EVALUATION OF CORE STATELESS GUARANTEED FAIR NETWORK ARCHITECTURE

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

BY

MUSTAFA İLHAN AKBAŞ

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

DECEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

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

Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI
Supervisor

Examining Committee Members

Prof. Dr. Semih Bilgen	(METU, EEE)	
Asst. Prof. Dr. Cüneyt F. Bazlamaççı	(METU, EEE)	
Prof. Dr. Hasan Güran	(METU, EEE)	
Dr. Ece Schmidt	(METU, EEE)	
Semih Gül (MSc)	(ASELSAN, HC)	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mustafa İlhan AKBAŞ

ABSTRACT

EVALUATION OF CORE STATELESS GUARANTEED FAIR NETWORK ARCHITECTURE

AKBAŞ, Mustafa İlhan
M.S., Department of Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI

December 2006, 106 pages

The problem of providing Quality of Service (QoS) in the Internet has been an extremely active area of research and various mechanisms have been proposed related to this subject. Developing network applications have requirements such as bounded delay, jitter, minimum bandwidth and maximum loss rate. There is also a need to support large bandwidth networks because of growing link speeds. Previous QoS efforts did not fully satisfy all these needs of future networks but more recent approaches aim to be both scalable and rich in the provision of guaranteed services. Consequently core-stateless systems received much attention in recent years because of their scalability in supporting per-flow QoS. The property of not maintaining any per-flow state in the core routers is known as being core-stateless.

In this thesis study, the need for core-stateless network architectures is pointed out and a literature survey about these schemes is carried out. Core-Stateless Guaranteed Fair (CSGF) network architecture, which provides deterministic fairness guarantees in a work-conserving manner, is selected and evaluated. Simulation studies about stateful Virtual Clock (VC) algorithm and CSGF's sub-protocols Core-Stateless Virtual Clock (CSVC), Core-Stateless Guaranteed Throughput (CSGT) and Core-Stateless Guaranteed Fairness (CSGF) are presented. Finally, the deficiencies in fairness of CSGF are demonstrated.

Keywords: Quality of Service, IP Networks, Core-Stateless Network Architectures, Performance Evaluation, Network Simulation

ÖZ

GARANTİLİ ADİL DURUM BİLGİSİZ MERKEZ AĞ MİMARİSİ DEĞERLENDİRMESİ

AKBAŞ, Mustafa İlhan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Cüneyt F. BAZLAMAÇCI

Aralık 2006, 106 Sayfa

İnternette hizmet niteliği sağlama sorunu son derece aktif bir araştırma konusu olmuştur ve bu konuyla ilgili birçok mekanizma önerilmiştir. Gelişen ağ uygulamalarının sınırlı gecikme, seğirme, asgari bant genişliği ve azami paket kayıp oranı gibi gereksinimleri vardır. Aynı zamanda büyüyen bağlantı hızları nedeniyle ağların geniş bant genişliklerini de desteklemeleri gerekmektedir. Geçmişte bu konuda yapılan çalışmalar gelecekteki ağların bu ihtiyaçlarının tamamını aynı anda karşılayamamıştır ancak yeni yaklaşımlar hem verdikleri servislerde zengin, hem de ölçeklenebilir olmayı hefeflemektedirler. Dolayısıyla durum bilgisiz çekirdek sistemler akım başına hizmet niteliği sağlama konusundaki ölçeklenebilirlikleriyle son yıllarda üzerinde durulan yaklaşımlar olmuşlardır. Durum bilgisiz olma, ağ merkezindeki yönlendiricilerde durum bilgisi bulundurmama özelliği olarak

bilinmektedir. Bu tez çalışmasında, durum bilgisiz çekirdek ağ mimarilerine olan ihtiyaç işaret edilmiş ve durum bilgisiz belirlenimci adillik garantileri veren çekirdek mimarileri ile ilgili bir literatür taraması gerçekleştirilmiştir. Bir durum bilgisiz çekirdek mimari olan ve iş koruyan bir şekilde belirlenimci adillik garantileri veren CSGF ağ mimarisi seçilmiş ve ayrıntılı olarak incelenmiştir. Daha sonra, durum bilgili Sanal Saat (VC), ve CSGF'nin alt protokolleri olan Durum Bilgisiz Çekirdek Sanal Saat (CSVC), Durum Bilgisiz Çekirdek Garantili İş (CSGT) ve Durum Bilgisiz Çekirdek Garantili Adil (CSGF) ağ mimarileri ile ilgili benzetim çalışmaları verilmiştir. Son olarak CSGF'nin adillik davranışındaki kusurlar ortaya konmuştur.

Anahtar Kelimeler: Hizmet Niteligi, IP Ağları, Durum Bilgisiz Merkez Ağ Mimarileri, Başarım Değerlendirmesi, Ağ Benzetimi

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACK	Acknowledgment
BCSQ	Bin-based Core-Stateless Fair Queuing
BE	Best Effort
CBR	Constant Bit Rate
CHOKe	CHOose and Keep/Kill
CJVC	Core-Jitter Virtual Clock
CSFQ	Core-Stateless Fair Queuing
CSGF	Core-Stateless Guaranteed Fair
CSGR	Core-Stateless Guaranteed Rate
CSGT	Core-Stateless Guaranteed Throughput
DSCP	Differentiated Services Code Point
DiffServ	Differentiated Services
DPS	Dynamic Packet State
FIFO	First In First Out
GR	Guaranteed Rate
IntServ	Integrated Services
IP	Internet Protocol
ISP	Internet Service Provider
JVC	Jitter Virtual Clock
OPNET	“Optimum Network Performance” Simulation Program
RFQ	Rainbow Fair Queuing
RSVP	Resource Reservation Algorithm
rsVC	Reduced Stateless Virtual Clock

SCORE	Stateless Core
SFQ	Start-time Fair Queuing
sVC	Stateless Virtual Clock
TCP	Transport Control Protocol
ToS	Type of Service
TUF	Tag-based Unified Fairness
UDP	User Datagram Protocol
QoS	Quality of Service
VC	Virtual Clock
VTRS	Virtual Time Reference System
WFQ	Weighted Fair Queuing

CHAPTER 1

INTRODUCTION

The growing Internet has brought many new and challenging network applications such as teleconferencing, interactive gaming, distance learning, Internet telephony, real-time multimedia playing, distributed computing and distributed database applications.

The development of high-speed networks opened a new research field, which is providing quality of service (QoS) for network applications [1]. Timely and satisfactory information delivery over a decentralized and shared network is challenging and complicated. A network that is originally designed for best-effort traffic such as the Internet makes things even worse.

Core-stateless QoS approaches received much attention in recent years because of their scalability in supporting per-flow QoS. There exist many efforts in the literature on Core-Stateless Quality of Service architectures. In this thesis work, following the presentation of the literature survey, a Core-Stateless Guaranteed Fair (CSGF) network architecture, which provides deterministic fairness guarantees in a work-conserving manner is selected and evaluated in detail. Simulation studies about Virtual Clock (VC), Core-Stateless Virtual Clock (CSVC), Core-Stateless Guaranteed Throughput (CSGT) and Core-Stateless Guaranteed Fairness (CSGF) are presented and the deficiencies in fairness of CSGF are demonstrated.

CSVC, which is the core-stateless version of VC, forms the basis of the idea behind CSGT and CSGF. The reasons behind the selection of CSGF for evaluation can be listed as follows:

i. This protocol is one of the recent ones among the surveyed protocols,
ii. It is classified as the first work-conserving core-stateless architecture that provides deterministic fairness guarantees and
iii. The author believes that there are deficiencies about its fairness concept.

CSVC, CSGT and CSGF have not been investigated in a simulation environment before and therefore CSGF is evaluated and its deficiencies from fairness point of view are demonstrated through our simulation study.

The main contributions of this thesis work can also be listed as follows:

- Firstly, the study presents a detailed literature survey for core-stateless network architectures.
- The router of the guaranteed service, stateful QoS architecture (VC-Virtual Clock) is implemented in OPNET simulation environment.
- Implementations in OPNET of the routers (edge and core) for Core-Stateless Virtual Clock (CSVC), the routers (ingress, egress and core) and the Sequencer for Core-Stateless Guaranteed Throughput (CSGT) network architecture and the routers for Core-Stateless Guaranteed Fair (CSGF) network architecture have been realized. To the best knowledge of the author, no Core-Stateless QoS router has been implemented in OPNET before.
- Finally, Core-Stateless Guaranteed Fair QoS Architecture is evaluated to gain insight into its operation and to investigate its fairness.

The organization of the thesis is as follows:

CHAPTER 2 presents the fundamentals of Quality of Service (QoS) in the Internet. The basics of QoS are described and traditional QoS protocols are presented and compared.

CHAPTER 3 investigates the core-stateless approaches and describes most popular solution approaches. These are also the Core-Stateless QoS Architectures that are most relevant to CSGF. The properties, advantages and disadvantages of these approaches are given in order to explain the reasoning behind the selection of CSGF.

CHAPTER 4 concentrates on the selected Core-Stateless QoS Architectures. Main ideas, approaches to provide guaranteed services, and fairness approaches of CSGF are explained in detail. The operations of the algorithms used in CSVC, CSGT and CSGF are described.

CHAPTER 5 firstly explains the simulation environment and the metrics used in this study. The implementation details are given in the second part of this chapter. Verification for the correctness of the associated router implementations is also given. The implemented routers, hence four types of core-stateless QoS architectures (VC, CSVC, CSGT, and CSGF), are added to OPNET simulation environment.

CHAPTER 6 gives an investigation of the CSGF QoS architecture and points out some deficiencies. The details of the simulation study and experiments are given including results of simulations and comments on the results.

Finally, CHAPTER 7 concludes the thesis with a summary of the performed study, with comments on the evaluation and some possible future research directions.

CHAPTER 2

QUALITY OF SERVICE IN THE INTERNET

2.1. Problem Description

The intuitive definition of Quality of Service represents quantities like how fast data can be transferred, how much the receiver have to wait, how correct the received data is likely to be and how much data is likely to be lost.

Current Internet applications such as multimedia have a developing nature and QoS issue in the Internet has been introduced with this nature of the Internet [2]. Current Internet is unable to support the needs of developing applications. As the Internet gets more commercial and global, users start to be ready to pay more to get better service and use multimedia applications through Internet. ISPs want to have a range of services such that the users can get a degree of service quality proportional to the price they pay. Therefore, different traffic flows on the Internet need different service.

The most important QoS parameters are rate, latency, jitter, error rate and loss rate [3]. These are defined as follows:

Table 2-1 Network QoS Parameters

Parameter	Description
Rate	The desired bit rate (bps) or bandwidth
Latency	Delay encountered by a packet, the sum of transmission delay, processing delays (includes router look-up), queuing delay etc.
Jitter	Variations in latency
Error Rate	The percentage of packets received in error
Loss Rate	Percentage of packets dropped or lost during end-to-end transmission

Each application has its unique QoS needs. QoS needs of a flow depend on information type it uses and application or end-user specific requirements. The applications of today require connections with certain quality. Whether this quality can be realized depends on available network resources, network properties and available end-system resources.

Current Internet supports Best Effort (BE) datagram delivery only. The Internet architecture is composed of stateless routers, which means the routers do not maintain any state about traffic except the routing state. This structure makes Internet scalable and robust but no guarantees can be made to real-time or multimedia traffic [4]. Two important QoS approaches are presented in the Internet QoS history: IntServ and DiffServ; but as the applications and technologies change, new QoS architecture approaches arise.

2.2. Background of Quality of Service Problem

2.2.1.Integrated Services (IntServ)

High-speed networks have enabled new applications and they need to deliver assurances from the network. Applications that are sensitive to the timeliness of data are called real-time applications examples of which are voice and video.

IntServ [5] enhances both Internet's service model and architecture model. The old service model in the Internet uses only a single best-effort service class, but the IntServ service class uses multiple service classes including best-effort class and QoS classes. The key architectural difference is the stateful structure of IntServ. IntServ routers maintain per flow states at routers. These states are setup by a signaling protocol and used for admission and scheduling purposes [6].

In IntServ, each flow has a fixed path and routers along the path maintain the state of the flow. This fixed path structure relies on the resource reservation that is handled by the Resource Reservation Algorithm (RSVP). RSVP [7] is used for setup and tear-down of the reservation state, it is a protocol for establishing a guaranteed QoS path between a sender and receiver(s), i.e. it establishes end-to-end reservations over a connectionless network. It is robust when routers/links fail. The traffic is re-routed and new reservations are established in the fail condition. It is receiver-initiated and so scales well for multicast.

The basic operation of RSVP is as follows:
Sender sends PATH message via the data delivery and each router adds its state and the address of the previous hop. Receiver sends RESV message on the reverse path specifying the reservation style, QoS desired and setting up the reservation state at each router.

RSVP, Admission control and Traffic Control are the main components of IntServ solution. Admission control mainly determines if there is enough resources in the network for the new flow. Traffic control classifies the packets to each flow and schedule packet transmission according to the state [8].

2.2.2.Differentiated Services (DiffServ)

DiffServ [9], which is proposed by IETF [10], is not based on resource reservation but prioritization. The packets are evaluated according to their DS field and their flows are not considered. DS field is the TOS (Type of Service) byte in the header of IPv4 packets [11] or Traffic Class byte of IPv6 packets ([12, 13]). Last two bits are not used and first 6 bits, called Differentiated Services Code Point (DSCP), are used for specifying QoS requirements.

The traffic is classified into a small number of classes (traffic aggregations) in DiffServ and no state information is used. Because of these properties, DiffServ scales well. However, since there is no explicit resource reservation, QoS guarantees are difficult to achieve and hence DiffServ model does not attempt to guarantee QoS but rather it provides a relative servicing.

There is a distinction between edge and core nodes in Diffserv. Edge routers are the routers at the network boundaries. The edge router classifies the packets entering the network according to their DS fields and then according to the Service Level Agreement (SLA) between the ISP and the customer, it (re)marks the packets if necessary and polices the flow for its agreement compliance.

The core routers are responsible for forwarding only. Although DiffServ model permits the mechanisms of the edge routers to be implemented in core routers, this

makes the core routers more complicated which is undesirable. Thus in Diffserv, edge routers are more intelligent and more complicated but core routers are simpler.

2.3. New QoS Approaches

There are important drawbacks of the existing QoS Architectures. The drawback of the stateful solutions is their complexity. On the control path, the routers should install and maintain per-flow state for data and control planes. Also on the data path, per-flow classification, per-flow buffer management and per-flow scheduling should be handled. It is a challenge to keep per-flow state consistent in the routers.

Opposite to stateful architectures described above, stateless solutions are more scalable and robust. However stateless solutions can not provide services as powerful and flexible as stateful solutions. They also can not provide low delay guarantees and high resource utilization simultaneously.

It is easy to see that since stateful solutions (e.g. IntServ) are rich in services, stateless solutions are more scalable. New approaches try to combine good properties of both architectures. The goal is having a scalable QoS architecture that is rich in services.

Core stateless systems have received considerable attention since 1999 also for supporting per-flow QoS guarantees. The core-stateless systems use scalable mechanisms in the core of networks and stateful approaches at the edges of the network.

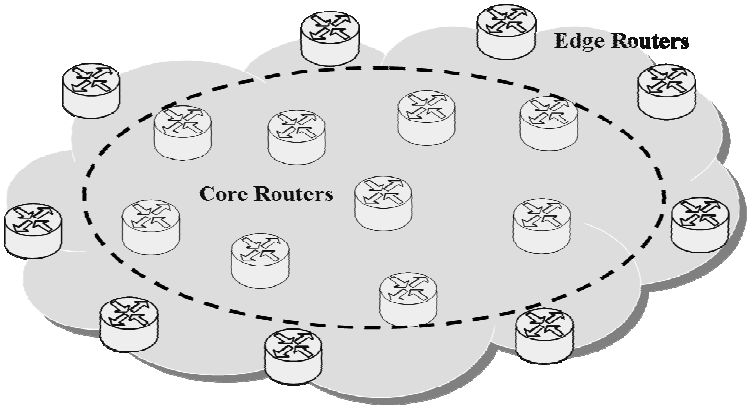

Figure 2-1 Core-Stateless Network Architecture

In core stateless systems, the node structures are defined as in DiffServ networks (Figure 2-1). The core nodes are simpler and they do not maintain per-flow state. Therefore in core routers there is no function like per-flow classification, per-flow queuing and per-flow scheduling. The edge nodes are more complex and keep per-flow state in core-stateless networks. The main idea is keeping the complexity out of the network core.

CHAPTER 3

CORE STATELESS QoS ARCHITECTURES

3.1. Problem Description

QoS architectures aim to support the needs of users and their applications, and the continuous growth of Internet applications demanding more number of reliable resources has resulted in the proposal of many new QoS architectures. Some of the most common design constraints of new QoS solutions are maximum guaranteed service, minimal complexity at the routers and minimum change in the existing protocols.

Together with the concept of high-performance networking, quality of service (QoS) architectures have become an important research issue. In addition to the common and ordinary QoS architectural design goals stated in proposed protocols like IntServ and DiffServ, a new QoS architecture should consider QoS constraints delay, jitter, bandwidth and packet loss in a scalable manner.

There has been more interest in services requiring certain QoS from networks, such as multimedia services providing audio and video traffic. Contrary to these QoS requirements, many proposed protocols provide QoS either in a small network or in poor granularity. Therefore we believe that the core-stateless solutions seem to be the future of QoS in the Internet.

The following sections, 3.2 and 3.3, present and compare important Core-Stateless QoS architectures proposed in the literature. Some of these are also important in the way they relate to the main topic of this thesis, namely the Core-Stateless Guaranteed Fair Network Architecture. The protocols are described and compared stating their differences in giving QoS services.

3.2. Solution Approaches

3.2.1. Core-Jitter Virtual Clock (CJVC)

CJVC [14, 15] aims to implement guaranteed services with levels of flexibility, utilization, and assurance similar to those provided with per-flow mechanisms. CJVC is a non-work-conserving QoS architecture. This means that the excess bandwidth is not used by the flows, a flow uses the bandwidth at most at its reserved rate even if the rest is idle. CJVC provides end-to-end delay, jitter and throughput guarantees (on average) at the expense of this non-work-conserving character.

A network architecture, called Scalable Core (SCORE), which is similar to the DiffServ Architecture is used in CJVC. SCORE [15] is a network in which edge nodes perform per flow management but core nodes do not. The approach firstly defines a stateful network that implements the desired rich services. Then the functionality of the reference network is tried to be emulated in a SCORE network.

In order to get rid of the per-flow state at core routers, the idea of “having packets carry per-flow state”, namely Dynamic Packet State (DPS) [15], is used in CJVC. With this technique, ingress node computes and inserts flow state in packet’s header, core nodes process a packet based on the state it carries and the state of the node itself, and updates both packet and node’s state. Egress node removes state from

packet's header. By using SCORE and DPS [15], CJVC provides unicast IntServ guaranteed service semantics with DiffServ-like scalability.

CJVC algorithm aims to approximate a network with each router implementing Delay-Jitter-Controlled Virtual Clock (Jitter Virtual Clock) on the data path and per-flow admission control on the control path. This network is chosen to be emulated since in Virtual Clock ([16], [17]), a packet's deadline depends only on the state variables of the flow it belongs to but not on the variables of other flows and this property makes it easier to convert VC to a core-stateless version. JVC is a non-work-conserving algorithm, which means that it has no statistical multiplexing property. CJVC inherits its non-work-conserving behavior from JVC.

DPS is used to approximate the Jitter-VC with CJVC that does not require core routers to maintain per flow state. The components of Jitter-VC are a delay-jitter rate-controller and a Virtual Clock scheduler. The algorithm assigns each packet an eligible time and a deadline upon its arrival. The packet is kept in rate-controller until it becomes eligible and then the scheduler schedule packets in increasing order of their deadlines. It is important to note that the algorithm eliminates the delay variation (jitter) of different packets by forcing all to incur maximum allowable delay. Jitter-VC guarantees that no packet misses its deadline. A network of Jitter-VC servers can provide the same delay guarantees as a network of Weighted Fair Queuing (WFQ) ([18],[19, 20]) servers.

The intuition of defining the eligible time and deadline of a packet belonging to a flow with reservation r is equal to the start and finish times of transmitting the packet in an ideal network in which the flow has dedicated links of capacity r. The eligible time is chosen as the maximum of the following: The arrival time, The sum of the packet's deadline at previous node and propagation delay, the previous packet's deadline at current node. Deadline is chosen as the sum of eligible time and (packet length) / (flow reserved rate).

CJVC aims to emulate a network of JVC routers without maintaining per flow state at core routers. In order to do this, the dependence on previous packet deadline has to be eliminated. This challenge is solved by introducing a slack variable s such that at each core node:

(packet deadline at prev. node + prop. delay + s) >= (deadline of previous packet)

Using this method, the eligible times and deadlines at the last hop are the same in both CJVC and Jitter-VC, i.e. CJVC and Jitter-VC provides the same worst case end-to-end delay bounds. The slack variable can be computed at ingress node and it depends on previous and current packet, slack variable associated to previous packet and the number of hops.

As a result, in CJVC algorithm:
Each packet carries in its header three variables

- slack variable s (inserted by ingress)
- flow's reserved rate (inserted by ingress)
- ahead of schedule (inserted by previous node)

Eligible time = (arrival time + ahead of schedule (deadline – departure time) + s)
Deadline = eligible time + (pckt. length) / (flow rate)
(arrival time + ahead of schedule at previous node) = (deadline at previous node + propagation delay)

CJVC algorithm eliminates the need to maintain per-flow classification and per-flow buffer management and per-flow scheduling on the data path at core routers. Actually per-flow classification is not needed anymore, there is only one buffer and not per-flow but per-packet scheduling is needed. In addition, the per-flow state on the control path is not needed to be installed and maintained on the control path.

In CJVC, a distributed admission control approach which depends on a light-weight signaling protocol is presented in order to eliminate the need for per-flow state on the control path. When this approach is used, each node keeps track of aggregate reservation rate for each outgoing link and makes local admission control decisions. A close upper bound on the aggregate reserved rate is estimated and by using this estimation, over-provisioning is avoided. The admission control algorithms used are robust against losses and partial reservation failures, they do not under-estimate the rate and they are self-correcting.

The results of the simulations in [21] show the non-work-conserving characteristic of CJVC algorithm and the delay guarantee provided by CJVC in the presence of aggressive best effort sources. The illustrations for admission control algorithms demonstrate the accuracy of estimations made in the algorithms and the computation of the upper bound on the rate at different conditions.

With respect to its characteristics, improvements, experimental computations and performance, CJVC is considered to be an important example of Core-Stateless Guaranteed Rate QoS Architecture. CJVC is referred by most of other core-stateless network architectures that follow it [21-23]. CJVC is also important in order to understand CSGF, which is the chosen and the investigated algorithm in this thesis. Hence CJVC is described in detail in this chapter.

3.2.2. CHOose and Keep/Kill (CHOKe)

As all other Core-Stateless Fair QoS architectures, CHOKe [24] (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows) is motivated by the need for a simple algorithm that can achieve flow isolation and/or approximate fair bandwidth allocation.

CHOKe penalizes misbehaving flows by dropping their packets. The total occupancy of the buffer is the only constraint for the penalty decision. CHOKe marks two thresholds on the buffer, a minimum threshold *minth* and a maximum threshold *maxth*.

If the average queue size is less than a pre-selected minimum, each arriving packet is queued into the FIFO buffer. When the average queue size is larger than the minimum, CHOKe [24] draws a packet from the FIFO buffer at random and compares it with the arriving packet. If both of the packets belong to the same flow, both are dropped, else the randomly chosen packet is not dropped and the arriving packet is admitted into the buffer with a probability that depends on the level of congestion, i.e. average queue size. Packets are definitely dropped if they arrive when the average queue size exceeds *maxth*.

The reasoning that forms the basis of this process is that a misbehaving flow has more packets than the others in the FIFO buffer and since packets of a misbehaving flow arrive more numerously, they are more likely to trigger comparisons. Therefore, packets of misbehaving flows are dropped more often than packets of well-behaved flows.

CHOKe's performance is improved by choosing more than one drop candidate packet. A process for the determination of the number of packets to choose is also given in CHOKe.

Two different models of CHOKe are presented in [24]: Front and Back CHOKe. Front CHOKe compares an incoming packet with the packet at the head of the buffer, while Back CHOKe compares it with the last packet at the buffer. Back CHOKe also drops incoming packets, but not packets from the buffer.

Simulations in [24] evaluates the performance of CHOKe for a single congested link, multiple congested links and multiple misbehaving flows. Simulation results show that well-behaved flows are protected from misbehaving flows in CHOKe. However the simulations show that a high-speed UDP can still get several times more bandwidth than it deserves in CHOKe.

CHOKe only addresses average fair share of bandwidth rather than weighted share [25]. CHOKe defines mechanisms as simple as Random Early Discard (RED, [26]) in the core routers. However, it improves but doesn't solve the fairness issue. While CHOKe is very simple to implement and does not require edge routers to maintain any per flow state, it has difficulties to accurately approximate fair queuing when the number of flows is large or in the presence of very high-speed flows .

3.2.3. Virtual Time Reference System (VTRS)

Virtual Time Reference System (VTRS) [22] is inspired by CJVC and aims to provide guaranteed services using DiffServ [9] paradigm. For this purpose, packet virtual time stamps that require no state for computation are used. Per-hop behaviors of core routers are defined via these packet virtual time stamps and this characterization provides end-to-end delay bounds. Packet virtual time stamps are computed by the state carried in the packet which makes the algorithm core-stateless.

VTRS architecture is composed of three main components which are packet state, traffic conditioning at the edge, per-hop virtual time reference system/update mechanism at the core. Packet state includes reserved rate or delay value of the flow, time stamp and a virtual time adjustment term. It's inserted in the packet header at the network edge after traffic conditioning. A flow is guaranteed to enter the network with no more than its reserved rate with the edge traffic conditioning. Per-

hop virtual time reference/update mechanism maintains the continuing progression of the virtual time embodied by the packet virtual time stamps.

The idea of virtual time has been used in many packet scheduling algorithms that require maintaining per-flow information. This notion is viewed as global in VTRS. By the virtual time spacing property of virtual time stamps at the core routers, the reserved rates of the flows are preserved at the core routers.

VTRS is defined to serve as a unifying scheduling framework where different scheduling algorithms can be employed. VTRS characterizes per-hop behaviors of core routers and the end-to-end properties of their concatenation in order to support delay guarantees. The bound on the end-to-end delay in a VTRS network experienced by packets of a flow can be expressed in terms of the reservation rate and the error terms of the routers along the path.

VTRS aims to provide deterministic delay and throughput guarantees but not fairness guarantees. Using VTRS, a bandwidth broker architecture for supporting admission control and QoS provisioning is also presented by the authors of VTRS in a different work [27].

3.2.4.Bin-based Core Stateless Queuing (BCSQ)

Existing core stateless packet schedulers require core routers to keep the received packets in sorted order based on their virtual finish times. This sorting operation can be quite expensive when the packet queue is long, which is not desirable in high-speed backbone networks. BCSQ [22, 28, 29] is developed within VTRS and tries to overcome this complexity.

Virtual time space is divided into equal slots or bins in a BCSQ scheduler partitions. A packet is placed into a bin if its virtual finish time is in that bin. Bins are ordered

and served according to the time intervals they represent. Packets in a bin are served in a FIFO manner.

The minimum number of bins to prevent over flows is investigated and simulation studies are also given to evaluate the performance of BCSQ. Simulation results show that by controlling the length of time intervals the bins represent, BCSQ has many trade-offs between performance and complexity. When the bin time intervals are sufficiently long, all incoming packets will fall in a single bin and BCSQ behaves just like a FIFO scheduler. As the length of time intervals decreases, BCSQ is able to provide improved per-flow QoS guarantees at the expense of greater scheduling complexity.

3.2.5. Stateless Virtual Clock (sVC)

Stateless Virtual Clock [23] concentrates on providing delay guarantees in a scalable manner. The authors of Stateless Virtual Clock investigates DPS and aims to provide similar delay bounds with a smaller amount of per packet overhead. Stateless Virtual Clock aims to approximate a virtual clock algorithm and two variants of Stateless Virtual Clock are proposed.

In Stateless Virtual Clock, it is stated that the DPS technique requires complex per packet processing in the scheduler which may cause problems in achieving high rates. Edge and core nodes behave like in other core-stateless architectures: While edge nodes shape guaranteed service flows and put the reserved rate in the header field, core nodes implement a simple scheduler. Two variants of the approach are proposed. In [23], it is worth noting that since these algorithms are approximations, they don’t give any deterministic guarantees.

Stateless Virtual Clock (sVC), approximates Virtual Clock [16] algorithm in a very simple manner, but this approach can be incorrect when the network jitter is large. The variant, Reduced State Virtual Clock (rsVC), requires more packet overhead and it needs to identify the flow of a packet, but behaves better than sVC especially in high-load conditions. The quality of the approximation depends mostly on the network load; sVC and rsVC do not behave well under severe load conditions.

From the simulation studies given in [23], it is seen that the Stateless Virtual Clock approximations, sVC and rsVC, give similar average delay guarantees with VC in non-severe situations (e.g. %96 network load) and rsVC performs better than sVC. However maximum delays are larger in sVC than VC.

3.2.6. Rainbow Fair Queuing (RFQ)

Rainbow Fair Queuing [30] divide each flow into a set of layers, based on rate. It is a combination of a color labeling scheme and a buffer management mechanism. The packets in a flow are colored at an edge router with a layer label. The state information carried by the packets is the color layers they belong to, rather than the explicit rate of their flows. The larger the number of colored layers, the higher the rate of the flow and flows with the same rate have the same number of colored layers. The colored layers provide a structure for controlled discarding in the network when congestion occurs. The core routers operate in FIFO fashion.

A core router maintains a color threshold and core routers only need to perform a simple operation, packets with a color label larger than this threshold are dropped. The discarding starts with the packets with the highest color value. During congestion, the color threshold is decreased; when congestion clears, the color threshold is increased. Because the coloring is based on rate, the discarding of packets is approximately fair.

Simulations in [30] present the performance of RFQ against CSFQ [31]. In these simulations, the performance of RFQ scheme is comparable to CSFQ when the application data does not contain any preferential structure. RFQ outperforms CSFQ when the application takes advantage of the coloring to encode preferences.

RFQ discusses average fair share and shows only the performance of its weighted version with an all-UDP case. There is no deterministic guarantee for its performance when both TCPs and UDPs of different weights and RTTs coexist [25]. RFQ avoids fair share rate calculation in the core routers and that is better adapted to layered encoding applications. It removes flow state but requires computation to determine dropping thresholds.

3.2.7. Tag-based Unified Fairness (TUF)

TUF [32] aims to realize the fair bandwidth sharing without per flow state in the routers, using a trivial queuing discipline. Packets are tagged near the source, depending on the nature of the flow. In the core of the network, routers use FIFO queues, and simply drop the packet with the highest tag value in case of congestion. TUF does not try to maintain instantaneous flow rates equal but takes into account the responsiveness nature of the flows, and adjust loss rates such that average rates are equal. TUF also differentiate between TCP and UDP flows in order to avoid the TCP flows being over-penalized due to their response to losses.

TUF allocates bandwidth max-min fairly if it is not possible to increase the satisfaction, namely average bandwidth, of a flow without simultaneously causing a decrease in the satisfaction of a less satisfied flow. In max-min fairness, small users get all they want and large users share the rest of the resources evenly. TUF is concerned with fairness between elastic flows for which the satisfaction is measured in terms of average rate and not instantaneous rate.

In TUF, the loss rates are differentiated to provide fair bandwidth allocation between flows sending at different rates. The state information called tag is carried in one of the packet's fields. This tag is numeric a value that represents the minimum fair share rate a router must support.

The congested core router uses the tags of the packets present in its queue to make a drop decision. Decision procedure is very simple as dropping the highest tag value when the queue is full. A tagging entity, called the "tagger" is responsible for placing a tag in each packet. This entity, that maintains flow state, is either a router at the edge of the network, or ideally the source itself.

Simulations given in [32] show that TUF achieves "approximately fair bandwidth sharing" as CSFQ, DRR and SFQ ([33]). It adapts to responsive flows whose throughput can be determined as a function of the loss rate. Therefore in heterogeneous environments, with non-negligible round trip times or bursty traffic, it provides better fairness than other stateless fair queuing algorithms that adapts instantaneous rates.

3.2.8. Core-Stateless Fair Queuing (CSFQ)

CSFQ [31] aims to use core-stateless network architecture to approximate the functionality of a network in which all nodes implement FQ. In CSFQ each edge node estimate the incoming rate of each flow based on exponential averaging and use it to label flow's packets. All nodes, both edge and core nodes, periodically estimate the fair rate along the outgoing link.

When the link is congested, the fair rate is computed such that the rate of the aggregate forwarded rate equals the link capacity. When the link is uncongested, fair rate is the maximum among the arrival rates of the incoming flows.

Upon a packet arrival each node computes its forwarding probability using the current estimated rate of the flow, which is contained in the packet label, and the fair rate of the output link. Then the packet is forwarded with this probability. To reflect the eventual change in flow's rate, when a packet is forwarded its label is renewed as the minimum between its previous value and the fair rate of the output link. At the next node the label will still represent the estimate rate of the flow's incoming traffic.

Only edge nodes need to perform per flow management, as they need to estimate the rate of each incoming flow. Core nodes need to know only the packet label and the fair rate of the output link.

The edge router’s design is still complicated in CSFQ and because of the rate information in the header, the core routers have to extract packet information differently from traditional routers. In simulations given in [31], CSFQ achieve fair allocations close to FQ and similar to or better than FRED under most scenarios. Simulations are preferred for checking average fair bandwidth sharing.

3.2.9.Core-Stateless Guaranteed Fair (CSGF) Network

To the best of our knowledge, CSGF is the first work-conserving core-stateless network providing deterministic service and fairness guarantees. CSGF is built upon Core-Stateless Guaranteed Rate Network (CSGR) that can provide end-to-end delay guarantees.

In CSGR [21], the upper bounds on packet deadlines at any core node can be computed using only per-flow state at the edge node. It is stated that a CSGR network, depending on this idea, provides same end-to-end delay as the networks using actual deadlines.

CSGR is combined with two mechanisms, namely tag re-use and source rate control, and this combination leads to Core-Stateless Guaranteed Throughput (CSGT) networks [34]. It's shown in [34], that CSGT provides throughput bounds that are comparable with the throughput bounds achieved by a network of core-stateful fair rate routers.

The design of Core-Stateless Guaranteed Fair (CSGF) networks on the other hand depends on two principles. Firstly, a network must provide end-to-end throughput guarantees to provide fairness. Secondly, this throughput guarantee is combined with two other mechanisms, namely fair access at the edge nodes and aggregation of flows in the core nodes

3.3. Overview of Core-Stateless Architectures

The following table presents an overview of the protocols that are summarized above, including their advantages and disadvantages.

Table 3-1 Overview of Core-Stateless Architectures

Architecture	Advantages	Disadvantages	Notes
CJVC [14]	End-to-end delay and jitter guarantee	Non-work-conserving. Higher average delays than stateful algorithms, no throughput or fairness guarantees	Inspired CSGR
CHOKe [24]	Flow isolation Approximately fair bandwidth allocation	A high-speed UDP may get much undeserved bandwidth	Two different Models: Front and Back CHOKe
VTRS [22]	Deterministic delay and throughput guarantees	No fairness guarantees, Non-work conserving	A flow is guaranteed to enter the network with no more than its reserved rate

Table 3-1-Cont. Overview of Core-Stateless Architectures

sVC [23]	Similar average delay guarantees with VC	No guarantees other than delay	Two variants: sVc and rsVC, which tries to approximate VC
BCSQ [28]	Reduced run time for packet insertion,	Great scheduling complexity	It has trade-offs between performance and complexity Developed within VTRS
TUF [32]	Worst-case end-to-end delay bounds, Approximately fair bandwidth sharing	Fairness in terms of only average rates, not instantaneous rates	It provides good fairness in heterogeneous environments, with bursty traffic
RFQ [30]	Average fair bandwidth sharing	Requires computation to determine dropping thresholds	Only its weighted version's performance with an all-UDP case is shown
CSFQ [31]	Average fair bandwidth sharing	Edge router's design is complicated	Inspired many other core-stateless algorithms.
CSGR [21]	Delay and average throughput guarantee	No throughput guarantee at short time-scale	It presents a method to convert every GR architecture to core-stateless
CSGT [34]	CSGR + throughput guarantee at short time-scale	No proportional throughput guarantee	First core-stateless QoS architecture providing throughput guarantee at short time-scale
CSGF [25]	CSGT + fairness guarantee	More complex than CSGT at edge routers	It is claimed to be the first work-conserving core-stateless algorithm providing guaranteed fair services

CHAPTER 4

CORE STATELESS GUARANTEED FAIR NETWORK ARCHITECTURE

In Chapter 3, important Core-Stateless QoS architectures are described and an overview and comparison of these architectures are given. The results summarized show that one of these protocols, namely CSGF, needs to be studied more thoroughly for a better understanding. In this chapter, the pros and cons of its underlying techniques, the assumptions made in the design of CSGF are discussed in detail and CSGF is questioned to see whether it reaches the motivations behind or not.

Since CSGF is actually built on top of CSGT and CSGR, all these are explained in sequence in the following sections.

4.1.Detailed Description

The design of CSGF is basically inspired by the CJVC [14] network that is described in Section 3.2.1. The motivation behind CSGF comes from the evaluation of CJVC. CJVC controls the end-to-end delay by controlling the jitter and having each packet encounter the maximum allowable delay. However this control leads to a non-work conserving structure and the excess bandwidth becomes useless for the

flows. In addition CJVC leads to higher average delays compared to its core-stateful counterparts.

The first goal of using the excess bandwidth and lowering the average delays encountered by flows is achieved by CSGR [21]. The second challenge of providing end-to-end throughput guarantee at finite and short timescales is achieved by CSGT [34]. The last challenge of providing delay, throughput and also proportionate allocation guarantees of spare bandwidth at the same time is achieved by CSGF. The other mentioned core-stateless networks that attempt to be fair provide only statistical (or approximate) fairness over large time-scales and for long-lived flows. CSGF aims to provide a core-stateless network architecture that can provide deterministic end-to-end fairness guarantees to flows.

In the following sections CSGR, CSGT and CSGF will be described in detail.

4.1.1. CSGR

CSGR [21] aims to provide core-stateless version of any GR (Guaranteed Rate) architecture, because delay guarantee is considered essential to provide other type of guarantees. The CSGR algorithm used in this study is Core-Stateless Virtual Clock (CSVC), which is the core-stateless version of the Virtual Clock (VC) [16] algorithm.

CSVC is inspired by CJVC [14], but unlike CJVC, its goal is to make it work-conserving in order to make use of statistical multiplexing. Due to ever increasing network service requirements, better utilization of bandwidth is important. Thus it is desirable to have a work-conserving core-stateless network that provides delay guarantees.

Since CSVC is core-stateless version of Virtual Clock, the first question asked in the design of CSVC is:

"Can the techniques used in deriving CJVC from JVC be applied to derive core-stateless version of VC?"

Virtual Clock uses the following equations in order to define the deadlines of packets [16]:

$$VC_{f,j}^{1} = a_{f,j}^{1} + \frac{l_f^1}{r_f^1} \tag{4.1}$$

$$VC_{f,j}^{k} = \max(a_{f,j}^{k}, VC_{f,j}^{k-1}) + \frac{l_f^k}{r_f} \tag{4.2}$$

where

$VC_{f,j}^{k}$ = VC value of the k[th] packet of flow f in router j

$a_{f,j}^{k}$ = Arrival time of the k[th] packet of flow f to router j

l_f^k = Size of the k[th] packet of flow f

r_f = Reserved rate of flow f

Packets are transmitted in increasing order of their VC values.

When deriving CJVC from JVC, the goal was to get rid of the VC value of the previous packet at the same node in order to make the algorithm core-stateless (see section 3.2.1.).

If the approach used in converting JVC to CJVC is adopted, we should add a slack variable to $a_{f,j}^{k}$ in (4.2) so that the resulting value of the maximum term is always greater than $VC_{f,j}^{k-1}$. Then the slack variable (δ) should satisfy [14]:

$$\delta_f^k \geq VC_{f,j}^{k-1} - a_{f,j}^{k} \tag{4.3}$$

In CJVC [14], non-work-conserving nature of JVC shapes the flows at their reserved rate and holds the packets until their eligible time. Consequently the jitter is bounded and the packets of a flow can not come back-to-back to a router. Conversely, in a network of work-conserving routers, packets can arrive back-to-back to a router. If packets arrive back-to-back, the value of δ in the above equation extremely grows. As a result, a network of such CSVC servers does not preserve the delay guarantee of the corresponding network of VC servers.

Then the design challenge becomes computing deadlines in a core-stateless network that doesn't maintain per-flow state. There are four main principles in deriving "CSVC from VC" [21]:

- Not the deadline itself but an upper bound on the deadline can be computed using only the state of the same packet at the previous node.
- By using the above observation recursively, the upper bound can be computed using only the state of the same packet at the first node.
- Ingress node, which is an edge node, maintains per-flow state.
- The network provides same end-to-end delay if the upper bounds are used instead of the actual deadlines.

The statement in the first bullet above is stated and proved in [14] for any GR algorithm. For CSVC its form is as follows [21]:

$$VC_{f,j}^{k} \le VC_{f,j-1}^{k} + \max_{i \in [1..k]} \frac{l_f^i}{r_f^i} + \pi_{j-1} + \frac{l_{j-1}^{\max}}{C_{j-1}} \qquad (4.4)$$

Where

π_{j-1} = Upper bound on propagation delay of the link connecting node (j-1) and j.

C_{j-1} = Outgoing link capacity at node j.

$l_{j-1}^{\max}$ = Maximum packet length served by node j-1 in bits.

In CSVC, a new term called ’core virtual clock (*VCore*) is defined as follows [21]:

$$VCore_{f,1}^{k} = VC_{f,1}^{k}$$
$$VCore_{f,j}^{k} = VCore_{f,j-1}^{k} + \beta_{f,j-1} + \pi_{j-1} + \max_{i \in [1..k]} \frac{l_f^i}{r_f^i} \qquad (4.5)$$

where

$$\beta_{f,j-1} = \frac{l_{j-1}^{\max}}{C_{j-1}} \text{ for flow } f \qquad , j \geq 2$$

Equation (4.5) can be rewritten as:

$$VCore_{f,j}^{k} = a_{f,j}^{k} + g_{f,j-1}^{k} + \max_{i \in [1..k]} \frac{l_f^i}{r_f^i} \qquad (4.6)$$
$$j \geq 2$$

where

$g_{f,j-1}^{k}$ = the time between the departure and deadline of p_f^k at server *j*-1.

Then $VCore_{f,j}^{k} \geq VC_{f,j}^{k}$. As a result, the need to maintain $VC_{f,j}^{k-1}$ is eliminated and by enabling edge routers to encode the rate *r* and by enabling server *j*-1 to encode $VCore_{f,j-1}^{k}$ in the packet, there is no need to maintain per-flow state in the core routers.

The delay guarantee of a network of CSVC routers is evaluated in [21] and it is proven that it is the same as that of a network of Virtual Clock routers.

To summarize, CSVC modifies the approach used to convert JVC to CJVC in order to make CSVC work-conserving. CSVC provides only end-to-end delay guarantees but it doesn’t provide throughput guarantee at short time-scales or fairness guarantees.

4.1.2. CSGT

Core-Stateless Guaranteed Throughput (CSGT) is the architecture offered to improve CSGR by providing end-to-end throughput guarantees at short time-scales ([34, 35]). Derivation of CSGT is done in two steps. Firstly, in order to provide throughput guarantee in addition to end-to-end delay guarantee, an algorithm providing delay guarantee is chosen to build CSGT on. There are core-stateless algorithms providing delay guarantee in the literature ([36], [21], [22]). In particular, CSVC is selected as the underlying CSGR network. Second, two mechanisms are introduced to be added to CSVC in order to form a work-conserving core-stateless network that guarantees end-to-end throughput bounds within an additive constant of the one obtained by a network of core-stateful routers.

4.1.2.1. End-to-end delay guarantee requirement

Consider a network providing lower-bound throughput guarantee of the form "$W_{f,H}(t_1,t_2) \geq r_f(t_2-t_1)-\Phi$" ($\Phi$ =a constant) to any flow *f*, source of which transmits at least at its reserved rate. If $t_1 = a^1_{f,1}$ (arrival time of the first packet of flow f to the ingress router) and $t_2 = d^k_{f,H}$ (departure time of the kth packet of flow f from the egress router), then:

$$W_{f,H}(a^1_{f,1}, d^k_{f,H}) = \sum_{i=1}^{k} l^i_f \text{ and } d^k_{f,H} \leq a^1_{f,1} + \frac{\Phi}{r_f} + \sum_{i=1}^{k} \frac{l^i_f}{r_f}.$$

Since source transmits at least at its reserved rate, the expected arrival time (EAT) of p^k_f is $EAT_1(p^k_f) = a^1_{f,1} + \sum_{i=1}^{k-1} \frac{l^i_f}{r_f}$. Then the flow *f* is provided with a delay guarantee $d^k_{f,H} - EAT_1(p^k_f) \leq \frac{\Phi + l^k_f}{r_f} \leq \frac{\Phi + l^{\max}_f}{r_f}$ [35].

Hence a network providing lower-bound throughput guarantee of the form " $W_{f,H}(t_1,t_2) \geq r_f(t_2-t_1)-\Phi$ " also provides delay guarantee: $d^k_{f,H} - EAT_1(p^k_f) \leq \frac{\Phi + l^k_f}{r_f} \leq \frac{\Phi + l^{\max}_f}{r_f}$ [35]. Therefore CSGT uses CSGR, which gives delay guarantee, as a building block and enhances it with a set of end-to-end mechanisms that allow the network to retain its delay properties while providing throughput guarantees at short time-scale.

In CSGR, if packets of any flow come back-to-back (which is allowed in a work-conserving structure) and if there is excess bandwidth for that flow to use, then it will be serviced at a rate greater than its reserved rate. When the excess bandwidth is loaded with other flows, this flow will be penalized because of the computation logic of deadline values. Throughput guarantee is important in order to:

- Satisfy bandwidth requirements at short time scales for those applications for which bit-rate requirement may vary considerably (e.g. VBR video) over short time-scales [37].
- Allow sources to transmit data in transient bursts (which will result in better utilization of resources in the network).

It is worth noting that CSVC does guarantee average throughput, so the throughput of any flow in any interval would be below its reserved rate if that flow receives service at a rate higher prior to this interval.

4.1.2.2. Properties of CSGT

CSVC networks must reduce the penalty given to flows because of their extra usage of resources when the network is idle in order to guarantee throughput at short time-

scales. It is obvious that the deadline determination method for a packet must be modified for this purpose. The deadline values encoded in the packets by the ingress router in CSVC algorithm are called service tags. In CSGT, the main principle on the re-use of service tags is as follows [35]:

"Allowing the ingress routers to re-use for future packets the deadline values of packets that reach the destination much prior to their deadlines."

Then in a CSGT network, on receiving a packet of a flow, the ingress router assigns to it a service tag. If the set *R* of re-usable tags is not empty, smallest tag from this set is used. If it is empty, the tag values are assigned just like in CSVC [35]:

$$F_1(p_f^k) = \max(a_{f,1}^k, \hat{F}(a_{f,1}^k)) + \frac{l_f^k}{r_f}$$
$$F_j(p_f^k) = F_1(p_f^k) + \sum_{h=1}^{j-1}(\beta_{f,h} + \pi_h + \max_{1 \le i \le k} \frac{l_f^i}{r_f^i}),\ j > 1 \qquad (4.7)$$

where,

$F_j(p_f^k)$ =Tag of the kth packet of flow *f* at *j*th router.

$\hat{F}(a_{f,j}^k)$ = Smallest tag value in *R* at router *j* for flow *f*

The two challenges of this design are:

- Deciding which tags are re-usable
- Re-ordering of packets at network exit

The two mechanisms that will be added on CSVC to form CSGT are used to overcome these two challenges.

1. Re-Usability

It is difficult to define a packet as re-usable because of two factors:

- Reuse of tags must not violate the deadline guarantees provided to other flows. To meet this requirement, the tag assigned to a packet must differ by at least l_f / r_f from the tags assigned to all packets that were transmitted prior to this packet but have not reached the destination [35]:

$$\forall p_f^i \in U : \left| F_j - F_j(p_f^i) \right| \geq \frac{l_f}{r_f} \tag{4.8}$$

where *U* is the set of packets transmitted by the ingress router prior to packet p_f^m but have not reached the destination by time t.

The deadline guarantee provided to other flows can not be violated if the egress router sends an ACK to ingress router when it transmits a packet. Ingress node will take a service tag into consideration for re-use and add that tag to *R* only if it receives an acknowledgement for the packet carrying this tag. This method eliminates the first factor described above.

- CSGT must provide a deadline guarantee on the re-used tag which means [35]:

$$t \leq F_1 - \frac{l_f}{r_f} \tag{4.9}$$

The second factor is checked after supporting the first factor. When receiving a packet, ingress router scans through *R* and assigns the tag that meets (4.9).

It is obvious that the tag of a packet is re-used only if that packet reaches the egress router much prior to its deadline which is formalized as [35]:

$$d_{f,H}^m + D^{\min} \leq F_1(p_f^m) + \frac{l_f}{r_f}$$

where,

$d_{f,H}^m$ = Ingress to egress propagation latency

$D^{\min}$ = Minimum latency encountered by the acknowledgement packet

H= Hop count

Then a tag is re-used only if:

$$d_{f,H}^{m} \le F_H(p_f^m) - (\sum_{j=1}^{H-1}(\beta_{f,j} + \pi_j + \max_{1\le i\le m}\frac{l_f^i}{r_f}) + \frac{l_f^{\min}}{r_f} + D^{\min}) \qquad (4.10)$$

The egress router sends an ACK only if the above condition holds.

2. Re-ordering

When the service tags are re-used, the packets can reach the egress router out-of order. The in-order delivery can be important for some applications, so a sequencer is used in CSGT in order to buffer the out-of-order packets and re-order them at network exit (Figure 4-1).

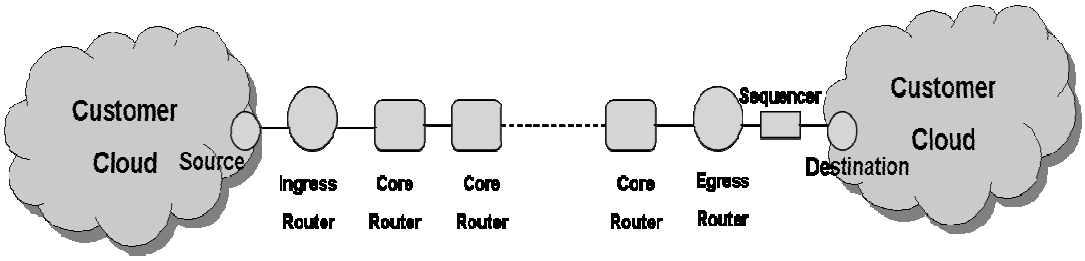

Figure 4-1 CSGT Network

Thus the number of packets in the buffer of the sequencer should be limited. The second mechanism of CSGT which is added on CSVC is used for this goal. This mechanism is called “Flow Control Algorithm” and it limits the maximum number

of deadlines in use. No packet is assigned a deadline larger than t + $P\ \frac{l_f}{r_f}$ where *P* is a configuration parameter. It is proven in [34] that packets of a flow will not be dropped at the sequencer due to the unavailability of buffers if *P* satisfies:

$$B \geq \left\{ \begin{array}{l} (N+1)(P-1)-(N)(N+1)\dfrac{k^{\min}}{2}, if\ \ T^{\min} \geq l_f / r_f \\ \dfrac{P(P-1)}{2k^{\min}}, if\ \ T^{\min} < l_f / r_f \end{array} \right\} \qquad (4.11)$$

where

B = The available sequencer buffer space.

$N = \left\lfloor \dfrac{P-2}{k^{\min}} \right\rfloor, k^{\min} = \dfrac{T^{\min}}{l_f / r_f}$ and T^{min} is a lower bound on the round-trip time.

When the maximum bandwidth a flow should get is determined as *r'*, a bound on *P* can de driven using [34]:

$$d^m_{f,H} \leq F_H(p^m_f) - (\sum_{j=1}^{H-1}(\beta_{f,j} + \pi_j + \max_{1 \leq i \leq m}\frac{l^i_f}{r_f}) + \frac{l^{\min}_f}{r_f} + D^{\min}) \qquad (4.12)$$

This value of P when substituted in equation of B, determines the minimum buffer requirement at the sequencer.

4.1.2.3. Performance Guarantees of CSGT

In [34], it is shown that the deadline guarantees of CSVC are preserved in CSGT. For the throughput guarantee provided by CSGT, the following theorem is stated and proved:

If the source of flow *f* transmits packets at least at its reserved rate, then the network guarantees a minimum throughput in any time interval (t_1,t_2) ($W_{f,H}(t_1,t_2)$), as follows [34]:

$$W_{f,H}(t_1,t_2) > r_f(t_2-t_1) - r_f\left((H+2)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j}\right) - r_f * D^{\max} \quad (4.13)$$

where $D^{\max}$ denotes the maximum acknowledgement time.

Further, the sequencer guarantees a minimum application throughput, $W_f^{app}(t_1,t_2)$, given by [34]:

$$W_f^{app}(t_1,t_2) > r_f(t_2-t_1) - r_f\left((H+1)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\beta_{f,j}\right) - r_f * D^{\max} - P * l_f \quad (4.14)$$

The network throughput provided by a core stateful network is:

$$W_f^{app}(t_1,t_2) > r_f(t_2-t_1) - r_f\left((H+1)\frac{l_f}{r_f} + \sum_{j=1}^{H-1}\pi_j + \sum_{j=1}^{H}\varepsilon_{f,j}\right)$$

where r_f is the reserved rate of flow *f* and $\varepsilon_{f,j}$ is the latency term in the packet scale rate guarantee of server *j*.

The bound on the network throughput derived for CSGT differs from that provided by a core stateful network, by a constant term [34]:

$$E_1 = r_f * \left[D^{\max} - \sum_{j=1}^{H}(\varepsilon_{f,j} - \beta_{f,j}) \right] + l_f$$

$\beta_{f,j}$ is $l^{\max}/C_j$ for a CSGT network derived from CSVC. Further, for most schedulers $\varepsilon_{f,j} \geq l^{\max}/C_j$ so $E_1 \leq r_f * D^{\max} + l_f$. The minimum non-zero throughput timescale of a CSGT network is therefore primarily governed by the maximum latency on the reverse path. The observations in [34] imply that by provisioning low-delay feedback channels, a CSGT network can provide non-zero throughput

guarantees at very short time-scales (100 -200 ms), and similar to those in core-stateful networks.

4.1.3. CSGF

After providing delay and throughput guarantees, the next and final step for CSGF design is providing fairness guarantees. The fair scheduling means allocating available bandwidth to flows proportional to their reserved rates. In this perspective, fairness differs from throughput and delay guarantees that are characterized by the corresponding flow's properties only. Fairness guarantee is a function of all flows that share the same bandwidth.

Classical fair network schedulers ([19, 33, 38, 39]) that are used in every node of core-stateful networks have some measures to define the fairness guarantees they provide. These measures are unfairness measure (*U*), error term (γ), and a constant *(I)* which are used in the following equations ([35], [40],[38]):

$$\left| \frac{W_{f,j}(t_1,t_2)}{r_f} - \frac{W_{m,j}(t_1,t_2)}{r_m} \right| \leq U_{j,\{f,m\}}$$

$$W_{f,j}(t_1,t_2) > r_f(t_2 - t_1) - r_f \gamma_{f,j}$$

$$\frac{W_{f,j}(t_1,t_2)}{r_f} > \frac{W_{m,j}(t_1,t_2)}{r_m} + I_{j,m,f}$$

where $W_{f,j}(t_1,t_2)$ is the throughput in time interval (t_1,t_2) for flow *f* in router *j*.

Derivation of CSGF is composed of two steps. Firstly, in order to provide fairness guarantees, the requirement to provide throughput guarantees is shown. In CSGF, CSGT is the underlying algorithm providing throughput guarantees. Second, two

mechanisms are added to CSGT in order to form a work-conserving core-stateless network that guarantees end-to-end throughput bounds within an additive constant of the one obtained by a network of core-stateful routers.

4.1.3.1. Throughput guarantee requirement

A work conserving server that provides fairness guarantees to flow *m*, also provides throughput guarantee of the form [41]:

$$
\begin{aligned}
&W_{m,j}(t_1,t_2) > r_m(t_2-t_1) - r_m * \gamma_{m,j} \\
&\frac{W_{f,j}(t_1,t_2)}{r_f} \le \frac{W_{m,j}(t_1,t_2)}{r_m} + U_{j,m,f} \\
&\textit{where} \\
&\gamma_{m,j} = \frac{l^{\max}}{C_j} + \frac{\sum_{f \in F} r_f U_{j,\{f,m\}}}{C_j}
\end{aligned}
\tag{4.15}
$$

In other words, the theorem states that a network can not provide fairness guarantee if it doesn't provide throughput guarantee. Thus a work conserving core-stateless network with throughput guarantees is a requirement for CSGF design.

4.1.3.2. Properties of CSGF

One of the design principles of CSGF is that the per-link proportionate bandwidth allocation is taken to be important and meaningful only for flows that share the entire end-to-end paths. Depending on this argument, CSGF aims to provide strong consistent end-to-end proportionate bandwidth allocation for the flows that share entirely same end-to-end path [41].

There are three main design steps for CSGF to provide fairness for the flows sharing end-to-end paths in a network with throughput guarantees:

1. Treating the aggregate flow between a pair of edge nodes as a single flow and providing throughput guarantees to it.
2. Employing a fair scheduling algorithm at the ingress node
3. Ensuring that the network preserves the order in which packets are transmitted.

The second item above guarantees the fairness between flows that will be treated as one aggregate flow according to first item. The first item will provide throughput guarantees to this aggregate flow and by the third item, they will exit the network in the same order as they enter the network. As a result, individual flows will be served with proportionate allocation guarantees. The third mechanism described above is already supported by CSGT network. Therefore the other two mechanisms need to be added on CSGT to derive CSGF.

An ingress node in a CSGF network is responsible from the deadline assignment and packet selection. For the deadline assignment, an aggregate flow packet is assigned a tag as in CSGT, but reserved rate is used as R= $\sum_{f \in F} r_f$. The next flow to select a packet from is chosen by using a fair scheduler. Core and egress nodes in CSGF act same as in CSGT. Aggregate is split into micro-flows at the egress. A sequencer reorders aggregate packets before they are split.

4.1.3.3. Performance Guarantees of CSGF

Since CSGF claims to be a fair network architecture, there must be an unfairness measure guaranteed to flows by the architecture as in core-stateful architectures. Therefore $\frac{W_{f,H}(t_1,t_2)}{r_f} - \frac{W_{m,H}(t_1,t_2)}{r_m}$ needs to be computed for CSGF. There are two

types of fairness defined in CSGF [41]. The first one is fairness in application throughput and the second one is fairness in network throughput. These are fairness measures after and before the scheduler, respectively [41].

In a CSGF network deployed by implementing the three factors above, the packets exit the network in the same order they enter the network. The ordering of packets is managed by the stateful fair scheduler at the ingress of the network and since the order is preserved during their travel in the network, the unfairness measure for CSGF in application throughput is equal to unfairness measure of the scheduler (e.g. SFQ) at the ingress node. Thus [41]:

$$\frac{W_f^{app}(t_1,t_2)}{r_f} \leq \frac{W_m^{app}(t_1,t_2)}{r_m} + U_{f,m}^{SFQ} \tag{4.16}$$

The fairness in network throughput is not so easy to define since at the measuring point, packets are not in the same order as they entered the network. In [25], a detailed evaluation for this fairness measure is given with computational proofs.

In the computational evaluation solutions given in [25], the unfairness measures of CSGF and a core stateful network are compared. All considered flows in the computations for fairness share the same entire end-to-end path. Results show that fairness of CSGF in application throughput is even better than core-stateful networks since packets depart the sequencer exactly in the same order as transmitted. However fairness in network throughput is weaker than core-stateful networks.

Computations also show that for flows with small reserved rates, the per-flow application throughput guarantee offered by CSGF is better than CSGT. CSGF is capable of providing application throughput guarantees at small time-scales of hundred milliseconds.

CHAPTER 5

IMPLEMENTATION

This chapter includes the implementations of the Virtual Clock scheduling algorithm, its Core-Stateless version CSVC, CSGT and finally CSGF. OPNET version 11.5 simulation tool has been used in order to implement the algorithms.

The features of the architectures are defined as properties of routers. Therefore, Virtual Clock, CSVC, CSGT and CSGF approaches are implemented in the process, node and network layers of OPNET as new router nodes. These routers are created using state transition diagram models, coded in embedded C or C++. The models have been added to the OPNET environment so that a new user can use them in the future for their simulations.

5.1. The Simulation Environment

OPNET is a comprehensive network simulation and management software, developed by OPNET Technologies, Inc. founded in 1986 [42]. Since then, twelve versions of the software have been released. OPNET provides an environment that supports modeling of communication networks and distributed systems. OPNET environment contains tools for different phases of a study, including design, simulation, data collection and data analysis.

There are three layers in the OPNET model hierarchy, which are called as the process, node and network layers, each having an associated editor with it [42].

5.1.1.Project Editor

The project editor is used to construct and edit the topology of communication network models. The interconnection and position of network nodes are adjustable in this editor. It also provides operations to support the simulation and analysis of these network models. This editor is the highest modeling level in OPNET in the sense that it uses the objects that are defined in the other modeling editors [42].

5.1.2.Node Editor

The node editor is used to define the structure and behavior of nodes used in the network domain (such as clients, servers, switches, routers, bridges and firewalls). Each network node is made up of several modules. Each of these modules defines one aspect of node behavior such as data generation, data storage, data forwarding, etc. These modules are connected together via packet streams or statistical wires. In addition to the node structure, this editor defines the interface of a node model, which determines what aspects of the node model are visible and can be defined by the user [42].

5.1.3.Process Editor

The process editor is used to specify the process models, which define the functionality of the modules used in the node models. In addition to the behavior of a process, this editor defines the model's interfaces, which determine what characteristics of the process model are visible and can be adjusted by users [42].

Process models are defined by finite state machines, which are composed of two main components: states and transitions. A process is always in exactly one state at a time. A process can move between states upon receiving some interrupts. The interrupts fulfill the conditions that make the process move from one state to another. The interrupts may be originated either from the process itself or from another process, called a parent process to the invoked process (child process) [43].

The operation of each state is defined in a distinct block written in embedded C or C++ code. These blocks are called executives. The executives of a state are split into two sections, called enter and exit executives. The enter executives are executed when a process enters a state and the exit executive is performed while the process is leaving a state. States are divided into two categories: forced states and unforced states that differ in execution timing. In unforced states, there is a pause between enter and exit executives. Once an enter executive is finished, the process returns the control to the process that has invoked it hence being suspended until it is invoked again. When it is invoked for a second time, the exit executive of the blocked state is then executed. In the forced states, the exit executive is executed by a process immediately after the completion of the enter executive. For this reason the exit executives of forced states are usually left blank [42].

We first aimed to implement the algorithms in OPNET's ip_output_iface process model, which is a child process to the IP layer process model of all IP routers. The *ip_output_iface* process model is in charge of assigning queues to data flows entering the router and scheduling packets based on one of the scheduling mechanisms implemented in OPNET. After working on one implementation we observed that adding the features we created requires longer time and effort than creating them and the advantage of this kind of implementation would be only its industrial use. Our work has an academic view and is a proof of concept study.

Therefore we implemented our routers as stand-alone nodes having only the functions we defined.

5.2.Implementations

Core-Stateless QoS architectures aim to use core-stateless queuing schemes in order to approximate the functionality of a network in which all nodes implement a stateful scheduler. In our case, VC is the selected stateful scheduling scheme to approximate. There is only one kind of stateful router in Virtual Clock. Since the ingress and egress routers of CSGR, CSGT and CSGF also keep states of the flows, VC router also forms the basis for these routers in the implementations. CSVC, which is the core-stateless version of VC, is selected as the CSGR network. All kinds of routers in CSVC, CSGT and CSGF are implemented in OPNET simulation environment and the correct operations of the implementations are demonstrated by simulations. CBR (Constant Bit Rate) traffic is used in all simulations throughout this chapter.

5.2.1.Implementation of Virtual Clock Router

VC Packet Format

Packet formats in OPNET [42] define the internal structure of packets as a set of fields. For each field, the packet format specifies a unique name, a data type, a default value, a size in bits, an encoding style, a conversion method and optional comments.

A new packet format is created in “OPNET Packet Editor” in order to use in VC Router simulations (Figure 5-1). Since VC Routers are stateful and classify packets according to their flows, a new field called “flowpk” is used to define the flow that the packet belongs to. Alternatively, the flow of the packet can be determined by the source destination address pair and/or TCP/UDP port or ToS ([12]) field. In IPv6 packets, there is a 20-bit field called “Flow Label”. This field is used to insert a label value that is common to the packets belonging to the same stream, session or flow. Throughout this study, we also used a 20-bit field for flow label information.

Another packet field called “VC” is used to insert the “Virtual Clock” value. In [15], a mechanism for state encoding is described by which rate and time values are encoded in 16-bits. This encoding provides a mechanism to represent large numbers ($\sim 2^{15}$) with a few bits. The state encoding is not within the scope of this thesis but we used a 16-bit VC field as in [15]. Consequently, most of the new fields in the packet formats will be 16-bit long throughout this thesis.

Figure 5-1 VC Packet Format

State Transition Diagram

The state transition diagram shown in Figure 5-2 is created in order to simulate the behavior of a VC router. It consists of five states: “init, arrival, scheduler, idle, send”. This state transition diagram also forms the basis of all other routers in this work and hence it is used in other router implementations. The init, arrival, scheduler and send states are forced states and the idle state is an unforced state. The state transition diagram functions as follows:

When a packet arrives at a node, the process model is invoked. At the invocation time, initializations for the variables and structures are done. The process enters the “idle” state immediately after the initializations. The process remains at idle until it receives an interrupt. When the interrupt originates from the arrival of a packet, the process enters the “arrival” state.

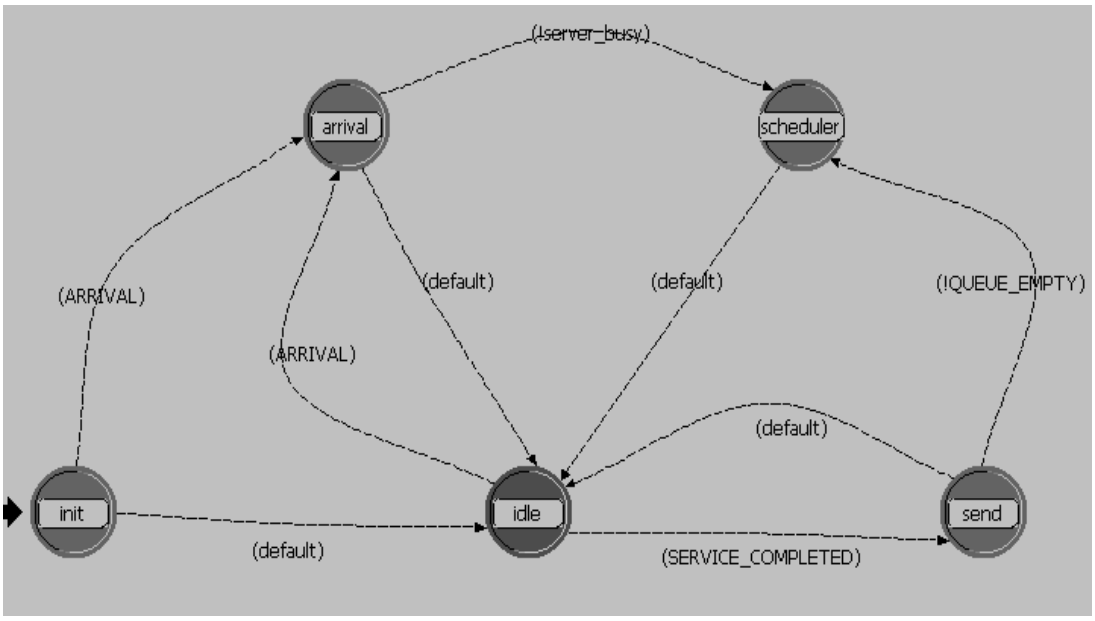

Figure 5-2 State Transition Diagram of Virtual Clock Router

In the “arrival” state, the packet is assigned to one of the existing queues according to the flow it belongs to. After the “arrival” state if the server is not busy, the process goes to the “scheduler” state, which is in charge of choosing the queue from which the next packet will be transmitted. When the queue is selected according to the VC algorithm, the process schedules itself an interrupt for sending the packet. This time period resembles the time required by the router in order to send the packet. In the “send” state, the selected packet is transmitted to the related outgoing link. The details of the “arrival” and “scheduler” states are further described step by step below.

5.2.1.1. Arrival State

1. The new packet is acquired from the stream generating the interrupt.
2. The queue to which the packet will be inserted is determined according to the “flowpk” field.
3. The packet is checked to see whether it is the first packet of its flow or not since it affects the VC value calculation. A “flow counter” is held in the system for this control.
4. After calculating the packet’s VC, it is saved as the last VC value (VC^{k-1}) for the related queue. The flow ID and VC^{k-1} are the state values that are stored in the router for that flow, which makes the VC algorithm stateful.
5. The packet is inserted into the related queue.

5.2.1.2. Scheduler State

1. The queues are investigated to find the nonempty ones.
2. The nonempty queues are traced in order to find the packet with the minimum VC. The VC values of all the packets at the “Head” positions of the queues are compared to find the packet to dequeue.
3. When the packet with the minimum VC is found, the queue associated with that packet is selected as the “sender queue”.
4. The packet at the “Head” of the sender queue is removed and a self interrupt is scheduled according to the size of the packet. The time required to send the packet depends on the size of the packet and the output capacity of the router at that time.
5. The server_busy flag is set.

5.2.2.Implementation of Core-Stateless Virtual Clock (CSVC) Routers

The edge and core router models of CSVC are implemented in OPNET as new nodes. Since there are new fields on the packets specific to CSVC algorithm, a new packet format is also defined.

CSVC Packet Format

Since the core routers are stateless, they need the reserved rate of the packet's flow in order to calculate the VCore value that will be assigned to the packet in service. CSVC Packet Format (Figure 5-3) has one new field called "ratepk" carrying the reserved rate information.

Figure 5-3 CSVC Packet Format

5.2.2.1. CSVC Core Router

The idea in constructing router state transition diagram is kept as it is in VC routers. The functions defined in the states create the difference between CSVC Core Routers and VC Routers. The similarity in state transition diagram, when combined with the difference in the C and/or C++ code and functions embedded in the states, gives a better understanding of the difference of core-stateless routers.

The initialization in CSVC core router is so simple that only "packet_counter" and "server_busy" state variables are initialized to zero. The VC router on the other side

keeps states and it has many state variables and structures that should be initialized. This is one of the simpler and distinct parts of the CSVC routers. State Variables are used in OPNET to represent the information accumulated and retained by a process. This name (State Variables) is due to the fact that these variables, together with the current position of a process within its state transition diagram, represent the complete state of a process at any time. Note that processes may generate or have access to much information over time that does not become encoded into their state; therefore state refers only to the information that the process itself decides to retain by recording it into state variables.

In the “arrival” state, there is no packet classification or queue assignment functions. This is the main property of the “arrival” state in previously described VC routers. In CSVC “arrival” state however, packet’s VCore value is calculated using its size and rate information written on the packet. Since there is no flow recognition in CSVC, it does not matter if the packet is the first packet of a flow or not. Then the packet is inserted into the only queue according to its VCore value. Since this is now a single queue system, there is no need to find a sender queue.

5.2.2.2. CSVC Edge Router

The process model of the CSVC Edge Router is similar to VC Router, so in the following paragraphs the differences between a CSVC and a VC router will be emphasized only.

Since the core routers do not keep any states, packets carry state information in CSVC. Therefore the information should be inserted into the packets on the edges of the network. CSVC edge router uses the “arrival” state for this purpose. In edge router, there exists a multiple queue for each flow. In the “arrival” state, the edge routers in CSVC insert the *rate* and *VCore* values to the packets when the packet

goes to its associated queue. In this thesis VC field in CSVC packet is used in this thesis for carrying the *VCore* value.

5.2.3.Implementation of Core-Stateless Guaranteed Throughput (CSGT) Routers

The design methodology of CSGT depends on two principles.

- In order to provide throughput additional to delay guarantees, CSGT is built upon a network providing delay guarantees (CSGR).
- Tag re-use and source rate control mechanisms, when integrated with CSGR architecture, lead to the design of CSGT.

Since CSVC provides delay guarantees, the new properties of the routers in CSGT are implemented upon the routers of CSVC. There is also a “Sequencer” in CSGT, which is used for satisfying the in-order delivery requirements of some applications.

CSGT Packet Formats:

CSGT Packet Format (Figure 5-4) depends on CSVC Packet Format but it has two new fields. One of the new fields, called “Type”, is used by routers to recognize the packets.

The “VC1” field is the other new field defined for CSGT. This field is created to be used in “Tag Re-use” mechanism. When a packet exits the network sufficiently prior to its deadline, the egress router sends an acknowledge packet to the ingress router indicating that the *VCore* value of this packet can be re-used. The first tag value is kept in a separate field called VC1 on the CSVC packet and the VC field changes at each router to be used for scheduling purposes. When the packet is

decided to be acknowledged at the egress router, the value in “VC1” is copied to the VC field in the acknowledge packet and sent to the ingress router.

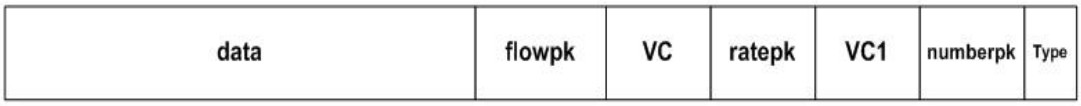

Figure 5-4 CSGT Packet Format

There is no need to include the data field in CSGT Acknowledge packet (ACK) (Figure 5-5). However additional information can also be sent with this packet. The information in acknowledge packets can also be piggybacked onto the data packets of the flows going to the opposite direction to save bandwidth.

ACK packets are recognized by the “Type” field in ingress routers. “VC” and “flowpk” fields contain the information in “*VC1*” and “flowpk” fields of the packet that is acknowledged.

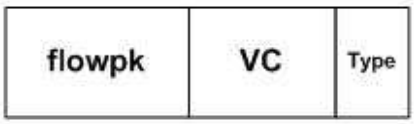

Figure 5-5 CSGT Acknowledge Packet Format

5.2.3.1. CSGT Core Router

The transition from CSVC to CSGT is provided mainly by two functions that are implemented on edge routers. The core routers of CSGT do not function differently than the core routers in CSVC except the recognition of the new packet types defined for CSGT.

When a packet comes to the CSGT Core Router, if it is an acknowledge packet, it is sent towards the related ingress router without any operation on it. If it is not an acknowledge packet, it is treated just like it is treated in CSVC Core Routers.

5.2.3.2. CSGT Ingress Router

CSGT Ingress Router functions similar to the CSVC edge router, so it won't be discussed here in detail. Only the differences compared to CSVC edge router will be emphasized in the following paragraphs.

In the "arrival" state, the router first checks the packet type. If the packet is an acknowledgement packet, this means that the *VCore* value carried by the packet is a candidate value to be re-used. Then the flow of the packet is read from the packet and *VCore* value is put into the array that keeps the re-usable tag values for that flow. It will be re-used if it satisfies equation 4.9 when a new packet of the same flow comes. After storing the tag value, the packet is destroyed.

If the packet is not an acknowledgement packet, it is assigned to one of the existing queues according to its flow ID. The assignment of *VCore* value and the insertion of the packet to the specified queue are not same as CSVC:

- If the array that keeps the re-usable tag values for the flow is not empty, a value satisfying the condition in equation 4.9 is searched. The values that violate the condition are deleted since they will also violate the same condition for future packets of the same flow. If no value satisfying the condition in equation 4.9 can be found, *VCore* value is calculated as in CSVC.
- *VCore* value decided to be used is put into both "VC" and "VC1" fields of the packet.

- The packet is inserted to its corresponding queue. The packets in the queue are positioned according to their “VC” values.

5.2.3.3. CSGT Egress Router

Only the differences compared to CSGT ingress router implementation are given in this section.

CSGT egress routers do not need to check the packet to learn its type since they do not receive acknowledge packets. Therefore there is no array for holding re-usable tags and the *VCore* value of the packet is calculated as in CSVC edge routers.

Before a packet is sent to the related output interface, its end-to-end delay is calculated. Then if the packet departs the network much before its *VCore* value, i.e. if the end-to-end delay of the packet satisfies equation 4.12, it is acknowledged by the generation of a new “CSGT Acknowledge Packet”.

5.2.3.4. Sequencer

As described in 4.1.2.2. , a sequencer is used in CSGT in order to buffer the out-of-order packets and re-order them at the network exit. The sequencer can be implemented as an internal part of the Egress Router or a separate individual node. The sequencer is implemented as a separate node in this study because applications may not require in-order delivery and so the exclusion of the sequencer.

For the applications that require in-order delivery of packets, a new field called “numberpk” that includes the packet number is added to the packet format of CSGT. The sequencer is positioned after the Egress Router in the network. Queue process

model of OPNET is appropriate to implement the sequencer. The sequencer functions as follows:

1. The incoming packet is acquired and its flow is determined.
2. “numberpk” field of the CSGT packet is gathered. If numberpk=1, then this packet will be assigned to a queue and it will immediately be scheduled to be sent. The number of the last packet served by the queue is kept for future use.
3. If “numberpk” is not equal to one and it is one greater than the number of the last packet served by the sequencer’s related queue, then the packet sequence is in order. The packet will be inserted into the related queue.
4. The sequencer has an array for each queue. The arrays hold payload, packet number and *VCore* values of the packets. When an old packet needs to be extracted from the set and inserted into the queue, these values are extracted; a new packet is created with these values and put into the queue.
5. The set that includes the out-of-order packets for the flow of the packet will be searched to find the consecutive packet. If it is found, it will be inserted to the queue. This search-find-insert series will be repeated recursively.
6. If “numberpk” of the incoming packet is not one greater than the number of the last packet served by the sequencer’s related queue, then the packet will be sent to the set that keeps such out-of-order packets for that flow.

5.2.4.Implementation of Core-Stateless Guaranteed Fair (CSGF) Routers

The design methodology of CSGF depends on two principles:

- For a network to provide fairness guarantees, it must also provide throughput guarantees.
- The two mechanisms described in 4.1.3.2. , when integrated with an architecture that provides throughput guarantees, lead to the design of CSGF.

Since CSGT provides throughput guarantees already, CSGF routers (Ingress, Egress and Core) are built upon CSGT routers. The “Sequencer” in CSGT is also used in CSGF. As described in Section 4.1.3, there are three design steps when adding fairness properties to CSGT and none of these are directly related to the core routers. Thus the core and egress router and the sequencer structures in CSGT are kept unchanged in CSGF.

There are no new required fields on the CSGT packet formats specific to CSGF algorithm, so the packet formats in CSGT are used in CSGF.

5.2.4.1. CSGF Ingress Router

The features needed to be implemented in a CSGF router are:

1. Treating the aggregate traffic between two edge routers as a single flow and providing throughput guarantees to this aggregate flow.
2. Employing a fair scheduling algorithm at the ingress node to the flows (micro-flows) that make up the aggregate flow (macro-flow).

These features affect only the structure of the ingress node. Therefore only the ingress router is modified in the implementation of CSGF. The internal structure of the CSGF Ingress Router is shown in Figure 5-6. Macro-flow F is formed by aggregating micro-flows f_1, f_2 and f_3. The same applies for macro-flow G and micro-flows g_1, g_2 and g_3. When these micro-flows enter the node, a fair scheduling algorithm is applied to these flows and they form the aggregate macro-flow. Micro-flows that share the same entire end-to-end path are handled by the same fair scheduling algorithm. Following this fair scheduling, aggregate macro-flows are processed by a second scheduling mechanism at the CSGT Ingress Node.

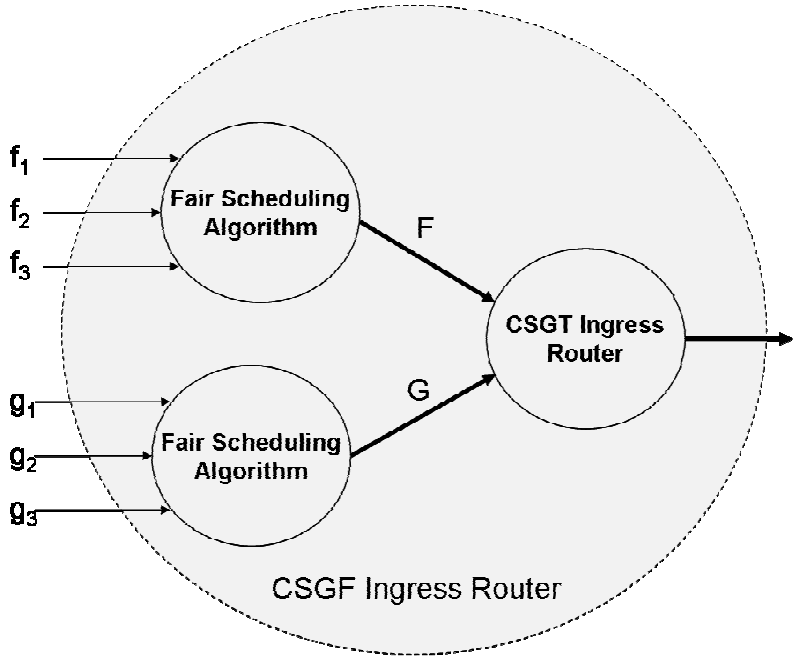

Figure 5-6 CSGF Ingress Router

The fair scheduling algorithm handling the micro-flows should allocate a fair share of the aggregate throughput to individual flows within the aggregate. Any fair scheduler that guarantees proportionate allocation can be used for this purpose. Weighted Fair Queuing is used to guarantee proportionate allocation in our implementation. The implemented fair scheduler functions as follows for an incoming packet:

1. The packet is assigned to the queue of its flow in the fair scheduler.
2. A virtual finish time is calculated for the packet by the fair scheduling algorithm according to its arrival time, flow ID and length. This value is inserted to the "VC" field of the packet.
3. Virtual finish times of packets at the head positions of all queues are compared and the packet with the minimum virtual finish time is selected.

4. The selected packet is scheduled to be sent to the part of the router that functions as a CSGT ingress router scheduler. Then a new temporary header field representing the corresponding macro-flow is inserted in the packet. Each macro-flow in CSGF ingress router takes a unique Flow ID. This field, named macrofl, functions as the “flowpk” field when the packet arrives at the internal CSGT Ingress Router and it is stripped off when sending the packet to the network core. All packets served by the same fair scheduling algorithm take the same rate value, which is the sum of the rates of micro-flows. This new value is inserted to the “ratepk” field of the packets.
5. The selected packet is sent to the CSGT ingress router scheduler.

One fair scheduling algorithm runs for one group of micro-flows that share the entire end-to-end path. Therefore the number of fair scheduling functions applied at the ingress router is equal to the number of end-to-end paths used by the flows in the network.

5.3. Validation of Implementations

In this chapter, the correctness of the implemented models is observed via simulations.

New source and sink node models are created in OPNET. “Source Model” is used to generate any format and any size packets at any rate and the “Sink Model” is used to sink packets and to keep statistics in the simulations.

Source Model

The source model has the following attributes: “Flow Rate”, “Packet Interarrival Time”, “Packet Size”, “Packet Format”, “Start Time”, “Stop Time”, “flow” (Figure 5-7).

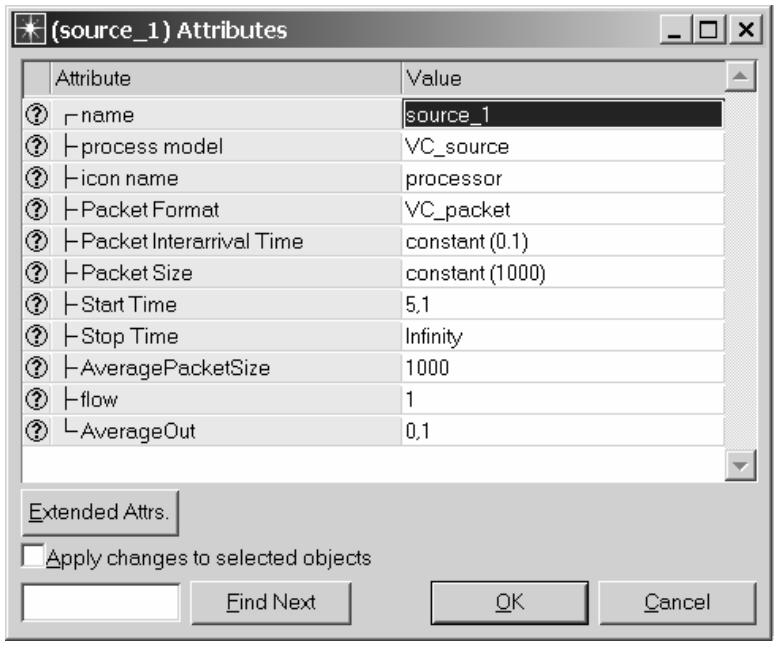

Figure 5-7 Attributes menu of the Source Model

The “Flow Rate” attribute defines the *reserved rate* of the flow that the source will generate. When the packets of a flow are formed at this rate, then it is called a conforming flow. “Packet Interarrival Time” defines the time intervals between the packets formed by that source. This value defines the real rate of the flow. A user can define a conforming or non-conforming flow by using this attribute of the source.

“Packet Size” is the length of the packet that will be generated. A PDF or a constant value can be used for this attribute. “Packet Format” is the format of the packet to

be generated. A formatted packet contains fields defined prior to a simulation using the Packet Format Editor. An unformatted packet's fields are specified dynamically during a simulation. "Flow" is the flow ID of the packet that the source will generate. "Start Time" and "Stop Time" attributes give the starting and stopping times of the packet generation process..

The state transition diagram, given in Figure 5-8, is created for the Source Model. There are three states: init, schedule and stop. In our study, different features are needed for sources in different simulations but the state transition diagram remains the same with minor differences in the specific properties of the states.

At the "init" state, the attribute values of the source are read. Generation of both formatted and unformatted packets are possible. If a valid value (smaller than the stop time) is entered in the "Start Time" attribute, then the process goes into the "schedule" state at the start time of the packet generation.

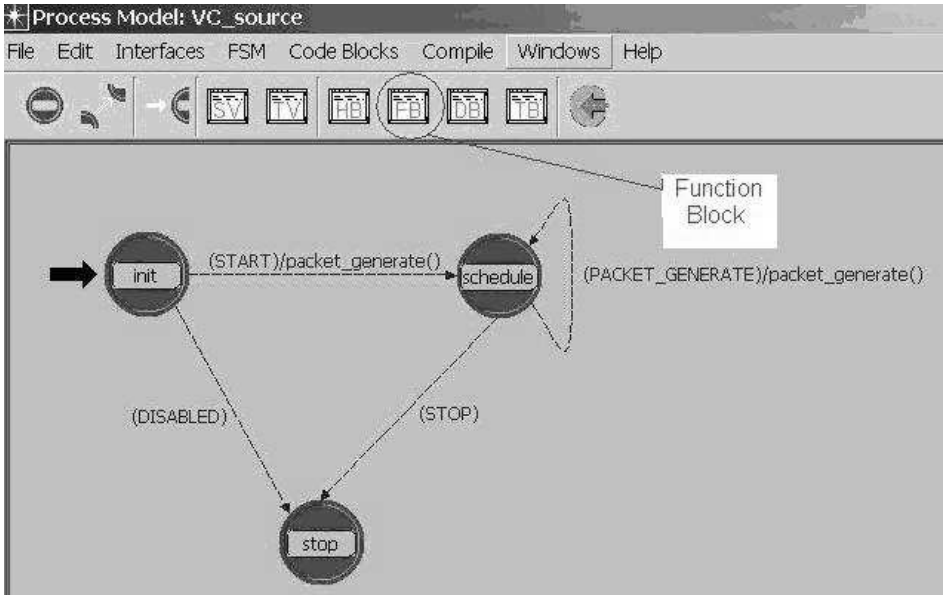

Figure 5-8 Process Model of Virtual Clock Source

During the transition to the ”schedule” state, packet generation starts. “packet_generate()” function is responsible for creating packets based on the packet generation specifications of the source model. The packet generation function is created in the “Function Block” (Figure 5-8) of the source model. The function block in OPNET process models contains C or C++ language functions that are associated with the process and that can be called by any of the statements in the process.

The arrival of the next packet is scheduled at the "schedule" state. According to the code of the interrupt, state transition conditions are evaluated and the packet generation continues until the stop time.

Sink Model

A simple sink node model is used in our simulations as packet sink and also as a station to take statistical data such as end-to-end delay. The process model of the Sink is shown in Figure 5-9. In “INIT” state, few variables used in the model are initialized and in the “DISCARD” state, the packet is obtained from the incoming stream. Then the required metrics are updated, statistical data is collected and the received packet is destroyed.

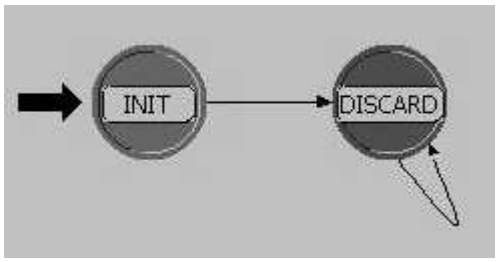

Figure 5-9 Process model of VC Sink

5.3.1.Validation of VC Router Implementation

The topology for validation is shown in Figure 5-10, where there are two sources creating traffic with constant packet generation rates. The packets' arrival and departure times are observed and VC values of the packets are examined to see how the VC Router selects and forwards packets according to these values.

The links between the sources and the VC Router are high-speed and delay on these links is negligible. The sources generate 1000 bit packets. Source_1 and Source_2 generate 10 and 5 packets in one second respectively. However, the VC Router can transmit only 8 packets per second to the sink. The reserved rate of the first flow is also two times the reserved rate of the second flow. Source_1 and Source_2 start generating packets at t=0 and at t=0.1 respectively.

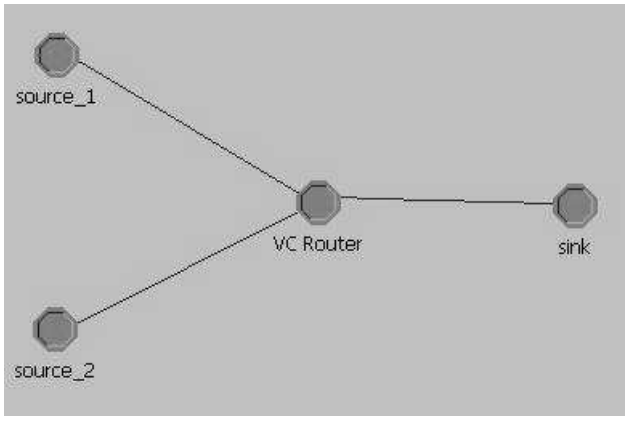

Figure 5-10 VC Router Simulation Topology

Table 5-1 presents the simulation results obtained from OPNET, which demonstrate the correct functioning of the router. Incoming packets of different flows are assigned to different queues with correct VC values that increase according to the rate of the packets as expected. The expected behavior of VC Router, i.e., sending two packets from flow 1 for each packet sent from flow 2, is observed in Table 5-1.

Table 5-1 Simulation Results for VC Router Implementation

Creation Time (sec)	VC (sec)	Flow Number	Sending Time (sec)
0	0.2	2	0.125
0.1	0.2	1	0.25
0.2	0,3	1	0.375
0.3	0.4	1	0.5
0.2	0.4	2	0.625
0.4	0.5	1	0.75
0.5	0.6	1	0.875
0.4	0.6	2	1
0.6	0.7	1	1.125
0.7	0.8	1	1.25
0.6	0.8	2	1.375
0.8	0.9	1	1.5
0.9	1	1	1.625
0.8	1	2	1.75
1	1.1	1	1.875
1.1	1.2	1	2
1	1.2	2	2.125
1.2	1.3	1	2.25
1.3	1.4	1	2.375
1.2	1.4	2	2.5
1.4	1.5	1	2.625
1.5	1.6	1	2.75
1.4	1.6	2	2.875
1.6	1.7	1	3
1.7	1.8	1	3.125
1.6	1.8	2	3.25

5.3.2. Validation of CSVC Router Implementations

5.3.2.1. Validation of CSVC Edge Routers

CSVC Edge Router is core-stateful and it works similar to VC Router. The topology of the introduced scenario for validation is same as the one used in validation of the VC Router simulation. The *VCore* assignment made by CSVC Edge Router is given in Equation 4.5 ($VCore_{f,1}^{k} = VC_{f,1}^{k}$). According to this definition, a VC Router and a CSVC Edge Router assign the same tag values to the packets. Therefore a CSVC Edge Router should behave the same as a VC Router in the same conditions. Additionally, CSVC Edge Router has to form the fields that are required by the core routers into the packet.

All parameters used in the sources, the router and the sink have the same values as they have in the above validation scenario of VC Router implementation. In CSVC edge router scenario, the packets' arrival and departure times are observed and *VCore* values of the packets are observed to see how the CSVC algorithm assigns it and how the method selects and forwards the packets according to these values. Results in Table 5-2 demonstrate that the router functions correctly.

Table 5-2 Simulation Results for CSVC Edge Router Implementation

Flow Number	Arrival Time (sec)	VCore (sec)	Rate Field (bps)	Sending Time (sec)
2	0.0	0.2	5000	0.067
1	0.1	0.2	10000	0.1670
1	0.2	0.3	10000	0.267
2	0.2	0.4	5000	0.334
1	0.3	0.4	10000	0.401
1	0.4	0.5	10000	0.468
2	0.4	0.6	5000	0.535

Table 5-2-Cont.-Simulation Results for CSVC Edge Router Implementation

1	0.5	0.6	10000	0.602
1	0.6	0.7	10000	0.669
2	0.6	0.8	5000	0.736
1	0.7	0.8	10000	0.803
1	0.8	0.9	10000	0.87
2	0.8	1	5000	0.937
1	0.9	1	10000	1.004
1	1	1.1	10000	1.071
2	1	1.2	5000	1.138
1	1.1	1.2	10000	1.205
1	1.2	1.3	10000	1.272
2	1.2	1.4	5000	1.339

5.3.2.2. Validation of CSVC Core Router Implementation

The topology of the CSVC Core Router simulation is similar to the one that is used for the CSVC Edge Router simulation. CSVC Core Router is a core-stateless router so it has only one queue. It serves the packets according to the *VCore* values they carry.

In the scenario, sources resemble two links inside the network core coming from different ingress routers. The links between the sources and CSVC Router is high-speed and delay on this link is negligible. However the speed of the link between the CSVC Router and the Sink is 1 packet/sec. The sources generate 1000 bit packets. Source_1 and Source_2 generate CSVC packets at constant rates, 5 and 0.5 packets in one second respectively. *VCore* values coming with the packets from second source are much smaller than the *VCore* values coming with the packets

from the first source. However, the confirming rates of the flows are the same. Source_1 and Source_2 start generating packets at t=2.0 and t=2.1 respectively.

Table 5-3 presents the simulation results obtained from OPNET. Incoming packets from Source_2 are dequeued earlier because of their smaller *VCore* values. The table also illustrates the values in "VC" fields of the packets. Packets are sent one by one from the flows as expected, since both of them have the same reserved rate. The aggressiveness of the first source is suppressed by the algorithm without keeping any state information at the router.

Table 5-3 Simulation Results for CSVC Core Router Implementation

Flow (Number)	Arrival Time (sec)	VCore (sec)	Queue Size (packet)	Sending Time (sec)
1	2.0	2.2004	1	3
2	2.1	0.3004	1	4
1	2.2	2.4004	2	5
1	2.4	2.6004	3	7
1	2.6	2.8004	4	9
1	2.8	3.0004	5	-
1	3	3.2004	5	-
1	3.2	3.4004	6	-
1	3.4	3.6004	7	-
1	3.6	3.8004	8	-
1	3.8	4.0004	9	-
1	4.0	4.2004	9	-
2	4.1	2.3004	10	6
1	4.2	4.4004	11	-
1	4.4	4.6004	12	-
1	4.6	4.8004	13	-
1	4.8	5.0004	14	-

Table 5-3 Cont.-Simulation Results for CSVC Core Router Implementation

1	5	5.2004	14	-
1	5.2	5.4004	15	-
1	5.4	5.6004	16	-
1	5.6	5.8004	17	-
1	5.8	6.0004	18	-
1	6.0	6.2004	18	-
2	6.1	4.3004	19	8

5.3.3. Validation of CSGT Router Implementations

5.3.3.1. Validation of CSGT Core Router Implementation

The functionality of the CSGT Core Router is the same as CSVC Core Router, so the performances of both router types in the same conditions should be similar. The main features different than CSVC Core Router and specific to CSGT Core Router are:

- CSGT Core Router has to recognize the type of the packet (normal or ACK) and insert the required information to the packet according to its type.
- The direction of the traffic on a CSGT Core Router is both towards the ingress routers and egress routers.

These features are observed in the following simulation scenario. In this scenario, the packets' arrival and departure times are observed and *VCore* values of the packets are examined to see how the CSGT Core Router assigns these values and

how it selects and forwards the packets according to these values. The results show that our CSGT Core Routers treat acknowledgement packets correctly.

In the topology in Figure 5-11, Source_1 generates 1000 bit packets of flow 1 with a rate of 2 packets/sec. The second source of flow 2, named as ACK-source, artificially generates only acknowledge packets with a rate of one packet per second. The links between the sources and CSGT Core Router is high-speed and delays on these links are negligible. The links between the router and the sinks are also fast links.

ACK_source starts generating packets at 1.25 seconds and Source_1 starts generating packets at 2 seconds. Table 5-4 presents the simulation results obtained from OPNET to show the validity of the implementation.

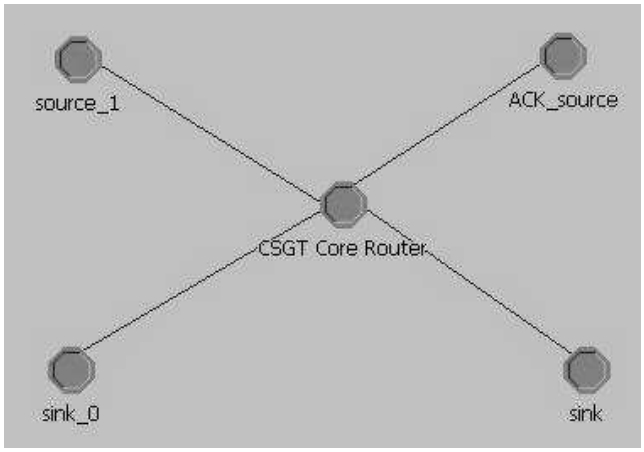

Figure 5-11 CSGT Core Router Simulation Topology

The table illustrates that VC values of the normal traffic packets are assigned according to the corresponding flows as expected. ACK packets are generated with *VCore* values that differ by an arbitrary 0.4 between two successive packets. It is observed that the router recognizes the ACK packets and VC values carried by ACK packets do not change when they pass the CSGT Core Router.

Table 5-4 Simulation Results for CSGT Core Router Implementation

Packet Type	Flow No	VCore (sec)	Arrival Time (sec)	Sending Time (sec)	Sink
ACK	2	0.100	1.25	1.260	Sink_0
Data	1	2.501	2	2.01	Sink
ACK	2	0.500	2.25	2.260	Sink_0
Data	1	3.001	2.5	2.51	Sink
Data	1	3.501	3	3.01	Sink
ACK	2	0.900	3.25	3.260	Sink_0
Data	1	4.001	3.5	3.51	Sink
Data	1	4.501	4	4.01	Sink
ACK	2	1.300	4.25	4.260	Sink_0
Data	1	5.001	4.5	4.51	Sink
Data	1	5.501	5	5.01	Sink
ACK	2	1.700	5.25	5.260	Sink_0
Data	1	6.001	5.5	5.51	Sink
Data	1	6.501	6	6.01	Sink
ACK	2	2.100	6.25	6.260	Sink_0
Data	1	7.001	6.5	6.51	Sink
Data	1	7.501	7	7.01	Sink
ACK	2	2.500	7.25	7.260	Sink_0
Data	1	8.001	7.5	7.51	Sink
Data	1	8.501	8	8.01	Sink
ACK	2	2.900	8.25	8.260	Sink_0
Data	1	9.001	8.5	8.51	Sink
Data	1	9.501	9	9.01	Sink
ACK	2	3.300	9.25	9.260	Sink_0
Data	1	10.001	9.5	9.51	Sink
Data	1	10.501	10	10.01	Sink

Table 5-4-Cont.- Simulation Results for CSGT Core Router Implementation

ACK	2	3.700	10.25	10.260	Sink_0
Data	1	11.001	10.5	10.51	Sink
Data	1	11.501	11	11.01	Sink
ACK	2	4.100	11.25	11.260	Sink_0
Data	1	12.001	11.5	11.51	Sink
Data	1	12.501	12	12.01	Sink
ACK	2	4.500	12.25	12.260	Sink_0
Data	1	13.001	12.5	12.51	Sink
Data	1	13.501	13	13.01	Sink
ACK	2	4.900	13.25	13.260	Sink_0
Data	1	14.001	13.5	13.51	Sink
Data	1	14.501	14	14.01	Sink
ACK	2	5.300	14.25	14.260	Sink_0
Data	1	15.001	14.5	14.51	Sink
Data	1	15.501	15	15.01	Sink
ACK	2	5.700	15.25	15.260	Sink_0
Data	1	16.001	15.5	15.51	Sink
Data	1	16.501	16	16.01	Sink

5.3.3.2. Validation of CSGT Ingress Router Implementation

CSGT Ingress Router functionality that is different than CSVC Edge Router and CSGT Core Router functionalities is summarized below:

- CSGT Ingress Router has to recognize packet types and treat them according to their types. The values that will be kept for future use should be extracted from the ACK packets and should be inserted to the appropriate packets when needed.

Experiments in this chapter for CSGT Ingress Router not only validate the implementation but also present the effects of tag re-use mechanism in CSGT.

The first scenario is shown in Figure 5-12. The router has a transmission capacity of 10 packets/sec. and there are ten sources in the topology. Each of them is a source of a different flow (from 1 to 10). The sum of reserved rates of all flows is equal to the transmission capacity of the router. The reserved rate of each flow is 1 packet/sec. At time t=1 to t=2, flow 1 is the only backlogged flow. In this setting, by t=2, 10 packets of flow 1 are already serviced by the router and *VCore* value of the 11^{th} packet is 12. All other flows become backlogged at t=2.

Since the router services packets in increasing order of *VCore* values, eleventh packet of flow 1 is not serviced until t=10.2; hence, flow 1 receives no throughput during the interval [3, 10.2]. Given any time interval of arbitrary length, it is easy to extend this example to show that flow 1 receives no throughput during the interval of interest. Therefore, for any interval length, the CSGT router does not provide any non-trivial (non-zero) lower bound on throughput when tags are not re-used i.e when it behaves as a CSVC Edge Router. This is consistent with the definition of CSGT Ingress Router.

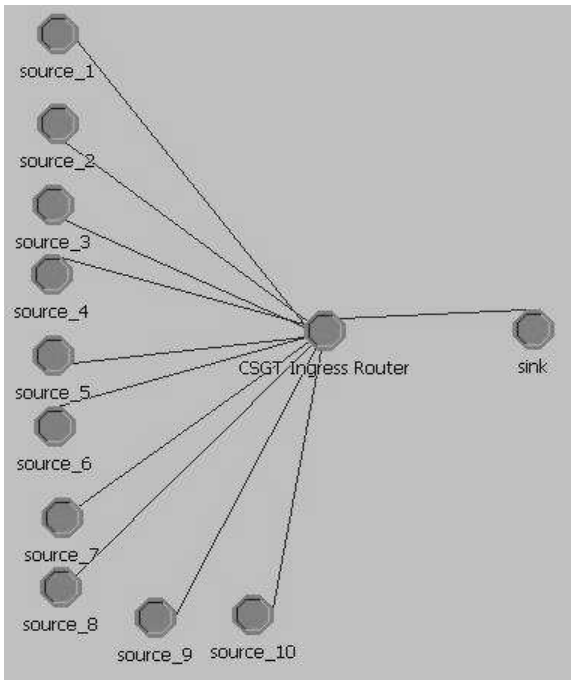

Figure 5-12 CSGT Ingress Router Simulation Topology-1

Table 5-5 Simulation Results for CSGT Ingress Router Validation Scenario-1

Flow No	VCore (sec)	A. Time (sec)	S Time (sec)		Flow No	VCore (sec)	A Time (sec)	S Time (sec)
1	2	1	1.1		4	7	2.4	6.5
1	3	1.1	1.2		5	8	2.5	6.6
1	4	1.2	1.3		6	8	2.5	6.7
1	5	1.3	1.4		7	8	2.5	6.8
1	6	1.4	1.5		8	8	2.5	6.9
1	7	1.5	1.6		9	8	2.5	7
1	8	1.6	1.7		10	8	2.5	7.1
1	9	1.7	1.8		2	8	2.5	7.2
1	10	1.8	1.9		3	8	2.5	7.3
1	11	1.9	2		4	8	2.5	7.4
5	3	2	2.1		5	9	2.6	7.5
6	3	2	2.2		6	9	2.6	7.6

Table 5-5-Cont.-Sim. Results for CSGT Ingress Router Validation Scenario-1

7	3	2	2.3		7	9	2.6	7.7
8	3	2	2.4		8	9	2.6	7.8
9	3	2	2.5		9	9	2.6	7.9
10	3	2	2.6		10	9	2.6	8
2	3	2	2.7		2	9	2.6	8.1
3	3	2	2.8		3	9	2.6	8.2
4	3	2	2.9		4	9	2.6	8.3
5	4	2.1	3		5	10	2.7	8.4
6	4	2.1	3.1		6	10	2.7	8.5
7	4	2.1	3.2		7	10	2.7	8.6
8	4	2.1	3.3		8	10	2.7	8.7
9	4	2.1	3.4		9	10	2.7	8.8
10	4	2.1	3.5		10	10	2.7	8.9
2	4	2.1	3.6		2	10	2.7	9
3	4	2.1	3.7		3	10	2.7	9.1
4	4	2.1	3.8		4	10	2.7	9.2
5	5	2.2	3.9		5	11	2.8	9.3
6	5	2.2	4		6	11	2.8	9.4
7	5	2.2	4.1		7	11	2.8	9.5
8	5	2.2	4.2		8	11	2.8	9.6
9	5	2.2	4.3		9	11	2.8	9.7
10	5	2.2	4.4		10	11	2.8	9.8
2	5	2.2	4.5		2	11	2.8	9.9
3	5	2.2	4.6		3	11	2.8	10
4	5	2.2	4.7		4	11	2.8	10.1
5	6	2.3	4.8		1	12	2	10.2
6	6	2.3	4.9		5	12	11	10.3
7	6	2.3	5		6	12	11	10.4

Table 5-5-Cont.-Sim. Results for CSGT Ingress Router Validation Scenario-1

8	6	2.3	5.1	7	12	11	10.5
9	6	2.3	5.2	8	12	11	10.6
10	6	2.3	5.3	9	12	11	10.7
2	6	2.3	5.4	10	12	10.7	10.8
3	6	2.3	5.5	2	12	10.8	10.9
4	6	2.3	5.6	3	12	10.9	11
5	7	2.4	5.7	4	12	11	11.1
6	7	2.4	5.8	1	13	11.1	11.2
7	7	2.4	5.9	5	13	11.2	11.3
8	7	2.4	6	6	13	11.3	11.4
9	7	2.4	6.1	7	13	11.4	11.5
10	7	2.4	6.2	8	13	11.5	11.6
2	7	2.4	6.3	9	13	11.6	11.7
3	7	2.4	6.4	10	13	11.7	11.8

The effect of tag re-use mechanism is presented in the second scenario in Figure 5-13. The only difference of this scenario compared to the first one is the existence of a source in the topology that creates acknowledgement packets for flow 1. This source generates ACK packets for packets 2, 3 and 4 and the router receives these packets at times 1.25, 1.35 and 1.45 respectively. This is the simulation of the case when these packets are received by the related egress router much before their *VCore* values. CSGT Ingress Router re-uses these values, so *VCore* value of the 11^{th} packet of flow 1 for this case is 9 instead of 12. As a result, the router again services packets in increasing order of virtual clock values, but the eleventh packet of flow 1 is serviced at t=7,5 not 10,2. The results (Table 5-6) show that the tag re-use mechanism works correctly. The penalty for flow 1 - because of its accumulated debit in the duration of [1, 2] - is reduced by the mechanism.

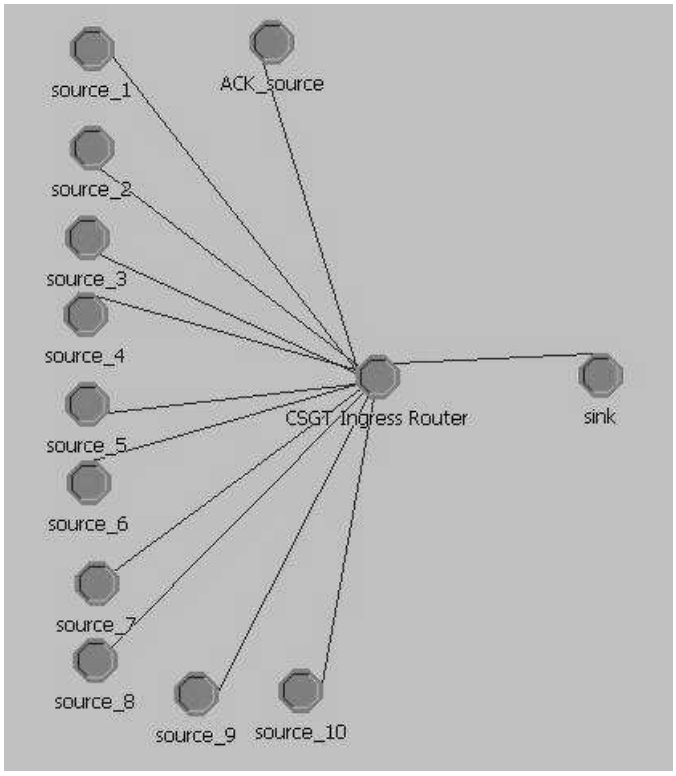

Figure 5-13 CSGT Ingress Router Simulation Topology-2

Table 5-6 Simulation Results for CSGT Ingress Router Validation Scenario-2

Flow No.	VCore (sec)	A. Time (sec)	S Time (sec)		Flow No.	VCore (sec)	A Time (sec)	S Time (sec)
1	2	1	1.1		4	7	2.4	6.5
1	3	1.1	1.2		5	8	2.5	6.6
1	4	1.2	1.3		6	8	2.5	6.7
1	2	1.3	1.4		7	8	2.5	6.8
1	3	1.4	1.5		8	8	2.5	6.9
1	4	1.5	1.6		9	8	2.5	7
1	5	1.6	1.7		10	8	2.5	7.1
1	6	1.7	1.8		2	8	2.5	7.2
1	7	1.8	1.9		3	8	2.5	7.3
1	8	1.9	2		4	8	2.5	7.4

Table 5-6-Cont.-Sim. Results for CSGT Ingress Router Validation Scenario-2

5	3	2	2.1		1	9	2	7.5
6	3	2	2.2		5	9	2.6	7.5
7	3	2	2.3		6	9	2.6	7.6
8	3	2	2.4		7	9	2.6	7.7
9	3	2	2.5		8	9	2.6	7.8
10	3	2	2.6		9	9	2.6	7.9
2	3	2	2.7		10	9	2.6	8
3	3	2	2.8		2	9	2.6	8.1
4	3	2	2.9		3	9	2.6	8.2
5	4	2.1	3		4	9	2.6	8.3
6	4	2.1	3.1		1	10	2.1	8.4
7	4	2.1	3.2		5	10	2.7	8.4
8	4	2.1	3.3		6	10	2.7	8.5
9	4	2.1	3.4		7	10	2.7	8.6
10	4	2.1	3.5		8	10	2.7	8.7
2	4	2.1	3.6		9	10	2.7	8.8
3	4	2.1	3.7		10	10	2.7	8.9
4	4	2.1	3.8		2	10	2.7	9
5	5	2.2	3.9		3	10	2.7	9.1
6	5	2.2	4		4	10	2.7	9.2
7	5	2.2	4.1		1	11	2.2	9.3
8	5	2.2	4.2		5	11	2.8	9.3
9	5	2.2	4.3		6	11	2.8	9.4
10	5	2.2	4.4		7	11	2.8	9.5
2	5	2.2	4.5		8	11	2.8	9.6
3	5	2.2	4.6		9	11	2.8	9.7

5.3.3.3. Validation of CSGT Egress Router Implementation

CSGT Egress Router functionality that is different than CSGT Ingress Router functionality is summarized below:

- When a CSGT Egress Router serves a packet, the router has to decide if *VCore* value of the packet is re-usable or not. If the *VCore* value is re-usable, the value that will be inserted to the acknowledge packet should be gathered from the packet itself. It should be inserted to the appropriate acknowledge packet, which should be then be sent towards the related ingress router.

The topology used for testing the above scenario is shown in Figure 5-14. In this topology, there are two sources creating traffic with constant packet generation rates. CSGT Egress Router decides if *VCore* value of the packet is re-usable or not by using the formula given in 4.10. Minimum latency encountered by the acknowledgement packet, namely D^{min}, is added to the router as a process model attribute for each flow and used in the formula.

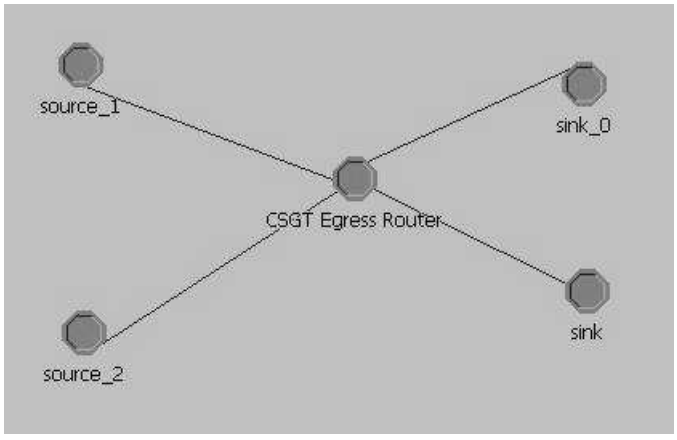

Figure 5-14 CSGT Egress Router Simulation Topology

In this scenario, source_1 and source_2 start generating packets at t=2 and t=3 respectively. Both sources generate five packets per second. The capacities at the

output interfaces of the router are 10 packets/sec. D^{min} is 5 for flow 1 and 4 for flow 2. Normal traffic packets are sent to “sink” resembling the output link and ACK packets are sent to “sink_0” resembling the first node on the path to the ingress router.

The results of the simulation are given in Table 5-7. The values of the fields on the packet are read when the packet is received by the related sink node. The results are consistent with the expected behavior. The router generates ACK packets correctly and sends them towards the correct direction. Since D^{min} is smaller for flow 2, its packets are acknowledged before the packets of flow 1.

Table 5-7 Simulation Results for CSGT Egress Router Implementation

Received by	Receiving time (sec)	Flow No	VCore (sec)
Sink	2.1	1	2.2
Sink	2.3	1	2.4
Sink	2.5	1	2.6
Sink	2.7	1	2.8
Sink	2.9	1	3
Sink	3.1	2	3.2
Sink	3.2	1	3.2
Sink	3.3	2	3.4
Sink	3.4	1	3.4
Sink	3.5	2	3.6
Sink	3.6	1	3.6
Sink	3.7	2	3.8
Sink	3.8	1	3.8
Sink	3.9	2	4
Sink	4	1	4
Sink	4.1	2	4.2

Table 5-7-Cont- Simulation Results for CSGT Egress Router Implementation

Sink	4.2	1	4.2
Sink_0	4.3	2	4.4
Sink	4.3	2	4.4
Sink	4.4	1	4.5
Sink_0	4.5	2	4.6
Sink	4.5	2	4.6
Sink	4.6	1	4.6
Sink_0	4.7	2	4.8
Sink	4.7	2	4.8
Sink	4.8	1	4.8
Sink_0	4.9	2	5
Sink	4.9	2	5
Sink	5	1	5
Sink_0	5.1	2	5.2
Sink	5.1	2	5.2
Sink	5.2	1	5.2
Sink_0	5.3	2	5.4
Sink	5.3	2	5.4
Sink_0	5.4	1	5.4
Sink	5.4	1	5.4
Sink_0	5.5	2	5.6
Sink	5.5	2	5.6
Sink_0	5.6	1	5.6
Sink	5.6	1	5.6

5.3.4.Validation of CSGF Router Implementations

CSGT core router, CSGT ingress router and CSGT sequencer are used directly in CSGF implementation as CSGF routers. None of their properties change so there is no validation requirement for these routers. However CSGF Ingress Router is a new implementation and needs to be validated.

5.3.4.1. Validation of CSGF Ingress Router Implementation

Since CSGF Ingress Router is built upon the already verified CSGT Ingress Router, the main focus of the validation becomes the fair scheduling algorithm implementation.

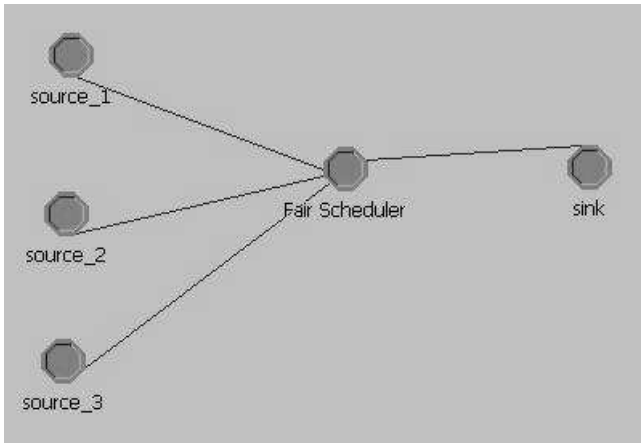

Figure 5-15 Fair Scheduler Simulation Topology

In the scenario (Figure 5-15), source_1, source_2 and source_3 start generating 1000 bit packets at t=1 from flows 1, 2, 3 respectively with a rate of 10 packets per second. The capacity at the output interface of the router is 3 packets per second. The reserved rates of the flows are 1000, 2000 and 3000 bits/s for flows 1,2 and 3 respectively. The weights of the flows are 1, 2 and 3 respectively. The number of packets in the queues of the scheduler will get larger since output capacity is smaller

than the number of coming packets. The scheduler should forward packets to the sink node at rates according to the weights of the flows. Fair scheduler should also insert the appropriate values to the “ratepk” value of the packets as it sends them. All packets sent from the same fair scheduler must have the same rate value written on the related fields.

The results of the simulation are given in Table 5-8. The values of the fields on the packets are read when the packets are received by the related sink node. The results are consistent with the expected behavior of scheduler. Flow 3, 2 and 1 shares the bandwidth proportional to their flow weights. Because of its higher weight, Flow 3 has the largest share of the bandwidth. In each two seconds, 3 packets from flow 1, 2 packets from flow 2 and 1 packet from flow 1 are sent by the scheduler.

Table 5-8 Simulation Results for Fair Scheduler Implementation

Time (sec)	Micro-flow	Macro-Flow	Rate before (bps)	Rate (bps)
1.34	1	1	1000	6000
1.68	3	1	3000	6000
2.02	2	1	2000	6000
2.36	3	1	3000	6000
2.70	3	1	3000	6000
3.04	2	1	2000	6000
3.38	3	1	3000	6000
3.72	2	1	2000	6000
4.06	3	1	3000	6000
4.40	1	1	1000	6000
4.74	2	1	2000	6000
5.08	3	1	3000	6000
5.42	3	1	3000	6000
5.76	3	1	3000	6000
6.10	2	1	2000	6000

Table 5-8-Cont.- Simulation Results for Fair Scheduler Implementation

6.44	3	1	3000	6000
6.78	2	1	2000	6000
7.12	1	1	1000	6000
7.46	2	1	2000	6000
7.80	3	1	3000	6000
8.14	3	1	3000	6000
8.48	1	1	1000	6000
8.82	2	1	2000	6000
9.16	3	1	3000	6000
9.50	2	1	2000	6000
9.84	3	1	3000	6000
10.18	3	1	3000	6000
10.52	1	1	1000	6000
10.86	2	1	2000	6000
11.20	3	1	3000	6000

CHAPTER 6

EVALUATION OF CSGF

6.1. Introduction

The design aim of CSGF is obtaining the first work-conserving, core-stateless QoS architecture that provides deterministic fairness guarantees. The most important properties of CSGF can be illustrated by considering its design procedure again (Figure 6-1):

- Firstly a stateful scheduling algorithm is chosen and core-stateless version of it is designed. VC is the selected scheduling algorithm and CSVC is its core-stateless counterpart. This process is inspired from JVC (the non work-conserving version of VC) and CJVC (its core-stateless version) pair [14] found in the literature. In summary, the two main properties of CSVC (and hence CSGF) are being core-stateless and work conserving.
- Second phase in the design procedure of CSGF is creating a QoS network architecture, which provides end-to-end throughput bounds within an additive constant, by integrating two mechanisms with CSVC. The throughput guarantee and two mechanisms (tag re-use and source rate control) are the two main properties of CSGT (and hence CSGF).
- As the name implies, one final important property of CSGF is its fairness guarantee.

In this section, CSGF architecture is evaluated in detail and some points conflicting with its main design aims are illustrated as deficiencies. The relationship between these features and their effects to each other are also investigated. The deficiencies of CSGF are shown, their effects are investigated and these ideas are supported by a simulation study of various cases. OPNET Version 11.5 is used for all simulation experiments in this section.

CBR traffic flows are used in the experiments. Since the main goal (observing the deficiencies of CSGF) is achieved with CBR traffic and since it is considered that CSGF will behave similar in terms of fairness with other types (e.g. Poisson, Bursty) of traffic, we believe it is appropriate to conduct simulations with CBR traffic only.

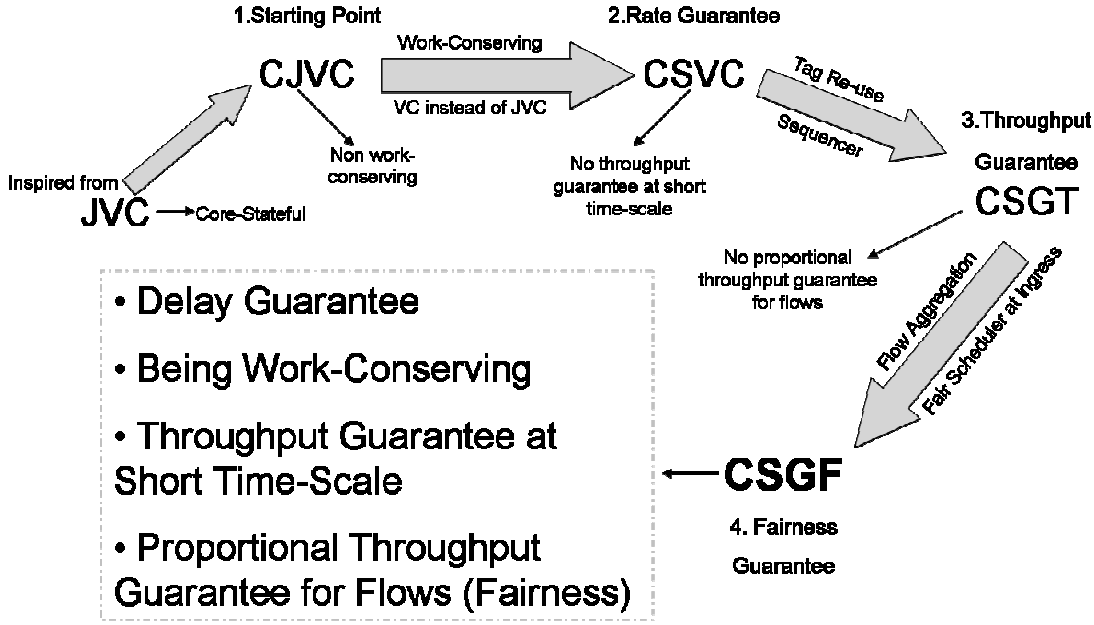

Figure 6-1 Design Phases and Aims of CSGF

6.2. Evaluation Cases

- **Case 1: No fairness guarantee for micro-flows of different ingress routers**

"Fairness guarantee" is defined in terms of excess throughput that each flow receives in CSGF. A network cannot be called "fair" if it provides throughput at reserved rate to one flow and allows another flow to use significantly more than its reserved rate. According to the most significant assumption made in the design of CSGF, the fairness provided in a network is meaningful only when it is applied to the flows that share the entire end-to-end path. That is why CSGF applies fair scheduling at the ingress routers to the flows that share the entire end-to-end path.

Consider two (or more) macro flows entering the network from different ingress routers and following the entire end-to-end path except their ingress routers. Then throughout the coinciding path (nearly the entire path), CSGF provides fairness for micro flows, only for the bandwidth that their corresponding macro-flow gets. This is the only fairness that CSGF provides. It doesn't provide any fairness guarantee between macro flows or between micro flows of different macro-flows although these flows share almost their entire paths. We believe that this case is conflicting with the main CSGF idea of fairness. The difference between the paths of two flows in Figure 6-2 is their first links only. The number of core routers can be increased to strengthen the idea in this case. If fairness is meaningful when it is applied to the flows sharing the entire end-to-end path, we think these two macro-flows are worth treating fairly.

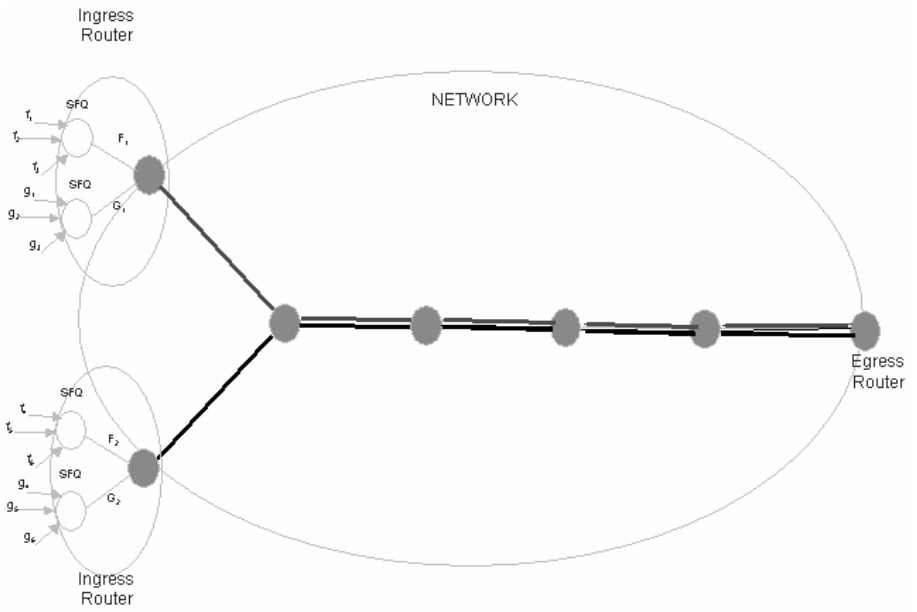

Figure 6-2 Two macro-flows sharing the entire path except the first link

CSGF gives no fairness guarantee to macro-flows of the above type. Several simulations are carried out in different conditions to see the correctness of our hypothesis. The bandwidth shares of macro-flows depend on D^{min}, sequencer buffer size, packet sizes and aggressiveness of the routers.

Simulation results in Figure 6-3 show different cases with the topology of Figure 6-2. We simulated three cases for a scenario where both of the macro-flows have 0.1 packets/sec reserved rate. However sources generate traffic with unconforming rates. All of the routers have 100 packets/sec transmission capacities except the last core router on the path, which has 5 packets/sec capacity. Since the sum of the reserved rates of the flows is equal to 0.2 packets/sec, the rest of the bandwidth, i.e 4.8 packets/sec, is excess bandwidth. A fair algorithm would divide the bandwidth into two equal parts if both of the flows produce packets with a rate more than 2.4 packets/sec.

1st case refers to identical unconforming rate values, 10 packets/sec, for both flows. In this case, the values are close to what is desired. The excess bandwidth shares of the flows are nearly equal.

2nd case refers to the case where flow 1 has a non-conforming rate of 10 packets/sec and flow 2 has a non-conforming rate of 5 packets/sec. Tag re-use mechanism is used with reserved feedback channels in this scenario. In order to be fair, the algorithm should protect the excess bandwidth from the aggressiveness of flow 1 and give fair shares to both flows. This is not the case for CSGF and it cannot achieve this. Flow 1 dominantly captures the excess bandwidth in Case 2.

The scenario in the 3rd case is same with the one in Case 2. However D^{min} is lower and sequencer buffer size is larger for Flow 2 compared to the same values of Flow 1. Consequently more tags are re-used for Flow 2 and as shown in Figure 6-3, flow 2 gets a larger share from the bandwidth compared to Case 2. The resultant bandwidth sharing is still unfair but not as significant as in the 2nd Case.

As a result of the experiments, the work-conserving nature of CSGF is clearly demonstrated. The excess bandwidth is used by the flows but sharing is not fair. The bandwidth shares that the flows take is proportional to many factors, so the proportional allocation cannot be guaranteed.

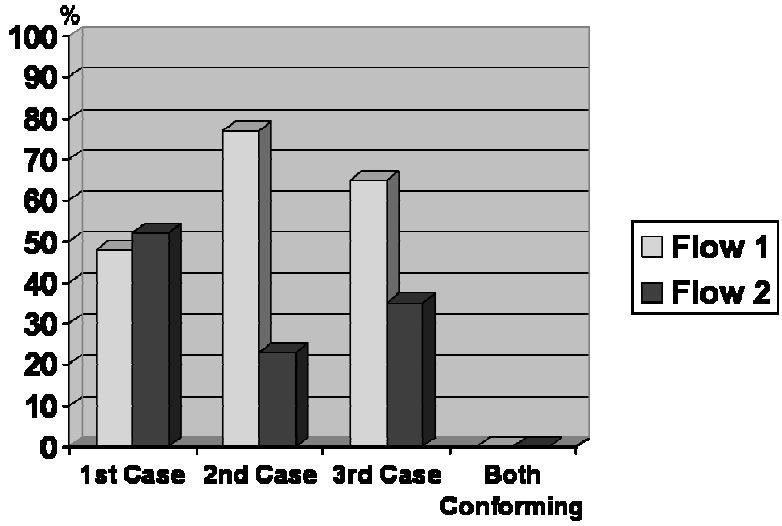

Figure 6-3 Use of Excess Bandwidth for two macro-flows sharing the entire path except the first link

- **Case 2: No fairness guarantee for micro-flows of same ingress router that share most of their paths**

When we consider two micro-flows entering the network from the same ingress router and following the same path all through the network except the last link, there will again be no fairness guarantee to these flows. The algorithm will treat these flows same as two flows entering the network from the same ingress router and following two entirely separate paths. However a micro-flow of one of these flows and a micro-flow of the other macro-flow will share almost entirely the same path and the same bandwidth. The bottleneck of both of these flows will most probably be on their coinciding path where no fairness is guaranteed.

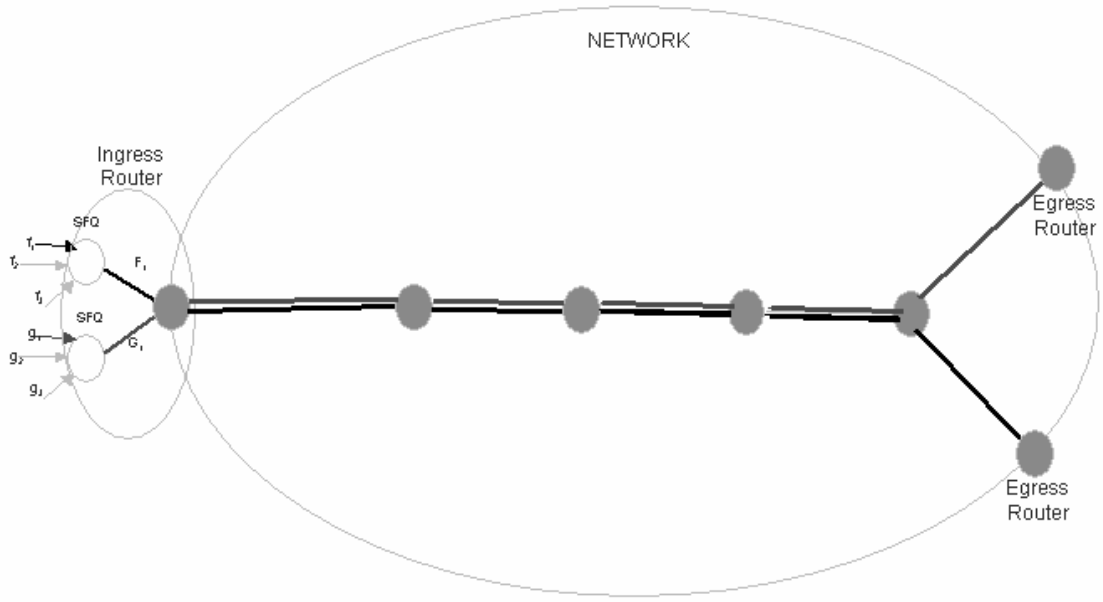

Figure 6-4 Two macro-flows sharing the entire path except the last link

In addition, the micro-flows described above are put into different fair schedulers just because of their last hops. Consequently they are inserted into different queues in CSGF Ingress Router. Several simulations are carried out in different conditions to see the correctness of our hypothesis. The results show that no fairness is guaranteed to such flows on CSGF network.

In Figure 6-5, simulation results are illustrated for three different cases on the topology of Figure 6-4. We simulated a scenario where both of the macro-flows have 0.1 packets/sec reserved rate. However sources generate traffic with unconforming rates. All of the routers have 100 packets/sec transmission capacities except the last core router on the path. That router can send 5 packets/sec only. Since the sum of the reserved rates of the flows is equal to 0.2 packets per second, the rest of the capacity, i.e 4.8 packets/sec, is excess bandwidth. When both flows have unconforming rates, the sum of which is greater than 2.5 packets/sec, the queue of the last core router fills up.

1st case refers to identical unconforming rate values, 10 packets/sec, for both flows. According to the simulation results, the excess bandwidth is used by both of the routers at similar rates.

2nd case refers to the case where flow 1 has a non-conforming rate of 8 packets/sec and flow 2 has a non-conforming rate of 5 packets/sec. Since both flows have rates greater than 2.5 and their reserved rates are the same, equal number of packets should be sent from the core router in order to be fair. Simulation results show that Flow 1 gets its larger share as it becomes more aggressive than Flow 2. This is not appropriate for proportional allocation principle. Tag re-use mechanism is used with reserved feedback channels in this scenario.

The work-conserving nature of CSGF is illustrated in this simulation again. The excess bandwidth is used and flows get larger rates if there is enough bandwidth on the way to the egress router. However there is no proportional bandwidth allocation.

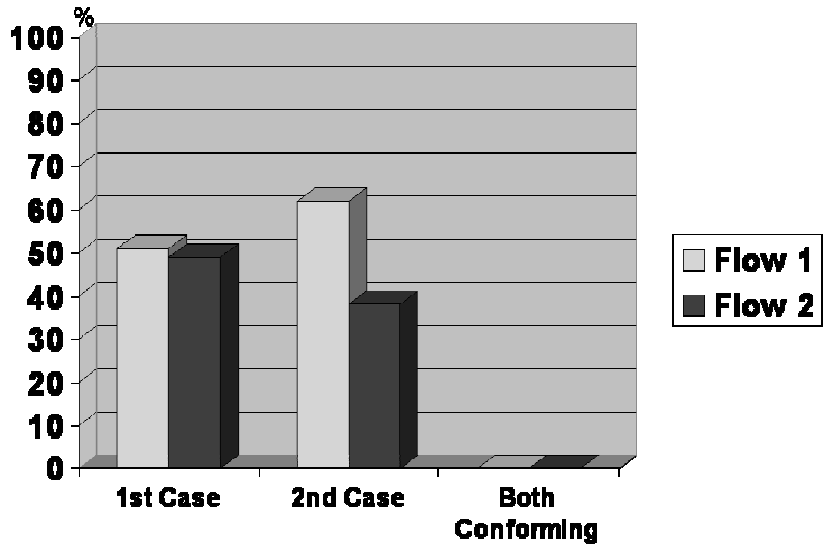

Figure 6-5 Use of Excess Bandwidth for two macro-flows sharing the entire path except the last link

- **Case 3: One and only bottleneck where no fairness guarantee exists**

In this case, the word “bottleneck” is used for describing the link with the heaviest traffic. If two or more distinct (with different paths) macro-flows share only one link and if it is the bottleneck link of all the flows, there will still be no guaranteed fairness for these flows. If we assume that all other links are fast links such that no congestion occurs, bottleneck link will be the most important link in terms of proportional bandwidth allocation hence fairness.

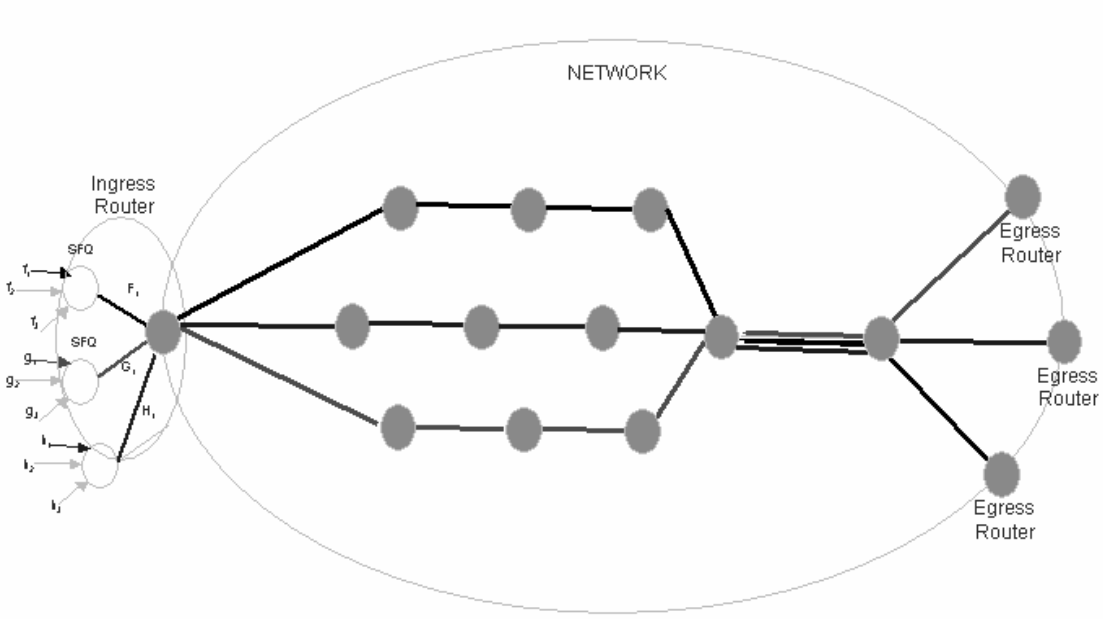

Figure 6-6 Three macro-flows sharing only one bottleneck link

We simulated a scenario where the macro-flows F, G and H have 1, 2 and 3 packets/sec reserved rates respectively. Sources generate packets of micro-flows and the micro-flows, which share their entire end-to-end path, form a macro-flow. Each macro-flow in this scenario has a non-conforming rate of 10 packets/sec. All of the links have 18 packets/sec capacities.

Each flow in the scenario shares its first, fifth and sixth nodes with the other two flows. Paths of the three flows coincide at the fifth node and the link between the fifth and sixth core router is common for all three flows. Since the link capacities are larger than even the non-conforming rates of the flows on their paths until the fifth node, there will be about 30 packets/sec input to this fifth core router. Sum of the reserved rates of the flows is equal to 6 packets per second. This rate must be guaranteed by the algorithm on this bottleneck link between the fifth and sixth routers. The rest of the capacity, i.e., 12 packets/sec, is excess bandwidth. In order to treat the flows fairly, the excess bandwidth should be given to flows proportional to their reserved rates. The ideal bandwidth shares and the ideal excess bandwidth shares of the flows in this case are illustrated in Figure 6-7.

The simulation results however show that the reserved rates are provided to the flows but no fairness is guaranteed in the scenario for excess bandwidth usage of these flows on the CSGF network. Figure 6-7 shows the excess bandwidth share of each flow in packets/sec. Each flow gets an excess bandwidth share that is close to the share of other flows. There is no proportional bandwidth sharing in this case. The total bandwidth shares of the flows including the excess bandwidth are also in Figure 6-7. Flow F gets a bandwidth share more than it should get in ideal conditions. Flow G gets a bandwidth share close to the ideal case and Flow H, which is the least non-conforming flow, gets a smaller portion from the bandwidth than it deserves.

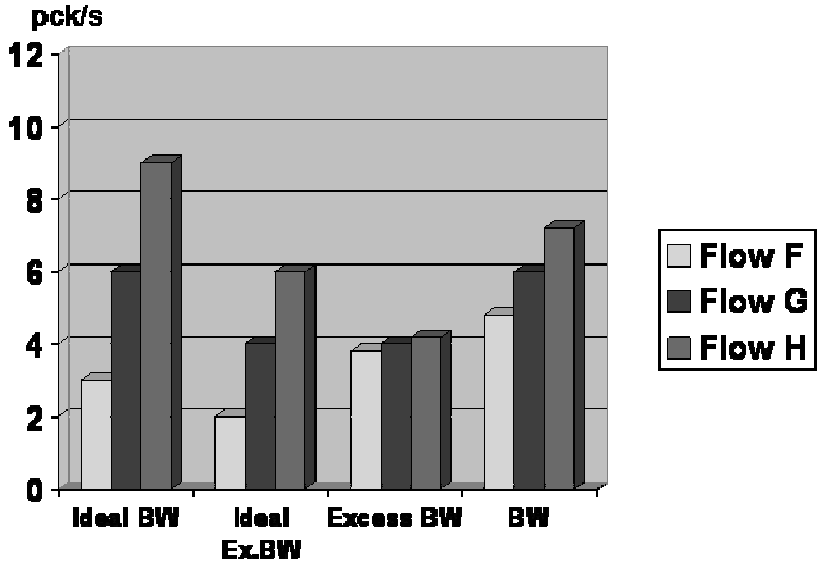

Figure 6-7 Simulation Results for macro-flows sharing only one bottleneck link

- **Case 4: Complexity of ingress router in CSGF**

In a given network, it is important to allocate resources in an effective way. In order to achieve this, two flows may be forced to follow two distinct paths even if their ingress and egress routers are the same. Actually network administrators may prefer to differentiate the paths of flows in a network as much as possible to distribute the load on the network evenly. The possibility of having bottlenecks may be reduced in this way. In a given network, there may be little number of flows having the same entire end-to-end path. In CSGF, the 'fairness' is applied only to flows that share the entire end-to-end path. When the percentage of the flows of this kind is reduced, the applied fairness also decreases. In this case, CSGF network behaves much more like a CSGT network. The fairness of CSGF is directly proportional to the percentage of flows sharing the entire end-to-end paths.

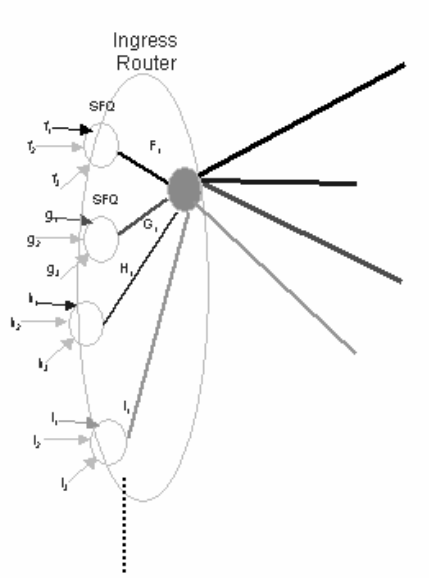

Figure 6-8 Complexity of ingress router in CSGF

CSGF applies fair scheduling at the ingress routers to the flows that share the entire end-to-end path. Therefore, the paths of the flows have to be defined when they enter the network, which mandates the use of a mechanism like strict source-routing or explicit routing in which the entire route of the packet is carried at the header.

6.3.Evaluation Results

In the cases described above, CSGF cannot guarantee fairness and behaves the same as CSGT. Since “fairness” is the only property added to the CSGT, we call these conditions as “deficiencies of CSGF”.

CSGF provides fairness guarantee only for flows entering the same ingress router, following the same entire end-to-end path. This type of flows may form only a small percentage of all flows in a network and since there is no fairness guarantee in CSGT, so will there be in the CSGF even though the complexity of edge routers is additionally increased for transforming a CSGT network to a CSGF network.

When we consider fairness in CSGF, we consider the excess bandwidth. Since CSGF has throughput guarantees, two flows will get their reserved rates when they share the same link. However in the conditions when CSGF applies no fairness, one of these flows can get most of the bandwidth when the other gets only its reserved rate. Since the reserved rates are provided, the proportional throughput guarantee of the excess bandwidth may seem to be a not necessarily needed property. However “being work-conserving” is one of the main features of the algorithm. CSVC is designed as a work-conserving algorithm in order to meet the rising traffic demands. Utilizing resources evenly is a desirable feature and it is a part of the basis of CSVC, correspondingly CSGF. Therefore, when CSGF mentions “Fairness”, proportional allocation of both reserved and the excess bandwidth is necessary.

Tag re-use gives CSGT the property of providing throughput guarantees at short time-scales. However it also creates uncertainty in terms of fairness. The re-usability of a packet depends on the link conditions and the size of the packet, both of which can change significantly in the life-time of a flow. Tag re-use mechanism

may be stopped or restricted in order to get rid of this uncertainty. However this would have two effects:

- The extent to which the source can utilize idle bandwidth in the network would be limited.
- The first router wouldn't transmit a packet before its expected arrival time, which means the router reduces to the non-work-conserving JVC router. More the mechanism is restricted; more the ingress router behaves like JVC router.

Both of the effects are undesirable. A CSGF network should utilize idle bandwidth as much as possible since "being work-conserving" is one of its most important properties. It is also undesirable to have a JVC-like router as the ingress router. JVC router is the starting point in the design (Figure 6-1) and it will almost mean returning to the point where the design procedure started. We also experienced the importance of tag re-use in terms of fairness in our simulations.

The one and only feature added on CSGT when designing CSGF is the use of fair schedulers at the Ingress Routers. Two flows are not put into the same fair scheduler in the CSGF Ingress Router even if %99 of their paths coincides. This situation generates two effects:

- "Fairness Application" criterion of CSGF is so narrow that two flows do not share the bandwidth fairly even if they almost fit the idea behind fairness in CSGF.
- There are a lot of path probabilities in a network and if two flows are not put into the same fair scheduler in the CSGF Ingress Router even if %99 of their paths coincides, then there will be a huge number of fair scheduler implementations in each CSGF Ingress Router.

Beyond all other features of CSGF (or CSGT or CSVC), the most important property of the algorithm is its core-stateless structure. Then we should never forget the main aim in designing core-stateless networks is giving QoS guarantees to networks with fine granularity and in a scalable way. If CSGF is scalable, it will be used in networks where there are a lot of end-to-end path probabilities. In case of heavy traffic, there will be many flows on Ingress Router, having distinct end-to-end paths. If one fair scheduler is used for each of these paths, then the Ingress Router will be too complex and hard to implement. It is already complex because of the extra state hold for reusable tag values. If the number of fair schedulers is bounded, then the level of the most important feature of core-stateless networks, scalability, is reduced.

The throughput guarantee of CSGF depends on the maximum delay and loss experienced in the feedback channels where acknowledgement packets are sent through. Adequately provisioned feedback channels between edge routers should be constructed.

The main goal of CSGF design is providing the fairness guarantee at the level that a network of core-stateful routers does. Core-stateful routers provide per-link proportional throughput guarantee. Since the control is only at the edges for a core-stateless network, we believe that it is not possible to provide fairness in core-stateless networks at the same level with the fairness in networks where link-based fairness is applied.

CHAPTER 7

CONCLUSION

The primary goal of providing Quality of Service (QoS) is to have better and more predictable network services by providing dedicated bandwidth, controlled jitter and latency, and improved loss characteristics. QoS achieves these goals by providing tools for managing network congestion, shaping network traffic, using expensive wide-area links more efficiently, and setting traffic policies across the network [44]. Providing QoS in the Internet has been a challenging task for the network community for a while. On one side, network architects want to keep the network core as simple and scalable as possible. However, designers who are in favor of the idea of obtaining fine grain service differentiation would like to use complex routers at the network core. There have been two major architectural proposals, Intserv and Diffserv, for providing QoS in the IP networks. Both of the proposals can support QoS in IP networks, obviously with pros and cons on each side. This thesis firstly reviews these two main models.

The drawback of the stateful solutions is their complexity. On the control path, the routers should install and maintain per-flow state. On the data path, per-flow classification, per-flow buffer management and per-flow scheduling should be handled. It is a challenge to keep per-flow state consistent in the routers. Stateless solutions are more scalable and robust. However stateless solutions cannot provide

as powerful and flexible services as stateful solutions. They also cannot provide low delay guarantees and high resource utilization simultaneously.

Core-stateless approaches try to take positive features of both sides. The core-stateless systems use scalable mechanisms in the core of the networks and stateful approaches at the edges of the network in order to get rid of the scalability problem and support QoS with fine granularity. Because of their scalability in supporting QoS, core-stateless systems have received considerable attention recently. The proposed architectures in the literature differ in terms of guarantees they provide. These mechanisms have been studied extensively and therefore a literature survey on past studies on the mechanisms of core-stateless architectures is also included. As a result of the survey, Core-Stateless Guaranteed Fair (CSGF) network architecture is chosen to evaluate in this thesis. The properties, advantages and disadvantages of the underlying approaches are given in order to explain the reasoning behind the selection of CSGF.

CSGF is built upon Core-Stateless Guaranteed Rate (CSGR) Network, which is a work-conserving, core-stateless network architecture that can provide end-to-end delay guarantees. Therefore CSGR is studied first. CSGR proposes a methodology to transform any Guaranteed Rate (GR) per-flow scheduling algorithm into a version that does not require per-flow state to be maintained in the core routers. In CSGR, the upper bounds on packet deadlines at core nodes are computed using per-flow state only at the edge node. A CSGR network provides the same end-to-end delay as the networks using actual deadlines. CSGR supports only delay guarantees and average throughput guarantees at large time-scales.

CSGR is combined with two mechanisms proposed in CSGT (Core-Stateless Guaranteed Throughput), namely tag re-use and source rate control to provide throughput guarantees at small time-scale. CSGT provides throughput bounds

within an additive constant of throughput bounds achieved by a network of core-stateful fair rate routers.

When CSGT is combined with fair access at the edge nodes and aggregation of flows in the core nodes, this combination leads to CSGF. CSGF claims to be the first core-stateless, work-conserving QoS architecture providing delay, throughput and fairness guarantees. This thesis presents detailed information about CSGR, CSGT, CSGF and discusses the basic mechanisms to support their desired behavior.

Implementation study of VC, CSVC, CSGT and CSGF is carried out using OPNET simulation program. VC is the selected Guaranteed Rate (GR) per-flow scheduling algorithm to be transformed to the core-stateless version by CSGR. Thus a detailed description of VC is also given. All of the routers used in selected algorithms are added to OPNET and implementation steps are described. Implementations are made in project, node and process models of OPNET version 11.5. The simulation environment of OPNET is also described.

Validation of our VC, CSVC, CSGT and CSGF implementations using OPNET simulation program is carried out and behaviors of the implemented routers are illustrated in this thesis. Simulation set ups and associated experiments are presented for each verification study. There are very few publicly available core-stateless QoS network architecture implementations in simulation environments. Therefore this simulation work may also prove useful in future work about CSGF.

The points in the design of CSGF conflicting with its expected features are shown as deficiencies. Most of the deficiencies are related to the assumption stating that fairness becomes meaningful only when it is applied for flows that share the entire end-to-end path. According to the evaluation results, CSGF cannot guarantee

fairness in general and behaves same as CSGT in other cases. Several conditions where fairness is not provided are conflicting with the above main assumption made for fairness in CSGF. CSGF has a very strict criterion to apply fairness for flows. Two flows do not share the bandwidth fairly even if they share most of their paths, i.e. even if they almost fit the idea behind fairness in CSGF.

The mechanisms added on CSVC when designing CSGT provide throughput guarantees at short time-scales. However they also create uncertainty in terms of fairness. Limiting these mechanisms reduces the ingress router to a JVC router, which is also undesirable.

In a large network with heavy traffic, there will be a huge number of fair scheduler functions in each CSGF Ingress Router and it will be a very complex node. If the number of fair schedulers is bounded, then the level of the most important feature of core-stateless networks, scalability, is reduced.

Core-stateful routers provide per-link proportional throughput guarantee. Since the control is only at the edges for a core-stateless network, we believe that it is not possible to provide fairness in core-stateless networks at the same fairness level achieved when link-based fairness is applied. Moreover, with the level of fairness it proposes, we think that CSGF is not sufficiently capable to be called a “Fair” QoS architecture. It may be called as “Improved CSGT”.

QoS architecture investigated in this thesis tries to provide QoS guarantees without maintaining per-flow state in core routers. Admission control is one of the issues that should be considered when applying this algorithm. Taking one of the admission control frameworks recently proposed in the literature ([45],[46],[47]) and using it in accordance with CSGF may be a future work. With this integration, CSGF can be evaluated in large networks to see its scalability and in many different traffic conditions to re-evaluate its performance.

CSGF treats two flows same as CSGT even if only a small part of one flow's path is different than the path of the other flow. As future work, taking a percentage of the route into account when providing fairness but not the entire path may be investigated and considered as an add-on to CSGF.

CSGT is the first work-conserving core-stateless network architecture that provides throughput guarantees at short time-scales. Its design aims are fulfilled, but we believe that the fairness provided by CSGF is not sufficient. As a future work, CSGT may also be investigated to provide fairness guarantees in a different way.

CHAPTER 8

REFERENCES

1. Odlyzko, K.C.a.A., The Size and Growth Rate of the InternetMarch 2001.
2. Clark, D.D., Internet cost allocation and pricing, MIT Press, in Internet economics. 1997: Cambridge, MA.
3. Janusz Gozdecki, A.J., and Rafal Stankiewicz, Quality of Service Terminology in IP Networks. IEEE Communications Magazine, March 2003. Vol.41(No.3).
4. Agnihotri, A., Study and Simulation of QoS for Multimedia Traffic, M.S. Project. November, 2002.
5. Braden, B., Clark, D., Shenker, S, "Integrated Services in the Internet Architecture: An Overview", IETF RFC 1633. 1994.
6. White, P.P. and J. Crowcroft, The integrated services in the Internet: state of the art. Proceedings of the IEEE, 1997. 85(12): p. 1934-1946.
7. Braden, R., et al., Resource Reservation Protocol (rsvp), rfc 2205, RFC 2205, Sept. 1997.
8. White, P.P., RSVP and integrated services in the Internet: a tutorial. IEEE Communications Magazine, 1997. 35(5): p. 100-106.
9. Kilkki, K., Differentiated Services for the Internet. 1999: Macmillan Publishing Co., Inc. Indianapolis, IN, USA.
10. http://www.ietf.org.
11. RFC 791 Internet Protocol Darpa Internet Program Protocol Specification. 1981 September.

12. K. Nichols, S.B., F. Baker, and D. L. Black, Definition of the Differentiated Services Field (DS Field) in the ipv4 and ipv6 Headers, Internet Draft. draf-ietfdiffserv-header-04.txt. October 1998.
13. Beijnum, I.v., IPv6 Internals. Cisco Internet Protocol Journal. 9.
14. Stoica, I. and H. Zhang, Providing guaranteed services without per flow management. Proceedings of ACM SIGCOMM, 1999. 99: p. 81-94.
15. Stoica, I., CMU-CS-00-176 Stateless Core: A Scalable Approach for Quality of Service, PhD Thesis, December 15, 2000 .
16. Zhang, L., VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks. ACM Transactions on Computer Systems (TOCS), May 1991. Volume 9(Issue 2): p. Pages: 101 - 124.
17. Norival R. Figueira, J.P., An upper bound on delay for the VirtualClock service discipline. IEEE/ACM Transactions on Networking (TON), Aug. 1995. v.3(n.4): p. p.399-408.
18. Keshav, S., On Efficient Implementation of Fair Queuing. Journal of Internetworking Research, September 1995: p. 157–173.
19. A. Demers , S.K., S. Shenker. Analysis and simulation of a fair queueing algorithm. in Symposium proceedings on Communications architectures & protocols. September 25-27, 1989. Austin, Texas, United States.
20. Configuring Weighted Fair Queueing, in Cisco IOS Quality of Service Solutions Configuration Guide, Cisco, 2005.
21. Kaur, J. and H.M. Vin, Core-stateless guaranteed rate scheduling algorithms. INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2001. 3.
22. Zhang, Z.L., Z. Duan, and Y.T. Hou, Virtual time reference system: a unifying scheduling framework forscalable support of guaranteed services. Selected Areas in Communications, IEEE Journal on, 2000. 18(12): p. 2684-2695.

23. Deleuze, C., Fdida, S., Stateless Virtual Clock: Scalable Packet Scheduling for Delay Guarantees in Core Networks. 2000, Universite Pierre et Marie Curie: Paris. p. 17.
24. Pan, R., B. Prabhakar, and K. Psounis, A stateless active queue management scheme for approximating fair bandwidth allocation. IEEE INFOCOM 2000, 2000. 212: p. 214.
25. Kaur, J. and H. Vin, Providing Deterministic End-to-end Fairness Guarantees in Core-stateless Networks. Book Quality of Service - IWQoS 2003: 11th International Workshop. Vol. Volume 2707/2003 2003, Berkeley, CA, USA: Springer Berlin / Heidelberg. Pages 401-421.
26. Dong Lin, R.M. Dynamics of Random Early Detection. in SIGCOMM'97. 1997. Cannes, France.
27. Duan, Z., et al., A Core Stateless Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. IEEE Trans. Parallel and Distributed Systems, 2004. 15(2): p. 167–182.
28. Zhenhai Duan, K.P., BCQ: Bin-based core stateless packet scheduler for scalable and flexible support of guaranteed services, in Computer Science Department. 2005, Florida State University: Florida.
29. Purnachandra, K.P., BCQ: Bin-based core stateless packet scheduler for scalable and flexible support of guaranteed services.
30. Cao, Z., Z. Wang, and E. Zegura, Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State. Proc. IEEE INFOCOM, 2000: p. 922-931.
31. Stoica, I., S. Shenker, and H. Zhang, Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. ACM SIGCOMM Computer Communication Review, 1998. 28(4): p. 118-130.
32. Clerget, A. and W. Dabbous, UF: Tag-based Unified Fairness. Proc. IEEE INFOCOM, 2001: p. 498-507.
33. P. Goyal, H.V., and H. Cheng. Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. in In Proceedings of ACM SIGCOMM'96. August 1996.

34. Kaur, J. and H.M. Vin, Core-stateless guaranteed throughput networks. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE. 3.
35. J. Kaur, H.V., Core-stateless Guaranteed Throughput Networks. Technical Report TR-01-47, Department of Computer Sciences, University of Texas at Austin. November 2001.
36. C. Li, E.K., Coordinated Network Scheduling: A Framework for End-to-end Services. In IEEE ICNP, November 2000.
37. Dallas E. Wrege , E.W.K., Hui Zhang , Jörg Liebeherr, Deterministic delay bounds for VBR video in packet-switching networks: fundamental limits and practical trade-offs. IEEE/ACM Transactions on Networking (TON), June 1996. v.4(n.3): p. p.352-362.
38. Zhang, J.C.R.B.a.H. WF^2Q: Worst-case Fair Weighted Fair Queuing. in In Proceedings of INFOCOM'96. March 1996.
39. J. C. R. Bennett, H.Z., Hierarchical Packet Fair Queuing Algorithms. Computer Communication Review, 1996. Vol. 26(No. 4).
40. J. Kaur, H.V. End-to-end Fairness Analysis of Fair Queuing Networks. in In Proceedings of the 23rd IEEE International Real-time Systems Symposium (RTSS). December 2002.
41. Kaur, J., Scalable Network Architecture for Providing Per-flow Service Guarantees, in Computer Science. 2002, University of Texas at Austin: Austin, Texas. p. 153.
42. OPNET Modeler Product Documentation Release 11.5, in 11.5. 2006, OPNET.
43. OPNET Support Website (www.opnet.com/support).
44. Quality of Service (QoS), in Quality of Service (QoS), Internetworking Technologies Handbook, Cisco, 2005.
45. Z.L. Zhang, Z.D., Y.T. Hou, L. Gao. Decoupling QoS Control from Core Routers:A Novel Bandwidth Broker Arcitecture for Scalable Support of

Guaranteed Services. in In Proceedings of ACM SIGCOMM. August 2000. Sweden.

46. Ion Stoica, H.Z. LIRA: An Approach for Service Differentiation in the Internet. in NOSSDAV'98. July 1998. London, UK.
47. S. Bhatnagar, B.R.B. Distributed Admission Control to Support Guaranteed Services in Core-stateless Networks. in In Proceedings of IEEE INFOCOM. April 2003.
48. Blake, S., et al., RFC2475: An Architecture for Differentiated Service. Internet RFCs, 1998.
49. Nicolas Christin, J.L., A QoS Architecture for Quantitative Service Differentiation. IEEE Communications Magazine, June 2003.