IMPROVING EFFICIENCY AND EFFECTIVENESS OF MULTIPATH ROUTING IN COMPUTER NETWORKS

A Dissertation

by

YONG OH LEE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Engineering

IMPROVING EFFICIENCY AND EFFECTIVENESS OF MULTIPATH ROUTING IN COMPUTER NETWORKS

A Dissertation

by

YONG OH LEE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	A. L. Narasimha Reddy
Committee Members,	Srinivas Shakkottai
	Jean-Francois Chamberland
	Radu Stoleru
Head of Department,	Costas Georghiades

May 2012

Major Subject: Computer Engineering

ABSTRACT

Improving Efficiency and Effectiveness of Multipath Routing
in Computer Networks. (May 2012)
Yong Oh Lee, B.S., Yonsei University;
M.S., Yonsei University
Chair of Advisory Committee: Dr. A. L. Narasimha Reddy

In this dissertation, we studied methods for improving efficiency and effectiveness of multipath routing in computer networks. We showed that multipath routing can improve network performance for failure recovery, load balancing, Quality of Service (QoS), and energy consumption. We presented a method for reducing the overhead of computing dynamic path metrics, one of the obstacles for implementing dynamic multipath routing in real world networks.

In the first part, we proposed a method for building disjoint multipaths that could be used for local failure recovery as well as for multipath routing. Proactive failure recovery schemes have been recently proposed for continuous service of delay-sensitive applications during failure transients at the cost of extra infrastructural support in the form of routing table entries, extra addresses, etc. These extra infrastructure supports could be exploited to build alternative disjoint paths in those frameworks, while keeping the lengths of the alternative paths close to those of the primary paths. The evaluations showed that it was possible to extend the proactive failure recovery schemes to provide support for nearly-disjoint paths which could be employed in multipath routing for load balancing and QoS.

In the second part, we proposed a method for reducing overhead of measuring dynamic link state information for multipath routing, specifically path delays used in Wardrop routing. Even when dynamic routing could be shown to offer conver-

gence properties without oscillations, it has not been widely adopted. One of reasons was that the expected cost of keeping the link metrics updated at various nodes in the network. We proposed threshold-based updates to propagate the link state only when the currently measured link state differs from the last updated state considerably. Threshold-based updates were shown through analysis and simulations to offer bounded guarantees on path quality while significantly reducing the cost of propagating the dynamic link metric information. The simulation studies indicated that threshold based updates can reduce the number of link updates by up to 90-95% in some cases.

In the third part, we proposed methods of using multipath routing for reducing energy consumption in computer networks. Two different approaches have been advocated earlier, from traffic engineering and topology control to hardware-based approaches. We proposed solutions at two different time scales. On a finer time granularity, we employed a method of forwarding through alternate paths to enable longer sleep schedules of links. The proposed schemes achieved more energy saving by increasing the usage of active links and the down time of sleeping links as well as avoiding too frequent link state changes. To the best of our knowledge, this was the first technique combining a routing scheme with hardware scheme to save energy consumption in networks. In our evaluation, alternative forwarding reduced energy consumption by 10% on top of a hardware-based sleeping scheme. On a longer time granularity, we proposed a technique that combined multipath routing with topology control. The proposed scheme achieved increased energy savings by maximizing the link utilization on a reduced topology where the number of active nodes and links are minimized. The proposed technique reduced energy consumption by an additional 17% over previous schemes with single/shortest path routing.

To my father in heaven

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. A. L. Narasimha Reddy, for his perceptive guidance and consistent support to whole graduate studies. I also appreciate to my committee members: Dr. Shakkottai, Dr. Stoleru, and Dr. Chamberland, for their valuable suggestions and comments on this research.

My thanks also go to mentors in AT&T Labs Research: Vinay Vaishampayan and Rittwik Jana, as well as collegues in the Computer Science department: Myounggyu Won and Wei Zhou, even though our works were not included in this dissertation.

I give to heartfelt thanks to my wife, mother, and family-in-law. Without their support, trust, and love, I could not complete this work.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

The Internet takes an increasingly central role in our communication infrastructure. Traditional application data were delivered in a manner of best efforts. However, the demands of delay-sensitive applications, such as voice over IP (VoIP), video streaming, and gaming, have been increasing. These applications require more continuous availability compared to data applications. Availability is not only related to failure recovery, but also to QoS such as end-to-end delay or available bandwidth.

Multipath routing is one of the promising schemes to improve availability. Most currently deployed routing protocols select only a single path for the traffic between a source-destination pair. However, single path routing takes additional time to compute a new path after a failure and when congested, and may not provide sufficient bandwidth to the application even when alternate paths exist between the source and the destination. Multipath routing can overcome these problems by forwarding to alternate paths and flexibly splitting traffic among multiple paths when the primary path does not meet availability. The well-known benefits of multipath routing include flexibility in meeting application performance requirements, improving end-to-end reliability, and avoiding congested paths [1].

Traditionally networks are built to handle peak traffic demands and varying traffic loads, the network may have excess capacity beyond the current requirements. The excess capacity results in wasted energy and there is a growing interest in reducing the energy consumption in networks. By providing increased number of options for routing traffic, multipath is expected to enable increased energy savings in networks.

The journal model is *IEEE Transactions on Automatic Control*.

This dissertation aims to improve efficiency and effectiveness of multipath routing. The first contribution of this dissertation is constructing disjoint paths for failure recovery and multipath routing. The infrastructure necessary for proactive failure recovery schemes is exploited to provide disjoint paths for multipath routing during normal time. The second contribution of this dissertation is in reducing overhead of link state updates for dynamic multipath routing, specifically Wardrop routing, in networks. One of the reasons why dynamic multipath routing was not deployed in today's Internet was overhead of link state updates. A method of reducing these overheads is proposed. The last contribution of this dissertation is reducing energy consumption using multipath routing. Multipath routing reduces network energy consumption compared to single path routing. The following of this chapter introduces these three parts of this dissertation.

A. Constructing disjoint paths for multipath routing

Applications, such as Voice over IP, video streaming, and gaming, require more continuous availability compared to the traditional data applications. Link/node failures are common in IP networks today [2]. Traditional routing schemes compute recovery paths after detecting a failure. Routing convergence can take several tens of seconds after a failure. During this transient time, from the time of a failure to the time when all the nodes have new routing tables computed, applications can observe severe disruptions in service. This disruption of service during failure situations can be a serious problem for continuous media applications. Several proactive recovery schemes have been recently proposed to reduce failure transient time [3–5]. In these schemes, backup paths are pre-computed before a failure. The failure-discovering router employs the backup next-hop after a failure, until the new routing tables are

computed taking the failure into account. As a result, the fast recovery mechanisms provide an almost instantaneous response to a failure. Proactive recovery schemes strive to provide continuous service even during the failure transients.

Proactive recovery schemes require additional infrastructure to provide fast recovery from failures. This additional support includes extra routing table entries, extra fields or bits in the packet headers to indicate which links or nodes are failed, or extra addresses depending on the employed scheme. Also, proactive recovery schemes may not employ some of the links of the primary path (before the failure) in the recovery/backup path (during the failure transient). This can result in increased backup path lengths. Increased backup path lengths can increase the load on the network which can result in unbalanced load and increased delay. The length of the recovery path when compared to the length of the primary path, is a measure of success, for these schemes. Ideally, the length of the recovery path is not much longer than the length of the primary path.

In this work, techniques for reducing the backup path lengths without increasing the overhead in network infrastructure are proposed. Furthermore, we study whether the recovery paths can be made disjoint, when possible, from the primary path. We explore if the primary path and the recovery path can be made disjoint, such that the additional infrastructure put in place for failure recovery, could be used potentially for multi-path routing during normal times when no failures occur. Thus, the same infrastructure can be utilized for two purposes: not only failure recovery when failures present, but also multi-path routing when no failures are present in the network.

The problem of building recovery paths disjoint from the primary paths while keeping the length of recovery paths close to the length of primary paths is considered here. Also, the cost of the proactive recovery schemes is analyzed. The construction of primary paths is not constrained and hence any routing algorithm can be employed

to construct the primary paths. The length of secondary paths is important for both failure recovery and multipath routing.

Two proactive recovery schemes are considered here: Multiple routing configurations (MRC) [3] and NotVia [4]. We study how those schemes for fast recovery can be enhanced to build disjoint recovery paths in those frameworks. To this end, techniques for disjoint multi-path computation are developed: disjoint multiple routing configuration (D-MRC) and disjoint NotVia (D-NotVia).

In this work, a secondary path is built, that is disjoint or maximally disjoint from the primary path, which can be used for failure recovery, load balancing or QoS routing. The focus is on computing efficient secondary paths and not on the schemes for utilization of secondary paths.

In this work, the contributions are following: algorithms for exploiting the MRC and NotVia frameworks for the construction of disjoint paths are proposed; it is shown through evaluations that MRC and NotVia can be enhanced to provide nearly disjoint paths with small increment of path length; and it is shown that the computed disjoint paths can be used for multi-path routing for load balancing and QoS.

B. Reducing overhead of dynamic multipath routing

Current routing algorithms utilize static link costs to compute routing tables between different nodes in the network. The link costs are static for long periods of time (over the duration of several hours) and are determined by the traffic engineering constraints of the network. The problem of determining the link costs has received significant attention [6–8]. Typically, the traffic matrix and several considerations, such as keeping maximum link utilization low, are factored into obtaining link costs. The problem of determining link costs may be simultaneously coupled with the problem of computing

routing paths in some approaches [9–11]. The current approaches to determine link costs take traffic matrices over several hours into account such that varying traffic matrices may be reasonably accommodated with one set of link costs [12, 13].

Dynamic link metrics, such as link delay, queueing lengths and available link bandwidth, have been considered earlier as potential link cost metrics for routing purposes. For example, routing high bandwidth video flows might benefit from an idea of available link bandwidth in QoS routing [14]. Similarly, path lengths or delays can be useful in Wardrop routing [15]. Early ARPAnet considered link delays as a cost metric and the resulting oscillations prompted the use of other metrics based on capacity. QoS routing has explored the use of different dynamic metrics in routing traffic, for example in [16–19]. Dynamic metrics such as available bandwidth and path delay have been proposed for use in routing video and audio traffic in the network. This body of work considered the tradeoff in keeping the link state information disseminated and the quality of paths that can be computed. Some of this work proposed techniques for finding new paths efficiently, for example [19].

Most current networks, however, do not employ dynamic link metrics for various reasons. Since these metrics are dynamic, as the link metrics change over time, the traffic might be routed at different times through different paths in the network, potentially causing oscillations with incorrect choice of link metrics or routing algorithms. Even when the routing algorithms are carefully designed to not cause oscillations, the cost of propagating the link metrics has been one of the obstacles to the adoption of these algorithms.

As the dynamic link metrics change over time, these metrics need to be measured and propagated to other nodes in the network in order to keep the routing paths from deviating far from ideal. The more frequently the link metric information is propagated, the more accurate the information that the nodes have about the state of

the network, and the better the efficiency of the computed network paths. However, higher frequency of updates leads to higher cost in propagating the link metric information. This tension or tradeoff has been studied through simulations, for example, in QoS routing [20].

Recently, dynamic routing algorithms, have received renewed interest for balancing load in wireless networks [15], for dynamic traffic management in wired networks [9] and traffic management across multiple paths in a multi-homed network [21]. These algorithms have used link delays or utilizations for dynamic routing.

Link updates can be sent periodically or triggered on link up/down events in OSPF routing. In order to prevent spurious link up/down events from generating excessive link update traffic, timers may be employed. These timers are in the range of several seconds (typically 30s). The importance of conveying reliable link information without generating excessive number of link updates has been earlier recognized [22, 23].

A related question that arises with the quality of link information is whether the routing algorithm can converge to a stable state despite the delay or inaccuracies in the link state information that is used in making the routing decisions.

In this work, these problems of reducing the cost of propagating dynamic link metric information across the network while ensuring stability of the routing algorithm are addressed here. Wardrop routing that employs link delays as a link cost metric is considered as an example dynamic routing approach. In Wardrop routing, the traffic is split across available paths in such a way as to equalize the delay across all the available paths at a node. The traffic splitting can be done at the end hosts [24] or further split at the routers in the network as traffic moves from one hop to the next [9]. However, the results can be equally applied to other algorithms, with suitable modifications.

This work makes the following significant contributions: we proposes a simple technique, called threshold-based propagation, for propagating link metric information; we presents an analysis that threshold propagation can guarantee that the observed path quality will be within an error bound of the optimal path quality if the exact information is available; and we show, through simulations, that threshold propagation reduces the cost of propagating link cost information significantly, in some cases by up to 90-95%.

C. Reducing energy consumption by multipath routing

Reducing energy consumption in wireless networks has received significant attention. In wireless networks, energy saving is important, because wireless network devices have limited life time that depends on battery energy. Energy saving in the wired networks has been traditionally overlooked because power supplies to wired network devices are unlimited. However, network researchers have started studying energy issues even in wired network due to environmental and cost considerations. The potential energy saving of the US network infrastructure could be 0.5-24 billion dollars per year [25].

The opportunity to save energy consumption comes from several factors. Network capacity is normally provisioned for peak traffic loads. High traffic load demands due to special events and the need for tolerating network failures may factor into estimating peak traffic loads. As a result, the average link utilization could be less than 30-40% and the duration of peak traffic load a small fraction of the entire day [26]. Reducing the link capacity during off-peak duration is a promising scheme to save energy consumption. To deal with the variations of traffic load at different times of the day, topology control and traffic engineering can be used to shutdown some links

and nodes of the network while leaving the network connected and with sufficient capacity to carry the traffic load. Energy consumption of network equipment remains substantial even when the network is idle. Forcing the link to sleep mode during idle duration is another approach to save energy consumption.

Traffic engineering approaches power off the network elements while the powered-up network capacity meets the traffic demand. Powering off the links is first attempted to reduce the energy consumption [27]. However, several studies report that the power consumption of a node is much higher than the power consumption of a link. In [28], authors attempt to power off nodes first with single shortest path routing. However, the number of active node is not greatly reduced.

We study how to reduce the number of active nodes for energy savings. A new topology control and multipath routing is proposed. The proposed topology control takes an approach of building an appropriately provisioned network to meet the demands of all the terminal nodes. This is in contrast with existing approaches where nodes and links are removed from the given network. Also, we study the effectiveness of employing multi-path routing to reduce energy consumption. While multi-path routing allows more possibilities for routing the traffic demands, if longer alternate paths are employed, the power consumption can actually increase. An effective multi-path routing strategy for reducing energy consumption is proposed. The proposed scheme is evaluated by various simulations. In most cases, even if single shortest path is used, our topology control achieves more energy saving with a smaller number of active nodes than previous schemes. The proposed topology control reduces the number of iterations for finding a suitable topology. In addition, multipath routing with the proposed topology control reduces the number of active nodes further, and achieves more energy savings.

Another avenue for energy saving in networks is through hardware mechanisms

operating at smaller time scales. Some of the network hardware can operate in different modes with multiple power consumption levels. When such hardware modes are available, the links can enter sleeping mode that consumes lower power to save energy. The performance of hardware schemes depends on its power level decision based on the estimation of the inter packet arrival time. However, network traffic is hard to estimate due to its dynamics and hardware is forced to wake up and operate in a higher power consuming mode when a packet arrives at the switch or the link.

We propose a scheme that combines routing with hardware sleeping modes to reduce energy consumption. The proposed scheme employs alternate path forwarding of packets to enable links in sleeping mode to staying that mode longer. It is expected that longer sleep cycles will lead to higher energy savings. Alternate forwarding has to be carefully employed as alternate paths can be longer than primary paths and hence may result in higher power consumption. However, alternate forwarding can reduce the latency of forwarding a packet as the delays in waking up sleeping links and switches can be avoided through the alternate path. The proposed alternative forwarding combining hardware sleeping scheme reduces energy consumption without increasing the packet forwarding latency.

CHAPTER II

CONSTRUCTING DISJOINT PATHS FOR FAILURE RECOVERY AND MULTIPATH ROUTING

In this chapter, techniques for building recovery paths disjoint from the primary paths while keeping the length of recovery paths close to the length of primary paths are proposed.

Two proactive recovery schemes are discussed here: Multiple routing configurations (MRC) [3] and NotVia [4]. If those schemes for fast recovery can be enhanced to build backup paths that are disjoint or maximally disjoint from the primary path, the backup paths can be used for failure recovery, load balancing, or QoS routing. We focus on computing efficient secondary paths and not on the schemes for utilization of secondary paths.

A. Proactive recovery schemes

We consider a network represented by a graph $G = (V, E)$. $s \in V$ is the source node, and $d \in V$ is the destination node. $i \in V$ is the current node where a routing decision needs to be made. We denote $P(s, d)$ as a set of links on the path from s to d. Traditional shortest path routing in IP networks computes the routing cost from s to d, $C(s, d)$, and the next-hop node for the route from s to d, $NH(s, d)$.

If there is no failure in G, a packet is forwarded to the next-hop node $NH(i, d)$ at each node i. We denote the primary next-hop node on G as $NH_p(i, d)$. When there is a failed link or node in G, proactive recovery scheme is used to detour the failure and to recover from the failure. In proactive recovery schems, the node i detecting the failure reroutes the packet to a different next-hop node, referred to as the backup next-hop node, $NH_b(i, d)$, in order to recover from the failure. The backup next-hop

nodes at different nodes in the network must be chosen in a consistent manner to avoid routing loops.

MRC [3] employs multiple configurations. A configuration is a network topology with associated link weights. The different configurations employed by MRC employ the same network topology, but with different link weights. In addition to normal routing configuration with no failures where all link weights are the same as the link weight on original topology, the additional backup configurations, G_k, $k = 1, ..., N$, are designed to cover the failure of some nodes and links. In each backup configuration, a number of nodes are *isolated* to model their failure and hence not employed in routing. The links connected to an isolated node should be either isolated or restricted in backup configurations. A number of links are *isolated*, i.e., link weights set to infinite, to model their failure and hence not employed in routing. A link may be *restricted*, with its weight set to a very large finite weight, such that this link is used only to reach the node attached to that link. The weight on a restricted link in a backup configuration prevents forwarding to the isolated as an intermediate node but allows losing connectivity in the backup configuration.

S_k is the set of the isolated nodes in G_k, and L_k is the set of the isolated links: $link(i, N^i)$, where $i \in S_k$ and N^i is the neighbor of i. Each link is isolated in at least one of backup configurations, G_k, $k = 1, ..., N$. It means $\bigcup_{k=1}^{N} S_k = V$ and $\bigcup_{k=1}^{N} L_k = E$. Every node maintains one routing table entry corresponding to each backup configuration for every destination. If $NH_p(i, d)$ or $link(i, NH_p(i, d))$ fails, a packet is routed over G_k where $NH_p(i, d) \in S_k$ or $link(i, NH_p(i, d)) \in L_k$. The indicator($k$) of backup topology($G_k$) over which the packet is forwarded is carried in the header of every packet.

We can show that the cost of recovery path by backup configuration is not less

than the cost of the primary path.

Theorem 1. *$C(s,d) \leq C_k(s,d)$ where $C(s,d)$ is the routing cost on G, and $C_k(s,d)$ is the routing cost on $G_k (k = 1, 2, ...N)$.*

Proof. If $\forall link(i, NH_p(i,d)) \notin L_k$ where $link(i, NH_p(i,d)) \in P(s,d)$, then $C(i,d) = C_k(i,d)$.
If $\forall link(i, NH_p(i,d)) \in L_k$ where $link(i, NH_p(i,d)) \in P(s,d)$, then $C(i,d) \leq C_k(i,d)$.
Suppose $link(i, NH_p(i,d)) \in L_k$ where $link(i, NH_p(i,d)) \in P(s,d)$.
If $C(i, NH_p(i,d)) + C(NH_p(i,d), d) < C(i, NH_b(i,d)) + C(NH_b(i,d), d)$,
then $C(i,d) < C_k(i,d)$. Then, $C(s,d) < C_k(s,d)$.
If $C(i, NH_p(i,d)) + C(NH_p(i,d), d) = C(i, NH_b(i,d)) + C(NH_b(i,d), d)$,
then $C(i,d) = C_k(i,d)$. Then, $C(s,d) = C_k(s,d)$. □

Each configuration results in extra infrastructure support at each node (proportional to the number of configurations, N) and a larger number of configurations also need a larger number of bits in the packet header ($log_2 N + 1$). In order to minimize the number of configurations(N), greedy algorithms are employed where as many nodes and links as possible are removed in a single backup topology. The focus on decreasing the number of configurations can result in longer backup paths.

Lemma 2. *If $S_k \subset S_l$, $C_k(s,d) \leq C_l(s,d)$.*

Proof. The proof follows from the fact the shortest paths available in the network with S_k failures is a superset of the paths available with S_l. □

In NotVia [4], routers are provided additional IP addresses. These additional addresses are used during a failure to route around the failed link or node. NotVia has two kinds of NotVia addresses: one is for the recovery of link failure (we denote it as $NV_{link}(i,j)$ which is used for recovery of the link (i,j)) and another is for the recovery

of a node failure (we denote it as $NV_{node}(i,d)$ which is used for recovery of node i and whose destination is d). When $NV_{link}(i, NH_p(i,d))$ is used, NotVia finds the shortest path destined to $NH_p(i,d)$ with the removal of the failed link(i,$NH_p(i,d)$). $NV_{node}(NH_p(i,d),d)$ allows finding the shortest path from i to $NH_p(NH_p(i,d),d)$ without $NH_p(i,d)$ (all the links connected to $NH_p(i,d)$ are removed). NotVia uses tunneling. The node detecting the failure encapsulates the packet with a NotVia address as a destination to route around the failure.

Each NotVia address is designated for tolerating an individual failure and the routing tables are computed accordingly to route packets destined to these NotVia addresses, ahead of time. However, NotVia increases the path length due to the increased hop count between the failure detecting node and the next-hop of the failure detecting node on the primary path.

Theorem 3. *$C(s,d) \leq C^{NV}(s,d)$ where $C(s,d)$ is the routing cost on G, and $C^{NV}(s,d)$ is the routing cost of the routing with the NotVia address.*

Proof. The path from s to d with $NV_{link}(s, NH_p(s,d))$ removes $link(s, NH_p(s,d))$. Removing the links on the primary path makes,
$C(s, NH_p(s,d)) \leq C(s, NV_{link}(s, NH_p(s,d)))$. As a result, $C(s,d) \leq C^{NV}(s,d)$.
The path from s to d with $NV_{node}(NH_p(s,d),d)$ removes $link(s, NH_p(s,d))$ and $link(NH_p(s,d), NH_p(NH_p(s,d),d))$. Removing the links on the primary path makes, $C(s, NH_p(NH_p(s,d))) \leq C(s, NV_{node}(NH_p(s,d),d))$. As a result, $C(s,d) \leq C^{NV}(s,d)$.

□

Moreover, the backup path is not disjoint from the primary path because the packet is expected to continue on the primary path after reaching the NotVia address.

Both MRC and NotVia precompute backup routing tables at each node based on available topology information which can be obtained through link-state routing

algorithms.

B. Building disjoint paths using proactive recovery schemes

The proposed schemes do not constrain the construction of primary paths, unlike other approaches that try to construct a pair of shortest disjoint paths simultaneously [29]. Disjoint path routing with the augmented cycles also could not use the shortest primary path [30]. Disjoint paths are computed and built using the same infrastructure that is put in place for failure recovery.

The simplest method to construct a secondary path is to find the shortest path after removing the primary path. However, the simplest method, while useful for source routing, can be very expensive in terms of the required infrastructure support for hop-by-hop routing.

In this chapter, *disjointness* is defined as follows (similar to the novelty measure in [31]). Let $P_{primary}(s,d) = \{(s,p_1),(p_1,p_2),...,(p_n,d)\}$ be denoted as a set of links on the primary path constructed by the routing scheme. Let $P_{backup}(s,d) = \{(s,b_1),(b_1,b_2),...,(b_n,d)\}$ be denoted as a set of links on the backup path. The disjointness of the backup path with respect to the primary path is measured as

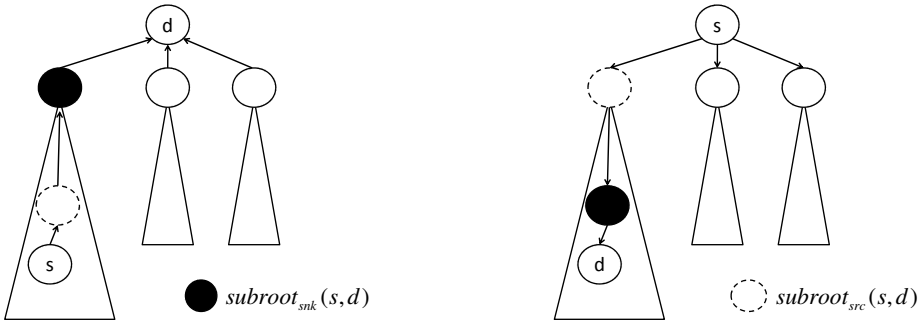
$$disjointness = 1 - \frac{|P_{primary}(s,d) \cap P_{backup}(s,d)|}{|P_{primary}(s,d)|} \tag{2.1}$$

We also define *stretch* of backup as the ratio of the path length of $P_{primary}$ to the path length of P_{backup}.

$$stretch = \frac{|P_{backup}(s,d)|}{|P_{primary}(s,d)|} \tag{2.2}$$

The proposed schemes try to construct backup/recovery paths whose disjointness is as close to 1 as possible, while keeping path stretch as small as possible. The success of the failure recovery schemes based on these two measures of disjointness and stretch

is meausred. The necessary infrastructure support of the different schemes is also compared.



(a) Subtree of sink routing tree (b) Subtree of source routing tree

Fig. 1. Computing subroots in the sink and the source routing trees

In a routing tree, the children of the root are called subroots. The trees rooted at the subroots are called subtrees. $subroot_{snk}(s,d)$ is the subroot of s in the sink routing tree destined to d (Fig. 1(a)). $subroot_{src}(s,d)$ is the subroot of d in the source routing tree rooted at s (Fig. 1(b)). It is assumed that sink and source routing tree are symmetric.

For a disjoint path, the routing protocol should forward the packet to a different subtree in the sink routing tree, as shown in figure 2. ($NH_b(i,d) = j$, and $j \notin$ subtree rooted from $subroot_{snk}$ (s,d)). Once the packet reaches a different subtree, the packet can be forwarded along its primary path (from j to d in Fig. 2). However, there are sometimes no neighbor nodes in the other subtrees. In such a case, the routing protocol should forward to a node in the same subtree, but one that is not used in the primary path ($NH_b(s,d) = i$ and $i \in$ subtree rooted from $subroot_{src}$ (s,d)). The constructed secondary path can be used both during a failure (that it is designed to tolerate) and as a disjoint path during normal operation with no failures.

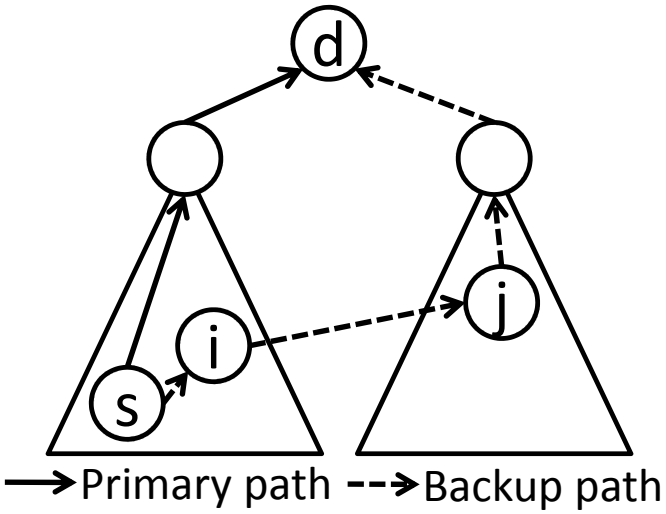


Fig. 2. Disjoint path forwarding from source s to destination d

In OSPF and IS-IS routing algorithms, the source routing trees are computed for routing tables. The $subroot_{src}(i, d)$ and $subroot_{snk}(i, d)$ are computed from the existing source routing trees. Each node computes $subroot_{src}(N^i, d)$ and $subroot_{snk}(N^i, d)$ from the source routing trees of its neighbor nodes (N^i) or the neighbor nodes can communicate their $subroot_{src}(N^i, D)$ and $subroot_{snk}(N^i, D)$ with each other.

1. Disjoint Multiple Routing Configuration (D-MRC)

Links and nodes can be *isolated* or *restricted* in backup configuration in MRC. Packets cannot be routed through an isolated node to another node in a backup configuration. The links connected with the isolated node are either isolated or restricted. The isolated link never delivers packets. For this purpose, the link weight of the isolated link is set to infinite. To prevent the last hop problem [3], the restricted link could deliver only the packets headed to the isolated node. The weight of the restricted link is set as a very high value (e.g., at least the sum of the weights of all the links in [3]).

D-MRC is developed based on MRC [3] to enhance disjointness and to reduce stretch. In order to compute disjoint or maximally disjoint backup paths whose stretch is close to 1 in the MRC framework, the following ideas are employed.

The set of isolated and restricted nodes/links are chosen carefully in each backup configuration. The maximum number of isolated nodes ($Max.Iso$) is restricted in a single backup configuration. This is expected to potentially provide shorter backup paths while keeping the number of backup configurations from getting too large. In contrast to this idea, MRC [3] is a greedy algorithm to minimize the number of backup configurations. As a result, the early computed backup configurations have a tendency to have more isolated nodes than the later computed backup configurations. A large number of isolated nodes could result in large path lengths in a single backup configuration. The maximum number of isolated nodes tries to distribute the number of isolated nodes evenly through all the backup configurations such that the backup paths are smaller in length.

The neighbor nodes of the isolated node play a key role in the construction of a disjoint path and keeping the path lengths short in backup configurations. Since an isolated node can only receive (or send sometimes) packets via the restricted link, the neighbor node of the isolated node, connected with this restricted link, carries all of the traffic of the isolated node in the backup configuration. Hence, it is important to choose this node carefully (termed restricted node here).

$$
\begin{aligned}
RD(i,j) &= \sum_{d=\{v \in V-\{i\}\}} rd_i^j(d) \\
rd_i^j(d) &= \left\{ \begin{array}{ll} 1 & NH_p(i,d) = j \\ 0 & otherwise \end{array} \right\}.
\end{aligned}
\tag{2.3}
$$

Routing density($RD(i,j)$) is defined as the number of times a neighbor node (node $j \in N^i$) is selected as the next-hop of node i to all destinations in normal routing. Routing density is computed using the entries of the routing table for the primary path.

The node which has the lowest routing density is selected as the restricted node. It is expected that since this node is used the least number of times in the primary paths, by making it the only option for routing to the isolated node in a backup configuration, the set of paths used in the backup configuration will be very likely different from the set of paths used in the primary configuration.

In order to facilitate this idea, a weight proportional to the routing density (as shown in (4)) is added to the link weights in backup topologies. This particular weight function retains the restrictions on routing to the isolated nodes.

$$w(i,j) = w(i,j) + \frac{(RD(i,j))}{\max_{k \in N^i} RD(i,k)} W \tag{2.4}$$

$w(i,j)$ is the link weight of $link(i,j)$ and $W = \sum_{(i,j) \in E} w(i,j)$.

The construction of backup topologies is given in algorithm 1 as a pseudo code. In algorithm 1, $div(i)$ is the function to check if isolating node i leaves the graph disconnected, and $N(s)$ is the number of elements in S.

In each configuration, D-MRC finds isolated nodes and restricted nodes until a maximum of $Max.Iso$ nodes. Based on the decision on isolated/restricted node/link in the configuration, the new link weight is assigned. D-MRC finds a sufficient number of configurations to cover the failure of all the nodes and links. Limiting the number of isolated nodes, choosing the isolated and restricted links based on routing density, and modifying link weights in the different configurations are expected to yield shorter, more disjoint paths than in MRC [3].

Algorithm 1 D-MRC

$p \leftarrow 0$
$S \leftarrow \emptyset$ {S is isolated nodes in all configurations}
$R \leftarrow \emptyset$ {R is restricted links in all configurations}
while $N(S) < N(V)$ **do**
  $p++$
  $G_p \leftarrow G$ {G_p is the graph in configuration p}
  $S_p \leftarrow \emptyset$ {S_p is isolated nodes in configuration p}
  for all $v_i \in V$ **do**
    for all $v_j \in N^{v_i}$ **do**
      $w_p(v_i, v_j) \leftarrow w_p(v_i, v_j) + \frac{(RD(v_i,v_j))}{\max_{k \in N^{v_i}} RD(v_i,k)} W$
      {adding to weight proportional to the routing density}
    end for
    if $v_i \notin S$ **then**
      if $div(v_i)$=FALSE &$N(S_p) < Max.Iso$ **then**
        $R_c \leftarrow \emptyset$
        for all $v_j \in N^{v_i}$ **do**
          if $v_j \notin R$ **then**
            $R_c \leftarrow R_c \cup v_j$
            $w_p(v_i, v_j) \leftarrow \infty$ {isolated link}
          end if
        end for
        $v_R = \arg\min_{v_k \in R_c} RD\,(v_i, v_k)$
        $R \leftarrow R \cup (v_i, v_R)$
        $w_p(v_i, v_R) \leftarrow 2W$ {restricted link}
        $S_p \leftarrow S_p \cup v_i$
      end if
    end if
  end for
  $S \leftarrow S \cup S_p$
end while

Two fields in the packet header to enable packet forwarding are employed. Backup topology indicator (BTI) field indicates which topology is being used for forwarding this packet. If BTI is 0, the packets are forwarded to $NH_p(i, d)$. Otherwise, the packets are forwarded to $NH_b(i, d)$ indicated by BTI. The switching number (SN) field indicates how many times a backup topology is switched while this packet has been forwarded. D-MRC allows switching topologies multiple times to increase the chance of creating a disjoint path from the primary path. In order to avoid potential loops in routing, the number of backup topologies utilized in routing a packet is limited to maximum switching number (MSN).

Every backup configuration is a connected graph. In a single backup configuration, routing on any given backup topology guarantees the delivery of a packet to the destination without a routing loop. Multiple backup topologies may be employed to increase the disjointness of the backup path with the primary path. However, when a packet is switched among multiple backup topologies, routing loops can occur and this is the reason for limiting the MSN such that the packet can be eventually delivered. For constructing a disjoint path, D-MRC use an alternate topology if the $NH(i, d)$ are identical in the primary configuration and the current configuration that is being employed for routing. In addition, if BTI is not 0, but $subroot_{snk}(i, d)$ is different from $subroot_{snk}(NH_b(i, d), d)$, BTI is changed to 0, and the packet is forwarded to $NH_p(i, d)$.

The state diagram in Fig. 3 shows the steps that are taken in a node's forwarding process. First, packets that are not affected by the failure are forwarded to primary next hop. Special steps are only taken for packets that would be forwarded along a backup path (BTI$\neq$0).

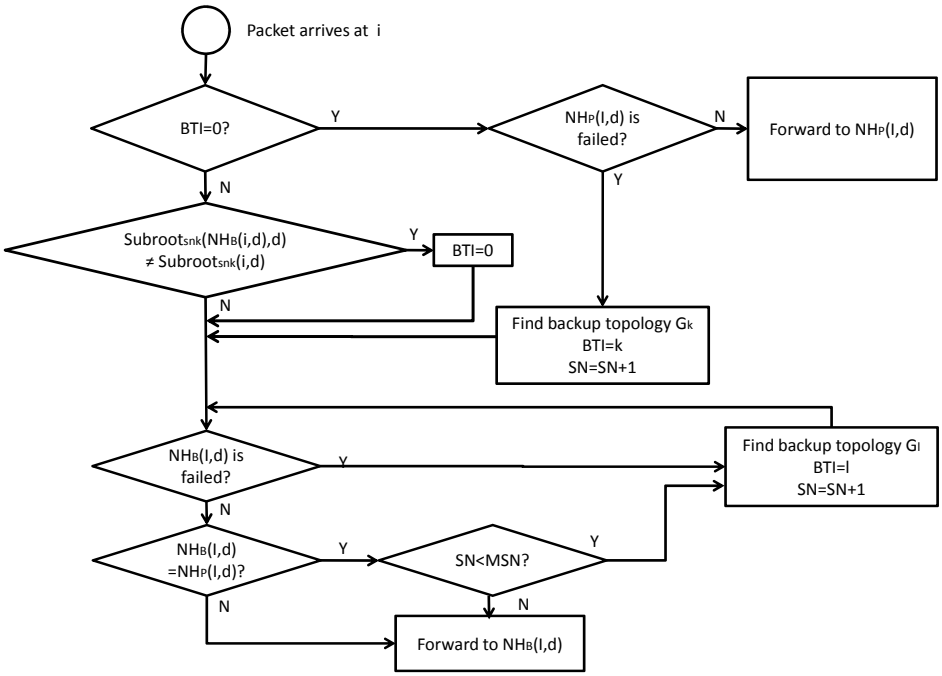


Fig. 3. D-MRC forwarding

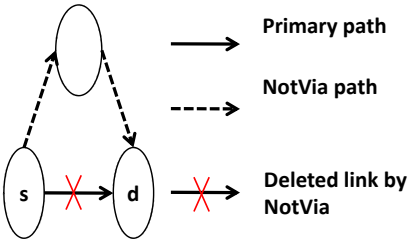
2. Disjoint NotVia (D-NotVia)

Let d' denote $subroot_{snk}(s,d)$. $dist(s,d)$ is hop count of the shortest path between s and d. It is assumed that the minimum node degree in the topology is 2, guaranteeing at least two link-disjoint paths for any source-destination pair.

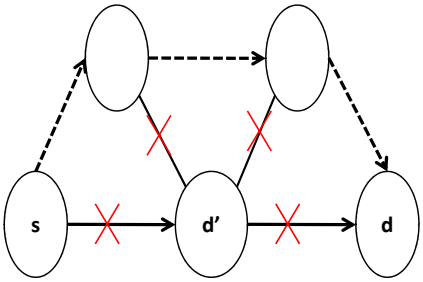
For constructing a disjoint path, D-NotVia uses $NV_{link}(d',d)$ or $NV_{node}(d',d)$ first. $NV_{link}(d',d)$ or $NV_{node}(d',d)$ guarantees the disjoint path if $dist(s,d) \leq 2$

Theorem 4. *If $dist(s,d)$ is 1, $NV_{link}(d',d)$ guarantees a disjoint path (case 1 in Fig. 4(a)). If $dist(s,d)$ is 2, $NV_{node}(d',d)$ guarantees a disjoint path (case 2 in Fig. 4(b)).*

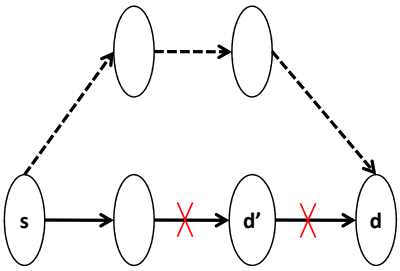
Proof. If $dist(s,d)$ is 1 or 2, $NV_{link}(d',d)$ and $NV_{node}(d',d)$ remove all the links on



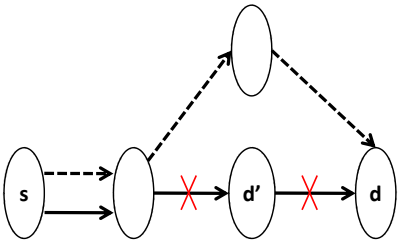
(a) Case 1: Using $NV_{link}(d', d)$ when dist(s, d)=1



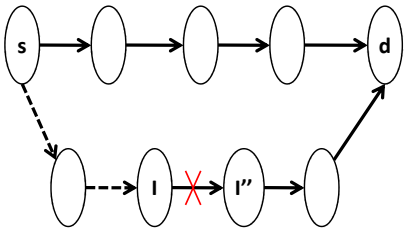
(b) Case 2: Using $NV_{node}(d', d)$ when dist(s,d)=2



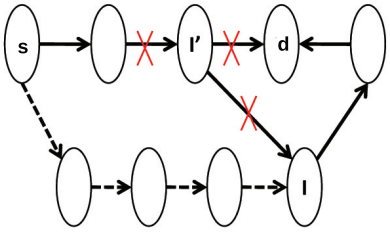
(c) Case 3: Using $NV_{node}(d', d)$ when dist(s,d)$\geq$ 2



(d) Case 4: Failed case using $NV_{node}(d', d)$ when dist(s,d)$\geq$ 2



(e) Case 5: Using $NV_{node}(N^I, I)$ is used when $I_{snk} \cap I_{src} \neq \emptyset$



(f) Case 6: Using $NV_{node}(I', I)$ is used when $I_{snk} \cap I_{src} = \emptyset$

Fig. 4. Examples of D-NotVia

the primary path. As a result, the backup path using $NV_{link}(d', d)$ and $NV_{node}(d', d)$ never uses the links on the primary path and hence a disjoint path is constructed. □

If $dist(s, d)$ is greater than 2, $NV_{node}(d', d)$ is used, but it does not guarantee a disjoint path.

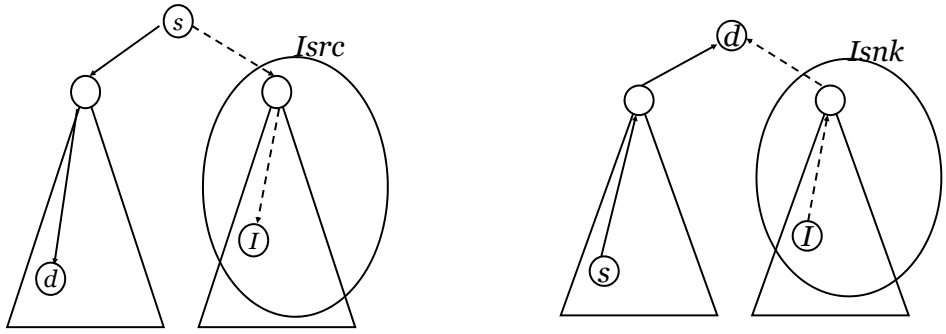
The case 3 in Fig. 4(c) finds a disjoint path, but the case 4 in Fig. 4(d) fails to find a disjoint path with this method. The failed case forwards to the node in the same tree which is used in the primary path.

Alternatively, when $NV_{node}(d', d)$ fails to find a disjoint path, D-NotVia finds an intermediate node (node I) whose primary path to node d does not have a common link of the primary path from s to d. Node I plays a role as a stepping stone router for creating a backup disjoint path between s and d. NotVia addresses is used to deliver packets to node I. After the packet reaches node I with NotVia address, the primary path to the destination is used from node I. The questions are how to find such a node I and how to guarantee the path between node s and node I to be disjoint from the primary path from s to d. The strength of forwarding to node I with NotVia address is gaurantee of decapsulting the original packet and reaching at d [32].

For each s-d pair, define:

- $I_{snk}(s, d)$: the nodes in subtrees which do not contain s in the sink routing tree destined to d
- $I_{src}(s, d)$: the nodes in subtrees which do not contain d in the source routing tree rooted from s

Forwarding to $I \in I_{src}(s, d)$ detours the failed link and guarantees the first hop of the backup path is different from the first hop of the primary path. Forwarding from $I \in I_{snk}(s, d)$ to d uses nodes in a different subtree in the sink routing tree. So, the path from $I \in I_{snk}(s, d)$ to d will be disjoint with the primary path.



(a) I_{src} (b) I_{snk}

Fig. 5. I_{src} and I_{snk}

The nodes in intersection of $I_{snk}(s,d)$ and $I_{src}(s,d)$ are candidates for node I.

Theorem 5. *Forwarding along the primary path from s to I and the primary path I to d constructs a disjoint path from s to d, when I belongs to the intersection of $I_{snk}(s,d)$ and $I_{src}(s,d)$.*

Proof. Since $I \in I_{snk}(s,d)$, it guarantees that the primary path from $I \in I_{snk}(s,d)$ to d is disjoint from the path from s to d (figure 5(b)). Since $I \in I_{src}(s,d)$, it guarantees that the first hop of the primary path from s to I is not a common link with the primary path from s to d (figure 5(a)). □

If candidates for node I exist, i.e., the intersection is not null, I whose $dist(s,I)+dist(I,d)$ is the shortest is selected node I in order to decrease the backup path length. To forward from s to I, $NV_{node}(I'',I)$ is used, where I'' is the neighbor node of I, but is not I' ($I' = subroot_{snk}(s,I)$). Case 5 in Fig. 4(e) is an example of finding node I.

If the intersection of $I_{snk}(s,d)$ and $I_{src}(s,d)$ is null, it means the first hop of the primary path from s to $I \in I_{snk}(s,d)$ is a common link with the primary path from s to d. In this case, $NV_{node}(subroot_{snk}(s,I), I \in I_{snk}(s,d))$ is searched to find node I.

If forwarding by $NV_{node}(subroot_{snk}(s, I), I \in I_{snk}(s, d))$ does not use the first-hop of the primary path from s to d, I is selected as node I, and then we use $NV_{node}(I', I)$ where I' is $subroot_{snk}(s, I)$. In case 6 in Fig. 4(f), the primary path from s to I fails to make a disjoint path, but the path computed by $NV_{node}(I', I)$ succeeds in finding a disjoint path.

If we cannot find a node I, $NV_{node}(NH_p(s, d), d)$ is used to create the backup path (which may not be completely disjoint from the primary path).

The strength of D-NotVia is

- the forwarding method to the intermediate node is simpler than D-MRC.
- the complexity of the scheme for computing the disjoint path is less than the existing complexity of computing routing table entries for NotVia addresses.

3. Overhead analysis

In this section, the complexity of the proposed schemes is analyzed.

The computational overhead is the overhead of constructing disjoint path on the proposed scheme's frame works. The complexity of computing the source routing tree is $O(Vlog(V) + E)$.

D-MRC computes backup topologies with complexity $O(B\delta_N V^2)$ where B is the number of backup topologies, and δ_N is the node degree. After that, all nodes compute the backup paths in all topologies with complexity $BVO(Vlog(V) + E)$. As a result, the computational overhead of D-MRC is $O(B\delta_N V^2) + BVO(Vlog(V) + E)$.

NotVia computes the routing trees for all NotVia addresses. This complexity is $(V^2 + E)O(Vlog(V) + E)$. The complexity of finding the intersection of I_{snk} and I_{src} is $O(Vlog(V))$. As a result, the computational overhead of D-NotVia is $(V^2 + E)O(Vlog(V) + E)$.

The memory overhead is the overhead of constructing and maintaining the routing table. In D-MRC, the number of entries in the routing table is proportional to the number of backup topologies, for a total of $O(BV)$ at each node. In addition, D-MRC has to maintain information about the topology in which a node is isolated.

In D-NotVia, the number of additional entries in the routing table is proportional to the number of NotVia addresses, $O(V^2 + E)$. In addition, D-NotVia should have information about which NotVia address corresponds to which link or node failure.

The packet overhead is the amount of additional bits in a packet for the proposed schemes. In D-MRC, BTI (2-4 bits), and SN (2 bits) are required.

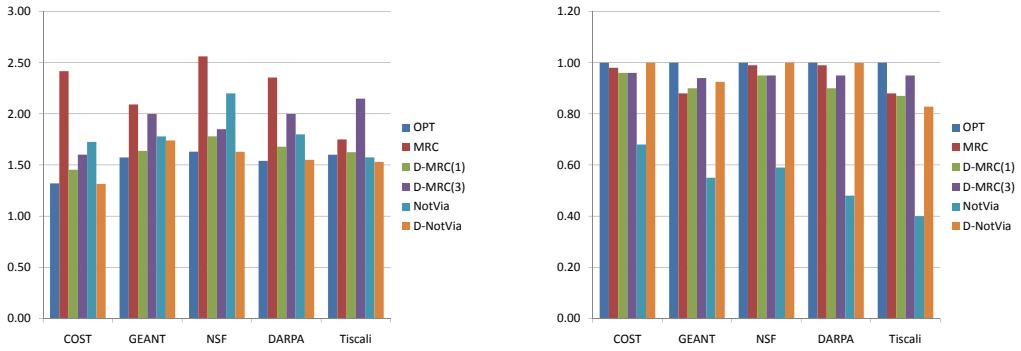
In D-NotVia, packet encapsulation is employed to redirect packets to NotVia addresses. Even though this does not require additional fields in the packet headers, packet payloads need to be smaller to avoid fragmentation after encapsulation.

C. Simulation

The different schemes are evaluated for simultaneous failure recovery and disjoint-path routing in this section. A number of network topologies are employed in this study. The networks used for this evaluation are COST 239 (11 nodes, 26 links), GEANT (19 nodes, 29 links), NSF (14 nodes, 22 links), DARPA (20 nodes, 32 links), and Tiscali (40 nodes, 67 links) networks [33].

1. Constructing disjoint paths

The number of backup topologies required for D-MRC is compared to MRC [3] in Table I. The maximum number of the isolated nodes in a single backup topology is 3 for COST239, GEANT, and NSF, and 4 for DARPA and Tiscali. D-MRC requires more backup topologies compared to MRC. It is because of the restriction of the



(a) Stretch of the secondary path (b) Disjointness of the secondary path

Fig. 6. Stretch and disjointness of the secondary path

maximum number of the isolated nodes in a single backup topology.

Table I. The number of backup topologies in MRC and D-MRC

	COST239	GEANT	NSF	DARPA	Tiscali
MRC	3	6	4	5	7
D-MRC	6	8	6	6	15

With the pre-computed backup topologies, the path stretch based on the average length of the backup paths and disjointness for all the source-destination pairs in the networks are measured. Path length is measured by the number of hops from the source to destination. The backup paths are computed by MRC [3], D-MRC (MSN=1), D-MRC (MSN=3), NotVia [4], D-NotVia, and OPT. To compute the disjoint path with MRC and NotVia, the first hop of the primary path is regarded as the failed link. We also show the results for optimal disjoint path computation OPT (computed by removing all the links of the primary path for each source-destination pair) for comparison purposes.

The results of creating disjoint paths using MRC and NotVia are shown in Fig.

6.D-MRC achieves similar disjointness to MRC. However, D-MRC has much lower stretch cost. When multiple backup topology switching is allowed to be used for creating a disjoint backup path, an improvement is seen in disjointness of backup paths, at the cost of slightly longer paths. It is also observed that nearly 100% of the time disjoint backup paths can be created using D-MRC in all the networks. In the networks such as COST239 and NSF network which have higher node degree, disjointness of the backup paths is very close to 1. In the remaining networks, allowing multiple backup topologies to be employed in constructing the backup path improves disjointness without significantly increasing the backup path length.

The disjointness of NotVia is poor because it uses primary path after forwarding to the first hop of the primary path. In D-NotVia, the source node selects NotVia address considering the disjointness of the backup path, so it improves disjointness compared to NotVia. D-NotVia shows similar disjointness to D-MRC. D-NotVia has smaller stretch on average than D-MRC, because it uses the primary path after forwarding to the node destined with NotVia address.

It is also observed that the stretches for D-MRC and D-Notvia schemes are not much larger than that of the optimal OPT scheme. In some networks, D-NotVia and D-MRC show smaller stretch than OPT because their disjointness is not 1 for all the backup paths.

2. Applying to multipath routing for load balancing and QoS

In this section, we show the utility of computed disjoint paths by considering multi-path routing. Primary path and backup path from D-MRC and D-NotVia are applied to DEFT [34] in order to exam how well the disjoint multipath contributes to load balancing. DEFT assigns flows to next-hops with the probabilities that decrease exponentially with the extra length of the path compared to the shortest path.

Hot source and hot sink scenarios are considered. Traffic demands from a single source are doubled, while the others are not changed in the hot source scenario. Similarly, traffic demands to a single sink are doubled, while the others are not changed in the hot sink scenario.

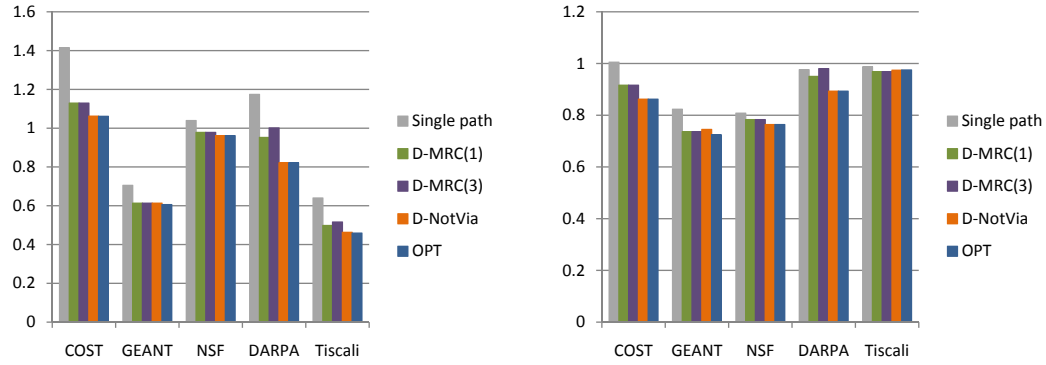
The given traffic demands are computed until the link utilizations are between 0.4 and 0.6 with single path routing. During the simulation, only one node can be selected as hot source or hot sink with a probability of $1/V$ at a time.

DEFT with the computed maximally disjoint paths (using D-MRC and D-NotVia) are compared to OSPF with a single route. The average link cost and the maximum link utilization as metrics are measured for evaluation. Link cost function (φ) given in [6] is employed with the utilization of link (i, j), $u(i, j)$.

$$\varphi'(i,j) = \left\{ \begin{array}{ccc} 1 & for & 0 \leq u(i,j) < 1/3 \\ 3 & for & 1/3 \leq u(i,j) < 2/3 \\ 10 & for & 2/3 \leq u(i,j) < 9/10 \\ 70 & for & 9/10 \leq u(i,j) < 1 \\ 500 & for & 1 \leq u(i,j) < 11/10 \\ 5000 & for & 11/10 \leq u(i,j) \end{array} \right\}. \tag{2.5}$$

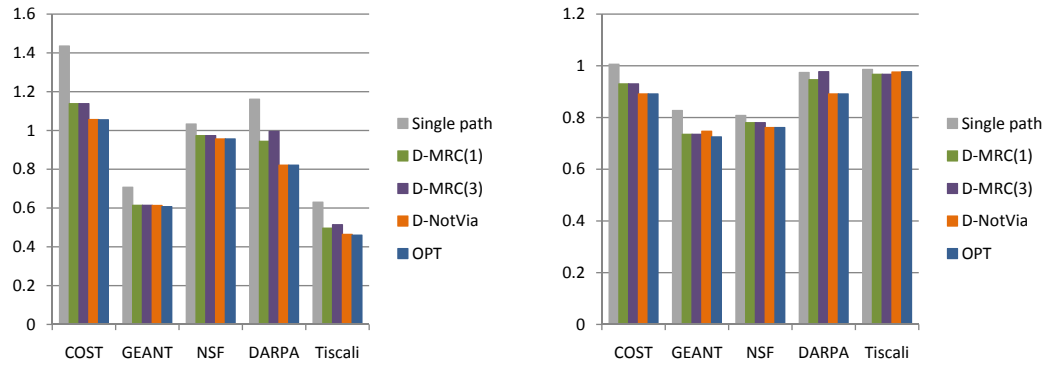
The results are shown Fig. 7 for hot source scenario and Fig. 8 for hot sink scenario. Single path routing shows the highest average link cost and maximum link utilization. D-MRC has lower average link cost and lower maximum utilization than single path but higher than OPT. The average link cost and maximum utilization of D-NotVia multi-path routing is very close to that of OPT with multi-path routing.

Primary path and backup path by D-MRC and D-NotVia is applied to the shortest-widest path routing [35] in order to examine how well the computed dis-



(a) Average link cost in hot source senario (b) Maximum link utilization in hot source senario

Fig. 7. Average link cost and maximum link utilization in hot source scenario



(a) Average link cost in hot sink senario (b) Maximum link utilization in hot sink senario

Fig. 8. Average link cost and maximum link utilization in hot sink scenario

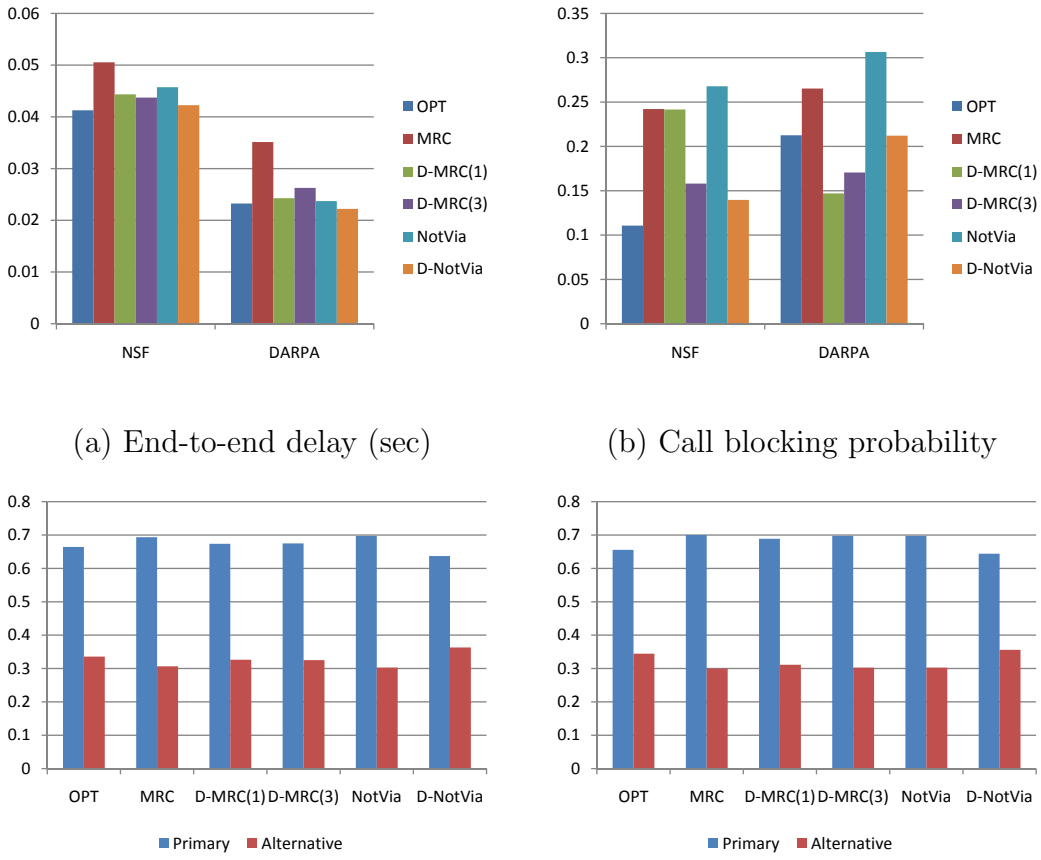
joint multipath contributes to improving QoS. The shortest-widest path finds feasible paths with the largest available bandwidth. If there are several paths, the path with the minimum hop count is selected. In this simulation, the shortest-widest path routing measures the minimum available bandwidth of the primary path. If the measured bandwidth is greater than the required bandwidth, the primary path is used. When the available bandwidth of the primary path is less than the required bandwidth, the shortest-widest path routing measures the available bandwidth of the alternative path. If the measured bandwidth is greater than the required bandwidth, the alternative path is used. If both paths cannot provide the required bandwidth, the flow or call is blocked or dropped. We measure the end-to-end delay, the fraction of the time primary/secondary path is selected, and the percentage of calls blocked.

As shown Fig. 9(a), the end-to-end delay is high in MRC and NotVia due to high stretch. D-MRC and D-NotVia have lower end-to-end delay than MRC and NotVia due to lower stretch. The end-to-end delay of OPT is the best.

As shown Fig. 9(c) and 9(d), the fraction of time alternative path is chosen is high in the proposed scheme and OPT. Also, Fig. 9(b) shows that the percentage of blocked calls is lower in D-MRC, D-NotVia and OPT. It is observed that higher disjointness of alternative path results in choosing the alternative path successfully more often and reduces the call blocking probability.

3. Applying to failure recovery

In this section, the average length of recovery paths against link/node failures is measured. The backup paths constructed in D-MRC and D-Notvia is used for tolerating failures and compare them against the failure recovery paths constructed in MRC and NotVia. All source-destination pairs and all link/node failures are not considered. Only source-destination pairs whose paths contain the failed link/node



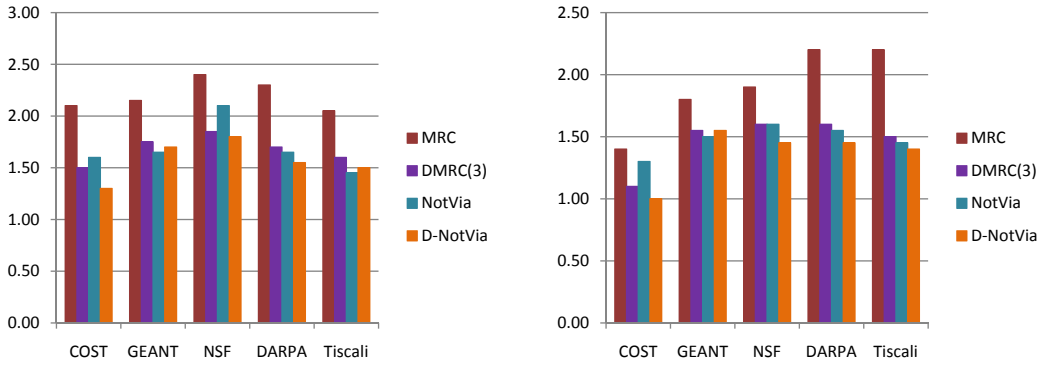
(a) End-to-end delay (sec)

(b) Call blocking probability

(c) Path selection probability on NSF network

(d) Path selection probability on DARPA network

Fig. 9. End-to-end delay, call blocking, and path selection probability on realistic topologies



(a) Stretch of the proposed schemes for link-failure

(b) Stretch of the proposed schemes for node-failure

Fig. 10. Stretch of the proposed schemes for link- and node-failure

is counted for this evaluation. The results of this comparison are shown in Fig. 10. It is observed that D-NotVia has lower stretch than NotVia except in GEANT and Tiscali networks. D-MRC has lower stretch than MRC because it selects the failed nodes in backup configurations carefully and since D-MRC employs more topologies (shown in Table I). D-NotVia has lower stretch than D-MRC.

In sum, it has been shown that the proposed algorithms for constructing disjoint paths in MRC and NotVia frameworks can build secondary paths that are nearly disjoint with the primary paths while providing secondary paths that are close to optimal (stretch similar to OPT). It is also shown that the constructed disjoint paths can be used with multipath routing algorithms to diffuse traffic hot spots to reduce maximum link utilizations. The constructed disjoint paths can be used for fast recovery from link/node failures with small stretch until the routing tables are recomputed taking the failures into account.

D. Related work

Multi-topology framework is investigated in [36]. Based on this framework, several QoS and failure recovery routings are proposed. QoS routing based on multi-topology is studied in [37]. In this scheme, the packet is delivered on different topologies according to priority. Failure recovery is not considered in their work. Schemes for load-balancing after proactive failure recovery are proposed in [38] and [39]. In these approaches, traffic engineering is applied to alternate topologies. rMRC, a scheme for increasing the diversity of backup topologies with no isolated links, but only restriced links, is proposed in [40]. These approaches based on [3] improve load balance and path selection diversity for failure recovery, but the disjointness of alternative paths is worse and stretch is higher than D-MRC, because they do not consider disjointness of paths [41].

Path splicing [31] provides multi-path routing through source level control of derouting a packet from the primary path. In path splicing, the link weight is randomly changed to improve the path disjointness, which is shown to vary from 20% to 100%. Compared to Path splicing, D-MRC and D-NotVia achieve higher disjointness of alternate paths. Red-blue tree construction [42] is proposed for disjoint multi-path routing. While this approach provides disjoint paths, the primary path may not be the shortest cost path and hence may result in providing higher cost even when no failures exist. A failure recovery scheme, using disjoint paths of coloring trees, is proposed in [43, 44]. Red-blue tree construction provides maximaly disjoint paths, but does not provide shortest primary paths.

LFA [5] is light-weight failure recovery scheme, but LFA does not guarantee a disjoint path from the primary path. Additional conditions can guarantee a disjoint path [41], but LFA next hop does not always exist at a node to all the destinations.

According to an analysis on real ISP topologies, over 40% links and nodes are not protected by LFA [45].

Classical algorithms for computing disjoint backup paths remove or reverse links along the primary path. This approach has high complexity and is hard to implement in the manner of hop-by-hop routing. Disjoint path routing schemes are proposed in [30] and [29]. However, these schemes constrain the construction of the primary path, hence cannot work with the currently used routing protocols.

The idea of using node I in D-NotVia is similar to [32]. However, only 40% of the time the intermediate nodes are employed in this approach instead of 100% of the time in steeping stone routing. In addition, the NotVia infrastructure provides for decapsulation at the network layer.

MADSWIP [46] provides maximally disjoint paths. Since this approach is based on [29], it constrains the computation of the primary path. Our work here tries to utilize MRC and NotVia frameworks for computing maximally disjoint secondary path without constraining the computation of the primary path.

CHAPTER III

REDUCING OVERHEAD OF LINK STATE UPDATE FOR WARDROP EQUILIBRIUM IN NETWORKS

In this chapter, the technique of reducing the cost of propagating dynamic link metric information across the network is proposed while ensuring stability of the routing algorithm. The threshold-based update, proposed for propagating link metric information only if the change of the dynamic link metric is high, guarantees that the observed path quality will be within an error bound of the optimal path quality, and it reduces the cost of propagating link cost information significantly.

A. Threshold based updates

Most earlier approaches assume that nodes measure the state of their links at regular intervals and propagate this information to the other nodes in the network. The nodes compute new routing tables or new traffic distribution ratios when all the information is received. In order to keep the information from getting too stale, the measurements and the propagation of the link state are carried out at regular intervals. The cost of propagating this information and the staleness of link state are controlled by controlling the frequency or rate of measuring and the propagation of the link state.

A possible approach to reducing the cost of updates is based on observing the local link state. Every node keeps track of the last link state that is propagated to the rest of the network. When the currently measured link state differs from the last updated state considerably, and the difference exceeds a threshold, only then does the node propagate the link state information to other nodes in the network. We call such a policy *threshold based updates*. The thresholds can be based on allowable

absolute error in link state or on the maximum allowable relative error. For example, thresholds can be such as 1ms or 20%. Absolute error thresholds may not be universally applicable. A 1ms error threshold may be reasonable when link delay is say 10ms, but may not be reasonable when link delays are in the range of 100ms or 1ms. Relative error thresholds can cover wide range of link states. However, relative thresholds can be problematic as the links get heavily loaded. While, in this regime, it may be necessary to propagate information more quickly in order to distribute the load to other parts of the network and higher thresholds would not be beneficial.

In order to accommodate all the conflicting needs, we pursue a policy here that tries to combine both absolute and relative error thresholds. A node propagates its link state if its current link state exceeds the minimum of absolute or relative error thresholds. More formally, if $|l_i - lj| \geq min(e_{abs}, e_{rel}l_j)$, where l_j is the last link state that is propagated by the node, l_i is the current link state and e_{abs} and e_{rel} are the absolute and relative error thresholds, then link state l_i is propagated and remembered locally as the last propagated link state.

The rationale for the threshold based update policy is simple: propagate link state only when not propagating the state will lead to errors beyond acceptable tolerance limits. With such an approach, we expect that we can bound the error in the link state while reducing the cost and the number of updates of link state across the network. While triggered updates of link state are used, for example, in OSPF routing on link up/down events and available link bandwidth changes in [20], we consider granular change in link state and directly relate the impact of the thresholds used in determining the propagation of link state to the final quality of routing goals (both through analysis and simulations).

The allowed or acceptable error thresholds may depend on the link state and the role played by the link state in the routing algorithm. Again, to demonstrate

the potential viability of such threshold based updates, we will focus on one routing algorithm, Wardrop routing [9, 24], one of the dynamic routing approach employing link delays as a link cost metric. In Wardrop routing, the traffic is split across available paths in such a way as to equalize the delay across all the available paths at a node. The traffic splitting can be done at the end hosts [24] or further split at the routers in the network as traffic moves from one hop to the next [9].

The allowed error in link state updates directly gets reflected in allowed error in the path delay metrics used in routing. It is assumed that nodes operate synchronously in order to make the analysis portion simple. As the link state propagation gets delayed, the routing decisions can be made on stale information. As inaccuracy is allowed in link delays, the resulting decisions can be erroneous, potentially leading to oscillations, where stability could be guaranteed with exact information.

The following sections answer several questions as we go forward: (a) how does the allowed link state update error influence the maximum observed path delay? This reflects on the path quality degradation as a result of threshold based updates. (b) can we still guarantee convergence of Wardrop routing, now albeit a looser notion of convergence i.e., do different paths converge to approximately equal delays (the errors or bounds being determined by (a))? (c) how much gain can be had in reducing the cost of link state updates through threshold based updates?

B. Convergence and error analysis

We consider a network represented by a graph $G = (V, E)$. The traffic demand is specified by a set of commodity flows K with commodity $k \in K$ corresponding to traffic λ_k from a source $s_k \in V$ to a destination $d_k \in V$. Let P be the set of all allowed paths connecting source-destination pairs and $P_k \subseteq P$ be the set of paths connecting

s_k to d_k. Note that each path $p \in P$ is a set of edges $e \in E$. We assume that the maximum length of a path in the network is bounded by L_{max}. For simplicity, we assume that P_k are disjoint. The routing state of the network is given by a flow vector $\vec{x} = \{x_p\} \in \mathbb{R}^{|P|}$, where $|P|$ denotes the cardinality of P. For a flow vector to correspond to a feasible routing state requires $\vec{x} \geq 0$ and $\sum_{p \in P_k} x_p = \lambda_k$. The flow on an edge $e \in E$ is denoted $x^e = \sum_{p \ni e} x_p$. The latency of an edge e supporting flow x^e is specified by a function $l_e(x^e)$, and that the function is upper bounded by $l_{\max}$ for all $e \in E$. The latency of a path is then given by $l_p(\vec{x}) = \sum_{e \in p} l_e(x^e)$. We denote $l^k_{\min}(\vec{x}) = \min_{p \in P_k} l_p$.

For simplicity, we focus in this section on the class of adaptive routing policies whose convergence properties under a model with periodic updates have been studied in [47]. Consider a fluid model with an infinite number of agents each make routing decisions for an infinitesimal fraction of the traffic on the network. A flow vector $\vec{x}$ corresponds to x_p agents routing traffic over path p. The route used by each agent is revised periodically at discrete points in time based on the available information about the link and path metrics. The current path delay metric for path p that is available to all agents (potentially with errors) is denoted $\hat{l}_p$. An agent controlling traffic belonging to commodity k and currently using path $p \in P_k$ samples a path $q \in P_k$ with probability σ_{pq} and switches to the path q, if it is better, with probability $\mu(\hat{l}_p, \hat{l}_q)$. In this chapter, only the subset of policies that are α-smooth is considered, i.e., policies that satisfy $\mu(\hat{l}_p, \hat{l}_q) \leq \alpha(\hat{l}_p - \hat{l}_q)$. In the case of the threshold based scheme, an agent decides that the sampled path q is better only if the sampled latency $\hat{l}_q$ is less than the current latency of path p, $\hat{l}_p$, by an error margin which will be discussed further in the sequel.

1. Convergence to approximate Wardrop equilibria

Through such threshold based updates, we do not aim to converge precisely to a Wardrop equilibrium but instead to an approximate equilibrium defined below.

Definition 6. *A flow vector $\vec{x}$ corresponds to a δ - approximate Wardrop equilibrium if: $x_p > 0$ only if $\exists$ a commodity k such that $l_p(\vec{x}) \leq l^k_{\min}(\vec{x}) + \delta$.*

As simulation study in the next section, such an approximate equilibrium allows the network to leverage the benefits of Wardrop routing at a fraction of the cost.

Lemma 7. *If link updates are propagated with an absolute error threshold $e_{abs} < \frac{\delta}{2L_{max}}$, and an agent shifts traffic from path p to q only if $(\hat{l}_p - \hat{l}_q) \geq 2L_{max}e_{abs}$, and σ_{pq} assigns positive probability to all paths, the flow vector $\vec{x}$ converges to a δ-approximate Wardrop equilibrium.*

Proof. Assume that the network is in a routing state $\vec{x}$ that is not a δ-approximate Wardrop equilibrium. When agents sample path with higher latency, no traffic is shifted. If an agent decides that path q is better than p based on the measured latencies, then $(\hat{l}_p - \hat{l}_q) > 2L_{max}e_{abs}$. Since link metrics are propagated whenever the change in link delay exceeds e_{abs} and the maximum path length is L_{max}, the maximum error in the sampled latency of a path is bounded by $L_{max}e_{abs}$. Thus, using an error margin of $2L_{max}e_{abs}$, it is guaranteed that if an agent decides to switch traffic from path p to q based on the propagated link metrics, then indeed $l_p > l_q$ also. Since any feasible path is sampled with positive probability, an agent not on the lowest latency path will eventually sample a better path and switch to it with a positive probability.

As shown in [47], the potential function

$$\Phi = \sum_{e \in \mathcal{E}} \int_0^{x^e} l_e(x) \mathrm{d}x \tag{3.1}$$

is a Lyapunov function which is minimized at the Wardrop equilibrium. When the network is not at a δ-approximate Wardrop equilibrium, there will eventually be a traffic shift between paths that lowers the potential function while no shifts that increase the function are possible. Thus, the gradient of the potential function is negative as long as the system is not in a δ-approximate Wardrop equilibrium. This in turn implies the lemma. □

2. Speed of convergence

In this part, we consider for simplicity the case with one commodity coupled with an adaptive routing scheme using

1. Uniform sampling: $\sigma_{pq} = |P|^{-1}$, $\forall p, q \in P$
2. Linear rule to switch traffic: $\mu(\hat{l}_p, \hat{l}_q) = \frac{(\hat{l}_p - \hat{l}_q)}{l_{\max}}$.

As in [47], the time is bound to reach a (δ, ϵ)-approximate Wardrop equilibrium, defined in [47] as:

Definition 8. *If at most ϵ agents use paths p such that $l_p(\vec{x}) > l_{\min}(\vec{x}) + \delta$, then $(\vec{x})$ corresponds to a (δ, ϵ)-approximate Wardrop equilibrium.*

Lemma 9. *Assume that agents sample alternate paths at a rate ω, and the threshold update policy is used with the absolute error threshold chosen such that $e_{abs} < \frac{\delta}{2L_{max}}$. Then, for the uniform sampling policy that shifts traffic between paths following a linear rule, the time spent in a routing state that is not a (δ, ϵ)-approximate Wardrop equilibrium is upper bounded by*

$$\frac{|P| l_{\max}^2}{\omega \epsilon \delta (\delta - 2L_{max} e_{abs})} \tag{3.2}$$

Proof. An agent using a path p such that $l_p(\vec{x}) > l_{\min}(\vec{x}) + \delta$ samples the cheapest path with a probability of at least $\frac{1}{|\mathcal{P}|}$. The agent shifts the controlled traffic to the cheapest path with probability of at least $\frac{(\hat{l}_p - \hat{l}_{\min})}{l_{\max}} \geq \frac{(\delta - 2L_{max}e_{abs})}{l_{\max}}$. Also, until reaching the (δ, ϵ)-approximate Wardrop equilibrium, there are at least ϵ such agents. Thus, the rate at which agents switch to the current minimum latency path thereby reducing their latency by at least δ is at least

$$\frac{\omega\epsilon(\delta - 2L_{max}e_{abs})}{|P|l_{\max}}, \tag{3.3}$$

and the rate at which the potential function decreases is then given by

$$\frac{\omega\epsilon\delta(\delta - 2L_{max}e_{abs})}{|P|l_{\max}}, \tag{3.4}$$

The potential function Φ is bounded above by $l_{\max}$ and below by 0. Thus, the time to reach the (δ, ϵ)-approximate Wardrop equilibrium is bounded by

$$\frac{|\mathcal{P}|l_{\max}^2}{\omega\epsilon\delta(\delta - 2L_{max}e_{abs})} \tag{3.5}$$

□

C. Simulation

In this section, the behavior of the fixed interval update scheme which sends link updates at fixed time intervals to that of the threshold based update scheme is compared. In the fixed interval update scheme, the interval of link update is set to $T = 2$ seconds. The link state update, and the traffic splitting ratio changes are based on the algorithm in [9].

It is noted that in order to slow down the propagation of update messages during rapid traffic changes, the link state measurements are controlled by a measurement

interval of $T = 2$ seconds or 1 second, even in the threshold based update scheme.

For the link state update, the fixed interval update scheme and the threshold based update scheme set the same interval, T. Since traffic (and hence link latencies) can be very bursty at short timescales, each node measures the current latency ($l(i,j)$), and then updates its link latency by computing an exponential moving average ($\widehat{l}(i,j)$) at every T to smooth out the noise.

$$\widehat{l}(i,j) = \gamma\widehat{l}(i,j) + (1-\gamma)l(i,j) \tag{3.6}$$

For propagating the link state and updating the path latency, we define $L(i,j,k)$ and $L_P(k,j)$.

- $L(i,j,k)$: the expected path latency from i to j via neighbor k.

$$L(i,j,k) = \widehat{l}(i,k) + L_P(k,j) \tag{3.7}$$

- $L_P(k,j)$:the expected path latency from k to j

$$L_P(k,j) = \sum_{n_j \in N(k,j)} p(k,j,n_j)L(k,j,n_j) \tag{3.8}$$

where $p(i,j,k)$ is the splitting ratio from i to j via k, and N^k is the neighbor of k.

$L(i,j,k)$ is computed and $L_P(i,j)$ is propagated to the network based on the update policy of the schemes. In the fixed interval update scheme, $L(i,j,k)$ is computed and $L_P(i,j)$ is propagated at every T. In the threshold based update scheme, $L(i,j,k)$ is computed at every T, but $L_P(i,j)$ is propagated only when the change of $\widehat{l}(i,j)$ is greater than $min(e_{abs}, e_{rel}\widehat{l}(i,j))$. In all simulation, e_{abs} is set to 7.5 msec and e_{rel} is set to 0.1. The maximum allowed path length,L_{max}, is 3.

The splitting ratio update is determined by the condition of the (δ,ϵ)-approximate Wardrop equilibrium. The traffic splitting ratios $p(i,j,k_1)$ are changed only when $L(i,j,k_1) - L(i,j,k_2) > e$ where $e = min(e_{abs}, e_{rel} * \widehat{l}(i,j)) * 2L_{max}$.

The amount of change (Δ) in traffic splitting ratios is given by, from [9],

$$\Delta = \kappa \left((1-\beta)\, p\,(i,j,k_1) + \frac{\beta}{|N(i,j)|} \right) \frac{L(i,j,k_1) - L(i,j,k_2)}{L(i,j,k_1) + \alpha} \tag{3.9}$$

The details about weigh shift factor(κ), virtual latency offset α, and exploration ratio (β) are desgined for preventing oscillations and exploring new paths. (Please refer [9]).

The topologies ranging from simple four node graphs to medium and large topologies in [33] are considered. Identical weights are assigned to links so that hop count is the routing cost, and it increases equal cost multiple paths. To simulate traffic on the networks, a workload is generated based on the Web workload generators in [48]. The workload mimics that generated by a user requesting a web page, and then remaining idle for a period while reading the page and then requesting another web page and so on. The sizes of the files (requests) are drawn from a heavy-tail distribution. This results in a mix of short-term flows and a considerable number of long-term flows.

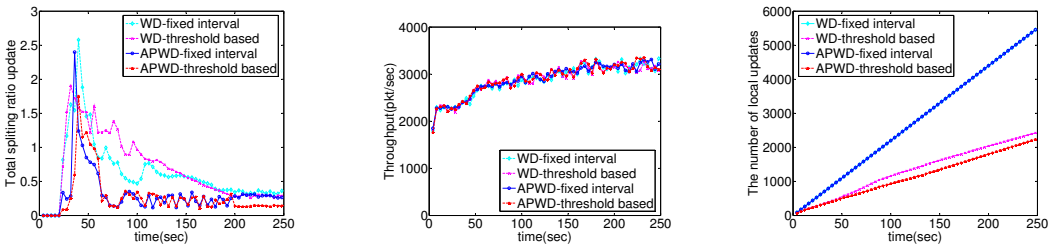
In all simulations, the threshold-based scheme is simulated along with the fixed interval update scheme to compare their performance. The following metrics are measured.

- Total splitting ratio changes: the sum of the splitting ratio change(Δ) at time t (for subsection 1)
- Throughput: the sum of the throughput at time t (for subsection 1)
- Splitting ratio: $p(r_1, r_4, r_2)$ and $p(r_1, r_4, r_3)$ (for subsection 2)
- Route utilization: $U(r_1{-}r_2{-}r_4)$ and $U(r_1{-}r_3{-}r_4)$ where $U(P)$ is the utilization of path(P) (for subsection 2)
- Path latency gap($G_p(t)$): $L(r_1, r_4, r_2)$-$L(r_1, r_4, r_2)$ at time t (for subsection 3-4)

- Cumulative path latency gap: $\sum_{i=1}^{t} G_p(i)$ (for subsection 3)
- The number of link updates: the cumulative number of update message (for subsection 1-4)

1. Comparison between approximate and Exact Wardrop routing

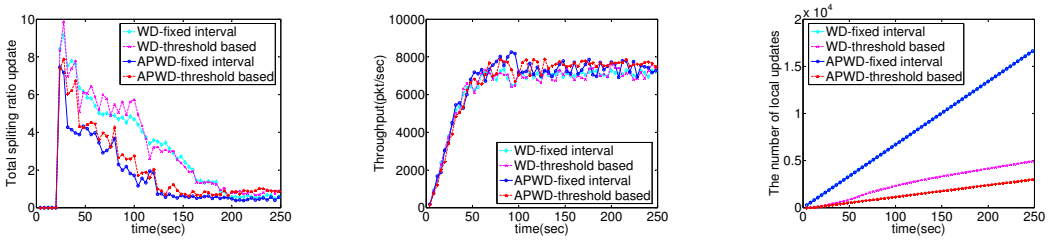
In this section, The approximate Wardrop routing (APWD) is compared with exact Wardrop routing (WD) on realistic networks: NSF topology with 14 nodes and 22 links for medium size network simulation, and tiscali topology with 40 nodes and 67 links for large size network simulation [33]. The results from simulations in other realistic topologies and other workloads are similar.



(a) Total splitting ratio changes (b) Throughput (c) The cumulative number of link updates

Fig. 11. Splitting ratio, number of link updates, and throughput on NSF topology.

The simulation results of both timer-based link updates and threshold-based updates with exact and approximate Wardrop equilibria are shown in Fig. 11 and Fig. 12. It is shown that approximate Wardrop routing reaches similar performance as exact Wardrop routing, within the allowed error bounds (shown in Fig. 11(b) and Fig. 12(b)). It is noted that both the schemes employed threshold based (or timer based) link updates and that the only difference in the two schemes is the goal for convergence. In the exact Wardrop routing case, the traffic splitting ratios are



(a) Total splitting ratio changes

(b) Throughput

(c) The cumulative number of link updates

Fig. 12. Splitting ratio, number of link updates, and throughput on Tiscali topology.

updated to make the differences in path delays go to zero and in the approximate case, the traffic splitting ratios are updated to make the differences in the path delays approach the allowed error bounds for the paths.

The sum of all route splitting ratio changes at all routers to test the oscilation in the schemes is measured. This measurement is expected to show quick convergence and small fluctuations after the convergence. Also, adjustment of splitting ratios is expected to contribute to the improvement of throughput of TCP connections.

In this experiment, 6 clients generate HTTP traffic between every source-destination pair during the simulation time (=250 sec). The splitting ratio chage is allowed after 20 sec.

Both schemes show small fluctuations of total splitting ratio changes after 70 seconds as seen in Fig. 11(a) and after 120 seconds as seen in Fig. 12(a). As the traffic splitting ratio is changed, the total throughput increased in both the networks, especially during the period between 30s and 60s. Even though both schemes show similar performance, the link update overhead of the threshold based update scheme is less than 50% compared to that of the fixed interval update scheme (in Fig.11(c) and Fig.12(c)).

The results show that approximate Wardrop routing provides similar perfor-

mance, with a slightly lower number of traffic splitting updates and slightly lower number of link updates, within each type of link update mechanism. It is also noted that the threshold based mechanisms required far fewer link updates than timer based link update mechanisms.

From here on, only approximate Wardrop equilibrium is considered in following experiments .

2. Convergence to approximate Wardrop equilibrium

The first simulated network topology consists of 4 routers. (As shown in Fig. 13). The 20 clients connected with r_4 download the HTTP files from the routers connected with r_1. There are two possible paths: r_1-r_2-r_4, and r_1-r_3-r_4 between r_1 and r_4.

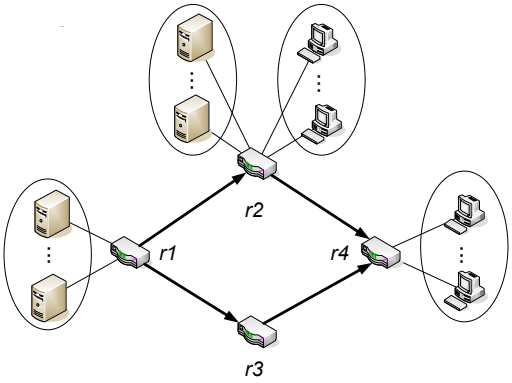
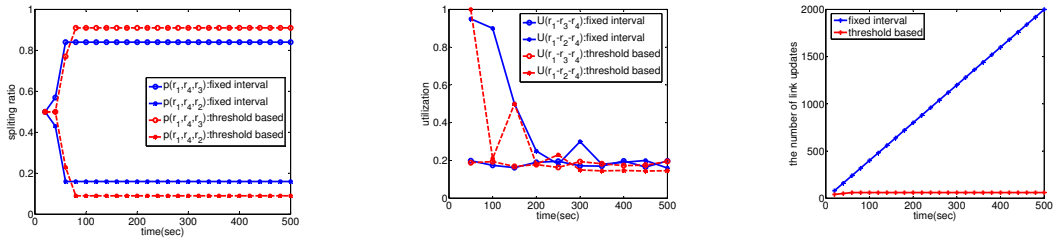


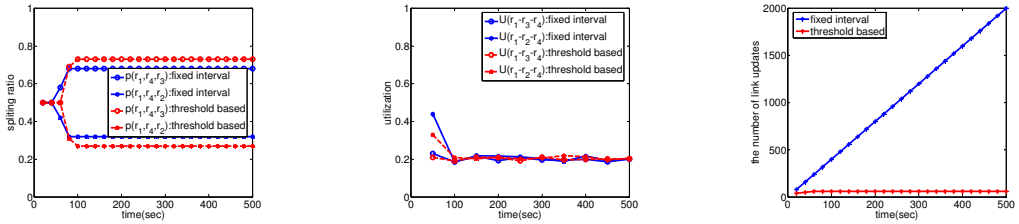
Fig. 13. Simple topology for approximate Wardrop routing simulations

To study the performance of the Wardrop routing scheme using threshold based updates, we examine the splitting ratio and the route utilization of the two routes over time under three cases. We assign bandwidths of 1, 5, and 10 Mbps to link(r_1, r_2) and link(r_2, r_4) while keeping the other links at 10Mbps. These three cases are denoted 1/10 Mbps, 5/10 Mbps and 10/10 Mbps. If the algorithm works well, we expect the load on both routes to be balanced. The optimal splitting ratio, ignoring the



(a) Splitting ratio to the routes (b) Route utilization (c) The cumulative number of link updates

Fig. 14. Splitting ratio, route utilization, and the number of link updates for the case of 1/10 Mbps

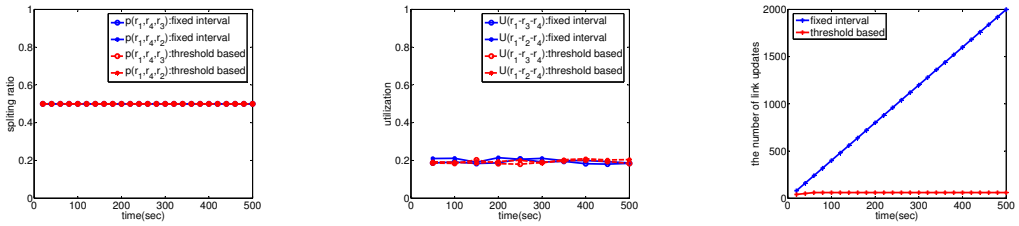


(a) Splitting ratio to the routes (b) Route utilization (c) The cumulative number of link updates

Fig. 15. Splitting ratio, route utilization, and the number of link updates for the case of 5/10 Mbps

traffic variability, should consequently converge to 0.09/0.91, 0.33/0.67, and 0.5/0.5 respectively.

In Fig. 14(a),15(a), and 16(a), both schemes converge to near optimal splitting ratios. Then we can see the route utilizations converge to nearly the same utilization on both paths in both schemes (in Fig. 14(b),15(b),and 16(b)). However, the number of link updates of the fixed interval update scheme is much higher than the number of link updates of the threshold based update scheme (in Fig. 14(c),15(c),and 16(c)). The fixed interval update scheme keeps sending the path latency information after



(a) Splitting ratio to the routes (b) Route utilization (c) The cumulative number of link updates

Fig. 16. Splitting ratio, route utilization, and the number of link updates for the case of 10/10 Mbps

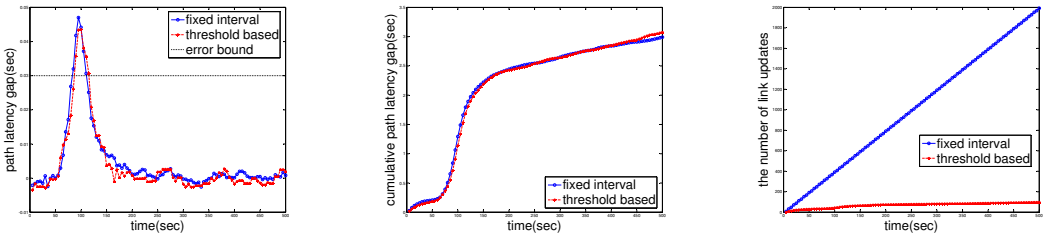
convergence. However, the threshold based update scheme sends the path latency information scarcely after convergence.

3. Impact of traffic bursts

It is assumed that the traffic demands are static in the previous subsection. However, the dynamic routing protocol should respond to the changes in the traffic rapidly. This simulation examines if the threshold based update scheme can adjust quickly to changes in the traffic.

This simulation adds a traffic burst to different links to see the impact of how quickly the traffic burst results in making traffic adjustments across the two paths and how much overhead is required for adjusting to changes in traffic. Additional traffic on one or more links is generated in the network of Fig. 13. At the begining of the simulation, clients connected to r_4 download HTTP files from the routers connected with r_1. Additional HTTP clients, become active after a burst start time (=50 sec). Before the burst of traffic due to additional HTTP clients, the traffic is balanced well between the two routes. The traffic ratios are changed as a result of this traffic burst.

In scenario 1, all link capacities are 2 Mbps and the burst traffic is put on link



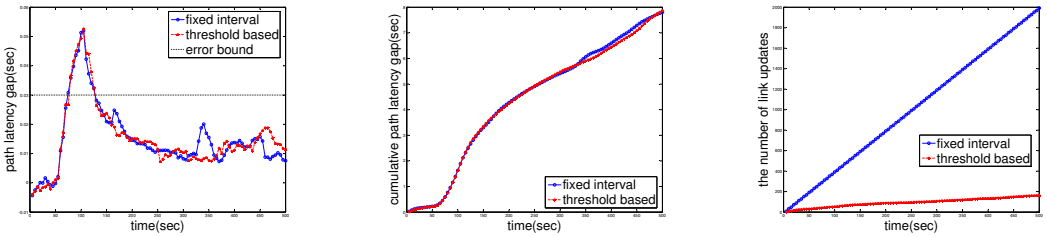
(a) Path latency gap (b) Cumulative path latency gap (c) Cumulative number of link updates

Fig. 17. The path latency gap and the number of link updates for scenario 1

(r_1, r_2). In this case, r_1 detects the traffic change through link latency measurements. In addition, r_1 has an alternative path (path$(r_1 - r_3 - r_4)$) avoiding the congested path (path$(r_1 - r_3 - r_4)$).

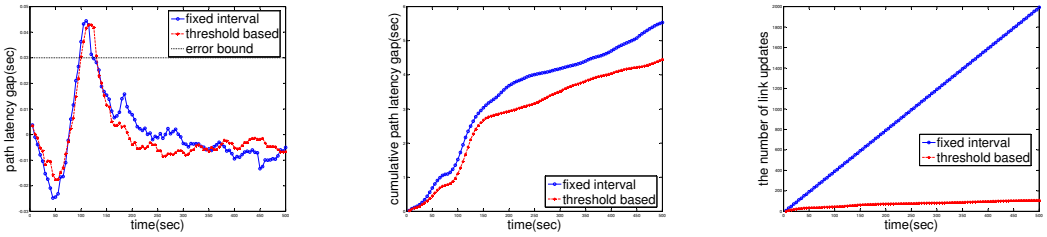
In scenario 2, all link capacities are 2Mbps and the burst traffic is put on link(r_2, r_4). Node r_1 cannot detect the traffic change directly. Instead, r_2 can detect the traffic change, but it does not have any alternative path. In this case, r_2 propagates the traffic change with link updates, and r_1 changes the splitting ratio based on the received updates. We examine how fast link update is propagated with the threshold based update scheme.

In scenario 3, we consider different link capacities by setting link capacities on link(r_1, r_3) and link(r_3, r_4) to 4Mbps and 4/3Mbps respectively and the burst traffic is put on link(r_1, r_2) while the capacities of link(r_1, r_2) and link(r_2, r_4) are 2 Mbps. In this setting, the path latencies of two paths is the same as before the burst traffic. We put the burst traffic on link(r_1, r_2). Similar to scenario 1, r_1 detects the traffic change and it has an alternative path. However, the link update is important in this scenario. Since, the capacity of link(r_3, r_4) is less than that of link(r_2, r_4), swithcing the traffic on link(r_1, r_2) to link(r_1, r_3) can cause another congestion on link(r_3, r_4). The splitting ratio change is dependent both on the link state update in this scenario.



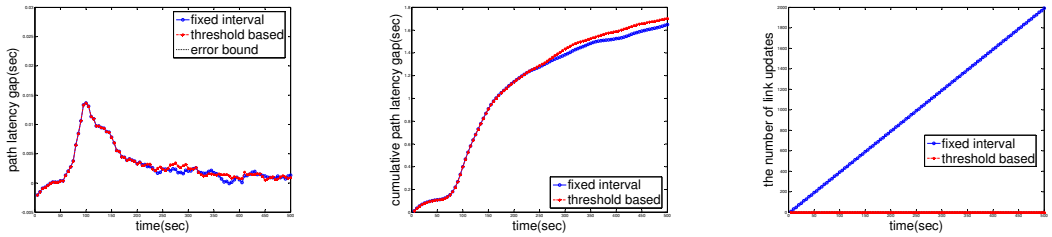
(a) Path latency gap (b) Cumulative path latency gap (c) Cumulative number of link updates

Fig. 18. The path latency gap and the number of link updates for scenario 2



(a) Path latency gap (b) Cumulative path latency gap (c) Cumulative number of link updates

Fig. 19. The path latency gap and the number of link updates for scenario 3



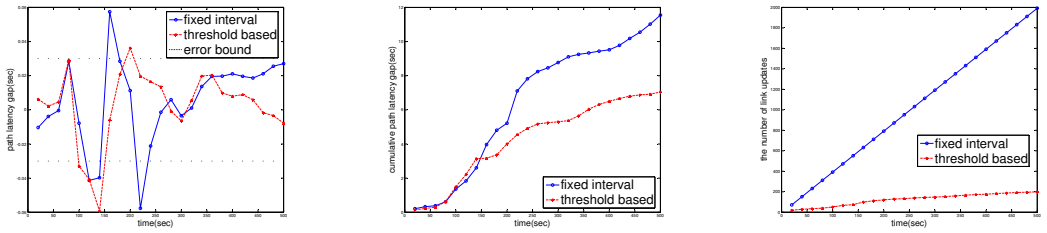
(a) Path latency gap (b) Cumulative path latency gap (c) Cumulative number of link update

Fig. 20. The path latency gap and the number of link updates for scenario 4

In scenario 1,2 and 3, the fixed interval update scheme and the threshold based update scheme both change the traffic splitting ratios. The path latency gap ($|L(r_1, r_4, r_2)$-$L(r_1, r_4, r_3)|$) is measured. The behaviors of the path latency gap are similar in both schemes (in Fig. 17(a),18(a), and 19(a)). However, the difference in the number of link updates in the two schemes is significant as seen in Fig. 17(c),18(c), and 19(c).

In scenario 1, 2, and 3, the link utilizations are high before the burst traffic. If the link utilization is high when the additional burst of traffic joins the link, the link latency is likely to change and hence impact the path latency. However, if the link utilization is low before the new burst of traffic joins the link, the path latency may not be affected much. As a result, the traffic splitting ratio across available paths may not change. The threshold based update scheme does not exchange the link latency information in this case, so we can reduce the overhead. For simulation of this case, all link capacities are 4 Mbps and the burst traffic is put on link (r_1, r_2) (we refer this as scenario 4).

Under the low utilization scenario, the burst traffic does not make the path latency gap higher than the error bound (in Fig. 20(a)). As a result, there is no change to traffic splitting ratios across the paths. In this case, there are little or no link updates in the theshold based scheme while the fixed interval update scheme



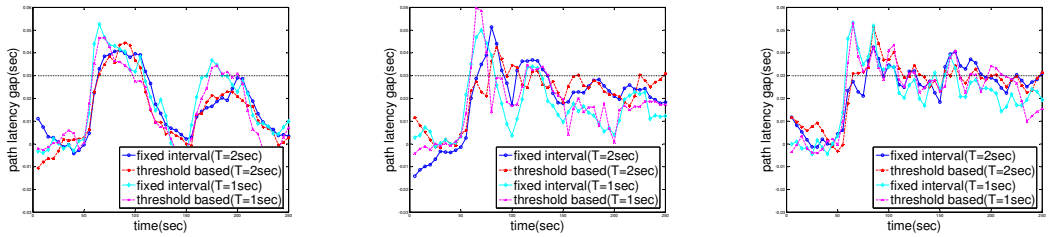
(a) Path latency gap (b) Cumulative path latency gap (c) Cumulative number of link updates

Fig. 21. The path latency gap and the number of link updates for scenario 5

continues sending the same number of link updates as seen in Fig. 20(c).

To see the adoption for dynamic change of the traffic, the following simulations are examined. The capacities of all links are set to 2Mbps. The burst traffic is on link(r_1, r_2) between 50 sec and 450 sec, and the burst traffic is on link(r_1, r_3) between 100 sec and 400 sec. Node r_1 can react these traffic changes with measuring the link state. In addition, the burst traffic is on link(r_2, r_4) between 150 sec and 350 sec, and the burst traffic is on link(r_3, r_4) between 200 sec and 300 sec. It can be tested how both schemes react to the dynamic change of traffic (We refer this scenario 5).

Even in such dynamic situation of rapid fluctuations in traffic across different links, threshold-based updates maintain the paths within the error bounds (As seen in Fig. 21(a)). Fig. 21(b) shows that the cumulative path latency gap, that measures the total cumulative difference in path quality across the simulation time, is slightly better for threshold-based updates. It is again noted that the threshold-based propagation requires far fewer updates to reach similar routing goals. The number of link updates in the threshold based update scheme is also much lower than the fixed interval update scheme.



(a) Burst traffic period=100sec (b) Burst traffic period=50sec (c) Burst traffic period=20sec

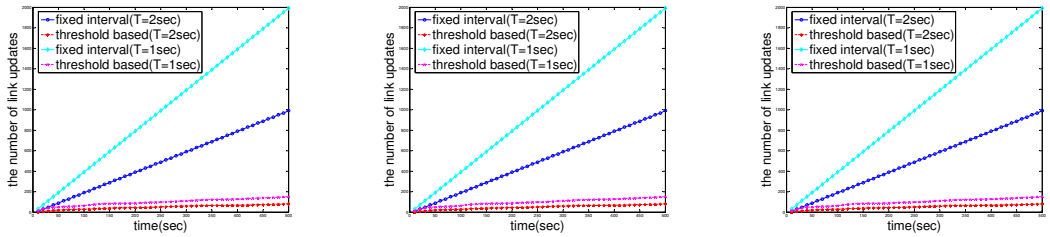
Fig. 22. The path latency gap with the different T on the periodic burst traffic

4. Impact of the update interval

In this section, the impact of the update interval is tested with the different T: T= 2 sec and T=1 sec. The more frequent updates are expected to lead to faster convergence.

For the simulation, we put a periodic burst traffic from 50 second on link(r_2, r_4) of the network (Fig. 13). The burst traffic is active in the first half of the burst traffic period, and then the burst traffic is inactive in the second half of the burst traffic period. We exame three burst traffic periods: 100, 50, and 20 seconds. If the burst traffic period is long enough to make splitting ratio near to the optimal splitting ratio, the path latency gap does not exceed the error bound in next burst traffic period. This simulation is expected to test the speed of convergence.

Both update schemes with T=1 update the increased latency information faster than those with T=2. As a result, both update schemes with T=1 achieve convergence earlier than those with T=2 (in Fig. 22(a), 22(b), and 22(c)). Even worse, both update schemes with T=2 do not converge at the end of the simulation if the burst traffic period is 20 sec. The update scheme with T=1 outperform the update scheme with T=2, but the problem is overhead of link updates. As seen in Fig. 23(a), 23(b),



(a) Burst traffic period=100sec (b) Burst traffic period=50sec (c) Burst traffic period=20sec

Fig. 23. The number of link updates with different T on the periodic burst traffic

and 23(c), the number of link updates in the update schemes with T=1 is twice of that in the update schemes with T=2. However, the threshold based update scheme with T=1 has lower overhead than that with the fixed interval update scheme with T=2.

D. Related work

Dynamic routing has received much attention. Early ARPAnet considered link delays as a cost metric and the resulting oscillations prompted the use of other metrics based on capacity.

In most networks, the link costs are now determined based on traffic engineering considerations [6, 49].

Simultaneous traffic engineering and routing table computation is considered in DEFT [34]. DEFT splits a flow's traffic across multiple paths based on an exponential function of the path delay differences, preferring smaller delay paths.

Recently, intelligent route control devices have been employed to route traffic efficiently when stub networks are multi-homed [21]. These devices measure path delays through Internet and utilize this information in making decisions on which of the available network connections will be utilized for routing traffic. Oscillations and

convergence issues are considered [50, 51]. In such systems, path delay information is obtained at the end stub network, through passive or active measurements and individual network link state is not propagated by the network elements.

Dynamic routing has been studied widely. Dynamic routing is proposed recently for wireless networks [24], and for dynamic traffic engineering in wired networks [9] and for multi-path adaptive routing [52]. These approaches utilize network delays for making routing decisions. These approaches rely on network elements propagating the dynamic link state information around the network and hence can directly benefit from this work reported here.

The problem of reducing link state updates have been studied previously in [53, 54] for QOS routing and shortest-path routing. Our threshold based update scheme combines both fixed and relative error thresholds in order to be more widely applicable and we formally prove the impact of the proposed scheme on stability of the routing paths along with bounding the expected errors.

The work [20] considered triggered updates of available bandwidth for QOS routing. The updates are triggered based on inverse of the available bandwidth at a link. It was shown that the triggered updates can be effective for QOS routing. Available link bandwidth is a dynamic link metric, as it fluctuates with call admission and departure in QOS routing. Our work is similar to this earlier work, but focuses on utilizing link delays as the dynamic link metric in Wardrop routing, and rigorously examines the effect of the triggered update rate on algorithm performance.

While the focus in [53] and [20] is on simulation study, we additionally analytically bound the errors in link delays and study the impact on the performance of the routing algorithm in reaching approximate Wardrop equilibrium. In [20], no generally applicable answer to the question of "how big a change in the link metric is significant?" is determined. In this work, we present a method to set the threshold and

thus the rate of triggered updates based on the deviation from the exact equilibria that is tolerable as well as the acceptable rate at which the algorithm converges.

Delay and convergence analysis of Wardrop routing with regular update of link state information is studied in [47, 55]. Our analysis here shows that threshold-based updates can reach convergence (within error bounds instead of exact equalization of delays) with less restrictive assumptions. Note that, in [55], the link latency function is assumed to have a bounded slope. This condition is satisfied, for example, when the growth of the latency function is polynomially bounded. Both the interval at which the regular link updates must be sent as well as the speed of convergence depend on the upper bound of the rate at which link latency can grow. Thus, the periodic update policy will have to be reconfigured for different networks with different link latency characteristics and also when networks expand by adding new links. The threshold based update policy proposed in this paper will adaptively vary the rate of updates, implicitly taking into account changes in the rate of growth of the latency function both as the regime of operation (low or high load) as well as the network topology vary. Also, since link latency also includes queueing latency, the rate of growth of link latency in many practical systems cannot be bounded by polynomial functions. Consider for example, even a simple M/M/1 queueing model to see that this is true. The convergence and indeed even the speed of convergence of the threshold based policy does not depend on the characteristics of the latency function, enabling its use in various practical systems with queueing delay at the links. Thus, the threshold based policy will adaptively vary the rate of updates and converge to the approximate Wardrop equilibrium without requiring configuration based on knowledge of the link latency growth rates and can be used even in systems with link delays that can grow at unbounded rates depending on the load on the link.

REPLEX [9] proposed Wardrop routing for dynamic traffic engineering purposes.

Our work is motivated by this and other earlier work on Wardrop routing. REPLEX utilized periodic updates of state and used the most recent observed state in making routing decisions. Our work used REPLEX for comparing the threshold-based updates to the fixed interval updates. Our results are equally applicable whether Wardrop equilibrium is used for routing or for traffic engineering purposes among the paths made available by the underlying routing algorithm.

This work here focused on reaching approximate equilibrium in Wardrop routing (a form of multi-path routing). However, the results can be generalized for other routing algorithms.

This work does not attempt to exactly equalize the delays across different paths. However, this loss in exact equilibrium is well compensated by the observed reduction in update traffic of the link state information.

CHAPTER IV

REDUCING ENERGY CONSUMPTION USING MULTIPATH ROUTING I - TRAFFIC ENGINEERING APPROACH

In this chapter, the techniques of energy saving in the networks by reducing the number of active network elements. A new topology control based on Steiner tree approach and a multipath routing based bin-packing approach are proposed. The topology control takes an approach of building an appropriately provisioned network to meet the demands of all the terminal nodes. This is in contrast with existing approaches where nodes and links are removed from the given network. Also, the effectiveness of employing multi-path routing is studied to reduce energy consumption. While multi-path routing allows more possibilities for routing the traffic demands, if longer alternate paths are employed, the power consumption can actually increase. We propose an effective multi-path routing strategy for reducing energy consumption. The topology control reduces the number of iterations for finding a suitable topology. In addition, multipath routing with the topology control reduces the number of active nodes further, and achieves more energy savings.

A. Problem formulation

To minimize energy consumption, an optimization problem considering the network topology and the traffic demand is solved. An optimal solution minimizes the number of nodes and links, while all traffic demands are delivered with maximally utilized link capacities.

$G(V, E)$ represents the network as a directed graph that consists of a set V of nodes and a set E of links, where each link $(i, j) \in E$ between two nodes $i, j \in V$ has a capacity $c(i, j)$. The total number of nodes is $N = |V|$, and the total number

of links is $L = |E|$. Let t^{sd} be traffic demand: the amount of traffic going from node $s = 1, ..., N$ to node $d = 1, ..., N$.

Let $x_{ij} \in [0, 1], i = 1, ..., N, j = 1, ..., N$ be binary variable being the value of 1 if link (i, j) is present and powered on, and the value of 0 if link (i, j) is not present or powered off. Similarly, let $y_i \in [0, 1], i = 1, ..., N$ be binary variables of router power status.

Let $f_{ij}^{sd} \in [0, t^{sd}]$ denote the amount of flow from s to d routed through link (i, j). Similarly, let f_{ij} be the total traffic flowing on the link (i, j).

Let P_L and P_N be the power consumption of link and node, respectively. Here, it is assumed the power consumption of link and node is constant, because turning on network components consumes most of power, and today's network components are not energy proportional. While focusing on a constant power budget for a powered-on link here, it is straightforward to modify this analysis to account for higher power consumption at higher link utilization.

Given the previous definitions, the problem is formulated as follows.

Minimize

$$P_N \sum_{i=1}^{N} y_i + P_L \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij} \tag{4.1}$$

Subject to:

$$\sum_{j=1}^{N} f_{ij}^{sd} - \sum_{j=1}^{N} f_{ji}^{sd} = \begin{cases} t^{sd}, & \forall s, d, i = s \\ -t^{sd}, & \forall s, d, i = d \\ 0, & \forall s, d, i \neq s, d \end{cases} \tag{4.2}$$

$$f_{ij} = \sum_{s=1}^{N} \sum_{d=1}^{N} f_{ij}^{sd} \quad \forall i, j \tag{4.3}$$

$$f_{ij} \leq c_{ij} x_{ij} x_{ji} \quad \forall i, j \tag{4.4}$$

$$\sum_{j=1}^{N} x_{ij} + \sum_{j=1}^{N} x_{ji} \leq My_i \quad \forall i \tag{4.5}$$

Constraint 4.3 is the flow conservation constraints. Eq. 4.4 constraints the total load on a link to be less than the link capacity with power state of bi-directional links. Constraint 4.5 states that a node can be turned off only if bi-directional links are powered off. The big-M method is used to force this constraint, $M = 2N$ [28].

This problem is a multi-commodity minimum cost flow problem known as NP-hard [56]. In addition, P_N and P_L vary widely depending on the devices, but generally P_N is much higher than P_L [57]. So, the assumption is $P_L << P_N$. In this case, this problem aims to switch off the largest possible number of network nodes first and then switches off as many links as possible.

B. Overview of topology control and multipath routing

The proposed scheme relies on the intuition that the energy saving achieved by powering off nodes is higher than by powering off single links, and powering off a node is more difficult than powering off a single link.

For powering off nodes, a network is constructed such that it has the connectivity between the terminal nodes with a minimum number of nodes. If the capacity of the constructed network is not enough to carry the traffic demand, nodes are added until the capacity meets the traffic demand. Different from the schemes in [28] which visits all nodes and tries to delete a node iteratively, the proposed scheme has less complexity of finding a subset of nodes with connectivity than the complexity of the feasibility test, and reduces the number of iterations of deleting nodes and feasibility testing. The detail of this topology control in section C. The required topology can be constructed either with single path routing or multi-path routing. In single path

routing, the traffic from an origin takes the shortest path to the destination and all the traffic between that od-pair takes the same path. Multi-path routing allows the traffic between an od-pair to be split up and sent along multiple paths. It allows more possibilities for consolidating the traffic onto fewer nodes and links and hence the constructed topology depends on the choice of the routing algorithm.

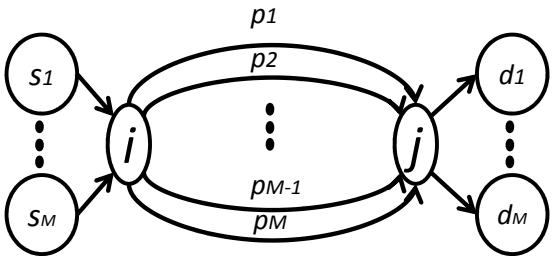


Fig. 24. Example of comparing energy saving between single path and multipath

Multipath routing has the potential for multipath routing to aid in energy savings. When the topology control checks whether the capacity of a subset of network elements is enough to carry traffic demand, multipath routing may need smaller number of network elements than single shortest path routing. The following example shows that multipath routing can power-off more nodes and links on the topology of Fig. 24. The given topology has M od-pairs ($s_1 - d_1$,$s_2 - d_2$,...,$s_M - d_M$). Each od-pair has M available paths (p_1,p_2,...,p_M), and the capacity of each path between i and j is 1 ($C_{ij}(p_k) = 1, k = 1, 2, ...M$, where $C_{sd}(p)$ is the minimum link capacity on the path p from s to d). Suppose the traffic demand of each od-pair is $x \in [0, 1]$. Any path can deliver $\lfloor \frac{1}{x} \rfloor$ od-pairs , if od-pairs use a single path, and do not allow traffic

split. So, the minimum number of path for M od-pair is $\left\lceil \frac{M}{\lfloor \frac{1}{x} \rfloor} \right\rceil$ with single path routing. However, if od-pairs use multiple paths and allow traffic split, the minimum number of paths for M od-pair is $\lceil Mx \rceil$. The number of path used for delivering traffic can be reduced as $\left\lceil \frac{M}{\lfloor \frac{1}{x} \rfloor} \right\rceil$-$\lceil Mx \rceil$ by using spliting traffic on multiple paths. In case of $x > 0.5$, a single path without traffic split uses M path, but multipath routing with split can potentially limit the number of paths to $\lceil Mx \rceil$. This motivation is to study multipath routing's impact on energy savings. We describe the detail of this topology control in section D.

C. Topology control

Algorithm 2 Topology control

Input: A graph $G = (V, E, w)$ and a terminal set $V_T \in V$ **Output:** T

```
{Switching off nodes}
T=MST-Steiner(G, V_T)
nodeArray = sortNodes(V − T,routingDensity or btwCentrality)
i ← 0
paths = compute K-shortest path with extra two hops
compute all link flow(paths)
while checkFlow(paths) = FALSE do
  i++
  T ← T∪ nodeArray(i)
  paths = compute K-shortest path with extra two hops
  compute all link flow(paths)
end while
{Switching off links}
linkArray=sortLink(E,routingDensity)
for all j ∈ linkArray do
  disableLink(linkArray(j))
  paths = compute K-shortest path with extra two hops
  compute all link flow(paths)
  if checkPath(paths)=FALSE || checkFlow(paths) = FALSE then
    enableLink(linkArray(j))
  end if
end for
```

Output: T

Topology control is described in Algorithm 2. Since turning off a node can achieve more energy saving than turning off a link, we turn off nodes first, and then turn off links.

Turning off nodes begins with finding a small number of nodes that can provide connectivity to all the terminal nodes. Finding the minimum number of nodes with connectivity is NP-hard [58]. A heuristic based on the construction of a Steiner tree is proposed. The original Steiner tree problem is to minimize the total link weight with connectivity among the terminal nodes. To change this problem to the problem of minimizing the number of nodes, we need the modified minimal spanning tree approximation of [58].

The proposed topology control realizes that traffic routing between od pairs depends on a number of factors such as the number of paths between the od pairs, their path length and how many od pairs may share a link or node etc. In order to account for these factors, a virtual topology of N_T is constructed, which connects a terminal node pair with a single link with a weight which is a function of these different factors

$$\frac{(\delta_N(s)+\delta_N(d))(max_{(i,j)\in od-pairs} dist(i,j)^{|PATH(s,d)|})}{dist(s,d)^2}, \tag{4.6}$$

where s is a source node, d is a destination node. We denote $\delta_N(i)$ as degree of node i, $|PATH(s,d)|$ as the number of equal cost paths from s to d, and $dist(s,d)$ as the shortest path length from s to d. The rationale of Eq. 4.6 is the following. The nodes on the shortest path of od-pair which has a small number of multiple equal cost paths should be powered on first, because path length of such an od-pair increases significantly after removing nodes on its shortest path. Similarly, the nodes on the shortest path whose path length is high should be powered on first in order to reduce increment of path length of entire od-pairs and to reduce the number of iterations of

the construction of spanning tree. The degree of node is used for tie break. Then a spanning tree is computed with smallest edge weights on the virtual topology, in which each edge corresponds to one shortest path on the original graph. Finally, the spanning tree is transformed back to a Steiner tree by replacing each edge with the shortest path and some straightforward postprocessing to remove any possible cycle. The pseudocode is presented in Algorithm 3.

Algorithm 3 MST-Steiner

Input: A graph $G = (V, E)$ and a terminal set $V_T \in V$
Output: A Steiner tree T

```
Construct the virtual topology G_{V_T} on the terminal set V_T
Find an MST T_{V_T} on G_L with weight of Eq. 4.6
T ← 0
for all e = (u, v) ∈ E(T_{V_T}) do
  Find a shortest path P from u to v on G
  if P contains less than two vertices in T then
    Add P to T
  else
    Let p_i and p_j be the first and the last vertices already in T
    Add subpaths from u to p_i and from p_j to v to T
  end if
end for
```

Output: T

After finding the minimum number of nodes with connectivity, traffic demand flowing through the network element is routed by the proper routing scheme. (This routing scheme is introduced in section VI.) When the traffic is assigned, traffic demand constraints (Eq. 4.3) and the link utilization constraints (Eq. 4.5) are checked. If no violation is present, then we move to switching off links. If a violation occurs, we add nodes by the following sorted order.

The node set is sorted by the following criteria: (1) routing density, which is the sum of routing density of links, $X_i = \sum_{j=1}^{N} x_{ij} RD(i,j) + \sum_{j=1}^{N} x_{ji} RD(j,i)$, where $RD(i,j)$

is the routing density of link(i, j) in chapter II. Routing density of a link is a count of how many times a link is incident on the shortest path between different od pairs. (2) betweenness centrality [59], which is the number of shortest paths of all possible od-pairs that pass through a node, $Y_i = y_i BC_i = \sum_{s \neq i \neq d} \frac{\sigma_{sd}(i)}{\sigma_{sd}}$, where σ_{sd} is total number of shortest paths from node s to d, and $\sigma_{sd}(i)$ is the number of those paths that pass through node i. Nodes with large value of criteria are checked first, i.e., V is sorted in decreasing value of routing density or betweenness centrality. We use the criteria of routing density as a default and compare with the criteria of betweenness centrality in section 2 and 3. The complexity of computing routing density and the complexity of between centrality is dominated by the complexity of the length and the number of all shortest paths. Consequently, the complexity of routing density and the complexity of between centrality is $O(|V||E| + |V|^2 log|V|)$ [60].

Adding extra nodes to the topology is repeated until traffic can be routed with no violations of constraints stated earlier. It is expected that adding nodes with higher aggregate link capacity will lead to adding larger amount of capacity to the network and thus requiring fewer nodes and hence lead to larger energy savings.

After finding the minimum number of nodes, the minimum number of links on the subset of network elements is computed. The link set is sorted by amount of flow on the link. Links with a smaller amount of flow on the link are checked first, i.e., E is sorted in increasing value of $F_{ij} = f_{ij} + f_{ji}$. Switching off the cable (i, j) affects link(i, j) and link(j, i),in both directions and it is expected that links with smaller amount of traffic can be switched off and the traffic on this link can be routed to other links in the reduced topology (without this cable). If switching off a link does not cause violate connectivity, traffic demand and link utilization constraints, the link is switched off. The algorithm progresses turning off as many links as possible and terminates when all $e \in E$ is visited.

Here, the complexity of the proposed topology control is compared to the complexity of the topology control in [28]. Both topology controls use a similar scheme for switching off links, only the complexity of switching off nodes is compared here. In the proposed topology control, the complexity of the Steiner tree graph with the minimum number of nodes is $O(|V||E|(log|V|)(log|E|))$. The complexity of sorting the removed nodes and finding K-shortest paths is $O((|V|-|V_m|)|V|(|E|+|V|^2 log|V|))$, where $|V_m|$ is the number of the active nodes on topology with the minimum number of nodes. The latter one is greater than the former. The complexity of the proposed topology control is the complexity of sorting the removed nodes and finding K-shortest paths times the complexity of the feasibility test to ensure traffic can be routed.

With single shortest path routing, the complexity of feasibility testing does not change based on the topology control algorithm. It means the complexity comparison between two topology controls is dependent on the number of iterations of feasibility test. The iteration number of the proposed topology control is $O(|V|-|V_m|)$, compared to earlier scheme of $|V|$. We can say the proposed topology control is slightly less complex than the topology control in [28].

The complexity of feasible test with multipath routing becomes as K times as the complexity of feasible test with single shortest path routing. The complexity comparison between the proposed topology control and the topology control in [28] is conditional to network topology.

D. Multipath routing

In this section, the approach to assign the traffic demands of an od-pair among the multiple paths allowed by the multi-path routing algorithm is presented. It is

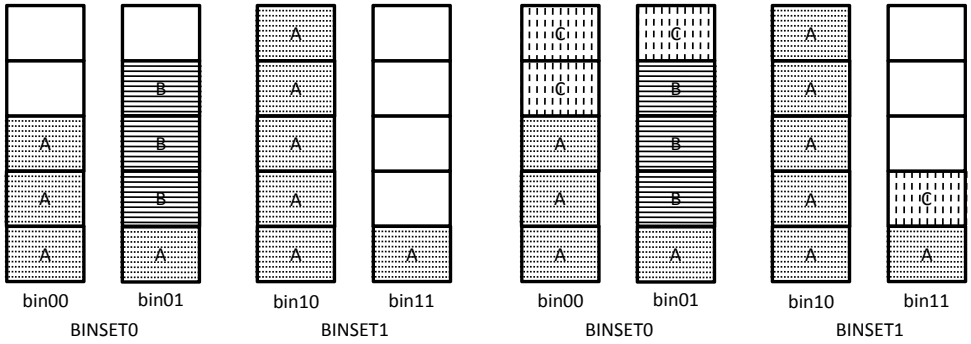
noted that the proposed approach does not constrain the choice of multi-path routing algorithms. Among the many choices available to route the traffic between an od-pair, the algorithm allocates the traffic or fractions of that traffic to different paths with a view to minimizing the number of nodes and links in the network.

This problem of assigning the traffic demand to multiple paths can be seen as a modified bin-packing problem, where available paths or the bins and the traffic demand is the object to be packed. In order to assign the traffic demand to a path, we look at the minimum residual capacity of all the links on that path and employ it as the bin capacity for that path. In single path routing, the traffic demand between an od-pair cannot be split and hence has to be assigned as a whole to one of the paths. In single shortest path routing, the traffic demand is sent along the shortest path. In multi-path routing, the traffic demand can be split and allocated to the available paths. The choice of paths and the choices of splitting the traffic demand can determine the performance. Two bin packing algorithms are proposed in order to assign the traffic demand of an od-pair to multiple paths.

Multipath routing is 'compute all link flows' in Algorithm 2. Link flows are computed by multipath routing done through bin packing in order to minimize the number of nodes and links in topology control. This section introduces two bin packing algorithms and explains how to use these bin packing algorithms for multipath routing. At any given time, the choice of available paths for an od-pair constrain the choices of bins. All available paths are considered as bins, and divided into these bins into several set of 'closed bin set' which satisfies a certain condition, i.e. path length. If the traffic demand cannot be met by the capacity of all the bins, it would be needed to add nodes/links to the topology to increase capacity. The intent of the algorithms below is to assign traffic demands to minimize the number of needed bins.

- **Closed first fit (Algorithm 4):** Closed first fit is first fit algorithm within the closed bin set (a subset of all the bins which satisfy some conditions). For each item, it attempts to place the item from the first bin to the last bin among the closed bin set that can accommodate the item. If there are no bins to accommodate the item in the closed bin set, closed first fit returns failure.

- **Closed next fit with fragmentation (Algorithm 5):** Closed next fit with fragmentation is next fit with fragmentation algorithm within the closed bin set. In each stage, there is only one open bin. The items are packed, according to their order in the list, into the open bin. When an item does not fit in the open bin, it is fragmented into two parts. The first part fills the open bin and the bin is closed. The second part is packed into a new bin which becomes the open bin. If there are no more bins to open in the closed bin set, and a fragment remains, closed next fit with fragmentation returns failure.



(a) Closed first fit (b) Closed next fit with fragmentation

Fig. 25. Example of the proposed bin pack algorithms

Algorithm 4 closedFirstFit($PATH, TD, linkUsage$)

$i \leftarrow 0$
$TD' \leftarrow TD$
while $i < N(PATH)$ & $TD' > 0$ **do**
  $minCapa = min_{(i,j) \in PATH(i)} c(i,j) - linkUsage(i,j)$
  if $minCapa \geq TD$ **then**
    for all $(j,k) \in PATH(i)$ **do**
      $linkUsage(j,k) \leftarrow linkUsage(j,k) + TD$
    end for
    $TD' \leftarrow 0$
  end if
  $i++$
end while

Output: $TD', linkUsage$

Algorithm 5 closedNextFitFrag($PATH, TD, linkUsage$)

$i \leftarrow 0$
$TD' \leftarrow TD$
while $i < N(PATH)$ & $TD' > 0$ **do**
  $minCapa = min_{(i,j) \in PATH(i)} c(i,j) - linkUsage(i,j)$
  if $minCapa \geq 0$ **then**
    for all $(j,k) \in PATH$ **do**
      $linkUsage(j,k) \leftarrow linkUsage(j,k) + minCapa$
    end for
    $TD' \leftarrow TD' - minCapa$
  end if
  $i++$
end while

Output: $TD', linkUsage$

Examples are shown in Figure 25(a) and 25(b). 'A' items are packed in $BINSET0$ and $BINSET1$. When allocating three 'B' items, we try closed first fit in $BINSET0$. $bin00$ is not enough to allocate three 'B' items, we move to $bin01$. $bin01$ has enough capacity, we allocate three 'B' items on $bin01$. Next, we try to allocate four 'C' items in $BINSET0$. $bin00$ and $bin01$ do not have enough capacity for closed first fit. However, closed next fit with fragmentation can allocate three 'C' items in $bin00$ and $bin01$. We still have a 'C' item. Then we move to $BINSET1$ and allocate it in $bin11$.

2. Multipath routing based on bin packing algorithms

In order to assign all the traffic demand on a small number of nodes and links, multipath routing is proposed for energy saving with the bin packing algorithms.

First, the order of od-pairs is sorted. Before sorting, each od-pair computes multiple paths whose path length is from the length of the shortest path to the length of the shortest path plus two extra hops. If the number of possible paths is larger than K, available paths are restricted to K-shortest paths. To enhance usage of multipath, the order of od-pairs is sorted by the number of paths. If there is no room for assigning traffic on the shortest path, the proposed scheme can assign traffic on an alternative (possibly longer) path. However, some od-pairs have small number of paths. In order to increase the possibility of finding alternate paths, od-pairs with smaller number of paths is first considered. As traffic gets assigned to available links, the number of possible paths with sufficient capacity to route a given od-pair decrease. Hence having a larger number of choices for later assigned od-pairs is expected to increase the chances of success of assigning the traffic demand to a smaller number of links and nodes.

Similarly od-pairs that have smaller paths lengths and smaller traffic demands

are expected to be easier to route. Based on this rationale, these sorting criterions are tested. The number of paths is the most critical, so od-pairs are sorted by the number of paths first. The result from the sorting order by path length first shows assigning more traffic on the same topology than the result from the sorting order by volume of traffic demand first. As a result, the second sorting criterion is path length, and the third is volume of traffic demand. In sum, the od-pairs are sorted by the number of paths first, then by the path length and then by the volume of traffic demand.

After sorting od-pairs, the traffic demand is assigned to multiple paths. To assign the traffic demand of an od-pair, the multiple paths are sorted by path length, routing density, and capacity. Routing density is defined as how many times a link on the path is used as a next-hop of all possible od-pairs and it is expected to given an indication of how critical or useful this link can be in routing various flows. If multiple paths are options for the traffic of an od-pair, paths and the amount of traffic split on the paths are chosen so that remaining network capacity of the links remains as high as possible. It is expected that maximizing the remainder capacity makes it easier to route later od-pairs. The preferred path is the path with low path length, and low routing density and retains large residual capacity for other traffic demands. Multiple paths are sorted by path length first, then by routing density, and then by residual capacity.

$PATH0(s,d)$ is the set of paths from s to d with the shortest path length. $PATH1(s,d)$ is the set of paths from s to d with an extra hop. $PATH2(s,d)$ is the set of paths from s to d with two extra hops. t^{sd} is the traffic demand from s to d. The path capacity is the minimum remaining link capacity on the path.

First, t^{sd} is assigned on $PATH0(s,d)$ by closed first fit. $t^{s,d}$ is assigned on the first path of $PATH0(s,d)$. If the capacity of the first path of $PATH0(s,d)$ is not

Algorithm 6 Multipath routing for energy saving

Input: G,odpairArray,TD,$PATH$
Output: checkFlow

odpairSortedArray=sorting(odpairArray, [$PATH$, TD])
$i \leftarrow 0$
$overCapa \leftarrow$ FALSE
for all $(i, j) \in E$ **do**
  $u(i, j) \leftarrow 0$
end for
while $i < N(odpairSortedArray)\&overCapa =$ FALSE **do**
  $s = odpairSortedArray[i].src$
  $d = odpairSortedArray[i].dst$
  (TD', u)=closedFirstFit($PATH0(s, d), t^{sd}, u$)
  if $TD' > 0$ **then**
    (TD', u)=closedNextFitFrag($PATH0(s, d), TD', u$)
  end if
  if $TD' > 0$ **then**
    (TD', u)=closedFirstFit($PATH1(s, d), TD', u$)
  end if
  if $TD' > 0$ **then**
    (TD', u)=closedNextFitFrag($PATH1(s, d), TD', u$)
  end if
  if $TD' > 0$ **then**
    (TD', u)=closedFirstFit($PATH2(s, d), TD', u$)
  end if
  if $TD' > 0$ **then**
    (TD', u)=closedNextFitFrag($PATH2(s, d), TD', u$)
  end if
  if $TD' > 0$ **then**
    $overCapa \leftarrow$ TRUE
  end if
end while

enough to assign t^{sd}, next trial is to assign t^{sd} on the next path of $PATH0(s,d)$. If closed first fit fails, t^{sd} is assigned by closed next fit with fragmentation. For detail, t^{sd} is first assigned on the first path of $PATH0(s,d)$. If the capacity of the first path of $PATH0(s,d)$ is not enough to assign t^{sd}, t^{sd} is fragmented into the capacity of the first path of $PATH0(s,d)$ and the rest of t^{sd}. Then the first fragmentation is assigned to the first path of $PATH0(s,d)$. These steps are repeated util t^{sd} is assigned to $PATH0(s,d)$ or closed next fit with fragmentation. If it fails, we move to $PATH1(s,d)$ with t^{sd}' which is the rest of t^{sd} after assigning to $PATH0(s,d)$. Closed first fit and closed next fit with fragmentation are repeated in $PATH1(s,d)$. If $PATH1(s,d)$ fails, closed first fit and closed next fit with fragmentation in $PATH2(s,d)$ are repeated. The pseudo code is presented in Algorithm 6.

E. Simulation

To evaluate the effect of the proposed techniques, various scenarios are examined on several n-by-n grid topologies (4-by4 grid topology has 16 core nodes, 16 edge nodes, and 64 links. 5-by-5 grid topology has 25 core nodes, 20 edge nodes, and 92 links. 6-by-6 grid topology has 36 core nodes, 24 edge nodes, and 124 links.) and a random topology. It begins with a simple scenario where all link capacities are identical and the traffic demands of all od-pairs are one in sections 1 and 2. The scenario where the link capacities are different and traffic demands of all od-pairs are also randomly generated with a mean value of one is considered in sections 2 and 3. These scenarios are examined with different utilizations in the network. Link capacity is calculated corresponding to the desired average network utilization. The set of od-pairs is all the possible combinations of nodes in the highest tier (i.e., edge nodes on

Table II. Power consumption of various energy models in watts

	EM1	EM2	EM3
P_N	151	133	76
P_L	11	10.5	6.5

grid topologies and aggregation nodes on a random topology). The proposed topology control with shortest path routing is denoted by ST-S. The proposed topology control with multipath routing is denoted by ST-M. In each scenario, the number of active nodes (N_R) and links (L_R) are measured, then the ratio of active nodes to total number of node (N_R/N) and the ratio of active links to total number of link (L_R/L) are calculated. Also, the energy consumption is measured on the topology (E_R) based on three different energy consumption models of nodes and links, shown in TABLE II [65].

Then energy saving is computed as

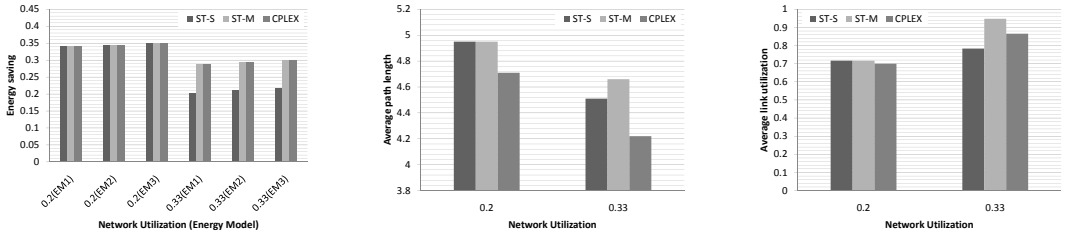
$$1 - \frac{E_R}{E} = 1 - \frac{\sum_{(i,j)\in E} x_{ij} P_L^x + \sum_{i \in V} y_i P_N^x}{L P_L^x + N P_N^x}, \tag{4.7}$$

where x is the energy model index.

The average path length and the average link utilization after removing nodes and links for saving energy is also measured. These metrics give an idea of the cost of the energy savings. Higher path lengths and higher link utilizations can lead to longer delays for delivering traffic.

1. Comparison between optimum solution and the heuristics

In this section, we evaluate the performance of ST-S and ST-M and compare the results to the optimal solution. We use CPLEX to find the optimal solution. The optimal solution is denoted as 'CPLEX'. The results of ST-S and ST-M are from the



(a) Energy saving on a 4-by-4 grid topology

(b) Average path length on a 4-by-4 grid topology

(c) Average link utilization on a 4-by-4 grid topology

Fig. 26. Comparison between optimum solution and the hueristics

simulation. A 4-by-4 grid topology is used for this comparison.

Fig. 26 and TABLE III show simulation results with traffic load on the original topology with 0.2 and 0.33 average network utilization.

The energy saving of ST-S is the same as the energy saving of CPLEX at 0.2 average network utilization, because N_R/N and L_R/L of both schemes are the same. Even though ST-S has the same energy saving as CPLEX, ST-S has higher path length and higher link utilization than CPLEX. The higher path length and link utilization are the result of different selection of active nodes and links between ST-S and CPLEX. However, the energy saving of ST-S is worse than the energy saving of CPLEX at 0.33 average network utilization, because N_R/N and L_R/L of ST-S are higher than those of CPLEX. ST-S could be expected less path length and less link utilization than CPLEX, because ST-S has more nodes and links than CPLEX. The link utilization of ST-S is less than the link utilization of CPLEX, but the path length of ST-S is higher than the path length of CPLEX. ST-S's higher path length is caused by the selection of different active nodes and links. In this simulation, ST-S has less energy saving than CPLEX (at 0.33 utilization), or ST-S has the same energy saving as CPLEX with higher path length and higher utilization (at 0.2 utilization).

At 0.2 average link utilization, ST-M has the same energy saving, path length,

Table III. N_R/N and L_R/L of ST-S, ST-M, and CPLEX on a 4-by-4 topology

Network utilization		ST-S	ST-M	CPLEX
0.2	N_R/N	0.75	0.75	0.75
	L_R/L	0.48	0.48	0.48
0.3	N_R/N	0.94	0.81	0.81
	L_R/L	0.53	0.52	0.52

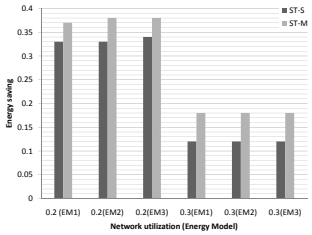
and link utilization as ST-S, because both schemes have the same topology, and ST-M uses only shortest paths with enough capacities to sustain shortest path routing. ST-M improves energy saving at 0.33 average network utilization. The energy saving of ST-M is the same as CPLEX. However, ST-M shows higher path length and higher link utilization than CPLEX. In this simulation, ST-M improves energy saving and reaches the same energy saving of CPLEX when ST-S has lower energy saving than CPLEX. However, the different selection of active nodes and links between ST-M and CPLEX makes ST-M have higher path length and higher link utilization.

The simulation results of CPLEX is shown only on a 4-by-4 topology, because CPLEX is hard to compute. For these experiments, CPLEX took 10 hours. For larger topologies, the running time of CPLEX may exceed the dynamics of timescales of traffic fluctuations and hence it is not practical.

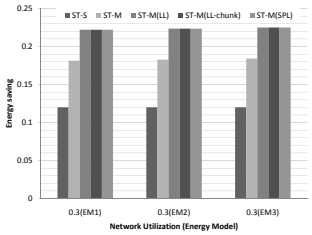
2. Simulation on grid topologies

In this section, the performance of ST-S and ST-M are evaluated in other grid topologies with more nodes and links. A 5-by-5 topology and a 6-by-6 topology are used in this section. First, link capacities and traffic demands of od-pairs are identical.

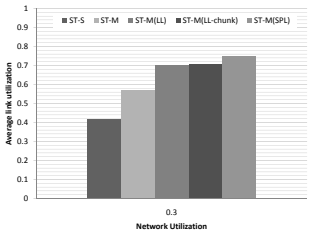
The results of both schemes are shown in Fig. 27(a) and 27(d). ST-M improves



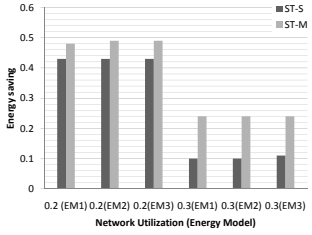
(a) Energy saving on a 5-by-5 grid topology (ST-S and ST-M)



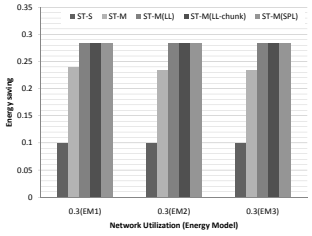
(b) Energy saving on a 5-by-5 grid topology (ST-M(LL), ST-M(LL-chunk), and ST-M(SPL))



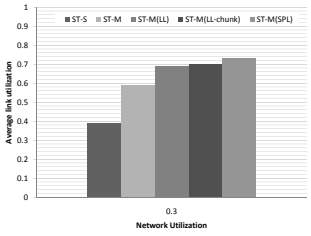
(c) Average link utilization on a 5-by-5 grid topology



(d) Energy saving on a 6-by-6 grid topology (ST-S and ST-M)



(e) Energy saving on a 6-by-6 grid topology (ST-M(LL), ST-M(LL-chunk), and ST-M(SPL))



(f) Average link utilization on a 6-by-6 grid topology

Fig. 27. Comparison between ST-S, ST-M, ST-M(LL), ST-M(LL-chunk), and ST-M(SPL) with identical traffic demand and link capacity on grid topologies

energy savings by 4% to 6% on a 5-by-5 topology and from 5% to 14% on a 6-by-6 topology, compared to ST-S. (The percentages are relative to the network without any energy controls, and relative to the energy consumed in ST-S, the improvements are higher). These results show that multipath routing contributed to improved energy savings in these scenarios. ST-M achieves higher energy savings by employing fewer nodes and links in the network than ST-S, as seen by the N_R and L_R numbers, while meeting the traffic demands of all the od-pairs.

Three more schemes to enhance multipath routing are considered. For **ST-M(LL)**, the utilization of the K-shortest paths is measured, and then the traffic to the path that shows the minimum utilization is assigned. To enhance ST-M(LL), the traffic of od-pair is divided into chunks, and then is assigned chunk traffic to the path that shows minimum utilization (This scheme is called as **ST-M(LL-chunk)**). In this simulation, the traffic demand of an od-pair is divied into 10 chunks, and then each chunk of traffic is assigned to the path that has the minimum utilization among the K-shortest paths. For **ST-M(SPL)**, the traffic based on path utilization and path length is split. Splitting ratio is set proportional to the path utilization multiplied by exponential penalty for extra hops on the alternate paths [34].

If the traffic demand cannot be assigned by the ST-M(LL) and ST-M(SPL), the remaining traffic is assigned by ST-M.

ST-M(LL), ST-M(LL-chunk), and ST-M(SPL) are evaluated with 0.3 average network utilization on a 5-by-5 grid topology and a 6-by-6 grid topology.

Fig. 27(b) and 27(e) show the results of ST-M(LL), ST-LL(LL-chunk), and ST-LL(SPL) along with ST-S and ST-M.

The energy saving of ST-M(LL) is 4-5% better than that of ST-M. These energy saving comes from the employment of smaller number of nodes and links, as seen by N_R/N shown in TABLE IV.

Table IV. N_R/N of ST-M, ST-M(LL), ST-M(LL-chunk), and ST-M(SPL) on a 5-by-5 and a 6-by-6 topologies

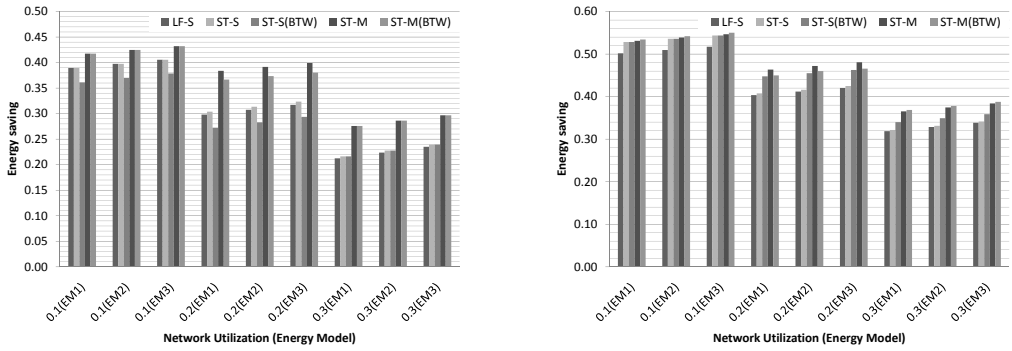
Topology	Network utilization	ST-M	ST-M(LL)	ST-M(LL-chunk)	ST-M(SPL)
5by5	0.3	0.84	0.80	0.80	0.80
6by6	0.3	0.73	0.69	0.69	0.69

There is no difference in energy savings among ST-M(LL),ST-M(LL-chunk), and ST-M(SPL). The difference among ST-M(LL), ST-M(LL-chunk), and ST-M(SPL) is observed in the average link utilization shown in Fig. 27(c) and 27(f). ST-M(SPL) uses more alternative paths than ST-M(LL) and ST-M(LL-chunk) and ST-M(LL-chunk) uses more alternative paths than ST-M(LL). As a result, the average path length of ST-M(SPL) is the highest, and that of ST-M(LL-chunk) is next, and that of ST-M(LL) is the lowest. The link utilization follows the same order with ST-M(SPL) being the highest, ST-M(LL-chunk) in the middle and ST-M(LL) being the lowest. All of them have higher utilization than ST-M.

The complexities of ST-M, ST-M(LL), and ST-M(SPL) are the same as the complexity of finding K-shortest paths. However, the complexity of ST-M(LL-chunk) is higher than the complexity of ST-M by a factor of the chunk size.

The results in this section show that multipath routing improves energy savings. It is possible to improve the energy savings further with proposed modifications to choice of paths in multipath routing. However, these differences in path choices within multipath routing may result in slightly higher path lengths.

Previously, the network links have identical capacities and each od-pair traffic demand is the same even though od-pairs are chosen randomly. From now on, different link capacities and traffic demands is considered, which represents a more realistic



(a) Energy saving on a 5-by-5 grid topology

(b) Energy saving on a 6-by-6 grid topology

Fig. 28. Comparison between ST-S and ST-M with different traffic demand and link capacity on grid topologies

scenario. Link capacity is set proportional to the routing density of the primary paths of od-pairs. Traffic demand is modeled to have pareto distribution whose mean is 1.

Fig. 28 shows that multipath routing (ST-M) can contribute to higher energy savings than single path routing (ST-S). Multipath routing provides an improvement of 0% to 11% on a 5-by-5 topology, and from 0% to 9% on a 6-by-6 topology (the improvements are again measured relative to the base energy consumption and these numbers would be higher relative to ST-S energy consumption). The higher energy saving is a direct result of employing lower number of nodes and links in the network to meet the same traffic demands.

We compare the proposed approaches ST-S and ST-M to earlier work in [28]. We denote the earlier scheme as LF-S. LF-S tries to delete nodes and links from the entire network topology, but the proposed topology control for nodes starts with a minimum number of nodes and links, and then adds nodes until the topology can support traffic demands. Our scheme has low complexity due to less number of iterations. Both LF-S and ST-S uses single shortest path routing. On both topologies, energy saving of ST-

S is the same or slightly better than LF-S. On grid topologies, the proposed topology control achieves more energy savings, when single path routing is used. Multipath routing further improves the energy savings beyond what is achieved with better topology control.

As mentioned in section D, betweenness centrality could be used for topology control. We denote topology control that sorts adding nodes on minimum topology by betweenness centraliy as ST-S(BTW) (single path routing) and ST-M(BTW) (multipath routing). On a 5-by-5 topology, energy saving of ST-S(BTW) is the same or slightly worse than ST-S. Energy saving of ST-M(BTW) is the same or slightly worse than ST-M. Using multipath routing shows more energy saving than using single path routing. On a 6-by-6 topology, energy saving of ST-S(BTW) is slightly better than ST-S. Energy saving of ST-M(BTW) is the same or slightly better than ST-M. Using multipath routing shows more energy saving than using single path routing. On the grid topology, routing density metric and betweenness centrality metrics provided similar energy savings and did not differ significantly in finding efficient topologies for energy.

Multipath routing achieves better energy savings at the cost of higher path length and link utilization than ST-S. To reduce average link utilization with smaller increment of path length, different techniques for finding alternate paths are considered. Od-pairs are sorted based on the path length differences between their shortest paths and alternate paths. Preference is given to od-pairs with higher path length differences such that they can be mostly assigned to their shortest paths. This scheme is called as ST-M(R).

Path length of the proposed schemes are shown in TABLE V. The numbers in brackets present ST-M(R) fails to assign all the traffic on the topology, and requires us to fall back to ST-M. On a 5-by-5 topology, ST-M(R) has the same average path

Table V. Average path length of the proposed schemes on a 5-by-5 topology

Network Utiliization	ST-S	ST-M	ST-M(R)
0.1	5.48	5.48	5.48
0.2	5.17	6.94	(6.94)
0.3	5.79	6.31	5.94

length as ST-M at 0.1 average network utilization. At lower network utilizations, the traffic is not routed on alternate paths many times and hence we do not observe much difference. ST-M(R) does not reduce the path length at 0.2 average network utilization, because assigning traffic demand of the newly sorted od-pairs on the topology does not have enough capacity. ST-M(R) reduces path length at 0.3 average network utilization. Similar results are observed on a 6-by-6 topology. Even though ST-M(R) reduces the path length, the reduction in path length is small as well as ST-M(R) fails to route traffic in a few cases where ST-M succeeds. ST-M is a better option.

3. Simulation on a random topology

In this section, randomly generated topologies [28] is considered. In particular,10 core nodes, 30 edge nodes, and 120 aggregation nodes are considered. Nodes are assumed to be placed on a plane. Core nodes are randomly connected to other core nodes with a probability $p = 0.5$. Each edge node is then connected to the two closest core nodes and to another randomly selected edge node. Finally, aggregation nodes are connected to the two closest edge nodes. Only aggregation nodes are traffic sources and sinks, and the traffic volume is uniformly distributed from 0.5 to 1.5.

Fig. 29 and Fig. 30 show simulation results of LF-S, ST-S, ST-S(BTW), ST-M,

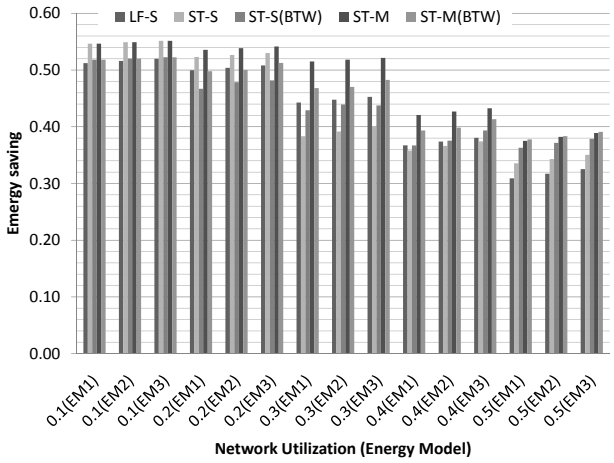
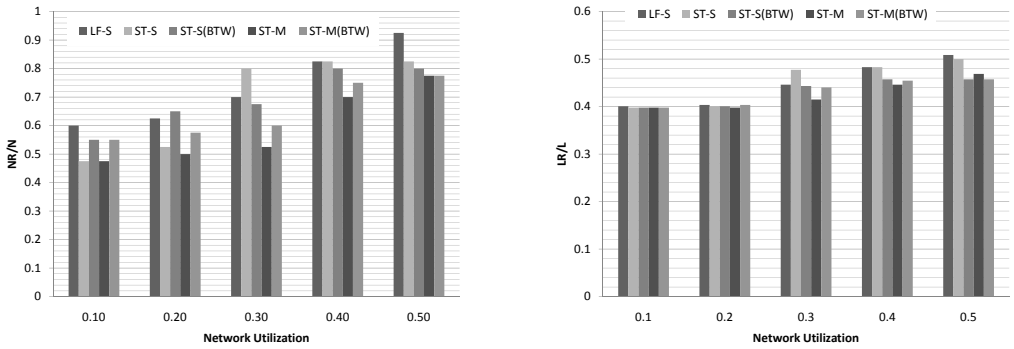


Fig. 29. Energy savings on a random topology



(a) The ratio active node to total node on a random topology

(b) The ratio active link to total link on a random topology

Fig. 30. N_R/N and L_R/L on a random topology

and ST-M(BTW) at 0.1-0.5 network utilization.

In Fig. 29, energy saving of LF-S, ST-S, ST-S(BTW), ST-M, and ST-M(BTW) are shown. At low average network utilization, energy saving gap between ST-S and ST-M is small. At 0.3 and higher average network utilization, multipath routing substantially improves energy consumption, ranging from 8 to 17% of base energy consumption. Multipath routing achieves significant energy savings by using fewer nodes in the network.

The results of N_R/N and L_R/L are shown Fig. 30. The gap of N_R/N between ST-S and ST-M is 0 - 0.35. The gap of L_R/L between ST-S and ST-M is 0 - 0.04 smaller than the gap of N_R/N between ST-S and ST-M. The dominant factor of the energy savings is the reduced number of nodes.

At low utilization (0.1 average network utilization), the number of nodes employed (N_R) by both single-path and multipath routing schemes was the same. However, N_R/N of ST-S increases from 0.58 to 0.85 when average network utilization changes from 0.2 to 0.3, but N_R/N of ST-M increases by only 0.03. This result shows that multipath routing uses the increased network capacity more effectively than single path routing.

LF-S is compared with ST-S and ST-M. On a random topology, the energy saving of ST-S is better than LF-S when the average network utilization is 0.1, 0.2, and 0.5. However, LF-S shows better energy saving than ST-S, when the average network utilization is 0.3 and 0.4. As mentioned before, N_R/N for ST-S increases when the traffic demand increases. In order to meet the traffic demand of an od-pair, a number of internal nodes may have to be added and as a result the network capacity grows in bursts. During this capacity growth bursts, the proposed topology control may require more tuning.

To improve energy saving, topology control runs again over the topology com-

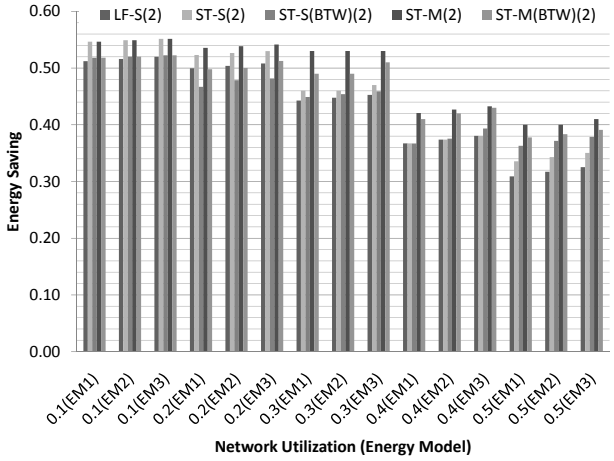
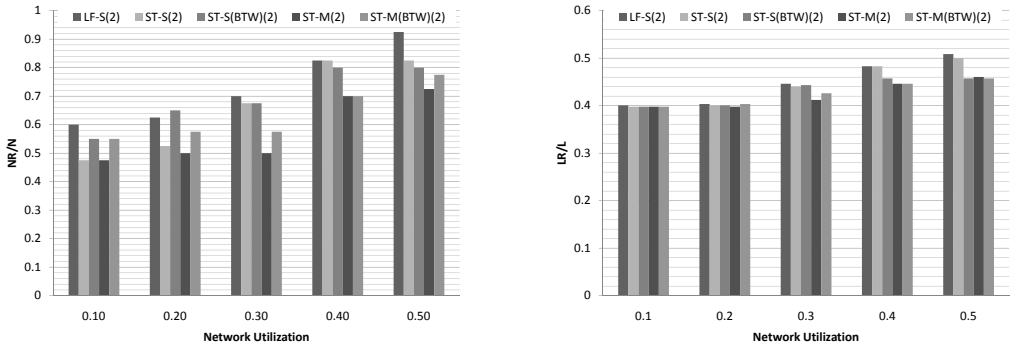


Fig. 31. Energy savings after second run of topology control on a random topology



(a) The ratio active node to total node with second run of topology control on a random topology

(b) The ratio active link to total link with second run of topology control on a random topology

Fig. 32. N_R/N and L_R/L after second run of topology control on a random topology

puted by the first run of topology control. For example, ST-S re-runs on the topology computed by ST-S. We denote the second run of various schemes as LF-S(2), ST-S(2), ST-S(BTW)(2), ST-M(2), and ST-M(BTW)(2). The order of adding a node uses routing density or betweenness centrality on the original topology. While the topology control is running iteratively, nodes and links are turned off, and the topology becomes a subset of original topology. Even if the routing density or betweeenness centrality on a subset of topology could be changed, the order of adding a node does not consider these changes. In the second run, routing density or betweenness centrality on the topology computed by the first run of topology control is more similar to those of topology that topology control currently uses, because the change of topology in the second run is expected to be less than the change of topology in the first run. Adding a highly utilized node on the currently-used topology is expected to turn off more nodes in the topology control than adding a highly utilized node on the original topology.

The simulation results of the second run are shown in Fig. 31 and Fig. 32. With the second run of topology control, ST-S(2) shows better energy saving than LF-S(2) at 0.3 and 0.4 avearge utiliztion (shown in Fig. 31). We can see NR/N of ST-S(2) is reduced at at 0.3 and 0.4 avearge utiliztion, compared to ST-S (shown in Fig. 32).

ST-M shows better energy saving than LF-S at all levels of network utilization. In some cases, ST-M(2) improve energy saving compared to ST-M. This shows that a combination of the proposed proposed topology control and multipath routing improves energy savings substantially.

The most of average network utilization, ST-M(BTW) shows worse energy saving than ST-M.

F. Related work

The problem of energy saving in the wired networks is discussed in a position paper by Gupta et al. [62].

Traffic engineering approach allows to power off unnecessary nodes/links while the remaining network capacity meets traffic demand [27], [28], [63]. Furthermore, the approach in [64] shuts down individual links in a bundled link in a similar manner. The approach in [65] also optimizes the energy consumption of data centers by powering down unneeded links and nodes in a similar manner. These approaches propose heuristic algorithms based on underlying optimization problems. All of these schemes use single shortest path routing on the topology where unnecessary nodes and links are removed. However, these schemes do not result in reducing a big fraction of the number of nodes in the network.

We consider the potential for reducing energy consumption through multi-path routing. While multi-path routing increases the routing possibilities and hence the chances of consolidating the traffic into fewer links, longer alternate paths can result in higher resource consumption. Also, traffic splitting on multiple paths could cause TCP performance loss due to the reordered packets at the destination [65]. Reordering problem can be solved by employing a flow-based routing scheme or through reorder-resistant versions of TCP [66].

We selectively power off nodes and links of the topology. As mentioned, power consumption of node is much higher than that of links. So, we minimize the number of powered-on nodes first, and then reduce the number of powered-on links. Similar strategies have been employed in [28] and [65]. Some earlier schemes focus only on link removal [27], [63], and [64].

CHAPTER V

REDUCING ENERGY CONSUMPTION USING MULTIPATH ROUTING II - HARDWARE APPROACH

In this chapter, techniques for reducing energy, which operate at finer time scales, are proposed. The proposed scheme employs multipath routing on a multi sleep state router.

A. Multi-state power mode and alternative forwarding

We consider communication links supporting three power states. The most current links support at least two power states. Here, we allow possibilities of more than two link power state, because multiple power state is are quite common and found in CPUs and memory systems [67].

- **ACTIVE**: A link is in active state. The power is highest among link states (power(ACTIVE)=1). The energy consumption is a little dependent on the link usage [57].
- **SLEEP0**: A link is in sleeping state. The power is less than ACTIVE (power(SL EEP0)=0.5). The exit latency to ACTIVE is 10's and 100's ns.
- **SLEEP1**: A link is in deeper sleeping state than SLEEP0. The power is less than SLEEP0, and much less than ACTIVE (power(SLEEP1)=0.1). The exit latency to ACTIVE is several milliseconds.

A simple way of energy saving is to put idle links to SLEEP1 directly. However, this sleeping scheme causes high entry/exit latencies due to frequent changes of power modes. The proposed sleeping scheme uses multiple power modes of links. It enters

SLEEP0 state when a link becomes idle and then enters SLEEP1 state if a link is still idle after waiting for a given time. A link in SLEEP0 or SLEEP1 changes its state to ACTIVE whenever a new packet is sent or received. Waiting time before going to SLEEP1 from SLEEP0 is important to reduce exit latency of SLEEP1 and energy saving. Network traffic is hard to predict, and the estimation scheme should be simple because hardware scheme should operate at finer granularities of time. In a situation of successive bursts on a link, the inter arrival time is quite similar to random [69]. Inter arrival time is estimated by exponential moving average. This sleeping scheme has less latency, but more energy consumption than a sleeping scheme which enters into SLEEP1 directly. The proposed scheme tries to balance the forwarding latency and energy savings by aggressively entering into SLEEP0 state and careful transition into SLEEP1 state.

Alternate path routing is employed to forward packets on paths with links in ACTIVE or SLEEP0 state and thus enabling links in SLEEP1 state to stay longer in their sleeping states, thus improving energy savings. In the power management of links with a static single-path routing (i.e., SPF), exit latency is caused whenever the primary next-hop is in either sleeping state. However, if the alternative next-hop is in ACTIVE (no latency) or SLEEP0 (10's or 100's ns latency), a packet could be forwarded to the alternative next-hop, instead of forwarding to the primary next-hop in SLEEP1 with high exit latency (several miliseconds). Alternative forwarding is to keep using ACTIVE or SLEEP0 links as much as possible, instead of changing power state of SLEEP1 link.

Alternative forwarding uses only local information because it works with the hardware scheme operating at finer granularities of time. However, alternative forwarding can cause routing loops, increased path lengths, and more latencies.

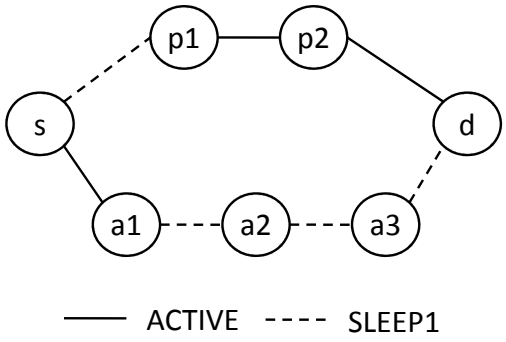


Fig. 33. Example of alternative forwarding for energy saving

In Fig. 33, s-p1 (primary next-hop of node s) is in SLEEP1 and s-a1 (alternative next-hop of node s) is in ACTIVE state. Alternative forwarding forwards a packet to a1. In this case, a1 could forward the packet to s again. To prevent a routing loop, alternative forwarding should be limited by the number of alternative forwarding attempts. This policy bounds routing loops. In this chapter, the allowable number of alternative forwarding attempts is set to 2.

Even with limited alternative forwarding attempts, alternative forwarding increases path length, which can increase delivery latencies. The higher path length can also increase path latency due to more power state changes of the remaining links. In Fig. 33, a packet is forwarded through path s-a1-a2-a3-d with one alternative forwarding. In this case, the path length is 4, which is greater than the shortest path length, 3. Even worse, path s-a1-a2-a3-d has 3 exit latencies, when the primary path has 1 exit latency. To solve this problem, alternative forwarding is allowed only on ECMP paths. It can limit the increment of path length and exit latencies.

The comparisons of energy consumption and end-to-end delay are shown in table VI. When we allow two more extra hops for alternative forwarding, alternative forwarding has 40% more energy consumption and 13% more end-to-end delay than alternative forwarding to ECMP.

Table VI. The performance of alternative forwarding corresponding to the length of alternative path

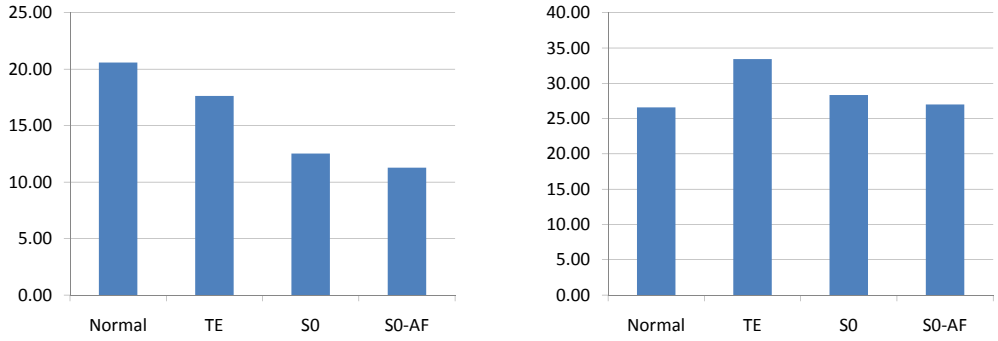
	Two extra hops	ECMP
Path length (hops)	4.10	2.71
Energy consumption	14.71	11.28
End-to-end delay (μsec)	30.45	27.00

B. Simulation

To evaluate the proposed scheme, energy consumption and end-to-end delay are measured with various combinations of sleeping schedules, routing schemes and topology controls shown in table VII. The performance is evaluated on a 4-by-4 grid topology and NSF topology [33]. Each link capacity is set to 1Gbps, and packet size is set to 1000 Bytes. Entry and exit latency between SLEEP1 to ACTIVE is set to 10 μsec

Table VII. Simulation setup for alternative forwarding for energy saving

Index	Sleep schedule	Alternative forwarding	Topology control
Normal	No	No	No
TE [28]	No	No	Yes
S0	Yes	No	No
S0-AF	Yes	Yes	No



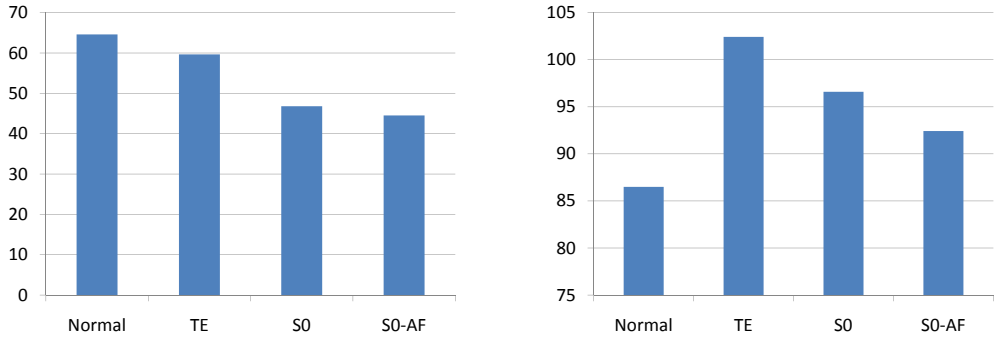
(a) Energy consumption on a 4-by-4 grid topology

(b) End-to-end delay on a 4-by-4 grid topology(μsec)

Fig. 34. Energy consumption and end-to-end delay on a 4-by-4 grid topology

and that between SLEEP0 and ACTIVE is set to 10 ns. Traffic demand is modeled as follows. Inter flow arrival time follows exponential distribution and flow size follows pareto distribution. The average flow size is 75MBytes with shape parameter 2. Inter flow arrival rate of s-d pairs is controlled in order to meet the desired average network utilization, 0.15. In each simulation, 500 flows are generated. Fig. 34(a) shows the energy consumption with different schemes. S0 consumes less energy than Normal by forcing links to sleep. Also, S0 consumes less energy than TE. The primary reason for this is the increased path lengths due to reduced number of links from topology control in TE. The average number of hops without topology control is 2.71, but the average number of hops with topology control is 3.90. Alternative forwarding reduces the energy consumption further. In Fig. 34(a), S0-AF has 10% energy saving gain compared to S0, because it does not increase path length, but decreases the number of power mode changes.

Figure. 34(b) shows the end-to-end delay. Compared to TE, S0 has lower end-to-end delay due to less path length and less link utilization. Also, S0-AF has lower end-to-end delay than S0 in Fig. 34(b). This shows that alternative forwarding



(a) Energy consumption on NSF topology

(b) End-to-end delay on NSF topology(μsec)

Fig. 35. Energy consumption and end-to-end delay on NSF topology

reduces end-to-end delay through decreased number of power state changes.

Simulation results in realistic topology are shown in Fig. 35. Similar to the results in grid topology, TE consumes less energy than Normal routing, but S0 and S0-AF provide higher energy savings than TE. S0 and S0-AF acheive this energy savings at a lower end-to-end latency penalty than TE.

In the NSF topology, there are fewer equal-cost multipaths than in a grid topology. As a result, the energy saving gains are smaller in the NSF topology than in the grid topology. However, S0 still show less energy consumption and end-to-end delay than TE. Alternative forwarding (S0-AF) reduces energy and end-to-end delay further. S0-AF reduces energy consumption by 5% and end-to-end delay by 4% compared to S0.

In summary, hardware sleeping schemes, operating at finer granularities of time, consume less energy and offer better end-to-end delivery times than the traffic engineering approach. Alternative forwarding enhances energy savings and end-to-end delay gain compared to no alternative routing.

C. Related work

Traffic engineering approach allows to power off unnecessary nodes/links while the remaining network capacity meets traffic demand, as discussed in the previous chapter. In these schemes, to save more energy, more nodes/links enter the sleeping mode, and network may operate with a lower capacity margin. Hardware schemes operate in finer-time granularities. There are two kinds of hardware schemes: sleeping and rate-control. Rate control scheme is proposed in [61] and [67], but hardware could support slow-speed mode with low power consumption. Opportunistic sleeping is also examined in [68] and [61]. In [68], they evaluate the application of opportunistic sleeping in a campus LAN environment. Techniques for more opportunistic sleeping are proposed in [61] by shaping network traffic to be more bursty. In [67], various sleeping schedules with multiple sleep states are discussed. Hardware schemes can have performance degradation due to latencies in waking up sleeping links along the primary path. Extra latency and energy consumption for changing from sleeping mode to active mode are unavoidable when only primary path is considered.

CHAPTER VI

CONCLUSION

In this dissertation, we have studied methods of improving efficiency and effectiveness of multipath routing in computer networks.

First, we investigated the potential for providing disjoint paths utilizing the same infrastructure that might be provided for proactive failure recovery. We proposed D-MRC and D-NotVia to provide backup paths with small path stretch and disjointness close to 1. Our work showed that it is possible to provide nearly disjoint backup paths utilizing the fast failure recovery mechanisms MRC and NotVia. We also showed that the disjoint backup paths could be used for multipath path routing to enhance load balancing and QoS. D-NotVia is slightly better than D-MRC in terms of stretch and disjointness. As a result, D-NotVia showed better performance in QoS and load balancing than D-MRC. However, overhead of D-NotVia is higher than D-MRC.

Next, we considered the problem of reducing the cost of updating link state for dynamic routing algorithms. We proposed threshold based link state updates that could limit the errors on paths in network to known bounds to enable reaching approximate equilibria of dynamic routing. In particular, the threshold based scheme was shown to enable reaching approximate Wardrop equilibria with quantifiable bounds in the differences of different path latencies. The simulation results showed that threshold-based updates could reduce the cost of link updates significantly compared to a model of fixed-interval updates. The number of link updates could be reduced by up to 50% to 90%. These results are encouraging and may point to making dynamic routing more viable.

Last, we studied energy savings through multipath routing in networks. The first approach employed topology control and multipath routing. We proposed a

topology control algorithm that built the necessary network connecting all the traffic sources and sinks based on a Steiner tree approach. We proposed a multipath routing algorithm based on bin packing to meet the traffic demands with minimal network resources. We simulated the proposed algorithms in different networks with different load and capacity constraints. We showed that the proposed topology control resulted in better energy savings in more scenarios than a previous topology control algorithm with single path routing. Also, our topology control had slightly less compleixy than the existing scheme when single shortest path routing was used. When combined with multipath routing, our approach resulted in significantly better energy savings, by up to 17% of the base energy consumption, than previous approaches.

The second approach employed alternative forwarding to allow hardware sleeping mechanisms to keep links in sleeping mode longer. This proposed technique worked in concert with hardware schemes for putting idle links into low-power modes. The simulation results showed that while, in general, alternative forwarding might not be beneficial due to increased path lengths and link utilization, alternative forwarding to equal cost paths could reduce energy consumption and end-to-end delay. The alternative forwarding showed 5-10% energy savings and 4-10% end-to-end delay gain compared to no alternative routing. The proposed technique combined hardware layer schemes with Layer 3 information to improve both energy consumption and performance.

REFERENCES

[1] J. He and J. Rexfored, "Towards Internet-wide multipath routing," *IEEE Network*, vol. 22, pp. 16–21, March 2008.

[2] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyyam, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. the Internet Measurement Workshop*, Marseille, France, Nov 2002.

[3] A. Kvalbein, F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Multiple routing configurations for fast IP network recovery," *IEEE/ACM Trans. Networking*, vol. 17, pp. 473-486, April 2009.

[4] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using notvia addresses," *Internet draft,* http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-00.txt, December 2006.

[5] S. Bryant and M. Shand, "A framework for loop-free convergence," *Internet draft,* http://tools.ietf.org/html/draft-bryant-shand-lf-conv-frmwk-03.txt, October 2006.

[6] B. Fortz, J. Rexford, and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," *Proc. INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[7] L. Buriol, M. Resende, C. Ribeiro, and M. Thorup, "A memetic algorithm for OSPF routing," in *Proc. the 6th INFORMS Telecomunication conference*, Boca Raton, FL, 2002

[8] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *IEEE/ACM Trans. Network*, vol. 13, pp. 234–247, April 2005.

[9] S. Fischer, N. Kammenhuber, and A. Feldmann, "REPLEX: dynamic traffic engineering based on wardrop routing policies," *Proc. Conference on emerging Networking EXperiments and Technologies (CoNext)*, Lisboa, Portugal, December 2006.

[10] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001.

[11] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: responsive yet stable traffic engineering," *Proc. ACM SIGCOMM*, Philadelphia, PA, PAugust 2005.

[12] C.Zhanga, J. Kurose, D. Towsley, Z. Ge, and Y. Liu, "Optimal routing with multiple traffic matrices: tradeoff between average and worst case performance," *Proc. IEEE International Conference on Network Protocols (ICNP)*, Boston, MA, November 2005.

[13] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation, " *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[14] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *Proc. ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[15] V. Raghunathan and P. R. Kumar, "Wardrop routing in wireless networks," *IEEE Trans. Mobile Computing*, vol. 8, pp. 636–652, November 2009.

[16] E. Crawley, R. NAir, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the internet," *Internet draft,* http://tools.ietf.org/html/RFC

2386, August 1998.

[17] I. Matta and U. Shankar, "Type-of-service routing in datagram delivery systems," *IEEE Jour. on Selec. Areas in Commun.*, vol. 13, pp. 1411–1425, August 2002.

[18] Q. Ma and P. Steenkiste, "Supporting dynamic inter-class resource sharing: A multi-class QoS routing algorithm," *Proc. IEEE INFOCOM*, New York, NY, March 1999.

[19] S. Kweon and K. Shin, "A new distributed QOS routing algorithm based on Fano' s method," *Computer Networks Journal*, vol. 48, pp. 155-174, December 2004.

[20] A. Shaikh, J. Rexford, and K. G. Shin, "Evaluating the impact of stale link state on quality-of-service routing," *IEEE/ACM Trans. Networking*, vol. 9, pp. 162–176, April 2001.

[21] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multihoming," *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[22] A. Lambert, M.-O. Buob, and S. Uhlig, "Improving internet-wide routing protocols convergence with MRPC timers," *Proc. CoNEXT*, Rome, Italy, December 2009

[23] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong, "Route oscillations in I-BGP with route reflection, " *Proc. ACM SIGCOMM,* , Pittsburgh, PA, August 2002

[24] V. Borkar and P. Kumar, "Dynamic Cesaro-Wardrop equilibration in networks," *IEEE Tran. Automatic Control*, vol. 48, pp. 382–396, March 2003.

[25] K.W. Roth, F. Goldstein, and J. Kleinman "Office and telecommunications equipment in commercial buildings - volume I: Energy consumption baseline," *Technical Report 72895-00, Authur D. Little, Inc.*, 2002.

[26] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, pp. 6–16, November 2003.

[27] M. Gupta and S. Singh, "Dynamic ethernet link shutdown for power conservation on ethernet links," *Proc. IEEE Conference on Communication (ICC)*, Glasgow, Scotland, June 2007.

[28] L. Chiaraviglio, M. Mellia, and F. Neri "Reducing power consumption in backbone networks," *Proc. ICC*, June 2009.

[29] J. W. Suurballe, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325 - 336, 1984.

[30] M. Medard, S. Finn, R. Barry, and R. Gallagher, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. Networking*, vol. 7, pp. 641–652, October 1999.

[31] M. Motiwala, N. Feamster, and S. Vempala, "Path splicing: reliable connectivity with rapid recovery," *Proc. ACM SIGCOMM*, SEATTLE, WA, August 2008.

[32] P. Key,L. Massouli, and Don Towsley, "Combining multipath routing and congestion control for robustness," *Proc. CISS*, Princeton, NJ, March 2006.

[33] Rocketfuel topology mapping. *WWW http://www.cs.washington.edu*.

[34] D. Xu , M. Chiang , and J. Rexford, "DEFT: Distributed exponentially-weighted flow splitting," *Proc. IEEE INFOCOM*, Anchorage , AK, May 2007.

[35] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," *Proc. IEEE ICNP*, Atlanta, GA, October 1997.

[36] P. Psenak, S. Mirtorabi, A. Roy, L.Nguyen, and P. Pillay-Esnault, "Mt-ospf: Multi topology MT routing in ospf," *Internet draft,* http://tools.ietf.org/html/draft-ietf-ospf-mt-04.txt, April 2005.

[37] K. W. Kwong, R. Guerin, A. Shaikh, and S. Tao, "Improving service differentiation in IP networks through dual topology routing," *Proc. ACM CoNEXT*, New York, NY, December 2007

[38] A. Kvalbein, T. Cicic, and S. Gjessing, "Post-failure routing performance with multiple routing configurations," *Proc. IEEE INFOCOM*, Anchorage, AK, May 2007.

[39] G. Apostolopolous, "Using multiple topologies for IP-only protection against network failures: A routing performance perspective," *Tech. Report 377, ICS-FORTH*, April 2006.

[40] T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartman, R. Martin, and M. Menth, "Relaxed multiple routing configurations for IP fast reroute," *IEEE Network Operation and Management Symposium*, Salvador, Brazil, April 2008.

[41] Y. Lee and A.L.N Reddy, "Disjoint multi-path routing and failure recovery," *Tech. Report TAMU-ECE-2009-06, Texas A& M University*, June 2009.

[42] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz, "Disjoint multipath routing using colored trees," *Computer Networks*, vol. 51, pp.2163 – 2180, June 2007.

[43] G. Jayavelu, S. Ramasubramanian, and O. Younis, "Maintaining colored trees for disjoint multipath routing under node failures," in *IEEE/ACM Trans. Networking*, vol. 17, pp.346–359, Feburuary 2009.

[44] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen, "Fast recovery from dual link failures in IP networks," *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[45] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," *Proc. ACM CoNEXT*, Toulouse, France, October 2005.

[46] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-Service routing using maximally disjoint paths," *Proc. International Workshop of Quality of Service*, Charleston, SC, June 1999.

[47] S. Fischer, H. Racke, and B. Vocking, "Fast convergence to Wardrop equilibria by adaptive sampling methods," *Proc. 38th Ann. ACM. Symp. on Theory of Comput. (STOC)*, Seattle, WA, May 2006.

[48] C. Vollmert, "A Web workload generator for the SSFNet network simulator," *Bachelors thesis, Technische Universitat Munchen*, 2004.

[49] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *IEEE/ACM Trans. Networking*, vol. 13, pp. 234-247, April 2002.

[50] R. Gao, C. Dovrolis, and E. Zegura, "Avoiding oscillations due to Intelligent Route Control Systems," *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.

[51] Y. Liu and A. L. Narasimha Reddy, "Multihoming route control among a Group of Multihomed Stub Networks," *Journal on Computer Communication*, vol. 30, pp. 3335-3345, November, 2007.

[52] A. Kvalbein, C. Dovrolis, and C. Muthu, "Multi-path load-adaptive routing: Putting on the emphasis on robustness and simplicity," *Proc. of IEEE ICNP*, Princeton, NJ, October 2009.

[53] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Quality of service based routing: a performance perspective," *Proc. ACM SIGCOMM*, Vancouver, BC, August 1998.

[54] K. Levchenko, G. M. Voelker, R. Paturi, and S. Savage, "XL: An efficient network routing algorithm," *Proc. ACM SIGCOMM*, SEATTLE, WA, August 2008.

[55] S. Fischer and B. Vocking, "Adaptive routing with stale information," *Proc. 24th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC)*, Las Vegas , NV, July 2005.

[56] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problem," *16th Annual Symposium on Foundations of computer science,* page 184-193, October 1975

[57] J. Chabarek, J. Sommers, and P. Barford, "Power awareness in network design and routing," in *Proc. IEEE INFOCOM*, Phoenix, AZ, April 2008.

[58] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," in *Acta Inform.*, vol. 12, pp. 141–151, 1981

[59] M. E. J. Mark, "Networks: an introduction," *Oxford University Press*, 2010

[60] U. Brandes, "A faster algorithms for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, pp. 163–177, 2001

[61] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-Adaptation," *Proc. the 5th USENIX NSDI*, San Francisco, CA, August 2008.

[62] M. Gupta and S. Singh, "Greening of the Internet," *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[63] N. Vasic and D. Kostic, "Energy-aware traffic engineering," *EPFL Technical report*, 2008

[64] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links," *Green Networking Workshop in SIGCOMM*, New Delhi, India, August 2010.

[65] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. MacKeown, "ElasticTrees: saving energy in data center networks," *Proc. USENIX NSDI*, San Francisco, CA, September 2010.

[66] N. M. Piratla and A.P. Jayasumana, "Reordering of packets due to multipath forwarding - an analysis," *Proc. ICC*, Instanbul, Turkey, June 2006

[67] K. Kant, "Multi-state power management of communication links," *Proc. COMSNETS*, Bangalore, India, January 2011.

[68] M. Gupta and S. Singh, "A Feasible Study for Power mangement in LAN Switches," in *Proc. ICNP*, October 2004

[69] P. J. Brockwell and R.A. Davis, "Introduction to time series and forecasting," in *Springer-Verlag*, 1996

VITA

Yong Oh Lee received the B.S. and M.S. degrees in electrical engineering from Yonsei Univeristy, Seoul, Korea in 2005 and 2007, respectively. He received Ph.D. degree in computer engineering form Texas A&M University in 2012. His research interests are in routing protocols in wireless and wired networks as well as analysis and modeling of network traffic. You can contact to me by following mailing address or e-mail address.

- Mailing address: 332G Wisenbaker, Texas A&M University, College Station, TX 77840

- E-mail: yongoh.lee@tamu.edu