Load Balancing for the Agile All-Photonic Network

Imad Khazali

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

March 2012

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Imad Khazali

Entitled: Load Balancing for the Agile All-Photonic Network

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Electrical & Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Deborah Dysart-Gale — Chair

Dr. Michel Kadoch — External Examiner

Dr. Olga Ormandjieva — External to Program

Dr. Ahmed K. Elhakeem — Examiner

Dr. Ferhat Khendek — Examiner

Dr. Anjlai Agarwal — Thesis Supervisor

Approved by

Chair of Department or Graduate Program Director

March 29, 2012

Dean of Faculty

ABSTRACT

Load Balancing for the Agile All-Photonic Network

Imad Khazali, Ph.D.

Concordia University, 2012

The Agile All-Photonic Network (AAPN) uses Time Division Multiplexing (TDM) to better utilize the bandwidth of Wavelength Division Multiplexing (WDM) systems. It uses agile all-photonic switches as advances in the photonic switching technology made the design of all-photonic devices with switching latency in the sub-microseconds feasible. The network has a simplified overlaid star architecture that can be deployed in a Metropolitan Area Network (MAN) or a Wide Area Network (WAN) environment. This overlaid architecture, as opposed to general mesh architecture, scales network capacity to multiples of Tera bits per second, simplifies routing, increases reliability, eliminates wavelength conversion, and the need for accurate traffic engineering.

The objective of this thesis is to propose and analyze different load balancing methods for the deployment of the AAPN network in a WAN environment. The analysis should provide interested Internet Service Providers (ISPs) with a comprehensive study of load balancing methods for using the AAPN network as their backbone network. The methods balance the load at the flow level to reduce packet reordering. The methods are stateless and can compute routes quickly based on the packet flow

identifier. This is an important issue when deploying AAPN as an Internet backbone network where the number of flows is large and storing flow state in lookup tables can limit the network performance.

The load balancing methods, deployed at the edge nodes, require reliable signaling with the bandwidth schedulers at the core nodes. To provide a reliable channel between the edge and core nodes, the Control Messages Delivery Protocol (CMDP) is proposed as part of this thesis work. The protocol is designed to work in environments where propagation delays are long and/or the error rates are high. It is used to deliver a burst of short messages in sequence and with no errors. Combined with the reliable routing protocol proposed previously for the AAPN network, they form the control plane for the network.

To extend the applicability of the load balancing methods to topologies beyond AAPN overlaid star topology, the Valiant Load Balancing (VLB) method is used to build an overlaid star topology on top of the physical network. The VLB method provides guaranteed performance for highly variable traffic matrices within the hose traffic model constraints. In addition to the guaranteed performance, deploying the VLB method in the AAPN network, eliminates signaling and replaces the dynamic core schedulers with static scheduler that can accommodate all traffic matrices within the hose traffic model boundaries.

Acknowledgments

I have had the good fortune of having Dr. Anjali Agarwal as my thesis advisor. She has not only offered invaluable assistance and supported me financially but also given me the freedom to explore and find my own research topic.

Deepest thanks to the members of my supervisory committee, Dr. Ahmed Elhakeem, Dr. Ferhat Khendek, Dr. Olga Ormandjieva and my external examiner Dr Michel Kadoch without their knowledge and assistance this thesis would not have been successful.

I owe my deepest thanks to my father, mother, brothers and sisters for deep love and encouragements they have given me.

Next, I would like to thank all the people in my lab for making it a friendly and lively working place.

Lastly, I would like to express my deep gratitude to my wife Ola for her assistance in many ways for the successful completion of this thesis. During the many late nights and long weekends I spent working on this thesis; her patience was tried, but never failed. Most importantly, she has provided me with two beautiful children, Yamen and Lana, who have brought more happiness into my life than I deserve.

To My Mother, father, and Wife

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AAPN	Agile All-Photonic Network
AHRW	Adaptive Highest Random Weight
ARQ	Automatic Repeat reQuest
ATM	Asynchronous Transfer Mode
BE	Best Effort
CAC	Connection Admission Control
CARP	Cache Array Routing Protocol
CMDP	Control Messages Delivery Protocol
CP	Control Processor
EF	Expedited Forwarding
F–FSNDP-FF	Fractal-Fractal Shot Noise Driven Poisson Point Process-Fractal Filter
F-FSNDP-EF	Fractal-Fractal Shot Noise Driven Poisson Point Process-Exponential Filter
FE	Forwarding Engine
FEC	Forward Error Control
GNT	Grant

H-FSNDP	Homogeneous-Fractal Shot Noise Driven Poisson Point Process
HRW	Highest Random Weight
IP	Internet Protocol
ISP	Internet Service Provider
LLP	Least Loaded Path
MAN	Metropolitan Area Network
MDP	Markov Decision Process
MPLS	Multi-Protocol Label Switching
MTP2	Message Transport Part 2
NPU	Network Processing Unit
OSPF	Open Shortest Path First
PDP	Protocol Data Packet
QoS	Quality Of Service
REQ	Request
SP	Strict Priority
SS7	Signaling System 7
SSCOP	Service Specific Connection Oriented Protocol
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
VLB	Valiant Load Balancing
VOQ	Virtual Output Queue
WAN	Wide Area Network

WDM Wavelength Division Multiplexing

Chapter 1

Introduction

In this chapter, we briefly describe the problem of deploying the AAPN network as the backbone network of an Internet Service Provider (ISP). We also present the motivation and application of the proposed deployment. After that, we list the objectives and contributions of this research. Finally, an outline section is added to describe the structure of this thesis.

1.1 Problem Statement

ISPs are facing great challenges in provisioning their network resources due to the rapid growth in the number of Internet users and the complexity of its traffic patterns. Many emerging applications like voice over IP, peer-to-peer, and video on demand are characterized with highly variable traffic that is very difficult to predict and estimate. Currently, ISPs handle this variability in traffic by overprovisioning their backbone

networks capacity with typical utilizations of around 20% [1].

Research has been conducted recently to design high-speed, high-bandwidth, inexpensive backbone networks that can handle the rapid growth in Internet users and their high traffic volume; the Agile All-Photonic Network (AAPN) is an example of such networks [2]-[4]. The network is designed to provide the simplicity and flexibility ISPs are looking for to better utilize their network resources. It has an overlaid star topology, that provides simplified routing and protection. It also provides the agility needed to handle variable traffic patterns as advances in the photonic switching technology made the design of all-photonic devices with switching latency in the sub-microseconds feasible.

This thesis focuses on studying the problem of deploying AAPN as the backbone network for ISPs where IP routers are interconnected by the AAPN network. More specifically, it focuses on designing a routing architecture for balancing the IP traffic load over the AAPN network while minimizing packet reordering within one flow identified by common fields within the packet header. Since the number of flows is large in the backbone networks, storing flow state in lookup tables can limit the network performance. Hence the routing architecture should eliminate the need for flow lookup tables. Static and adaptive load balancing methods are to be studied and their behavior and effect on the network performance in terms of packet drop and flow remapping are to be analyzed.

To provide a reliable channel between the edge and core nodes in the AAPN network, the Control Messages Delivery Protocol (CMDP) is proposed as part of this

thesis. The protocol is designed to work in environments where propagation delays are long and/or the error rates are high. It is used to deliver a burst of short messages in sequence and with no errors. Combined with the reliable routing protocol proposed previously for the AAPN network [5], they form the control plane for the network that supports the operation of the load balancing methods.

To extend the applicability of the load balancing methods to topologies beyond AAPN overlaid star topology, the Valiant Load Balancing (VLB) method [6], [7] is used to build an overlaid star topology on top of the physical topology. Deploying the VLB method in the AAPN network, eliminates signaling and replaces the dynamic core schedulers with a static scheduler that can accommodate all traffic matrices within the hose traffic model boundaries [8].

1.2 Motivation and Application

This thesis work is motivated by the extensive research in the areas of agility in all-photonic networks and load balancing of Internet traffic over the ISP backbone networks. The thesis should provide interested ISPs with a comprehensive study of load balancing methods for deploying the AAPN network as their backbone network. It should also support the deployment of the AAPN network at the core of an IP network using the overlay Internetworking model, where control in the AAPN network is hidden from control in the IP network.

1.3 List of Objectives and Contributions

The main objectives and contributions of this thesis are:

1. A distributed routing architecture that balances the IP traffic in the AAPN network. An instance of this architecture is associated with every source-destination edge node pair in the network.

2. A Control Messages Delivery Protocol (CMDP) that is used to reliably deliver signaling messages in the AAPN network.

3. A static load balancing method that is stateless, resilient, and preserves packets ordering.

4. Adaptive load balancing methods that improve on the static method by reducing the packet dropping probability:

   (a) A variation that adapts the balancing weights based on the traffic load on the downstream links from the source edge node to the core nodes.

   (b) A variation that adapts the balancing weights based on the traffic load on the upstream links from the core nodes to the destination edge node.

   (c) A variation that adapts the balancing weights based on the traffic load on both the downstream and upstream links.

5. An adaptive load balancing method that reduces the number of flows remapped while keeping the AAPN network balanced.

6. An adaptive load balancing method that supports two classes of traffic: the Expedited Forwarding (EF) class that requires low jitter and packet drop rate, and the Best Effort (BE) class.

7. A distributed routing architecture that balances the IP traffic in the VLB network. This architecture extends the applicability of the load balancing methods to networks with general topologies.

1.4 Thesis Outline

The thesis is organized as follows. Chapter 2 presents the previous fundamental work on the fields of AAPN network and load balancing for backbone networks. It then presents the adaptive Highest Random Weight (adaptive HRW) method on which the AAPN routing architecture is based.

Chapter 3 describes the Control Messages Delivery Protocol (CMDP). The chapter starts with the motivation and usage of this protocol. It then describes its features and design. It also demonstrates the CMDP protocol deployment in the AAPN network through a number of examples. The examples show the operation of the protocol under different scenarios, each illustrates a specific aspect of the protocol operation. The examples cover the reliable transport of the two AAPN signaling messages, namely: the AAPN Request message (REQ) and the AAPN Grant message (GNT). The chapter concludes with simulation analysis of the protocol behavior when deployed in MAN and WAN environments.

Chapter 4 describes the load balancing routing architecture for the AAPN network. The chapter starts with the literature review for routing in the AAPN network. It then describes the schematic diagram of the architecture with the different mapping levels used. It later describes the different static and adaptive variations of the architecture. The chapter concludes with simulation analysis of the architectures in WAN environment.

Chapter 5 describes the load balancing QoS routing architecture for the AAPN network. The chapter starts with a description of the schematic diagram of the architecture. The architecture is based on the static and adaptive architectures described in chapter 4. The chapter concludes with simulation analysis of architecture in WAN environment. The analysis uses traffic that belongs to two DiffServ traffic classes, namely: Expedited Forwarding (EF) that is sensitive to variations in end-to-end delay and traffic drop rate, and Best Effort (BE) that can tolerate variations in end-to-end delay.

Chapter 6 extends the application of the routing architecture to the general mesh topology. The idea is to create a two-hop virtual topology similar to the overlaid star topology on top of the physical network. The Valiant Load Balancing (VLB) method is used to create the virtual topology.

Chapter 7 concludes the thesis and lists the future work.

Appendix A deploys the minimal flow remapping scheme described in chapter 4 in the Internet router architecture from [38]. The objective is to show that load balancing methods proposed in this thesis are also applicable to the router architecture from

[38]. Deploying the QoS architecture described in chapter 5 in the router architecture is left for future work. The scheme identifies, for an underutilized (overutilized) network processor, the input ports causing the network processor to be underutilized (overutilized). The objective is to further reduce the number of flows remapped while keeping the packet processing workload balanced among the different processing units within the router.

Chapter 2

Background

2.1 Introduction

In this chapter we review the previous fundamental work in fields related to this thesis work. First, we review the up to date research on the Agile All-Photonic Network (AAPN) proposed recently as an agile, high capacity core network. Second, we review the related research in the area of load balancing, more specifically in the areas of load balancing for backbone networks design and hashing and load balancing. We end this chapter by presenting the adaptive Highest Random Weight (HRW) algorithm used to design load balanced Internet routers.

2.2 The Agile All-Photonic Network

The emergence of WDM systems [9] has dramatically increased the raw capacity of optical links. Using WDM systems, one fibre can carry a number of optical wavelengths, each with a capacity of up to 40 Giga bits per second. With time, new optical devices were developed that are capable of performing networking functions such as optical channel switching and optical channel add/drop [10]. This enabled photonic networks (a cloud of optical devices) to establish an end-to-end light paths, that are transparently able to carry different types of payloads at different bit rates. This eliminated the costly optical-electronic-optical conversions and header processing at the core nodes.

Early photonic networks have a modularity problem as light paths, established by concatenating wavelengths, have a fixed bandwidth capacity in the order of few Giga bits per second. To solve this problem, research toward multiplexing on the wavelengths has been conducted. Two types of multiplexing methods are studied: statistical multiplexing and synchronous multiplexing. Statistical multiplexing has two forms: burst multiplexing [11] and packet multiplexing [12]. Burst multiplexing allows bursts (can be thought of as containers that can carry variable number of packets) of different sizes from different sources to statistically share the wavelength. Packet multiplexing allows packets of different sizes from different sources to statistically share a wavelength. Synchronous multiplexing uses time division multiplexing technique to share the wavelength between the different sources. Each source is given

a fixed length timeslot to transmit its data over the shared wavelength. The Agile All-Photonic Network (AAPN) [2]-[4] is an example of a network that uses synchronous time division multiplexing to better utilize the bandwidth of WDM systems. AAPN research originates from the research into the PetaWeb network [13]-[16].

AAPN uses agile all-photonic switches as advances in the photonic switching technology made the design of all-photonic devices with switching latency in the sub-microseconds feasible. It proposes a simplified overlaid star network architecture that can be deployed as a MAN or WAN network. This overlaid architecture, as opposed to general mesh architecture, scales network capacity to multiples of Tera bits per second, simplifies routing, increases reliability, eliminates wavelength conversion and the need for accurate traffic engineering [16]. With careful topological design, AAPN can achieve the wide geographical area coverage needed to be deployed as an Internet backbone transport network [4]. The use of all-photonic switches at the core introduces new network control challenges as photonic buffers are proven to be infeasible at least in the near future.

AAPN has an overlaid star topology, where each edge node is connected to a number of photonic core nodes, see figure 2.1. AAPN core node operates at the timeslot modularity where bandwidth allocated for a source-destination edge node pair is a multiple of timeslots.

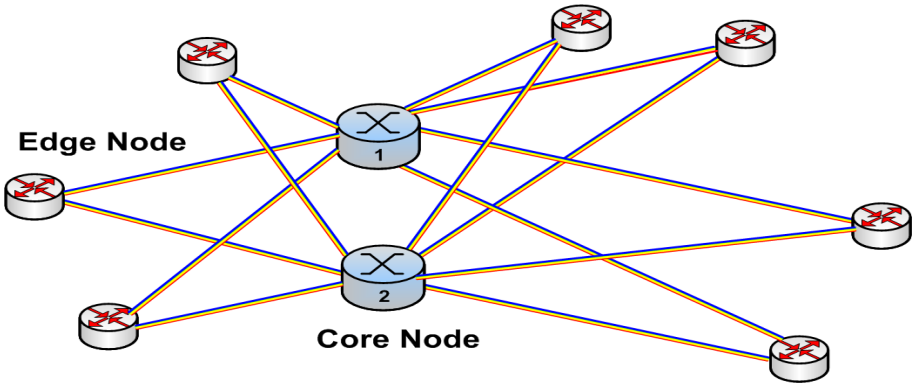


Figure 2.1: The AAPN overlaid architecture.

2.2.1 The Edge Node Architecture

The main function of an edge node in an AAPN network is to perform traffic aggregation, see figure 2.2. IP Packet flows are terminated at the AAPN edge nodes and are put into a unified data format before transmission. The data units are then sent to a Virtual Output Queue (VOQ) that is associated with a destination edge node and is served by an outgoing link to the core node. Data units from the VOQs are then aggregated and packaged into timeslots and sent to the core node. The timeslot takes 10 μs to transmit and is the unit into which time intervals are sliced. At the receipt of a timeslot, the destination edge node reassembles the constituent packet flows, releasing them into their respective IP routers.

The AAPN network must have the ability to identify and aggregate multiple IP packet flows into a timeslot. Regardless of the routing scheme used, consecutive timeslots carrying traffic that belongs to the same flows can be routed through the same core node or independently through different core nodes. The later case requires

re-sequencing buffers to be implemented at the destination edge node to reconstruct the original IP packet flows.

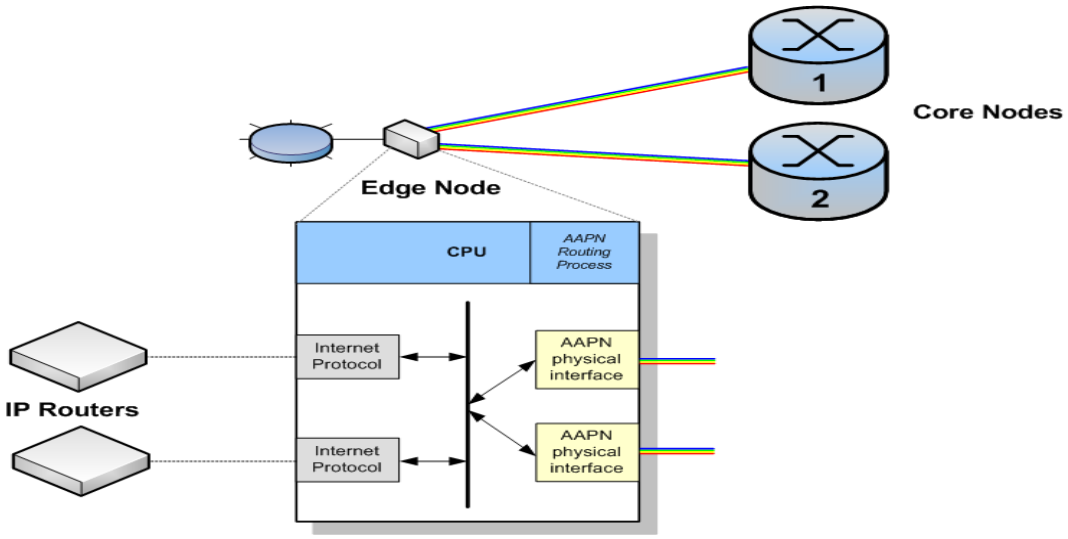


Figure 2.2: The AAPN edge node architecture.

2.2.2 The Core Node Architecture

AAPN is an agile self-configuring network, in which a core photonic node connects high capacity, fast electro-photonic edge nodes. An AAPN core node operates at the timeslot modularity where the bandwidth allocated for a given source-destination edge node pair, at any time, is an integer multiple of timeslots. The core node in AAPN is a layered space switch, see Figure 2.3. Each layer is used to switch timeslots on a specific wavelength. The number of layers is equal to the number of

input wavelengths coming from each edge node. No wavelength conversion is used in AAPN; an incoming timeslot on an incoming wavelength on an incoming port is switched, with no conversion, to the same timeslot position on the same outgoing wavelength on an outgoing port.

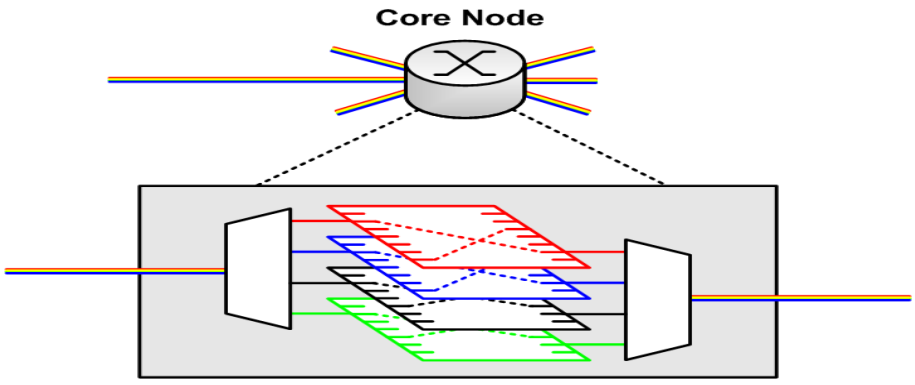


Figure 2.3: The AAPN layered core node architecture.

2.2.3 AAPN Core Scheduling Algorithms

Scheduling algorithms for the AAPN network can be divided into three categories: frame-by-frame, slot-by-slot and burst scheduling. In frame-by-frame scheduling [17]-[20], the core configuration is updated every frame to follow the traffic changes at the edge nodes. A frame is a fixed number of contiguous timeslots. The traffic demands of an edge node are signaled to the core nodes at the beginning of every frame [21]. With the frame-by-frame scheduler running at the core node, an AAPN core configuration

can follow traffic demand changes at the network edges as fast as a frame interval.

In slot-by-slot scheduling [22]-[23], the core configuration is updated every timeslot to follow the traffic changes at the edges. The traffic demands of an edge node are signaled to the core at the beginning of every timeslot [21]. With the slot-by-slot scheduler running at the core node, an AAPN core configuration can follow traffic demand changes at the network edges as fast as a timeslot interval.

In burst scheduling [24], the core configuration is updated continuously to follow the traffic changes at the edges. Burst allocation requests are signaled to the core when needed [21]. Such a signal or request informs the core scheduler of the burst length and the burst destination.

2.3 Load Balancing

Load balancing is a technique for distributing the workload evenly over two or more processors, network paths, or other resources. Load balancing can serve different objectives like increasing throughout, resource utilization, reliability, and reducing response time. For a general survey on load balancing, see [25]. Load balancing techniques can be either static or dynamic [26]. Static load balancing schemes require low overhead as the system state is known in advance but can not adapt to fluctuations in system state. Dynamic load balancing schemes, on the other hand, require changes in system state before making the decisions.

2.3.1 Backbone Network Design and Load Balancing

Recently, researchers have been investigating the design of load balanced backbone networks that can accommodate highly variable traffic matrices. They have proposed load balancing network designs that can provide guaranteed performance for highly variable traffic matrices [6], [7]. The research is based on deploying the Valiant Load Balancing (VLB) as the method for routing the traffic over the network. The VLB method was first proposed by Valiant for processor interconnection networks [27] and was used for designing scalable load balanced routers with performance guarantees [28]-[30]. The method routes traffic over the network twice: first the traffic entering the network is balanced over a set of intermediate nodes and then it is sent to the final destination.

The research is separately conducted by groups from Stanford university and Bell Labs. The network design from Stanford described in [6], [31] assumes a logical mesh topology for the backbone network, and routes traffic using the VLB method over the full mesh. This allows the network to efficiently accommodate highly variable traffic with no dynamic reconfiguration of the network resources. The two phase routing method from Bell Labs described in [7], generalizes the VLB method to general network topologies by using arbitrary load balancing on the intermediate nodes. They provide different linear programming formulations of the problem for different objectives [32]-[34].

The VLB load balancing method has many advantages: It handles unpredictable

traffic that falls within the constraints of the hose traffic model for the network. Moreover, it avoids any need for dynamic reconfiguration of the network to accommodate changes in the traffic matrix. Routing is oblivious and independent of the specific traffic matrix. Finally, it is efficient in the sense that it has minimal total capacity provisioned.

Keslassy et al. [29]-[30] considered the use of two phase load balancing for designing the interconnect inside the load balanced switch. Keslassy's work proved that using two phase load balancing on a uniform mesh gives the highest throughput for a given interconnect capacity.

2.3.2 Hashing and Load Balancing

Hashing has been used extensively in networking domain for workload distribution. [35] describes a hash-based scheme for web caches, called Highest Random Weight (HRW), that can achieve high web cache hit rate, load balancing, and minimum disruption in case of server failure or reconfiguration. The HRW has been extended in [36] to accommodate heterogeneous servers.

Hashing is widely used in designing multi-processor routers because, compared to round robin and minimum-load mappings, it preserves the packet order of individual TCP flows. Hashing distributes the incoming packet processing load over the different processers inside the router based on one or more fields within the incoming IP packet header. Studies in [37] has shown that static hashing, even though preserves packet ordering, could result in load imbalance. [37] reduces imbalance in load distribution by

dynamically re-mapping large flows to different processors. [38] proposes an adaptive load balancing method that extends the work from [36] by adding an adaptive control loop to dynamically map incoming flows to processers while preserving the minimum disruption property, its called the adaptive HRW method. [39] extends the work in [37] by re-mapping bursts inside the flows, this is possible due to the bursty nature of Internet traffic.

Hashing is widely used for Internet traffic balancing over multiple links [40]. In [41], the performance of various static hashing methods and one adaptive method for balancing traffic over multiple links is evaluated. [42] measures the flow volumes periodically to detect high volume flows and moves them to lightly loaded links. [43] exploits the burstiness of Internet traffic to balance load bursts within a flow over the multiple links.

2.4 The Adaptive HRW algorithm

The adaptive HRW algorithm, described in [38], balances the incoming processing load among the different Network Processor Units (NPUs) in the router. It dynamically maps incoming packet flows to NPUs using a weighted hash function while preserving the minimal disruption property in [35]. The mapping is done using the Highest Random Weight (HRW) [35]-[36] hash function over the flow identifier $\vec{v}$. Periodically, the workload intensity of each NPU is measured and forwarded to a central processor (CP). The CP determines the system state (underutilized or overutilized)

and the mapping weights are adjusted accordingly.

2.4.1 HRW Packet-to-Processor Mapping

Mapping of packets with a flow identifier vector $\vec{v}$ is computed as follows:

$$\begin{aligned} & f(\vec{v}) = j \\ & \Leftrightarrow \\ & x_j g(\vec{v}, j) = \max_{k \in \{1...m\}} x_k g(\vec{v}, k) \end{aligned} \tag{2.1}$$

where $x_j \in R+$ is the weight assigned to NPU j and $j \in \{1...m\}$, m is the number of NPUs, $g(\vec{v}, j)$ is a pseudorandom function with $g : \vec{v} \times \{1...m\} \rightarrow (0, 1)$ (i.e., $g(\vec{v}, j)$ is assumed to be a random variable in $(0, 1)$ with uniform distribution). The weights $\vec{x} = (x_1...x_m)$ determine the fraction of the identifier vector space assigned to each NPU and has a 1-to-1 correspondence with the partition vector $\vec{p}$ computed using theorem 1 [36].

Theorem 1 (Ross) Let $p_1...p_m$ be given target probabilities. Reorder the destinations so that $p_1 \leq ... \leq p_m$. Let

$$x_1 = (mp_1)^{\frac{1}{m}} \tag{2.2}$$

and let $(x_2...x_m)$ be calculated recursively as follows:

$$x_n = [\frac{(m-n-1)(p_n - p_{n-1})}{\prod_{i=1}^{n} x_i} + x_{n-1}^{m-n-1}]^{\frac{1}{m-n-1}} \tag{2.3}$$

Then the HRW mapping algorithm with multipliers $(x_1...x_m)$ will map the fraction p_n of incoming objects to the nth destination, $n = 1, ..., m$.

The HRW mapping possesses the following significant advantages over other hash-based load balancing schemes:

Load balancing The mapping provides load balancing over the request object space, even for the heterogeneous case. It allows to split the hashed objects into hash buckets of arbitrary size, as determined by predefined weights.

Minimal disruption In case of a processor failure, removal or addition, the number of request objects that are re-mapped to another destination is minimal.

2.4.2 Adaptive HRW Triggering Policy

The triggering policy exploits two dynamic thresholds to determine whether an NPU is underutilized or overutilized. To evaluate the status of individual NPUs, periodic NPU workload intensity $\rho_j(t)$ is measured and filtered every ΔT:

$$\overline{\rho}_j(t) = \frac{1}{r}\rho_j(t) + \frac{r-1}{r}\overline{\rho}_j(t - \Delta T) \tag{2.4}$$

where r is an integer constant.

A similar measure is used for the total system workload intensity:

$$\overline{\rho}(t) = \frac{1}{r}\rho(t) + \frac{r-1}{r}\overline{\rho}(t - \Delta T) \tag{2.5}$$

Triggering policy uses the dynamic threshold:

$$\varepsilon_\rho(t) = \frac{1}{2}\left(1 + \overline{\rho}(t)\right) \tag{2.6}$$

and a fixed hysteresis bound $\varepsilon_h > 0$, to prevent adaptation within the following interval:

$$\left[\left(1 - \varepsilon_h\right)\overline{\rho}(t), \left(1 + \varepsilon_h\right)\overline{\rho}(t)\right] \tag{2.7}$$

ε_h is set to a value close to 0, for example 0.01, thus preventing adaptation when the load stays within 1% of the total system workload intensity.

2.4.3 Adaptive HRW Adaptation Policy

The adaptation policy preserves the minimal disruption property by having a subset of the elements in the weight vector $\vec{x}$ multiplied by a constant factor α. For values of $\alpha < 1$, the subset of elements multiplied will have their load partition decreased while the rest of elements will have their load partition increased. For values of $\alpha > 1$, the subset of elements multiplied will have their load partition increased while the rest of elements will have their load partition decreased. The following equation is used to compute α when the system is underutilized (i.e., $\overline{\rho}(t) \leq 1$) and one or more NPUs are overutilized (i.e., $\overline{\rho}_j(t) > \varepsilon_\rho(t)$ [38]:

$$\alpha(t) = \left(\frac{\varepsilon_\rho(t)}{\min\left\{\overline{\rho}_{\mathrm{j}}(t), \overline{\rho}_{\mathrm{j}}(t) > \varepsilon_\rho(t)\right\}}\right)^{1/m} \tag{2.8}$$

Conversely, α is computed in a symmetrical way when the system is overutilized (i.e., $\overline{\rho}(t) > 1$):

$$\alpha(t) = \left(\frac{\varepsilon_\rho(t)}{\max\left\{\overline{\rho}_{\mathrm{j}}(t), \overline{\rho}_{\mathrm{j}}(t) < \varepsilon_\rho(t)\right\}} \right)^{1/m} \tag{2.9}$$

2.5 Summary

In this chapter we reviewed the previous work in fields related to this thesis work. First, we review the research on the Agile All-Photonic Network (AAPN) proposed recently as a high capacity core network. Second, we review the related research in the area of load balancing, more specifically in the areas of load balancing for backbone networks design and hashing and load balancing. The chapter ended by presenting the adaptive Highest Random Weight (HRW) algorithm used to design load balanced Internet routers.

Chapter 3

Control Messages Delivery Protocol

3.1 Introduction

This chapter describes the Control Messages Delivery Protocol (CMDP) that supports the reliable delivery of short control messages in a high delay and/or high error rate environment. The protocol uses a hybrid FEC/ARQ mechanism to recover from damaged (or lost) messages. It dynamically adapts its Forward Error Correction (FEC) capabilities to the channel error rate. Its FEC mechanism automatically recovers damaged messages before resorting to the Automatic Repeat reQuest (ARQ) mechanism. Its FEC mechanism minimizes the round trip delay by reducing the number of messages that need to be retransmitted using the ARQ mechanism.

3.1.1 Motivation for Reliable Signaling in AAPN

Traffic arriving at an AAPN edge node is classified and stored in the appropriate VOQ. A request for timeslot resources is then constructed and sent to the core node. The core node collects and classifies all requests received from the edge nodes and runs its scheduling algorithm to generate a new configuration of the core switches. Grants corresponding to the new configuration are constructed and sent to their corresponding edge nodes. Since requests and grants messages correspond to data awaiting service at the edge nodes, damaged (or lost) requests or grants result in VOQ buffers building up. As VOQ buffers build up data awaiting service starts to experience longer buffering delays. To prevent long buffering delays, an efficient delivery protocol is needed to provide for reliable delivery of the requests and grants messages.

3.2 Literature Review

In the literature, many reliable link and transport layer protocols have been designed and implemented for high-speed environments. Doeringer et al. [44] provides a comparative survey of eight transport protocols that depend on conventional ARQ mechanisms to recover from damaged messages. Signaling System 7 (SS7), designed for the telephone network, uses a link layer protocol called Message Transport Part 2 (MTP2) that depends on Go-Back-N ARQ mechanism. ATM control plane, designed for the B-ISDN network, uses a link layer protocol called Service Specific Connection

Oriented Protocol (SSCOP) [45] that depends on Selective-Repeat ARQ mechanism. Even though these protocols can be used in an AAPN environment, most of them are heavily dependent on the conventional ARQ mechanisms.

As conventional ARQ mechanisms are time costly when used in AAPN's long delay WAN environment, a reliable delivery protocol that is based on a hybrid FEC/ARQ mechanism is being proposed in this work. Although the FEC/ARQ concept is not new, this work is different from existing FEC/ARQ mechanisms [46]-[49]. These works focus mainly on optimizing throughput and/or power in wireless transmissions, our work, on the other hand, focuses more on minimizing the end to end delay especially for links with long propagation delays.

We propose a light weight FEC mechanism to reduce processing at the receiver side, this is important for systems with a receiver side that interacts with large number of senders at the same time. For example, in the AAPN network the core node acts as a receiver of bandwidth requests from all source edge nodes. The proposed protocol is based on the SSCOP protocol that targets large bandwidth delay product environments.

3.3 The Control Messages Delivery Protocol

3.3.1 CMDP Protocol Usage and Applicability

The CMDP protocol is mainly designed to work in environments where propagation delays are long and/or the error rates are high. It is used to deliver a burst of

short messages in sequence and with no errors. Small messages are mainly control messages, hence the name Control Messages Delivery Protocol (CMDP). The protocol is originally designed for the AAPN network where propagation delays between the edge nodes and the core nodes are significant and any retransmission could result in long buffering latency at the edge nodes.

The protocol's applicability can be extended beyond the AAPN networks to remote control in wireless environments with high noise rates and/or long propagation delay characteristics. For example, the protocol can be used in satellite communications to reliably deliver a sequence of commands to control the movement and operation of a remote satellite in real time. Another possible application is the remote control of robots, on earth or in outer space, by uploading the robot with a sequence of instructions to program and control the movement and operation of its devices.

3.3.2 CMDP Protocol Features

The protocol is a link layer protocol that builds on an environment that delivers messages with possible errors and gaps. Its main responsibility is the detection, report, and correction of damaged messages. The protocol uses a hybrid FEC/ARQ mechanism to recover from damaged messages. The FEC mechanism allows for self-recovering from damaged messages. For situations where the FEC mechanism fails to recover from severe errors, a Selective-Repeat ARQ mechanism takes over and triggers retransmissions of undelivered messages.

The protocol FEC error control mechanism is designed for AAPN's high speed

environment. Some transmission bandwidth is sacrificed to minimize processing at the receiver side. It also optimizes operation for the normal error free communication. This is important because AAPN core nodes receive requests for bandwidth from all edge nodes in the network, which makes processing minimization at the core node very crucial. The protocol adapts its FEC capabilities to the environment's error rate. The sender and receiver sides of the protocol periodically negotiate their FEC capabilities by the periodic exchange of CMDP control packets.

3.3.3 CMDP Protocol Packets

The CMDP has three packets to support the reliable transfer of control messages: the Protocol Data Packet (PDP), for delivering the control messages, and the POLL and STAT packets for controlling the CMDP PDP packet flow.

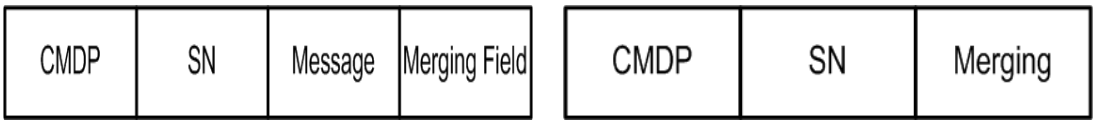
The PDP packet extends the control message (CM), with a sequence number (SN) and a variable size merging field, see figure 3.1a. The two redundant fields support the protocol's FEC/ARQ capabilities. The merging field consists of the last (m) control messages generated by the sender side. The CMDP field identifies the protocol instance the PDP packet belongs to.

The protocol control flow uses a sender-dependent strategy that lets the sender instruct the receiver to generate a complete state report by sending a POLL packet. This is desirable in an AAPN environment with different propagation delays associated with different edge nodes, since it eliminates the need for timers tuned to the round trip propagation delay. POLL packets are sent periodically by the sender,

and the receiver responds with a STAT packet that represents the complete state of the receiver. The time period (t) between consecutive POLL packets is a protocol parameter; its value varies with the channel error rate and is controlled by the sender.

The POLL packet has three fields: one field identifies the CMDP instance the POLL packet belongs to, the second field contains the sequence number of the next PDP packet to be sent, and the third field contains the merging parameter value (m) used at the sender, see figure 3.1b.

The STAT packet has three fields: one field identifies the CMDP instance the STAT packet belongs to, the second field contains the merging parameter value (m) used at the receiver, and the third field is variable in length and contains a list of all currently outstanding PDP packets, see figure 3.1c. The receiver knows which PDP packets are outstanding by examining gaps in the received PDP packets and the PDP sequence number contained in the POLL packet. The protocol POLL/STAT packet exchange is resilient to damage or loss since each STAT packet contains the complete state of the receiver.



(a) The Protocol Data Packet.

(b) The POLL Packet.

(c) The STAT Packet.

Figure 3.1: The CMDP protocol packets.

The protocol has two phases of operation, a handshake phase and a transfer phase. The protocol begins with the handshake phase during which the sender and receiver exchange POLL/STAT packets to initialize the protocol merging parameter (m). After the handshake phase is complete, the sender and receiver are all setup to start the transfer phase. During the transfer phase, control messages received by the CMDP sender are encapsulated in PDP packets and sent to the CMDP receiver. At the receiver, PDP packets are checked for in-sequence delivery. In-sequence PDP packets are considered error free and the control message it carries is extracted and passed up the hierarchy. POLL packets trigger the ARQ error control mechanism to recover those damaged PDP packets.

3.3.4 CMDP Protocol FEC and ARQ Mechanisms

The FEC mechanism merges the previously sent control message(s) (or part of it) with the current control message to form the CMDP PDP packet. The number of messages merged is a protocol parameter, called merging parameter (m), and its value varies with the channel's error rate. Figure 3.2 shows an example of a PDP packet with merging parameter set to 1. This PDP packet is capable of recovering the previous control message if the previous PDP packet is damaged.

For the AAPN network with fiber-optic channels, an initial value of 1 can be assigned to the merging parameter. As the channel's error rate increases due to reasons like interference, the protocol can negotiate higher values for the parameter to better recover from damaged PDP packets. A good indication of the variation in

the channels error rate is the variation in the number of PDP packets that escape the FEC mechanism and need to be recovered using the ARQ mechanism.

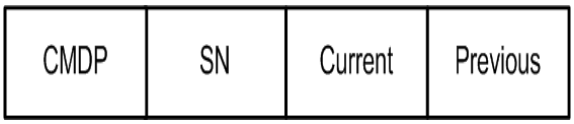


Figure 3.2: Protocol Data Packet (PDP) with m = 1.

The FEC mechanism is called only when PDP packets are damaged. Damaged PDP packets are detected when out-of-sequence PDP packets are received. In-sequence PDP packets are handled with minimum processing overhead as the merging field related to the FEC mechanism is discarded.

The ARQ mechanism used in the CMDP protocol is a Selective-Repeat mechanism. The sender sends PDP packets to the receiver, storing them for potential retransmission until acknowledged by the receiver. The PDP packets are numbered and the sender retransmits only those that have been explicitly requested by the receiver. The receiver requests explicit retransmission of those PDP packets that could not be recovered using the FEC mechanism.

PDP packets handled by the ARQ mechanism have FEC capabilities, where one retransmitted PDP packet can recover up to the number of control messages embedded in it, for example the PDP packet in figure 3.2 can recover two control messages at the same time. Such feature reduces the PDP packets that need to be retransmitted.

3.4 The CMDP Protocol Operation

The operation of the CMDP protocol is best understood using examples. First, we describe the AAPN signaling protocol control messages. Next, the CMDP protocol deployment in the AAPN network is demonstrated through a number of examples. The examples show the operation of the protocol under different scenarios, each illustrates a specific aspect of the protocol operation. The examples cover the reliable transport of the two AAPN signaling messages, namely: the AAPN Request message (REQ) and the AAPN Grant message (GNT).

3.4.1 AAPN Signaling Protocol Control Messages

An AAPN network can cover a wide range of geographical areas and can be deployed as a MAN or WAN network. As such, a signaling protocol is needed to facilitate the operation of the scheduler implemented at the core and edge nodes. The protocol has two control messages, a request (REQ) and a grant (GNT). The REQ message originates at an edge node and is destined to a core node, see figure 3.3a. It is associated with a given destination and mainly consists of two fields: one field specifies the destination edge node (D) to which timeslots are being requested, and the other field specifies the number of timeslots requested (R).

The GNT message originates at a core node and is destined to an edge node, see figure 3.3b. It is associated with a given timeslot on a given wavelength and mainly consists of three fields: one field specifies the timeslot scheduled (T), the second field

specifies the wavelength (W) on which the timeslot resides, and the last field specifies the destination edge node (D) to which the timeslot is allocated.

(a) The REQ message. (b) The GNT message.

Figure 3.3: The AAPN network signaling messages for timeslot resources.

3.4.2 Deployment for the AAPN REQ Message

The following conventions are used in the examples for the CMDP protocol version used to reliably deliver the REQ control messages from an edge node to a core node.

1. The sender is an edge node, the receiver is a core node, and the control message delivered is the REQ message.

2. Examples are associated with a destination edge node whose address identifies the CMDP instance (address is not shown in the CMDP packets).

3. POLL CMDP packet is represented as: **POLL (m, S)**.

4. STAT CMDP packet is represented as: **STAT (m, outstanding list)**.

5. PDP CMDP packet is represented as: **PDP (S, R, merging field)**.

3.4.2.1 The CMDP Handshake Phase

Figure 3.4 shows the time flow diagram for the handshake phase of the CMDP protocol. The diagram depicts error free operation of the protocol's startup phase. The merging field m = 1, which means that each PDP packet will carry its REQ control message and the (R) field of the previously sent REQ control message (i.e., the FEC mechanism will be able to recover one damaged or lost REQ control message). The handshake phase is repeated when a new value for m is needed. For example, if more messages are recovered using the ARQ mechanism, the merging field can be increased to try to reduce the number of retransmissions. On the other hand, if less messages are recovered using the ARQ mechanism, the merging field can be decreased to reduce the amount of bandwidth used by the protocol data packets.

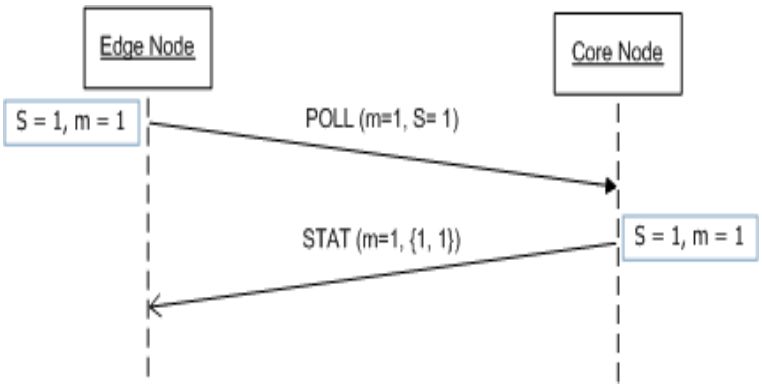


Figure 3.4: The CMDP Handshake Phase

3.4.2.2 The CMDP Error Free Operation for the REQ Message

Figure 3.5 shows the time flow diagram of the CMDP protocol operation in the error free case. The diagram shows an edge node signaling three REQ control messages (5, 2, and 8) to a core node. The CMDP protocol instance, associated with the intended destination edge node, generates three PDP packets. Each PDP packet carries its REQ control message and the (R) field of the previously sent REQ control message. The POLL packet sent by the edge node, informs the core node that the next PDP packet to be sent will have a sequence number equal to 4. The STAT packet, sent by the core node, informs the edge node that all PDP packets have been received error free.

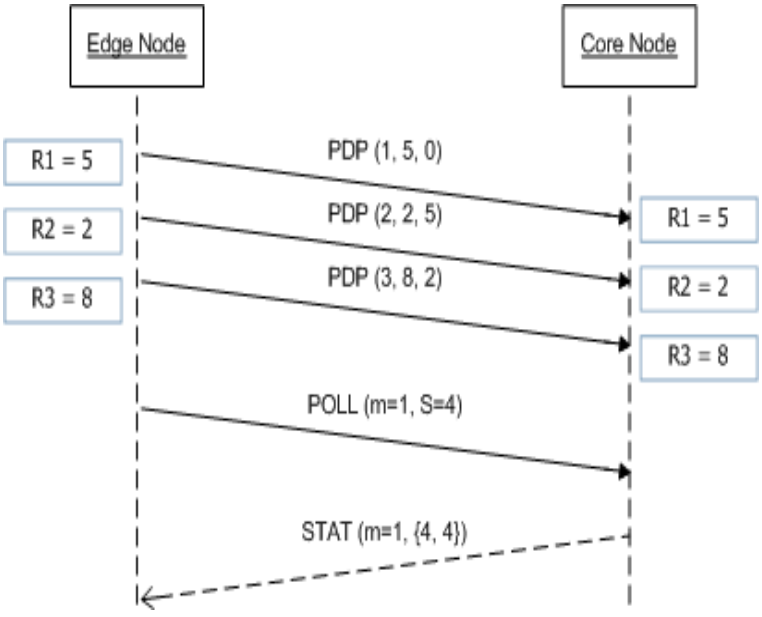


Figure 3.5: The CMDP error free operation for REQ messages.

3.4.2.3 The CMDP FEC Error Recovery for the REQ Message

Figure 3.6 shows the time flow diagram of the CMDP protocol operation in a faulty environment. The diagram shows an edge node signaling three REQ control messages (5, 2, and 8) to a core node. The CMDP protocol instance, associated with the intended destination edge node, generates three PDP packets. Each PDP packet carries its REQ control message and the (R) field of the previously sent REQ control message. The core node recovers the second PDP packet as soon as it receives PDP packet number 3. The POLL packet, sent by the edge node, informs the core node that the next PDP packet to be sent will have a sequence number equal to 4. The STAT packet, sent by the core node, informs the edge node that all PDP packets have been received error free.

3.4.2.4 The CMDP ARQ Error Recovery for the REQ Message

Figure 3.7 shows the time flow diagram of the CMDP protocol operation in a faulty environment. The diagram shows an edge node signaling three REQ control messages (5, 2, and 8) to a core node. The CMDP protocol instance, associated with the intended destination edge node, generates three PDP packets. Each PDP packet carries its REQ control message and the (R) field of the previously sent REQ control message. The POLL packet, sent by the edge node, informs the core node that the next PDP packet to be sent will have a sequence number equal to 4. Using the received POLL packet, the STAT packet, sent by the core node, informs the edge

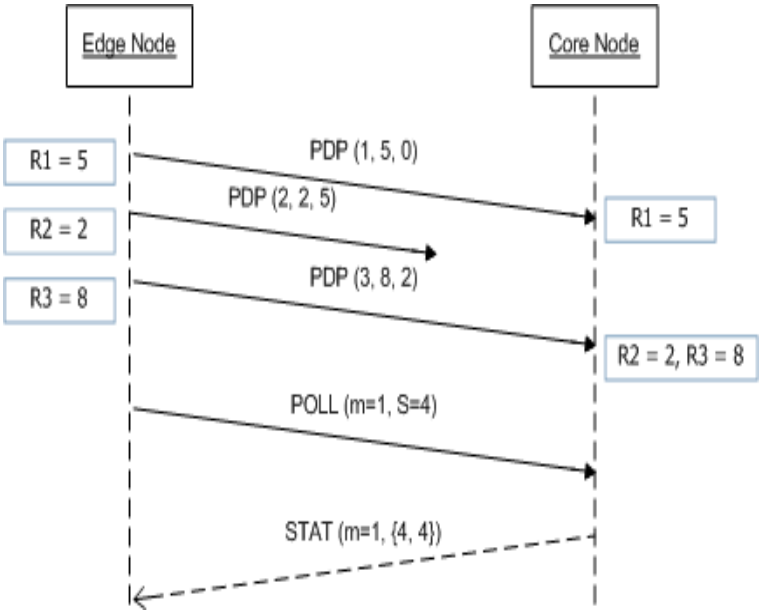


Figure 3.6: The CMDP FEC error recovery for REQ messages.

node that PDP packets two and three have been damaged or lost and need to be retransmitted (list {1, 3} in the STAT packet means PDP 1 has been received while PDPs 2 and 3 are lost).

3.4.3 Deployment for AAPN GNT Message

The following conventions are used in the examples for the CMDP protocol version used to reliably deliver the GNT control messages from a core node to an edge node.

1. The sender is a core node, the receiver is an edge node, and the control message delivered is the GNT message.

2. Examples are associated with a wavelength whose number identifies the CMDP instance (wavelength number is not shown in the CMDP packets).

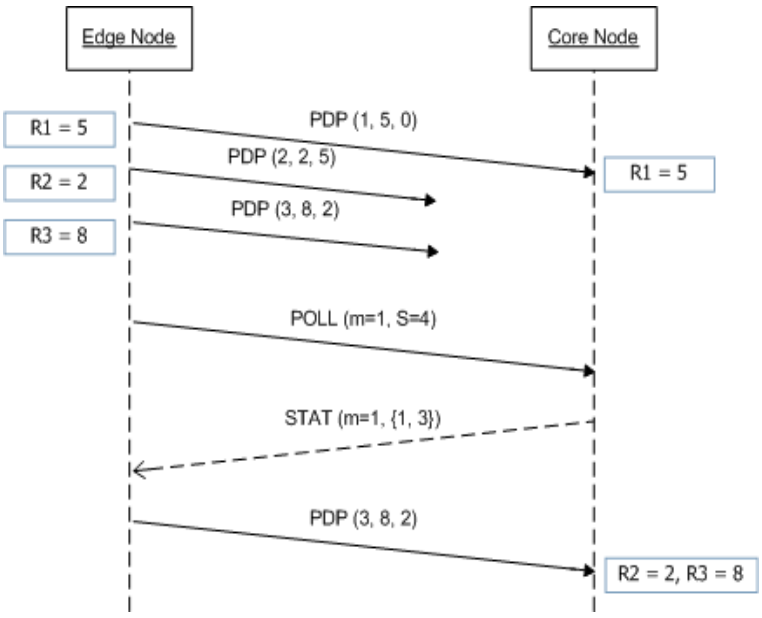


Figure 3.7: The CMDP ARQ error recovery for REQ messages.

3. Timeslots not assigned to the source edge node are given D = S in the PDP packet (i.e., the address of the source edge node), see example in figure 3.11.

4. POLL CMDP packet is represented as: **POLL (m, S)**

5. STAT CMDP packet is represented as: **STAT (m, outstanding list)**

6. PDP CMDP packet is represented as: **PDP (S, T, D, merging field)**

3.4.3.1 The CMDP Error Free Operation for the GNT Message

Figure 3.8 shows the time flow diagram of the CMDP protocol operation in the error free case. The diagram shows a core node signaling three GNT control messages (timeslot 15 to destination 7, timeslot 16 to destination 1, and timeslot 17 to destination 3) to an edge node. The CMDP protocol instance, associated with the

wavelength on which the timeslots reside, generates three PDP packets. Each PDP packet carries its GNT control message and the (D) field of the previously sent GNT control message. The POLL packet, sent by the core node, informs the edge node that the next PDP packet to be sent will have a sequence number equal to 4. The STAT packet, sent by the edge node, informs the core node that all PDP packets have been received error free.

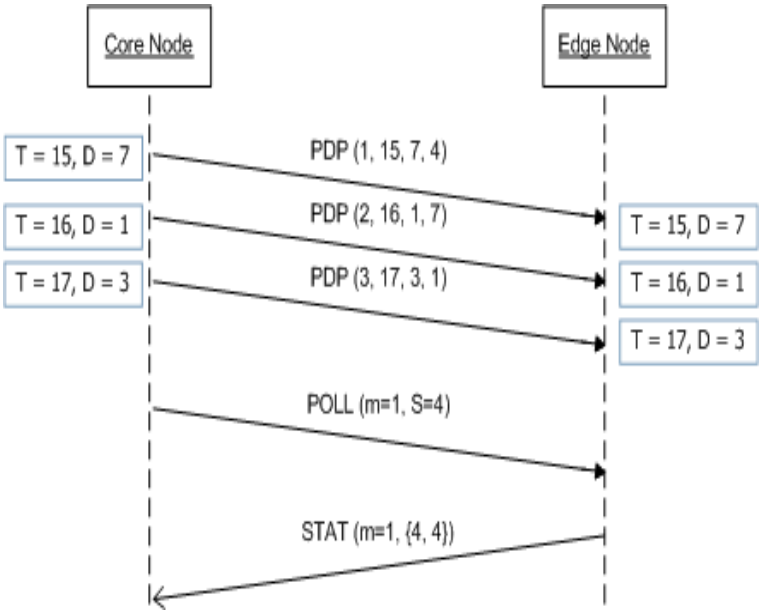


Figure 3.8: The CMDP error free operation for GNT messages.

3.4.3.2 The CMDP FEC Error Recovery for the GNT Message

Figure 3.9 shows the time flow diagram of the CMDP protocol operation in a faulty environment. The diagram shows a core node signaling three GNT control mes-

sages (timeslot 15 to destination 7, timeslot 16 to destination 1, and timeslot 17 to destination 3) to an edge node. The CMDP protocol instance, associated with the wavelength on which the timeslots reside, generates three PDP packets. Each PDP packet carries its GNT control message and the (D) field of the previously sent GNT control message. The edge node recovers the second PDP packet as soon as it receives PDP packet number 3. The POLL packet, sent by the core node, informs the edge node that the next PDP packet to be sent will have a sequence number equal to 4. The STAT packet, sent by the edge node, informs the core node that all PDP packets have been received error free.

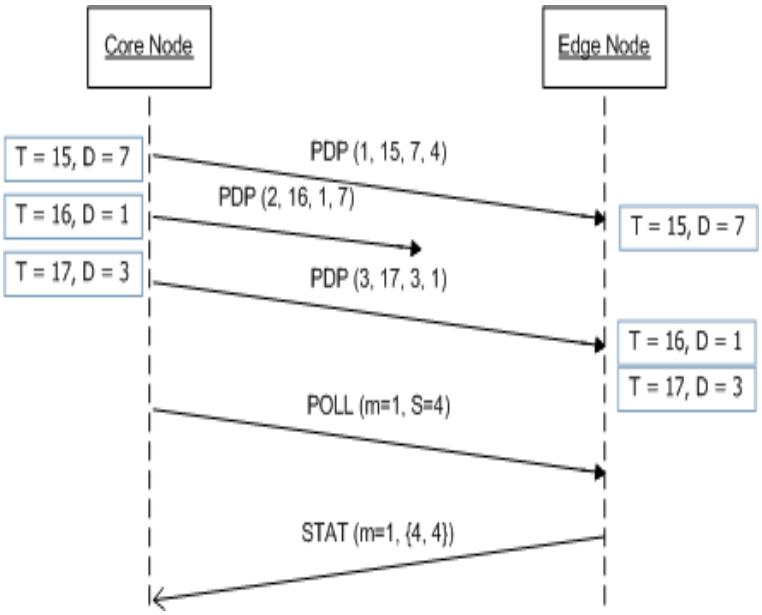


Figure 3.9: The CMDP FEC error recovery for GNT messages.

3.4.3.3 The CMDP ARQ Error Recovery for the GNT Message

Figure 3.10 shows the time flow diagram of the CMDP protocol operation in a faulty environment. The diagram shows a core node signaling three GNT control messages (timeslot 15 to destination 7, timeslot 16 to destination 1, and timeslot 17 to destination 3) to an edge node. The CMDP protocol instance, associated with the wavelength on which the timeslots reside, generates three PDP packets. Each PDP packet carries its GNT control message and the (D) field of the previously sent GNT control message. The POLL packet, sent by the core node, informs the edge node that the next PDP packet to be sent will have a sequence number equal to 4. Using the received POLL packet, the STAT packet, sent by the edge node, informs the core node that PDP packets two and three have been damaged or lost and need to be retransmitted (list {1, 3} in the STAT packet means PDP 1 has been received while PDPs 2 and 3 are lost).

3.4.3.4 The CMDP For Non-Consecutive GNT Messages

Figure 3.11 shows the time flow diagram of the CMDP protocol operation when non-consecutive timeslots need to be signaled to an edge node. The diagram shows a core node signaling two GNT control messages (timeslot 15 to destination 7, and timeslot 18 to destination 3) to an edge node. The CMDP protocol instance, associated with the wavelength on which the timeslots reside, generates three PDP packets. Each PDP packet carries its GNT control message and the (D) field of the previously sent

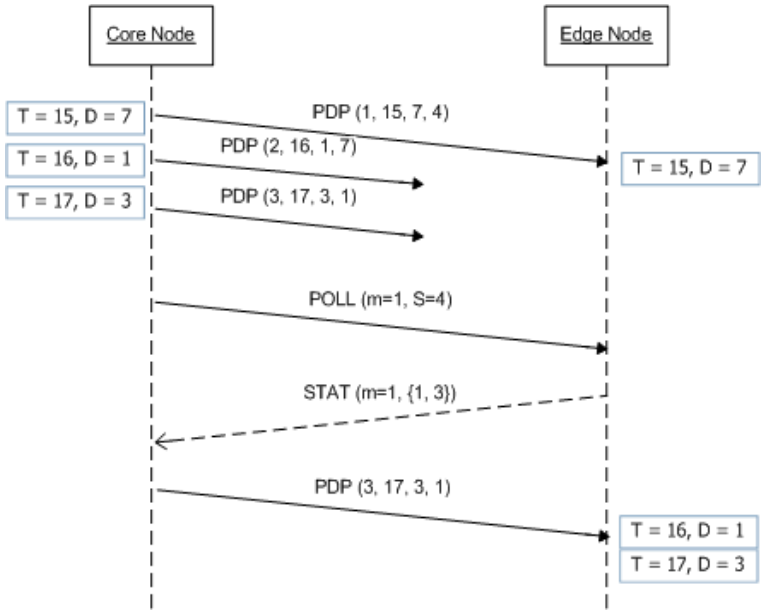


Figure 3.10: The CMDP ARQ error recovery for GNT messages.

GNT control message. Notice that PDP packet number 2 does not correspond to a real GNT control message (D=S), but is sent to allow for the recovery of PDP packet 1 in case it is damaged or lost. Notice also that the merging field for PDP packet number 3 is redundant as it recovers PDP packet number 2 which carries no real GNT control message (D=S). The POLL packet, sent by the core node, informs the edge node that the next PDP packet to be sent will have a sequence number equal to 4. The STAT packet, sent by the edge node, informs the core node that all PDP packets have been received error free.

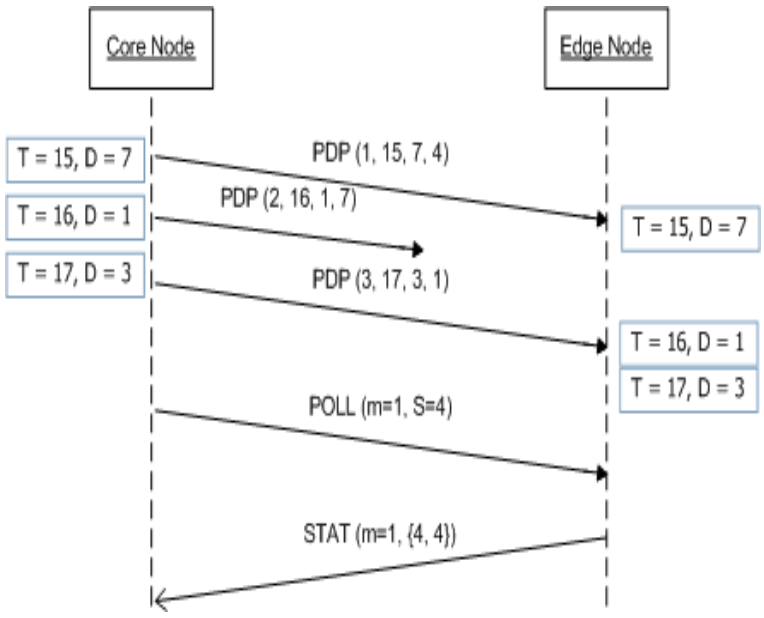


Figure 3.11: The CMDP for non-consecutive GNT messages.

3.5 Numerical Results

In this section, the CMDP protocol average delivery delay performance, for different values of the merging field (m), is studied in a simulated AAPN environment. The simulation model is implemented using the OPNET tool. It contains a CMDP protocol instance that is running between an edge node that acts as the CMDP sender and a core node that acts as the CMDP receiver. The protocol instance is used to reliably deliver REQ messages associated with a given destination edge node. The transmission channel generates random bit errors.

To reduce error check at the core node, a thin layer that multiplexes PDPs from different CMDP instances is used. PDP packets from different CMDP instances are multiplexed to reduce processing at the core. It is assumed that the new multiplexed

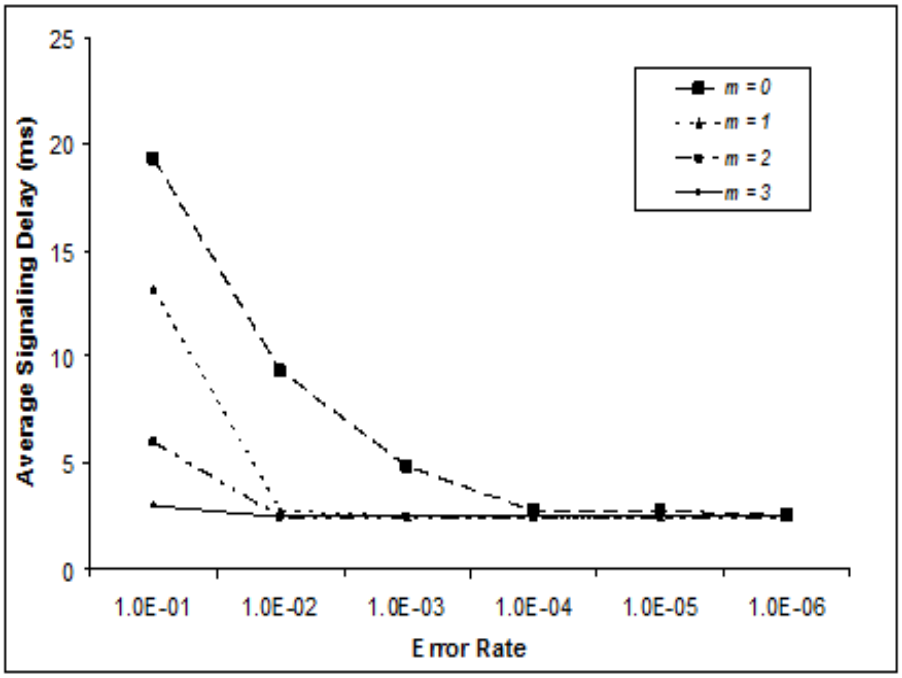
packet average size is equal to 1000 bits.

The protocol is evaluated in a transmission channel that produces random errors with bit error rates ranging from 10^{-4} (packet error rate of 10^{-1}) to 10^{-9} (packet error rate of 10^{-6}). The time period (t) between consecutive POLL packets is assumed to be fixed and is equal to the round trip propagation delay. It is assumed that a Slot-By-Slot scheduling algorithm is used at the core node, where traffic changes (i.e., REQ control messages) is signaled to the core node every timeslot (i.e., every 10 μs). Two scenarios, Metropolitan Area Network (MAN) and Wide Area Network (WAN), are studied. MAN is defined as an AAPN network with 500 km optical links $d = 2.5$ ms, while WAN is defined as an AAPN network with 8000 km optical links $d = 40$ ms.

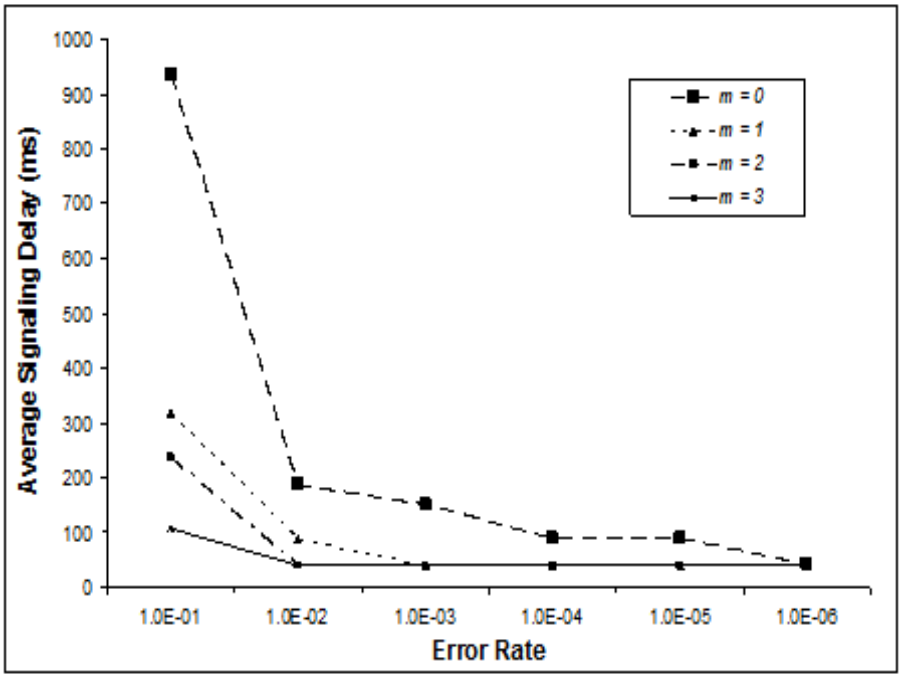
Figures 3.12a and 3.12b show the delivery delay performance of the protocol for merging field values ranging from $m = 0$ to $m = 3$ in AAPN MAN and WAN environments respectively. For $m = 0$, only the ARQ mechanism is used to recover damaged REQ messages and as expected the delay is too high. As (m) increases more REQ messages are recovered using the FEC mechanism and the delivery delay drops. The figures show how powerful the FEC mechanism is, as it takes full control of recovery at $m = 1$ for packet error rate equal to 10^{-3} and at $m = 2$ for packet error rate equal to 10^{-2}.

Notice the huge gap in average delay between $m = 0$ and $m = 1$ at packet error rate equal to 10^{-1} for AAPN WAN environment in figure 3.12b. After inspection of the simulation results it was noticed that the average delay at $m = 0$ and packet

error rate equal to 10^{-1} was unstable and kept on increasing as a function of the simulation time. Using the FEC mechanism with $m = 1$ stabilized the average delay and brought it within boundaries.



(a) AAPN MAN environment.



(b) AAPN WAN environment.

Figure 3.12: Average Delay versus Error Rate in AAPN MAN and WAN environments.

3.6 Summary

We have described the Control Messages Delivery Protocol (CMDP) that is mainly designed to work in environments where propagation delays are long and/or the error rates are high. It is used to deliver a burst of short messages in sequence and with no errors. Small messages are mainly control messages, hence the name Control Messages Delivery Protocol (CMDP).

The protocol is originally designed for the AAPN network where propagation delays between the edge and core nodes are significant and any retransmission could result in long buffering latency at the edge nodes. The protocol delay performance has been evaluated for both MAN and WAN AAPN environments. The results showed how powerful is the FEC mechanism as it takes full control of recovery at $m = 1$ for packet error rate equal to 10^{-3} and at $m = 2$ for packet error rate equal to 10^{-2}.

We demonstrated the CMDP protocol deployment in the AAPN network through a number of examples. The examples showed the operation of the protocol under different scenarios, each illustrates a specific aspect of the protocol operation. The examples covered the reliable transport of the two AAPN signaling messages, namely: the AAPN Request message (REQ) and the AAPN Grant message (GNT).

Chapter 4

Load Balancing for the AAPN Network

4.1 Introduction

This chapter describes a routing architecture that balances incoming Internet flows over the AAPN network. The architecture is based on the adaptive Highest Random Weight (adaptive HRW) algorithm proposed to design load balanced Internet routers. It assigns traffic load balancing weights to each source-destination edge node pair in the network. The weights are adapted based on the traffic load of the downstream and upstream links in the network.

The architecture can be seen as a combination of adaptive scheduling at the core nodes and adaptive load balancing at the edge nodes. The architecture is stateless and can compute routes quickly based on the packet flow identifier.

4.2 Literature Review

In [50], routing at the connection level has been studied and simulated. The methods perform connection admission control each time a new connection request is received, if no timeslot resources are available on the selected path, the connection is rejected.

In [51], the Valiant Load Balancing method [6], [7] has been used to eliminate the need for adaptive scheduling at the core nodes. The method replaces the adaptive scheduler [17]-[20] with a fixed scheduler that can handle any input traffic matrix within the boundaries of the hose traffic model [8]. The method routes at the timeslot level and so requires resequencing at the destination edge nodes. Since the traffic travels the network twice, end to end delay becomes a serious issue when deployed in a WAN environment.

In [52] and [53], routing and protection of MPLS LSPs over the AAPN network has been studied. The methods deploy the AAPN network as the OSPF backbone area [54]-[57].

4.3 The Adaptive Routing Architecture for the AAPN Network

In this section, we describe the adaptive routing architecture for the AAPN network. The architecture is distributed and is implemented at the different edge nodes in the network. Figure 4.1 represents the block diagram of the routing architecture at the

source edge node with ID equal to one. The diagram consists of two processing stages: a classification stage and a two-level mapping stage associated with every destination edge node in the AAPN network.

At the classification stage, the destination edge node of the incoming packets is determined using the AAPN routing tables described in [5]. The classified packets associated with a destination edge node are then mapped twice: the first map $f_c(\vec{v})$, called core selection, routes the packets to a core node while the second map $f_w^c(\vec{v})$, called wavelength selection, routes the packets to a wavelength through core node c selected by $f_c(\vec{v})$. The mapped packets are then stored in the Virtual Output Queue (VOQ) associated with the destination edge node on the selected wavelength.

The $f_c(\vec{v})$ and $f_w^c(\vec{v})$ mappings route packets at the flow level and are computed using equation 2.1. Each core node is assigned a mapping weight from the weight vector $\vec{x} = (x_1...x_d)$, where d is the number of core nodes. Each wavelength is assigned a mapping weight from the weight vector $\vec{y} = (y_1...y_e)$, where e is the number of wavelengths through the core node. For AAPN networks with single wavelength per fibre, only the $f_c(\vec{v})$ mapping function is used.

Core and wavelength selections work in tandem with no interaction to balance the incoming Internet flows over the AAPN network. For a given source-destination edge node pair, $f_c(\vec{v})$, balances the Internet flows over the fiber link paths going through the different core nodes, then $f_w^c(\vec{v})$, balances the Internet flows mapped to the fiber link path going through core node c over the different wavelengths in this path. The subscripts c in $f_c(\vec{v})$ and w in $f_w^c(\vec{v})$ are used to define core and wavelength selec-

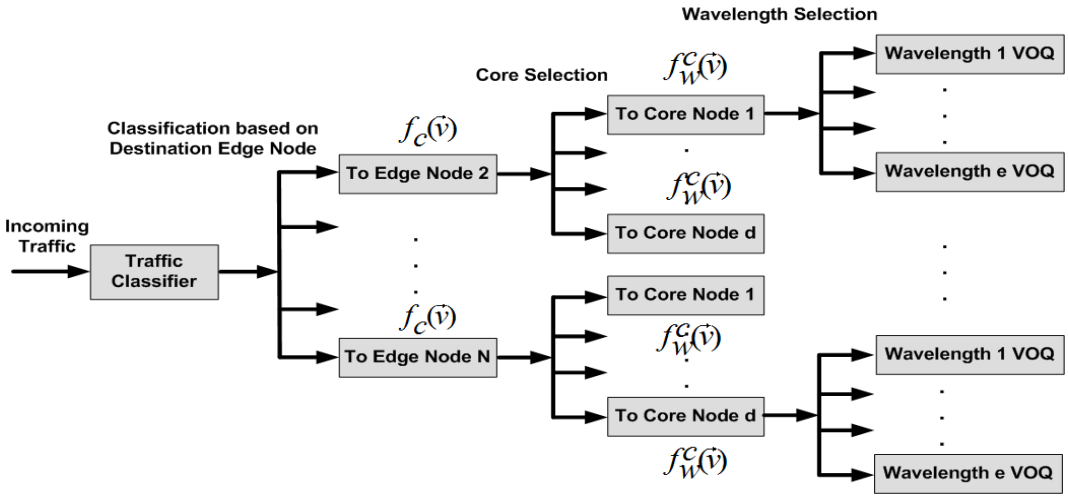


Figure 4.1: The adaptive routing architecture for the AAPN network

tion types, while superscript c in $f_w^c(\vec{v})$ defines the instance of wavelength selection associated with core node c selected by the $f_c(\vec{v})$ mapping function.

Ideally, the two-level mapping functions should balance the incoming traffic over the different fibre links and wavelengths of the AAPN network, but in reality, the functions could result in gross imbalance due to the Internet flow duration distribution [37], [58] and flow identification distribution [59]. Thus an adaptation loop is needed to keep the fibre links, and wavelengths within, from being congested. It adapts the weight vectors $\vec{x}$ and $\vec{y}$, based on the fibre links and wavelengths traffic loads, using equations 2.8 and 2.9. A traffic load is measured every frame and is defined as the percentage of timelsots requested per frame.

Compared to the adaptive HRW algorithm, the mapping in AAPN consists of two functions working in tandem with no interaction. Also, in the adaptive HRW the CP triggers adaptation of the weights based on the workload measurements of the

set of NPUs, while in AAPN mappings, two sets of traffic load measurements are used to trigger weights adaptation. For the $f_c(\vec{v})$ function associated with a source-destination edge node pair, the two traffic load measurement sets are the set of traffic load measurement of the downstream fibre links associated with the source edge node and the set of traffic load measurement of the upstream fibre links associated with the destination edge node. For the $f_w^c(\vec{v})$ function associated a source-destination edge node pair and core node c, the two traffic load measurement sets are the set of traffic load measurement of the downstream wavelengths on the fibre link from the source edge node to core node c and the set of traffic load measurement of the upstream wavelengths on the fibre link from core node c to the destination edge node.

Notice that triggering weight adaptation by two interrelated sets of traffic load measurements could result in instability and oscillations (i.e., triggering adaption to balance the traffic load over the downstream set of fibre links (wavelengths) could result in imbalance on the upstream set of fibre links (wavelengths) and visa versa). For this, one major objective of the simulations study is to prove the stability of the weights adaption triggering policy.

The adaptation triggering policy for the AAPN network is different from the policy used in the adaptive HRW algorithm. The next two sections describe two variations of the AAPN adaptation policy.

4.3.1 The AAPN Adaptation Triggering Policy

The default triggering policy applies to the adaptation of both weight vectors $\vec{x}$ and $\vec{y}$. The term channel is used in this section to generalize the description of the triggering policy, but when used for core selection it means a fiber link and when used for wavelength selection it means a wavelength.

We categorize the channels into downstream and upstream types. A downstream channel connects a source edge node to a core node while an upstream channel connects a core node to a destination edge node. The traffic load on a downstream channel originates at the source edge node and is destined to the other edge nodes, while the traffic load on an upstream channel originates at the source edge nodes and is destined to the destination edge node.

The channel set is defined differently for the fiber links and wavelengths. The downstream fiber link set associated with an edge node is defined as the set of links connecting this edge node to all the core nodes, while the upstream fiber link set associated with an edge node is defined as the set of links connecting the core nodes to this edge node. The downstream wavelength set associated with an edge node is defined as the set of wavelengths connecting this edge node to a core node, while the upstream wavelength set associated with an edge node is defined as the set of wavelengths connecting the core node to this edge node.

The policy uses the two channel sets traffic loads to adapt the weight vectors associated with a source-destination edge node pair. In cases where both traffic

load sets require adaptation, only one set, chosen randomly, is allowed to trigger the adaptation. The traffic loads for the two sets of channels are measured at different points in the network. The downstream channel set traffic loads are measured at the source edge node, while the upstream channel set traffic loads are measured at the core nodes. Measurement and filtering are performed periodically every frame, while the computation of dynamic threshold, channel set average traffic load, and weight vectors adaptation are performed at the edge nodes every ΔT. ΔT is chosen to be an integer multiple of the frame size and greater than the maximum round trip signaling delay between the edge and core nodes. This gives the weight vector changes enough time to propagate and take effect at the core nodes.

The traffic load of channel j, $\rho_j(t_n)$, is computed every frame as follows:

$$\rho_j(t_n) = \frac{\lambda_j(t_n)}{F} \tag{4.1}$$

where $\lambda_j(t_n)$ is the total number of timeslots requested in the nth frame, t_n is a discrete time measured at the end of each frame, and F is the frame size in timeslots.

Notice that if channel j is a fiber link then $\lambda_j(t_n)$ represents the total timeslots requested on all z wavelengths that goes through the fiber link and F is replaced by zF.

The channel filtered traffic load, $\overline{\rho}_j(t_n)$, is also computed every frame as follows:

$$\overline{\rho}_j(t_n) = \frac{1}{r}\rho_j(t_n) + \frac{r-1}{r}\overline{\rho}_j(t_n - t_{n-1}) \tag{4.2}$$

where $t_n - t_{n-1} = F * T_s$, is the frame size in seconds and r is the filtering constant computed using equation 4.3 to reduce the effect of short term fluctuations in the channel traffic load.

$$r = \frac{\Delta T}{F * T_s} \tag{4.3}$$

The channel set average traffic load, $\overline{\rho}(t_n)$, is computed every ΔT as follows:

$$\overline{\rho}(t_n) = \frac{\sum_{j=1}^{L} \overline{\rho}_j(t_n)}{L} \tag{4.4}$$

where t_n is the time at the end of the nth frame and L is the number of channels in the channel set.

Based on the computed values, a channel set is considered underutilized if $\overline{\rho}(t_n) \leq 1$ and overutilized if $\overline{\rho}(t_n) > 1$. A channel is considered overutilized if $\varepsilon_\rho(t_n) \leq \overline{\rho}_j(t_n)$ and underutilized if $\varepsilon_\rho(t_n) > \overline{\rho}_j(t_n)$ where the dynamic threshold, $\varepsilon_\rho(t_n)$, is computed every ΔT using equations 2.6 and 2.7.

The policy triggers adaptation of the weight vectors when imbalance occurs in the network. For example, assume that the upstream channel set going to a destination edge node is underutilized (overutilized) and one or more channels within the set are overutilized (underutilized), this will force all edge nodes to adapt their weight vectors to that destination edge node. In another example, assume that the downstream channel set coming from an edge node is underutilized (overutilized) and one or more channels within the set are overutilized (underutilized), this will force the edge node

to adapt his weight vectors to all destination edge nodes.

4.3.2 The AAPN Minimal Flow Remapping Triggering Policy

The default triggering policy requires all edge nodes to adapt their weight vectors when the channel set is underutilized (overutilized) and one or more channels within the set is overutilized (underutilized). In this section, we describe a variation of the policy that triggers those edge nodes that are most responsible for the channel state (also called the selection scheme). In this variation, not only a channel total traffic load is measured and filtered every frame, but the traffic load contributed by each source-destination edge node pair is also measured and filtered using equation 4.5.

Equation 4.5 is interpreted differently depending on the channel type. For a downstream channel l connecting a source edge node to a core node, $\overline{R}_x^l(t_n)$ and $R_x^l(t_n)$ are the filtered traffic load and the traffic load to destination edge node x in the nth frame respectively. For an upstream channel l connecting a core node to destination edge node, $\overline{R}_x^l(t_n)$ and $R_x^l(t_n)$ are the filtered traffic load and the traffic load from source edge node x in the nth frame respectively.

$$\overline{R}_x^l(t_n) = \frac{1}{r} R_x^l(t_n) + \frac{r-1}{r} \overline{R}_x^l(t_n - t_{n-1}) \tag{4.5}$$

To identify those edge nodes responsible for channel l state, we define the following two dynamic thresholds that are computed every ΔT:

$$\varepsilon_u(t_n) = \overline{R}^{mean}(t_n) + \overline{R}^{SD}(t_n)$$
$$\varepsilon_o(t_n) = \overline{R}^{mean}(t_n) - \overline{R}^{SD}(t_n) \tag{4.6}$$

where $\overline{R}^{mean}(t_n)$ is the mean value and $\overline{R}^{SD}(t_n)$ is the standard deviation of the filtered traffic loads at the end of the nth frame:

$$\overline{R}^{mean}(t_n) = \frac{\sum_x^N \overline{R}_x^l(t_n)}{N-1} \tag{4.7}$$

$$\overline{R}^{SD}(t_n) = \sqrt{\frac{\sum_x^N \left(\overline{R}_x^l(t_n) - \overline{R}^{mean}(t_n)\right)^2}{N-1}} \tag{4.8}$$

where N is the number of edge nodes in the network.

Based on the computed values, if the set of downstream channels from a source edge node is underutilized and one of the channels l is overutilized, then destination edge node x is contributing to the channel state if $\varepsilon_u(t_n) \leq \overline{R}_x^l(t_n)$. Conversely, if the set of downstream channels is overutilized and one of the channels is underutilized, then the traffic load to the destination edge node x is contributing to the channel state if $\varepsilon_o(t_n) \geq \overline{R}_x^l(t_n)$. The traffic load to the destination edge node x is not contributing to the channel state if $\varepsilon_o(t_n) < \overline{R}_x^l(t_n) < \varepsilon_u(t_n)$. Similar argument applies to the upstream channels where the traffic is going to a destination edge node.

4.3.3 Complexity Analysis

When a new packet arrives at an edge node, it needs to go through the different processing stages shown in figure 4.1. At the classification stage, a lookup table

operation is performed using the AAPN routing table to determine the destination edge node of the packet. This operation should be quick as the routing table is relatively small, for more details see [5].

The packet is then mapped twice: the first map $f_c(\vec{v})$, called core selection and the second map $f_w^c(\vec{v})$, called wavelength selection. Based on equation 2.1, the computational effort for mapping the packet to the right VOQ is $O(d) + O(e)$, where d is the number of core nodes in the network and e is the number of wavelengths going through the core node selected by $f_c(\vec{v})$. An important component in the performance of the mapping functions is the hashing function $g(\vec{v}, j)$, for a survey of hash functions for implementing the mapping, see [60]. In our simulations, we have used Fibonacci hashing to compute the $g(\vec{v}, j)$ function.

The architecture requires traffic loads for two sets of channels to be measured at different points in the network. The downstream channel set traffic loads are measured at the source edge node, while the upstream channel set traffic loads are measured at the core nodes. Measurement and filtering are performed periodically every frame, while the computation of dynamic threshold, channel set average traffic load, and weight vectors adaptation are performed at the edge nodes every ΔT. To get the measurements information from the core nodes to the edge nodes, a signaling component is needed. An extension of the CMDP protocol could added to handle the reliable signaling of the measured information.

4.4 Numerical Results

The OPNET tool has been used to simulate an AAPN network with 16 edge nodes and 8 core nodes. Since core and wavelength selections work independently, fiber links with one wavelength are used. Each wavelength has a capacity of 1 Gb/s. The round trip delay, RTD, is set to 40 ms and the frame size on each wavelength, μ, is set to 100 timeslots.

The objectives of the simulations are threefold. First, the performance of the routing architecture is studied and compared to the shortest path routing from the literature. Second, the performance of the routing architecture with three different channel set traffic load measurements is studied: using traffic load measurements from the downstream channel set, upstream channel set, and downstream and upstream channel sets. Third, the performance of the routing architecture with minimal flow remapping is studied. Any edge node can be selected to study the behavior; we choose edge node number 8.

4.4.1 Simulations Input

Each edge node generates self-similar traffic represented by the H-FSNDP fractal point process [61]. In the H-FSNDP model, the flow arrival process follows a Poisson model and the flow duration is heavy tailed Pareto model. Packets inter-arrival time within a flow is exponentially distributed. The following parameters describe the input traffic, the parameters are chosen to keep the load on the network close to 1.0:

1. The flow average arrival rate is 223,000 flow/s.

2. The Hurst parameter is 0.8 and the Fractal Onset Time Scale is 0.001.

3. The average flow duration is 0.3 s.

4. The flow packet average arrival rate is 128 packets/s.

5. The packet size is exponentially distributed with average size of 1024 bits (i.e., on average, each timeslot carries $(10^9 * 10\mu)/1024 = 10$ packets).

The incoming flows at each edge node are uniformly distributed to all other edge nodes in the network. The flow identifier distribution is a truncated normal distribution `N`(0, 1) out of the 32-bit integer space [38]. This approximates the distribution of IP source and destination addresses as described in [59].

4.4.2 The AAPN Routing Architecture Parameters

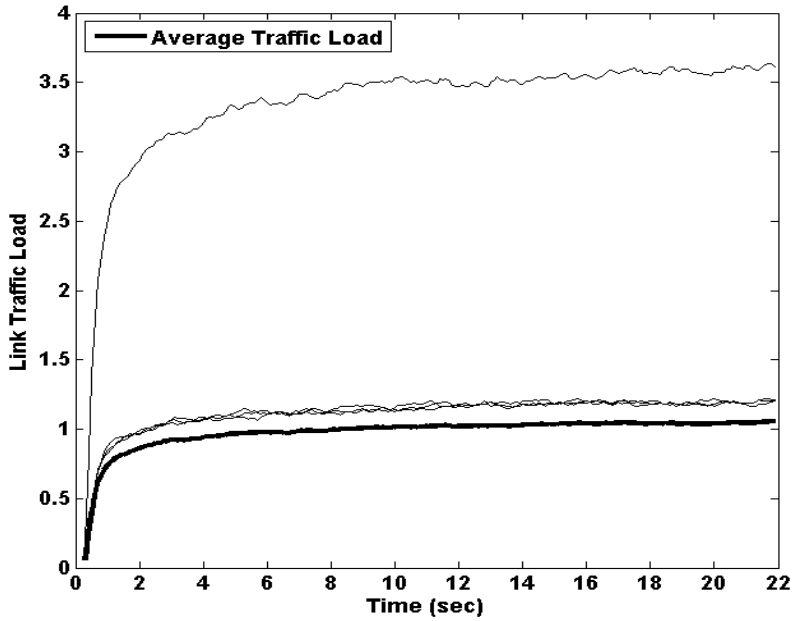
Link traffic loads at the edge and core nodes are computed every frame, where the frame length is equal to $100 * 10\mu = 1ms$. The adaptation period, ΔT, is set to 100 ms (i.e., k is set to 2.5). The Fibonacci hashing used in [38], is used here to compute $g(\vec{v}, j)$. The filtering constant, r, is set to 100. The hysteresis bound, ε_h, is set to 0.01 and the initial values for the adaptation weights are set according to the link capacities using equation 2.3 (i.e., $x_c = 1.0$)[36].

4.4.3 Routing Architecture Performance Simulations

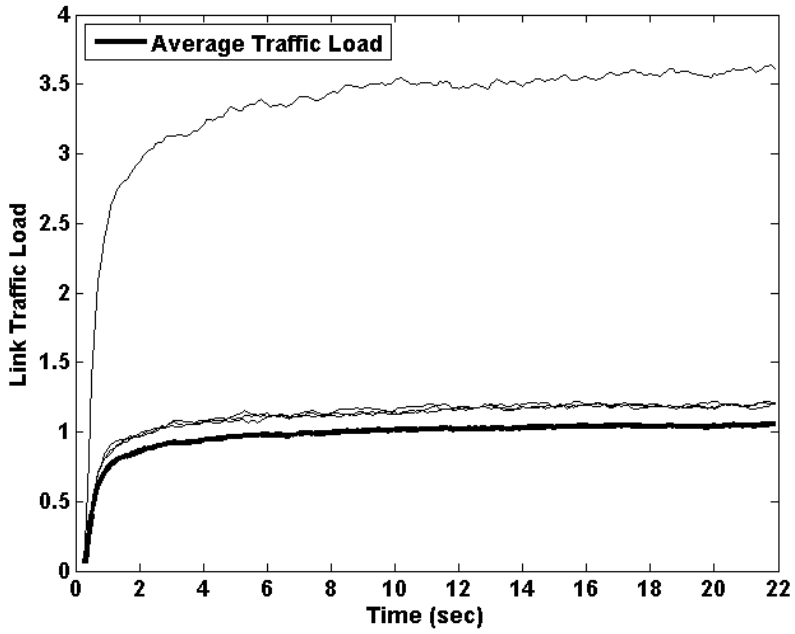
The behavior of three different routing architectures are compared in this section: 1) a static version of the routing architecture with no adaptation (also called the static load balancing method); 2) an adaptive version of the routing architecture with adaptation based on the upstream and downstream link sets traffic load (also called the adaptive load balancing method); 3) the shortest path routing, where the traffic between any edge node pair is sent through the core node that is part of the shortest path between the edge node pair.

Figure 4.2 shows the link traffic loads when shortest path routing is deployed. Figure 4.2a shows the traffic load on the downstream links from edge node 8, while figure 4.2b shows the traffic load on the upstream links to edge node 8. The average traffic load for the link sets is shown in bold, it is used as the ideal reference traffic load. The figures show how the traffic loads are widely spread around the average traffic load (some links have zero traffic load); this is due to the combination of the shortest path routing and the flow duration Pareto distribution.

Figures 4.3 shows the link traffic loads when the static load balancing method is deployed. Figure 4.3a shows the traffic load on the downstream links from edge node 8, while figure 4.3b shows the traffic load on the upstream links to edge node 8. The average traffic load for the link sets is shown in bold, it is used as the ideal reference traffic load. The figures show how the traffic load on the links is relatively spread around the average traffic load. To measure the spread around the average,

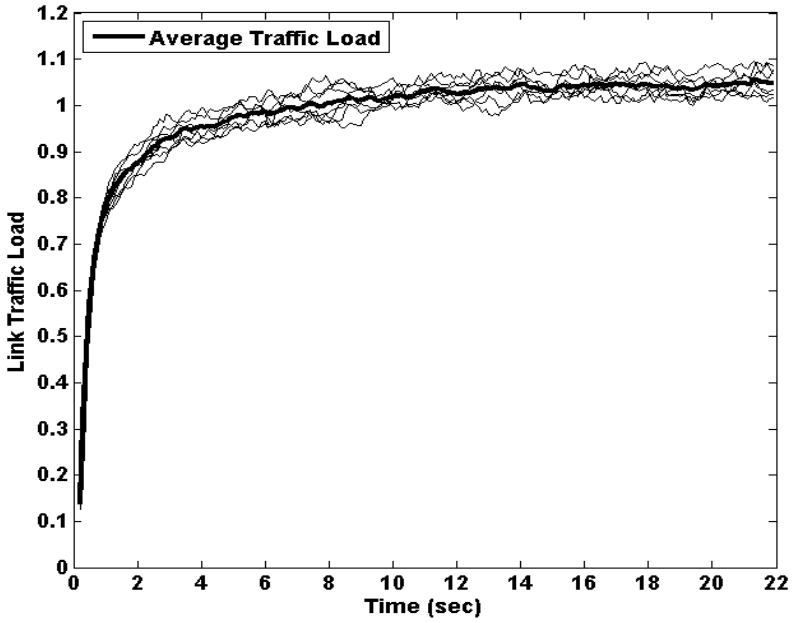


(a) The traffic load on the downstream links.

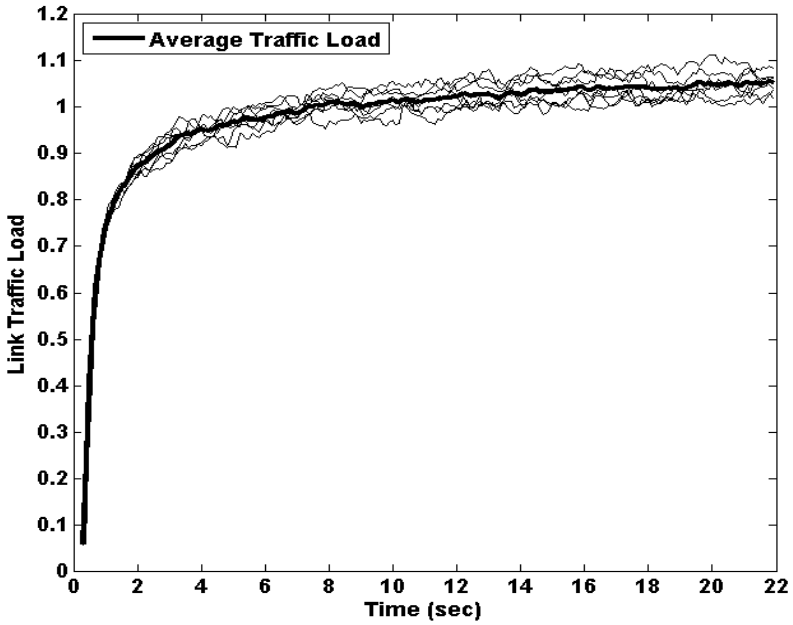


(b) The traffic load on the upstream links.

Figure 4.2: The traffic load on the downstream and upstream links with shortest path routing.

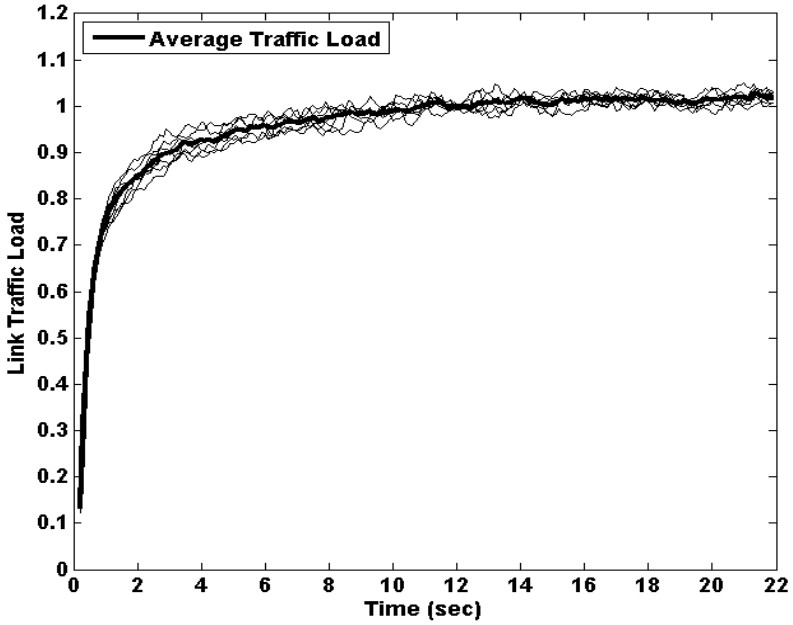


(a) The traffic load on the downstream links.

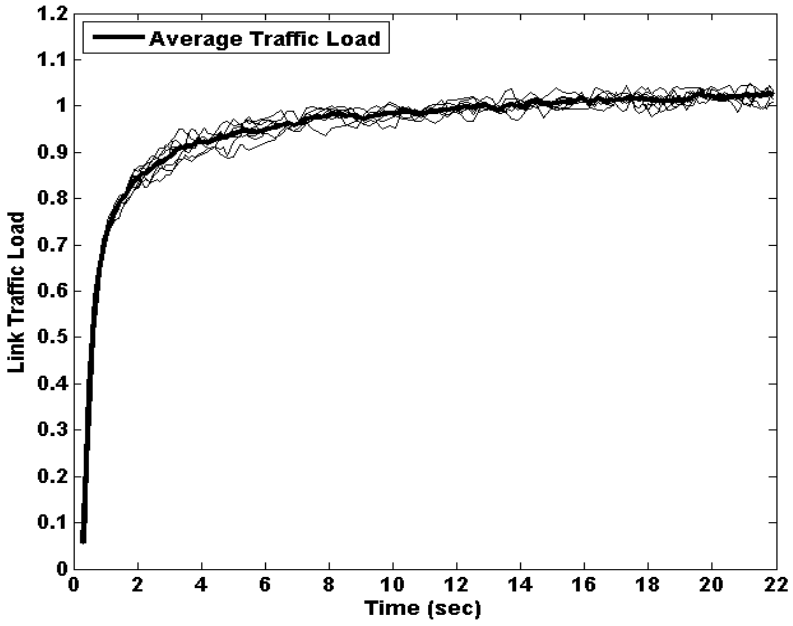


(b) The traffic load on the upstream links.

Figure 4.3: The traffic load on the downstream and upstream links with the static load balancing method.



(a) The traffic load on the downstream links.



(b) The traffic load on the upstream links.

Figure 4.4: The traffic load on the downstream and upstream links with the adaptive load balancing method.

the variance (confidence interval) of the traffic load on the different links with respect average traffic load is computed. Using the results it was found that the confidence interval is 0.10%.

Figures 4.4 shows the link traffic loads when the adaptive load balancing method is deployed. Figure 4.4a shows the traffic load on the downstream links from edge node 8, while figure 4.4b shows the traffic load on the upstream links to edge node 8. The average traffic load for the link sets is shown in bold, it is used as the ideal reference traffic load. The figures show how the traffic load on the links remain within the closest vicinity of the average traffic load when the adaptive load balancing method is deployed. They also show how stable the adaptive load balancing method is even when the weights adaptation is triggered by the downstream and upstream link sets traffic load. To measure the spread around the average, the variance (confidence interval) of the traffic load on the different links with respect average traffic load is computed. Using the results it was found that the confidence interval is 0.06%. This numerically shows that the adaptive load balancing outperforms the static load balancing.

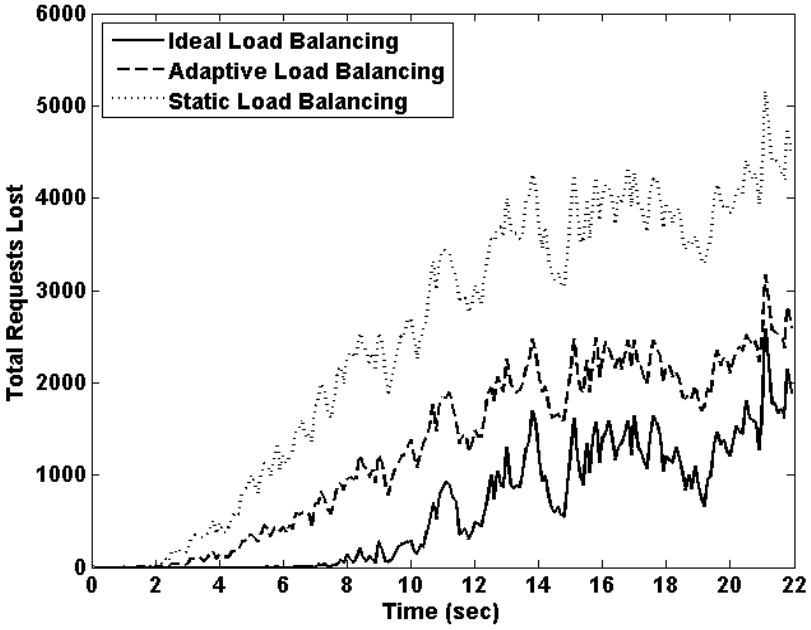
Figure 4.5 shows the requests dropped on the downstream and upstream links every ΔT period. As mentioned earlier, each request is for a timeslot and each time slot can carry up to 10 packets on average. Figure 4.5a shows the requests dropped on the downstream links of edge node 8, while figure 4.5b shows the requests dropped on the upstream links of edge node 8. The figures show the number of requests dropped every ΔT period when adaptive load balancing and static load balancing

	Downstream Links	*Upstream Links*
Ideal Load Balancing	0.80%	0.76%
Adaptive Load Balancing	1.67%	1.60%
Static Load Balancing	3.10%	3.00%

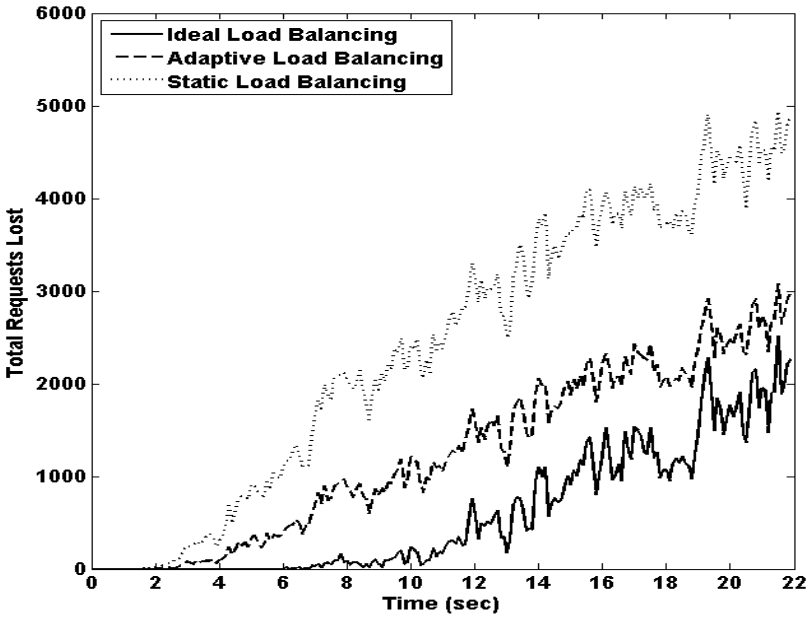
Table 4.1: Requests dropped statistics for the ideal, static, and adaptive load balancing methods.

methods are deployed. The ideal load balancing is also shown for comparison. The ideal request drop is computed as the difference between the requests received and the requests the link set can handle in ΔT period. The requests the link set can handle in $\Delta T = 0.1s$ is equal to *number of links in the link set (8) * number of frames in ΔT (100) * Frame size in requests (100)*; any requests received in excess of this value (i.e., $80,000$) are dropped for the ideal load balancing. The figures show how the adaptive load balancing method outperforms the static load balancing method. Table 4.1 summarizes the total requests dropped, as a percentage of the total requests sent on the downstream and upstream link sets, for the different methods.

Figure 4.6 shows, in a logarithmic scale, the number of flows appearing and the flows remapped every ΔT period for edge node 8 when the adaptive load balancing 4.6a and the minimal remapping adaptive load balancing 4.6b methods are deployed. The thick line at the top is actually 15 lines, each representing the flows leaving edge node 8 and going to each of the other 15 edge nodes every ΔT. From the figure, the flows going to each node is around 6,000, which is expected due to the adaptive load



(a) The number of requests dropped every ΔT period on the downstream links.



(b) The number of requests dropped every ΔT period on the upstream links.

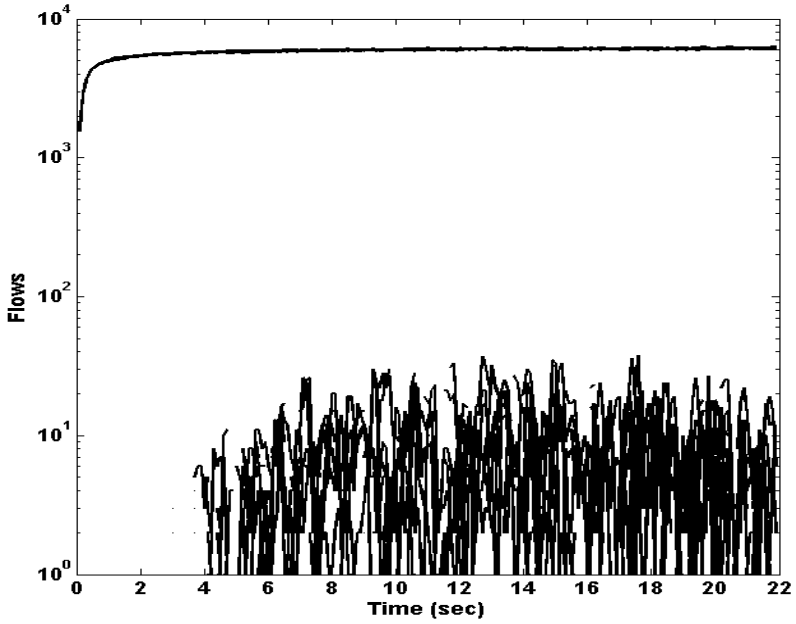
Figure 4.5: The number of requests dropped every ΔT period on the downstream and upstream links.

balancing. Each curve at the bottom is associated with a destination edge node, it represents the number of flows remapped every ΔT. The figure shows the stability of remapping for the different destination edge nodes and it also shows that less than 1% of the appearing flows is actually remapped.

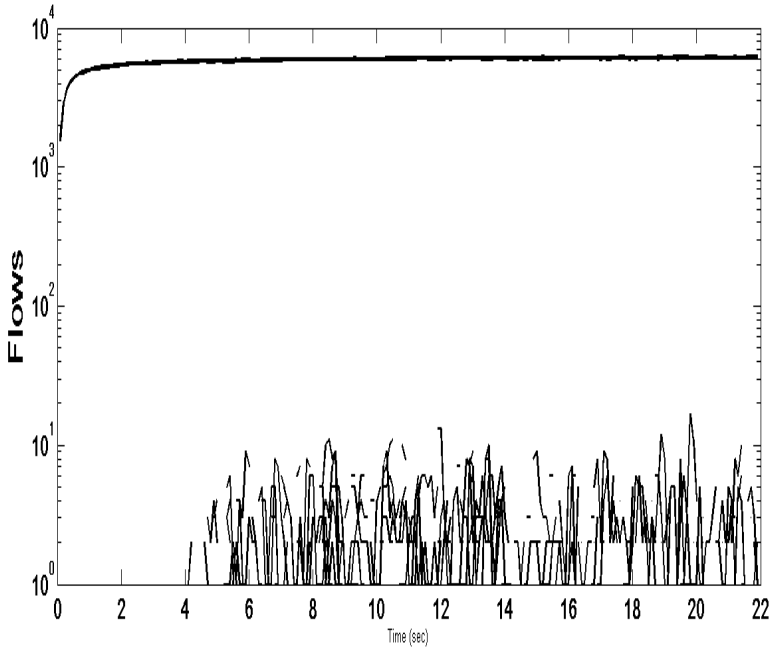
4.4.4 Routing Architecture with Different Link Sets Measurement Simulations

The behavior of three different routing architectures are compared in this section: 1) an adaptive version of the routing architecture with adaptation based on the downstream link set traffic load (also called downstream adaptive load balancing method); 2) an adaptive version of the routing architecture with adaptation based on the upstream link set traffic load (also called upstream adaptive load balancing method); 3) an adaptive version of the routing architecture with adaptation based on the downstream and upstream link sets traffic load (adaptive load balancing method from the previous section).

The figures 4.7 and 4.8 show the link traffic loads when the downstream link set traffic load and the upstream link set traffic load are used to trigger vector weights adaptation respectively. The average traffic load for the link sets is shown in bold, it is used as the ideal reference traffic load. Figure 4.7a shows how the traffic load on the downstream links remain within the closest vicinity of the average traffic load when the adaptation triggering policy is based on traffic loads from the downstream links,



(a) Flows appearing and remapped for the adaptive method.



(b) Flows appearing and remapped for the minimal remapping method.

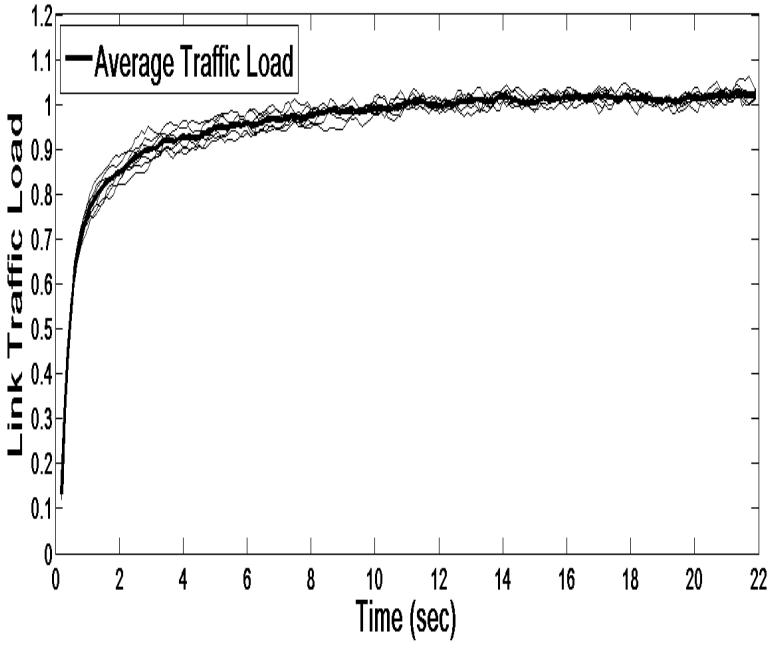
Figure 4.6: The number of flows appearing and remapped every ΔT period for the adaptive and minimal remapping adaptive load balancing methods.

	Downstream Links	*Upstream Links*
Downstream Adaptive Load Balancing	1.68%	3.40%
Upstream Adaptive Load Balancing	3.50%	1.63%
Adaptive Load Balancing	1.67%	1.60%

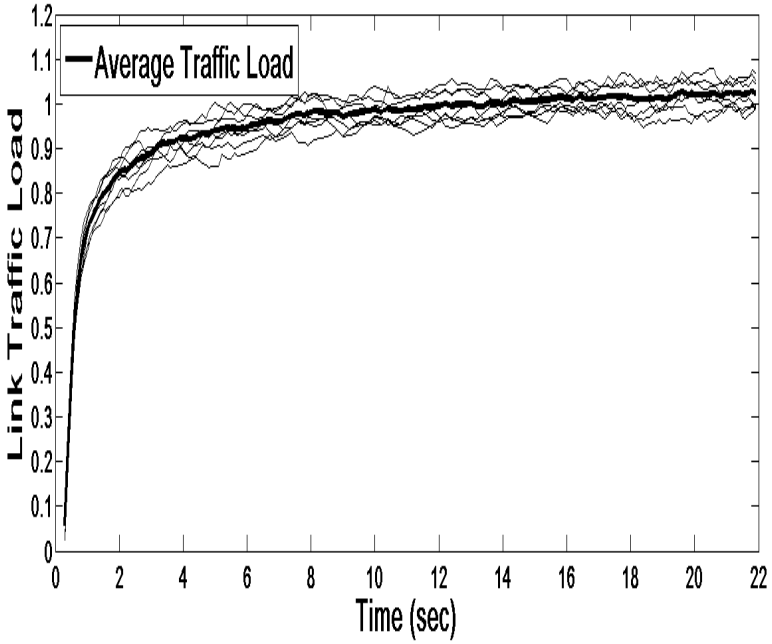
Table 4.2: Requests dropped statistics for the downstream, upstream, and adaptive load balancing methods.

while figure 4.7b shows how the traffic load on the upstream links spreads around the average traffic load. Figure 4.8b shows how the traffic load on the upstream links remain within the closest vicinity of the average traffic load when the adaptation triggering policy is based on traffic loads from the upstream links, while figure 4.8a shows how the traffic load on the downstream links spreads around the average traffic load. Compared to figure 4.4, it is clear that the adaptive load balancing method outperforms the downstream and upstream load balancing methods by keeping the traffic load within the closest vicinity of the average traffic load for both the downstream and upstream link sets.

Table 4.2 summarizes the total requests dropped, as a percentage of the total requests sent on the downstream and upstream link sets, for the different methods. Compared to table 4.1, it shows that the total requests dropped for the upstream links when the downstream adaptive load balancing method is deployed and for the downstream links when the upstream adaptive load balancing method is deployed are worse than that for the static load balancing method.

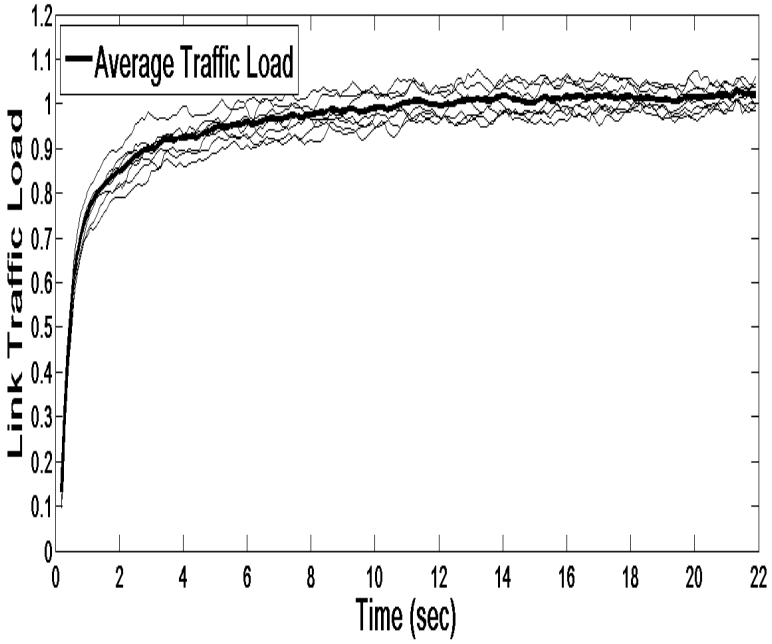


(a) The traffic load on the downstream links.

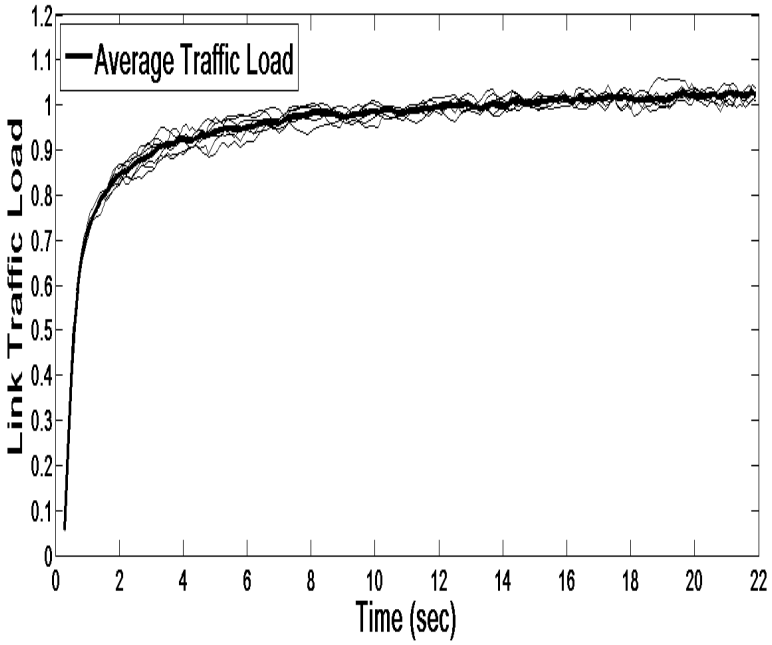


(b) The traffic load on the upstream links.

Figure 4.7: The traffic load on the downstream and upstream links when the adaptation triggering policy is based on traffic loads from the downstream links.



(a) The traffic load on the downstream links.



(b) The traffic load on the upstream links.

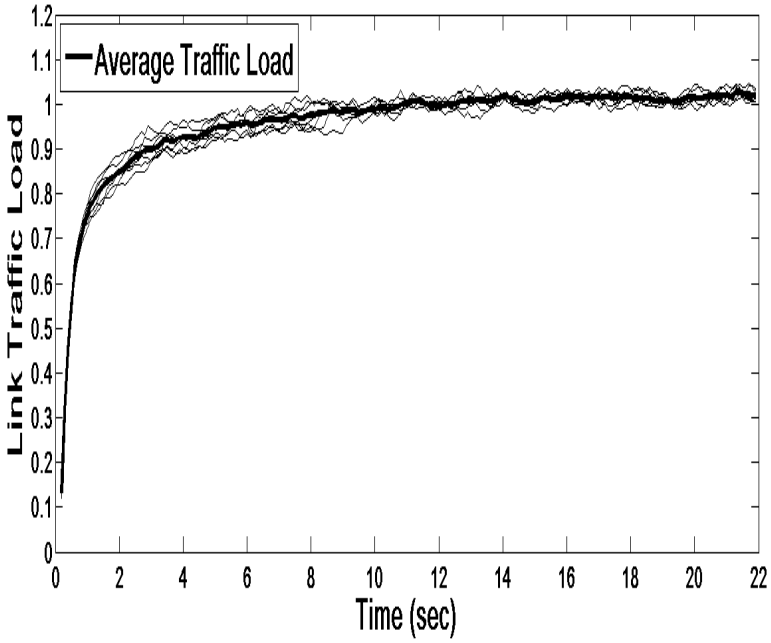
Figure 4.8: The traffic load on the downstream and upstream links when the adaptation triggering policy is based on traffic loads from the upstream links.

4.4.5 Routing Architecture with Minimal Remapping Performance Simulations

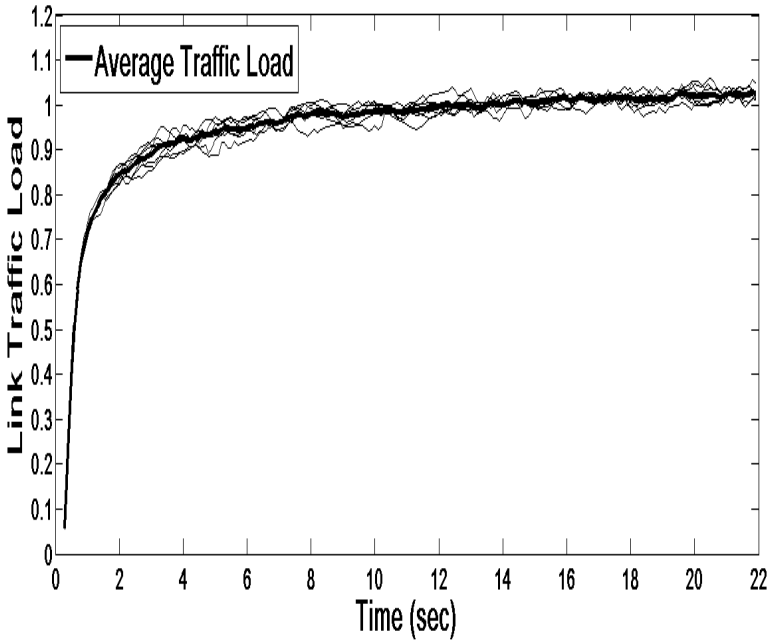
In this section, the behavior of the minimal remapping load balancing routing architecture (also called the minimal remapping load balancing method) is studied and compared to the adaptive load balancing method discussed perviously. As mentioned earlier, the objective of this method is to reduce the number of flows remapped while keeping the traffic load balanced over the different links in the network.

Figure 4.6b shows the number of flows appearing and the flows remapped every ΔT period when the minimal remapping adaptive load balancing method is deployed. Compared to the adaptive load balancing method remapping in figure 4.6a, the figure shows significant reduction in flow remapping. When adding up the remapped flows over the simulation period it was found that around 1% and 0.2% of the appearing flows are remapped when the default and minimal flow remapping triggering policies are deployed respectively. This gives the minimal flow remapping triggering policy a significant reduction in flow remapping compared to the default triggering policy.

Figure 4.9 shows how the traffic load on the downstream and upstream link sets remain within the closest vicinity of the average traffic load when the minimal remapping load balancing method is deployed. Figure 4.9a shows the load on the downstream links from edge node 8, while figure 4.9b shows the load on the upstream links to edge node 8. The average load for the link sets is shown in bold, it is used as the ideal reference load.



(a) The traffic load on the downstream links.



(b) The traffic load on the upstream links.

Figure 4.9: The traffic load on the downstream and upstream links.

	Downstream Links	*Upstream Links*
Minimal Adaptive Load Balancing	1.68%	1.63%
Adaptive Load Balancing	1.67%	1.60%

Table 4.3: Requests dropped statistics for the minimal remapping and adaptive load balancing methods.

Table 4.3 summarizes the total requests dropped, as a percentage of the total requests sent on the downstream and upstream link sets, for the adaptive load balancing and the minimal remapping adaptive load balancing methods. It shows that the minimal reampping adaptive load balancing method has a total request drop that is very close to that of the adaptive load balancing method.

4.5 Summary

We have described a routing architecture that balances incoming Internet flows over the AAPN network. The architecture is based on the adaptive Highest Random Weight (adaptive HRW) algorithm proposed to design load balanced Internet routers. It assigns traffic load balancing weights to each source-destination edge node pair in the network. The weights are adapted based on the traffic load of the downstream and upstream links in the network. The architecture can be seen as a combination of adaptive core node scheduling and adaptive load balancing at the edge nodes. It is stateless and can compute routes quickly based on the packet flow identifier.

We presented an extensive numerical evaluation of static and adaptive variations of the routing architecture and studied their effect on the network performance in terms of packet drop and flow remapping. The first part of the simulation results shows the stability of the adaptive load balancing method even when the weights adaptation is triggered by two different groups of traffic load. It also shows that the load on the links remain within the closest vicinity of the average load when the adaptive load balancing and minimal remapping adaptive load balancing methods are deployed with a small fraction of flows remapped.

The second part of the simulation results shows that neither the downstream nor the upstream load balancing method is enough to ensure balancing on all the links in the network; the adaptive load balancing method outperforms them both.

The third part of the simulation shows that the minimal remapping adaptive load balancing method enhances the adaptive load balancing method while reducing the number of flows remapped.

Performance measurements, in terms of requests dropped, show that the adaptive load balancing method significantly outperforms the static load balancing method.

Chapter 5

Load Balancing with QoS for the AAPN Network

5.1 Introduction

This chapter describes a Quality of Service (QoS) routing architecture that balances Internet flow traffic with different QoS requirements in the AAPN network. The architecture is based on the static and adaptive load balancing methods described in chapter 4. The static load balancing method is based on the static HRW used for designing load balanced web caches while the adaptive load balancing method is based on the adaptive HRW used for designing load balanced Internet routers.

The routing architecture performance is investigated using traffic that belongs to two DiffServ traffic classes, namely: Expedite Forwarding (EF) that is sensitive to variations in end-to-end delay & traffic drop rate and Best Effort (BE) that can

tolerate variations in end-to-end delay.

Different routing methods are used to handle the two traffic classes: while the static load balancing method is used to route the EF traffic, the adaptive load balancing method is used route the BE traffic. The objective is to have EF packets that belong to the same flow traverse the same path and to preserve load balancing by remapping the BE flows when needed.

5.2 The QoS Routing Architecture for the AAPN Network

The architecture is distributed and is implemented at the different edge nodes in the AAPN network. Figure 5.1 represents the block diagram of the QoS routing architecture. It exploits the static and adaptive load balancing methods described in chapter 4. The architecture splits the incoming traffic into two classes: Expedite Forwarding (EF) and Best Effort (BE). It then routes the BE traffic using the adaptive load balancing method and the EF traffic using the static load balancing method.

The static and adaptive load balancing methods were studied and deployed separately to route traffic that belongs to a single traffic class. The results have shown that the adaptive method outperforms the static method in terms of packets dropped at the cost of 1% flow remapping.

The static and adaptive load balancing methods share the same block diagram,

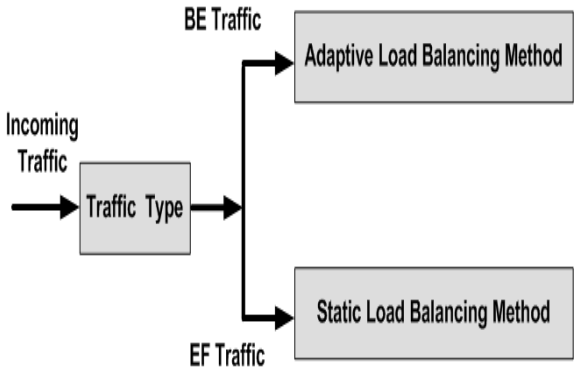


Figure 5.1: The QoS routing architecture for routing the BE and EF traffic.

see figure 4.1 in chapter 4. It consists of two processing stages: classification stage and a two-level mapping stage associated with every destination edge node in the network. At the classification stage, the destination edge node of the incoming packets is determined using the AAPN routing tables described in [5]. The classified packets are mapped twice: the first map $f_c(\vec{v})$, called core selection, routes the packets to a core node while the second map $f_w^c(\vec{v})$, called wavelength selection, routes the packets to a wavelength through core node c selected by $f_c(\vec{v})$. The mapped packets are stored in the VOQ associated with the destination edge node on the selected wavelength.

5.2.1 Complexity Analysis

When a new packet arrives at an edge node, it needs to go through the different processing stages shown in figure 5.1. At the splitting stage, a single check on one or more fields in the packet header is performed to determine the traffic class the packet belongs too. This operation should be quick as only two traffic classes are

assumed. Regardless of the traffic class the packet belongs too, it should go through the different processing stages shown in figure 4.1. For more information, see the complexity analysis section in chapter 4 for the static and adaptive load balancing architectures.

5.3 Numerical Results

The OPNET tool is used to simulate an AAPN network with 16 edge nodes and 8 core nodes. Since core and wavelength selections work independently, fiber links with one wavelength are used. Each wavelength has a capacity of 1 Gb/s. The maximum round trip signaling delay is set to 40 ms and the frame size on each wavelength, F, is set to 100 timeslots.

5.3.1 Simulations Input

The F-FSNDP-EF and F-FSNDP-FF fractal point processes are used for generating traffic at the different edge nodes [61]. The processes represent scenarios where the flow arrival process is fractal while the flow duration is exponentially distributed for the F-FSNDP-EF process and heavy tailed Pareto for the F-FSNDP-FF process. Packets inter-arrival time within a flow is exponentially distributed. The following parameters describe the traffic for both processes, the parameters are chosen to keep the traffic load on the network around 1.0:

1. The flow average arrival rate is 198,000 flow/s.

2. Arrival process Hurst parameter is 0.7 and the Fractal Onset Time Scale is 1.0.

3. The average flow duration is 0.3 s.

4. The flow packet average arrival rate is 128 packets/s.

5. The packet size is exponentially distributed with average size of 1024 bits (i.e., on average, each timeslot carries $(10^9 * T_s)/1024 = 10$ packets).

For the F-FSNDP-FF process, the flow duration hurst parameter is 0.8 and the Fractal Onset Time Scale is 0.001. The flow identifier distribution for both processes is a truncated normal distribution N(0, 1) out of the 32-bit integer space [38]. This approximates the distribution of IP source and destination addresses as described in [59].

Bernoulli distribution is used to divide the incoming flow traffic into BE and EF classes, where the probability p represents the percentage of EF traffic with respect to the total BE and EF traffic. The probability p takes values over the [0, 1.0] range with 0.1 increases. The objective is to study the architecture for different amounts of BE and EF traffic. Notice that for $p = 0$, all the incoming traffic is BE traffic and only the adaptive load balancing method is used, while for $p = 1.0$, all the incoming traffic is EF traffic and only the static load balancing method is used.

5.3.2 The Static and Adaptive Load Balancing Methods Parameters

Link traffic loads at the edge and core nodes are computed every frame, where the frame length is equal to 1 ms. The adaptation period, ΔT, is set to 100 ms. The Fibonacci hashing used in [38], is used here to compute $g(\vec{v}, j)$. The filtering constant, r, is set to 100. The hysteresis bound, ε_h, is set to 0.01 and the initial values for the adaptation weights are set according to the link capacities using equation 2.3 (i.e., x_c = 1.0)[36].

5.3.3 Simulations Results

Any link set of any edge node can be selected to study the behavior; we choose the downstream link set of edge node number 8, where a downstream link connects a source edge node to a core node, while an upstream link connects a core node to a destination edge node.

Figure 5.2 shows the actual traffic dropped, as a percentage of the incoming traffic, when the F-FSNDP-FF traffic model is used to generate traffic at the network edge nodes. It represents the scenario where the percentage of EF traffic, p, changes over the range [0, 1.0]. For each value of p, the simulation is run for 20 seconds and the actual traffic dropped is computed, at the end of the simulation, as the sum of the traffic dropped on the link set every frame. The figure also shows the ideal traffic dropped, as a percentage of the incoming traffic, which is computed as the sum of

the ideal traffic dropped every frame. The ideal traffic dropped on a link set is the difference between the incoming traffic and the link set capacity.

As the figure shows, the traffic dropped is the lowest at $p = 0$, since all the incoming traffic belongs to the BE class and is available for remapping by the adaptive load balancing method. As p increases, less and less BE traffic is available for remapping and so the traffic dropped starts to slightly increase. The traffic dropped is the highest at $p = 1.0$, since all the incoming traffic belongs to the EF class and is routed by the static load balancing method. Notice that, since schedulers with Strict Priority (SP) discipline are deployed at the core nodes, most the traffic dropped belongs to the BE class, except for $p = 1.0$ since all the incoming traffic belongs to the EF class.

As for the percentage of BE flows remapped, it was found that as the percentage of EF traffic, p, changes over the range [0, 0.9], the flows remapped slightly drops from 1% to 0.87%. This is because as the EF traffic increases less and less BE traffic is available for remapping.

Figures 5.3a and 5.3b shows the traffic load on the different downstream links when the F-FSNDP-EF and F-FSNDP-FF traffic models are respectively used to generate traffic at the network edge nodes. The average traffic load for the link set is shown in bold, it is used as the ideal reference load. The figure represents the scenario where 0.9 of the traffic is statically routed (i.e., EF traffic) and 0.1 is adaptively routed (i.e., BE traffic) to re-balance the network. This represents the extreme case in our simulations since only 0.1 of the incoming traffic is available for adaptation (i.e., remapping). The

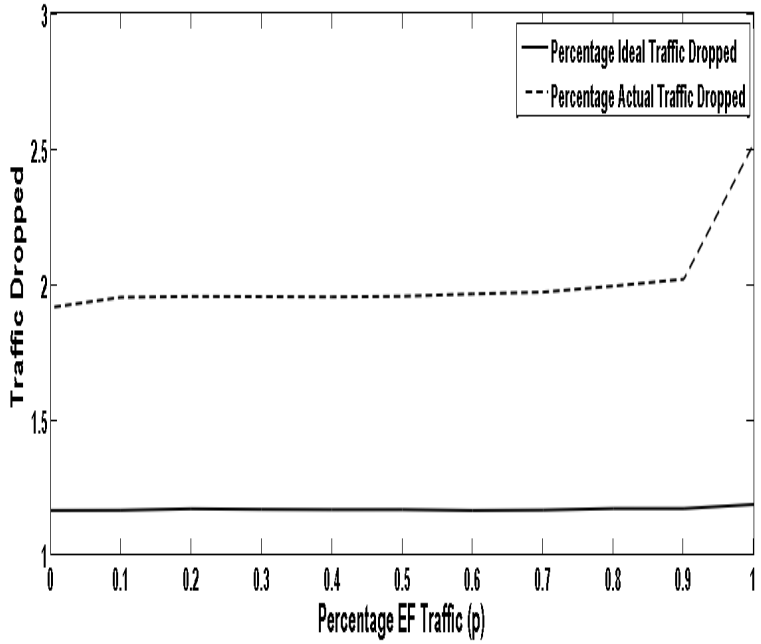
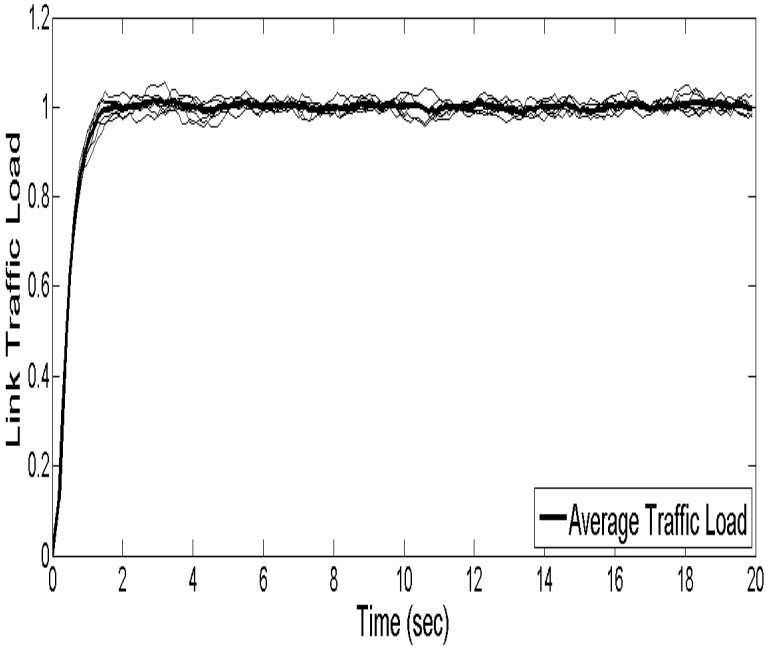


Figure 5.2: The actual and ideal traffic dropped on the downstream links of edge node 8.

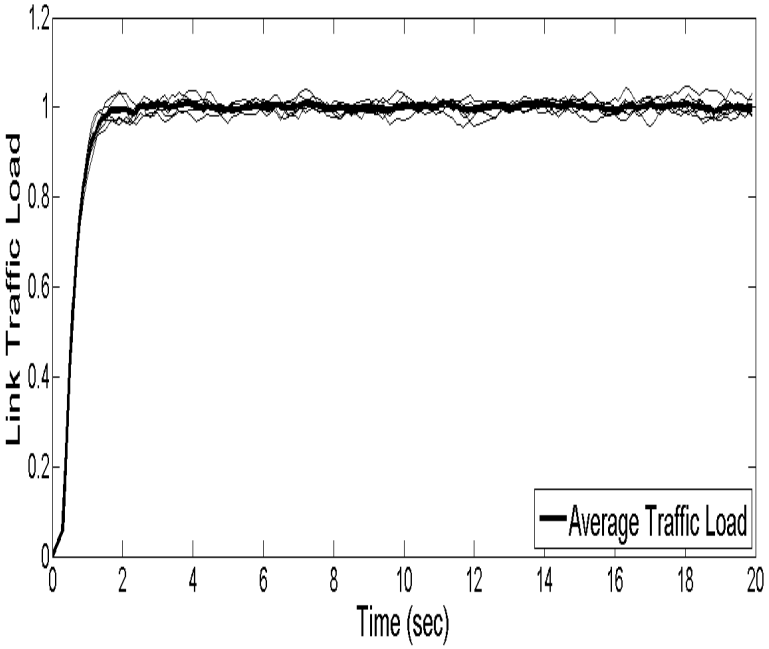
results show a stable behavior of the architecture using the two traffic models with the traffic load on the links remain within the closest vicinity of the average traffic load.

5.4 Summary

We have described a QoS routing architecture that balances Internet flow traffic with different QoS requirements in the AAPN network. Different routing methods are used to handle the two traffic classes: while the static load balancing method is used to route the EF traffic, the adaptive load balancing method is used route the BE traffic.



(a) The traffic load on the downstream links for the F-FSNDP-EF traffic model.



(b) The traffic load on the downstream links for the F-FSNDP-FF traffic model.

Figure 5.3: The traffic load on the downstream links for the F-FSNDP-EF and F-FSNDP-FF traffic models.

The objective is to have EF packets that belong to the same flow traverse the same path and to preserve load balancing by remapping the BE packets when needed.

Simulation results show a stable behavior of the architecture using the F-FSNDP-EF and F-FSNDP-FF input traffic models with the traffic load on the links remain within the closest vicinity of the average traffic load.

Chapter 6

Flow-Based Routing Architecture for Valiant Load-Balanced Networks

6.1 Introduction

This chapter describes a routing architecture that balances incoming Internet flows over the Valiant Load-Balanced (VLB) networks. The architecture is based on the adaptive HRW algorithm proposed to design load balanced Internet routers. To reduce flow remapping, the architecture extends the adaptive HRW algorithm with a minimal flow remapping selection scheme that identifies the traffic causing imbalance in the network and needs to be rerouted, where rerouting is implemented by adapting the weight vectors associated with the traffic.

Compared to the adaptive HRW method, the selection scheme further reduces flow remapping and the effect of packets reordering. The architecture is stateless and can compute routes quickly based on the packet flow identifier. This is an important issue when deploying the VLB network as an Internet backbone network where the number of flows is large and storing flow state information in lookup tables could limit the network performance.

6.2 Literature Review

Recently, researchers have been investigating the design of load balanced backbone networks that can accommodate highly variable traffic matrices. They have proposed load balancing network designs [6], [7] that can provide guaranteed performance for highly variable traffic matrices within the hose traffic model constraints [8]. The research is based on deploying the Valiant Load Balancing (VLB) as the method for routing the traffic over the network. The VLB method was first proposed by Valiant for processor interconnection networks [27] and was used later for designing scalable load balanced routers with performance guarantees [28]-[30].

The valiant load balanced networks deploy the VLB method in a logical mesh topology network, see Fig. 6.1. The mesh topology could be implemented using tunneling or MPLS LSPs or another method depending on the underlying technology. The VLB method routes traffic over the network in two phases: in the first phase (load balancing phase), each node balances its traffic to all N nodes in the network,

regardless of the packet destination. Each node receives $1/N$-th of every node's traffic. In the second phase (forwarding phase), packets are delivered to their destination. Suppose that the incoming traffic rate to each node is less than or equal to r, and the traffic is uniformly balanced over the N nodes in the network, the actual traffic on each link due to the first routing is at most r/N. Since each node can receive traffic at a maximum rate of r, and it receives $1/N$-th of the traffic from every node, the actual traffic on each link due to the second routing is also at most r/N. Therefore, to guarantee 100% throughput for any traffic matrix within the constraints of the hose traffic model, a full-mesh network where each link has capacity $2r/N$ is sufficient.

In this chapter, we propose an adaptive routing architecture for the balancing phase. It routes traffic at the flow level to reduce the effects of packets reordering at the destinations.

6.3 Problem Description

One concern with VLB method is packet re-sequencing at the destination. When packets from the same flow travel through different routes with different delays, they may arrive at the destination out of order. By using flow-based hashing, it is possible to ensure that no single Internet flow is sent over multiple routes. But as shown in [37] and [41], static hashing, even though preserves packet ordering, could result in load imbalance. Due to uneven packet flow popularities and highly skewed size dis-

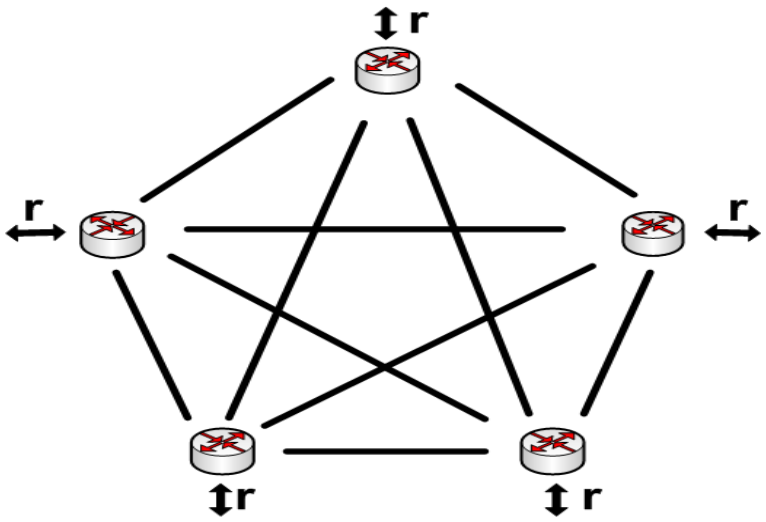


Figure 6.1: Five edge nodes interconnected by a fully logical mesh.

tributions, significant imbalance can occur and so dynamic hashing is a more efficient candidate.

In this chapter, we present a routing architecture that balances incoming Internet flows over the VLB networks. It is based on the adaptive HRW algorithm proposed to design load balanced Internet routers [38] and load balanced AAPN networks [62]. The architecture extends the adaptive HRW algorithm with a minimal flow remapping selection scheme that identifies those source-destination edge node pairs whose flow traffic is causing imbalance in the network and need to adapt their weights. The architecture is stateless and can compute routes quickly based on the packet flow identifier. This is an important issue when deploying the VLB network as an Internet backbone network where the number of flows is large and storing flow state information in lookup tables could limit the network performance.

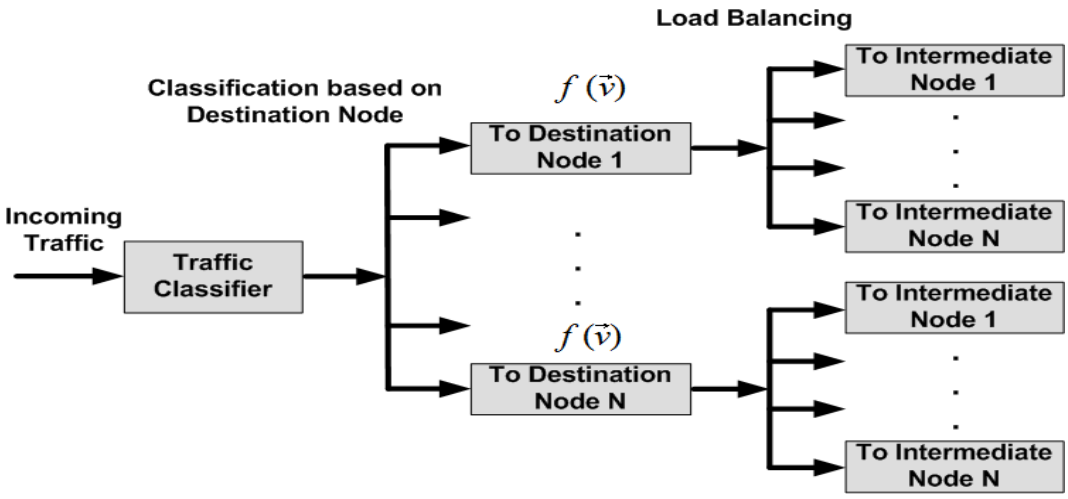


Figure 6.2: Routing architecture for balancing the Internet traffic over the VLB network.

6.4 The Adaptive Routing Architecture for the VLB Network

The architecture is distributed and is implemented at the different nodes in the network, see Fig. 6.2. It consists of two processing stages: a classification stage and a mapping stage. The mapping stage consists of a number of mapping functions $f(\vec{v})$, each function is associated with a destination node in the network. At the classification stage, the destination node of the incoming packets is determined. The classified packets are then mapped to an intermediate node using the mapping function $f(\vec{v})$ associated with that destination node.

The $f(\vec{v})$ function balances the Internet flows over the two-hop routes going through the different intermediate nodes. Notice that only the load balancing phase traffic could be rerouted, forwarding phase traffic is sent directly to its destination.

6.4.1 Mapping Policy

The mapping function $f(\vec{v})$, associated with a destination node, route packets to that destination at the flow level and is computed using equation 2.1. Each intermediate node is assigned a mapping weight from the weight vector $\vec{v} = (v_1...v_N)$, where N is the number of intermediate nodes.

6.4.2 Triggering Policy

The objective of the policy is to monitor the traffic loads on the set of logical links connecting a node to all the intermediate nodes. It triggers adaptation when imbalance in the logical link set is discovered. Imbalance occurs if the logical link set is overutilized (underutilized) and one or more links are underutilized (overutilized).

The policy requires periodic measurement and filtering of the traffic load on each link. The traffic load $\rho_j(t)$ on link j, is computed every ΔT period as follows:

$$\rho_j(t) = \frac{\lambda_j(t)}{2r/N} \tag{6.1}$$

where $\lambda_j(t)$ is the measured rate on link j and $2r/N$ is the link rate.

The filtered traffic load on link j, $\overline{\rho}_j(t)$, is also computed every ΔT period using

equation 2.4. To evaluate the status of the logical link set, the filtered traffic load $\overline{\rho}(t)$ of the link set is measured using equation 2.5. The set traffic load, $\rho(t)$, is defined as the measured rate on the link set divided by the link set capacity, where the set measured rate is the sum of the individual link measured rates and the set capacity is $N * (2r/N)$.

Based on the computed values, the link set is underutilized if $\overline{\rho}(t) \leq 1$ and overutilized if $\overline{\rho}(t) > 1$. A link is overutilized if $\varepsilon_\rho(t) \leq \overline{\rho}_j(t)$ and underutilized if $\varepsilon_\rho(t) > \overline{\rho}_j(t)$ where the dynamic threshold, $\varepsilon_\rho(t)$, is computed every ΔT using equation 2.6 and 2.7.

6.4.3 Adaptation Policy

Ideally, the mapping functions $f(\vec{v})$ should balance the incoming Internet traffic over the different logical links in the network, but in reality, the functions could result in gross imbalance. Thus adaptation and rerouting of some of the traffic flows is needed to keep links from being congested. Adaptation is done by recomputing the weight vectors $\vec{v}$, based on measured links traffic loads, using equations 2.8 and 2.9. It should be noted that only the balancing phase traffic is subject to rerouting.

6.4.4 Minimal Flow Remapping Selection Scheme

When a node detects imbalance in the link set, adaptation of the weight vectors $\vec{v}$ is needed, but the question is, which weight vectors to adapt? The architecture as described so far, requires the node to adapt all its weight vectors. In this section, we

describe a selection scheme that identifies the traffic causing the imbalance and needs to be rerouted. Since the traffic entering a node is classified using its destination, identifying the traffic also means identifying the destination node and the weight vector associated with that node.

The scheme requires the node to measure and filter the traffic load entering the network to all destination nodes in the network. This is done using equation 6.2.

$$\overline{\rho}_{ij}(t) = \frac{1}{r}\rho_{ij}(t) + \frac{r-1}{r}\overline{\rho}_{ij}(t - \Delta T) \tag{6.2}$$

where $\rho_{ij}(t)$ and $\overline{\rho}_{ij}(t)$ are the measured and filtered traffic load for destination i on link j.

To identify the traffic causing link j to be underutilized or overutilized, we define the following two dynamic thresholds. The thresholds are computed every ΔT:

$$\begin{aligned} \varepsilon_u(t) &= \overline{R}^{mean}(t) + \overline{R}^{SD}(t) \\ \varepsilon_o(t) &= \overline{R}^{mean}(t) - \overline{R}^{SD}(t) \end{aligned} \tag{6.3}$$

where $\overline{R}^{mean}(t)$ is the mean value and $\overline{R}^{SD}(t)$ is the standard deviation of the filtered traffic loads from equation 6.2:

$$\overline{R}^{mean}(t) = \frac{\sum_i^N \overline{\rho_{ij}}(t)}{N-1} \tag{6.4}$$

$$\overline{R}^{SD}(t) = \sqrt{\frac{\sum_i^N \left(\overline{\rho_{ij}}(t) - \overline{R}^{mean}(t)\right)^2}{N-1}} \tag{6.5}$$

where N is the number of nodes in the network.

Based on the computed values, if a link set, associated with a node, is underutilized and one of the links j is overutilized, then the traffic going to node i is contributing to the link state if $\varepsilon_u(t) \leq \overline{\rho_{ij}}(t)$. Conversely, if the link set is overutilized and one of the links j is underutilized, then the traffic going to node i is contributing to the link state if $\varepsilon_o(t) \geq \overline{\rho_{ij}}(t)$. The traffic going to node i is not contributing to the link state if $\varepsilon_o(t) < \overline{\rho_{ij}}(t) < \varepsilon_u(t)$.

6.4.5 Complexity Analysis

When a new packet arrives at a node, it needs to go through the different processing stages shown in figure 6.2. At the classification stage, a lookup table operation is performed to determine the destination edge node of the packet. This operation should be quick as the routing table is relatively small.

The packet is then mapped using the function $f(\vec{v})$. Based on equation 2.1, the computational effort for mapping the packet to the right intermediate node is $O(N)$, where N is the number of intermediate nodes in the network. An important component in the performance of the mapping functions is the hashing function $g(\vec{v}, j)$, for a survey of hash functions for implementing the mapping, see [60]. In our simulations, we have used Fibonacci hashing to compute the $g(\vec{v}, j)$ function.

The architecture requires traffic loads to be measured locally at different source nodes in the network. Measurement and filtering are performed periodically every ΔT.

6.5 Numerical Results

The OPNET tool is used to simulate a fully meshed VLB network with $N = 16$ nodes. All nodes are of equal capacity r with traffic parameters for each node described in the next section. The network diameter (i.e., delay from a source node to an intermediate node) is set to 40 ms. Link capacity is set to $2r/N$.

6.5.1 Simulations Input

Each node receives self-similar traffic represented by the F-FSNDP-EF fractal point process [61]. In the F-FSNDP-EF model, the flow arrival process is fractal while the flow duration is exponentially distributed. Packets inter-arrival time within a flow is exponentially distributed. The traffic entering and leaving the network is set to meet the constraints of the hose traffic model at all times. The following parameters describe the input traffic:

1. The flow average arrival rate is 100,000 flow/s.
2. The Hurst parameter is 0.8 and the Fractal Onset Time Scale is 0.001.
3. The Average flow duration is 2.0 s.
4. The flow packet average arrival rate is 128 packets/s.
5. The packet size is exponentially distributed with average size of 1024 bits (i.e., on average, each timeslot carries $(10^9 * T_s)/1024 = 10$ packets).

The flow identifier distribution is a truncated normal distribution N(0, 1) out of the 32-bit integer space [38]. This approximates the distribution of IP source and destination addresses as described in [59]. The traffic is uniformity distributed to all destinations in the network, the objective to see how the traffic from different sources to different destinations affect each other when the network is working at full capacity.

6.5.2 Routing Architecture Parameters

The adaptation period, ΔT, is set to 100 ms which is greater than the network diameter. This is important to make sure that effects of any weight adaptation propagates through the network. The Fibonacci hashing used in [38], is used to compute $g(\vec{v}, j)$. The filtering constant, r, is set to 100. The hysteresis bound, ε_h, is set to 0.01 and the initial values for the adaptation weights are set according to the logical link capacities using equation 2.3 (i.e., $x_j = 1.0$)[36].

6.5.3 Simulations Results

Simulation results shown here are for one of the nodes, similar results were generated for the other nodes.

Figures 6.3, 6.4, and 6.5 show how the traffic load on the different logical links remain within the closest vicinity of the link set average traffic load (which represents the ideal traffic load solution) when no selection scheme and minimal flow remapping selection scheme are deployed in the routing architecture. The figure also shows the case when the static routing architecture is used (i.e, no adaption of the weight

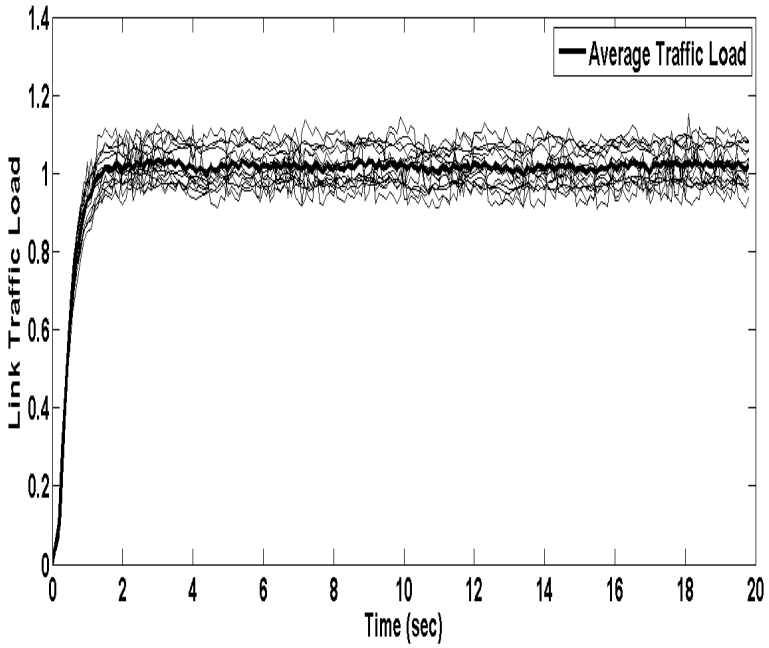


Figure 6.3: The traffic load on the link set with the static HRW scheme.

vectors). The results show that it is enough to trigger adaptation of the traffic that is contributing to the imbalance to keep the network balanced.

Figure 6.6 shows, in a logarithmic scale, the total number of flows appearing and the total number of flows remapped, every ΔT, when no selection scheme and minimal flow remapping selection scheme are deployed in the routing architecture. The figure shows significant reduction in the amount of flows remapped when the selection scheme is deployed.compared to the case where no selection scheme is deployed.

Table 6.1 presents the percentage of flows remapped when the selection scheme is

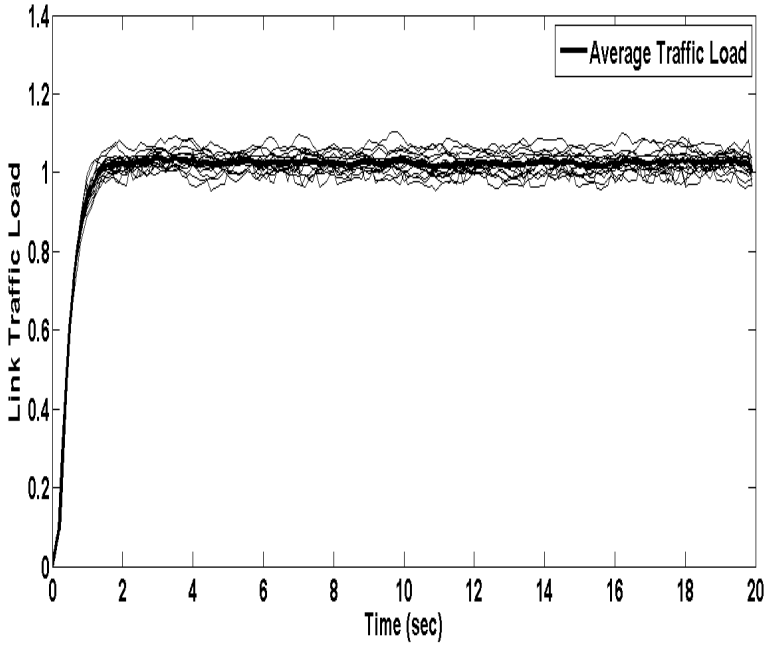


Figure 6.4: The traffic load on the link set with the adaptive HRW scheme.

deployed in the routing architecture. It shows that using the selection scheme gives 51% improvement compared to the routing architecture with no selection scheme.

Fig. 6.7 shows the percentage of traffic dropped every ΔT in excess of the ideal drop when static, no selection, and minimal flow remapping selection schemes are deployed. The ideal traffic drop is computed as the difference between the traffic received and the link set capacity. The figure shows how the adaptive architectures with no selection and minimal flow remapping selection schemes outperform the static architecture. It also shows that the drop rate of the minimal remapping selection and the no selection schemes are within the same range.

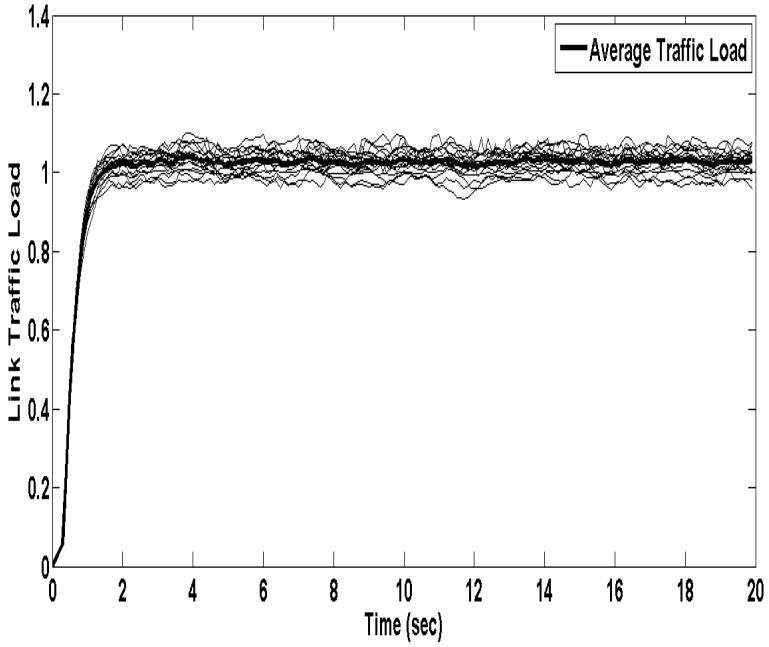


Figure 6.5: The traffic load on the link set with the adaptive HRW scheme with minimal flow remapping.

6.6 Summary

We have described a routing architecture that balances Internet flows over the VLB network. The architecture is based on the adaptive HRW algorithm proposed to design load balanced Internet routers. To reduce flow remapping, the architecture extends the adaptive HRW algorithm with a minimal flow remapping selection scheme that identifies the traffic causing imbalance in the network and needs to be rerouted. The architecture is stateless and can compute routes quickly based on the packet flow identifier.

The simulation results show that the load on the links remain within the closest

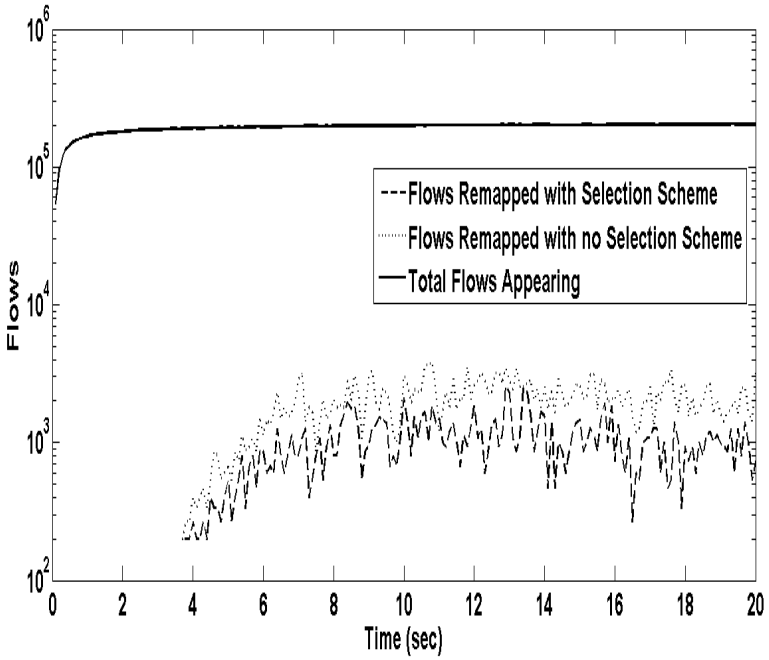


Figure 6.6: The total number of flows appearing and the total number of flows remapped every ΔT period.

vicinity of the average load at the cost of remapping a small fraction of flows. The results show significant reduction in the amount of flows remapped when the selection scheme is deployed. Performance measures, in terms of traffic dropped, show that the adaptive routing architecture significantly outperforms the static routing architecture.

Deploying the routing architecture requires measuring the traffic load on the logical link set. We believe that the significant reduction in flow remapping and traffic drop are worth this measurement. Furthermore, these measurements are local to the source node and should not have big impact on the performance of current powerful edge nodes.

The focus of this work was on VLB networks with nodes of equal capacity, the

Flow Remappings	%all
Routing Architecture with no Selection Scheme	0.84%
Routing architecture with Selection Scheme	0.41%
Improvement of selection over no selection	%
1 - (selection / no selection)	51.0%

Table 6.1: Flow remappings when using no selection scheme and when using the selection scheme in the VLB network.

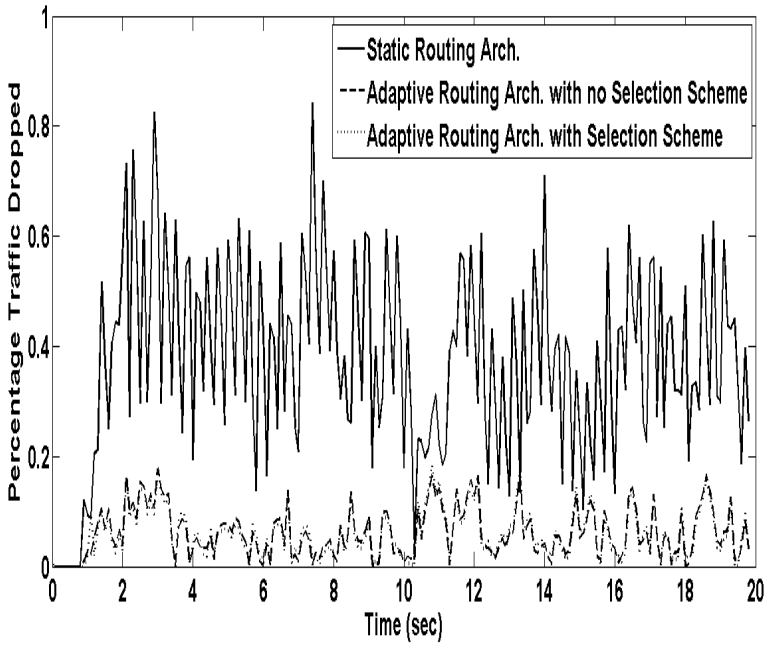


Figure 6.7: The figure shows the percentage of traffic dropped every ΔT in excess of the ideal drop when static, no selection, and minimal flow remapping selection schemes are deployed.

work should also be applicable to networks with general topologies and heterogenous nodes. In this case, linear programming formulation is used to compute the logical link capacities and identify the intermediate nodes [7]. The result is a mesh network with different logical link capacities and node capacities. The routing architecture described here should work fine with the computed logical link capacities since the HRW algorithm [36] is designed for heterogeneous environments too.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The problem of deploying AAPN as the backbone network for ISPs where IP routers are interconnected by the AAPN network is studied in this thesis. More specifically, the thesis focused on designing a routing architecture for balancing the IP traffic load over the AAPN network while minimizing packet reordering within one flow identified by common fields within the packet header.

Since the number of flows is large in the backbone networks, storing flow state in lookup tables can limit the network performance. Hence the routing architecture eliminates the need for flow lookup tables. Different static and adaptive load balancing methods were studied and their behavior and effect on the network performance in terms of packet drop and flow remapping is analyzed.

The applicability of the load balancing methods to topologies beyond AAPN over-

laid star topology is extended using the Valiant Load Balancing (VLB) method. The method is used to build an overlaid star topology on top of the physical topology. Deploying the VLB method in the AAPN network, eliminated signaling and replaced the dynamic core schedulers with a static scheduler that can accommodate all traffic matrices within the hose traffic model boundaries.

The Control Messages Delivery Protocol (CMDP) is proposed as part of this thesis to provide a reliable channel between the edge and core nodes in the AAPN network. The protocol is designed to work in environments where propagation delays are long and/or the error rates are high. It is used to deliver a burst of short messages in sequence and with no errors. Combined with the reliable routing protocol proposed previously for the AAPN network, they form the control plane for the network that supports the operation of the load balancing methods.

In summary, this thesis work should provide interested ISPs with a comprehensive study of load balancing methods and a control plane for deploying the AAPN network as their backbone network. The thesis work should support the deployment of the AAPN network at the core of an IP network using the overlay internetworking model, where control in the AAPN network is hidden from control in the IP network.

The static load balancing method has the advatage that no remapping is performed and so requires not measurments. It has the disadvantage that the traffic drop rate is high compared to the adaptive load balancing method. The adaptive load balancing method, event though it has low drop rate, requires traffic measurements on both the upstream and downstream link sets. It also requires some flows to

be remapped (i.e., some resquencing is needed). It was found in the simulations that 1% of the incoming flows were remapped. The minimal flow remapping load balancing method reduces the flows remapped compared to the adaptive load balancing. It was found in the simulations that 0.2% of the incoming flows were remapped. It requires further measurements on both the upstream and downstream link sets. The QoS load balancing method supports two classes of traffic and is dependent on both the static and adaptive load balancing methods. Either the adaptive load balancing or the minimal flow remapping load balancing method can be used in the QoS method.

7.2 Future Work

Following the previous discussion, it would be interesting to extend our work to routing at the connection level. Most of the modern routing methods at the connection level are state dependent. Two interesting methods are: the Markov Decision Process (MDP) model [63]-[66], which allows for formulating the problem as an optimization of an objective function, and the Least Loaded Path (LLP) method [63], [67]. Our objective is to present a new formulation of the routing and Connection Admission Control (CAC) problem that maximizes the revenue from the network while reducing the average end to end delay.

The formulation is not restricted to AAPN's overlaid star topology but can be used in networks with general topology. Routing at the connection level is motivated by the deployment of AAPN network at the core of a circuit-switched network like

MPLS. The method requires the connection state on each link and can produce a near optimal routing in terms of network revenue and average delay. The exact MDP optimal model is relatively heavy in processing and memory requirements, but using simplifications the method can produce a sub-optimal method with manageable levels. The method facilities traffic distribution control and can provide for efficient trade off between revenue and average delay objectives.

The LLP method is well known in the literature and industry and is considered one of the best performing routing methods in use. LLP checks the current state of the different paths in the network and routes the new connection over the least loaded path. A path load is equal to the load of the most loaded link a long the path. The method requires the aggregate state information from all links in the network. The method has been implemented in real networks in both Canada and the US.

Appendix A

Load Balancing with Minimal Flow Remapping for Network Processors

This appendix deploys the minimal flow remapping scheme described in chapter 4 in the Internet router architecture from [38]. The objective is to show that load balancing methods proposed in this thesis are also applicable to the router architecture from [38]. Deploying the QoS architecture described in chapter 5 in the router architecture is left for future work.

A.1 Introduction

Maintaining high performance in parallel processing routers while preserving packet ordering within the flows is a difficult problem. To preserve packet ordering, hashing at the flow level has been used to distribute packet processing workload among the

router processing units. Even though it preserves ordering, hashing alone may cause significant workload imbalance and thus adaptive methods are usually needed.

In this appendix, we present an input port selection scheme that can be augmented with the adaptive Highest Random Weight (adaptive HRW) method. The adaptive HRW is a hash-based method that works at the flow level and is used to balance packet processing workload among the router processing units. When imbalance occurs, the adaptive HRW method triggers all input ports to re-balance their workload among the processing units. When augmented the selection scheme, the adaptive HRW method should be able to identify the subset of input ports responsible for the imbalance.

Due to the exponential growth of the Internet traffic, old fashion centralized routers with a single general purpose processor have been replaced with high performance multiprocessor routers [68], [69]. The multiprocessor router architecture utilizes special purpose processing units, also called network processors (NP) or forwarding engines (FE), and can be classified into distributed and parallel architectures.

The distributed architecture [70], deploys the network processors at the different router input ports; each network processor (or set of network processors) is responsible for processing the packet workload for that input port. Since the traffic at the different input ports is rarely evenly distributed, underutilization is a major drawback of this architecture.

The parallel architecture [71], [72], enhances the router utilization by deploying a pool of network processors far from the router input ports. The objective is to remove any processor-port association. The architecture suffers from two drawbacks: first,

the network processors interconnect topology could limit the router's performance and second, the scheduler responsible for balancing the incoming processing workload among the different network processors is a single point of failure for the entire router.

This appendix presents an input port selection scheme that can be augmented with the adaptive Highest Random Weight (adaptive HRW) method [38]. The method was proposed for balancing processing workload among the pool of network processors. When deployed with the adaptive HRW method, the selection scheme should further reduce the number of flows remapped while balancing the packet processing workload among the different network processors within the router.

A.2 Related Work

In the networking domain, load balancing among web caches was the main objective of the work in [35]. It lists the main design goals of a load mapping method, namely: low overhead, load balancing, high cache hit rate, and minimal disruption. The work presents the Highest Random Weight (HRW) hashing method that maps web object requests to different web caches. The method eliminates object duplications, achieves fault tolerance, and keeps objects disruption to minimum. Minimal disruption is important when adding/removing web caches since only a minimum number of object requests are migrated among the web caches. Ref. [36] enhances the HRW method by extending it to heterogeneous web caches. The method assigns multipliers (weights) to the different web caches that are proportional to the caches storage capacities. The

method is implemented by some Microsoft products and is known as the Cache Array Routing Protocol (CARP).

In the router design domain, load balancing among the router network processors was the main objective of the work in [37] and [43]. The main goal is to design load sharing methods for balancing packet processing workload among the network processors while trying to preserve packets order within the flows. Ref. [37], presents a hash-based method that exploits the highly skewed Internet flow size distribution to design a scheduler that only shifts few aggressive flows from heavily loaded forwarding engines. Ref. [43], presents another hash-based method that exploits the bursty nature of the Internet traffic to design a scheduler that shifts bursts within the flows. It is based on the insight that TCP flow bursts are the primary source of burstiness of Internet traffic. The two methods are adaptive but require considerable flow state information to be stored in the router.

The adaptive HRW method [38], presents a load balancing scheme for processing packets in IP routers. The method is based on the HRW hashing [36] augmented with an adaptive control loop. Although the HRW method ensures load balancing over the request object space, load imbalance could happen due to uneven popularity of the individual objects. An important goal of the method is to preserve the minimal disruption property when balancing the processing workload among the network processors. A comparison, between the adaptive HRW method and the load balancing method from [37], is presented by the work in [39].

In this appendix, we present a selection scheme that can be augmented with the

adaptive HRW method. The objective is to further reduce flow remapping while keeping the IP router balanced. When imbalance occurs in the router's network processor set, the adaptive HRW method triggers all input ports to adapt their weights to re-balance the router. The selection scheme should help identify and trigger the subset of input ports that is responsible for the router imbalance.

A.3 The Minimal Flow Remapping Selection Scheme

The motivation behind the selection scheme is that the workloads due to the actual traffic at the input ports may be different and so require different weight vectors $\vec{x}$. Some input ports may have well behaving traffic workload that is already balanced while other ports may have badly behaving traffic workload that requires weight vector adaptation. The adaptive HRW method monitors the status of the router and the individual network processors and if imbalance occurs, it adapts the weight vector at all input ports. The objective of the selection scheme is to adapt the weight vectors at a subset of the input ports. Compared to the adaptive HRW scheme this should further minimize the amount of flows remapped while keeping the router balanced.

The selection scheme identifies, for an underutilized (overutilized) network processor, the input ports causing the network processor to be underutilized (overutilized). The scheme is based on the triggering policy used to determine the status of the individual network processors. To identify the input ports causing imbalance, the scheme requires for each network processor j the periodic measurement of the work-

load intensity $\rho_{ij}(t)$ from every input port i. Network processor j workload intensity from input port i is defined as the actual network processor workload from the input port divided by the network processor capacity. Equation $A.1$ shows how the filtered workload intensity at network processor j from input port i is calculated.

$$\overline{\rho}_{ij}(t) = \frac{1}{r}\rho_{ij}(t) + \frac{r-1}{r}\overline{\rho}_{ij}(t - \Delta T) \tag{A.1}$$

where $\overline{\rho}_{ij}(t)$ is the filtered workload intensity for input port i at network processor j and r is the filtering constant.

The scheme uses the dynamic threshold $\varepsilon_j(t)$ defined in equation $A.2$ to determine whether an input port is contributing to network processor j underutilization or overutilization status:

$$\varepsilon_j(t) = \frac{1}{2N}\left(1 + \overline{\rho}_j(t)\right) \tag{A.2}$$

where N is the number of input ports in the router and $\overline{\rho}_j(t)$ is the filtered workload intensity for network processor j computed using equation 2.4. N is used in equation $A.2$ since a network processor workload intensity from an input port is computed using the network processor capacity.

Algorithm 1 shows how the selection scheme works when the router is underutilized and one or more network processors are overutilized. The algorithm has the following input: the set of input ports, the set of overutilized network processors, the set of filtered workload intensities $\overline{\rho}_{ij}(t)$, and the set of dynamic thresholds $\varepsilon_j(t)$. The

output of the algorithm is a subset A of the input ports whose weight vectors $\vec{x}$ will be adapted. The algorithm loops through the overutilized network processors and for each processor it identifies the input ports contributing to the processor's overutilization status. An input port i is contributing to the overutilization of network processor j if $\overline{\rho}_{ij}(t) \geq \varepsilon_j(t)$.

Input: overutilized network processors set, input ports set, $\overline{\rho}_{ij}(t)$, $\varepsilon_j(t)$

Output: subset of input ports that require adaptation

$A = \emptyset$

foreach *overutilized network processor* j **do**

  foreach *input Port* i **do**

    if $\overline{\rho}_{ij}(t) \geq \varepsilon_j(t)$ **then**
      $A = A \cup \{i\}$

    end

  end

end

Algorithm 1: The selection scheme when the router is underutilized and one or more network processors are overutilized

Algorithm 2 shows how the selection scheme works when the router is overutilized and one or more network processors are underutilized.

Input: underutilized network processors set, input ports set, $\overline{\rho}_{ij}(t)$, $\varepsilon_j(t)$

Output: subset of input ports that require adaptation

$A = \emptyset$

foreach *underutilized network processor* j **do**

  foreach *input Port* i **do**

    if $\overline{\rho}_{ij}(t) < \varepsilon_j(t)$ **then**
      $A = A \cup \{i\}$

    end

  end

end

Algorithm 2: The selection scheme when the router is overutilized and one or more network processors are underutilized

A.4 Numerical Results

The OPNET tool is used to simulate a router with 16 input ports and 8 network processors. Depending on the parameters of the input traffic described later, it is assumed that each network processor is capable of processing the full workload from two input ports. The goal is to investigate the performance with respect to the router bounds.

A.4.1 Simulations Input

Each input port generates self-similar traffic represented by the F-FSNDP-FF fractal point process [61]. In the F-FSNDP-FF model, the flow arrival process is fractal and the flow duration is heavy tailed Pareto model. Packets inter-arrival time within a flow is exponentially distributed. The following parameters describe the input traffic, the parameters are chosen to keep the total workload intensity on the router around 1.0:

1. The flow average arrival rate is 5,000 flow/s.
2. Arrival process Hurst parameter is 0.7 and the Fractal Onset Time Scale is 1.0.
3. The Average flow duration is 0.3 s.
4. The flow duration Hurst parameter is 0.8 and the Fractal Onset Time Scale is 0.001.
5. The flow packet average arrival rate is 128 packets/s.

The flow identifier distribution is a truncated normal distribution N(0, 1) out of the 32-bit integer space [38]. This approximates the distribution of IP source and destination addresses as described in [59].

A.4.2 The Adaptive HRW Method Parameters

The adaptive HRW parameters are taken from [38] where the adaptation period, ΔT, is set to $15ms$, the filtering constant, r, is set to 3, and the hysteresis bound, ε_h, is set to 0.01. The Fibonacci hashing is used here to compute $g(\vec{v}, j)$. It is assumed that network processors with equal processing capacities are used in the router and the initial values for the adaptation weight vector are proportional to the network processors capacities using equation 2.3 (i.e., $x_j = 1.0$)[36].

A.4.3 Simulations Results

Figure A.1 shows the filtered workload intensity on the different network processors when no selection (a) and minimal flow remapping selection scheme (b) are deployed with the adaptive HRW method. The filtered router workload intensity is shown in bold, it is used as the ideal reference workload intensity. It shows how the workload intensity for the different network processors remain within the closest vicinity of the router workload intensity (which represents the ideal workload intensity solution) when no selection scheme and the minimal flow remapping selection scheme are deployed with the adaptive HRW method. This proves that it is enough to trigger weight vector adaptation of a subset of the input ports whose flow workload is

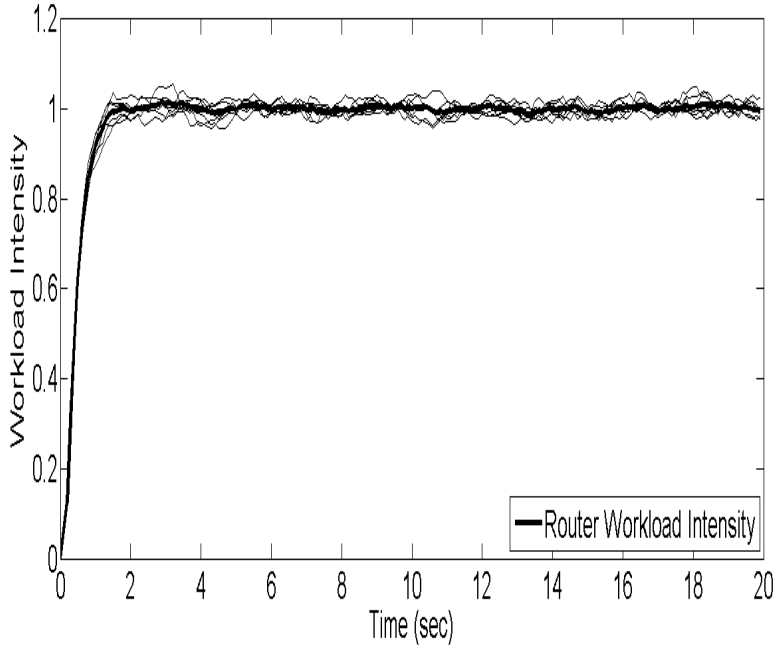
contributing to the imbalance to keep the router balanced.

Figure A.2 shows, in a logarithmic scale, the total number of flows appearing and the total number of flows remapped, every ΔT at all input ports, when no selection scheme and the minimal flow remapping selection scheme are deployed with the adaptive HRW method. The figure shows significant reduction in the amount of flows remapped when the selection scheme is deployed compared to the case where no selection scheme is deployed.

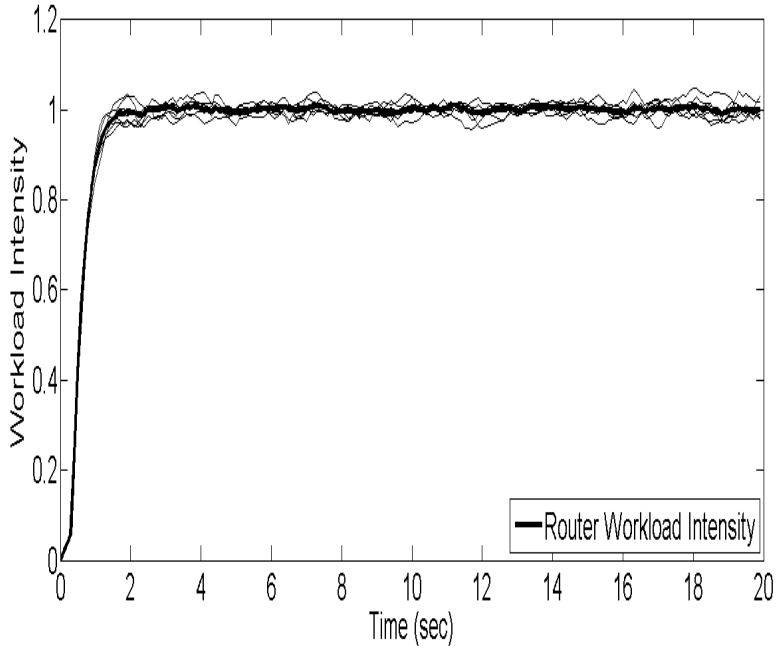
Table A.1 presents the percentage of flows remapped when the selection scheme is deployed with the adaptive HRW method. It shows that using the selection scheme gives 57% improvement (i.e., reduction in flow remapping) compared to the adaptive HRW with no selection scheme.

Flow Remappings	%all
Adaptive HRW with no Selection Scheme	1%
Adaptive HRW with Selection Scheme	0.43%
Improvement of selection over no selection	%
1 - (selection / no selection)	57.0%

Table A.1: Flow remappings when using no selection scheme and when using the selection scheme in the load balanced router.



(a)



(b)

Figure A.1: The filtered workload intensity on the different network processors.

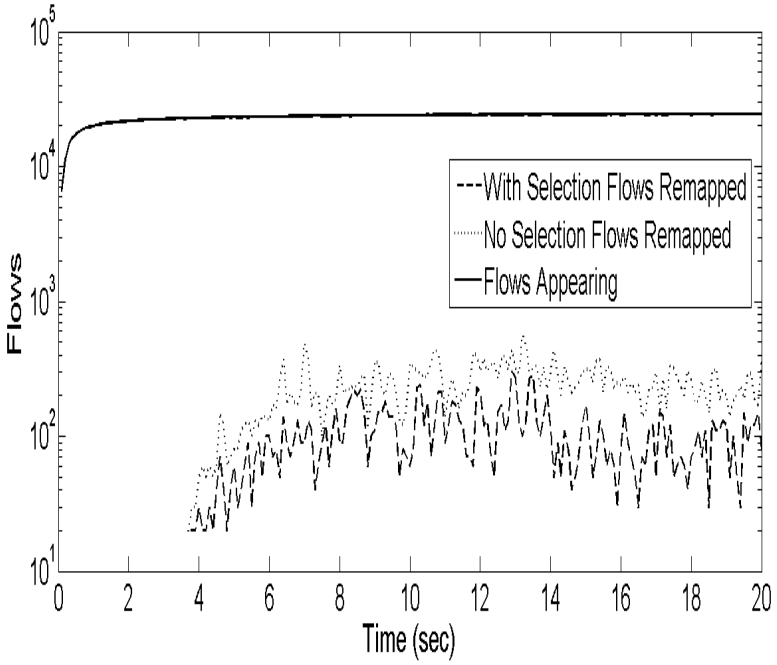


Figure A.2: The total number of flows appearing and the total number of flows remapped every ΔT period.

Figure A.3 compares the number of packets dropped every ΔT when no selection and minimal flow remapping selection schemes are deployed. It shows how both curves closely follow the total router curve which represents the ideal packet drop. The ideal packet drop is computed as the difference between the packets received and the network processor set capacity.

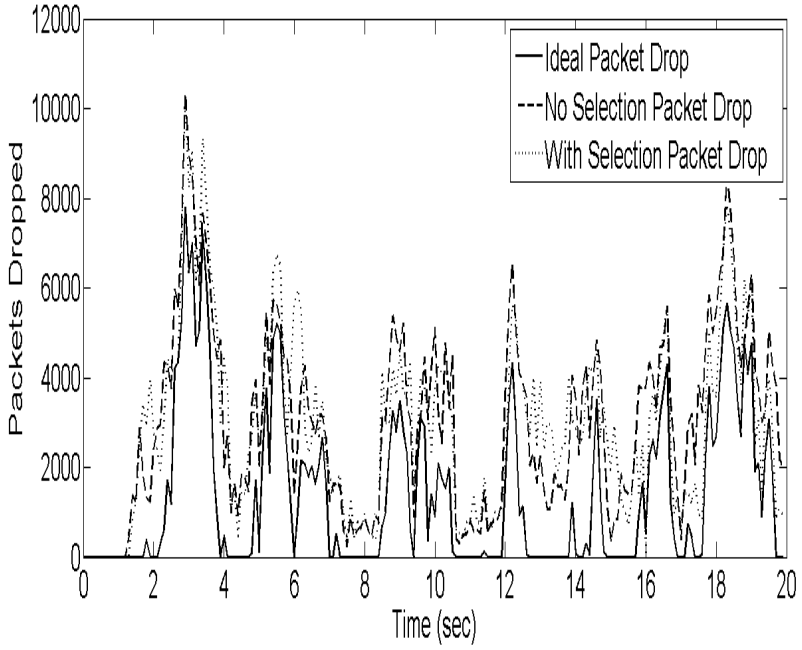


Figure A.3: The figure shows the number of packets dropped every ΔT.

A.5 Summary

We described a selection scheme that can be augmented with the adaptive HRW method. The selection scheme identifies, for an underutilized (overutilized) network processor, the input ports causing the network processor to be underutilized (overutilized). The simulation results show that deploying the selection scheme with the adaptive HRW method further reduces the number of flows remapped while keeping the packet processing workload balanced among the different processing units within the router.

Deploying the selection scheme with the adaptive HRW method requires measuring the processing workload at each network processer from every input port. We

believe that the significant reduction in flow remapping is worth this measurement. Furthermore, these measurements should not have big impact on the performance of current powerful routers.

Bibliography

[1] Medina, A. and Taft, N. and Salamatian, K. and Bhattacharyya, S. and Diot, C., "Traffic Matrix Estimation: Existing Techniques and New Directions", vol. 32, no.4, pp. 161-174, ACM SIGCOMM 2002.

[2] G. Bochmann, et al., "The Agile All-Photonic Network: An Architectural Outline", IEEE QBSC 2004.

[3] I. Khazali and R. Vickers, "The Agile All-Photonic Network: Architectures, algorithms, and protocols", IEEE SSD 2008.

[4] L. Mason, A. Vinokurov, N. Zhao, and D. Plant, "Topological design and dimensioning of agile all-photonic networks", Journal of Computer Networks, vol. 50, pp. 268-287, 2006.

[5] I. Khazali, R. Vickers, "A Reliable Routing Protocol for Agile All-Photonic Networks", IEEE CCECE, 2008.

[6] R. Zhang-Shen and N. McKeown "Designing a Predictable Internet Backbone Network", Third Workshop on Hot Topics in Network (HotNets-III), 2004.

[7] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic Oblivious Routing for Guaranteed Bandwidth Performance", IEEE Communication Magazine, vol. 45, no. 4, pp. 46-51, 2007.

[8] J. A. Fingerhut, S. Suri, and J. S. Turner, "Designing Least-Cost Non-blocking Broadband Networks", Journal of Algorithms, vol.24, no 2, pp. 287-309, 1997.

[9] C. S. R. Murthy and M. Gurusamy, "WDM Optical Networks: Concepts, Design, and Algorithms", Prentice Hall, 2002.

[10] Leon Garcia and Indra Widjaja, "Communication Networks: Fundamental Concepts and Key Architectures", McGraw-Hill, 2004.

[11] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet", Journal High Speed Networks, vol. 8, pp. 69-84, 1999.

[12] D.K. Hunter and I. Andonovic, "Approaches to optical Internet packet switching", IEEE Communications Magazine, vol. 38, pp. 116-122, 2000.

[13] R. Vickers and M. Beshai, "PetaWeb Architecture", Toward Natural Networks: International Telecommunication Network Planning Symposium, 2000.

[14] A. Reinert, B. Sanso and S. Secci, "Design Optimization of the Petaweb Architecture", IEEE/ACM Transactions on Networking, vol.17, pp. 332-345, 2009.

[15] R. Vickers and M. Beshai, "Agile optical-core distributed packet switch", U.S. patent no. 6486983, 2002.

[16] F.J. Blouin, A.W. Lee, A.J.M. Lee, and M. Beshai, "Comparison of Two Optical-Core Networks", Journal Optical Networks, Vol. 1, no. 1, pp. 56-65, 2002.

[17] N. Saberi and M.J. Coates, "Minimum Rejection Scheduling in All-Photonic Networks", IEEE BROADNETS, 2006.

[18] N. Saberi and M. J. Coates, "Scheduling in overlaid star all-photonic networks with large propagation delays", Photonic Network Communications, vol. 17, no. 2, pp 157-169, 2009.

[19] N. Saberi and M. Coates, "Bandwidth Reservation in Optical WDM/TDM Star Networks", IEEE QBSC, 2006.

[20] C. Peng, G. v. Bochmann and T. J. Hall, "Quick Birkhoff-von Neumann Decomposition Algorithm for Agile All-Photonic Network Cores", IEEE ICC, 2006.

[21] I. Khazali and A. Agarwal, "Control Messages Delivery Protocol", IEEE QBSC, 2010.

[22] X. Liu, N. Saberi, M. Coates, and L. Mason, "A comparison between time slot scheduling approaches for all-photonic networks", IEEE International Conference on Information, Communications and Signal Processing, 2005.

[23] A. Vinokurov, X. Liu, and L. G. Mason, "Resource sharing for QoS in agile all photonic networks", in OPNETWORK, (Washington DC), 2005.

[24] R. Vickers and M. Beshai, "Burst Switching in A High Capacity Network", U.S. Patent 6907002, 2005.

[25] B. A. Shirazi, A. R. Hurson, and K. M. Kavi (editors), "Scheduling and Load Balancing in Parallel and Distributed Systems", IEEE CS Press, 1995.

[26] Y. Li and Z. Lan, "A Survey of Load Balancing in Grid Computing", LNCS, vol. 3314, pp. 280285, CIS 2005.

[27] L. Valiant and G. Brebner, "Universal Schemes for Parallel Communications", Proc. of the 13th Annual Symposium on Theory of Computing, pp. 263-277, 1981.

[28] C.S. Chang, D. S. Lee and Y. S. Jou, "Load balanced Birkhoffvon Neumann switches, part I: one-stage buffering", IEEE HPSR, 2001.

[29] I. Keslassy, C. S. Chang, N. McKeown, and D. S. Lee, "Optimal Load balancing", IEEE INFOCOM, 2005.

[30] I. Keslassy, S. T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, N. Mckeown, and N. McKeown, "Scaling Internet Routers Using Optics", SIGCOMM, 2003.

[31] Rui Zhang-Shen McKeown, N., "Designing a Fault-Tolerant Network Using Valiant Load-Balancing", IEEE INFOCOM, 2008.

[32] M. Kodialam, Lakshman, T.V. Orlin, J.B. and Sengupta, S., "Oblivious Routing of Highly Variable Traffic in Service Overlays and IP Backbones", IEEE/ACM Transaction on Networking, vol. 17, no. 2, pp. 459-472, 2009.

[33] M. Kodialam, Lakshman, T.V. Orlin, J.B. and Sengupta, S., "Locally Restorable Routing of Highly Variable Traffic", IEEE/ACM Transaction on Networking, vol. 17, no. 3, pp. 752-763, 2009.

[34] R. Zhang Shen, M. Kodialam, and T. V. Lakshman, "Achieving Bounded Blocking in Circuit-Switched Networks", IEEE INFOCOM, 2006.

[35] D. G. Thaler, C. V. Ravishankar, "Using Name-Based Mappings to Increase Hit Rates", IEEE/ACM Transaction on Networking, vol. 6, no 1, pp. 1-14, 1998.

[36] K. W. Ross, "Hash Routing for Collections of Shared Web Caches", IEEE Network, vol. 11, no 6, pp. 37-44, 1997.

[37] Weiguang Shi, M. H. MacGregor, and Pawel Gburzynski, "Load Balancing for Parallel Forwarding", IEEE/ACM Transaction on Networking, vol. 13, no. 4, pp. 790-801, 2005.

[38] Kencl L., Le Boudec J. Y., "Adaptive Load Sharing for Network Processors", IEEE/ACM vol. 16 no 2, pp. 293-306, 2008.

[39] Weiguang Shi and Lukas Kencl, "Sequence-Preserving Adaptive Load Balancing", IEEE/ACM ANCS, 2006.

[40] S. Kandula, D. Katabi, S. Sinha A. Berger, "Dynamic Load Balancing Without Packet Reordering", ACM Journal of Computer and Communication Review, Vol.37, no 2, pp. 51-62, 2007.

[41] Zhiro Cao, Zhang Wang, and Ellen Zegura, "Performance of Hashing-Based Schemes for Internet Traffic Balancing", IEEE INFOCOM, 2000.

[42] Ju-Yeon Jo and Yoohwan Kim, "Hash-Based Internet Traffic Load Balancing", IEEE IRI, 2004.

[43] Weiguang Shi, M. H. MacGregor, and Pawel Gburzynski, "A Scalable Load Balancer for Forwarding Internet Traffic: Exploiting Flow-Level Burstiness", IEEE/ACM ANCS, 2005.

[44] W. A. Doeringer, et al, "A Survey of Light-Weight Transport Protocols for High-Speed Networks", IEEE Transaction on Communication, 1990.

[45] ITU-T Recommendation Q.2110, "B-ISDN-ATM Adaptation Layer Service-Specific Connection Oriented Protocol", COM-11-R30, (Geneva, Switzerland), 1994.

[46] F. Vacirca, A. De Vendictis, and A. Baiocchi, "Optimal Design of Hybrid FEC/ARQ Schemes for TCP over Wireless Links with Rayleigh Fading", IEEE Transactions on Mobile Computing, vol. 5, no. 4, pp.289-302, 2006.

[47] R. El Azouzi, T. Peyre, and A. Benslimane, "Optimal Design of Hybrid FEC/ARQ Schemes for Real-Time Applications in Wireless Networks", in International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, pp. 11-18, 2006.

[48] R. Puri, K. Ramchandran, and A. Ortega, "Joint Source Channel Coding with Hybrid ARQ/FEC for Robust Video Transmission", in IEEE Multimedia Signal Processing Workshop, 1998.

[49] P. Chou, A. Mohr, A. Wang, and S. Mehrotra, "FEC and Pseudo-ARQ for Receiver-Driven Layered Multicast of Audio and Video", in IEEE Data Compression Conference (DCC), 2000.

[50] J. Zheng, C. Peng, G. v. Bochmann, and T. J. Hall, "Load Balancing in All-Optical Overlaid-Star TDM Networks", IEEE Sarnoff Symposium, 2006.

[51] S. A. Paredes, T. J. Hall, "A Load Balanced Agile All-Photonic Network", IEEE ISCC, 2007.

[52] P. He, G. v. Bochmann, "Routing of MPLS flows over an agile all-photonic star network", IASTED Intern. Conf. on CSA, 2006.

[53] P. He, G. v. Bochmann, "Inter-area shared segment protection of MPLS flows over agile all-photonic star networks", IEEE Globecom, 2007.

[54] J. Moy, "OSPF Version 2", IETF RFC 2328, 1998.

[55] D. Katz, K. Kompella, and D. Yeung, Traffic Engineering (TE) Extensions to OSPF Version 2, IETF RFC 2328, 2003.

[56] K. Kompella and Y. Rekhter, OSPF Extensions in Support og Generalized MPLS, IETF Internet Draft, 2003.

[57] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "OSPF for IPv6", IETF RFC 5340, 2008.

[58] A. Shaikh, J. Rexford, and K. G. Shin, "Load Sensitive Routing for Long-Lived IP Flows", ACM SIGCOMM, 1999.

[59] "WAN Traffic Distribution by Address Size, Fix-West Trace", National Lab. For Applied Network Research (NLANR), 1997.

[60] R. Russo, L. Kencl, B. Metzler, and P. Droz, "Scalable and adaptive load balancing on IBM PowerNP", IBM Zurich Research Lab., Research Report RZ 3431, 2002.

[61] B. Ryu and S. Lowen, "Fractal Traffic Models for Internet Simulation", IEEE ISCC, 2000.

[62] I. Khazali and A. Agarwal, "Flow-Based Load Balancing Architecture for the Agile All-Photonic Network", Journal of Photonic Network Communications, vol. 22, 2011.

[63] Dziong Z., Mason L., "Call Admission and Routing in Multi-Service Loss Networks", IEEE Transactions on Communication, vol. 42, no. 2, 1994.

[64] Nordstrom E., Dziong Z, "CAC and Routing for Multi-Service Networks with Blocked Wide-Band Calls Delayed, part I: Exact Link MDP Framework", European Transaction on Telecommunications, 2005.

[65] Nordstrom E., Dziong Z, "CAC and Routing for Multi-Service Networks with Blocked Wide-Band Calls Delayed, part II: Approximate Link MDP Framework", European Transaction on Telecommunications, 2006.

[66] Nordstrom E., Carlstrom J, "A New Reward Model for MDP State Aggregation with Application to CAC and Routing", European Transaction on Telecommunications, 2004.

[67] Ash G. "Dynamic Routing in Telecommunications Networks", McGraw-Hill: New York, 1998.

[68] H. C. B. Chan, H. M. Alnuweiri, and V. C. M. Leung. "A framework for optimizing the cost and performance of next-generation IP routers". IEEE Journal on Selected Areas in Communications, vol. 17, no. 6, pp. 10131029, 1999.

[69] J. Turner and T. Wolf. "Design issues for high performance active routers". IEEE Journal on Selected Areas in Communications, vol. 19 no. 3, pp. 404409, 2001.

[70] "Cisco Express Forwarding (CEF)". Cisco Systems white paper, 1997.

[71] "A. Asthana, C. Delph, H. V. Jagadish, and P. Krzyzanowski. Towards a Gigabit IP router", Journal of High Speed Networks, vol. 1, no. 4, pp. 281288, 1992.

[72] G. C. Fedorkow. "Cisco 10000 Edge Services Router (ESR) technology overview", 2000.