END-TO-END OPTIMAL ALGORITHMS FOR TRAFFIC ENGINEERING, FAILURE DETECTION AND RECOVERY IN CONNECTIONLESS NETWORKS

by

SUKRUTH SRIKANTHA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2006

ACKNOWLEDGEMENTS

I would like to extend my gratitude to my advisor, Dr. Hao Che, for giving me an opportunity to work on this challenging research topic and providing me the right guidance and support through the course of this research. I am grateful to Mr. Mike O'Dell and Dr. Jeff Lei for serving on my thesis committee.

I am very grateful to my parents and family who have been a constant source of inspiration during my entire academic career. I would like to thank Prathiba, Shreyas and Janakiram for their invaluable help and advice all the way.

November 15, 2006

ABSTRACT

END-TO-END OPTIMAL ALGORITHMS FOR TRAFFIC ENGINEERING, FAILURE DETECTION AND RECOVERY IN CONNECTIONLESS NETWORKS

Publication No. _____

Sukruth Srikantha, MS

The University of Texas at Arlington, 2006

Supervising Professor: Dr. Hao Che

In this thesis we propose a novel scheme to achieve intra-domain Traffic Engineering (TE), Failure Detection and Recovery (FR) in connectionless networks. This scheme addresses rate adaptation, load balancing and stability issues of the OSPF protocol namely, network convergence times and route flapping.

With the current default settings of the OSPF parameters, the network takes several tens of seconds to recover from a failure. The main component in this delay is the time required to detect the failure using the Hello protocol. Route flap is another undesirable phenomenon and needs to be eliminated to achieve greater stability and robustness in computer networks. Also, performing rate adaptation and load balancing at the edge routers without any feedback from the network core is a challenging task.

In this thesis, we address the above issues with a scheme that does not require any modifications to the underlying routing protocols. We focus on the application of a set

of distributed control laws on the edge routers in a connectionless network. The control laws provide optimal data rate adaptation and load balancing where multiple disjoint paths are available between an ingress-egress pair. The control laws require information of whether a forwarding path is congested or not for their traffic engineering operations. We have implemented a source inferred congestion detection scheme, to find congested paths and node/link failures along a forwarding path. The information inferred from this mechanism serves as input to the control laws to provide optimal rate adaptation and load balancing.

The proposed approach endows the network with the important property of stability and robustness with respect to node/link failures. The congestion detection scheme also serves as a failure detection mechanism and thus helps to overcome the IGP convergence problem in OSPF. On the occurrence of a link or router breakdown, the congestion detection mechanism is capable of detecting the failure within a few milliseconds. This information is provided to the control laws, which reroute traffic away from the inoperative node/link to converge to the optimal allocation for the "reduced" network. In comparison, OSPF takes several tens of seconds for the failure detection and convergence process. Hence, this approach helps to get around the IGP convergence problem.

Highly scalable TE and FR features are implemented on edge routers, based on these control laws and the congestion detection mechanism, without the involvement of the routers in the network core. Simulation results are presented to demonstrate the advantages of the proposed approach under a variety of network scenarios. Results show that the convergence time reduces from few tens of seconds to the order of milliseconds. The average throughput is improved considerably due to dynamic rate adaptation and load balancing provided by the control laws.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Internet's communication data paths are controlled by routing protocols. Link state routing protocols (e.g. OSPF [9]), are used in Internet Service Provider networks for intra-domain routing. They are responsible for finding alternative paths/routes in the face of network failure, such as hardware failures, excessive load conditions, incorrect implementations, network upgrades/routine maintenance, etc. In such cases, communication across the network can be partially, or even totally lost until the routing protocols find the next best path for all the routes using the network that has failed.

Improving user performance and making more efficient use of network resources requires adapting the routing of traffic to the prevailing demands. This task is referred to as traffic engineering [1]. Recently, Traffic Engineering (TE) has received tremendous attention in the Internet community. Traffic engineering has a wide scope and covers diverse forms of routing that address survivability, QoS and policy-based routing. However, congestion management and bandwidth utilization are of pressing importance and generally more difficult to address.

In this thesis we propose a novel scheme to achieve intra-domain Traffic Engineering (TE), Failure Detection and Recovery (FR) in connectionless networks. We study the performance improvements that can be achieved through the application of a set of distributed control laws [1] on the edge routers in a network. We have implemented a source inferred congestion detection scheme, to find congested paths and node/link failures along a forwarding path. The information inferred from this mechanism serves as input to the control laws to perform optimal rate adaptation and load balancing. The

congestion detection scheme also serves as a failure detection mechanism and, thus, helps to overcome the IGP convergence and route flapping problems in OSPF.

1.1 Motivation

With the current default settings of the OSPF [10] parameters, the network takes several tens of seconds before recovering from a failure. During these periods of delayed convergence, the end-to-end Internet paths will experience intermittent loss of connectivity, as well as increased packet loss and latency. The main component in this delay is the time required to detect the failure using Hello protocol. Failure detection time can be speeded up by reducing the value of HelloInterval. However, too small a value of HelloInterval will result in an increased chance of network congestion causing loss of several consecutive Hellos, thus leading to false breakdown of adjacency between routers. A number of schemes have been proposed to address the IGP convergence problem in OSPF. Goyal et al. [27] investigate what is the optimal value for the HelloInterval that will lead to fast failure detection in the network while keeping the false alarm occurrence within acceptable limits. Francois et al [26] analyse the various factors that influence the convergence time of intradomain link state routing protocols. They suggest that using more advanced router architecture, higher performance processors, and prioritization techniques the components of the network convergence can be improved further. Choudhury et al. [14] address the issues of fast restoration under failure conditions and improved network scalability and stability. The authors show that special marking and prioritized processing of certain key messages can allow both objectives to be achieved. Section 2.2.4 discusses these schemes and their limitations in more detail.

Another stability issue in IGP is route flapping. Route flapping occurs when a router alternately advertises a destination network first via one route then another (or as unavailable, and then available again in quick sequence). Route flap is an undesirable

phenomenon and needs to be eliminated for more stable and robust networks. Several solutions have been proposed to dampen the effects of route flaps. Ohara et al [13] propose a scheme for damping the route flaps and show how this methodology can solve such problems in OSPF. The authors propose an algorithm by maintaining a figure of merit named penalty for each route, which is incremented each time it flaps. Section 2.2.4 discusses these schemes and their limitations in more detail.

Most of the traffic engineering schemes proposed thus far relate only to connection-oriented networks. Lagoa et al. [2] propose a traffic-engineering scheme for connection-oriented networks that requires feedback about the number of congested links in the forwarding path. A number of schemes have been proposed for connection-oriented networks that do not require any form of explicit feedback from the network, as they can use source inferred information [24].

There is very little work done in the area of traffic engineering in connectionless networks to address the stability issues in OSPF. Almost all schemes proposed so far require some form of explicit feedback from the network. Movsichoff et al. [25] propose a decentralized traffic engineering scheme for connectionless networks, that requires feedback about the number of congested links in the forwarding path.

The motivation behind this thesis lies in the fact that high network availability and fault tolerant behavior are seen as crucial network properties in IP networks. In order to provide high availability, it is crucial that the routing system react quickly to failures and outage with minimal feedback from the network.

1.2 Contribution and Scope

This thesis focuses on traffic engineering, failure detection and recovery in connectionless networks. It involves the implementation of a set of distributed control laws to address stability issues in OSPF. We have implemented a source inferred congestion de-

tection scheme, to find congested paths and node/link failures along a forwarding path. The scheme requires no explicit feedback from the routers in the network core. The information inferred from this mechanism serves as input to the control laws to perform optimal rate adaptation and load balancing.

The novel feature of this scheme involves the implementation of edge-based control for rate adaptation, load balancing and failure detection, using only source-inferred information in a connectionless network. The distributed control laws and the congestion detection mechanism are implemented only on the ingress and egress routers in an autonomous system.

So far, all schemes proposed for connectionless networks, are not edge-based and require some form of feedback from the network core. A number of edge-based control schemes have been proposed, but they can only be applied to connection-oriented networks. Our scheme combines the benefits of both approaches and proposes an edge-based control algorithm that can be applied to connectionless networks.

Our scheme helps the network to converge faster by detecting link/node failure within milliseconds without any change to the underlying routing protocol. The approach does not require any explicit feedback from the network and achieves higher resilience compared to OSPF routing protocol. The control laws re-route the traffic through other available shortest paths within a few milliseconds when node/link failure occurs. Our approach dampens route flapping and is able to dynamically adapt to changing network conditions.

1.3 Organization

The remaining part of this thesis is organized as follows. Chapter 2 gives an overview of the background and literature available in this area of research. Chapter 3 presents our proposed family of control laws and the considerations for implementation

of this scheme. Chapter 4 discusses the experiments conducted and results obtained. Finally, Chapter 5 provides a conclusion and future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 OSPF: The Basic Protocol

In 1987 the Internet Engineering Task Force (IETF) decided to develop a new Intra-Domain routing protocol able to handle the growing routing tables in the evolving IP-based networks, while responding quickly to topology changes and requiring minor protocol traffic. The first version of the OSPF protocol was standardized in 1989. It was one of the first protocols to be fully designed within the IETF and it is still the most deployed intra-domain routing protocol today. Its current version is defined in [9]. Some extensions of the protocol, such as support of QoS and interaction with Border Gateway Protocol (BGP) are standardized in further Request For Comments (RFC).

2.1.1 Brief Overview of Link State Routing

Link state routing protocols consist mainly of two components: a database flooding component and a shortest path calculation component. In order to distribute topological information, link state routing protocols reliably flood a topological database to all routers in a flooding scope. After receiving topological information, routers calculate a shortest path tree (SPT) using a variant of the Dijkstra shortest path calculation to all destinations in the network. The result of this calculation is the next-hop interfaced neighbors to reach all destinations. Because all routers use the same database and the same path computation algorithm, the result is loop-free forwarding information.

Link state routing protocols flood topology information in a robust and loop-free manner. Neighbors describe their databases to each other in the form of link state

advertisements (LSA) and flood the most recent copies of this information. In order to ensure freshness of their link states, both a sequence number and an age are associated with each LSA. Link state routing protocols operate in the software control-plane of routers. The hop-by-hop flooding of link state information is performed by a software process. This is in contrast to the data forwarding plane of routers, which is typically implemented as hardware for fast forwarding of data packets.

Thus, the main tasks of the link state protocol are:

- detect any failure of network element or change in topology,
- distribute topology information to the area,
- calculate routing tables on the basis of topology and routing metrics,
- configure the Forwarding Information Base (FIB).

2.1.2 Detection of Topology Changes: Hello Protocol

The Hello Protocol is responsible for establishing and maintaining relationships between adjacent routers. Each router periodically sends Hello packets on all its outgoing interfaces. The adjacent routers detect these packets. If a router has not received Hello packets from an adjacent router within the 'Router Dead Interval', the link between the interfaces of the two routers is considered down until Hello packets are received again. Considering the crucial timing issues, the interval between the sending of Hello packets as well as the Router Dead Interval must be equal in the whole network. After the detection of a topology change, the information is broadcasted to all neighbors via a Link State Advertisement (LSA).

2.1.3 Distribution and Storage of the Network Map

Each router maintains a complete view of its area's topology. This topology information is stored in a local database of LSA, or link-state database, within each

router. Each LSA represents one link of the network. Adjacent routers synchronize their databases via a reliable (i.e. acknowledged) exchange, or flooding, of Link State Update (LSU), which are bundles of LSAs. To acknowledge receipt of each LSA, the receiving router, bundles LSA acks into Link State Acknowledgement (LS Ack) packets, and sends them to the appropriate neighbour. On receiving an LSU packet, the router processes all the LSAs contained in the packet. Each LSA is classified as new or duplicate, according to the sequence number contained in the LSA. A LSA is considered as duplicate if its sequence number is the same as the one of a matching LSA instance in the router's link-state database.

When a new LSA is received, the database is updated, a SPT calculation is scheduled and the information is broadcasted on all outgoing interfaces (flooded), except the interface on which it was originally received. When a duplicated LSA is received, the LSA is discarded.

After a short period of time, the change in the topology is known to each router in the OSPF area. However, due to the flooding of the LSA on more than one outgoing interface, it is possible that several copies of the same information are processed in the network at the same time. If more recent information is available at a router, the outdated LSA is discarded and the LSA with the more recent information is sent back to the adjacent sending router.

Actual flooding of LSAs may not always happen immediately after LSA processing. Indeed, modern routers employ pacing mechanisms as a form of flow control while sending out OSPF packets. Thus, while processing a LSA, OSPF determines on which interfaces it needs to be flooded out but does not actually send it. The LSA is sent along with the other LSAs to be sent when the timer associated with the interface fires. When processing an LSA, the router has to determine if a SPT calculation is necessary. Not all LSAs indicate a topology change: OSPF requires periodic refreshing of LSAs even

when the topology has not changed. In addition, the SPT calculation is not handled immediately but merely scheduled: it gives the router a chance to receive more LSAs indicating a change and then amortize the cost of the calculation over a number of LSAs. When processing a LSA, the router has to determine if a SPT calculation is necessary.

2.1.4 Calculation of the Routing Table

In a stable state, each router in the area has an identical LSA database. This view of the network map can be seen as a directed graph with nodes representing the routers and edges representing the links of the network. A configurable cost value is associated with the output side of each router interface. From this graph, each router generates a SPT with the router itself as root of the tree. The tree shows the entire path between the router and any destination inside the area. However, only the next hop is used in the forwarding process. Current router implementations use Dijkstra's algorithm or similar algorithms based on Bellmann-Ford's. The OSPF standard [9] also allows the use of multiple cost metrics, depending on different possible service classes, as well as the use of paths with equal costs simultaneously called Equal Cost Multi-Path (ECMP). With ECMP the outgoing traffic is distributed equally on the resulting outgoing interfaces.

2.1.5 Installation of the new routes

Once the routing table is computed, the FIB of the router must be configured. The FIB is used to determine the output interface in the packet forwarding process in a router. Routing protocols like OSPF run on a route processor. It receives LSAs bundled in LSU packets as shown in the figure and processes these to build the link state database. OSPF then uses this link state database to perform a SPT calculation and applies the result to build the FIB. In most modern routers, the FIB is maintained in specialized memory in order to maximize the forwarding performance. By doing so, data packets do

not increase the load of the route processor. Once a data packet arrives on an interface card, the card consults the FIB to determine the next hop and forwards the packet to the corresponding outgoing interface through a switching fabric as shown in the figure. Modern routers have at least two different FIBs: a primary and a shadow FIB. During the writing of the shadow FIB the primary FIB is used to forward the traffic and vice versa.

The following figure shows the processing initiated by the receipt of a LSU

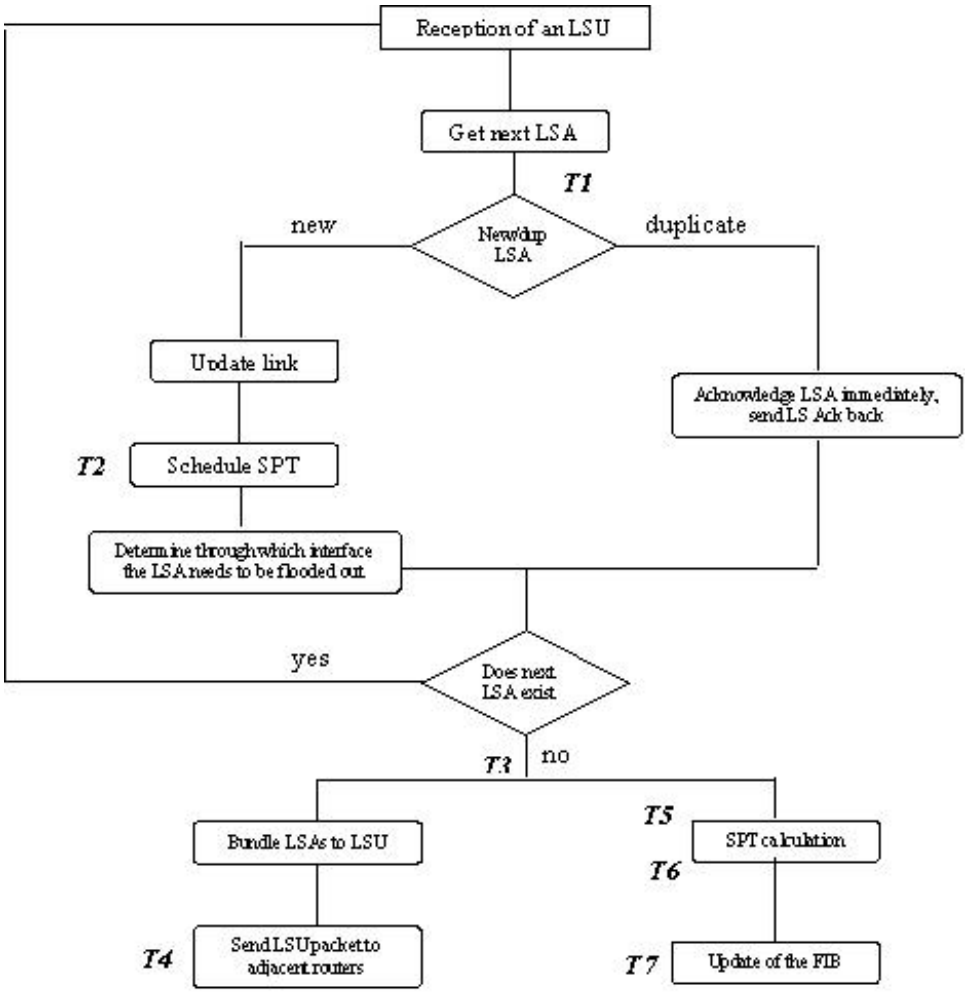

Figure 2.1. Processing initiated by the receipt of a LSU.

The following table shows the time constants in the OSPF protocol.

Table 2.1. Time Constants in OSPF

Name	Typical Value	Description	Fig 2.1
T_{Hello}	10s	Interval between successive Hello packets	
T_{Dead}	4 x T_{Hello}	Router Dead Interval	
T_{lsa}	0.6-1.1ms	Processing of a LSA: check if LSA is new and update link-state database	T3 - T1
$T_{sptDelay}$	5s	Minimum time between database update and start of SPT computation	T5 - T2 $\leq$ TsptDelay
$T_{sptHold}$	10s	Minimum time between consecutive SPT computations	
T_{spt}	O(nlogn)	SPT calculation	T6 - T5
$T_{lsaFlood}$	33ms	LSA flooding time: LSA processing, bundle of LSAs into a LSU and wait for pacing timer to expire	T4 - T1
T_{fib}	100-300ms	Update of the FIB: from end of LSA processing to end of new routes installation	T7 - T5

2.2 Stability Issues in OSPF

High network availability and fault tolerant behavior are seen as crucial network properties in future carrier and Internet service provider IP networks. In order to provide high availability, it is crucial that the routing system react quickly to failures and outages. Basu et al. [12] identify network convergence, route flapping and routing load on processors as indicators of network stability:

2.2.1 Network Convergence

The network convergence time is the time taken by all the OSPF routers in the network to go back to steady state operations after there is a change in the network state (in the absence of any further failures). For example, when a link failure occurs, the network convergence time is the total time taken for all the routers to update their link-state databases to reflect the fact that the failed link is down and to reroute all the traffic-engineered paths (if any) around the failure. The network convergence time is determined by the diameter of the network (maximum number of hops between any two routers), congestion, routing load on processors, and several other factors. A low convergence time indicates a stable network. This is because the network can quickly come back to a steady state when perturbed.

Current link state protocols may require a significant amount of time to converge in certain scenarios. Cain [11] identifies the following as performance problems in current link state routing protocols Failure Detection: Failures may be detected at one or more network layers. Link layer control protocols typically perform link management functions. Most routing protocols contain their own keep-alive protocols for detecting link and node failures. If a link layer detects a failure, failure detection may be instantaneous. However, if routing protocol keep-alive messages are used, then detection time may be in the range of 5 to 40 seconds. This is due to the fact that routing protocol keep-alive messages are typically processed in software and, therefore, can only be sent periodically.

Flooding: Link state updates are processed hop-by-hop by router control plane software. This requires a minimum of 1 to 5 milliseconds per router and is partially dependent on the total processing load of the control-plane of a router. If a router or subset of routers is busy with other processing, link state convergence time may be delayed for all routers within the flooding scope.

Holddown (or SPT hold down): Link state protocols usually implement a holddown time, which is a waiting period to receive multiple LSAs before starting a shortest path calculation. This prevents multiple sequential shortest path calculations when multiple failures occur (thereby causing multiple LSAs to be flooded). Most implementations use a holddown timer value of 1 to 5 seconds.

Shortest Path Calculation: The time required to perform an SPT calculation is highly dependent on the efficiency of an implementation, the processing load of a router's control-plane, and the number of nodes in the network. A reasonable estimate of this time is in the range of 50 to 500 milliseconds.

Installation of routes: This is the time required to propagate new routing information from a router's control-plane to its data forwarding hardware. This time is negligible.

Link state routing protocols are robust for route distribution. However, these protocols do not offer the convergence times needed for high availability and fast recovery. Adding Traffic Engineering extensions to IGPs complicates the issue. In the pure state, IGP protocols send out state-change update messages only when a failure occurs in the network. In the case of IGPs with TE extensions, state-change update messages are also sent out when there is a change in the resources (e.g., link bandwidth) available in the network. Furthermore, when network failures occur, traffic-engineered paths may need to be rerouted. As the amount of control information exchanged and the frequency with which it is exchanged increases, it becomes increasingly likely that the network will be driven to unstable operating regimes. For example, there could be scenarios where a combination of network failures and resource changes lead to bursts of update messages that prevent the network from converging to a stable state.

According to Basu et al. [12] there are two components to the network convergence time when TE extensions are used. First, we have the propagation component, which is the time for the new information to be flooded across the network after a node/link

failure/repair occurs or there is a change in the unreserved bandwidth on a link. This determines the time for all the routers to update their link-state databases. Second, is the reroute component, which is the time taken to reroute all the traffic-engineered paths when a node or a link fails. Even if alternate routes have been pre-computed, there are overheads involved in tearing down the old route and setting up the new one. Clearly, a smaller reroute time indicates a more stable network since it can resume forwarding data traffic with less disruption.

2.2.2 Route Flaps

In computer networking and telecommunications, route flapping occurs when a router alternately advertises a destination network first via one route then another (or as unavailable, and then available again in quick sequence). The central symptom of route instability is the disappearance of a route that previously existed in the routing table. Such routes may disappear and reappear intermittently, a condition referred to as "flapping". Route flapping is caused by pathological conditions (hardware errors, software errors, configuration errors, unreliable connections, etc.) within the network, which cause certain reachability information to be repeatedly advertised and withdrawn. The most common causes of route flapping are configuration errors and intermittent errors in communications links. Route flapping often forces a router to recalculate a new or preferred route to a particular network, while traffic destined for that network is in transit through the router. Route flap is an undesirable phenomenon and needs to be eliminated for more stable and robust networks. Ohara et al. [13] show how flaps adversely affect OSPF routing and communication environment in general. The current OSPF's implicit timer damping function is found inefficient as each flap may lead to a few seconds of lost connectivity.

Route flaps refer to routing table changes in a router, usually in response to a network failure or a recovery. In some sense, the number of route flaps characterizes the intensity of the perturbation in the network. This is in contrast to the duration of the perturbation, which is characterized by the convergence time. For obvious reasons, a large number of route flaps in a short time interval adversely affects network stability [12].

2.2.3 Routing Load on Processors

The routing load on processors is a measure of how much time a router spends in processing control packets from the routing protocol. A high routing load is a possible indication of incipient network instability. This is because a high routing load is either caused by frequent changes in the network state (that generates a heavy volume of control traffic) or due to slow processing of control packets. In either case, ingress queues get filled and incoming packets get dropped. In particular, incoming control packets get dropped which causes timeouts, which in turn generates more control packets and higher routing loads. Clearly, such scenarios can ultimately lead to a network meltdown. Even high-end routers that have separate data and control paths are also subject to this kind of meltdown. This is because higher routing loads can lead to packet drops from the internal queues that are used to store incoming control packets before they are processed by the route control processor [12].

2.2.4 Related Work

Goyal et al. [26] point out that with the current default settings of the OSPF parameters, the network takes several tens of seconds before recovering from a failure. The main component in this delay is the time required to detect the failure using Hello protocol. Failure detection time can be speeded up by reducing the value of HelloIn-

terval. However, too small a value of HelloInterval will lead to too many false alarms in the network which cause unnecessary routing changes and may even lead to routing instability. In their paper, they investigate what is the optimal value for the HelloInterval that will lead to fast failure detection in the network while keeping the false alarm occurrence within acceptable limits. They examine the impact of both network congestion and the network topology on the optimal HelloInterval value. Additionally, they investigate the effectiveness of faster failure detection in achieving faster failure recovery in OSPF networks.

According to the simulation results shown in the paper, the optimal value for HelloInterval for a network is strongly influenced by the expected congestion levels and the number of links in the topology. While the HelloInterval can be much lower than current default value of 10s, they suggest that it is not advisable to reduce it to millisecond range as it will lead to too many false alarms. Further, it is difficult to prescribe a single HelloInterval value that will perform optimally in all cases. The network operator should set the HelloInterval conservatively taking in account both the expected congestion levels as well as the number of links in the network topology.

Francois et al [26] analyse the various factors that influence the convergence time of intradomain link state routing protocols. This convergence time reflects the time required by a network to react to the failure of a link or a router. To characterize the convergence process, they use detailed measurements to determine the time required to perform the various operations of a link state protocol on currently deployed routers. They build a simulation model based on those measurements and use it to study the convergence time in large networks.

They point out that the convergence time can be characterized as D + O + F + SPT + RIB + DD where the detection time (D), the LSP origination time (O) and the distribution delay (DD) are small compared to our sub-second objective. The flooding

time (F) depends on the network topology and thus on the link propagation delays. The SPT computation time depends on the number of nodes in the network, but can be significantly reduced by using an incremental SPT computation. Finally, the RIB time that corresponds to the update of the RIB and the FIB is the most significant factor as it depends linearly on the number of prefixes affected by the change. They suggest that using more advanced router architecture, higher performance processors, and prioritization techniques the components of the network convergence can be improved further.

Choudhury et al. [14] address the issues of fast restoration under failure conditions and improved network scalability and stability. The paper shows that these two objectives are in conflict with each other and improvement in one leads to a degradation of the other. The authors show that special marking and prioritized processing of certain key messages can allow both objectives to be achieved.

Ohara et al [13] propose a scheme for damping the route flaps and show how this methodology can solve such problems in OSPF. The authors propose an algorithm by maintaining a figure of merit named penalty for each route which is incremented each time it flaps. This gives the degree of instability of that route. Penalty is increased by a constant default penalty when a down event occurs for the route. For the routes which are flapping, it is their availability which is kept suppressed rather than their unavailability. When the penalty goes beyond a configured suppress value, the route is kept from being advertised. During the period which goes without a flap, the penalty continues to be decayed exponentially at a rate by which the penalty will be reduced by half in configured half-life seconds. When the penalty falls below a configured reuse value, the route that was suppressed is now advertised and the router originates an AS-External-LSA with age set to 0. The results presented in the paper show considerable improvement over the ones where none of such techniques were applied.

2.3 Traffic Engineering

Traffic engineering involves adapting the routing of traffic to the network conditions, with the joint goals of good user performance and efficient use of network resources.

2.3.1 Need for Traffic Engineering

In some sense, IP networks manage themselves. A host implementing the Transmission Control Protocol (TCP) adjusts its sending rate to the bandwidth available on the path to the destination, and routers react to changes in the network topology by computing new paths. The TCP window flow control algorithms use minimum information from the network as input to allow fully distributed traffic control. In other words, the only needed feedback information for the TCP window flow control is whether the forwarding path is congested or not. This allows the TCP source node to infer path congestion by counting the number of repetitive acknowledgments of the same packet or measuring end-to-end round-trip delay, making TCP a truly end-to-end protocol without the assistance of the underlying internetworking layer infrastructure. This has made the proliferation of the Internet applications at the global scale possible. An excellent example is the fast, ubiquitous adoption of World Wide Web due to its use of TCP as its underlying transport. It has made the Internet an extremely robust communication network, even in the face of rapid growth and occasional failures [14].

However, these mechanisms do not ensure that the network runs efficiently. For example, a particular link may be congested despite the presence of under-utilized links in other parts of the network. For example, real-time applications, such as Voice over IP (VoIP) and video-phone, have stringent delay and delay jitter requirements, which cannot be adequately supported by today's Internet protocols [1].

Improving user performance and making more efficient use of network resources requires adapting the routing of traffic to the prevailing demands. This task is referred

to as traffic engineering [1]. Recently, Traffic Engineering (TE) has received tremendous attention in the Internet community and several IETF drafts have appeared. Typically in a TE approach, the goal is to determine a set of flows with associated paths and bandwidths that meet the optimization criteria for a given input traffic matrix. Traffic engineering has a wide scope and covers diverse forms of routing that address survivability, QoS and policy-based routing. However, congestion management and bandwidth utilization are of pressing importance and generally more difficult to address [15].

Traffic engineering depends on having a set of performance objectives that guide the selection of paths, as well as effective mechanisms for the routers to select paths that satisfy these objectives. Most large IP networks run Interior Gateway Protocols (IGPs) such as OSPF (Open Shortest Path First) that select paths based on link weights. These weights are typically configured by the network administrators and are static. Routers use the OSPF protocol to exchange link weights and build a complete view of the topology. OSPF is limited to routing scenarios that can be specified with a single weight on each link. Also in its basic form, it does not adapt the link weights in response to changes in traffic or the failures of network elements, and the path-selection process does not directly incorporate any performance objectives. This makes traffic engineering with OSPF a challenging task.

2.3.2 Feedback-based flow control

Flow control is the process of managing the rate of data transmission between two nodes. Flow control is important because it is possible for a sending computer to transmit information at a faster rate than the destination computer can receive and process them. This can happen if the receiving computers have a heavy traffic load in comparison to the sending computer, or if the receiving computer has less processing power than the sending computer.

Flow control mechanisms can be classified by whether or not the receiving node sends feedback to the sending node. Feedback based flow control is a congestion avoidance technique in which sources adjust their transmission rate in response to congestion signals sent (implicitly or explicitly) by network gateways. The goal is to design flow control algorithms, which provide time scale invariant, fair, stable, and robust performance.

Feedback-based flow control in wide-area networking suffers from the unavoidable fact that the time constants associated with the adaptive processes are of the order of the large propagation delays, which, when coupled with high speeds, translates to a major limitation. On the other hand, there is a certain broad class of bursty application processes such that, when the demand for bandwidth arises, it persists for time periods, which are comparable to the aforementioned time constants or longer. For users of such processes, feedback-based flow control offers value. Also, besides the direct goal of allocating bandwidth, feedback algorithms are in the related business of providing "backward information propagation," i.e., communicating allocations from the network to the user. Without such information, the user cannot fully exploit the bandwidth allocated to it, no matter how generous, and how efficient and fair the process [7].

2.3.3 Distributed Traffic Control

Distributed traffic control involves feedback laws for distributed control of dynamic systems. The approach is based on instantaneous control, i.e. control at every time step to the underlying system.

2.3.4 Related Work

There is extensive literature on distributed traffic control. In particular, algorithms with a focus on TCP types of traffic were developed, including both empirical algorithms ([17], [18]) and algorithms based on control theory ([7], [19]).

Bonomi et al. [7] propose a class of flow control algorithms for the adaptive allocation of bandwidths to virtual connections (VC) in high-speed, wide-area ATM networks. The feedback rate to the source from the network is parsimonious, with each feedback bit indicating whether the buffer at a distant switch is above or below a threshold. The service discipline at the switch is First- Come-First-Served. The important goal of adaptability aims to make all of the network bandwidth available to the active VC's, even though the number of such VC's is variable over a given range. Each VC has two parameters, one giving its minimum guaranteed bandwidth and the other is the weight for determining its share of the uncommitted bandwidth. Judicious selection of these parameters defines distinctive services, such as Best Effort and Best Effort with Minimum Bandwidth. The paper derives design rules for selecting the parameters of the algorithms such that the appropriate guarantees and fairness properties are exhibited in the dynamical behavior. The systematic use of "damping" in right proportion with "gain" is shown to be a powerful device for stabilizing behavior and achieving fairness. The authors' analyses are based on a simple analytic fluid model composed of a system of first-order delay-differential equations, which reflect the propagation delay across the network. Extensive simulations examine the following: (1) fairness, especially to start-up VC's; (2) oscillations; (3) transient behavior, such as the rate of equalization from different initial conditions; (4) disparate bandwidth allocations; (5) multiple paths with diverse propagation delays; (6) adaptability and robustness with respect to parameters; and (7) interoperability of different algorithms. This paper gives a class of flow control algorithms, which, while quite general by design, is primarily intended for the environment where feedback to the user from the network is parsimonious and in the form of single bits. In the base case the rate of the feedback stream is approximately uniform with the inter-arrival time between bits a parameter, which may range typically from about 0.1 to 1 .0 times the round-trip propagation delay, T. The authors envisage the algorithms

being applied to virtual connections (VC) in wide-area networks, perhaps spanning the continental USA, and hence, propagation delays may be large. The feedback information conveyed by each bit is simply whether the instantaneous queue length at the distant switch is above a threshold or not. Thus the feedback is explicit.

The paper also formalizes certain desirable properties of bandwidth allocations which deal with fairness and guarantees on minimum bandwidth, and identify dynamical behavior which converges to give such properties. The proposed scheme gives a unified framework for designing a broad class of dynamical feedback algorithms which satisfy the aforementioned properties. The paper shows that the systematic use of "damping" is a powerful device for stabilizing behavior and for achieving fairness; when damping is used in the right proportion with "gain", then the disadvantages of damping, namely, reduced dynamic range and sluggish transient behavior, may be effectively nullified. Finally, the authors demonstrate through simulations that, when properly designed, the algorithms demonstrate adaptability and performance, which is fair, stable and responsive. Each VC has two nonnegative parameters associated with its bandwidth allocation, v3 and cj, for virtual connection j. The rate parameter vj is the minimum bandwidth and o3 is the positive weight given to the VC in determining its share of the uncommitted bandwidth. Various services with distinctive characteristics may be designed by appropriate choice of these parameters. The algorithms given in this paper are for VC's which share a common buffer at the network node and the service discipline at the node is First-Come-First-Served.

Simulation results show that there are two distinct operating regimes, Unsaturated and Saturated. In steady state, the former has unutilized network bandwidth, oscillation-free flow and empty buffers. In the latter regime all bandwidth is utilized, and both flow and queue behavior oscillates around and QT, respectively. An important attribute of the algorithms is that as the degree of saturatedness increases, as happens with increasing

number of VC's, the amplitude and energy of the oscillations increase gracefully. That is, there is a very broad range of parameters in the Saturated regime for which performance is acceptable in terms of high utilization and low delay. This robustness is at the core of the adaptability of the algorithms. That is, when the number of virtual circuits J is allowed to vary over a broad range and parameters of ongoing connections are not rescaled with J, performance goals continue to be satisfied. An important goal is to make all of the network bandwidth available to the active VC's, with the individual allocations commensurate with their allocation parameters. However, the algorithm designed in this paper assumes a single path and is not optimization based.

Since flows with different ingress-egress node pairs share the same network resources, the key challenge in the design of DCLs is the fact that there is a high degree of interaction between different flows due to the resource constraints. One approach to get around this is to incorporate a link congestion cost into the overall utility function, which replaces the link resource constraints. Then, the problem is solved using a gradient type algorithm, resulting in families of DCLs that support point-to-point multi-path load balancing for rate adaptive traffic, e.g., Golestani, et al. [3], Elwalid, et al. [21], and Guven, et al. [20].

Guven et al. [20] propose a new architecture for efficient network monitoring and measurements in a traditional IP network. This new architecture enables establishment of multiple paths (tunnels) between source-destination pairs without having to modify the underlying routing protocol(s). Based on the proposed architecture the authors propose a measurement-based multi-path routing algorithm derived from simultaneous perturbation stochastic approximation. The proposed algorithm does not assume that the gradient of analytical cost function is known to the algorithm, but rather relies on noisy estimates from measurements. Using the analytical model presented in the paper the authors prove the convergence of the algorithm to the optimal solution. Simulation

results are presented to demonstrate the advantages of the proposed algorithm under a variety of network scenarios. The focus of this paper is the traffic mapping (load balancing) problem; that is the assignment of traffic load onto pre-established paths to meet certain requirements. The authors propose an asynchronous distributed optimal routing algorithm based on stochastic approximation theory using local network state information. Stochastic Approximation (SA) is a recursive procedure for finding the root(s) of equations in the presence of noisy measurements and is particularly useful for finding extrema of functions. This allows the number of measurements required for estimating the gradient to be reduced greatly, while having approximately the same level of accuracy as the classical finite differences method at each iteration. By reducing the number of measurements a better overall convergence rate is achieved. Simulation results show that the proposed algorithm swiftly and effectively minimize the congestion and distribute traffic load efficiently under dynamic network conditions.

Recently, significant research effort has been made in the design of DCLs with link capacity constraints explicitly taken into account. At the core of this endeavor is the development of DCLs which converge to an operation point where a given global utility function is maximized [1]. This line of research has been proven to be fruitful. Large families of DCLs of this kind are obtained based on the nonlinear programming techniques, e.g., the work by Kelly, et al. [4], Low et al. [6] , La and Anantharam [5], and Kar, et al. [8].

Low et al. [6] propose an optimization approach to flow control where the objective is to maximize the aggregate source utility over their transmission rates. Network links and sources are viewed as processors of a distributed computation system to solve the dual problem using a gradient projection algorithm. In this system, sources select transmission rates that maximize their own benefits, utility minus bandwidth cost, and network links adjust bandwidth prices to coordinate the sources' decisions. The scheme allows feedback

delays to be different, substantial, and time varying, and links and sources to update at different times and with different frequencies. It provides asynchronous distributed algorithms and proves their convergence in a static environment. Measurements are obtained from a preliminary prototype to illustrate the convergence of the algorithm in a slowly time varying environment.

The paper proposes an optimization approach to flow control, where the control mechanism is derived as a means to optimize a global measure of network performance. The authors consider a network that consists of a set of unidirectional links of capacities The network is shared by a set of sources, where source is characterized by a utility function that is concave increasing in its transmission rate The goal is to calculate source rates that maximize the sum of the utilities over subject to capacity constraints. Solving this problem centrally would require not only the knowledge of all utility functions, but worse still, complex coordination among potentially all sources due to coupling of sources through shared links. Instead, they propose a decentralized scheme that eliminates this requirement and adapts naturally to changing network conditions. The key is to consider the dual problem whose structure suggests treating the network links and the sources as processors of a distributed computation system to solve the dual problem using gradient projection method. Each processor executes a local algorithm, communicates its computation result to others, and the cycle repeats. The algorithm takes the familiar form of reactive flow control. Based on the local aggregate source rate each link calculates a "price" for a unit of bandwidth at link A source is fed back the scalar price where the sum is taken over all links that uses, and it chooses a transmission rate that maximizes its own benefit utility minus the bandwidth cost. These individually optimal rates may not be socially optimal for a general price vector i.e., they may not maximize the aggregate utility. The algorithm iteratively approaches a price vector that aligns individual and social optimality such that indeed maximizes the aggregate utility. The algorithm is

partially asynchronous in which the sources and links may compute based on outdated information, they may communicate at different times and with different frequencies, and the communication delays may be substantial, different and time-varying. The authors claim that as long as the intervals between updates are bounded, the algorithm converges to yield the optimal rate. In equilibrium, sources that share the same links do not necessarily equally share the available bandwidth. Rather, their shares reflect how they value the resources as expressed by their utility functions and how their use of the resources implies a cost on others. This could be a basis to provide differentiated services in terms of different rate allocations. The basic algorithm is derived and its convergence proved in a static environment, where link capacities and the set of active sources remain unchanged and seems to track the optimum when network conditions vary slowly. The algorithm presented in this paper requires communication between sources and links so a binary feedback mechanism needs to be implemented.

Vutukury et. al. [16] propose a traffic engineering solution that adapts the minimum-delay routing for a given traffic matrix. The scheme uses the minimum delay routing or optimal routing formulated in Bertsekas and Gallager [22] as the basis for its traffic engineering technique. For a given traffic matrix, multiple flows between each source destination pair are determined using an offline algorithm and are then established using routing parameters. Packets are then forwarded according to routing parameters using hash techniques similar to OSPF-OMP described in [23]. The authors use IETF's Differential Services model to implement the TE system. The hash computation for a packet is done once at the ingress node and inserted into the packet. Within the network, the hash key is directly used to determine the next hop without further hash computation. The use of diff-serv model enables complex operations, such as peeking into the TCP header, to be done only once at the ingress node. Also finer granularity of distribution can be achieved by decoupling hash computation from source-destination address. To

further speed up forwarding, the next-hop structure uses a table with next-hop entries instead of boundary values.

The authors have tested the effectiveness of the scheme through a series of simulation experiments. The delays of single-path routing (SP) are compared with the delays obtained in the proposed traffic-engineering (TE) scheme. Results show that the load that the network can carry is much greater in the TE scheme for a given average delay. At very low loads SP performed better because of the tendency of TE to route along longer than shortest paths under low loads. However, the proposed TE scheme significantly outperforms the single-path routing as the load in the network increases. Because of the use of multi-paths in the TE scheme congestion and, therefore, the delays are reduced. The paper also shows that delays close to minimum-delays can be obtained when a large number of TCP flows are present.

A problem that has not been addressed in this paper is adaptation to traffic fluctuations and link failures. Thus, the scheme is not able to adapt to dynamically changing network traffic conditions. Also it requires changes to the underlying packet structure.

Lagoa et al. [2] address the problem of optimal decentralized traffic engineering when multiple paths are available for each call. More precisely, given a set of possible paths for each call, the paper aims at distributing the traffic among the available paths in order to maximize a given utility function. To solve this problem, a large family of decentralized sending rate control laws was proposed having the property that each of the members of this family "steers" the traffic allocation to an optimal operation point. The approach taken relies on the control theory concept of Sliding Modes. These control laws allow each ingress node to independently adjust its traffic sending rates and/or redistribute its sending rates among multiple paths. The only nonlocal information needed is binary feedback from each congested node in the path. It is also shown that these control laws are robust with respect to failures; i.e., they automatically reroute traffic if

a link failure occurs. The family of control laws that reduces the sending rate oscillation caused by implementation constraints like delays and quantization. The introduced optimization-based approach allows distributed, optimal traffic engineering in the presence of both multiple CoSs and multiple paths. This approach results in a large family of control laws which ensures that the traffic rates between edge nodes converge to a distribution which maximizes a given utility function. Simulation studies showed that the proposed approach leads to high performance in terms of resource utilization, as measured by the utility function. It also demonstrated that the approach can effectively reroute the traffic to an optimal state even in the presence of sudden link failures

Movsichoff et al. [25] address the problem of optimal traffic engineering in a connectionless autonomous system based on Nonlinear Control Theory, the approach taken in this paper provides a family of optimal adaptation laws. These laws enable each node in the network to independently distribute traffic among any given set of next-hops in an optimal way, as measured by a given global utility function. This optimal traffic distribution is achieved with minimum information exchange between neighboring nodes. The proposed decentralized control scheme enables optimal traffic redistribution in the case of link failures.

The problem with the above approaches [2] [25] is that, they require explicit feedback from the network about congestion information for rate adaptation and load balancing. Hence both approaches are not scalable.

This gives a brief overview of the background literature available in this field. We see that most schemes require some form of explicit feedback from the network. Also, they can only be applied to point-to-point multi-path connection-oriented network.

CHAPTER 3

PROPOSED SCHEME

In this chapter we address the problem of Traffic Engineering (TE), Failure Recovery and Detection, Network Convergence and Route Flapping in a disjoint multipath connection-less oriented network.

In this thesis, we address the above issues with a scheme that does not require any modifications to the underlying routing protocols. We focus on the application of the set of distributed control laws discussed in Section 3.2 on the edge routers in a connectionless network. These control laws provide optimal data rate adaptation and load balancing where multiple disjoint paths are available between an ingress-egress pair. They only require information of whether a forwarding path is congested or not, for their traffic engineering operations. We have implemented a source inferred congestion detection scheme, to find congested paths and node/link failures along a forwarding path. The information inferred from this mechanism serves as input to the control laws to provide optimal rate adaptation and load balancing.

The proposed approach endows the network with the important property of stability and robustness with respect to node/link failures. The congestion detection scheme also serves as a failure detection mechanism and thus helps to overcome the IGP convergence problem in OSPF. On the occurrence of a link or router breakdown, the congestion detection mechanism is capable of detecting the failure within a few milliseconds. This information is provided to the control laws, which reroute traffic away from the inoperative node/link to converge to the optimal allocation for the "reduced" network. In comparison, OSPF takes several tens of seconds for the failure detection and convergence

process. Hence, this approach helps to get around the IGP convergence problem. The control laws require only the congestion information along the path. It uses the source inferred congestion detection without any explicit feedback from the network. Congestion is detected by sending echo packets from each source node to every other destination node and measuring the round trip time. This approach is highly scalable since the control laws are implemented only on the edge nodes.

3.1 Overview of the proposed approach

As discussed in the previous chapter, the end-to-end algorithms for traffic engineering proposed so far require some form of explicit feedback from the network. Also, they can only be applied to point-to-point multi-path connection-oriented networks. Feedback based rate adaptation and load balancing suffer from the fact that the time constants associated with the adaptive processes are of the order of the large propagation delays, which, when coupled with high speeds, translates to a major limitation. Several other methodologies have been proposed to address the problem of optimal Traffic Engineering as discussed in Chapter 2.

Our work is based on a family of control laws proposed by Movsichoff et al [1]. The approach is free from the limitations imposed by feedback and is the first work to propose a comprehensive optimal solution in the multi-path environment, using only per path binary congestion information. This scheme is source inferred, hence does not require any explicit feedback from the network core.

Our approach allows both point-to-point multi-path and point-to-multipoint multi-path, making it applicable to a connectionless IP network. It uses multiple source rooted shortest paths found by an underlying intra-domain routing protocol. The only information needed by the control laws for providing rate adaptation and load balancing is the congestion information along the forwarding path. For this, we have implemented a

source inferred congestion detection scheme, which is capable of detecting congestion and node/link failures within a few milliseconds. This scheme enjoys TCP like behavior, as this family of DCLs needs to consider only the two edge nodes in an autonomous system.

The control laws [1] implemented in this thesis, drive the network to an operation point that maximizes a given global utility function; e.g., network revenue. The salient features of this family of control laws include:

- The control laws are implemented only on the edge nodes which allows ingress nodes to independently adjust their sending rates and redistribute traffic among available multiple paths, based on source inferred congestion detection.
- Input to each control law is information about whether the forwarding path in a multi-path environment is congested or not.
- Allows source inferred congestion detection, without the need for explicit feedback from the network.
- Recovers from node/link failure quickly compared to OSPF routing protocol, reducing the network convergence time.
- Enables dynamic multi-path load balancing and rate adaptation, thus, preventing the underlying intra-domain routing protocol to cause route flaps.

3.2 System Model

The scheme operates on top of the existing OSPF protocol and no changes are required to the underlying routing operations. The system has three stages of operation namely multipath computation; source inferred congestion detection and distributed control laws for rate adaptation and load balancing.

In the first stage, the OPSF routing protocol computes multiple shortest paths from ingress nodes to egress nodes. In the second stage, each ingress node periodically sends echo packets to each of its egress nodes to detect possible congested paths. Conges-

tion detection is based on smoothed round trip timeout (SRTT) estimates of these echo packets. This is similar to the mechanism used in the TCP protocol. Source inferred congestion detection and notification using echo packets to infer path congestion is also used to infer possible node/link failures. In the third stage, the control laws periodically calculate the data rate for each of the paths between the ingress-egress pairs using the binary congestion information and adjust the sending rate accordingly. The data rate is increased linearly if the paths are not congested and decreased exponentially in case of congestion.

3.2.1 Notations

We assume a fluid flow model for traffic flow and the only resource taken into account is link bandwidth. We consider a computer network where different types of calls exist. Here "types" denote aggregate of calls between the ingress-egress pair. In this thesis we consider only Best Effort (BE) traffic. The notations are shown in Table 3.1

Table 3.1. Notations

Symbol	Defination
L	Set of links
c_l	Capacity of link l $\in L$
N	Number of types of flows
n_i	Number of paths available for flows of type i
$L_{i,j}$	Set of links used by flows of type i taking path j
s	Service class (=BE)
$x^s_{i,j}$	Total data rate for flows of type i taking path j belonging to class s
$x_{i,j}$	Total data rate for flows of type i taking path j ($=\sum_s x^s_{i,j}$)
x^s_i	Total data rate for flows of type i belonging to class s
x_i	Total data rate for flows of type i ($=\sum_s x^{s_i}$)
$c_{i,j}$	Congestion information for calls $x_{i,j}$ i.e. calls of type i taking path j
$\overline{c_{i,j}}$	Denotes the logical not operation on $c_{i,j}$
$z_{i,j}(t,x)$	Oscillation Reduction function
$X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$	Denotes the vector containing the data rates allocated to different paths taken by calls of type i.

3.2.2 Problem Statement

With the current default settings of the OSPF parameters, the network takes several tens of seconds to recover from a failure. The main component in this delay is the time required to detect the failure using the Hello protocol. Route flap is another undesirable phenomenon and needs to be eliminated to achieve greater stability and robustness in computer networks. Traffic Engineering extensions to IGPs complicates the issue. With TE extensions, state-change update messages are also sent out when there is a change in the resources (e.g., link bandwidth) available in the network. As the amount of control

information exchanged and the frequency with which it is exchanged increases, it becomes increasingly likely that the network will be driven to unstable operating regimes.

3.2.3 The Solution

We aim to reduce the network convergence time by detecting node/link failure instantly and re-routing the traffic away from broken node/links. Our congestion detection mechanism is source inferred operating on the ingress and egress nodes. The proposed mechanism treats congestion and node/link failures in the same way. Hence, we do not rely on OSPF to detect congestion and node/link failures through Hello packets. The control laws provide rate adaptation and load balancing functionality and are responsible for re-routing traffic away from the congested or broken path, based on the information supplied by the congestion detection mechanism. This is the first work that applies binary-feedback-based control laws to study how it affects throughput performance in a connectionless network.

Our approach can be divided into three phases, which are inter-dependent. The MultiPath Computation is performed by modified OSPF routing protocol, which computes all the parallel shortest paths for the set of ingress-egress pairs. The computed paths are used by both the Source Inferred Congestion Detection and the Distributed Control Laws.

3.2.3.1 Multi-Path Computation

The OSPF routing protocol calculates the multiple shortest paths available from each node to every other node in the network using Dijkstra's algorithm. Our scheme uses the available multiple shortest paths between ingress-egress nodes to monitor the network and distribute the traffic along these paths. The paths between each ingress-egress node may be disjoint i.e. no common links or they may share some links.

3.2.3.2 Source Inferred Congestion Detection

The congestion detection mechanism used in this thesis is source inferred and has a TCP like behavior. No explicit feedback is required from the network. Each ingress node sends echo packets to each of its egress nodes via all shortest paths identified by the multi-path computation operation. Echo packets are sent at regular intervals of T time units. Echo packets are sent on the paths j = 1,2,..., n_i, for all ingress-egress pairs i, where i = 1,2,...,n on a given network topology. On receiving an echo packet, the egress node sends back an echo-ack along the same path on which it was received.

Figure 3.1 describes the congestion detection algorithm.

```
for each ingress node
   do for each egress node
      do for each path between ingress-egress pair
         do seqNum <- seqNum + 1
            if srtt = 0 then
               srtt <- rtt
               rttVar <- rtt / 2
            else
               rttVar <-((1 - beta) * rttVar) +
                        (beta * abs(srtt - rtt))

               srtt <- expr ((1 - alpha) * srtt) +
                             (alpha * rtt)
            end if

            rt_timeout <- srtt + kValue * rttVar
            set round trip timeout for the echo
packet
            send the packet to the egress node
         end for
      end for
   end for
end for

event: receive a probe packet
if seq_num = expectSeqNum then
   expectSeqNum <- expectSeqNum + 1
   congestion_Id <- 1

elseif seq_num > expectSeqNum
   expectSeqNum <- seq_num + 1
   congestion_Id <- 0
end if
```

Figure 3.1. Congestion Detection Algorithm.

Each echo packet has a unique sequence number field used for identification. The sequence number is included in the echo-ack packet returned by the egress node. Round trip time measurements are made for each echo packet. Congestion detection is based on smoothed round trip timeout (SRTT) estimates of these echo packets. The round trip time for the echo packet is estimated based on measured RTT values for previous packets. A timeout can occur under two circumstances. The first is if there is congestion in the

path and the packet is not received within the timeout period, and the second if there is a node or link failure in the path. Our approach does not differentiate between congestion and link/node failure. It and treats both as congestion, and sets the congestion flag along that path when a timeout occurs.

3.2.3.3 Distributed Control Laws

Distributed control laws run on the edge nodes. This phase uses the path computed in the MultiPath Computation phase and the congestion information gathered during the Source Inferred Congestion Detection phase. Optimal End-End control laws are used to maximize the resource utilization by balancing the load on the available paths and re-routing the traffic in case of link failure or congestion through other available paths.

We have implemented the control laws proposed by Movsichoff et al. in [1]. Here, we give a brief overview of these control laws. Consider a computer network whose set of links is denoted by L and let C_l be the capacity of link $l \in L$. Let n be the number of types of calls, n_i be the number of paths available for calls of type i and $L_{i,j}$ be the set of links used by calls of type i taking path j. For simplicity, we restrict ourselves to the point-to-point multi-path network. Our aim is to develop data rate adaptation laws that maximizes the given utility function of the form:

$$U(x) = \alpha \sum_{n}^{i=1} U_i(X_i) = \alpha \sum_{n}^{i=1} U_i(x_{i,1}, x_{i,2}, x_{i,3}), ...x_{i,n})$$

where α is a positive scaling factor, subject to network capacity constraints

The Class of Service (CoS) is called Best Effort (BE), which does not have any service requirement.

The utilization is measured by the function

$$U(x) = \sum_{n}^{i=1} 0.1log(0.5 + \sum_{n_1}^{j=1} x_{i,j})$$

n_i is the number of paths available for calls of type i. The term 0.5 is included to avoid an infinite derivative at 0 data rate. The control law for Best Effort (BE) calls is given as follows:

$$x_{i,j} = z_{i,j}(t,x)[(1 - e^{-0.1(\sum_{n_i}^{j=1} x_{i,j}+0.5)^{-1}} - (1 - c\bar{g}_{i,j})]$$

The control laws on edge routers increases the data rate linearly along the path if there is no congestion according to the above formula. In case of congestion, the control laws reduces the data rate exponentially along the path. The data rate calculated is based on the sum of the data rates for a particular "type" of traffic, congestion information along the path and oscillation reduction function ($z_{i,j}$ value). $z_{i,j}$, oscillation damping function is calculated every time congestion occurs. The increase and decrease in data rate is controlled by $z_{i,j}$ reducing oscillation in throughput.

An oscillation reduction scheme as in [2] is used where $z_{i,j}$ is computed as

$$z_{i,j}(t) = w(t - t0)$$

where

$$w(t) = 1.8(0.25 + 0.65t)$$

It has a big impact on performance and helps to reduce the oscillation. The maximum value allowed for the time-varying $z_{i,j}$ = w(0) = 2.25.

Path rate is calculated every t time interval for dynamic rate adaptation. The time interval is chosen to be very small in the order of milliseconds. Figure 3.2 shows the algorithm for rate adaptation and load balancing.

```
for each flowType
   do sum <- 0.0
end for
for each flowPath
   do sum <- sum + pathRate[flowType[flowPath]]
      expo <- exp[-0.1 * [1 / sum + 0.5]]
      if congestion_Id[flowType[flowPath]] == 0 then
         congTime <- now
         timeDiff <- congTime - prevCongTime[flowType[flowPath]]
          zValue[flowType[flowPath]] <- 1.8 *
                                        (0.25 + pow(0.65, timeDiff))
          prevCongTime[flowType[flowPath]] <- congTime
      end if
      rate_diff <- (zValue[flowType[flowPath]] *
                   ((1 - expo) -
                   (1 -  congestion_Id[flowType[flowPath]]))
      pathRate[flowType[flowPath]] <- pathRate[flowType[flowPath]]
                                      + rate_diff
      if pathRate[flowType[flowPath]] < 0 then
         pathRate[flowType[value],out_val] <- 0
      end if
end for
```

Figure 3.2. Algorithm for Rate Adaptation and Load Balancing.

3.2.3.4 Congestion and Link Failure

The control laws presented in this thesis excel at re-routing traffic upon congestion or a failure in a node or link. The control laws will reroute the traffic through other available shortest paths as the path rates are updated. In our approach link failures will be seen as a case of congestion. Upon node/link failure, timeout occurs and the congestion flag will be set. The control laws recalculate the date rate and the traffic can be pulled back to zero immediately. The distributed control laws work independent of the congestion detection mechanism.

3.2.3.5 Implementation Consideration

The new family of DCLs used in this thesis provides the mathematical foundation which allows the use of source inferred congestion detection and notification to maintain layer abstraction. Our approach allows the rate control to be coupled with the congestion detection mechanism. The implementation of the DCL in this family needs to consider only the two end nodes, provided that a source inferred congestion detection and notification scheme is available.

There are two main components in the implementation of the proposed approach: the implementation of the control laws on the edge nodes (router) and the design and implementation of source inferred congestion detection and notification mechanisms. The implementation of the DCL control plane and data plane functions in the edge nodes or end hosts are similar to the one described in [25]. In an edge node, the control plane involves the implementation of distributed control laws for Best Effort traffic based on flow aggregation method mapped to a multipath. The rate of the flow aggregate is updated regularly by the distributed control law, and the information is sent to the data plane i.e. the rate $x_{i,j}$ is updated regularly and is passed to the data plane, where j=1, 2,, n_j are the number of paths available. These control plane functions can be implemented easily in software.

The data plane involves balancing the load dynamically for each flow aggregate and a measurement mechanism to keep track of the current rates for each flow aggregate. Upon receiving an update for a flow aggregate from the control plane, its corresponding dynamic load balancer limits the aggregate rate to the new x_i. Then, it sends the limited rate $x_{i,j}$ to different multiple paths based on x_i values. The data plane functions can also be implemented easily in software.

The only information required by the control laws is congestion notification. We adopt an echo packet timeout mechanism as described in the above algorithm to set the congestion bit,$c_{i,j}^{-}$. The binary value of $c_{i,j}^{-}$ indicates the presence or absence of congestion. If the bit is set, it indicates congestion.

Thus, we have proposed a novel approach for End-to-End Optimal Algorithms in presence of multiple shortest paths. The algorithm engineers the traffic and recovers from the failure simultaneously, using only source-inferred information. It can also overcome the problems of route flaps and delayed network convergence of the OSPF routing protocol. Our algorithm detects node/link failure within a few milliseconds and hence achieves network convergence much faster compared the OSPF routing protocol. This is the first work that applies a binary-feedback-based control laws to study how it affects throughput performance in a connectionless network.

CHAPTER 4

EXPERIMENTS AND RESULTS

This chapter describes the implementation details and shows the performance results from simulations. We performed experiments by building a simulation model of the proposed scheme. The experiments are designed to test how the proposed approach performs in a network where multiple paths are available between each ingress-egress pair, and how it compares with the existing system. We show that our scheme achieves stable load balancing of traffic over multiple paths and that the network converges faster than the underlying routing protocol in case of node or link failure.

4.1 Simulation Model

In this thesis, we used NS2 (Network Simulator-2) v2.29 to conduct the experiments. NS2 is an object-oriented, discrete event driven network simulator developed at UC Berkeley. It separates the data path and control path implementations and is written in C++ and Object Oriented TCL (OTcl).

We have also added two modules to NS2 namely EOTE (Edge Optimized Traffic Engineering) Multipath Rate Adaptation module and EOTE Congestion Detection. We have changed Link State Protocol implementation to behave like OSPFv2 and modified Multipath Classifier module for data plane load balancing.

4.1.1 EOTE Congestion Detection

EOTE is an agent that resides on the egress and ingress nodes and is responsible for providing congestion information to the control laws. At the ingress node, the agent

sends an echo packet at regular time intervals to each of its egress nodes through all the available shortest paths. The shortest paths are calculated by the underlying routing protocol. When the EOTE at the egress receives an echo packet, it checks the path from which the echo packet arrived and sends it back to the ingress node via the same path. The intermediate nodes have no knowledge of the congestion detection process and forward the echo packets in the same way as data packets. Upon receiving the echo packet back, the ingress node measures the round trip time and estimates the SRTT (smoothed round trip timeout) for the next echo packet. The timeout of the echo packet determines the presence of congestion. If the packet returns before the timeout, the congestion bit is not set. The congestion bit $c_{i,j}^{-}$ for a particular path is set, if timeout occurs for that path. Link or node failures and congestion are viewed as the same in our approach. This process is implemented on Otcl in NS2.

4.1.2 EOTE Multipath Rate Adaptation Module (Control Plane function)

The data rate $x_{i,j}$ is calculated regularly at ingress node. This is a control plane function and is implemented in OTcl. Different traffic "types" = 1, 2,.n are stored in a hash array. For each flow type, different flows (paths) are stored in a hash array, The flow path hash array will be an index of the flow type hash array. The data rate $x_{i,j}$ and the oscillation-damping factor $z_{i,j}$(t,x) are stored for each path. The rate adaptation is implemented using a timer at ingress node. The data rate is increased linearly if there is no congestion. Upon node/link failure or congestion, the congestion bit gets set and the data rate is reduced exponentially. Every T seconds the rate adaptation function is called which dynamically updates the rate for each flow. It then notifies the data plane which can either increase or decrease its traffic rate based on the information supplied.

4.1.3 Multipath Classifier (Data Plane function)

The Data plane function is implemented in C++ for faster data process. The multipath classifier sends the traffic through all the available paths in a round-robin fashion according to the data rate calculated in control plane. In case of link/node failure the classifier steers the traffic away from the broken link/node. Every time a packet comes to the classifier, it finds one of the available next hops for the packet. The classifier gets the congestion information calculated by the source inferred congestion detection module. If the congestion bit is set for any path, the classifier will find the next available path and send the packet through the non-congested path.

4.2 Network Topology

The network topology on which our scheme is simulated is shown in fig. In our simulation we have assumed 25 nodes out of which 3 are ingress nodes and 4 are egress nodes. Nodes 9, 23 and 24 are ingress nodes and 9, 10, 22 and 23 are egress nodes. Here nodes 9 and 23 act as both ingress and egress. Nodes 0, 1, 6, 7 and 8 are CBR traffic sources. Nodes 1, 2, 3, 4 and 5 are destination hosts.

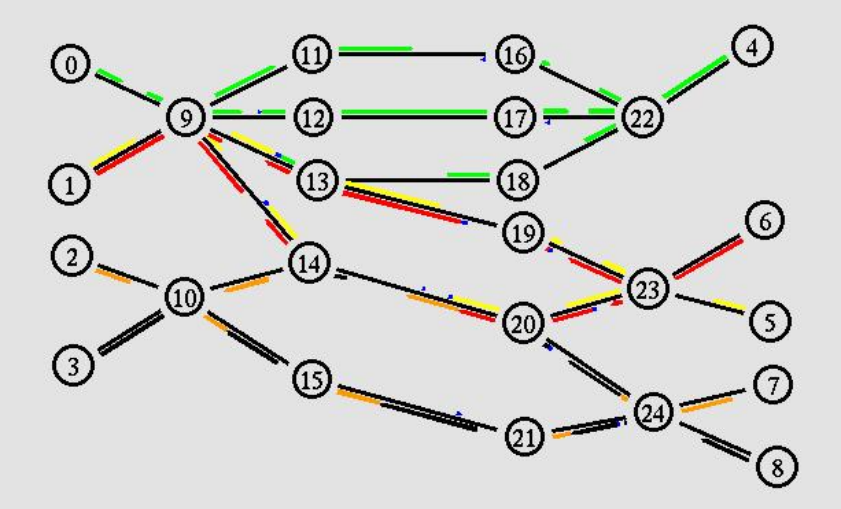

Figure 4.1. Network Topology.

For the above topology shortest paths are listed in Table 4.1. These paths are computed by the OSPF routing protocol during the Multipath computation phase.

Table 4.1. Table of Multiple Shortest Paths

	Type 1	Type 2	Type 3	Type 4
Ingress	9	9	23	24
Egress	22	23	9	10
Shortest Paths	9-11-16-22 9-12-17-22 9-13-18-22	9-13-19-23 9-14-20-23	23-19-13-9 23-20-14-9	24-20-14-10 24-21-15-10

The following are the control law parameters used in our simulation. Rate Adaptation interval = 50ms $z_{i,j}$ is computed as $z_{i,j}$(t) = w (t - t_0)

where w(t) = 1.8 (0.25 + 0.65^t) where t is the current time and t_0 is the time at which the congestion was detected for the first time along a particular path.

4.3 Throughput Analysis

Table 4.2 shows the throughput for all the traffic sources using our scheme. The figure shows the throughput from different source and destination when the network is performing normally without any failures. We can see from the graphs that the aggregate throughput achieved varies around optimal throughput. The different scenarios shown here include multiple paths between different hosts and also include the case where common path are shared among different sources and destinations.

Table 4.2. Average Throughput

Type	Average Throughput (MBPS)
1	1.7
2	1.2
3	1.5
4	2.3

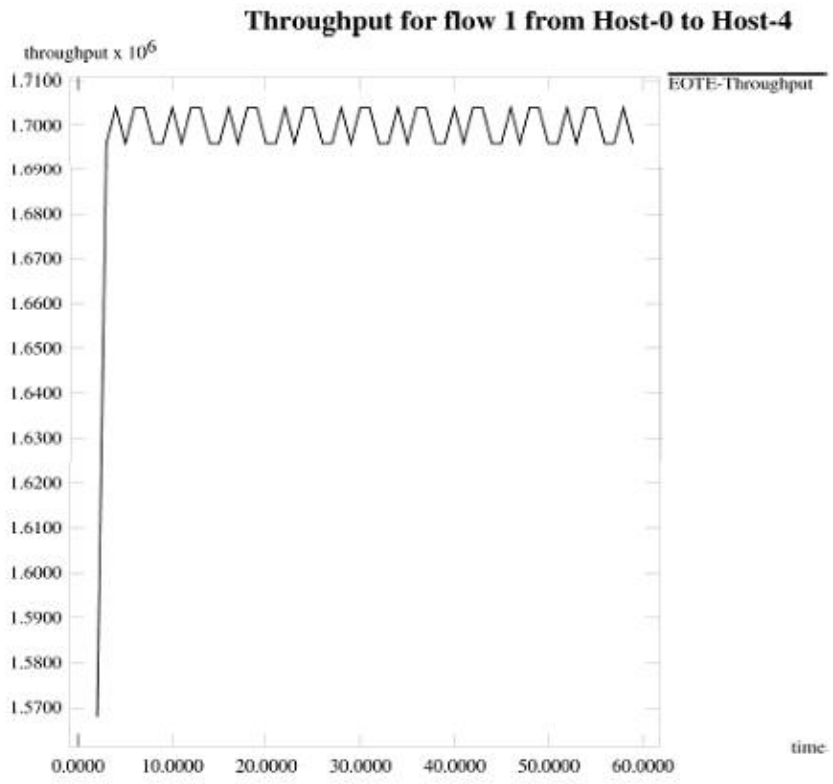

Figure 4.2. Throughput-Host 0 to Host 1.

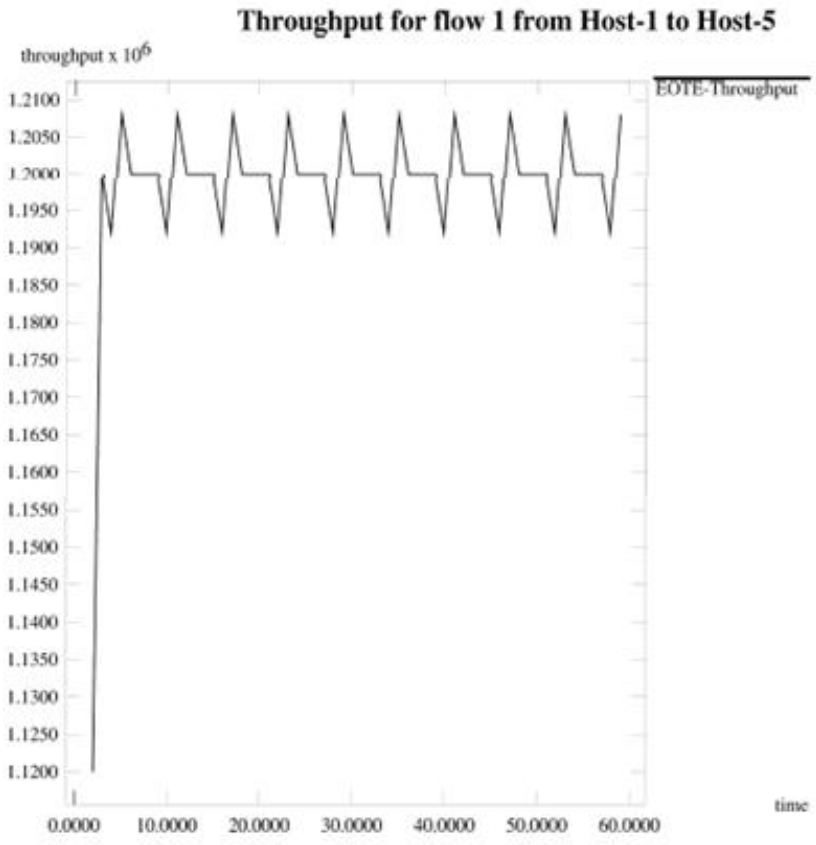

Figure 4.3. Throughput-Host 1 to Host 5.

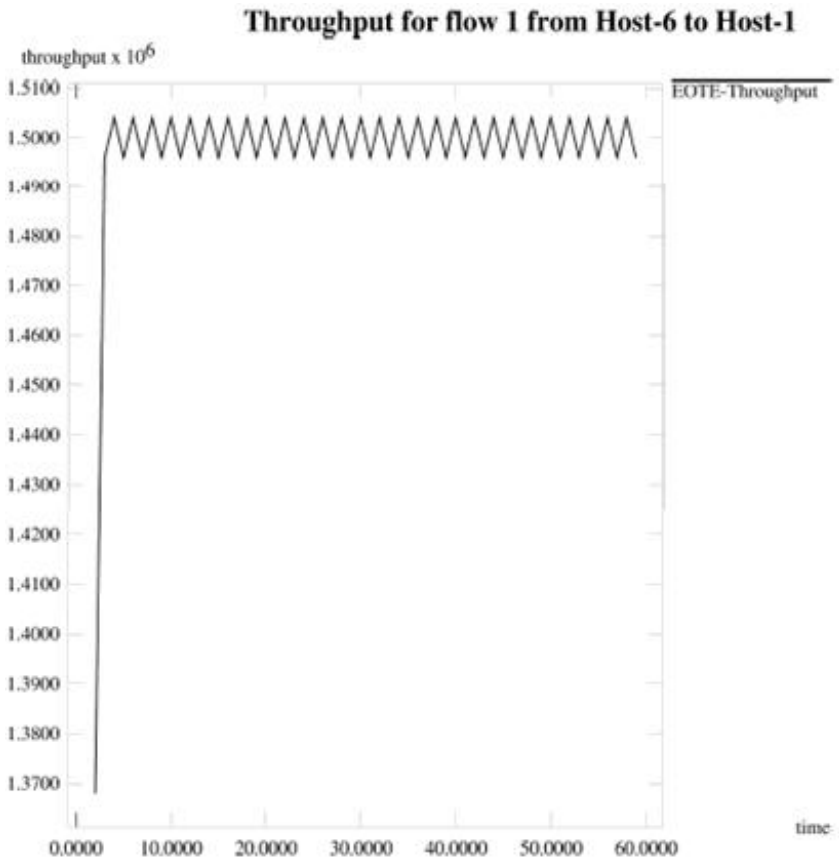

Figure 4.4. Throughput-Host 6 to Host 1.

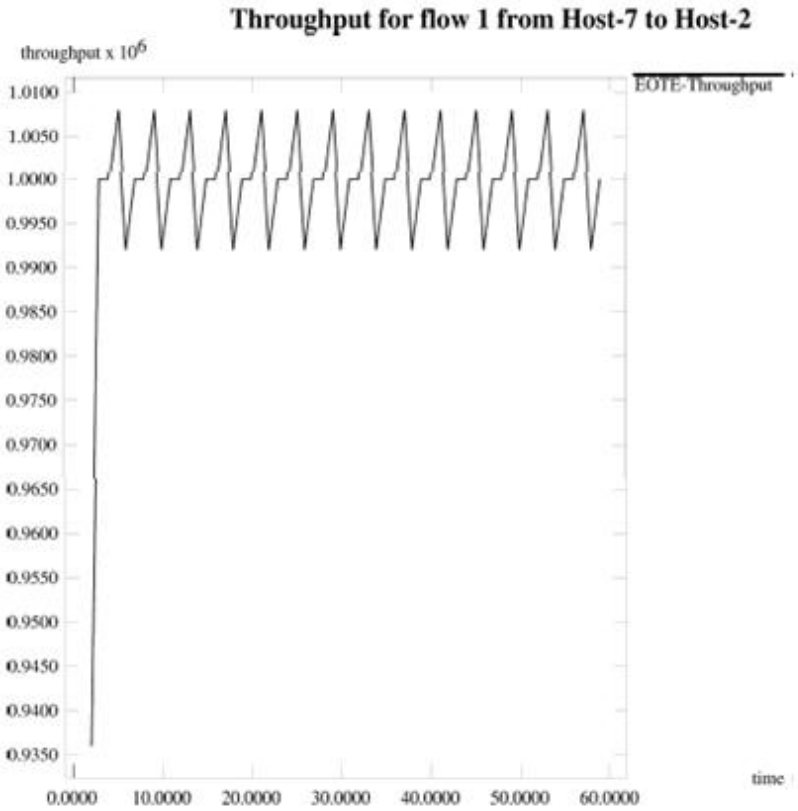

Figure 4.5. Throughput-Host 7 to Host 2.

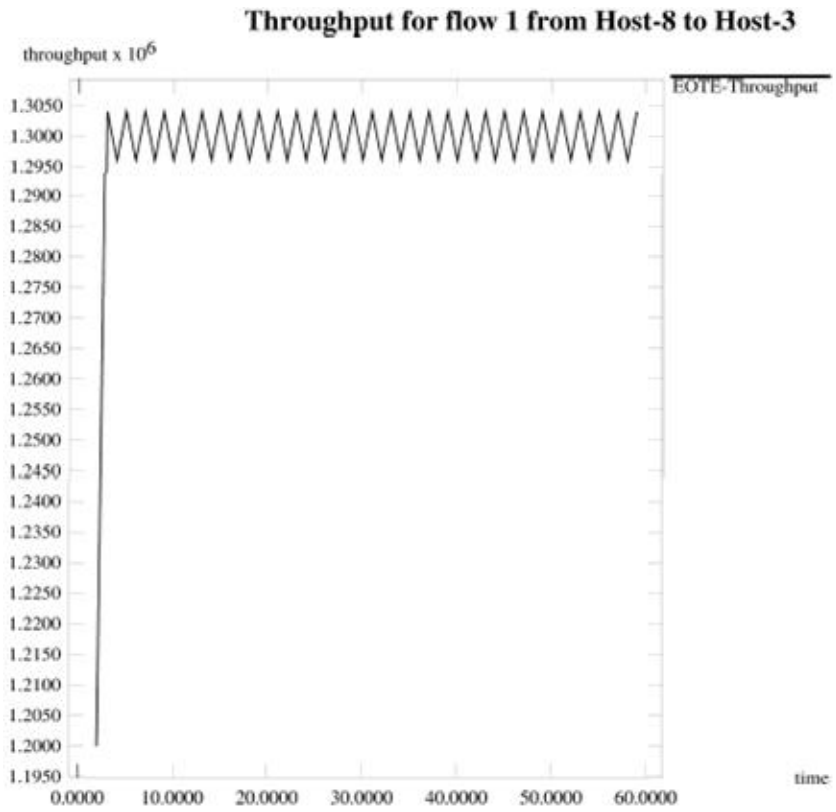

Figure 4.6. Throughput-Host 8 to Host 3.

4.4 Loss Ratio Analysis

The following figure shows the results for the loss ratio seen in the network using our scheme. The network is performing normally without any failures. We measure the packet loss by measuring the difference between the number of packets sent and number of packets received. Here we observe that the packet loss ratio recorded is very low for all types of traffic.

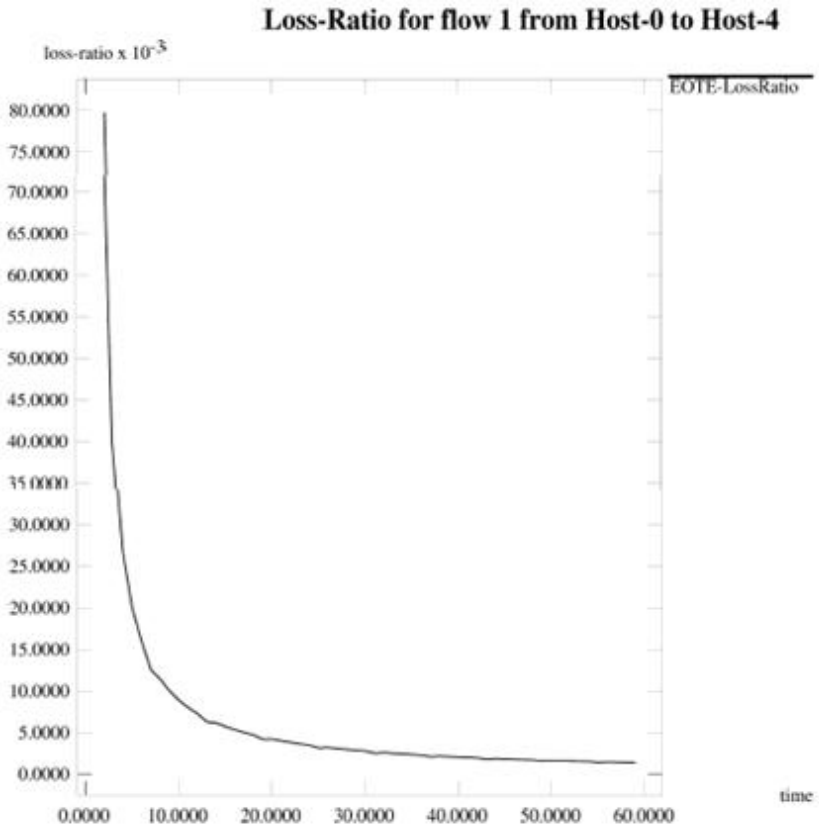

Figure 4.7. Loss Ratio - Host 0 to Host 4.

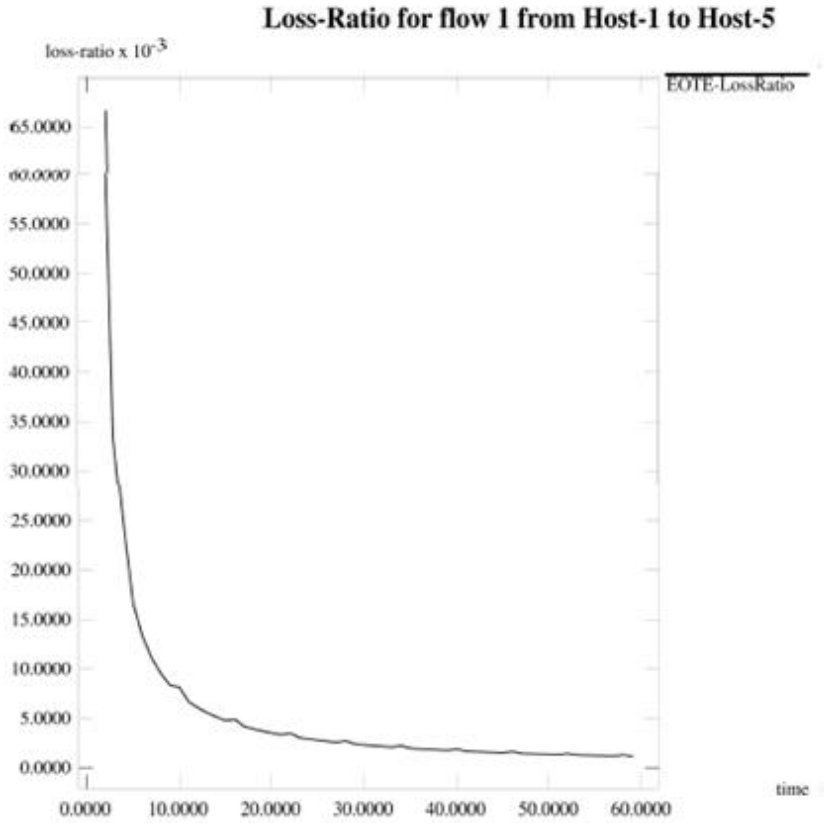

Figure 4.8. Loss Ratio - Host 1 to Host 5.

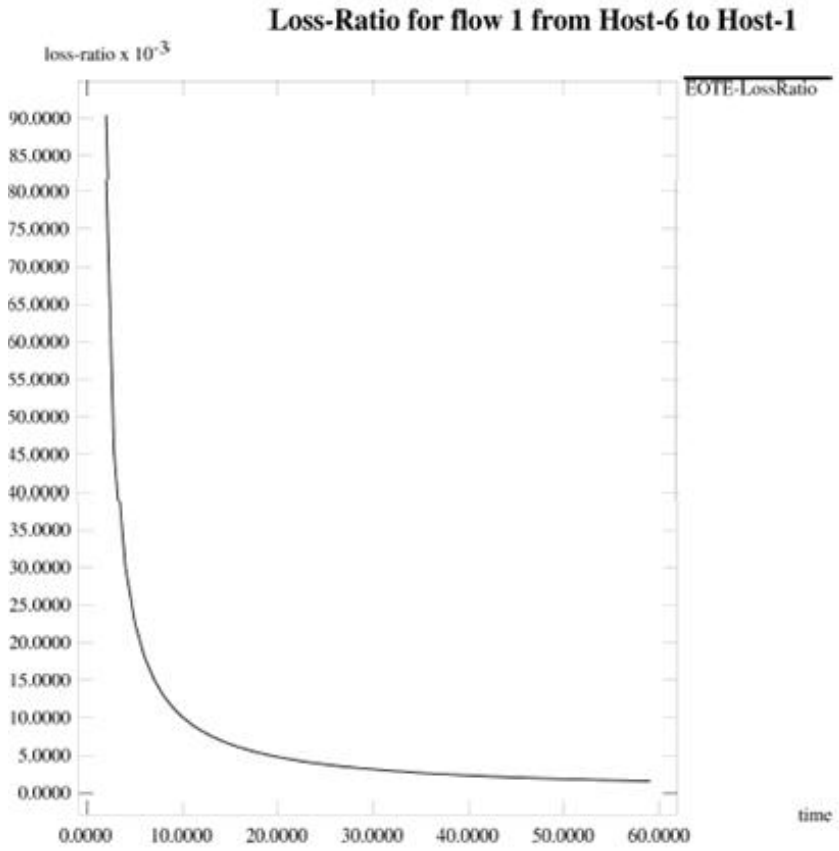

Figure 4.9. Loss Ratio - Host 6 to Host 1.

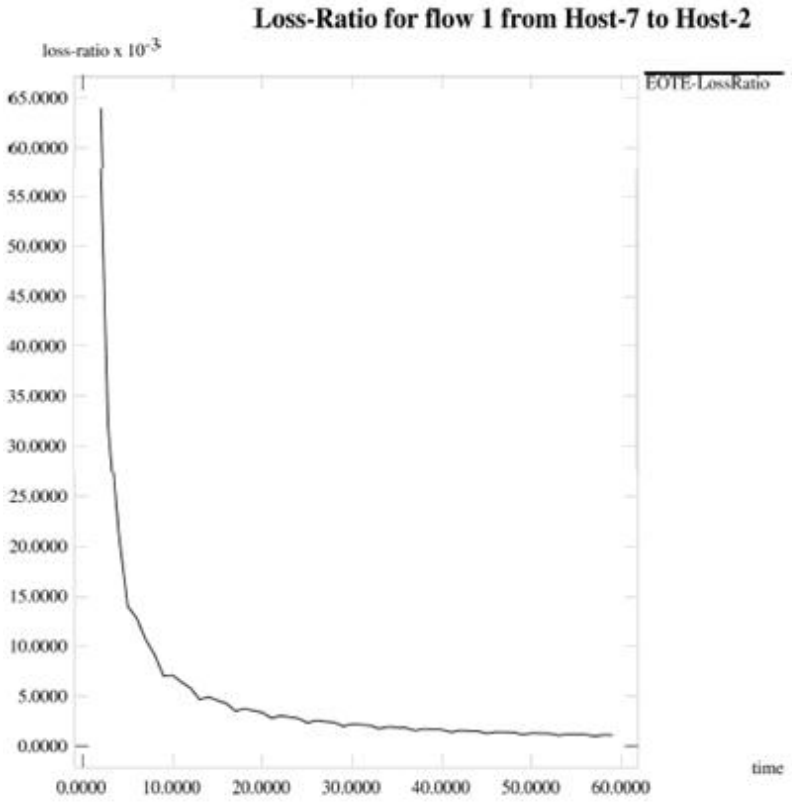

Figure 4.10. Loss Ratio - Host 7 to Host 2.

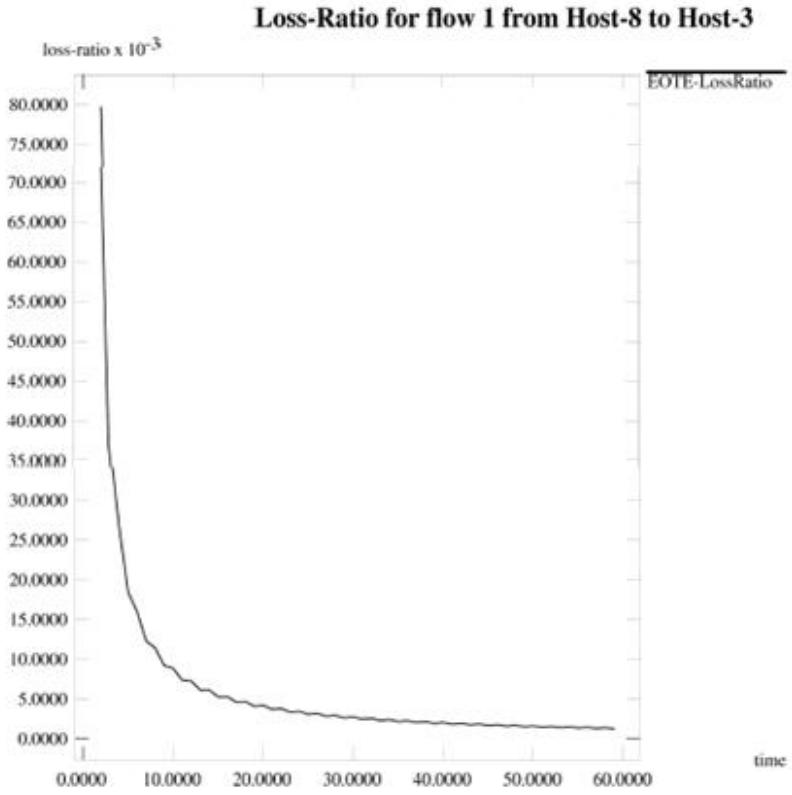

Figure 4.11. Loss Ratio - Host 8 to Host 3.

4.5 Link Failure Analysis

The link between node 12 and 17 is broken at the 15^{th} second of the simulation as shown in Fig below.

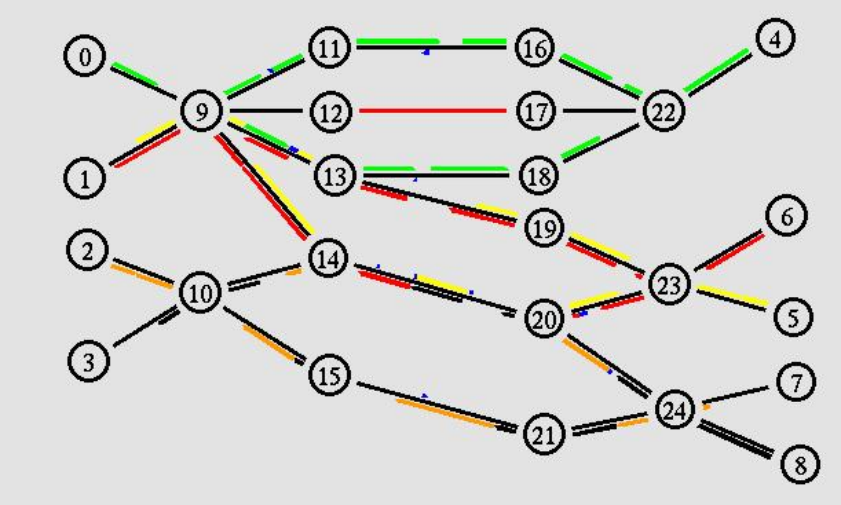

Figure 4.12. Network Topology.

We have seen the results of our scheme in the case of normal network behavior, we now compare our scheme against a scenario where OSPF alone is used to perform multipath routing in the same network.

Our scheme detects the link failure and completely seizes the sending rate through the broken path in 200ms whereas OSPF takes 10 seconds. Hence our approach significantly improves the throughput gain compared to OSPF by re-routing the traffic away from the broken path.

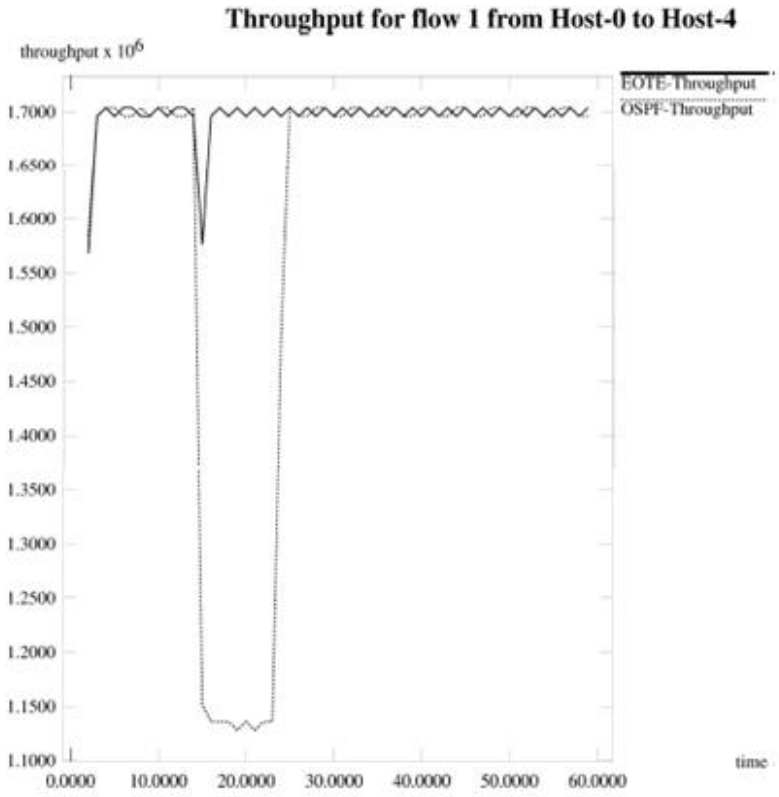

Figure 4.13. Throughput-Host 0 to Host 4.

The following graph shows the loss ratio comparison between OSPF and our scheme. It can be seen that the loss ratio for our scheme is 1.1% where as it is 13.5% for OSPF. This is observed because our approach detects the congestion immediately and steers the traffic away from congested/broken path.

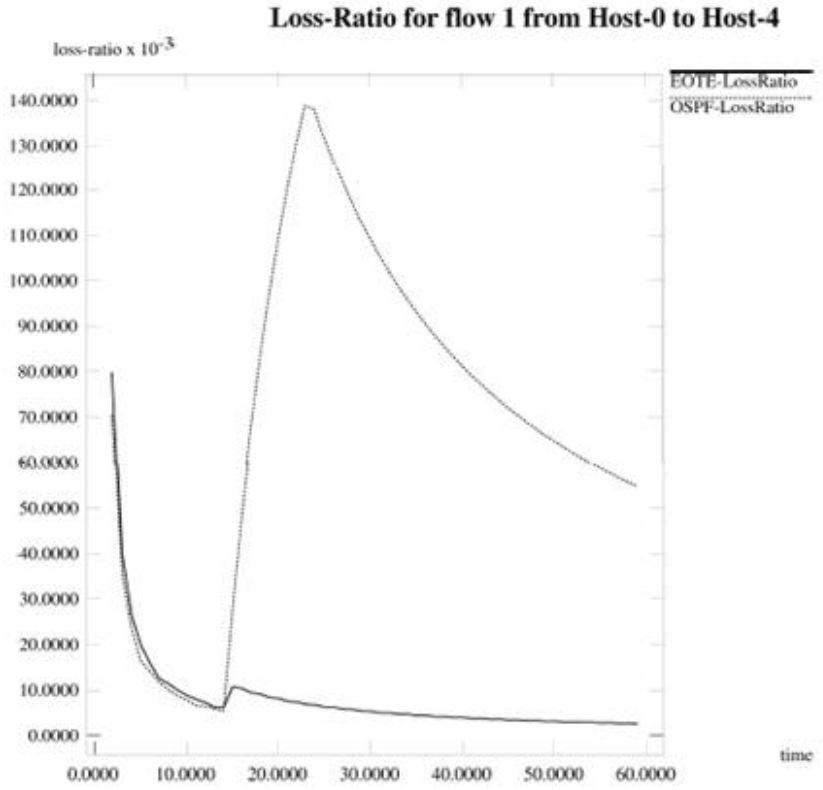

Figure 4.14. Loss Ratio - Host 0 to Host 4.

4.6 Route Flapping Analysis

We compare route flapping effects of OSPF and our scheme under similar network conditions. We can observe that OSPF takes considerable amount of time in case of link/node failure in the previous section, but it detects immediately if the link comes up (good news travels faster). Route Flapping occurs if the link goes up and down frequently. The graph below shows that OSPF detects immediately, when the link comes up but takes 10 seconds again if it goes down. This behavior reduces the throughput and increases the loss in case of OSPF proportional to number of times the route flaps.

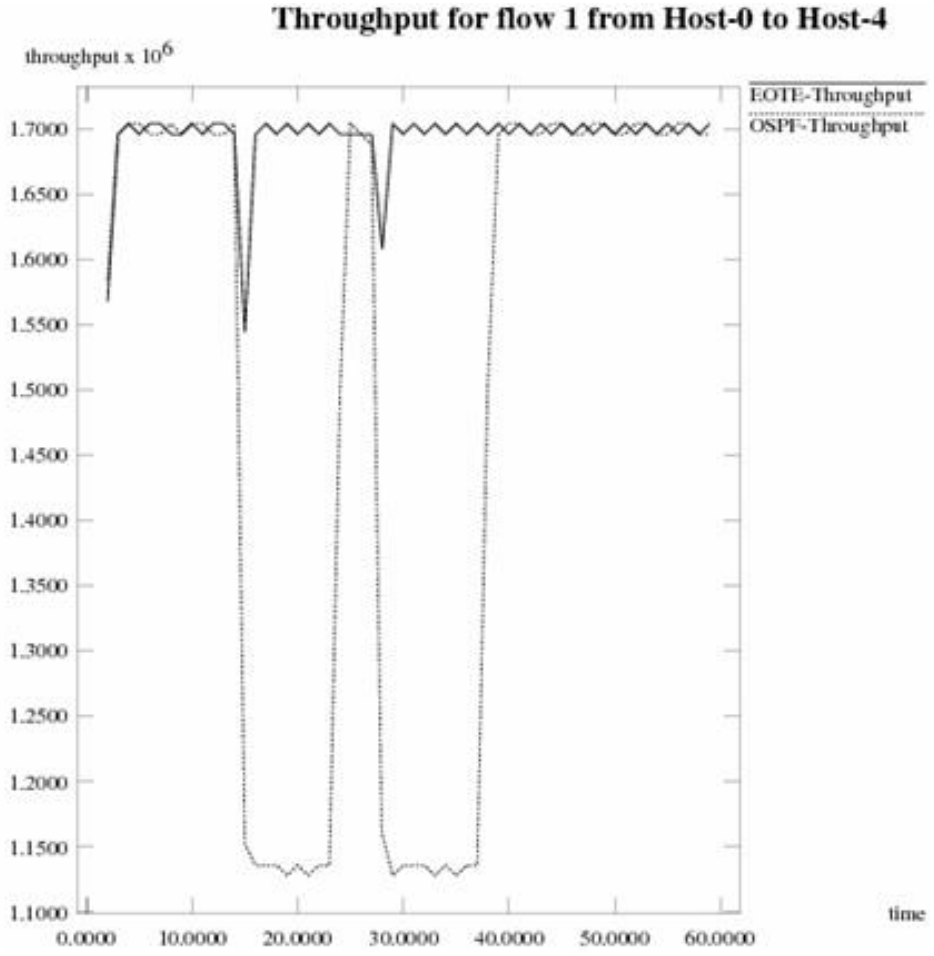

Figure 4.15. Throughput - Host 0 to Host 4.

In our approach we can see very less loss because of flapping since the data rate is increased slowly when a link comes up as shown in throughput and loss ratio graphs below.

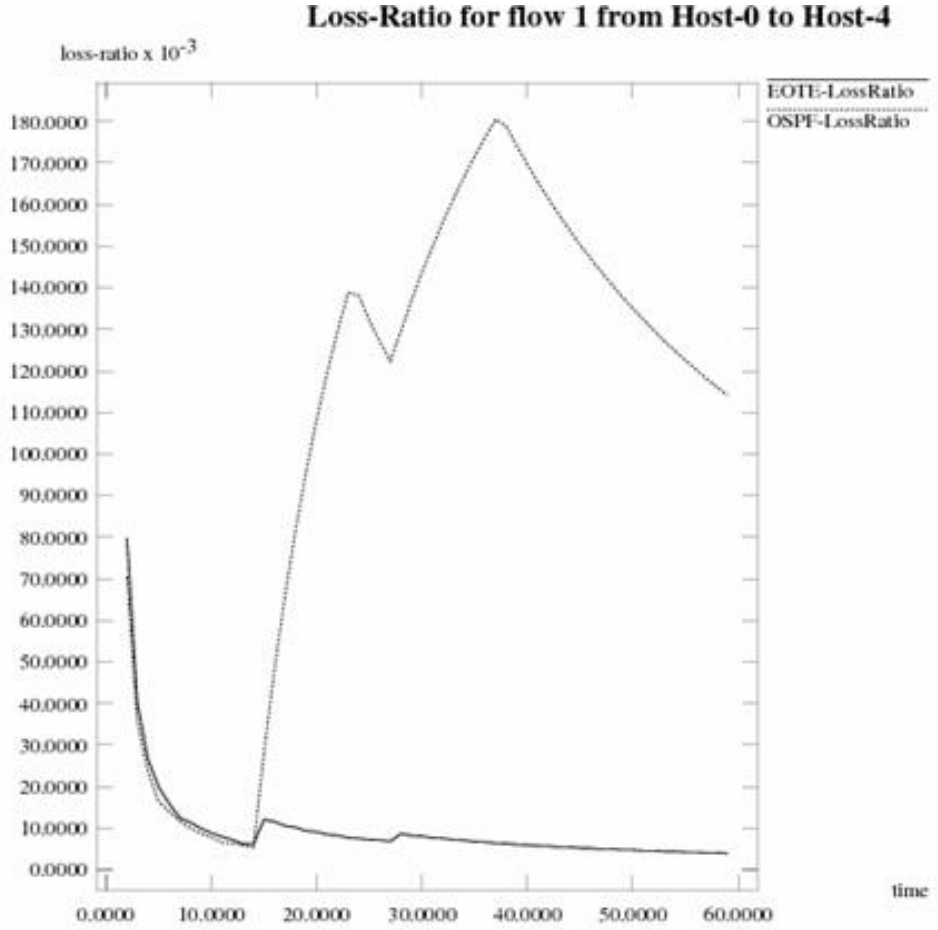

Figure 4.16. Loss Ratio - Host 0 to Host 4.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, we have proposed a novel approach for End-to-End Optimal Algorithms in presence of multiple shortest paths. We have implemented a new adaptive control algorithm, which enables scalable traffic engineering, faster failure detection and recovery in connection-less oriented network where disjoint multiple paths are available between any ingress-egress pair. The algorithm engineers the traffic and recovers from the failure simultaneously, using only source-inferred information. It can also overcome the problems of route flaps and delayed network convergence of the OSPF routing protocol. Our algorithm detects node/link failure within a few milliseconds and hence achieves network convergence much faster compared the OSPF routing protocol. This is the first work that applies a binary-feedback-based control laws to study how it affects throughput performance in a connectionless network.

A salient feature of this family of control laws is that the only input to each control law is, whether a forwarding path in a multipath is congested or not. We have implemented a source inferred congestion detection mechanism, without explicit feedback from the network core. This is done by sending echo packets from ingress node to egress node through all the available paths and by measuring the round trip timeout. Our approach makes it possible to design a highly scalable network since it involves only the edge routers.

Our scheme achieves higher resilience compared to OSPF routing protocol by re-routing the traffic through other available shortest paths within few milliseconds upon

node/link failure. When a node or link fails, the control laws deviate the traffic away from inoperative link/node. Simulation results show that the proposed algorithm can swiftly and effectively minimize the congestion and distribute traffic load efficiently under dynamic network conditions. We have shown how our approach helps underlying routing protocols perform better, in terms of throughput, delay loss ratio and other network parameters. We have also demonstrated that our approach can effectively reroute the traffic to an optimal state in the presence of sudden link failures.

5.2 Future Work

In this thesis we have studied only best effort class of service. The approach needs to be studied for different CoS. Our approach makes use of only the equal cost shortest paths available. It can be extended to use K cost shortest paths.

We have studied the performance in case single link failure. We would like to explore in more detail the interactions between multiple node failures and how they affect stability. We have used smooth round trip timeout mechanism to detect congestion. A better scheme to detect source inferred congestion will be an interesting work for future.

Finally, simulations could be extended to multiple OSPF areas to explore interactions with an Exterior Gateway Protocol such as BGP.

REFERENCES

[1] C. Lagoa B. Movsichoff and Hao Che. End-to-end optimal algorithms for integrated qos, traffic engineering, and failure recovery. In *IEEE/ACM Transactions on Networking*, 2007.

[2] Hao Che C. Lagoa and B. Movsichoff. Adaptive control algorithms for decentralized optimal traffic engineering. In *IEEE/ACM Transactions on Networking*, volume 12, June 2004.

[3] S. J. Golestani and S. Bhattacharyya. A class of end-to-end congestion control algorithms for the internet. In *Int. Conf. Network Protocols, ICNP, Austin*, volume 137-150, Oct 1998.

[4] A. K. Maulloo F. P. Kelly and D. K. H. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. In *J. Oper. Res. Soc*, volume 49, Mar 1998.

[5] R. J. La and V. Anantharam. Charge-sensitive tcp and rate control in the internet. In *Proc. IEEE INFOCOM 2000*, volume 3, Mar 2000.

[6] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. In *IEEE/ACM Trans. Networking*, volume 7, Dec 1999.

[7] D. Mitra F. Bonomi and J. B. Seery. Adaptive algorithms for feedback-based flow control in high-speed, wide-area networks. In *IEEE J. Select. Areas Commun*, volume 7, Sep 1995.

[8] S. Sarkar K. Kar and L. Tassiulas. A simple rate control algorithm for max total user utility. In *Proc. IEEE INFOCOM 2001*, volume 1, Apr 2001.

[9] Shaikh A and A. Greenberg. Experience in blackbox ospf measurement. In *ACM SIGCOMM Internet Measurement Workshop (IMW)*, volume 1, Nov 2001.

[10] J. Moy. Ospf version 2. In *Network Working Group*, Apr 1998.

[11] B Cain. Fast link state flooding. In *Global Telecommunications Conference, 2000*, volume 1, Dec 2000.

[12] A. Basu and J. Riecke. Stability issues in ospf routing. In *Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM SIGCOMM*, volume 1, Aug 2001.

[13] M.; Osamu N.; Murai J Ohara, Y.; Bhatia. Route flapping effects on ospf. In *Applications and the Internet Workshops, 2003*, volume 1, Jan 2003.

[14] A.S.; Sapozhnikova V.D Choudhury, G.L.; Maunder. Faster link-state igp convergence and improved network scalability and stability. In *Local Computer Networks, 2001*, Nov 2001.

[15] J.; Thorup M Fortz, B.; Rexford. Traffic engineering with traditional ip routing protocols. In *Communications Magazine, IEEE*, volume 40, Oct 2002.

[16] J.J Vutukury, S.; Garcia-Luna-Aceves. A traffic engineering approach based on minimum-delay routing computer communications and network. In *Proceedings. Ninth International Conference*, Oct 2000.

[17] S. Floyd and T. Henderson. The new reno modification to tcp's fast recovery algorithm. In *IETF RFC 2582*, Apr 1999.

[18] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. In *IEEE/ACM Trans. Networking*, volume 1, Aug 1993.

[19] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. In *J. Comput. Networks and ISDN Syst*, volume 17, Jun 1989.

[20] R. La M. Shayman T. Guven, C. Kommareddy and B. Bhattacharjee. Measurement based optimal multi-path routing. In *Proceedings of INFOCOM'04*, volume 1, Mar 2004.

[21] S. Low A. Elwalid, C. Jin and I. Widjaja. Mate: Mpls adaptive traffic engineering. In *Proceedings of IEEE INFOCOM'01*, volume 3, Apr 2001.

[22] D. Bersekas and R. Gallager. Data networks. In *Prentice-Hall*, 1995.

[23] C. Villamizar. Ospf optimized multipath. 1999.

[24] R.P.; Nelakuditi, S.; Zhi-Li Zhang; Tsang. Adaptive proportional routing: a localized qos routing approach. In *INFOCOM 2000. IEEE*, volume 3, Mar 2000.

[25] C. M. Lagoa B. A. Movsichoff and H. Che. Decentralized optimal traffic engineering in connectionless networks. In *IEEE J. Select. Areas Commun.*, volume 23, Feb 2005.

[26] John Evans Olivier Bonaventure Pierre Francois, Clarence Filsfils. Achieving subsecond igp convergence in large ip networks. In *ACM SIGCOM Computer Communication Review*, volume 35, Jul 2005.

[27] K.K.; Wu-chi Feng; Goyal, M.; Ramakrishnan. Achieving faster failure detection in ospf networks. In *IEEE International Conference*, volume 1, May 2003.

BIOGRAPHICAL STATEMENT

Sukruth Srikantha was born in Mysore, India in 1981. He received B.E. in Computer Science and Engineering from Bangalore Institute of Technology, Bangalore, India.

He began his study toward the M.S. degree in the department of Computer Science and Engineering at The University of Texas at Arlington in August 2004. His research interests focus on Traffic Engineering and Routing Protocols.