text{(Equation 5.9)}$$

If the previous packet $A_{(i-1)}$ was also an in-between packet,

$$Cext = \max(0, (T - N) - tvint_i) \quad \text{(Equation 5.10)}$$

where $tvint_i$ is the virtual interarrival time between $A_{(i-1)}$ and A_i and SA_i is the size of A_i in cells. (see Appendix C for details)

5.5.2.1 VAT COMPUTATION PROCEDURE

The VAT computation procedure of FVATS is described by the state diagram in Figure 5.9.

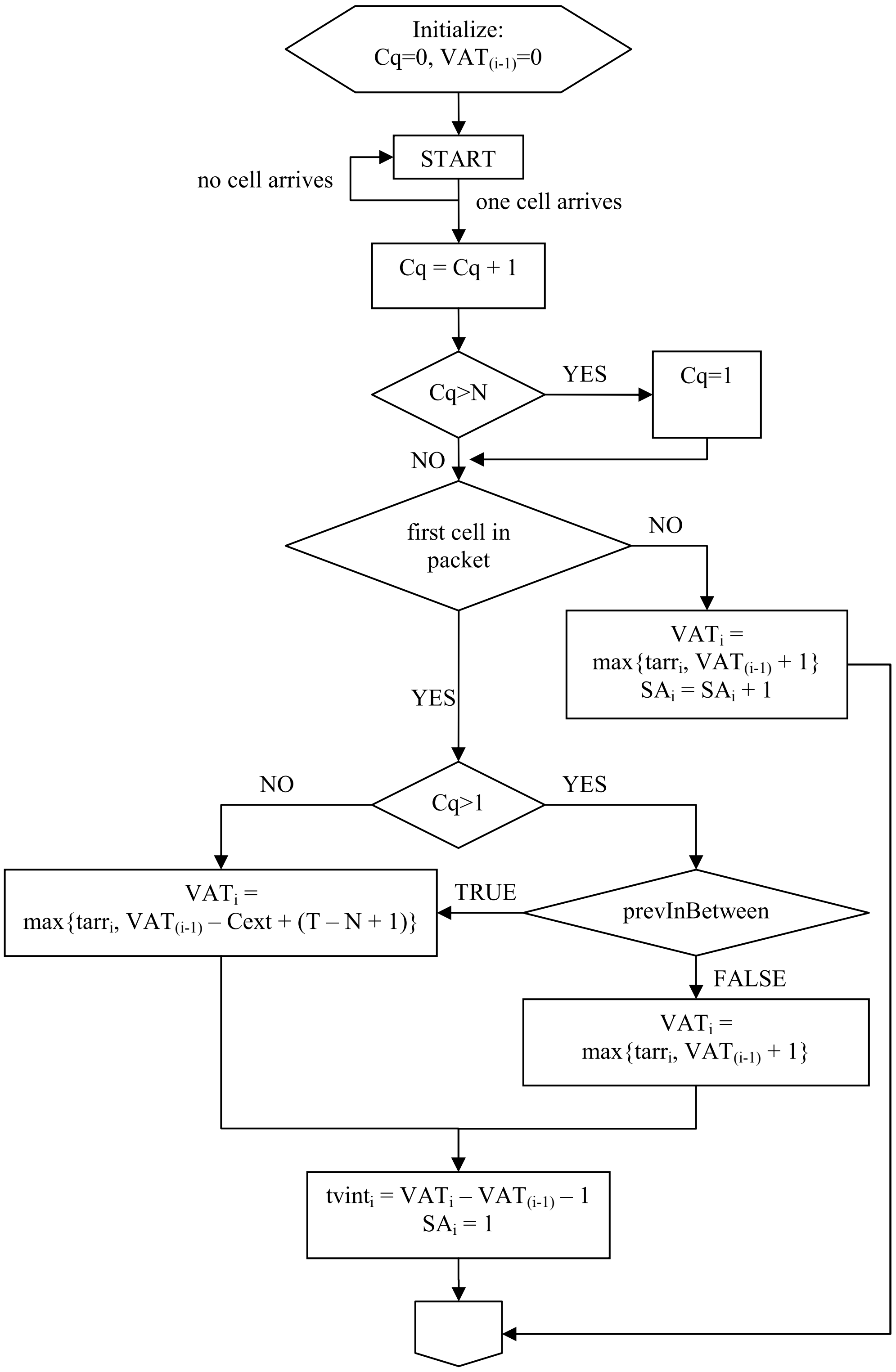
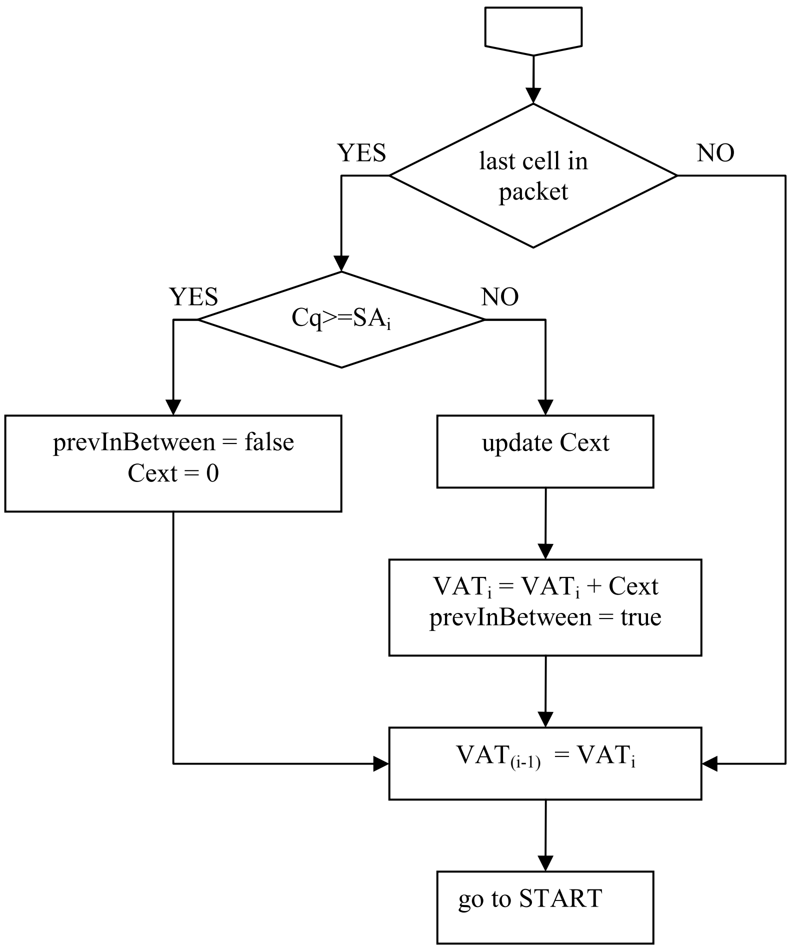

Initialize:
Cq=0, VAT(i-1)=0
START
no cell arrives
one cell arrives
Cq = Cq + 1
Cq>N
YES
Cq=1
NO
first cell in packet
NO
VATi = max{tarri, VAT(i-1) + 1} SAi = SAi + 1
YES
NO
Cq>1
YES
VATi = max{tarri, VAT(i-1) – Cext + (T – N + 1)}
TRUE
prevInBetween
FALSE
VATi = max{tarri, VAT(i-1) + 1}
tvinti = VATi – VAT(i-1) – 1 SAi = 1

Figure 5.9 State diagram for VAT computation



update Cext method:

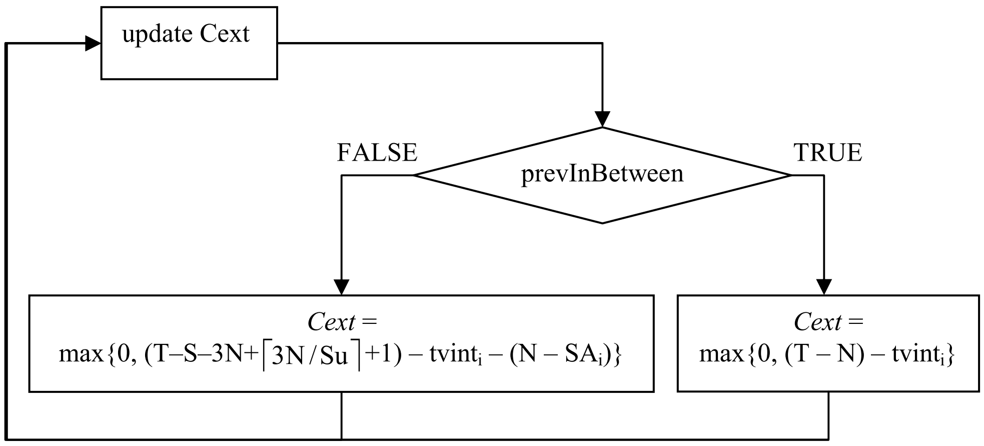

update Cext
FALSE
prevInBetween
TRUE
Cext = max{0, (T–S–3N+⌈3N / Su⌉+1) – tvint_i – (N – SA_i)}
Cext = max{0, (T – N) – tvint_i}

Figure 5.9 (continued)

The definitions of the variables and methods employed in the state diagram are defined as follows:

Cq: quota counter
Cext: extra component added to compute the VAT of an in-between packet
prevInBetween: variable to show whether the previous packet was an in-between packet; true if the previous packet was an in-between packet, false if the previous packet was not an in-between packet
SA_i: the size of the packet that i^{th} cell belong to in cells
tarr: actual arrival time of the cell
$tvint_i$: the virtual interarrival time between $A_{(i-1)}$ and A_i
update *Cext*: computes and updates *Cext* according to Equation 5.9 and Equation 5.10 as shown in Figure 5.9
VAT_i: the virtual arrival time of the i^{th} cell of the current connection

Due to the shaper operation and the VAT assignment procedure of FVATS, the virtual arrival time of any packet at the first node after the shaper is bounded by:

$$AATA_{i1} \leq VATA_{i1} \leq AATA_{i1} + (T - N) \qquad \text{(Equation 5.11)}$$

where $AATA_{ij}$ is the actual arrival time of the i^{th} packet at the j^{th} node which shows the arrival of a packet at a node in real time.

The lower bound that $AATA_{i1} \leq VATA_{i1}$ is trivial because FVATS does not assign a virtual arrival that is smaller than its actual arrival time to any packet. Figure 5.10 illustrates a system in which $N = 3$, $S = 3$ and $T = 9$. A new service frame for connection A in the shaper starts with the departure of the first cell of packet A_1 from the shaper. A_1 arrives immediately at the first node and a new frame starts in the first node upon its arrival. Packet A_2 which consists of only one cell (A_{21}) is sent out of the traffic shaper in the last slot of the first service frame in Figure 5.10. A_2 arrives at the first node in the last slot of the first frame of node 1. Hence, A_2 is assigned a VAT which is equal to its actual arrival time ($AATA_{21} = VATA_{21}$). Packets A_1 and A_3 are transmitted in the first slots of their respective service frames from the shaper. They also arrive in the first slots of their respective frames in the first node. Thus, FVATS assigns them virtual arrival times in the last N slots of their respective frames in the first node. FVATS assigns to the first cell in a frame a virtual arrival time in the $(T - N + 1)^{th}$ slot of the corresponding frame and to the next cells the consecutive slots in the same frame if the actual arrival times of the cells are not greater than these slots. Otherwise, it assigns the actual arrival time of the cell as its virtual arrival time. Therefore, the upper bound for the VAT of a packet is $(T - N)$ slots after its actual arrival time ($VATA_{i1} \leq AATA_{i1} + (T - N)$).

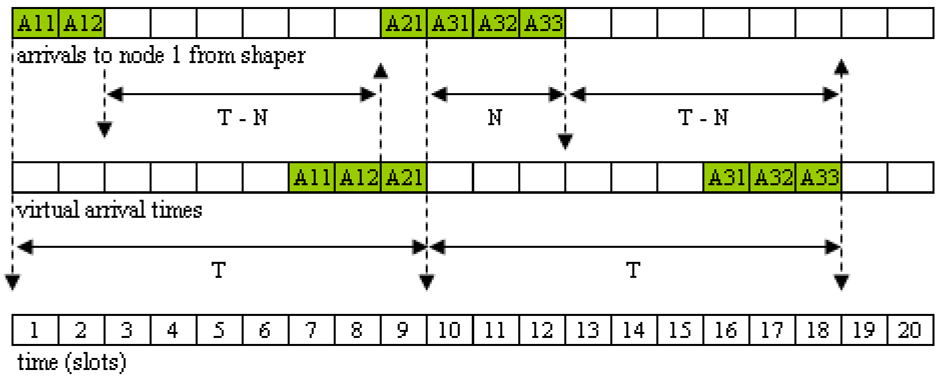


Figure 5.10 Virtual arrival time assignment at the first node after the traffic shaper.

5.5.3 VIRTUAL NODE DELAY

The *node delay* of a packet is the time between the actual arrival time of the last cell of the packet to the node and the departure of the last cell of the packet from the node. The *actual arrival time* of a packet at a node is defined as the real time in slots in which the last cell of the packet arrives at the node.

Similarly, the *virtual node delay* of a packet is the time between the virtual arrival time of the packet in the node and the departure of the last cell of the packet from the node. The departure time of a packet from a node is defined as the departure time of the last cell of this packet from the node. The maximum virtual node delay of a packet occurs when the last cell of the packet departs from the output port at the latest time within the possible limits.

The virtual node delay of packet A_i in the h^{th} node on its end-to-end path is given by:

$$VNDA_{ih} = tdepA_{ih} - VATA_{ih} \qquad \text{(Equation 5.12)}$$

where $VNDA_{ih}$ is the virtual node delay of packet A_i in the h^{th} node, $tdepA_{ih}$ is the departure time of packet A_i from the h^{th} node and $VATA_{ih}$ is the virtual arrival time of packet A_i in the h^{th} node.

A packet experiences the maximum virtual node delay when the maximum inter-switching delay occurs for that packet and the output is busy sending another packet of another connection at the time that packet arrives at the output port. The maximum inter-switching delay in a system with $Su = 2$ is shown to be equal to $(T - N)$ by Equation 5.7.

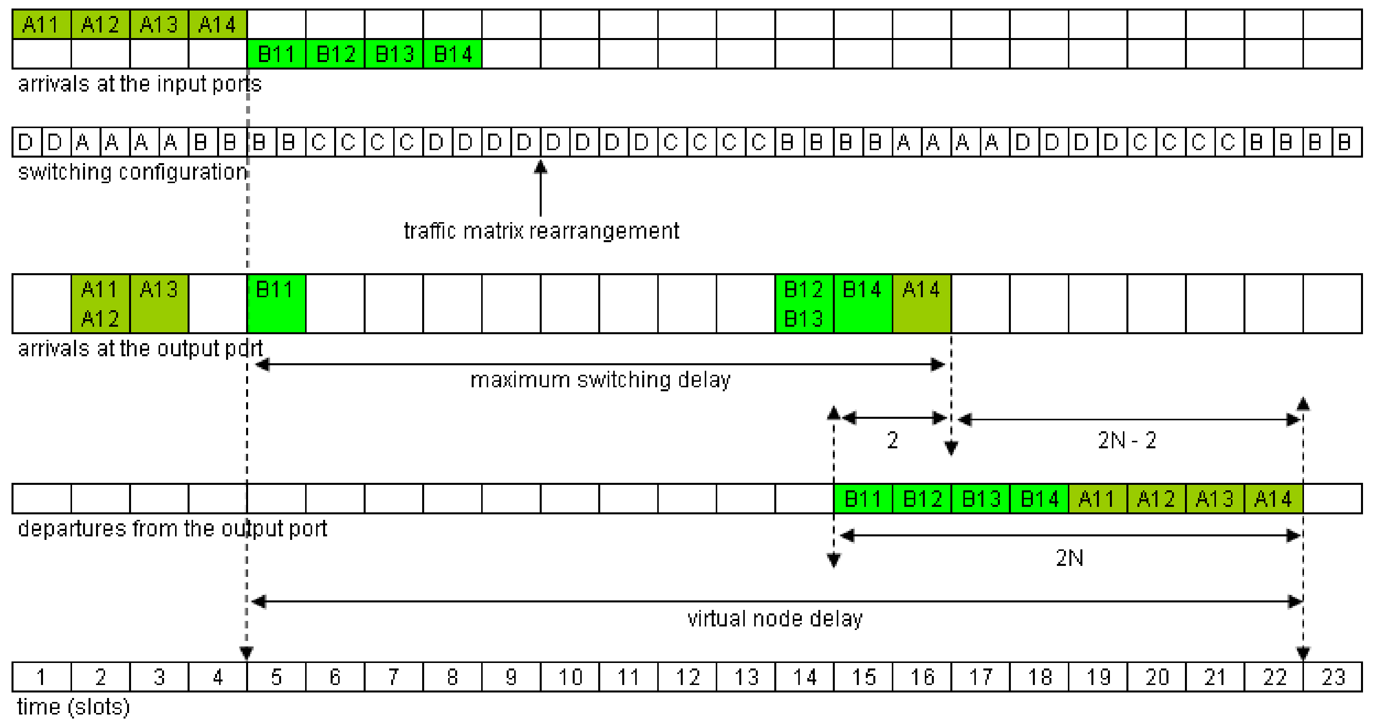

A11 A12 A13 A14
B11 B12 B13 B14
arrivals at the input ports
D D A A A A B B B B C C C C D D D D D D D D C C C C B B B B A A A A D D D D C C C C B B B B
switching configuration
traffic matrix rearrangement
A11 A12 A13 B11 B12 B13 B14 A14
arrivals at the output port
maximum switching delay
2
2N - 2
B11 B12 B13 B14 A11 A12 A13 A14
departures from the output port
2N
virtual node delay
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
time (slots)

Figure 5.11 Maximum virtual node delay for packet A_1.

Figure 5.11 above illustrates the maximum virtual node delay of packet A_1 due to TM rearrangement and a packet of connection *B*. The virtual arrival time of A_1 is equal to its actual arrival time. Packet A_1 is exposed to the maximum inter-switching delay as described in Section 5.4.1 and completes its switching to the output port in slot 16. It cannot be transmitted immediately because the output port has already started sending another packet, B_1. Thus, A_1 has to wait until the end of the transmission of B_1. This waiting time is maximum when the length of B_1 is *N* and the output port starts transmitting B_1 one slot before A_1 is completely switched to the output port. If the output port did not start transmitting B_1 before A_1 is completely switched to the output port, then B_1 would have to contend with A_1 for the transmission order. In this case, it is desired that B_1 wins the contention so that A_1 waits for B_1 to leave the output port. For B_1 to win the contention, it must have a VAT smaller than or equal to that of A_1. This requires B_1 to have arrived at the node before or at the same time A_1 arrives at the node in slot 4. Because the fabric schedule is rearranged to give A_1 the maximum inter-switching delay, every other connection is scanned at least once before the rearrangement that takes place after slot 9. Therefore, B_1 would be completely switched to the output port before slot 10 if it arrived before slot 4. The same case applies for all other connections that will contend with A_1 in slot 16. Hence, any packet belonging to the other connections cannot win the contention against A_1 in slot 16 and the maximum time A_1 has to wait after it is fully switched to the output port occurs if the output port has already begun to transmit another packet of length *N*. Since the first cell of B_1 is already sent out before slot 16 and the second cell of B_1 is being transmitted during slot 16, A_1 waits for the transmission of the remaining $(N - 2)$ cells of B_1. It takes *N* slots to transmit A_1 itself. Therefore, there are $(N - 2) + N = (2N - 2)$ slots between the arrival of the last cell of A_1 at the output port and the departure of the last cell of A1 from the output port. A_1 experiences a maximum inter-switching delay of $(T - N)$ slots until the end of slot 16 and another ($2N - 2$) slots between slot 16 and the slot in which its last cell departs from the node. Hence, the maximum virtual node delay of a packet A_i due to TM rearrangement and other connections in the h^{th} node on its path, $VNDA_{ih}^{max}$, can be expressed as follows:

$$\begin{aligned} VNDA_{ih}^{max} &= (T - N) + (2N - 2) \\ &= T + N - 2 \end{aligned} \qquad \text{(Equation 5.13)}$$

Lemma 5.1: The maximum departure time of a packet A_i from any node h is given by:

$$tdepA_{ih}^{max} = VATA_{ih} + (T + N - 2) \qquad \text{(Equation 5.14)}$$

where $tdepA_{ih}^{max}$ shows the latest departure time of the packet A_i from the h^{th} node on its end-to-end path which is guaranteed by FVATS.

Proof:

Substituting Equation 5.13 into Equation 5.12 and rearranging yields

$$tdepA_{ih}^{max} = VATA_{ih} + VNDA_{ih}^{max} \qquad \text{(Equation 5.15)}$$

Lemma 5.2: If a non-in-between packet is transmitted from a node experiencing the maximum virtual node delay, then its virtual arrival time at the next node can be its actual arrival time at that node plus ($T - S - 3N + \lceil 3N / Su \rceil + 1$) if the previous packet is an in-between packet.

Proof:

Suppose the virtual arrival time of an in-between packet A_i is computed by adding an extra component of $Cext_h = (T - S - 3N + \lceil 3N/Su \rceil + 1)$ in the h^{th} node on its path. If $A_{(i+1)}$ is non-in-between, then the virtual interarrival time between A_i and $A_{(i+1)}$ is equal to $T - N - Cext_h$.

Suppose $A_{(i-1)}$ is transmitted from node h earlier than its maximum virtual node delay while A_i and $A_{(i+1)}$ are sent from the same node with experiencing the maximum virtual node delay. Then, in the $(h+1)^{\text{th}}$ node $Cext_{(h+1)}$ for A_i becomes 0 since the virtual interarrival time between $A_{(i-1)}$ and A_i at the $(h+1)^{\text{th}}$ node is greater than T. Then, the virtual arrival time of A_i at the $(h+1)^{\text{th}}$ node equals its actual arrival time. Since $Cext_{(h+1)}$ is 0, the virtual interarrival time between A_i and $A_{(i+1)}$ at the $(h+1)^{\text{th}}$ node is computed to be $T - N - Cext_{(h+1)} = T - N$. Since it was $T - N - Cext_h$ in the h^{th} node and it is $T - N$ in the $(h+1)^{\text{th}}$ node, the virtual arrival time of $A_{(i+1)}$ at the $(h+1)^{\text{th}}$ node is $Cext_h$ slots later than its arrival time at the $(h+1)^{\text{th}}$ node where $Cext_h = (T - S - 3N + \lceil 3N/Su \rceil + 1)$.

Lemma 5.3: If a packet is transmitted from a node experiencing the maximum virtual node delay, then its virtual arrival time at the next node equals its actual arrival time at that node if the previous packet is not an in-between packet.

If $tdepA_{ih} = tdepA_{ih}^{max} = VATA_{ih} + (T + N - 2)$ then

$$VATA_{i(h+1)} = AATA_{i(h+1)} \qquad \text{(Equation 5.16)}$$

where $AATA_{ih}$ indicates the actual arrival time of packet A_i at the h^{th} node.

Proof:

Suppose packet A_i departs from the h^{th} node X slots before its maximum virtual node delay:

$$tdepA_{ih} = tdepA_{ih}^{max} - X \qquad \text{(Equation 5.17)}$$

by Equation 5.14, Equation 5.17 becomes

$$tdepA_{ih} = VATA_{ih} + (T + N - 2) - X \qquad \text{(Equation 5.18)}$$

FVATS assures that if two packets $A_{(i\text{-}1)}$ and A_i arrive in different frames, then their virtual arrival times are separated by at least $(T - N - SA_i)$ slots where SA_i is the size of the latter packet. Therefore,

$$VATA_{(i\text{-}1)h} \leq VATA_{ih} - (T - N + SA_i) \qquad \text{(Equation 5.19)}$$

Applying Equation 5.14, Equation 5.19 becomes

$$tdepA_{(i\text{-}1)h}^{max} \leq VATA_{ih} - (T - N + SA_i) + (T + N - 2) \qquad \text{(Equation 5.20)}$$

When $A_{(i\text{-}1)}$ experiences maximum virtual node delay in the h^{th} node, its departure time equals its maximum departure time:

$$\begin{aligned} tdepA_{(i\text{-}1)h} &= tdepA_{(i\text{-}1)h}^{max} \\ &= VATA_{ih} - (T - N + SA_i) + (T + N - 2) \qquad \text{(Equation 5.21)} \end{aligned}$$

The difference between the departure times of A_i and $A_{(i\text{-}1)}$ from the h^{th} node is:

$$\begin{aligned} tdepA_{ih} - tdepA_{(i\text{-}1)h} &= [VATA_{ih} + (T + N - 2) - X] - [VATA_{ih} - (T - N + SA_i) + \\ &\quad (T + N - 2)] \\ &= (T - N + SA_i) - X \qquad \text{(Equation 5.22)} \end{aligned}$$

Since $A_{(i\text{-}1)}$ experiences maximum virtual node delay in node h, by Lemma 5.3

$VATA_{(i-1)(h+1)} = AATA_{(i-1)(h+1)}$ (Equation 5.23)

Neglecting the propagation delay, the departure time of $A_{(i-1)}$ from the $(h-1)^{th}$ node equals its actual arrival time at the h^{th} node:

$tdepA_{(i-1)h} = AATA_{(i-1)(h+1)}$ (Equation 5.24)

Combining Equation 5.23 and Equation 5.24,

$tdepA_{(i-1)h} = VATA_{(i-1)(h+1)}$ (Equation 5.25)

Putting Equation 5.25 into Equation 5.22 yields

$tdepA_{ih} = VATA_{(i-1)(h+1)} + (T - N + SA_i) - X$ (Equation 5.26)

Neglecting the propagation delay, the actual arrival time of A_i at the $(h+1)^{th}$ node equals its departure time from the h^{th} node:

$AATA_{i(h+1)} = VATA_{(i-1)(h+1)} + (T - N + SA_i) - X$ (Equation 5.27)

$VATA_{i(h+1)}$ must be $(T - N + SA_i)$ slots after the VAT of $A_{(i-1)}$ in the $(h+1)^{th}$ node because $tdepA_{ih}$ is X slots earlier than $tdepA_{ih}^{max}$ and $tdepA_{(i-1)h}$ equal to $tdepA_{(i-1)h}^{max}$:

$VATA_{i(h+1)} = VATA_{(i-1)(h+1)} + (T - N + SA_i)$ (Equation 5.28)

By using Equation 5.27 and Equation 5.28 the difference between the virtual arrival time and actual arrival time of A_i equals:

$VATA_{i(h+1)} - AATA_{i(h+1)} = X$ (Equation 5.29)

Therefore, if packet A_i departs from the h^{th} node X slots before its maximum virtual node delay, then its VAT can be X slots later than its actual arrival time in the $(h + 1)^{th}$ node. If $X = 0$, then

$VATA_{i(h+1)} = AATA_{i(h+1)}$ (Equation 5.30)

which satisfies Lemma 5.3.

Theorem 5.1:

The end-to-end delay bound $dbe2eA_i$ provided by FVATS for any packet A_i on an H hop path is given by

$dbe2eA_i = (H + 1) \text{ x } (T + N - 2) - 2 \text{ x } (N - 1) + (T - S - 3N + \lceil 3N / Su \rceil + 1).$

(Equation 5.31)

Proof:

If A_i experiences maximum virtual node delay in every node on its end-to-end path, then by Equation 5.16 for all node h on its end-to-end path

$tdepA_{ih} = VATA_{ih} + (T + N - 2)$ (Equation 5.32)

By Lemma 5.3, for all node h on the end-to-end path of A_i

$VATA_{i(h+1)} = AATA_{i(h+1)} = tdepA_{ih}$ (Equation 5.33)

By Equation 5.32 and Equation 5.33 for all nodes on the end-to-end path of A_i

$VATA_{ih} = VATA_{i(h-1)} + (T + N - 2)$ (Equation 5.34)

Similarly,

$VATA_{i(h-1)} = VATA_{i(h-2)} + (T + N - 2)$ (Equation 5.35)

Iteratively tracing by substituting Equation 5.35 into Equation 5.34 leads to

$VATA_{i(h-m)} = VATA_{i(h-m-1)} + (T + N - 2)$ (Equation 5.36)

which is equal to

$$\begin{aligned} VATA_{i(h-m)} &= VATA_{i(h-k)} + ((h - m) - (h - k)) \text{ x } (T + N - 2) \\ &= VATA_{i(h-k)} + (k - m) \text{ x } (T + N - 2) \end{aligned}$$ (Equation 5.37)

The departure time $tdepA_{iH}$ of A_i from the last node equals

$tdepA_{iH} = VATA_{iH} + (T + N - 2)$ (Equation 5.38)

Therefore, on the H hop end-to-end path from the first node where $(h - k) = 1$ to the last node where $(h - m) = H$, the departure time of A_i from the H^{th} node is given by

$$\begin{aligned} tdepA_{iH} &= VATA_{iH} + (T + N - 2) \\ &= VATA_{i1} + (H - 1) \text{ x } (T + N - 2) + (T + N - 2) \\ &= VATA_{i1} + H \text{ x } (T + N - 2) \end{aligned}$$ (Equation 5.39)

If the packet is a non-in-between packet after an in-between packet and it is transmitted from a node experiencing the maximum virtual node delay as described in Lemma 5.2, then its virtual arrival time at the next node can be its actual arrival time at that node plus ($T - S - 3N + \lceil 3N/Su \rceil + 1$). Therefore, when this component is added to Equation 5.39, the departure time of A_i from the H^{th} node is given by:

$$tdepA_{iH} = VATA_{i1} + H \text{ x } (T + N - 2) + (T - S - 3N + \lceil 3N/Su \rceil + 1)$$

(Equation 5.40)

Since $VATA_{i1} \leq AATA_{i1} + (T - N)$ by Equation 5.11, Equation 5.40 becomes

$$tdepA_{iH} \leq AATA_{i1} + (T - N) + H \text{ x } (T + N - 2) + (T - S - 3N + \lceil 3N/Su \rceil + 1)$$

(Equation 5.41)

The end-to-end delay bound $dbe2eA_i$ of a packet A_i which is the difference between the maximum departure time of A_i from the H^{th} node and the actual arrival of A_i at the first node is given by

$$dbe2eA_i = tdepA_{iH} - AATA_{i1}$$ (Equation 5.42)

which can be upper-bounded by using Equation 5.41 as

$$dbe2eA_i \leq AATA_{i1} + (T - N) + H \text{ x } (T + N - 2) + (T - S - 3N + \lceil 3N/Su \rceil + 1) - AATA_{i1}$$

$$dbe2eA_i \leq (T - N) + H \text{ x } (T + N - 2) + (T - S - 3N + \lceil 3N/Su \rceil + 1)$$

$$dbe2eA_i \leq (H + 1) \text{ x } (T + N - 2) - 2 \text{ x } (N - 1) + (T - S - 3N + \lceil 3N/Su \rceil + 1)$$

(Equation 5.43)

When the fabric speed-up $Su = 2$ and $N = 1$, T equals S and the end-to-end delay bound for FVATS reduces to the end-to-end delay bound for FDS which is:

$dbe2eA_i \leq (H + 1) \text{ x } (T + 1 - 2) - 2 \text{ x } (1 - 1) + (T - S - 3N + \lceil 3/Su \rceil + 1)$

$dbe2eA_i \leq (H + 1) \text{ x } (T - 1) + (\lceil 3/Su \rceil - 2)$

$dbe2eA_i \leq (H + 1) \text{ x } (T - 1)$ (Equation 5.44)

5.5.4 REQUIRED BUFFER SIZE

The maximum virtual node delay of a packet A_i due to TM rearrangement and other connections in the h^{th} node on its path, $VNDA_{ih}^{max}$, is expressed, by Equation 5.12, as follows:

$VNDA_{ih}^{max} = T + N - 2$ (Equation 5.45)

Therefore, a node can send a packet whose virtual arrival time is at most $T + N - 2$ slots before the current time tc and any packet with an earlier VAT must have already transmitted from the node since FVATS assures that the virtual node delay of any packet is less than or equal to the maximum virtual node delay. Hence, at any time ta a node can send a packet to the downstream node if its virtual arrival time $VATA_i$ satisfies the following equation:

$VATA_i \geq ta - (T + N - 2)$ (Equation 5.46)

As stated in Section 5.5.1, FVATS serves a packet if the difference between the virtual arrival time of the packet and the current system time is less than T slots. Therefore, at any time ta a node can send a packet to the downstream node if its virtual arrival time $VATA_i$ satisfies the following equation:

$$VATA_i < ta + T \qquad \text{(Equation 5.47)}$$

Consequently, combining Equation 5.46 and Equation 5.47, a node can send packets to the downstream node at any time ta whose virtual arrival times satisfy the following equation:

$$ta - (T + N - 2) \le VATA_i < ta + T \qquad \text{(Equation 5.48)}$$

Equation 5.48 indicates that a node can transmit packets to the downstream node whose virtual arrival times can be in a range of $(2T + N - 2)$ slots. Since FVATS assigns virtual arrival times to the packets such that at most N one-cell packets can have virtual arrival times within a period of T slots, the maximum number of cells whose virtual arrival times are in a period of $(2T + N - 2)$ slots, Np, is given by:

$$Np = \lceil (2T + N - 2)/T \rceil \times N \qquad \text{(Equation 5.49)}$$

When $N > 2$,

$$Np = 3N \qquad \text{(Equation 5.50)}$$

As a result, the required buffer size for DLS/VPS to operate with no packet loss, BS, is bounded and given by:

$$\begin{aligned} BS &= 3NSC \\ &= 3TC \end{aligned} \qquad \text{(Equation 5.51)}$$

where C is the link rate.

5.6 DLS/VPS PERFORMANCE

This section presents the simulation results that demonstrate the average performance of the delay-limiter switch with variable packet size. The simulation results reflect the performance regarding the switching and scheduling of the packets. Any delay overheads of the guaranteed-service traffic due to traffic shaping at the network edge are not included in the simulation results.

For the experiments, we use our own simulator which is built in a similar manner with the same architecture used to perform the experiment in [2]. The simulation program is coded in C++.

There are 10 nodes connected in a linear fashion (H = 10) in our network topology. All links are assumed to have the same speed. Hence, a frame time is T slots on all of the links. There is a unit rate connection set up for each input/output port pair of each node. Hence, each output port receives packets from T distinct input ports. In this case, the number of packets that can arrive at an output port from distinct input ports at the same time maximizes the queuing delay and generates a worst case scenario for our experiments. Traffic sources for connection-oriented traffic are attached to the input ports of the nodes in parking-lot fashion through these nodes (Figure 4.1). We observe the performance of the connection-oriented traffic which go through from source node 1 to destination node 10. This end-to-end traffic competes with the background traffic which starts at each node and terminates at the following node. The foreground connection-oriented traffic load is kept constant at 100% while the background connection-oriented traffic load varies between 10% and 100% in all experiments.

There are four groups of experiments carried out with different frame and sub-frame sizes.

In the first group, the maximum packet size is 10 cells hence the sub-frame size *N* is 10 slots. The maximum number of unit-rate connection that can be simultaneously supported, *S*, is 50 and the frame size *T* is 500 slots. The switch size is 50x50. There are four different types of packet lengths. The shortest packet consists of one single cell and the largest packet involves 10 cells. The sizes of the two intermediate packets are 4 and 5 cells.

In the second group, the maximum packet size is 20 cells and the sub-frame size *N* is 20 slots. The maximum number of unit-rate connection that can be simultaneously supported, *S*, is again 50, thus the frame size *T* is 1000 slots. The switch size is 50x50. There are four different types of packet lengths. The shortest packet consists of one single cell and the largest packet involves 20 cells. The sizes of the two intermediate packets are 8 and 10 cells.

In the third group of experiments, the maximum packet size is 10 cells. Consequently, the sub-frame size *N* is 10 slots. The maximum number of unit-rate connection that can be simultaneously supported, *S*, is 100 and the frame size *T* is 1000 slots. The switch size is 100x100. There are four different types of packet lengths. The length of the shortest packet is 1 cell and the largest packet size is 10 cells. The sizes of the two intermediate packets are 4 and 5 cells.

In the fourth group, the maximum packet size is 20 cells. Therefore, the sub-frame size *N* is 20 slots. The maximum number of unit-rate connection that can be simultaneously supported, *S*, is 100 and the frame size *T* is 2000 slots. The switch size is 100x100. There are four different types of packet lengths. The length of the shortest packet is 1 cell while the length of the largest packet is 20 cells. The sizes of the two intermediate packets are 8 and 10 cells.

5.6.1 EXPERIMENTS AND RESULTS

Figure 5.12 shows the average end-to-end delay of the foreground connection-oriented traffic packets where $N = 10$, $S = 50$ and $T = 500$. Therefore, the end-to-end delay bound for the foreground connection-oriented traffic is 5500 slots. The simulated switches have 50 input ports and 50 output ports.

Figure 5.13 shows the average end-to-end delay of the foreground connection-oriented traffic packets normalized with the end-to-end delay bound for the same experiment in Figure 5.12.

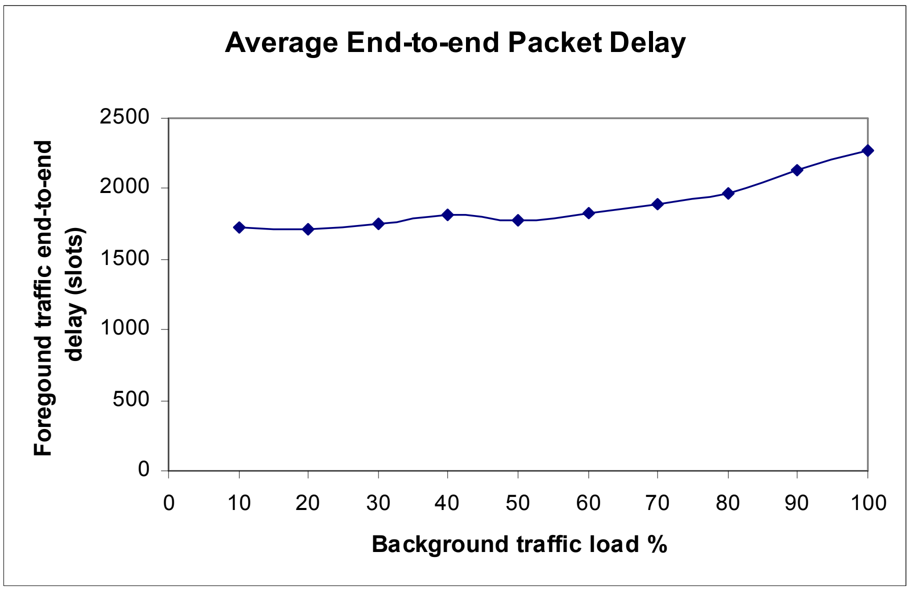


Figure 5.12 Average end-to-end delay for connection-oriented traffic with N = 10, S = 50 and T = 500.

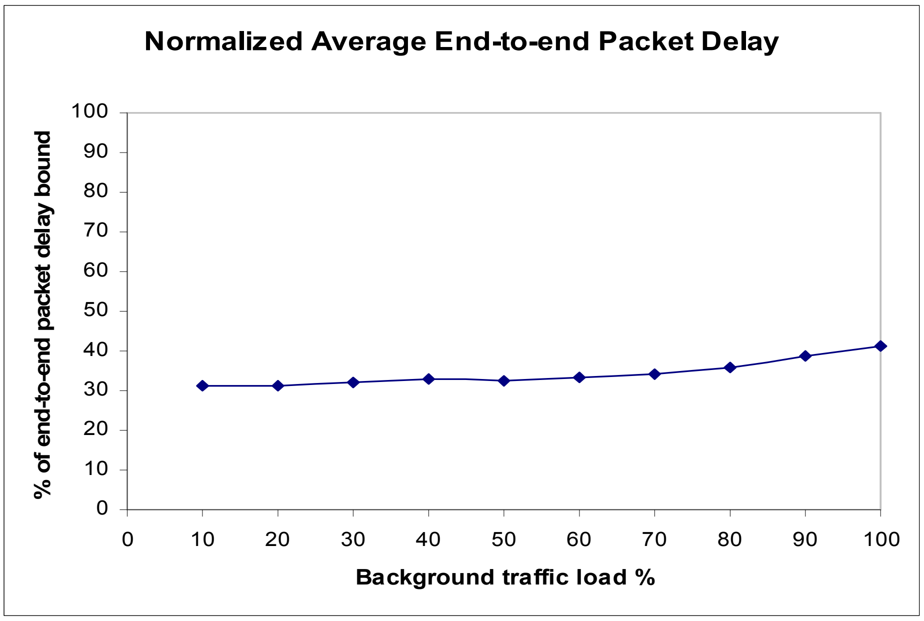


Figure 5.13 Normalized average end-to-end delay for connection-oriented traffic with N = 10, S = 50 and T = 500.

Figure 5.14 shows the average end-to-end delay of the foreground connection-oriented traffic packets when N = 20, S = 50 and T = 1000. The end-to-end delay bound for the connection-oriented traffic is 11000 slots. The simulated switch sizes are 50x50.

Figure 5.15 shows the average end-to-end delay of the foreground connection-oriented traffic packets normalized with the end-to-end delay bound for the same experiment as in Figure 5.14.

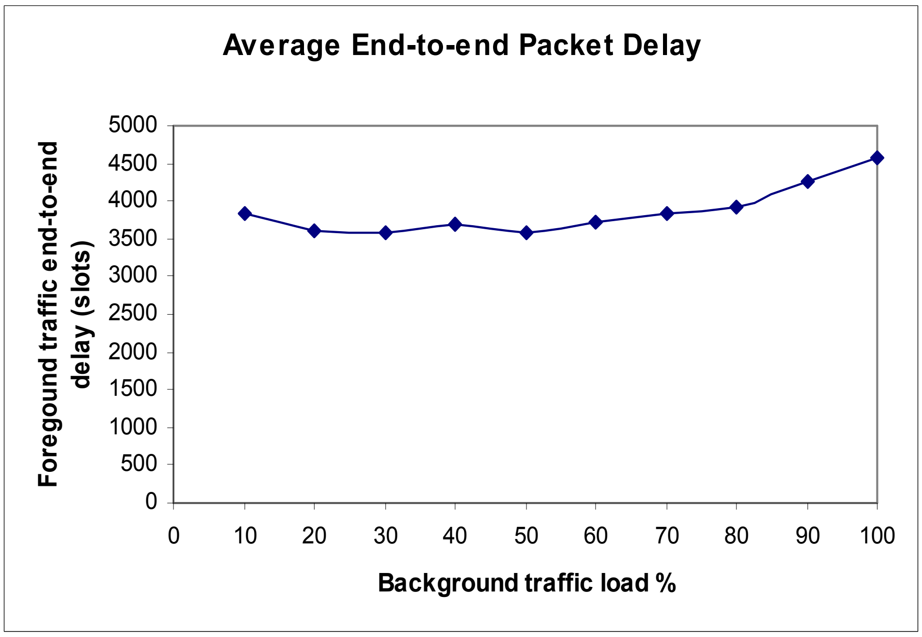


Figure 5.14 Average end-to-end delay for connection-oriented traffic with N = 20, S = 50 and T = 1000.

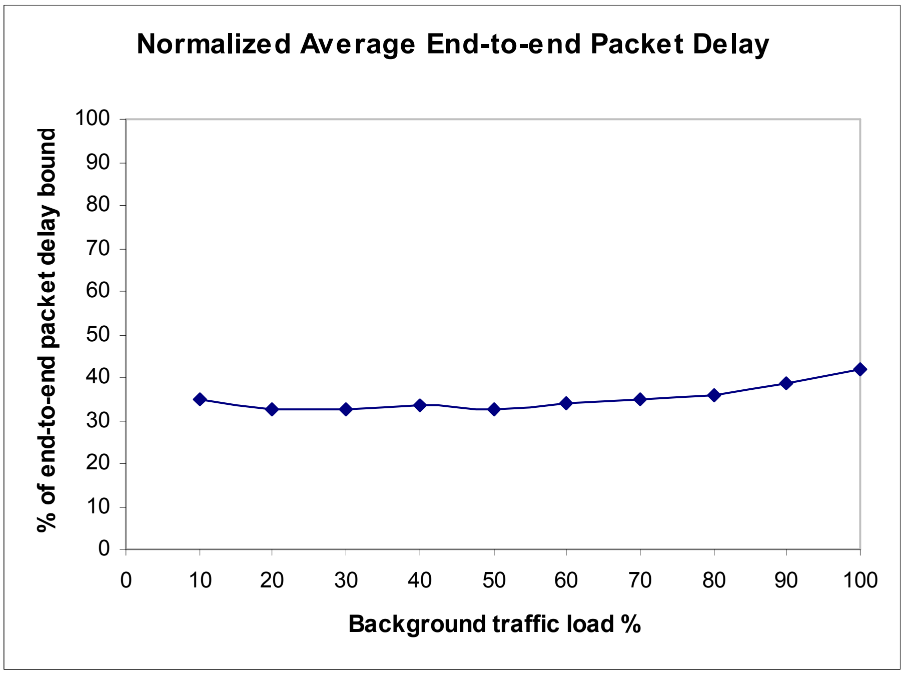


Figure 5.15 Normalized average end-to-end delay for connection-oriented traffic with $N = 20$, $S = 50$ and $T = 1000$.

Figure 5.16 shows the average end-to-end delay of the foreground connection-oriented traffic packets when $N = 10$, $S = 100$ and $T = 1000$. Consequently, the end-to-end delay bound for the connection-oriented traffic is 11000 slots. The simulated switch sizes are 100x100.

Figure 5.17 shows the average end-to-end delay of the foreground connection-oriented traffic packets normalized with the end-to-end delay bound for the same experiment as in Figure 5.16.

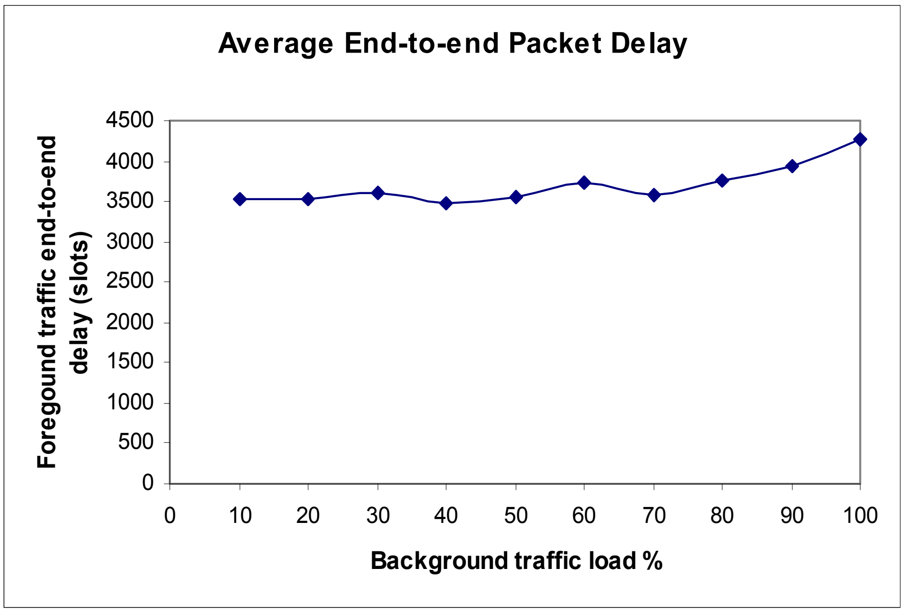


Figure 5.16 Average end-to-end delay for connection-oriented traffic with $N = 10$, $S = 100$ and $T = 1000$.

.

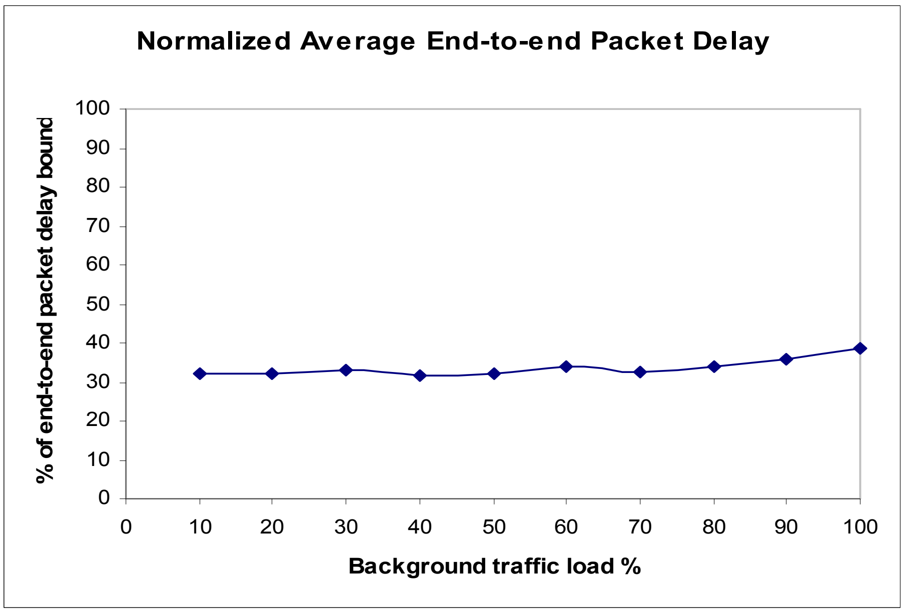


Figure 5.17 Normalized average end-to-end delay for connection-oriented traffic with $N = 10$, $S = 100$ and $T = 1000$.

Figure 5.18 shows the average end-to-end delay of the foreground connection-oriented traffic packets when N = 20, S = 100 and T = 2000. Therefore, the end-to-end delay bound for the connection-oriented traffic is 22000 slots. The simulated switch have 100 input ports and 100 output ports.

Figure 5.19 shows the average end-to-end delay of the foreground connection-oriented traffic packets normalized with the end-to-end delay bound for the same experiment as in Figure 5.18.

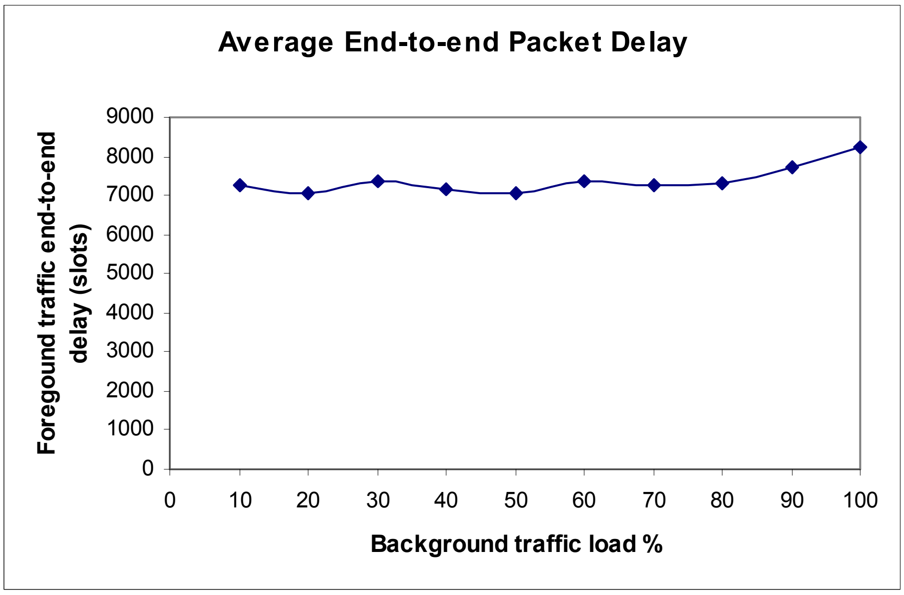


Figure 5.18 Average end-to-end delay for connection-oriented traffic with N = 20, S = 100 and T = 2000.

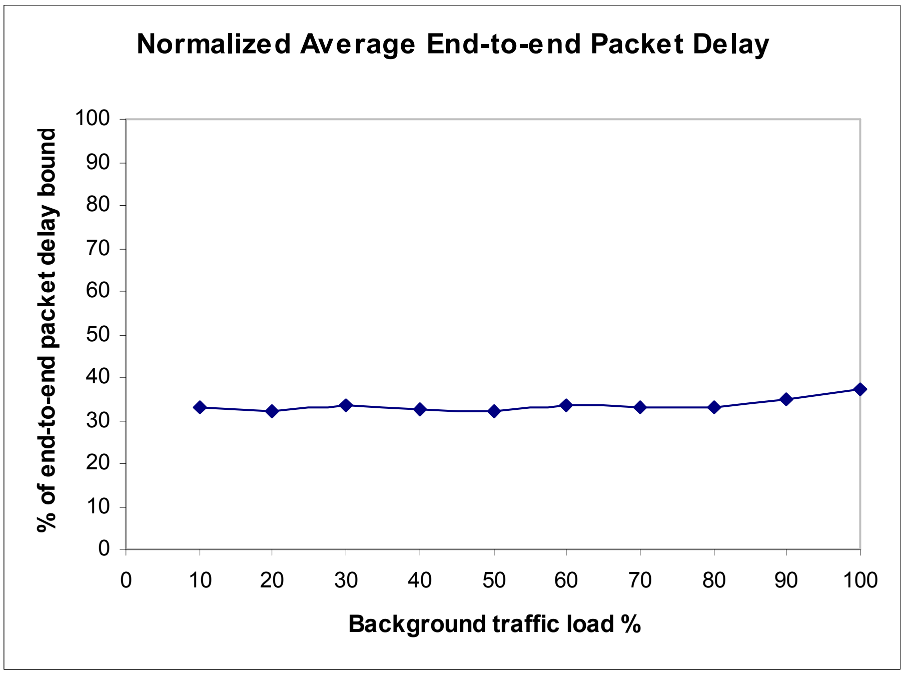


Figure 5.19 Normalized average end-to-end delay for connection-oriented traffic with $N = 20$, $S = 100$ and $T = 2000$.

The simulation results show that the average end-to-end delay performance of the DLS/VPS is less than half of the guaranteed delay bounds even when the network is fully utilized. The ratio of the average end-to-end delay to the given bound is around 30% when the background traffic load is 10%. As the background traffic utilization increases, the average end-to-end delay of the foreground traffic packets slightly increases but does not exceed 50% of the guaranteed delay bound in any case. This ratio is not affected by the packet size or the frame size in the system. Hence, DLS/VPS performance is robust against the changing system parameters and can provide guaranteed-service traffic of variable size packets with average end-to-end delay much lower than the guaranteed bounds in any case. The experiment results also show that DLS/VPS can

The simulation results show that the DLS/VPS can provide the same average end-to-end delay to the connection-oriented guaranteed-service traffic of variable size packets which can be offered by the original delay-limiter switch to the connection-oriented guaranteed-service traffic of fixed size packets.

CHAPTER 6

CONCLUSION

Routers are the contention points in networks where many transmission lines intersect. The possible congestion problems must be handled in the most efficient way. Therefore, the routers on the network backbone must operate at very high speeds and very efficiently. The data networks must be able to support various QoS requirements for various applications although the routers in the network may have different sizes, port numbers and operating speeds. The delay-limiter switch can provide QoS guarantees for the connection-oriented traffic in a scalable and low-complexity fashion.

This thesis study presents a novel method, Total Packet Count (TPC), which can be applied to the delay-limiter switch and enable it to support connectionless best-effort traffic in an efficient way. The simulation results show that depending on the best-effort traffic arrivals, the end-to-end delay of the connectionless best-effort traffic can drop to as low as 10% of the end-to-end delay values that result when the remaining resources are utilized in a rudimentary fashion after the connection-oriented guaranteed-service traffic. The simulation results also show that the QoS guarantees promised to the connection-oriented traffic are still valid when TPC is applied. TPC requires a control line between each input-output port pair in the switch for the transmission of the necessary information for QoS scheduling where these lines are possibly already present due to the dynamic fabric scheduler employed. Through one of these lines, $\lceil \log_2 T \rceil$ bits can be sent from an input port to the corresponding output port in a sub-frame time for the TPC algorithm if the binary logic is used. This control information has to be transmitted to the output ports and processed in every slot where many packets can be served within a single slot since the slot length is long enough to transmit a large sub-frame that involves many smaller packets. The TPC algorithm has a constant time complexity of O(1).

In the proposed architecture, each output port maintains a single FIFO queue for the best-effort connectionless traffic. This does not allow classifying the connectionless traffic into different service levels. Future work can be carried out on providing connectionless traffic with different service levels.

This thesis also presents a new switch architecture, Delay-Limiter Switch with Variable Packet Size (DLS/VPS), which makes it possible for the delay-limiter switch to work with variable-sized packets. The variable packet size support makes the delay-limiter switch compatible with the IP traffic packets and decreases the queuing delay of the packets in the traffic shapers at the network ingress. The simulation results show that the DLS/VPS can provide the same order QoS guarantees to the connection-oriented guaranteed-service traffic which can be offered by the original delay-limiter switch. The DLS/VPS works with constant time complexity and in a scalable fashion which make it possible for the DLS/VPS work in high-speed networks.

This thesis does not study how to support best-effort traffic efficiently through the Delay-Limiter Switch with Variable Packet Size. Future work can include supporting best-effort traffic efficiently with the unused resources by the guaranteed-service traffic without affecting the guaranteed-service traffic delay bounds.

REFERENCES

[1] S.E. Guran, Scheduling and Switching Architecture for Virtual Synchronous Frame (VSF), Ph.D. Thesis, Carnegie Mellon University, 2004.

[2] S.Ece (Guran) Schmidt, Hyong S. Kim; A new scalable service discipline for real-time traffic: The framed-deadline scheduler; Elsevier Computer Communications Volume 30, Issue 6, 26 March 2007, Pages 1258-1277

[3] Tzeng, N.-F.; Multistage-based switching fabrics for scalable routers; Parallel and Distributed Systems, IEEE Transactions on, Volume 15, Issue 4, April 2004 Page(s):304 – 318

[4] Chuanjun Li, Zheng S. Q.; Mei Yang, Scalable schedulers for high-performance switches; High Performance Switching and Routing, 2004. HPSR 2004 Workshop on, Pages: 198-202

[5] Hluchyj, M.G., Karol, M.J.; Queueing in space-division packet switching; INFOCOM '88. Networks: Evolution or Revolution? Proceedings. Seventh Annual Joint Conference of the IEEE Computer and Communications Societies., IEEE 27-31 March 1988 Page(s):334 - 343

[6] McKeown, N.; Mekkittikul, A.; Anantharam, V.; Walrand, J.; Achieving 100% throughput in an input-queued switch; Communications, IEEE Transactions on, Volume 47, Issue 8, Aug. 1999 Page(s):1260 - 1267

[7] R. E. Tarjan; Data Structures and Network Algorithms, Bell Labs, 1983

[8] Chao, H.J.; Lam, C.H.; Guo, X.; A fast arbitration scheme for terabit packet switches; Global Telecommunications Conference, 1999. GLOBECOM '99 Volume 2, 1999 Page(s):1236 - 1243 vol.2

[9] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, Charles P. Thacker; High-speed switch scheduling for local-area networks; November 1993 ACM Transactions on Computer Systems (TOCS), Volume 11 Issue 4

[10] Nick McKeown; The iSLIP Scheduling Algorithm for Input-Queued Switches; Networking, IEEE/ACM Transactions on, Volume 7, Issue 2, April 1999 Page(s):188 - 201

[11] Chao, J.; Saturn: a terabit packet switch using dual round robin; Communications Magazine, IEEE Volume 38, Issue 12, Dec. 2000 Page(s):78 - 84

[12] Chao, H.J.; Jin-Soo Park; Centralized contention resolution schemes for a large-capacity optical ATM switch; ATM Workshop Proceedings, 1998 IEEE 26-29 May 1998 Page(s):11 - 16

[13] Mekkittikul, A.; McKeown, N.; A practical scheduling algorithm to achieve 100% throughput in input-queued switches; INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume 2, 29 March-2 April 1998 Page(s):792 - 799 vol.2

[14] Chen, K.F.; Sha, E.H.-M.; Zheng, S.Q; A fast noniterative scheduler for input-queued switches with unbuffered crossbars.; Parallel Architectures, Algorithms and Networks, 2005. ISPAN 2005. Proceedings. 8th International Symposium on, 7-9 Dec. 2005 Page(s):6 pp.

[15] Young-Keun Park; Young-Keun Lee; Parallel iterative matching-based cell scheduling algorithm for high-performance ATM switches; Consumer Electronics, IEEE Transactions on Volume 47, Issue 1, Feb. 2001 Page(s):134 - 137

[16] Xiao Zhang; Bhuyan, L.N.; Deficit round-robin scheduling for input-queued switches; Selected Areas in Communications, IEEE Journal on, Volume 21, Issue 4, May 2003 Page(s):584 – 594

[17] Peng Wang; Hongbo Fang; Depeng Jin; Lieguang Zeng; Fang Tao; The iTFF scheduling algorithm for input-queued switches; Communications, Circuits and Systems, 2004. ICCCAS 2004. 2004 International Conference on Volume 1, 27-29 June 2004 Page(s):692 - 697 Vol.1

[18] Peng Wang; Depeng Jin; Lieguang Zeng; A novel scheduling algorithm for input-queued switches; Communications, 2003. APCC 2003. The 9th Asia-Pacific Conference on, Volume 2, 21-24 Sept. 2003 Page(s):768 - 772 Vol.2

[19] Ferrari, D.; Verma, D.C.; A scheme for real-time channel establishment in wide-area networks; Selected Areas in Communications, IEEE Journal on, Volume 8, Issue 3, April 1990 Page(s):368 – 379

[20] Cruz, R.L.; A calculus for network delay. I. Network elements in isolation; Information Theory, IEEE Transactions on, Volume 37, Issue 1, Jan. 1991 Page(s):114 - 131

[21] Golestani, S.J.; Congestion-free transmission of real-time traffic in packet networks; INFOCOM '90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. 'The Multiple Facets of Integration'. Proceedings., IEEE 3-7 June 1990 Page(s):527 - 536 vol.2

[22] Hui Zhang; Service disciplines for guaranteed performance service in packet-switching networks; Proceedings of the IEEE Volume 83, Issue 10, Oct. 1995 Page(s):1374 – 1396

[23] Fattah, H.; Leung, C.; An overview of scheduling algorithms in wireless multimedia networks; Wireless Communications, IEEE [see also IEEE Personal Communications] Volume 9, Issue 5, Oct. 2002 Page(s):76 - 83

[24] Zhang, H.; Ferrari, D.; Rate-controlled static-priority queueing; INFOCOM '93. Proceedings.Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future. IEEE 28 March-1 April 1993 Page(s):227 - 236 vol.1

[25] A. Ganz, and Y. Gao; Efficient algorithms for SS/TDMA scheduling; IEEE Trans. Commun, vol. 40, pp.1367 –1374, Aug 1992.

[26] K.L. Yeung; Efficient time slot assignment algorithms for TDM hierarchical and nonhierarchical switching systems; IEEE Trans on Communications, vol. 49, Feb 2001, pp.351–359.

APPENDIX A

A.1. CONNECTIONLESS BEST-EFFORT TRAFFIC DELAY

This section presents the end-to-end delay of the connectionless best-effort traffic when it is served in a rudimentary fashion after the guaranteed-service traffic in the delay-limiter switch with a switch size of 50x50. Figures A.1 to A.8 illustrate the end-to-end delay when the connectionless best-effort traffic load is 10%, 20%, 30%, 40%, 50%, 60%, 70% and 80% in the respective figures.

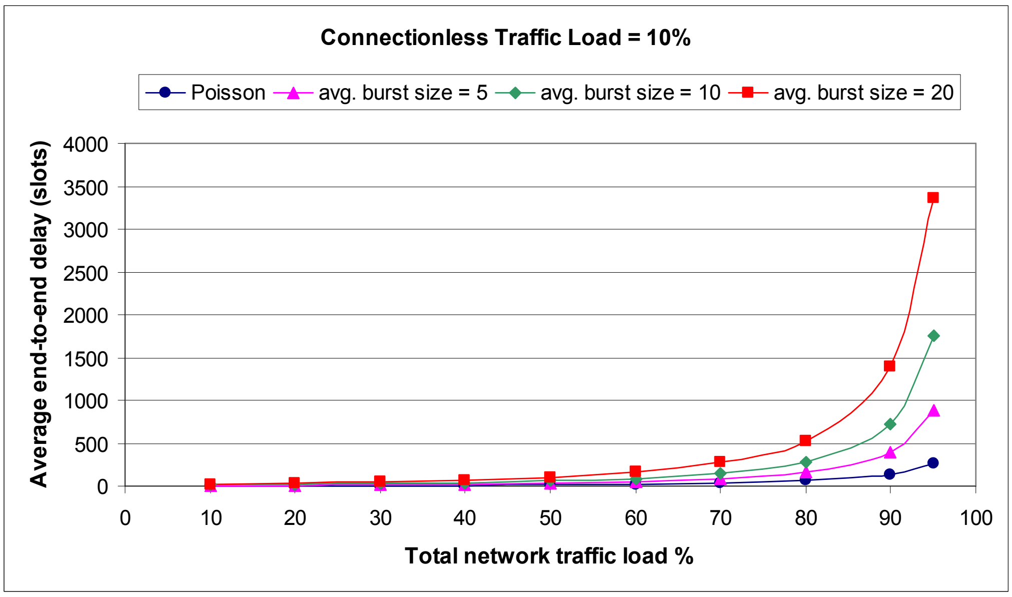


Figure A.1 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 10%.

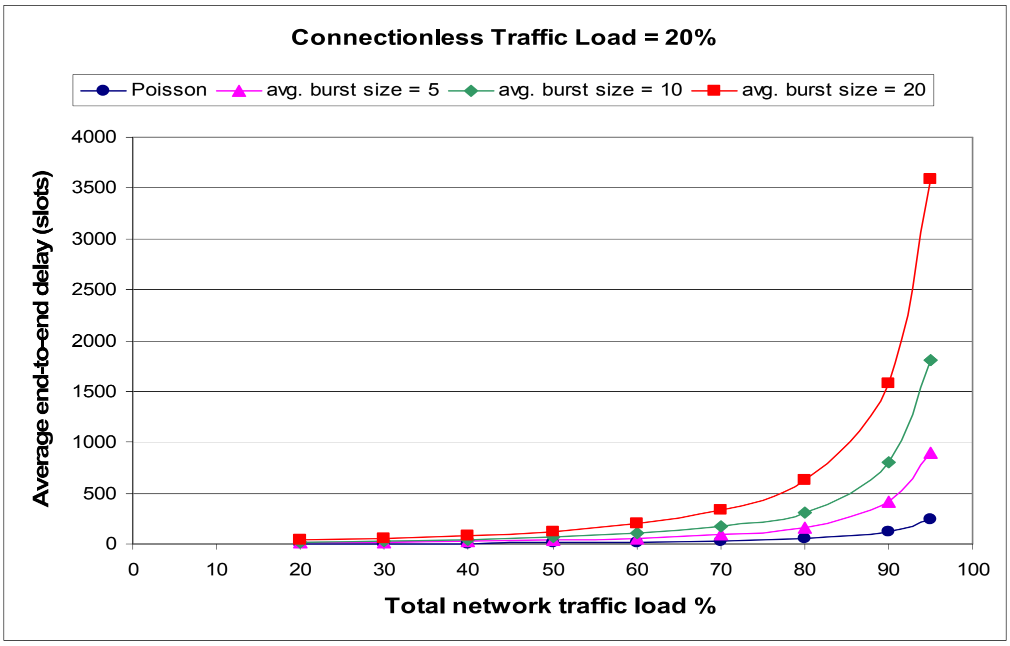


Figure A.2 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 20%.

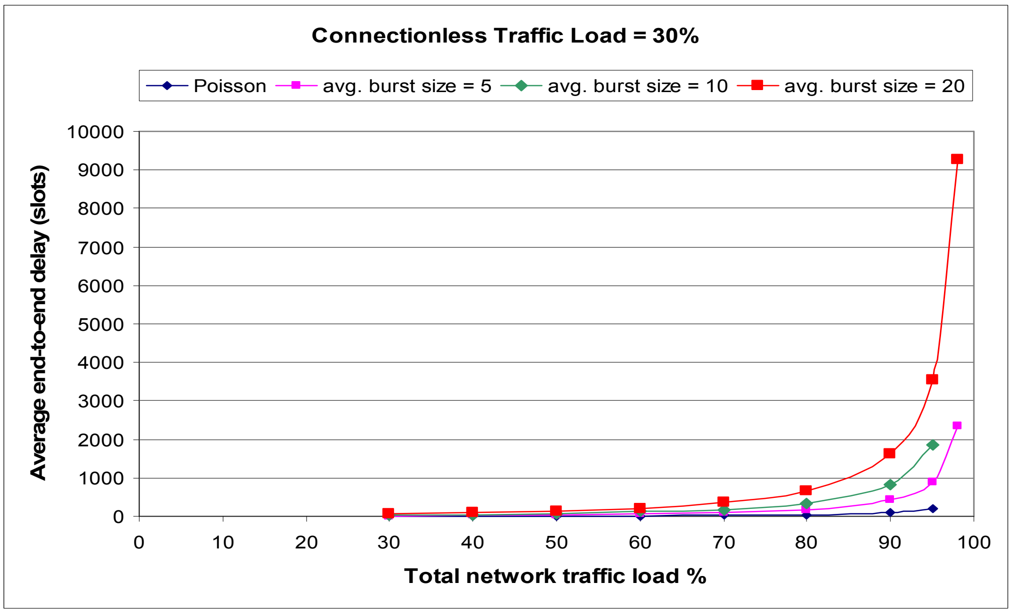


Figure A.3 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 30%.

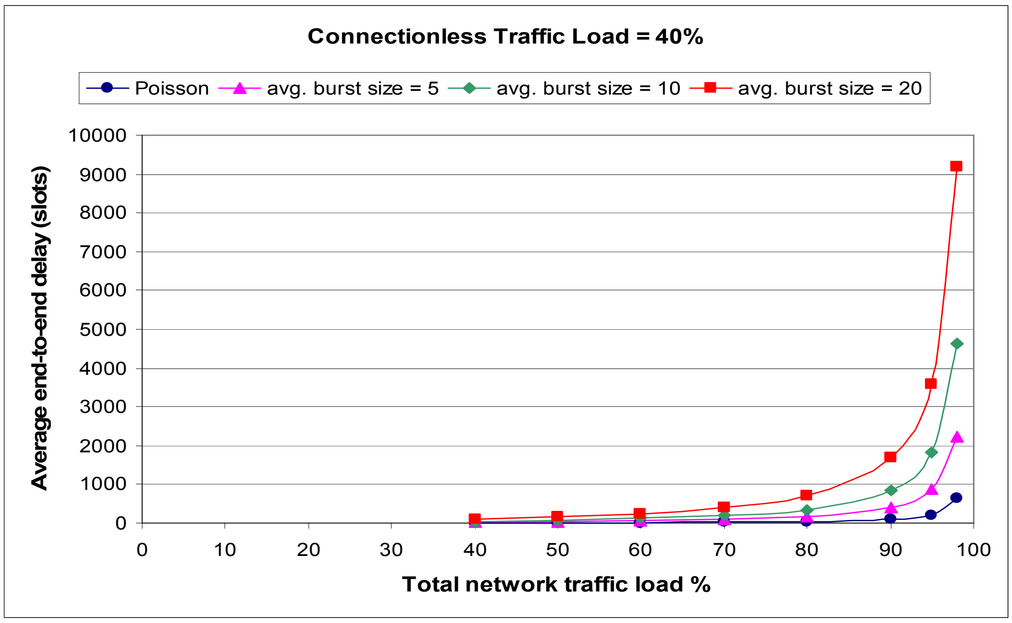


Figure A.4 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 40%.

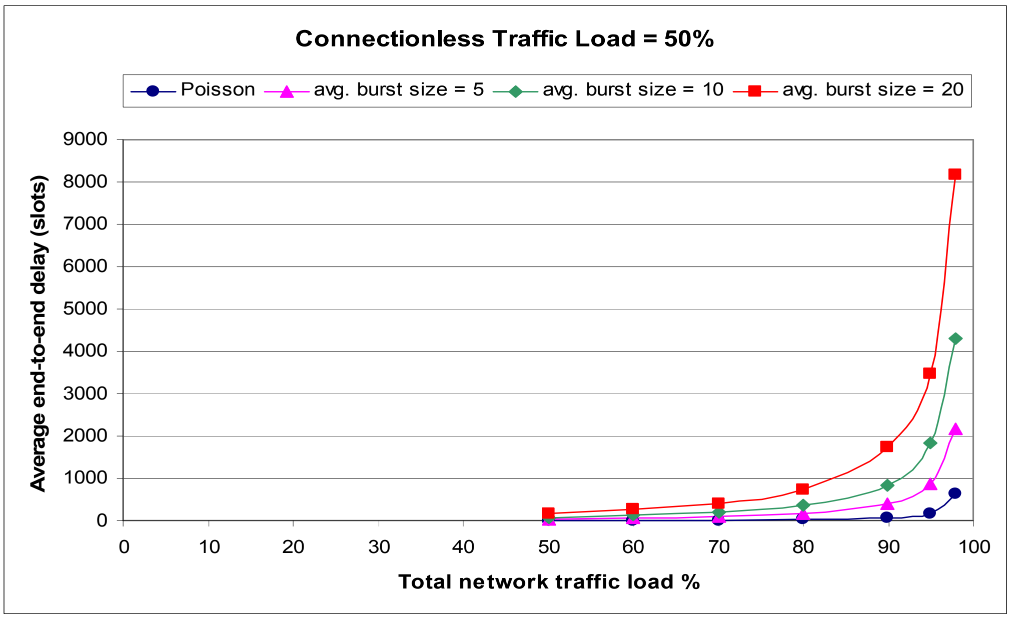


Figure A.5 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 50%.

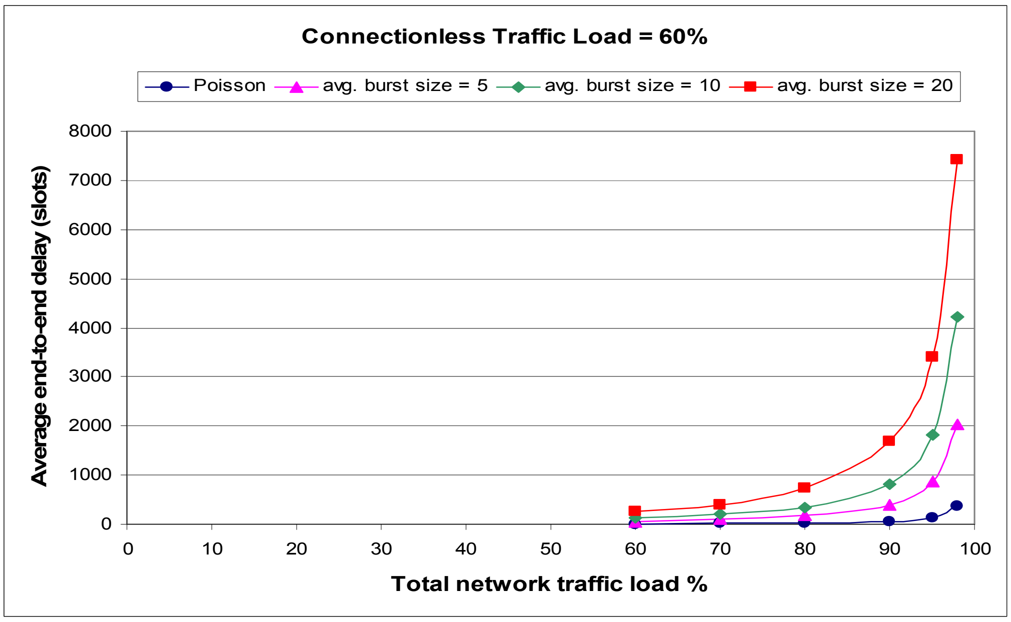


Figure A.6 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 60%.

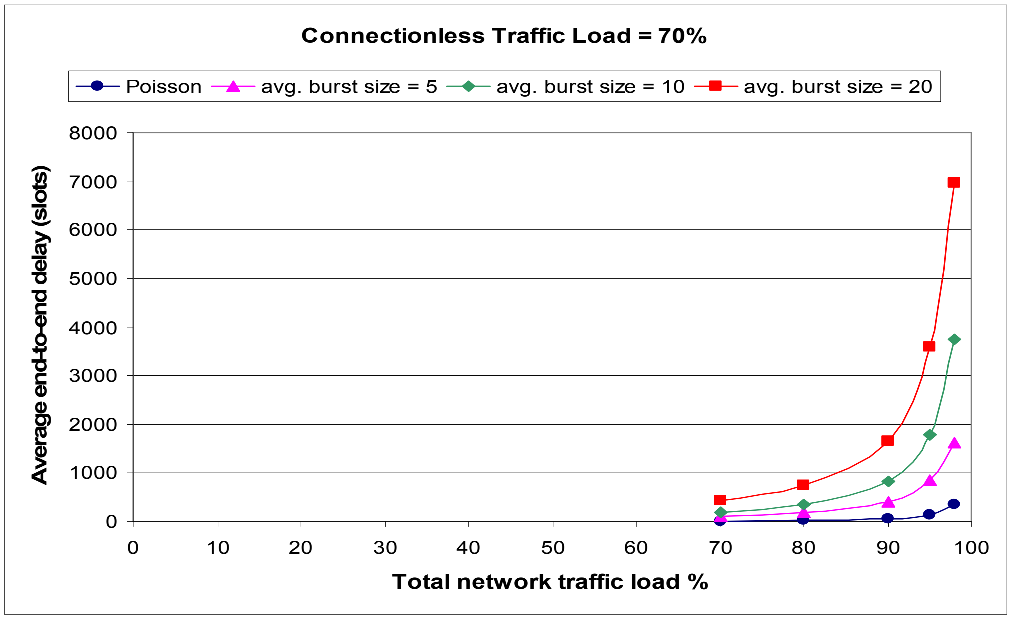


Figure A.7 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 70%.

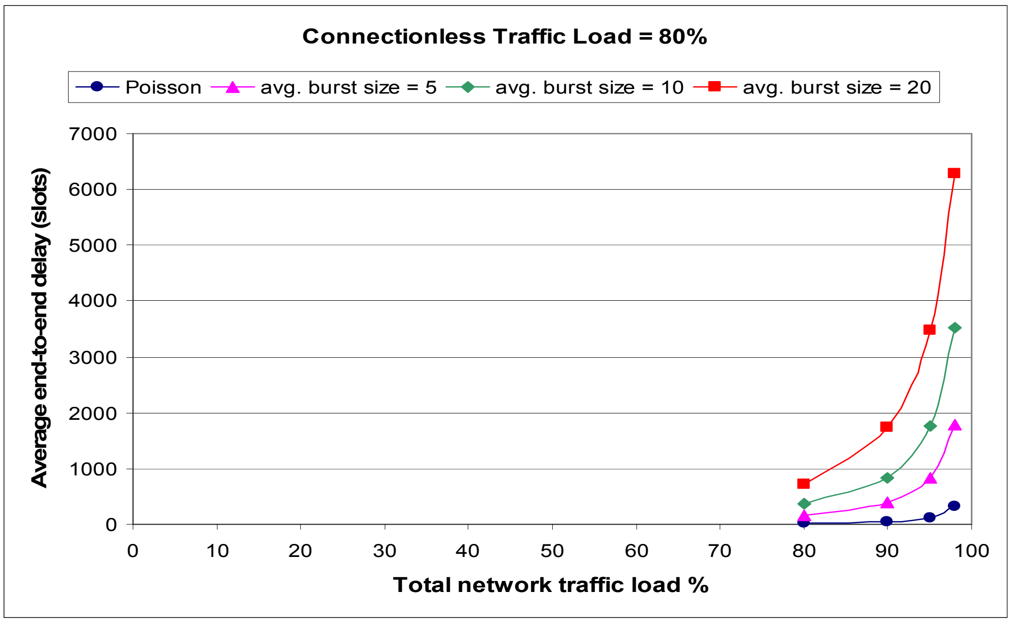


Figure A.8 End-to-end delay of the connectionless best-effort traffic in a rudimentary fashion when the connectionless best-effort traffic load is 80%.

A.2. GUARANTEED-SERVICE CONNECTIONS DELAY

This section presents the end-to-end delay normalized with the delay bound of the guaranteed-service foreground connections when the rudimentary method and the TPC algorithm are used for the best-effort connectionless traffic. Figures A.9 to A.32 illustrate the normalized end-to-end delay of the guaranteed-service foreground traffic when the connectionless best-effort traffic load is 10%, 20%, 30%, 40%, 50% and 60% under bursty and Poisson arrivals. It can be observed that when TPC algorithm is not applied, the normalized end-to-end delay values are consistent with those obtained in [1] and [2]. When the TPC algorithm is applied, the end-to-end delay values for the guaranteed-service connections are still within the guaranteed bounds.

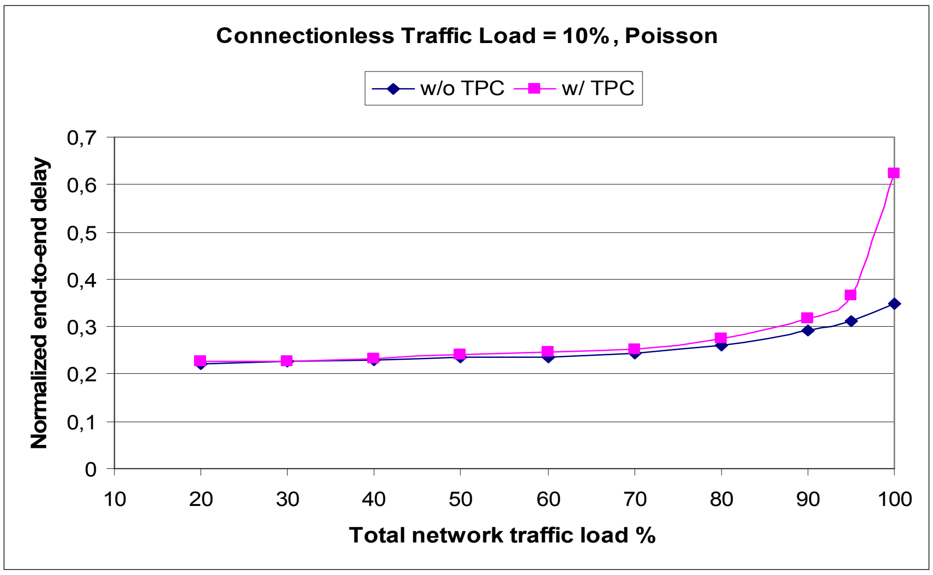


Figure A.9 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under Poisson arrivals with 10% load.

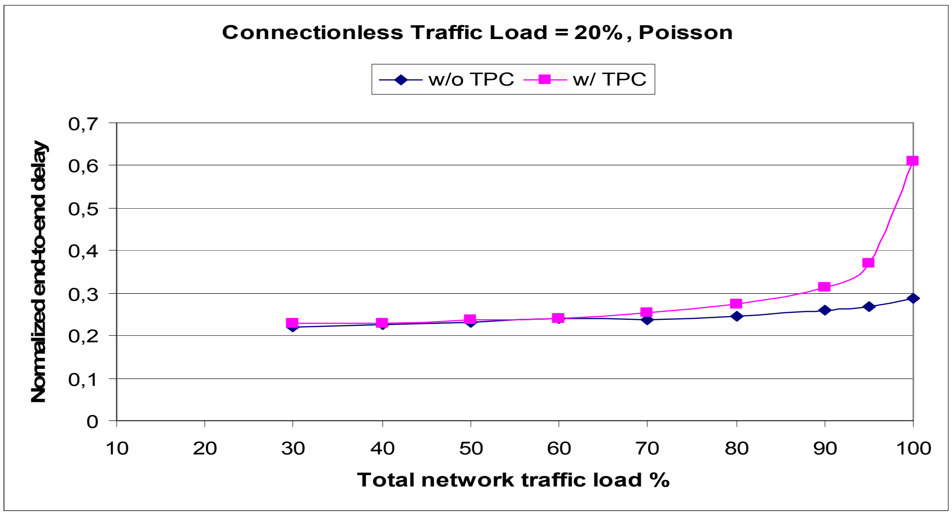


Figure A.10 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under Poisson arrivals with 20% load.

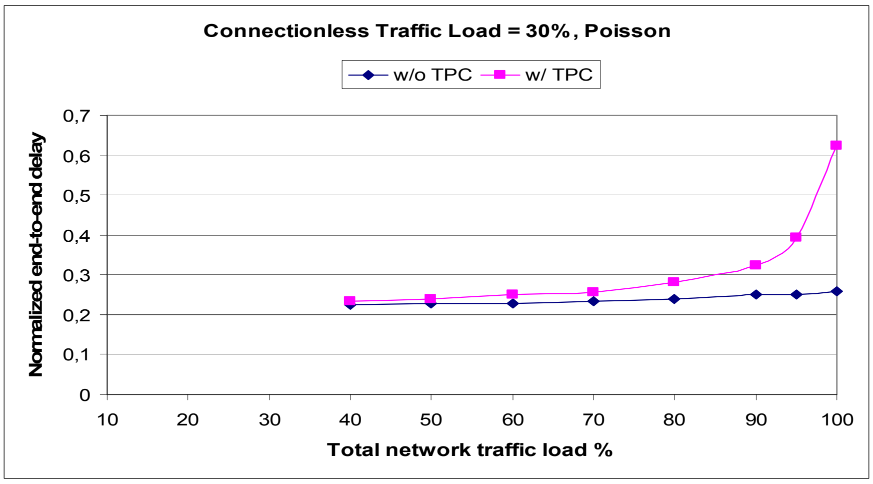


Figure A.11 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under Poisson arrivals with 30% load.

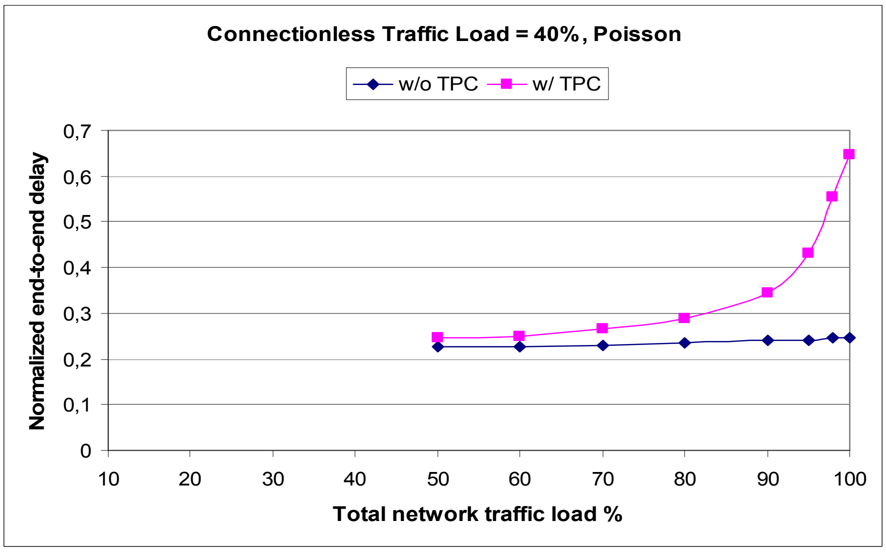


Figure A.12 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under Poisson arrivals with 40% load.

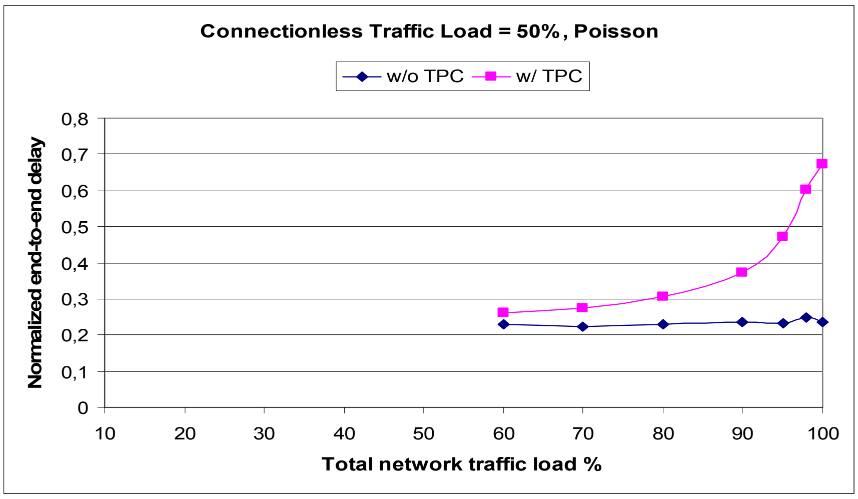


Figure A.13 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under Poisson arrivals with 50% load.

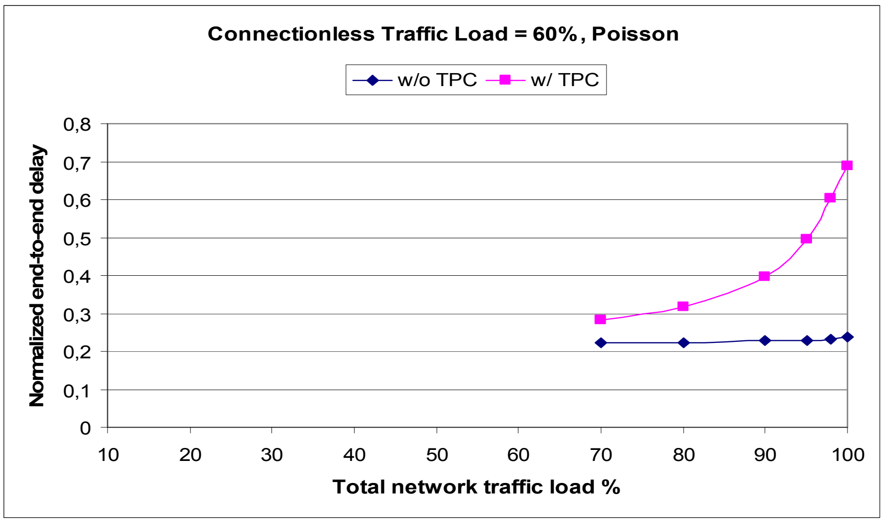


Figure A.14 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under Poisson arrivals with 60% load.

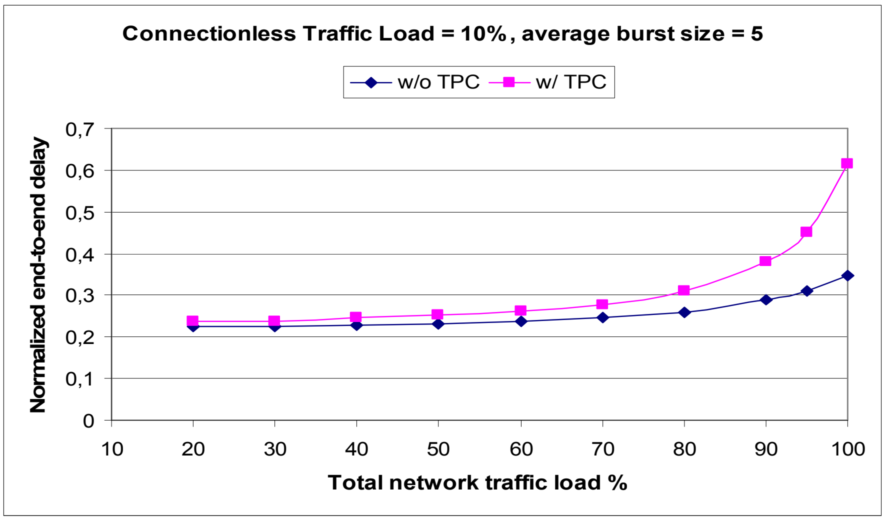


Figure A.15 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 5 with 10% load.

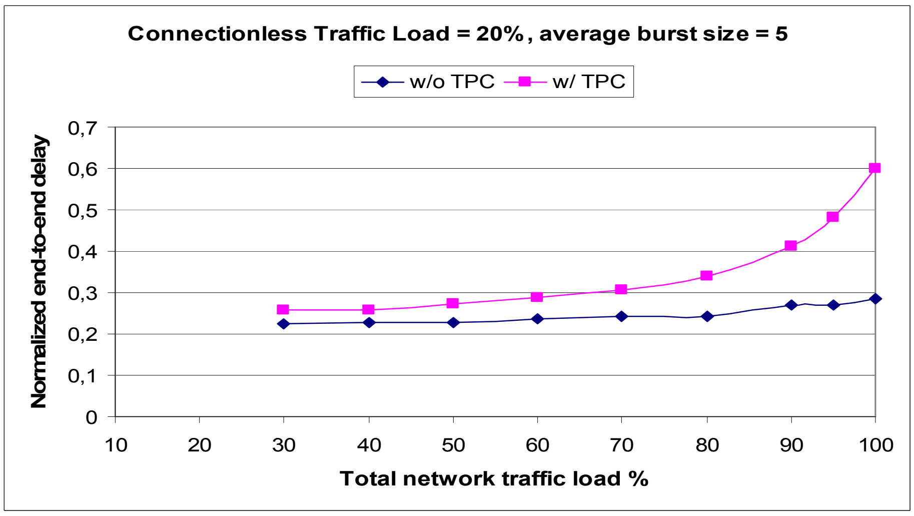


Figure A.16 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 5 with 20% load.

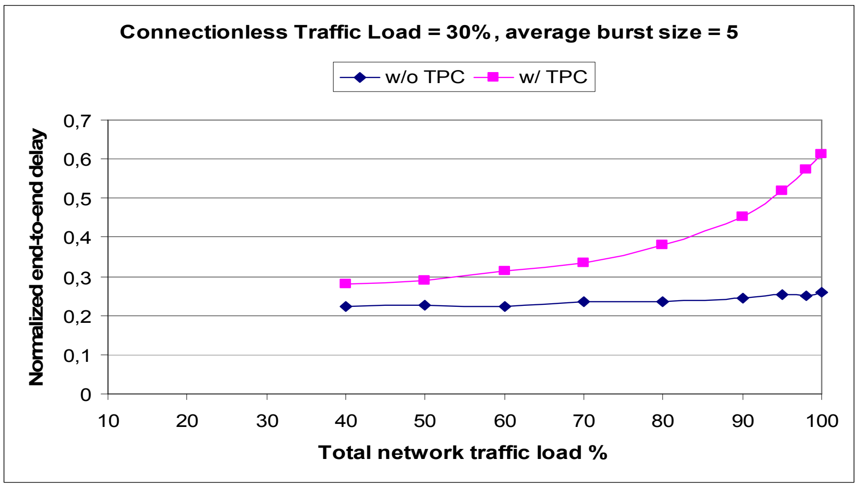


Figure A.17 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 5 with 30% load.

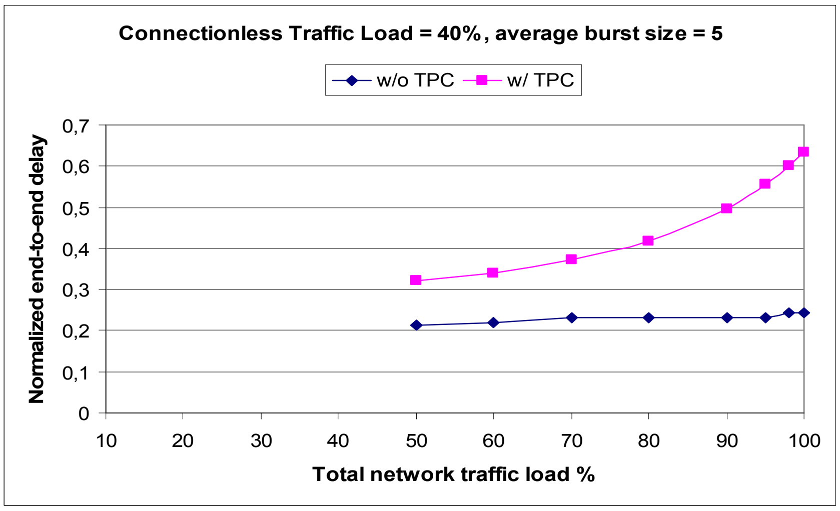


Figure A.18 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 5 with 40% load.

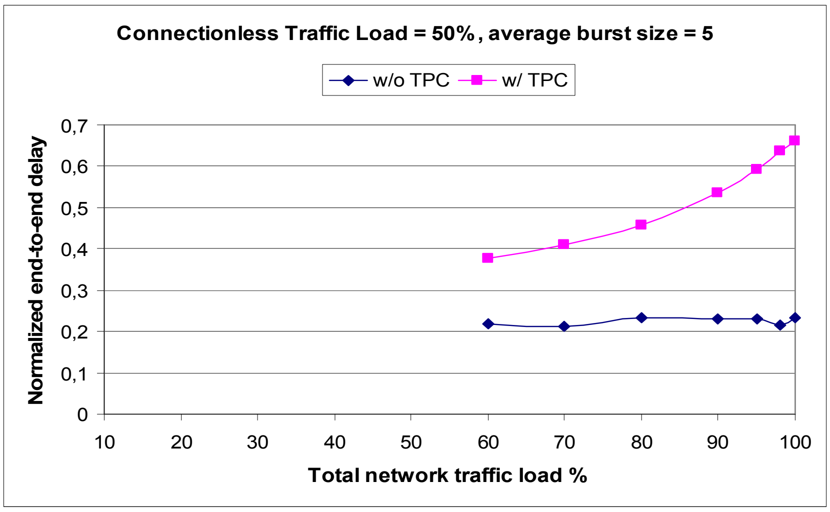


Figure A.19 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 5 with 50% load.

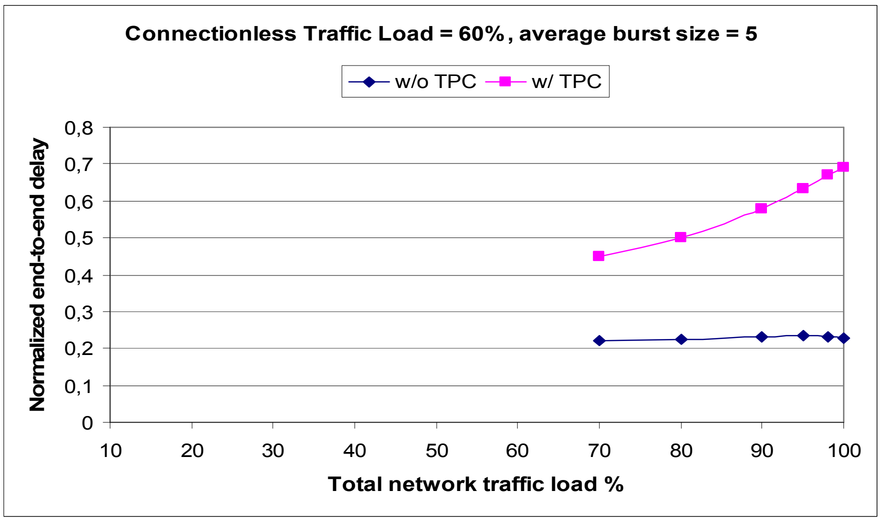


Figure A.20 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 5 with 60% load.

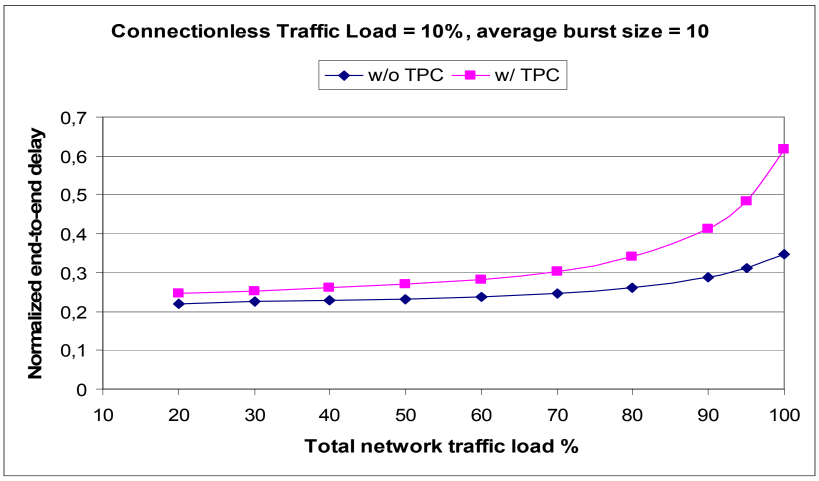


Figure A.21 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 10 with 10% load.

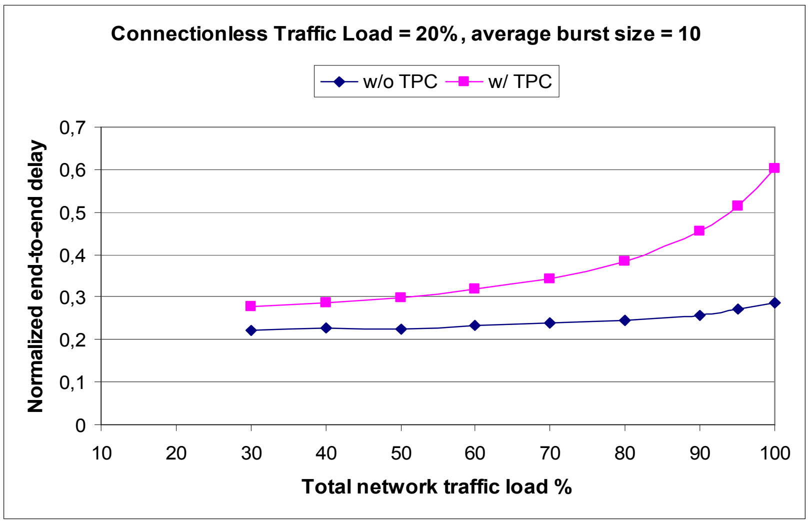


Figure A.22 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 10 with 20% load.

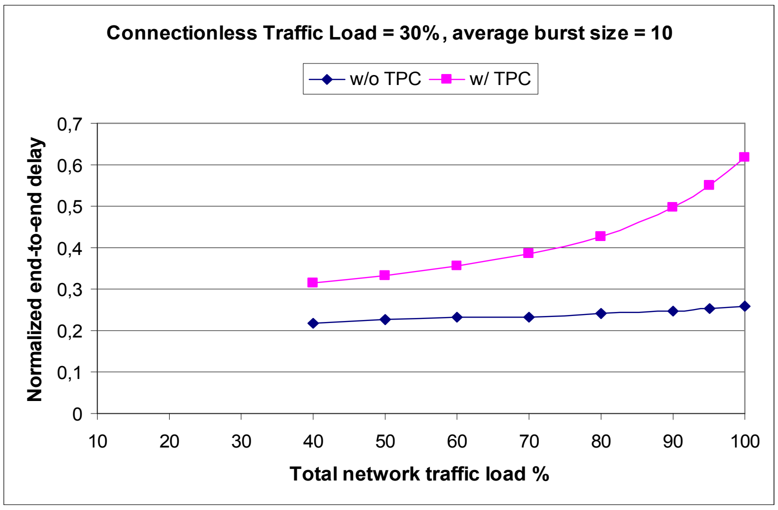


Figure A.23 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 10 with 30% load.

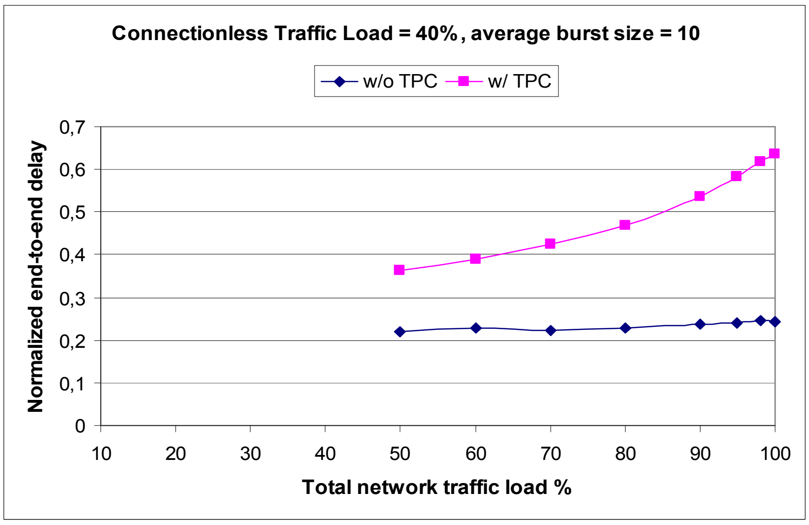


Figure A.24 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 10 with 40% load.

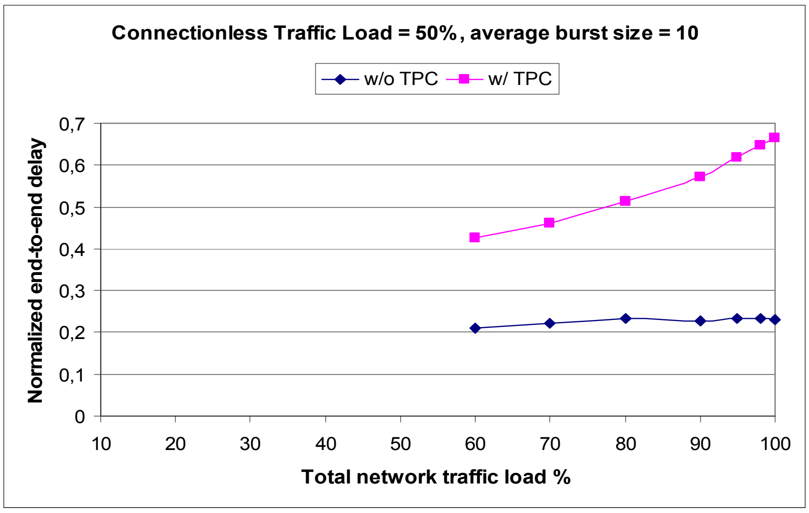


Figure A.25 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 10 with 50% load.

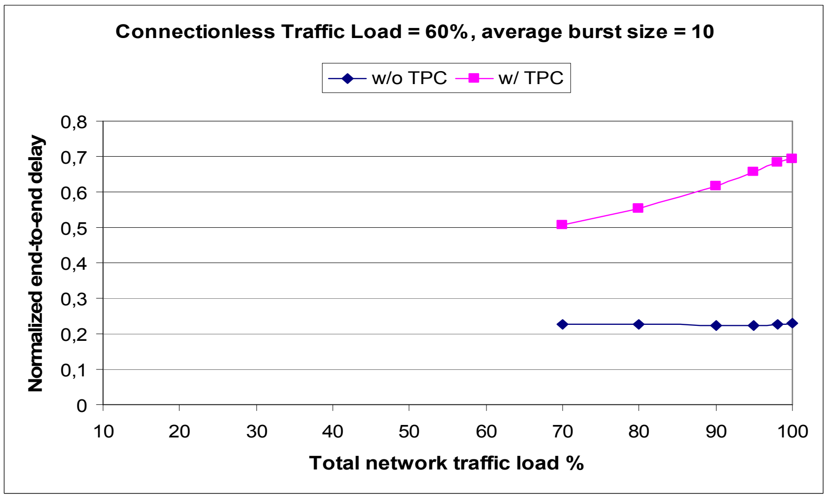


Figure A.26 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 10 with 60% load.

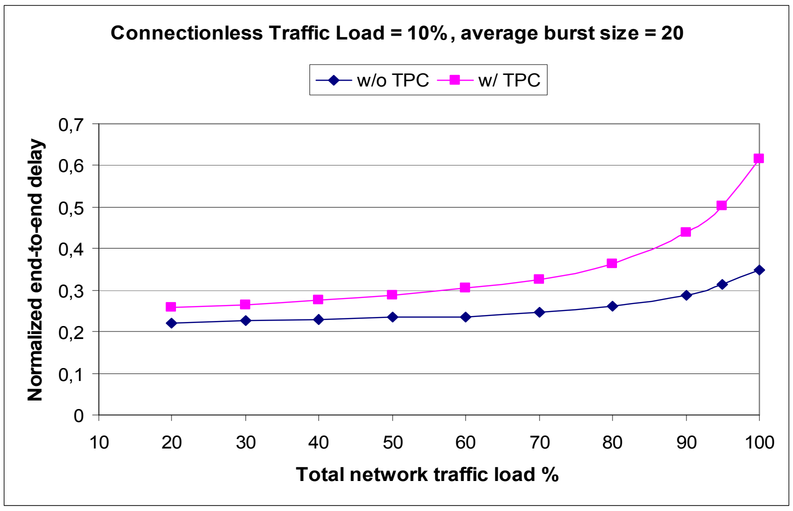


Figure A.27 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 20 with 10% load.

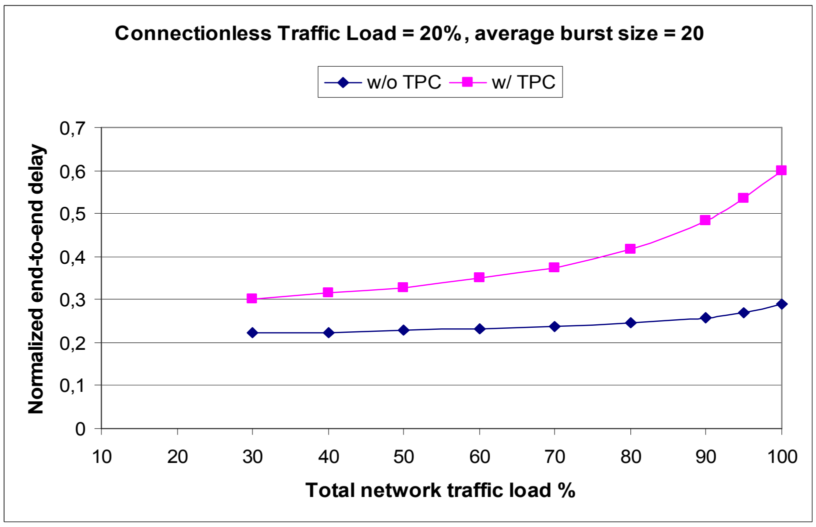


Figure A.28 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 20 with 20% load.

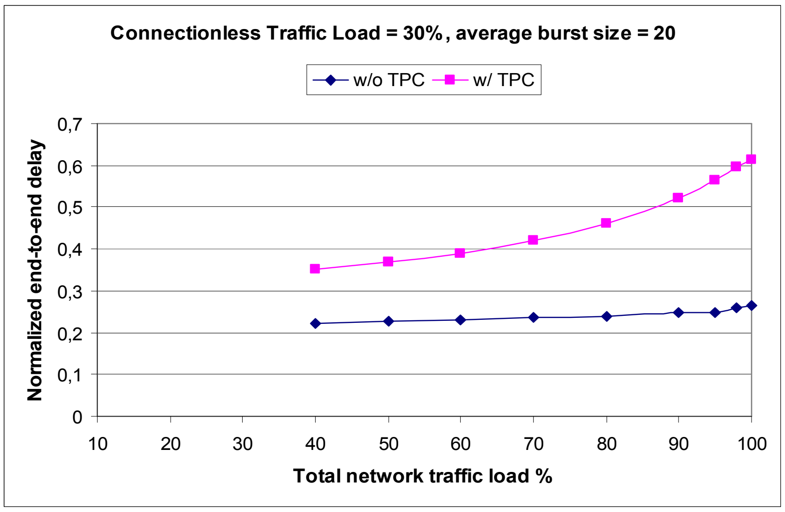


Figure A.29 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 20 with 30% load.

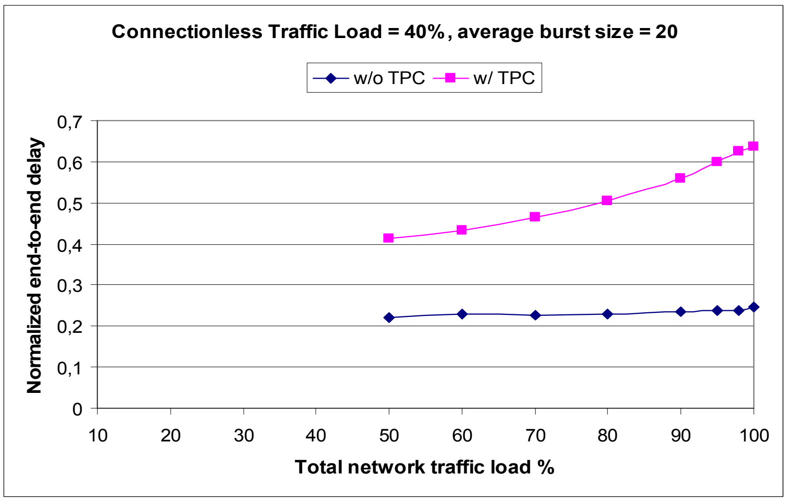


Figure A.30 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 20 with 40% load.

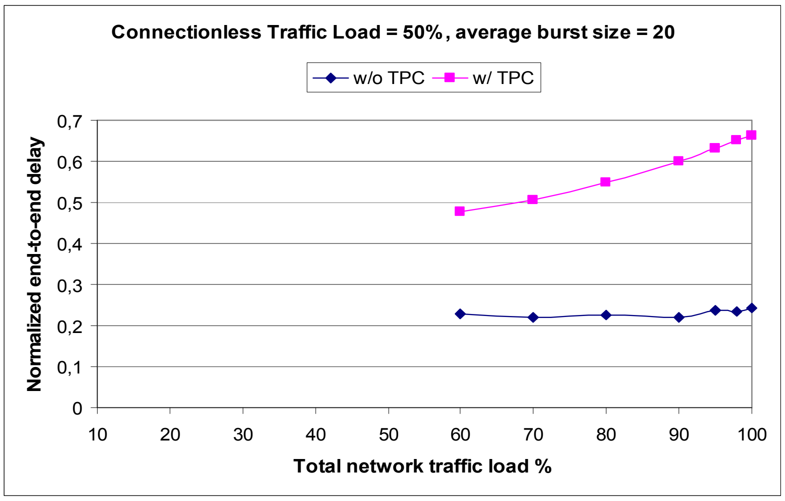


Figure A.31 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 20 with 50% load.

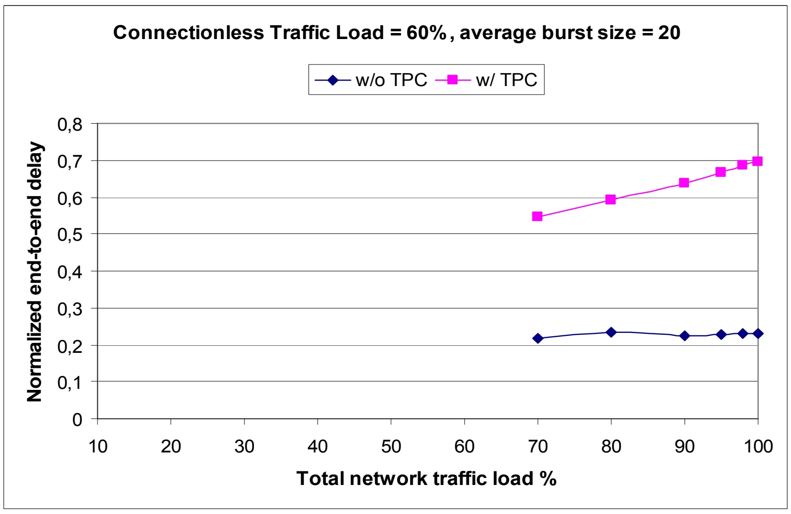
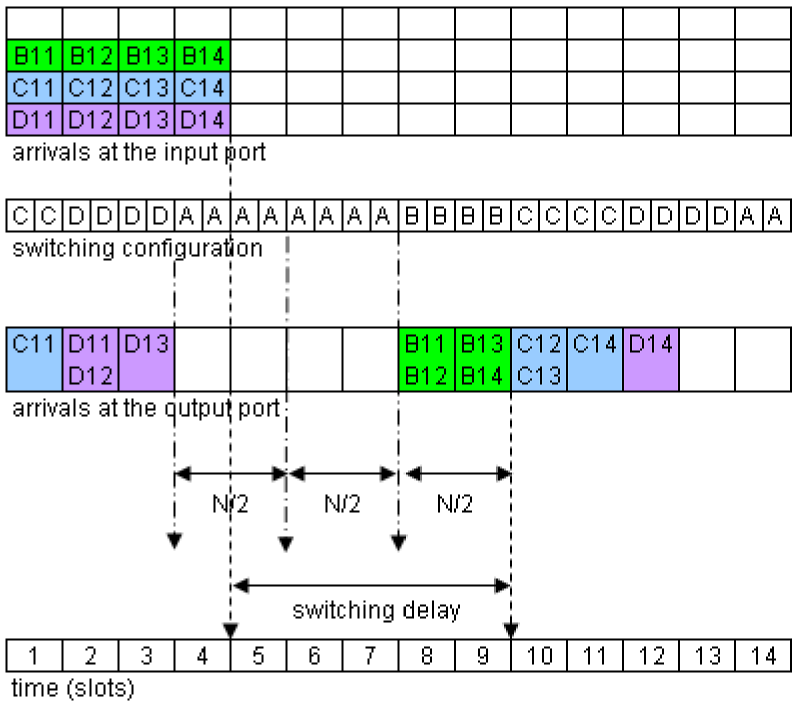


Figure A.32 Normalized end-to-end delay of the guaranteed-service connections when best-effort traffic is served in a rudimentary fashion compared to when the best-effort traffic is served with TPC under bursty arrivals of average burst size 20 with 60% load.

APPENDIX B

B.1. MAXIMUM SWITCHING DELAY IN CASE OF TM REARRANGEMENT

Although there are packets of more than one connection in the corresponding input ports, there can be some cases in which none of these packets are switched to the output port for a while.



B11 B12 B13 B14
C11 C12 C13 C14
D11 D12 D13 D14
arrivals at the input port
C C D D D D A A A A A A A A B B B B C C C C D D D D A A
switching configuration
C11 D11 D12 D13 B11 B12 B13 B14 C12 C13 C14 D14
arrivals at the output port
N/2
N/2
N/2
switching delay
1 2 3 4 5 6 7 8 9 10 11 12 13 14
time (slots)

Figure B.1 Maximum switching delay for a group of connections in case of TM rearrangement

Figure B.1 illustrates such a situation where N is 4, S is 4 and T is 16 slots. There are four unit-rate connections established through the same output port: connection A, B, C and D. Suppose one packet of length N arrives for each connection except *A,* all completing their arrivals in slot 4. In order for these packets not to be switched to the output port completely, the switch fabric is configured in such a way that connection A starts being connected to the output port in slot 4 in which the packets of other connections complete their arrivals. Connection A is kept connected to the output port for 2 ($= \lceil N/Su \rceil$) slots until the end of slot 5 and just after this period traffic matrix is rearranged so that again connection A is connected to the output port, occupying the output port for another 2 ($= \lceil N/Su \rceil$) slots until the end of slot 7. This fabric scheduling pattern causes the other connections to wait the maximum time before any one of them can complete switching a full packet to the output port. One of the connections, B in this case, is connected to the output port in slot 8. The packet of connection B completes its switching to the output port in slot 9. The reason for choosing B to be the next connection in the fabric schedule is to maximize the switching time because the packet of connection D would complete its switching in slot 8 according to the fabric schedule if it was the next connection since three cells of the first packet of connection D are already switched to the output port. With the completion of packet B_1, the output port can start transmitting this first packet. Therefore, the idle time of the output port, *tidle*, between the arrivals of the last cells of the packets at their input ports and the arrivals of the last cells of the packets at the output port can be expressed as follows:

t_1 = time during connection A is connected to the output port = 2 x $\lceil N/Su \rceil$

t_2 = time to switch the first packet of connection B = $\lceil N/Su \rceil$

$$
\begin{aligned}
tidle &= t_1 + t_2 - 1 - 1 \\
&= \lceil 3N / Su \rceil - 2 \qquad \text{(Equation B.1)}
\end{aligned}
$$

The first –1 term exists due to the fact that the slot in which the packets complete their arrivals coincide with the slot in which connection *A* starts occupying the switch fabric. Connection *A* starts being connected to the output port in slot 4 in which the packets of other connections complete their arrivals.

The second –1 term originates due to the fact that the output port starts transmitting packet B_1 in the same slot it completes its arrival at the output port.

APPENDIX C

C.1. AN IN-BETWEEN PACKET AFTER A NON-IN-BETWEEN PACKET

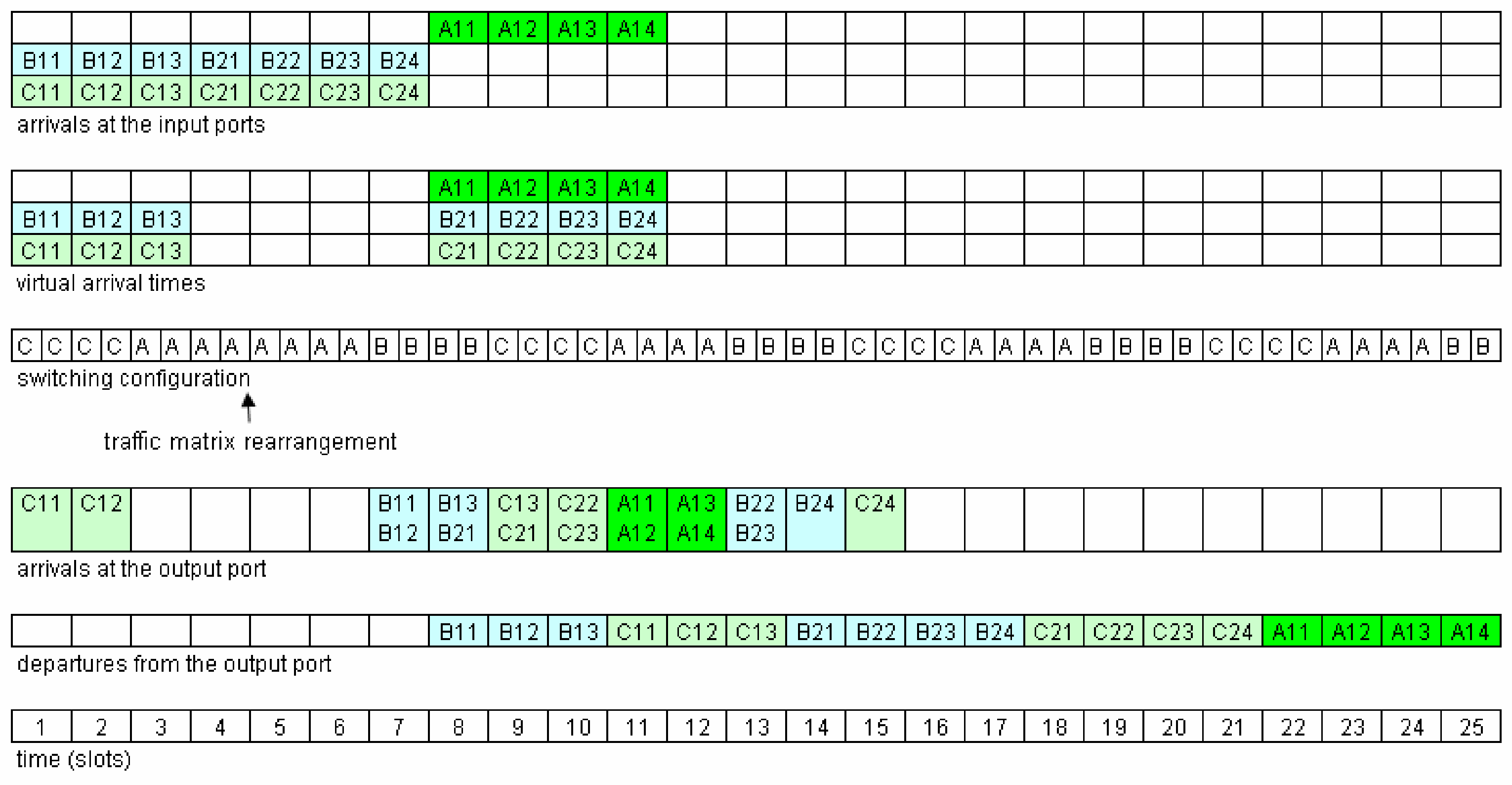
Figure C.1 shows a system where N is 4, S is 3 and T is 12 slots. There are three connections established through the node, connection A, B and C, each one coming to a different input port and destined to the same output port. Suppose that the frames for connection B and connection C start long before the slot labeled 1 in the figure and thus the cells B_{11}, B_{12}, B_{13} of connection B and C_{11}, C_{12}, C_{13} of connection C have virtual arrival times equal to their actual arrival times. No packet belonging to connection A arrives within T slots before the arrival of packet A_1, so the virtual arrival time of A_1 is equal to its actual arrival time. The first frames of connections B and C end after slot 4 with the arrival of B_{21} and C_{21} and their second frames start in slot 5. Therefore, the second packets of connection B and connection C are in-between packets. After slot 4, TM rearrangement occurs and the first packet of connection B is fully switched to the output port in slot 8. At that time packet B_1 is eligible and the output port starts transmitting it. After that, the output port transmits packets C_1, B_2 and C_2 until the end of slot 21. The first packet of connection A, A_1, starts getting served in slot 22 and finishes in slot 25. According to this scheme, packet A_1 must have arrived at its input port in slot 11 to comply with the maximum node delay, $(T + N - 2)$. If packet A_1 arrives X slots before slot 11, it must be sent out of the node X slots before slot 25 to satisfy the maximum node delay. Therefore, the second packets of connection *B* and connection *C*, which are in-between packets, must have virtual arrival times greater than or equal to slot 11 in order not to obstruct packet A_1. This summarizes the situation in which a connection must be protected against the in-between packets of the other connections where the packets prior to the in-between packets are not in-between.

This situation can be generalized as follows:

Consider a system with sub-frame size *N*, frame size *T* and number of unit-rate connections that can be supported *S*. In such a system a connection, connection *A* for example, may have to contend with at most $(S-1)$ in-between packets belonging to all other connections. Suppose one packet of size *N* arrives for connection *A* after a long silence period and there are two consecutive packet arrivals for all connections except connection *A*. None of the first packets of any connection is an in-between packet and they all have virtual arrival times equal to their actual arrival times. The second packets of all connections excluding *A*, which are all in-between packets, compete with the first packet of connection *A*, A_1. Since the second packets are in-between but the first packets are not, the largest size the first packets can have is $(N-1)$ cells. The second packets can involve *N* cells at most. Therefore,

$(S-1)$ connections all send packets of $(N-1)$ cells before A_1 and the time required to transmit these packets, *tf*, is

$tf = (S-1) \text{ x } (N-1)$ (Equation C.1)



A11 A12 A13 A14
B11 B12 B13 B21 B22 B23 B24
C11 C12 C13 C21 C22 C23 C24
arrivals at the input ports
A11 A12 A13 A14
B11 B12 B13 B21 B22 B23 B24
C11 C12 C13 C21 C22 C23 C24
virtual arrival times
C C C C A A A A A A A A B B B B C C C C A A A A B B B B C C C C A A A A B B B B C C C C A A A A B B
switching configuration
traffic matrix rearrangement
C11 C12 B11 B12 B13 B21 C13 C21 C22 C23 A11 A12 A13 A14 B22 B23 B24 C24
arrivals at the output port
B11 B12 B13 C11 C12 C13 B21 B22 B23 B24 C21 C22 C23 C24 A11 A12 A13 A14
departures from the output port
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
time (slots)

Figure C.1 Virtual arrival time assignment to in-between packets after non-in-between packets.

The output port stays idle during *tidle* = $\lceil 3N/Su \rceil - 2$ slots (see appendix B) between the completion of the arrivals of the packets of (S – 1) connections at their input ports and the first cell belonging to any of these connections starts being transmitted.

$$tidle = \lceil 3N/Su \rceil - 2 \qquad \text{(Equation C.2)}$$

Using Equation C.1 and Equation C.2, the total time required for the transmission of the first packets of all connections excluding connection A, *tfidle*, is

$$\begin{aligned} tfidle &= tf + tidle \\ &= (S-1) \text{ x } (N-1) + \lceil 3N/Su \rceil - 2 \\ &= T - S - N + \lceil 3N/Su \rceil - 1 \qquad \text{(Equation C.3)} \end{aligned}$$

After the transmission of the first packets of all connections excluding connection A, (S – 1) in-between packets belonging to the other connections are transmitted. The longest time necessary to transmit these packets occurs when all the packets are of size N. Thus, this time denoted by *ts* is

$$\begin{aligned} ts &= (S-1) \text{ x } N \\ &= T - N \qquad \text{(Equation C.4)} \end{aligned}$$

Suppose A_1 is transmitted after the (S – 1) in-between packets. The time required for the transmission of the second packets of other connections and the first packet of connection A, *tsa*, is

$$\begin{aligned} tsa &= ts + N \\ &= T \qquad \text{(Equation C.5)} \end{aligned}$$

Since maximum node delay for a packet is ($T + N - 2$), the maximum time between the arrival of the last cell of A_1 at the input port and the start of the transmission of the in-between packets, *tdl*, is

$$\begin{aligned} tdl &= (T + N - 2) - tsa \\ &= N - 2 \end{aligned} \qquad \text{(Equation C.6)}$$

As the size of A_1 is N, the maximum time between the arrival of the first cell of A_1 at the input port and the start of the transmission of the in-between packets, *tdf*, is

$$\begin{aligned} tdf &= tdl + N \\ &= 2N - 2 \end{aligned} \qquad \text{(Equation C.7)}$$

Assuming the output port immediately starts transmitting the in-between packets once the transmission of the first packets of all connections excluding A finishes, A_1 can arrive at the node ($2N - 2$) slots before the transmission of the first packets of other connections finishes. Since it takes ($T - S - N + \lceil 3N / Su \rceil - 1$) (by Equation C.3) to transmit the first packets of all connections excluding connection A, in order for A_1 to satisfy the maximum node delay bound, there must be at least *tdmin* slots between the earliest time that A_1 can start arriving at the node and the arrival of the last cell of the first packets of all other connections, where *tdmin* is

$$\begin{aligned} tdmin &= (T - S - N + \lceil 3N / Su \rceil - 1) - (2N - 2) \\ &= T - S - 3N + \lceil 3N / Su \rceil + 1 \end{aligned} \qquad \text{(Equation C.8)}$$

As a result, the smallest value that the virtual arrival time of the first cell of an in-between packet can take must be VAT_{min}, which is ($T - S - 3N + \lceil 3N / Su \rceil$ + 1) slots later than the virtual arrival time of the previous packet of the same connection if the previous packet is not an in-between packet. Otherwise, this in-between packet can cause a packet of another connection which arrives before VAT_{min} to be delayed more than the maximum node delay bound. Therefore, an extra component of *tdmin* derived by Equation C.8 must be added to compute the virtual arrival time of an in-between packet whose previous packet is not an in-between packet if there is no idle slot between the arrivals of these two packets.

If there are any idle slots between the two packets, the number of the idle slots should be subtracted while computing the virtual arrival time of this in-between packet so that it does not have a larger VAT than necessary. In addition, if the size of the in-between packet is Z cells less than N, the value Z should also be subtracted while computing the virtual arrival time of this in-between packet because this packet will cause other packets to be delayed Z slots than it does when its size is N cells.

C.2 AN IN-BETWEEN PACKET AFTER ANOTHER IN-BETWEEN PACKET

Figure C.2 illustrates a system with N = 4, S = 3 and T = 12. The first frame starts with the arrival of the first cell of the packet A_1. After the arrival of the cell A_{12}, the quota for connection A drops to 2. When the packet A_2 arrives, its size is greater than the quota for connection A. Therefore, the packet A_2 is an in-between packet. A_2 consumes 2 slot from the quota of the first frame and 2 slots from the quota of the second frame. When A_3 arrives, its size is again greater than the quota for the connection A. Thus, the packet A_3 is also an in-between packet. A_2 is an in-between packet after a non-in-between packet. According to FVATS, the virtual arrival time of packet A_2 turns out to be equal to its actual arrival time. A_3 is an in-between packet after an in-between packet. Hence, the last cells of A_2 and A_3 reside in two different frames. FVATS has to ensure that there are at least $T - N$ idle slots between the virtual arrival time of the last cell of A_2 and the arrival time of the first cell of A_3 in order to guarantee that the virtual arrival times of A_2 and A_3 are in different frames. As a result, FVATS first assigns to packet A_3 a virtual arrival time equal to its actual arrival time. Then, it finds the virtual interarrival time tvint_3 between A_2 and A_3. It adds $(\mathrm{T} - \mathrm{N} - \mathrm{tvint}_3)$ to the VAT of the last of the last cell of the packet A_3 and obtains the virtual arrival time of packet A_3 as shown in Figure C.2.

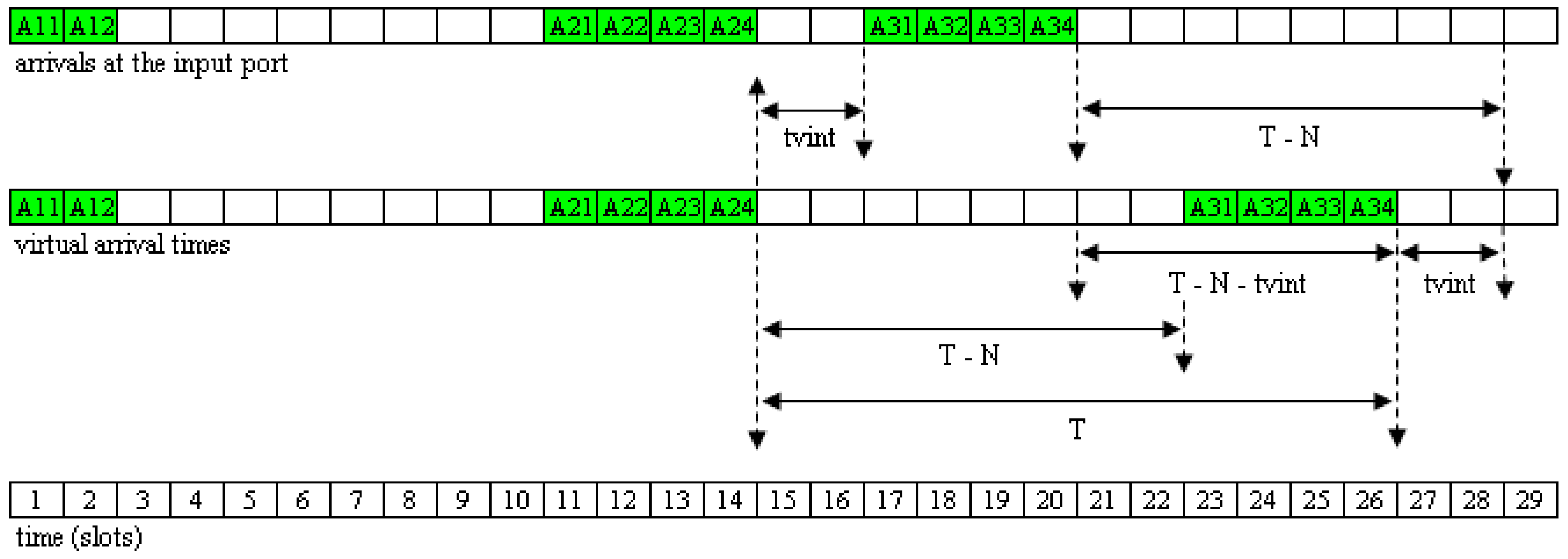


Figure C.2 Virtual arrival time assignment to in-between packets after in-between packets.