A Connectionless Approach to Intra- and Inter-Domain Traffic Engineering

Hema T. Kaur, Shivkumar Kalyanaraman
ECSE Department, Rensselaer Polytechnic Institute,
Troy, NY-12180
{hema,shivkuma}@networks.ecse.rpi.edu

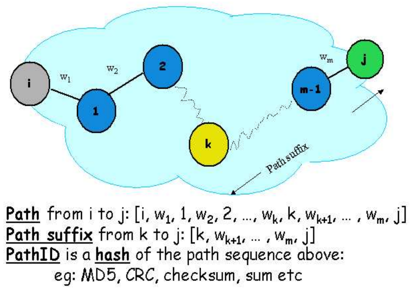
Traffic Engineering (TE) deals with the task of mapping traffic flows to the routes in an existing physical topology to improve the performance of operational IP networks. A desirable traffic engineering solution must provide network operators a precise control over the traffic flows within their routing domains. This enables the network operators to provide new services by appropriately managing the traffic.

Multi Protocol Label Switching (MPLS) allows explicit setup of one or more label switched paths (LSPs) between any source and destination and the traffic can be arbitrarily mapped to the available LSPs. However, MPLS uses a *connection-oriented* or *signaled* approach. This requires that all routers along the LSP be upgraded to support MPLS. In MPLS, the problem of setting-up LSPs is de-coupled from the problem of optimally splitting traffic among the available LSPs. In contrast, current work in the area *connectionless* intra-domain TE is to use a parametric approach. Essentially, the link weight parameter of the routing algorithm is changed to find "good" routes under quasi-stationary traffic assumption. The idea of optimizing OSPF link weights for the prevailing traffic conditions was proposed in [2], [4]. In this case, the paths on which the traffic is routed depend on traffic demands itself. This approach will hence lead to a route change for any desired change in traffic mapping or change in the demand matrix. A source-controlled traffic mapping on finer time-scales is not possible with the current connectionless TE approach, and leads to control traffic overhead (LSA re-advertisement) for every change in link weight. The routing algorithm (OSPF or IS-IS) only provides a single shortest path or multiple paths in case of Equal Cost Multi Path (ECMP) between any pair of nodes. The second approach is the multi-path approach which requires full upgrades and potential signaling. The prior work in the area of multi-path routing [3], [5], [1] have assumed support from all the routers.

We propose a connectionless approach to achieve explicit source-routing along multiple-paths without using signaling. Explicit source-routing can be achieved using the proposed approach in a partially upgraded network, where only a subset of routers support the multi-path forwarding. The key idea is to capture an intra-domain path, inter-domain AS-path or an exit route from an AS as a 32-bit hash in the packet header. We demonstrate how the proposed approach can be used to achieve TE in both intra- and inter-domain context with partial upgrades. Essentially, our approach trades off computational complexity to avoid signaling. We describe ways to manage the space (due to multiple paths) and computation complexity at the upgraded routers. Note that, our scheme, like MPLS, de-couples the problem of path computation from the problem of traffic splitting, i.e. the source may arbitrarily split traffic among the available paths without re-computing the paths. In this paper, we propose the connectionless building-blocks and do not address the problem of optimal (or near-optimal) traffic splitting or how these building blocks can be used to provide improved quality of service.

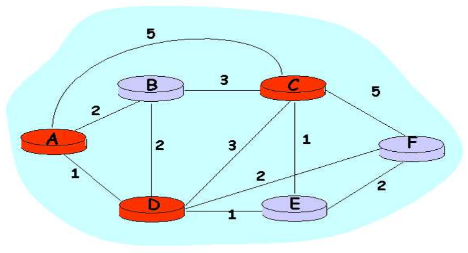
1 Proposed Connectionless Approach

Consider a network modelled as a graph $G = (V, E)$ where V is the set of vertices or nodes and E is the set of edges or links in the network. Each link $(i, j) \in E$ has a weight or cost associated with it, denoted by $w_{i,j}$. Consider a path from node i to node j shown in Figure 1(a), which passes through nodes $i, 1, 2, ..., m-1, j$ and links of weights $w_{i,1}, w_{1,2}, ..., w_{m-1,j}$. This path can be represented as a sequence $[i, w_{i,1}, 1, w_{1,2}, 2, ..., w_{m-1,j}, j]$. The path sequence can be represented by a *hash* of its elements. If this hash leads to a unique hash value for a unique input sequence with a high probability, we can use this hash to represent a path concisely. A path identifier, in short PathID, is defined as a hash of the sequence of node and/or link identifiers. In the case of intra-domain routing (for example, OSPF or IS-IS), the node IDs (i.e. router IDs) and the link weights are known at all routers (i.e. they are *globally known constants*). In the case of explicit AS-path choice in



Path from i to j: [i, w_1, 1, w_2, 2, ..., w_k, k, w_{k+1}, ... , w_m, j]
Path suffix from k to j: [k, w_{k+1}, ... , w_m, j]
PathID is a **hash** of the path sequence above:
eg: MD5, CRC, checksum, sum etc

(a)



(b)

Figure 1: Figure showing (a) Path, PathSuffix and PathID Concepts (b) Multi-Path Forwarding with Partial Upgrades

BGP, the node IDs can be the AS numbers (ASNs) which are well known for each AS-path that is available. In the case of explicit exit AS-border router (Exit-ASBR), the hash is simply the exit-ASBR IP address. Hashing a sequence of *globally known* quantities allows us to avoid signaling because each upgraded router on the path can unambiguously interpret the hash. Since the pathID is obtained from the global IDs through the hashing procedure, unlike dynamically assigned labels that are local identifiers, *signaling* is not necessary for path selection. The pathID is included in a fixed-size *routing header* in the packet.

The choice of the hash function is dictated by the need to minimize the collision probability which directly affects the uniqueness and utility of the hash. A simple hash of the path sequence may be obtained by using the sum or XOR function. The advantage of using a simple hash is that pathID computation is very simple and fast. On the other hand, it may lead to non-unique pathIDs for different paths with a high probability. Any easily computable hash-function with low collision probability can be used to find the PathID. We propose to use a 128-bit MD5 hash of the nodeIDs along the path, followed by a 32-bit CRC of the 128 bit MD5 hash to result in a 32-bit hash field. We use the notation (MD5 + CRC-32) hash to represent the above two-step hashing process. The PathID, along with the destination address (j) is used to forward a packet at the intermediate routers. If the sequence of node IDs along the path is unique (and assuming for simplicity that adjacent nodes do not have multiple links), then by the properties of the MD5 and CRC-32 hash functions, the tuple = [j, PathID] is very highly likely to be unique.

Figure 1(b) illustrates the forwarding procedure in a partially upgraded network. Nodes A, C and D in the figure are multi-path capable (MPC). Lets say that node A is the originating node for a packet destined to node F. The shortest path from intermediate node B to node F is B-D-F and path A-B-C-F is not available for forwarding because node B is a non-upgraded node. However, paths such as A-B-D-C-F, A-D-E-F, A-D-C-E-F are available. If the path A-B-D-E-F is chosen, then the PathID of an incoming packet will be Hash(A-B-D-E-F). A sets the PathID field to Hash(D-E-F), i.e. the hash of the path suffix from the next MPC router to destination. Node B forwards the packet on its shortest-path (i.e. to D). Node D sets the PathID to zero, because there is no MPC router on the path to F.

We now discuss the path computation and forwarding extensions for both the intra- and inter-domain case.

2 Application to Intra-domain TE

OSPF and IS-IS are the most common intra-domain routing algorithms in the Internet today. Since, both OSPF and IS-IS are link-state based algorithms, each router has a complete map of the network. Therefore, a router can *locally* compute the paths that are available to each destination. Even in the case of a *partially upgraded* network, the available paths can be computed locally at a router using the knowledge of upgraded routers. Observe that the fundamental tradeoff in this approach is local route computation and space complexity incurred at upgraded routers to avoid signaling. In link-state routing, each router has a complete map of the network in the form of link-state database. Using this link-state database and knowledge of upgraded routers, every router can locally compute the paths that are available to each destination. The knowledge of upgraded routers can be obtained by the upgraded multi-path capable (MPC) routers setting a bit (referred to as the MPC-bit) in their link state advertisements (LSAs). Non-upgraded routers merely ignore this bit, whereas upgraded routers use it to infer the network location of other upgraded routers. Using this model a combination of Dijkstra and Depth

First Search (DFS) can be used to compute *all* paths to a destination under the constraint that a known subset of nodes in the network have been upgraded to support multi-path routing. However, this can be computationally intensive as more routers are upgraded and lead to a large number of paths.

We show that an upgraded router may only compute a *subset* of available paths in order to manage the computation and space complexity. A subset of available paths can be computed using a multi-path computation algorithm such as k-shortest paths, all k-hop paths, DFS with constrained depth, k-disjoint paths, etc.. However, all upgraded routers must have the knowledge of the path computation algorithm and its parameters. We propose a new idea that given the knowledge of various upgraded routers and their path computation algorithms, an upgraded router can *locally* compute and validate the existing multi-paths. We propose a two phase validation algorithm for this purpose. The main idea behind the validation algorithm is that a path is "valid" or forwarding along a path exists if the path suffix is valid. In the first phase a router computes all the available paths. In the second phase, the paths are validated in increasing order of the number of hops. We know that all one-hop paths are always valid as it represents a direct link, this is first iteration. A two-hop path is valid is the corresponding one-hop path suffix is valid. Using induction, it can be shown that at the end of k^{th} iteration, all k-hop paths are valid. A validation algorithm is not needed if the subset of routers which are upgraded use DFS with partial upgrades to compute all available paths or all routers in the network are upgraded and keep exactly k-shortest paths with same value of k. However, the routers must keep the shortest path as the default path. Incoming packets with erroneous pathID are forwarded on the shortest paths and the pathID field set to zero.

The forwarding table entries of the existing OSPF/IS-IS routers are of the form **[destination prefix, outgoing interface]**, and a longest-prefix-match IP lookup procedure is used to match the destination IP address in the packet. At upgraded routers, we propose to include an "incoming pathID" and an "outgoing PathID" fields in the routing table entry to enable explicit forwarding along multiple paths. Hence, the upgraded routers will have the forwarding table entries of the form **[destination prefix, incoming pathID, outgoing interface, outgoing pathID]**. The "incoming pathID" field represents the hash of the explicit path starting from the current router to the destination prefix. The "outgoing pathID" field is the hash of the path from the *next upgraded router* on the explicit path specified by "incoming pathID" to the destination. Incoming packets at a router either already have a pathID specification in their routing header or just the destination IP address. An upgraded router first matches the destination IP address using the longest prefix match followed by an *exact match* of the pathID for that destination. If matched, the incoming pathID in the packet is replaced by the outgoing pathID, and the packet is sent to the outgoing interface.

The OSPF Link State Advertisements (LSAs) can be extended with one bit to indicate whether the router is multi-path capable (MPC). Since, this approach allows different upgraded routers to compute paths differently, we need some bits to indicate the choice of route computation algorithm along with its parameters e.g. the value of k in k-shortest paths algorithm. Note that this assumes that a router has same value of k for all destinations.

3 Application to Inter-domain TE

Inter-domain TE focuses on improving the performance between ASes. The goal is to balance load among peers. Due to policies at different ASes, an AS has very little visibility or control on policies for other ASes. Hence, it is much easier to achieve outbound load-balancing as compared to inbound load-balancing.

For inter-domain TE using BGP, we consider two cases: *explicit exit* forwarding and *explicit AS-PATH* forwarding. In the explicit exit case, exit-ASBRs for selected destination prefixes. This capability can be achieved completely within a single AS using partial router upgrades, without any co-operation from other ASes. This explicit exit capability would allow ISPs to do fine-grained outbound load-balancing for traffic generated within their own ASes. Our approach can be also used to allow explicit AS-path choice by extending the BGP interactions between ASes. In the subsequent sub-sections we discuss how our approach can be applied to these cases. The connectionless approach to inter-domain TE is an ongoing work and we have not yet analyzed the overhead due to re-advertisements and other deployment issues.

3.1 Explicit AS-Path Forwarding in BGP

A hash of the explicit AS-PATH is used as the external-PathID or *e-PathID* in the packet header to route the packet along this path. We propose that an upgraded BGP router advertises multiple AS paths to a destination and forwards packets based on both longest destination prefix and exact e-PathID match. Moreover, since BGP-4 is a path-vector protocol, the entire path is available to a router and no path computation is necessary. Extra control traffic is needed to convey the existence of multiple AS-paths between neighboring ASes. Given the scalability and instability issues with adding control traffic, ISPs may choose to only advertise a small set of multiple AS-paths only for a small subset of destination prefixes. This advertisement will be fruitless unless the neighbor AS is upgraded to take advantage of the multi-AS path feature. Moreover, if neighbor ASes do not re-advertise at least a *subset* of the multi-AS-paths available from an AS, remote ASes will not be able to take advantage of such multi-AS-paths. By this, we mean that eBGP routers of the neighbor AS must store a subset of the multiple AS-paths to a prefix in their Routing Information Bases (RIBs), and re-advertise them, even though they do not support multi-path forwarding entries in their Forwarding Information Bases (FIBs).

In a multi-AS-path capable AS, at least the entry and exit ASBRs need to be upgraded and synchronized on the multiple AS-paths available through the AS before such re-advertisements are made to other ASes. The incoming hash is swapped with the outgoing AS-path-suffix hash only at the exit AS border router. The exit AS border router for that AS-path is the BGP router which learns the AS-path from an *external peer*, i.e., its origin attribute is EBGP for that AS-path. Observe that other IBGP routers within this AS need not be aware (or upgraded) of the multiple AS-paths to chosen destination prefixes. We believe that due to these issues above, the explicit AS-path selection model is significantly more complex and requires more co-ordination between ASes compared to the explicit-exit-ASBR model that can be implemented within a single AS.

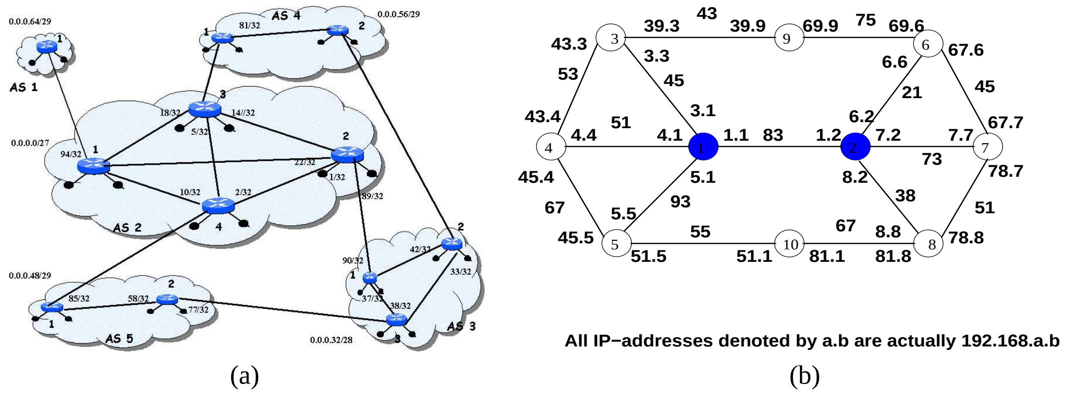


Figure 2: Figure showing (a) Inter-Domain TE Topology (b) Experimental Intra-Domain Topology on Utah Emulab using Linux Zebra/Click Platforms

Consider the AS-graph topology in Figure 2(a), and assume that we would like to send traffic from AS1 to AS5, i.e. to the IP prefix 0.0.0.48/29 along AS-path AS1-AS2-AS5, represented as (1 2 5). The AS-paths available are AS1-AS2-AS5, AS1-AS2-AS4-AS3-AS5, AS1-AS2-AS3-AS5. The explicit path (1 2 5) is chosen at router 1; the suffix AS-path is (2 5) whose hash is placed in the e-PathID field in the outgoing IP packet. The next-hop is an entry router in AS2, and the packet is forwarded to the exit router from AS2 to reach AS3. Observe that since this route (at Router 1 in AS 2) is learned from IBGP, the e-PathID is left unchanged. The e-PathID will be swapped only at the exit ASBR (i.e. Router 4 in AS2). At this exit router, the exact match of the prefix and e-PathID results in a next hop in AS5, and outgoing e-PathID will be set to 0 as AS5 is the destination AS.

3.2 Explicit Exit Forwarding in BGP

A simplifying assumption made in previous section is that the entry ASBR is directly connected to the exit ASBR. If not, the packet needs to be sent explicitly to the exit ASBR. We propose a 32-bit "address stack" field in the routing header to explicitly route a packet from either an EBGP router or an IBGP router to an exit ASBR. . The EBGP or IBGP router that decides an explicit exit for a destination prefix will simply "push" the destination IP address into the address stack field and replace it with the exit ASBR's IP address. The IP checksum is also updated appropriately. This is equivalent to using the hash of the exit ASBR in the packet. Any other upgraded IBGP router on the path will observe that the destination address has already been stacked. Non-upgraded IGP or IBGP routers will merely see the packet as destined to the exit ASBR and forward the packet normally. When the packet reaches the Exit ASBR (assumed to be upgraded), it will observe the destination address on the stack, and simply pop it out back into the IP destination address field (and adjust the IP checksum), before performing the e-PathID processing as described in the previous section. This address stacking procedure operates in the fast processing path at all routers (both upgraded and non-upgraded). Moreover, it allows flexibility for only a subset of routers to be upgraded to support such explicit exit choice.

4 Illustration using Linux Implementation

Figure 2(b) shows the topology of a simple validation experiment conducted on Utah's Emulab testbed with the Linux Zebra version 0.92a of OSPF upgraded with our traffic engineering building blocks. The forwarding plane was implemented in Linux using MIT's Click Modular Router package. Figure 2(b) also indicates the ip-addresses of various router interfaces and the link weights. The router ID is statically defined to be same as the ip-address of one of the router interfaces. However, for simplicity, we have chosen the smallest ip-address interface as the router ID.

Table 1 illustrates a partial forwarding table at node 1 (IP address 192.168.1.1) for destination 3 (192.186.3.3). Note that the path string shown in Table 1 is only for the sake of illustration and is not stored in the actual routing table. The pathIDs are the (MD5 + CRC-32) hashes of the router IDs (i.e. IP addresses of nodes) on the path. For example, the pathID 2084819824 corresponds to a hash of the set of router IDs {192.168.1.1, 192.168.1.2, 192.168.6.6, 192.168.39.9, 192.168.3.3 }. The path suffix ID is the hash of the suffix set formed after omitting 192.168.1.1. If the path goes through other nodes which are not upgraded (e.g. 1-4-3), the suffix path ID is the hash of the suffix path starting from the next upgraded router on the path. In the case of the path 1-4-3, both nodes 4 and 3 are not upgraded, so the suffix path ID is zero.

Outgoing I/f	Path	PathID	PathSuffixID
192.168.1.1	1-2-6-9-3	2084819824	664104731
192.168.3.1	1-3	599270449	0
192.168.4.1	1-4-3	4183108560	0
192.168.5.1	1-5-4-3	1365378675	0

Table 1: Partial routing table at 192.168.1.1 for destination 192.186.3.3

The results presented in this Section are for the sake of illustration. This is an ongoing work and more results can be found in a separate paper.

5 Conclusions and Future Work

The Internet today provides only a single path between any pair of hosts which fundamentally limits the throughput achievable between them. For example, dramatically faster large-file transfer or higher frame-rate video would be possible if applications could expect that multiple transport-level sessions would be mapped to different paths in the network. Our approach will enable higher bandwidth to the applications by using multi-path routing and allows explicit source-controlled traffic splitting at the source to achieve TE objectives. This is done by upgrading only a subset of routers and using the connectionless approach i.e. without using signaling. This is an important work towards extending the current routing algorithms to achieve multi-path routing with an evolutionary path to deployment.

Future work will focus on how new services and improved performance can be achieved using the proposed building blocks. This will address issues related to optimal or near-optimal traffic splitting, providing improved quality of service etc. using the building blocks proposed in this paper.

References

[1] J. Chen, P.Druschel, D.Subramanian, "An Efficient Multipath Forwarding Method," in *INFOCOM'98*, March, 1998.

[2] B. Fortz, M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights, in *Proceedings of the INFOCOM 2000*, pp. 519-528, 2000.

[3] P. Narvaez, K. Y. Siu, "Efficient Algorithms for Multi-Path Link State Routing," *ISCOM'99*, Kaohsiung, Taiwan, 1999.

[4] K.G. Ramakrishnan, M.A. Rodrigues, "Optimal Routing in Shortest Path Data Networks," *Bell Labs Technical Journal*, January-June 2001.

[5] S. Vutukury and J.J. Garcia-Luna-Aceves, " A Simple Approximation to Minimum-Delay Routing," *SIGCOMM '99*, September, 1999.