BANANAS: A New Connectionless Traffic Engineering Framework for the Internet *

Shiv Kalyanaraman, Hema Tahilramani Kaur, Satish Raghunath,
Jayasri Akella, Hemang Nagar, Kartikeya Chandrayana
ECSE Department
Rensselaer Polytechnic Institute
{shivkuma, hema, rsatish, kartikc}@networks.ecse.rpi.edu, {akellj, nagarh}@rpi.edu

ABSTRACT

We propose BANANAS, a connectionless framework for both intra-domain and inter-domain traffic engineering (TE) in the Internet. The key contributions of this framework are:
a) it allows the source to discover multiple paths and decide on how to split traffic among paths (assuming simple forwarding extensions in a subset of routers).
b) it does not require signaling, or high per-packet overhead.
c) it enables an incremental upgrade strategy for both intra-domain (OSPF) and inter-domain (BGP) routing to support TE capabilities.
d) in a fully upgraded network, every source can control how traffic is mapped to paths and therefore network-wide traffic engineering objectives can be achieved.

A path to a destination address is parsimoniously specified in a fixed-length "PathID" field in the packet header. "PathID" is the sum of link weights on the path (or the sum of Autonomous System (AS) numbers for inter-domain paths). This encoding allows efficient connectionless forwarding without using a signaling protocol. We describe extensions to OSPF, and BGP to support the proposed framework. We propose a simple multi-path computation algorithm under partial upgrade assumptions, discuss traffic splitting techniques and forwarding extensions. An ns-2 based simulation is used to demonstrate the framework and performance improvements.

1. INTRODUCTION

Traffic engineering (TE) is defined as *"..that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks..."* [2]. The goal of enhancing performance is accomplished by routing traffic in a way to utilize network resources efficiently and reliably. The term "traffic engineering" has been used to imply a range of objectives, including but not limited to, load-balancing, constraint-based routing, multi-path routing, fast re-routing and protection switching. Most work in the area of TE has focussed on solving one or more of the above problems within a *single, flat routing domain* (or area).

In this paper we develop a framework, called BANANAS, that would allow the incremental deployment of TE capabilities for both intra-domain and inter-domain settings within the hop-by-hop (or connectionless) routing model on the Internet. We focus on the TE objective of efficient utilization of network resources by using multi-path routing. The solution consists of multi-path computation and forwarding (at intermediate nodes) and multi-path computation (or discovery) and traffic-splitting (at the source) [1]. With partial upgrades, a subset of sources can benefit from these capabilities. With a fully upgraded network, every source can control how traffic is mapped to paths and therefore network-wide TE objectives can be achieved.

1.1 Hop-by-Hop vs Signaled Routing Models

Two broad classes of routing models dominate the current debate on next-generation routing and traffic engineering: *hop-by-hop* model [2] (distance-vector (DV), path-vector (PV) and link-state (LS)) and *signaled* model (implemented in technologies like MPLS [27] ATM and frame-relay).

In the hop-by-hop model, local knowledge is distributed to immediate neighbors, and ultimately reaches all nodes. Every node infers routes based upon this information. A consistency criterion ensures that the independent decisions made by nodes lead to valid, loop-free routes. The forwarding algorithm in this model is related to the control-plane algorithm because both use the same *global* identifiers (e.g. addresses, prefixes, link metrics, AS numbers). This relationship has, in the past, required changes in the forwarding algorithm whenever the control-plane algorithm was signifi-

*In loving memory of *Herbie goes Bananas* and great racing movie comedies, where inspite of impulsive multi-path choices and hop-by-hop misadventures, the racers never fail to reach the destination for a grand finale.

[1] By *"source"* we refer to some node in the data-path that takes multi-path computation (or discovery) and traffic-splitting decisions on behalf of the traffic originator (i.e. source host). Upgraded intermediate nodes provide next-hop forwarding to implement the source's path selection decision.

[2] a.k.a the *connectionless* model in this paper

cantly changed (e.g. subnet masking, CIDR). However, hop-by-hop routing protocols dominate the control-plane of the Internet (e.g. RIP, EIGRP, OSPF, IS-IS, BGP) for three important reasons; they support connectionless forwarding, they can be inter-networked easily, and they scale reasonably well. Traffic engineering capabilities in the hop-by-hop model, though attempted [24, 30, 5], have not found wide adoption in the Internet. Source routing in this model has typically meant that the entire path is enumerated in the packet – an undesirable overhead (e.g. IP, IPv6 options for strict/loose source route [16]). Multi-path algorithms for this model (e.g. [5]) have usually required the cooperation and upgrade of all routers in the network; and the decision of traffic-splitting is typically done in an ad-hoc manner at intermediate nodes without source control.

In the signaled model, local knowledge may be sent to all nodes through an approach similar to hop-by-hop algorithms. However, it is the source node or some central entity that, a) computes the desired paths and, b) decides what traffic is mapped to those paths. The intermediate nodes (switches) then set up local path identifiers (called *"labels"* in MPLS) for the paths. The signaling protocol allows autonomy in the choice of labels at switches, but ensures the consistency between label assignments at adjacent switches in the path. This leads to a label-switching forwarding algorithm where labels are switched at every hop. The forwarding algorithm in the signaled model is de-coupled from the control algorithms. This is because the forwarding algorithm uses *local* identifiers (labels), whereas the control algorithms use *global* identifiers (addresses). The signaling protocol maps and ensures consistency between local and global identifiers. This de-coupling between forwarding and control-planes allows the introduction of new TE capabilities by modifying the control plane alone. However, signaled approaches have historically been hard to inter-network (e.g. IP over ATM [22], Non-Broadcast Multiple Access(NBMA) routing [23] or multi-domain signaled TE), and hence have been limited to intra-domain or intra-area deployments (e.g. MPLS, ATM).

We conjecture that the key reasons for the lag in adoption of connectionless TE capabilities include the need for complete network upgrades, lack of source-based or explicit operator control over TE decisions, lack of a common reference framework that allows long-term evolution of TE capabilities. This paper proposes to fill these needs. The contributions in this paper include:

- A framework, called BANANAS, which allows sources to compute (or discover) multiple paths within the connectionless routing model and decide on how to split traffic among these paths.
- Allowing a subset of nodes to participate in the TE process, i.e., with partial upgrades.
- Developing a simple and efficient path encoding to specify the path as a short, fixed-length field in a packet; and a corresponding forwarding algorithm.
- Mapping the BANANAS framework to current intra- and inter-domain protocols (e.g. OSPF, BGP).
- Examining preliminary options for various sub-blocks of this framework (e.g. multi-path and traffic splitting algorithms for partially upgraded networks)

The BANANAS framework is not intended to replace MPLS-based TE within a routing domain, but it may provide an alternative to non-MPLS routing domains which currently deploy OSPF or IS-IS. To the best of our knowledge, the BANANAS framework is the first attempt to provide an *incremental upgrade strategy* for connectionless TE, and to support a broad set of TE capabilities for the *inter*-domain case in the Internet. Indeed, MPLS-TE within an AS (or area) can be complemented with BANANAS-TE across autonomous systems.

2. BANANAS FRAMEWORK: BASIC IDEAS

The BANANAS framework allows partial upgrade of intermediate nodes to support *multi-path computation and multi-path forwarding*; and partial upgrade of sources to support *multi-path computation (or discovery)* and *traffic-splitting* strategies. Recall that *"source"* is a node in the data-path that takes multi-path computation/traffic-splitting decisions on behalf of the traffic originator (i.e. source host) and has some visibility into the available paths in the network. The following sub-sections define paths, path suffixes and path identifiers, develop a forwarding algorithm and compare these concepts to labels in the signaled routing model.

2.1 BANANAS Forwarding Concepts

Consider a network modeled as a graph with links and nodes, where links are given weights (not necessarily unique). Consider a path from node i to node j, which passes through links of weights $w_1, w_2, ..., w_m$. This is illustrated in Figure 1. Define the $PathID(i, j, w_1, ..., w_m) = (w_1 + w_2 + ... + w_m) \bmod 2^b$, where b bits are used to encode $PathID$, which we propose to include as a new field (or option) in packet header. Note that the $PathID$ could alternatively be defined as the sum of node identifiers instead of link weights. We use this definition to map the framework to BGP-4, wherein we define $PathID$ as the sum of autonomous system numbers (ASNs). Observe that the tuple (destination address, $PathID(i, j, w_1, ..., w_m)$) at node i defines the path to the destination. We refer to this tuple henceforth as the *forwarding tuple*, and use the shorthand (Destination, PathID).

For simplicity of exposition, assume that the *forwarding tuples (Destination, PathID) are unique*. If the link weights vary between a sufficiently large range (i.e. take diverse values), the *forwarding tuples* (Destination, PathID) can be expected to be unique with a high probability. Since both link weight and AS number are 16-bit fields, any reasonably diverse assignment of link weights would suffice. The uniqueness probability also depends on the size of the network and connectivity, and we leave rigorous analysis of this issue for future work. However, note that if the *forwarding tuple* (Destination, PathID) is non-unique, i.e. *tuple collision* does occur, the router can apply a local heuristic (e.g. hashing) to map this traffic to the paths with the same forwarding tuple. In Figure 1, if k is an *intermediate node* on a path from i to j, we refer to the residual path from k to j as the *path suffix*. At any intermediate node k, a list of path suffixes to a destination prefix can be condensed as a list of *forwarding table entries (destination prefix, next-hop, PathSuffixID)*, where PathSuffixID is simply the PathID computed for the path suffix as shown in Figure 1. The forwarding table entry is indexed by processing the destination address and PathID

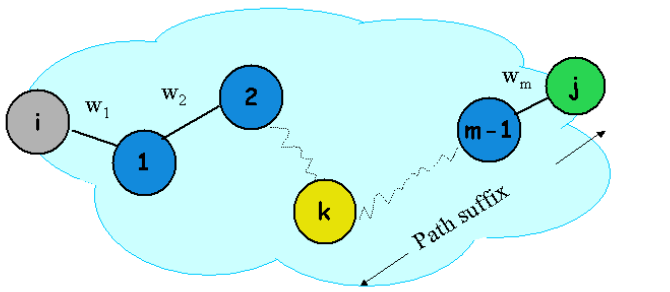

Figure 1: Path Suffix and PathID in BANANAS

fields in incoming packet headers. The PathID field in packets is initialized at the "source," which reflects source-based control of traffic splitting to paths on a packet-by-packet basis. Intermediate nodes merely honor the path selection choice of the source in a best-effort manner.

2.2 BANANAS: Abstract Forwarding Algorithm

The *abstract* packet-forwarding algorithm [3] in the BANANAS framework is as follows. Intermediate nodes find the longest destination address prefix match first, and then the *nearest* PathID match among paths to that destination. Nearest PathID matches the *largest* PathSuffixID less than or equal to the PathID on the packet, or the PathID of the default path. Once this match is found, nodes update the header PathID field by subtracting the next-hop link weight from it (modulo 2^b), and forward the packet. If the PathID field is smaller than the smallest PathSuffixID, then the PathID field is set to the value of the smallest PathSuffixID minus the next-hop weight before forwarding. Routers that do not support multiple paths ignore the PathID field; they simply look at the destination address field and apply the regular longest-prefix-match forwarding algorithm.

In a network where all nodes support multi-path forwarding, assuming that:
a) there exists a loop-free path from i to j through k, whose PathSuffixID at k is $PathID(k, j, w_{k+1}, ..., w_m)$, and next-hop is $k+1$;

b) the source has chosen an initial PathID W, and

c) the packet has crossed k links with weights $w_1, w_2..., w_k$, and we find a nearest PathID match $PathID(k, j, w_{k+1}, ..., w_m)$, then we have one of the following conditions satisfied:

C1: $W \geq \{w_1+w_2+...+w_k+PathID(k, j, w_{k+1}, ..., w_m)\} mod 2^b$ and $PathID(k, j, w_{k+1}, ..., w_m)$ is the largest PathSuffixID that satisfies the inequality, or

[3]We will map this abstract algorithm with some minor changes to the intra-domain case, and extend it for the inter-domain case.

C2: $W < \{w_1+w_2+...+w_k+PathID(k, j, w_{k+1}, ..., w_m)\} mod 2^b$ and $PathID(k, j, w_{k+1}, ..., w_m)$ is the PathSuffixID of the default path suffix (i.e. smallest PathSuffixID value).

- If condition **C1** is satisfied, w_{k+1} is subtracted from the PathID field (modulo 2^b) before forwarding.
- If condition **C2** is satisfied, the PathID field is set to $PathSuffixID(k, j, w_{k+1}, ..., w_m) - w_{k+1}$ before forwarding. Note that condition **C2** maps packets with errant PathIDs to the shortest path.
- The packet is forwarded to node $k+1$.

Given the uniqueness assumption of PathIDs between any pair of nodes, the above observations represent the new consistency criterion in our framework and guarantee a forwarding match. We choose to allow the inequality (rather than the equality) because we assume that sources choose the path autonomously; and at intermediate nodes traffic to non-existent paths can be distributed among the set of available paths (and certainly mapped to the default path in the worst case). In BGP we shall see that the default path need not be the shortest path.

2.3 BANANAS PathIDs vs MPLS Labels

It is interesting to compare the notion of PathID to the notion of "label" used in the signaled models such as ATM and MPLS:

- The forwarding tuple (destination address, PathID) can be thought of as a globally significant path identifier, just like an IP address is a *globally significant interface ID* and an IP prefix is a globally significant network ID. In contrast, the MPLS label has only a local meaning, requiring a signaling protocol to map labels to global addresses. The signaling requirement makes it hard to map a label-swapped routing system to OSPF and BGP. A caveat is that unlike addresses, non-unique forwarding tuples are possible with low probability.
- The PathID field by itself does not designate the path; it needs to be interpreted along with the destination address. In contrast, the label is a stand-alone field.
- Both PathID and labels are updated at every (upgraded) hop. But PathID is updated through a *computation* (subtract operation) whereas a label is *swapped* with a completely new label based upon a label-table.
- Since PathID can be defined in terms of link weights (or ASNs), it can be mapped to intra- and inter-domain protocols with minor modifications as discussed in later sections. In contrast, the label-swapping and the signaled model are hard to map to current inter-domain protocols (BGP).
- Though the use of the (Destination address, PathID) tuple still relates the forwarding and control planes due to the use of global IDs, it gives a valuable handle (global path identifier) for TE functions. Given this handle, a range of future TE control-plane functions may be deployed without any further forwarding-plane support at intermediate nodes. Recall that in MPLS,

the use of local IDs (labels) for forwarding and global IDs (addresses) for control de-couples the two planes, and allows deployment of new TE control functions without affecting the forwarding plane.

3. BANANAS: INTRA-DOMAIN MAPPING

To understand the mapping issues of the BANANAS framework in an inter-domain context, let us consider a single-area OSPF network, with point-to-point links (i.e. no hierarchy). We assume that each upgraded node knows all other nodes which support multi-path capabilities. This knowledge can be achieved through a single-bit ("multi-path capable" or MPC bit) in the link-state advertisement (LSA), that is zero by default. Multi-path capable routers set the MPC bit to 1 in every LSA they originate. In section 6.1 we describe a multi-path computation algorithm that finds *all* possible loop-free paths to any destination given that it knows the subset of nodes that also support multi-path capabilities.

Mapping to other link-state protocols like IS-IS is very similar to OSPF. Path-vector issues are considered for BGP in Section 4. We do not consider RIP because it is deployed in small networks, and allows only a simple hop-count as a PathID. In other distance-vector (DV) protocols (e.g. EIGRP), the PathID is simply the "distance" of the chosen path. So, a similar forwarding strategy to OSPF can be used if nodes are upgraded for multi-path forwarding.

The central problem in DV protocols, vis-a-vis multi-path computation under partial upgrades is the lack of topology visibility. This leads to two issues: a) multi-path enabled nodes do not see which other nodes are multi-path capable, and b) even if (a) were solved, nodes cannot figure out how to concatenate loop-free path segments such that the entire path is loop free. However, simple multi-path algorithms which compute a partial set of multi-paths under partial upgrades are proposed by Narvaez et al [24] and Vutukury et al [30]. These depend upon the criterion that if a neighbor's distance to a destination is smaller than the current shortest path, then it is safe (loop-free) to use a path through that neighbor.

3.1 Intra-Domain Multi-Path Forwarding

We split this section into four parts. First, in section 3.1.1 we consider the simplest case with the following assumptions:

- *All* nodes are multi-path capable.
- Multi-path capable nodes use the *same* multi-path computation algorithm, and support *forwarding to all available routes* to any destination.
- A single-area flat routing domain is used.

Under these assumptions, the views of multi-paths in the steady state is same at any multi-path enabled node. Second, in section 3.1.2 we consider the case where a subset of nodes support multi-path capabilities. Third, in section 3.1.3, we consider a case where the upgraded nodes may use different multi-path computation algorithms and/or may support forwarding to only a limited number of paths. Fourth, in section 3.1.4 we consider hierarchical intra-domain multi-path routing.

3.1.1 Fully Upgraded Network Case

In the case where *all* routers in the network support multi-path capabilities, the forwarding model closely follows the description in Section 2. In particular, for intra-domain operation we propose to extend the IP packet header with a 32-bit field (or new routing option), called the *"i-PathID."* A 32-bit field should be sufficient to assure no wrap-around because OSPF link metrics are 16-bit fields[4]. The i-PathID is initialized by the host or the first-hop router that participates in multi-path routing and traffic splitting.

The initialization value of i-PathID is the sum of weights of links along the path modulo the field space (2^b). The actual choice of the path for every packet depends upon the traffic splitting strategy (e.g. see Section 6.2). Every intermediate router does a longest-prefix-match on destination address, and a nearest PathID match (which turns out to be an exact PathID match in steady state, under the assumptions made) of i-PathID to determine the next-hop. The i-PathID value in the packet header is decremented by the value of the weight of the link to the next-hop before the packet is physically forwarded. When the packet reaches the destination, it will have an i-PathID value of zero. Note that an i-PathID value smaller than the smallest PathID will be re-mapped to the shortest path, with a new i-PathID corresponding to the shortest path. This way, it is guaranteed that even under transient routing conditions, the packet ultimately defaults to the shortest path.

3.1.2 Partially Upgraded Network Case

Next we consider the case of partial upgrades, i.e., not all nodes support multi-path computation and forwarding. In this case, the total number of paths to any destination is likely to be smaller. Moreover, nodes which do not support multi-path forwarding will ignore the i-PathID field, and will not update it. If the originating node is not multi-path enabled, the packet is sent in the default (shortest) path and the routing option is not used. In this case, the operator could configure a set of upgraded nodes to make multi-path decisions on behalf of hosts if packets from those hosts flow through them. Else, the forwarding is the default IP forwarding.

If the originating node ("source") is multi-path enabled, it first chooses a path for a packet. Then it uses a slight variant in the forwarding process. Before forwarding the packet, it decrements i-PathID by the *sum of link weights of consecutive links until a multi-path or destination* is reached. Essentially, the series of hops across non-upgraded nodes is viewed as a single *"virtual-hop"* for the purposes of the i-PathID decrementing function. Note that due to the lack of topology or path visibility, the virtual-hop feature cannot be implemented in DV protocols (e.g. RIP, EIGRP), but can be achieved in PV protocols like BGP.

For example in the Figure 2, nodes A, C and D are multi-path enabled, and node A is the originating node for a packet destined to node F. The shortest path from intermediate

[4]It would require the sum of at least 64K 16-bit numbers (> 64K-hop paths) to wrap-around a 32-bit field! Smaller fields have a risk of wrap-around and larger tuple collision probability.

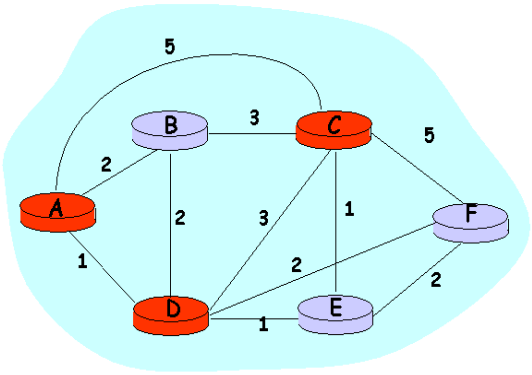

Figure 2: Multi-Path Forwarding with Partial Upgrades

node B to node F is B-D-F (with path weight of 4). Observe that the path A-B-C-F is not available for forwarding. Node B (which is not upgraded) cannot honor the path choice since the only possible next hop from B to destination F is node D. However, paths such as A-B-D-C-F, A-D-E-F, A-D-C-E-F are available because nodes A, C and D are multi-path capable.

If the path A-B-D-E-F is chosen, then the i-PathID is initially 7. But since B does not support multi-path forwarding (and i-PathID update) capability, A sets i-PathID to 3. In other words, A views the pair of hops A-B and B-D as a single virtual-hop for the purposes of i-PathID update. Node b ignores the i-PathID field and forwards it on its perceived shortest-path (i.e. to D). Node D is multi-path enabled, and realizes that the next-hop should be E. But since E is not multi-path capable, node D sets the i-Path ID to zero. Node E forwards the packet to F without looking at the i-PathID.

If path A-D-C-E is chosen, all nodes in the path are multi-path capable, and hence the i-PathID value transmitted to D is 8. Node D updates i-PathID to 5 and sends it to node C. Node C updates i-PathID to 0, and forwards to node F. This case is similar to the forwarding behavior in fully-upgraded networks explained in Section 3.1.1.

To enable this forwarding and update operation, we assume that the forwarding table entries at upgraded nodes consist of tuples: *(Destination prefix, PathSuffixID, Next-Hop, VirtualHop Weight)*. The first two entries of the tuple are matched as described earlier to determine the next-hop, and the VirtualHopWeight is subtracted from the i-PathID. The VirtualHopWeight is the link-weight of the outgoing link if the next-hop is multi-path enabled. Otherwise it is the sum of the link-weights of each link in the path till a multi-path enabled router or destination is hit. This value is also entered in the forwarding table as part of the multi-path computation algorithm (see Section 6.1).

3.1.3 Heterogeneous Multi-Path Capabilities

In this section, we assume that different multi-path computation approaches may be used at different nodes, and forwarding at a multi-path node may be supported only to a finite and arbitrary number of multi-paths per-destination. We still assume the use of the MPC-bit in LSAs, which allows multi-path enabled nodes know the subset of nodes that support multi-path capabilities.

For the purpose of multi-path computation, each node *assumes* that other multi-path enabled nodes *compute all* possible multi-paths as before. However, it makes an autonomous local decision on: a) how many multi-paths it *computes* and, b) how many it *stores* in its forwarding table (a filtering decision). Now, the problem is that a node may assume the existence of a path which in fact does not exist due to the autonomous filtering decisions of other multi-path enabled nodes. If packets are sent to this path, a remote multi-path node will re-map the packet to a different path (and in the worst case to the shortest path).

We refer to this kind of capability as "best-effort" traffic engineering support. In other words, the network makes a best-effort to send the packet on the chosen path, and remaps it to another potential path if the chosen path is not available. Optionally, sources can autonomously check for route existence (e.g. through traceroutes carrying PathIDs).

For example, consider a slight modification of the case shown in Figure 2. As shown, nodes A, C and D are multi-path enabled, and node A is the originating node for a packet destined to node F. However, in this case, assume that D autonomously decides *not* to store routes D-C-F and D-E-F. When a packet specifying D-C-F (i.e. destination F, i-PathID = 8) arrives, it will be mapped to the nearest match: D-C-E-F (i.e. i-PathID = 6). The packet will be forwarded to node C, with i-PathID=3. Node C then matches the packet to the default (shortest path), and observes that E is not upgraded. Hence node C sets i-PathID to zero and forwards it to E. Node E ignores the i-PathID (since it is not upgraded) and forwards the packet to node F.

3.1.4 Hierarchical Routing Case

The final assumption we will relax is that of a single-area link-state routed network. Large OSPF and IS-IS networks support hierarchical routing with up to two levels of hierarchy, with the root area called area 0. We consider "normal" and "totally stubby" areas [23]. In normal areas, summary LSAs (inter-area) and external LSAs (inter-AS) routes are flooded by Area Border Routers(ABRs). This allows internal nodes to choose an exit ABR, based upon advertised distances to remote areas. In "totally stubby areas" summary-LSAs and external-LSAs are not flooded within the area. In both cases, intra-area nodes cannot see the topology of area 0, or that of other areas. ABRs can see the topology of area 0, but cannot see the topology of other areas.

Our approach to handling both these cases is to view each area as a flat routing domain for purposes of multi-path computation. Multi-paths are found locally within areas, and crossing areas is viewed as like crossing to a new multi-path routing domain. In the case of normal areas, internal

nodes can choose an ABR and then decide on multi-paths to that ABR. In the case of totally stubby areas, internal nodes do not have a choice of ABRs since they forward to 0.0.0.0/0 (default route). However, they still can choose multi-paths within the area to address "0.0.0.0," resulting in multi-paths to the default exit ABR.

For inter-area multi-path forwarding, we propose to re-use the i-PathID field after crossing area boundaries. Note that this operation is *different from inter-domain multi-path forwarding* where we propose to use a new field (e-PathID, defined in Section 4.2). For example, if a source needs to send a packet outside an area, it chooses one of the multi-paths to the (default or chosen) area border router (ABR). Then, the ABR may choose among has several multi-paths within area 0 to other ABRs. The i-PathID field is re-initialized by the first ABR at the area-boundary.

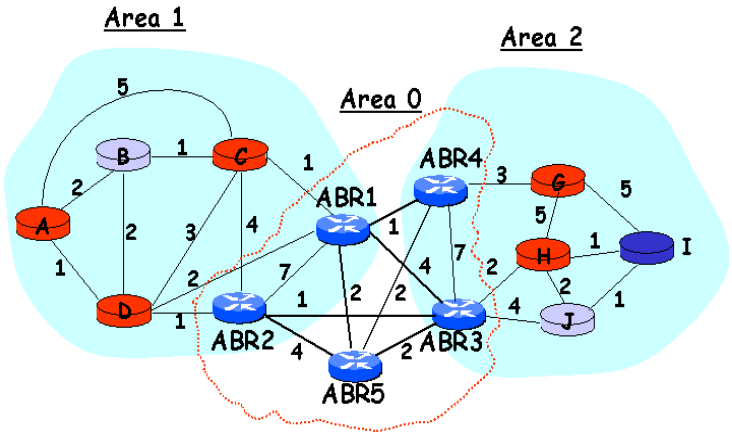

Figure 3: Hierarchical Multi-Path Forwarding

Consider the hierarchical routing scenario in Figure 3 which is an extension of Figure 2 (albeit, with some link-weight changes). As before in area 1, nodes A, C and D are multi-path enabled, and node A is the originating node. Assume that A wants to send packets to node I in area 2. ABR1 and ABR2 are the area border routers for area 1. Assume areas 1 and 2 are "normal" areas [23], i.e., summary-LSAs (and external-LSAs) are flooded into the area. ABR3 and ABR4 flood summary-LSAs into area 0 (advertising reachability to area 2) with costs 7 and 9 respectively (i.e. cost of longest path from the ABR to any node within area 2). ABR1 and ABR2 add their shortest inter-area costs to area 2 and advertise costs of 10 and 8 respectively within area 1. Therefore, normally nodes A and D would choose ABR2 as their exit ABR, whereas nodes B and C would choose ABR1 as their exit ABR to reach area 2 destinations. However, multi-path enabled nodes A, C, D can choose either exit ABR. For example, A can choose any of the paths: A-B-C-ABR1-area2, A-B-C-ABR2-area2, A-D-ABR1-area2, A-D-ABR2-area2, A-D-B-C-ABR1-area2 etc. As before the path prefix [A-B-D-...] is not available since B does not support multi-path forwarding and sends packets with destinations in area2 to node C. The i-PathID for A-B-C-ABR1-area2 is initially 4 (intra-area) + 10 (inter-area) = 14. As before, the two hops A-B-C is considered as a virtual hop with weight 3 for forwarding purposes.

Now when the packet reaches ABR1, the i-PathID field has a value 10 (which refers to path ABR1-ABR4-area2). However, since ABR1 may choose one of many area 0 paths to area 2, the i-PathID field set by A may be ignored and re-initialized by ABR1. For example, ABR1 may choose the paths ABR1-ABR5-ABR3-area2, ABR1-ABR3-area2 etc. Assuming it chooses ABR1-ABR5-ABR3-area2, the initial i-PathID is 2+2+7 = 11, and next-hop ABR5. When the packet reaches area2, ABR3 may choose only of many paths to reach I (eg: ABR3-H-I, ABR3-J-I, ABR3-H-G-I etc) and forward packets as described in Section 3.1.2. Note that if the areas were "totally stubby" areas, the only difference is that all intra-area nodes (multi-path or not) would have a default-exit-ABR (i.e. no choice of exit ABR). Multi-paths can be chosen within each area however, as described above.

4. BANANAS: INTER-DOMAIN MAPPING

BGP-4 is *the* inter-domain routing protocol in the Internet. It is a path vector protocol which announces paths to a destination prefix if the AS is actively using those paths. Our inter-domain TE goal in the BANANAS framework is to enable multi-AS-paths from the source to the destination. Within each transit AS, multi-paths may be chosen under the control of the entry border router (entry AS-BR). An AS may be structured internally as a hierarchical OSPF or IS-IS network, and the internal forwarding then follows the discussion of earlier sections.

Our first observation is that *BGP as-is does not disallow multiple AS-path advertisements* to any destination prefix. A quick scripting check on a number of recent routing tables from RIPE/NCC indicates that such multi-AS-path announcements do not happen today, consistent with single path inter-domain forwarding assumptions. So, if we extend a single AS to autonomously support multi-AS-path forwarding, then it can leverage BGP to advertise multiple AS paths (to any destination prefix) to its neighbor ASs. Therefore any AS can infer that its neighbor AS has multi-AS-path capabilities merely from the fact that it is advertising multiple AS-paths (and that the neighbor AS is the forking point for the multi-AS-paths) to the destination prefix of interest.

Moreover, since BGP-4 is a path-vector protocol, the multi-path *computation* algorithm extension at any BGP router is trivial. Today, BGP-4 applies policies as a series of tie-breaker rules to choose one route to a prefix. A multi-path computation extension would allow multiple paths to be chosen after they are pre-qualified by a set of filtering rules. But, upgrading a single BGP router in an AS is not sufficient. BGP expects synchronization between all i-BGP and e-BGP routers in an AS before routes can be advertised outside the AS. Also, because of the DV-nature of BGP, the multi-AS-path information may not be propagated beyond the immediate neighbors of a multi-AS-path enabled AS. This is because such neighbor ASs may not support

multi-path forwarding. We propose simple extensions BGP to address these issues in the following subsection.

4.1 Re-advertisement & Synchronization

In this section, we make a distinction between multi-path re-advertisement *within* an AS (which determines the complexity of upgrades of i-BGP and e-BGP nodes), and re-advertisement *across* AS-boundaries. *Across ASs* the central issue is that, if neighbor ASs do not relay (re-advertise) at least a subset of the multi-AS-paths available from an AS, remote ASs will not be able to take advantage of such multi-AS-paths. This is a direct result of the path-vector (i.e. extended distance vector) routing paradigm used by BGP-4. *Within an AS*, the central issue is that BGP expects synchronization between e-BGP and i-BGP nodes before information is advertised to other AS's.

Moreover, we distinguish between multi-AS-path *re-advertisement* and multi-AS-path *forwarding* capabilities at an AS. In particular, we allow selective multi-AS-path re-advertisement even when the AS itself does not support multi-path inter-domain forwarding internally. By this, we mean that i-BGP and e-BGP routers may store multiple AS-paths to a prefix in their Routing Information Bases (RIBs), and re-advertise them under certain conditions, but they need not support multi-path forwarding entries in their Forwarding Information Bases (FIBs) and need not possess any multi-path data-plane forwarding capabilities (see section 4.2).

4.1.1 BGP Multi-AS-Path Re-advertisement

Consider an example where AS0 supports and advertises multiple AS paths $\{p_1, p_2, ..., p_n\}$ to destination prefix d. Observe that *if neighbor AS1 chooses AS0 as its next-AS-hop for prefix d* (eg: on the basis of AS-path p_i) it can safely re-advertise all the AS-paths : $\{(AS1\ p_1), (AS1\ p_2), ..., (AS1\ p_n)\}$ even if it does not support multi-AS-path forwarding within AS1. This is possible because, irrespective of the source path choice, all traffic to prefix d in AS 1 would be forwarded to AS0 anyway. The particular AS-path choice would be made only at AS0. AS1 hence would act as a relay for multi-path traffic, even though it does not possess multi-path forwarding capabilities itself. Note that currently the e-BGP protocol in AS1 would announce only the AS-path $(AS1\ p_i)$.

A concrete example is shown in Figure 4. AS0 has three AS-paths to destination prefix "d" that is in AS4. These AS paths are represented as (0 4), (0 3 4) and (0 5 4). Now, AS0 can be simply configured to announce this to AS1. Assume AS1 chooses the AS-path (0 4) as its choice for forwarding packets to destination d. Normally, BGP will only announce the AS path (1 0 4) to AS2. However, we propose in this case that, since AS1 has a forwarding path through AS0, it is safe for AS1 re-advertise the other AS-paths to AS2, i.e. it would advertise { (1 0 4), (1 0 3 4), (1 0 5 4) } to AS2.

Let us look closely at the proposal to re-advertise the set of AS paths $\{(AS1\ p_1), (AS1\ p_2), ..., (AS1\ p_n)\}$. To avoid a host of ambiguities, we propose that this re-advertisement be tagged with a new BGP "re-advertisement" attribute which lists the ASNs of the AS's that are merely re-advertising

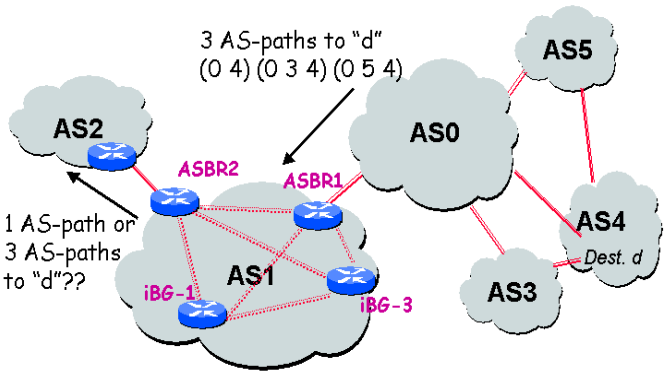

Figure 4: Re-advertisement of Multi-Paths by BGP

AS-paths, and do not support multi-path forwarding. When re-advertising routes without supporting multi-path forwarding, the AS will append its ASN to the list of re-advertising ASNs. This will allow a remote AS to unambiguously identify the AS's which support multi-path forwarding. Observe that a neighbor of AS1 (say AS2) will now parse and interpret these re-advertisements to mean that the remote autonomous system, AS0 supports multi-AS-paths (because it is the forking point for the AS-paths). Furthermore, it will know that AS1 is merely re-advertising these AS-paths.

4.1.2 BGP Synchronization Issues

Under current BGP-4 semantics the re-advertisement capability must be supported by both the i-BGP and e-BGP routers before the entire AS can be declared to have this re-advertisement capability. In particular, both i-BGP and e-BGP routers store multiple AS-paths for prefixes in the RIBs; but not necessarily in the FIBs. An alternative would be to weaken BGP's synchronization assumption between i-BGP and e-BGP, and require only the e-BGP nodes to synchronize on these re-advertisements. This method would work only if inter-domain multi-path packets are tunneled through the AS from the entry AS-BR to the exit AS-BR (see Section 4.2).

In either of these alternatives, observe that the first e-BGP AS-BR (that sees multi-path advertisements from neighbor AS's) alone can make a decision on a prefix-by-prefix basis whether to re-advertise AS-paths. In other words, the first AS-BR could decide to re-advertise a *subset* of the AS-paths $\{(AS1\ p_1), (AS1\ p_2), ..., (AS1\ p_n)\}$ once it accepts p_i. It can also decide to re-advertise only a subset of the AS-paths. Other BGP routers in the AS then merely relay such re-advertisements and populate their RIBs.

Note that this *selective re-advertisement* concept is similar in spirit to the link-state idea of propagating information from originating node throughout the network, albeit, only for a filtered subset of the information. Therefore,

once AS's support multi-AS-path readvertisement capabilities, even though BGP does not have full AS-topology information (unlike link-state algorithms), because of its path-vector nature it has full information about a subset of multi-AS-paths available to any destination prefix. This information is what allows multi-AS-path forwarding with source-control in the BANANAS framework.

4.2 Inter-domain Multi-Path Forwarding

Recall that the BANANAS framework concept of inter-domain multi-path is to allow the source AS control over the choice of the particular AS-level path. Intra-AS paths in transit AS's are decided locally by the entry AS-BR. To illustrate the issues in multi-path forwarding across transit AS's, consider the scenario in Figure 5. AS0 is a customer AS, which buys transit from AS1 and has traffic to destination d in AS4.

We propose that the exit AS-BR (AS border router) of AS0 initializes a new packet header field, the *inter-domain PathID* (or *e-PathID* for shorthand) to specify its AS-path choice for destination d. Recall that intra-domain PathIDs are called *"i-PathIDs."* The e-PathID for BGP is defined to be the sum of the AS numbers (ASNs) of the AS's on the path modulo 2^b where b is the e-PathID field length in bits. Observe that, from a graph-theoretic viewpoint, the e-PathID is a sum of node IDs as opposed to the i-PathID which is a sum of link weights. This change in semantics implies that all intermediate AS's compute PathSuffixIDs as sum of ASNs of AS-path-suffixes, and subtract their own ASN (or next-virtual-AS-hop ASN sum) from the e-PathID during the inter-domain forwarding process (i.e. at the entry AS-BR).

We recommend that the e-PathID field size be 32 bits, because current ASNs use a 16-bit space, and only the lower portion of the ASN space is allocated. Even though an ASN-space extension to 32-bits is proposed at the IETF, we do not expect this to be a problem because only the lower portions of the ASN space will be assigned in the foreseeable future. Moreover, unlike link-weights, ASNs are guaranteed to be *unique* since they are identifiers for autonomous systems. Hence the e-PathID (which is the sum of unique ASNs) and the inter-AS forwarding tuple (destination, e-PathID) has a much higher probability of being unique, even in the remote future possibility of e-PathID wraparound with 32-bit ASNs.

In Figure 5, the entry AS-BR (ASBR1) of the transit provider (AS1) uses the inter-domain forwarding tuple (destination, e-PathID) to determine the next AS-hop, i.e., the next AS to which the packet has to be transmitted. Assume that AS1 is multi-path forwarding enabled and has two AS-paths to destination d (or its prefix): one path through peer AS2 and another path through peer AS3. If AS0 chooses the path (0 1 3 4), it will initialize e-PathID to 8 to imply a next-AS-hop of AS3 at ASBR1. Moreover, assume that AS1 has two AS-BRs (ASBR2 and ASBR3) peering with AS3. This poses three forwarding problems for ASBR1:

1. How to ensure the packet forwarding within AS1 so that the packet reaches AS3, as specified by the source AS0 ? What forwarding enhancements in internal iBGP routers are needed ? Can the packet-forwarding be done if intermediate iBGP routers do not support these enhancements (partial upgrade scenario)?

2. How to choose the exit AS-BR to reach AS3 (ASBR2 or ASBR3)? How to ensure that the packet is forwarded to the chosen exit AS-BR ?

3. If multiple paths are available to the chosen AS-BR, can it be specified in a manner similar to the intra-domain multi-path case?

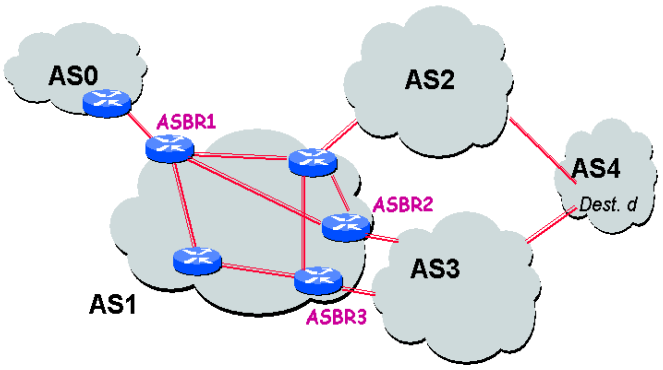

Figure 5: BGP Multipath Forwarding Scenario

First, we observe that the entry ASBR (ASBR1) is the only node that processes and updates the e-PathID field by subtracting its own ASN of the next-AS-hop. Similar to the discussion in Section 3.1.2, the entry-ASBR can update the e-PathID by subtracting the sum of ASNs of AS-hops which are known not to support multi-path forwarding (i.e. Virtual-AS-Hop ASN). The *conceptual* inter-domain forwarding table at the entry AS-BR would have a list of tuples: *(Destination prefix, AS-PathSuffixID, Next-AS-Hop, Virtual-AS-Hop-ASN)*. A minor difference in e-PathID processing is that packets with errant e-PathIDs are mapped to a *default* AS-path which may or may not be the shortest AS-path available (i.e. chosen by policy like in today's BGP).

Next, we note that the above inter-domain decision does not resolve the intra-AS transit forwarding issues raised. The simplest solution for these issues is to encapsulate (tunnel) the packets across the AS, with the chosen exit AS-BR address as the destination address, and the chosen intra-domain PathID in the outer header. The exit AS-BR is obtained by resolving the Next-AS-Hop field in the tuples mentioned above. Without loss of generality, we can consider the inter-domain forwarding tuples at entry-ASBR's to be of the form: *(Destination prefix, AS-PathSuffixID, exit AS-BR, Virtual-AS-Hop-ASN)*.

The exit AS-BR then de-capsulates the tunneled packet and performs e-PathID processing as mentioned earlier. No new forwarding plane support is needed from internal iBGP routers in the path (over-and-above optional intra-domain multi-path support discussed in previous sections). This solution runs into the usual tunneling/encapsulation issues: per-packet overhead, potential configuration of tunnel endpoints, and implementation of encapsulation in the slow-path of current routers (AS-BRs only). We believe these issues could be addressed due to growth in aggregate ISP bandwidth, automated configuration tools and potential implementation of encapsulation in fast-path of future routers.

The alternative transit forwarding strategy is to add a new field for the *exit AS-BR address* in the routing option. This field would be in addition to the previously proposed *i-PathID* and *e-PathID* fields. Internal iBGP routers of AS1 will be configured to ignore the destination address and simply use the exit AS-BR field as the destination; and i-PathID to specify the particular path to the exit AS-BR. This solution is in fact similar to the encapsulation approach, except for the fact that the AS-BR address and is put into the routing option, and the overhead of the remaining outer IP header fields is avoided. The downside is that *all* internal iBGP routers should support this enhanced forwarding plane before this alternative can be enabled. The exit-ASBR field can be 32-bits for both IPv4 and IPv6. The field would hold the exit AS-BR IPv4 address (for IPv4), and a condensed (or locally mapped) version of the IPv6 address (for IPv6).

In summary, the partial upgrade strategy in BGP is start with a re-advertisement capability (only at e-BGPs, or in both e-BGP and i-BGP). Then forwarding capabilities either are provided only at e-BGP routers (tunneled case) or all the routers are upgraded (exit AS-BR case).

5. PUTTING IT TOGETHER

The BANANAS framework proposes to support both intra-domain and inter-domain paths, parsimoniously encoded in just three 32-bit fields in packet headers. Observe that the *proposed per-packet overhead is smaller than a 128-bit IPv6 address.* The i-PathID is used for intra-AS multi-path forwarding, and is re-initialized after crossing area or AS boundaries. The i-PathID is the sum of link weights on the path suffix. The e-PathID is the sum of ASNs on the AS-path-suffix, and is processed only at AS-boundaries. The exit-ASBR field is used for transit forwarding within an AS. This field is not required if packets will be tunneled across AS's from entry ASBR to exit AS-BR.

The intra-domain forwarding tables at upgraded routers would have tuples *(Destination prefix, PathSuffixID, Next-Hop, VirtualHop Weight)*, which are indexed after processing the forwarding tuple (Destination, i-PathID) for longest prefix destination match and nearest-PathID match. The value VirtualHopWeight is subtracted from the i-PathID packet field. Packets with errant i-PathIDs will be mapped to the shortest path, and their i-PathID appropriately re-initialized. The OSPF LSA's [23] can be extended with one bit to indicate whether the router is multi-path capable (MPC). Note that this bit is required only on LSAs, and not on data-packets.

In distance-vector protocols, the lack of topology visibility allows only simple multi-path algorithms under partial upgrades which may not compute all available multi-paths [24, 30].

The inter-domain forwarding at entry AS-BRs (i.e. e-BGP routers) would have tuples *(Destination prefix, AS-PathSuffixID, exit AS-BR, Virtual-AS-Hop-ASN)*, which are indexed by processing the inter-domain forwarding tuple (Destination, e-PathID) for longest-prefix destination match and nearest-PathID match. The value Virtual-AS-Hop-ASN is subtracted from the e-PathID packet field. A minor difference compared to the intra-domain routing is that packets with errant e-PathIDs are mapped to a *default* AS-path which may or may not be the shortest AS-path available (i.e. is chosen by policy). The exit AS-BR value is either used to initialize the the tunnel header destination or in the proposed exit-ASBR field. The i-PathID field may be re-initialized to specify a transit intra-domain path through the AS.

Since BGP is a path-vector protocol, re-advertisement of multi-paths is critical for remote AS's to discover the available multi-AS-paths. BANANAS allows filtered re-advertisement capability of AS-paths through a neighbor, if the AS indeed forwards packets via through the particular neighbor AS. BANANAS allows a partial upgrade strategy of e-BGP routers alone, provided BGP synchronization semantics can be weakened, and tunneling of packets between e-BGP routers is possible. Unlike the intra-domain case, computation of multi-paths is trivial because paths are explicitly advertised in BGP. Forwarding from the entry-ASBR to the exit-ASBR can be either through tunneling or through special forwarding capabilities (using exit-ASBR field as destination) at all i-BGP routers.

Note also that while BANANAS allows "source" control over routing, this does not mean that end-systems see full routing tables. Instead, the BANANAS framework facilitates progressive decision making by nodes along the path that can take on the role of a "source" on behalf of the originating host (eg: source host, first-hop router, ABR, AS-BR in source AS, entry AS-BR in transit domains). Such sources would have the visibility into paths, and can make decisions on behalf of the original source within the level of abstraction in which they operate.

6. MULTI-PATH COMPUTATION AND TRAFFIC SPLITTING ALGORITHMS

In this section, we develop simple algorithms for multi-path computation in a partially upgraded network and explore simple heuristics for traffic splitting primarily for illustrative purposes. The framework is general enough to support heterogeneity and evolution of such algorithms and splitting heuristics without requiring major changes in the abstract forwarding plane operation.

6.1 Multi-Path Computation with Partial Upgrades

In this section we present a simple link-state algorithm 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. Narvaez et al [24] and Vutukury et al [30] propose multi-path algorithms applicable to distance vector or link state protocols, and may operate with partial upgrades. But, their discover only a (potentially small) subset of available paths that may depend upon the particular nodes upgraded.

Our link state algorithm (Algorithm 1) at upgraded node i uses the network map (graph) and first runs an all-pairs shortest path computation, i.e., the Floyd-Warshall Algorithm [7]. For any chosen node k and destination j, the Floyd-Warshall algorithm sets up the next-hop node l in the shortest path in node k's routing table. Given these routing tables, we do a depth-first search (DFS) rooted at node i to discover multiple paths from i to each destination j. We use a per-node variable *visited_nodes*, within each DFS pass, to mark the nodes *visited* by the DFS algorithm. By only picking nodes which have not been visited earlier to construct our paths, we ensure loop-free paths. If the DFS algorithm has arrived at node k (and appended k to relevant paths), it considers a subset of k's neighbors. If node k is known to be multi-path enabled, the DFS considers *all* its neighbors. Otherwise, it just considers the next-hop node on the shortest path from k to the destination. If the chosen next-hop node of k has not been *visited* earlier, its appends this node to the path, and repeats the above procedure recursively, using k's next-hop node as the source. Once the DFS is complete at node k, then the $visited\_nodes[k]$ is reset to zero. With minor extensions, Algorithm 1 can be used to obtain the *VirtualHopWeight* using a variable *flag* (initialized as true) for each path. The link weights are added to *VirtualHopWeight* if *flag* is set. The variable *flag* is reset if the next hop is a multi-path capable node and left unchanged otherwise.

The computational complexity of simple sequential Floyd-Warshall implementation is $O(N^3)$ where N is the number of nodes in the network. However, it has been shown that this shortest path problem can be viewed as a matrix multiplication problem that can be solved in $O(n^\omega), (\omega < 2.5)$ [26, 6]. The best known upper bound today is $O(n^{2.376})$ [7]. Alternatively, one may run Dijkstra *(N-k)* times where k is the number of multi-path capable nodes. The Dijkstra's algorithm with adjacency lists has complexity of $O(Elog(N))$, so varying over $N-k$ source nodes gives a complexity of $O((N-k)Elog(N))$. For multi-path nodes visit all the next hops in DFS, hence we do not need the shortest path next hop. This may be a better approach for large sparse graphs. In future work, we plan to investigate optimal and incremental multi-path algorithms under partial upgrade assumptions.

6.2 Traffic Splitting Strategies

In this section we present results to illustrate that even simple traffic splitting techniques at the *"source"* can lead to good load-balancing of traffic. Traffic splitting strategies could vary from simple heuristics to more complex optimal splitting (see for example, [13, 30]), There is no standard performance metric used in the literature for load-balancing. We use offered load to the heaviest loaded link in the network as the comparison metric. More elaborate metrics may

Algorithm 1 Algorithm for computing all paths between a source and destination with only some nodes supporting multi-path forwarding

```
/* adjacency_matrix[i][j]=link_weight if ∃ link from i to j,
else -1
partial_paths is the sum of link weights on the path, it is
initialized to zero
partial_paths_nexthop is the next-hop node on the path
no_paths denotes the number of path currently being tra-
versed, at the end of the procedure it will denote the num-
ber of paths found between a source and destination
N denotes the number of nodes in the network
The array visited_nodes marks a node if it has appeared
in the currently traversed path. It is initialized to zero */
procedure ComputePartialPaths(src, dst, no_paths, par-
tial_paths, partial_paths_nexthop, level)
begin
visited_nodes[src] ← 1
if src is a multi_path node then
  for i = 1 to N do
    save current value of partial_paths
    if (∃ link from src to i) && (visited_nodes[i]==0)
    then
      if level == 0 then
        partial_paths_nexthop[no_paths] ← i;
      end if
      partial_paths[no_paths]+=adjacency_matrix[src][i]
      if i==dst then
        partial_paths_nexthop[no_paths+1] ←
                        partial_paths_nexthop[no_paths]
        no_paths ++
      else
        ComputePartialPaths(i,dst,no_paths,
          partial_paths,partial_paths_nexthop,level+1)
      end if
    end if
  end for
else
  /* shortest_paths and shortest_paths_nexthop is com-
  puted by calling AllShortestPaths() */
  i ← shortest_paths_nexthop[src][dst]
  if visited_nodes[i]==0 then
    partial_paths[no_paths] += adjacency_matrix[src][i];
    if level == 0 then
      partial_paths_nexthop[no_paths] ← i;
    end if
    if i==dst then
      partial_paths_nexthop[no_paths+1] ←
                        partial_paths_nexthop[no_paths]
      no_paths ++
      ComputePartialPaths(i,dst,no_paths,partial_paths,
                partial_paths_nexthop, level+1)
    end if
  end if
end if
visited_nodes[src] ← 0
end
```

be designed to compare the performance of various splitting strategies. However, this is not the focus of this paper.

We compare the following traffic splitting strategies. It is important to note here that a global knowledge of the demand matrix is not assumed.

1. **Shortest Path:** (SP) All the traffic to a destination is sent on the shortest path to the destination.
2. **Proportional BW:** (Prop-BW) A subset of paths to a destination are computed and the minimum available capacity on each path is computed. The traffic is split proportional to the available capacity on the path. The paths longer than the maximum number of hops are not chosen.
3. **Independent-Maxflow**(I-Maxflow) The source picks an arbitrary path to the destination and sends the maximum it can on that path. This process is repeated until the entire traffic is assigned to paths. This splitting will give the fraction of traffic being routed on each of the paths. This is repeated independently for demand between every non-zero source-destination pair in the demand matrix.
4. **Sequential-Maxflow** (S-Maxflow) In this scheme, we assume that the available bandwidth after assigning one traffic demand is known before computing the split for the next demand. Maxflow is used to assign traffic to different paths.

We assume full-deployment of the BANANAS framework with enough memory to store all paths (or a subset of paths in case of Proportional BW) in the forwarding tables. We present results with two representative topologies shown in Figures 6, 7. All the links have 10Mbps bandwidth and 1ms delay. In this section, the shortest path computation assumes unit link weight for all the links.

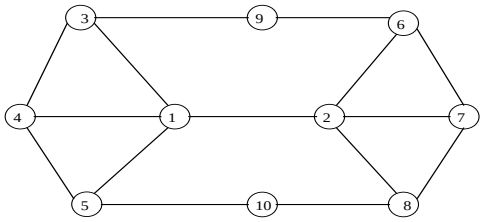

Figure 6: Topology 1: Simple 10-node network

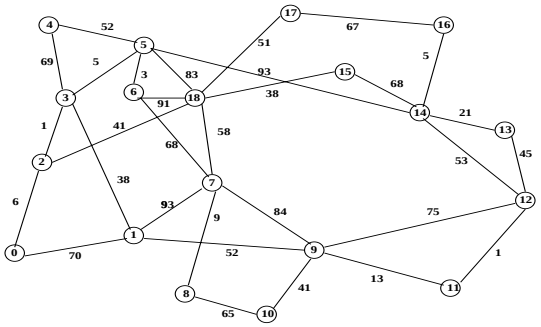

Figure 7: Topology 2: 19-node network topology

We compare the offered load on the maximum utilized link for different splitting schemes with randomly chosen demands. For Topology 1, four demand matrices with 12, 9, 5 and 7 randomly chosen source-destination pairs was constructed. Amongst these source-destination pairs, a total of 19, 20, 45 and 59Mbps traffic was sent respectively. Similarly, for Topology 2, demand matrices with 19, 15, 28 and 25 randomly chosen source destination pairs sending a total traffic of 31, 24, 36 and 45Mbps respectively were constructed. The traffic was assumed to be Constant Bit Rate (CBR). Comparative results are presented in Table 1 for Topology 1 and in Table 2 for Topology 2. Each row of the Table refers to a demand matrix (in the order described above).

D#	SP	$Prop-BW$	$I-Maxflow$	$S-Maxflow$
1.	0.90	0.34	0.90	0.90
2.	0.75	0.40	0.70	0.70
3.	1.80	0.87	2.70	1.00
4.	0.60	0.27	1.70	1.00

Table 1: Table comparing the offered load on the maximum utilized link for different traffic splitting schemes for Topology 1

D#	SP	$Prop-BW$	$I-Maxflow$	$S-Maxflow$
1.	0.50	0.27	1.20	1.00
2.	0.50	0.25	1.00	1.00
3.	0.80	0.40	1.60	1.00
4.	1.00	0.43	1.80	1.00

Table 2: Table comparing the offered load on maximum utilized link for different traffic splitting schemes for Topology 2

We expect the max-utilization for S-Maxflow to be 1.00 in most cases, if the traffic can be carried by the network. In this scheme, the maximum amount of traffic that can be sent on a path without overloading the path is actually sent on the path. However, as expected, if max-flow is done independently (I-Maxflow) without taking into account the actual residual bandwidth, some links will get overloaded quickly. We observe that the Proportional BW heuristic consistently outperforms the SP (shortest path with no traffic splitting) by balancing traffic evenly across the network (lower max-utilization). Although Tables 1,2 show results for only four representative demand matrices, these conclusions are consistent with the results that are not presented in this paper.

7. SIMULATIONS

In this section we illustrate the working of the framework using the Network Simulator (ns-2). The implementation allows network scenarios with partial upgrades to be simulated. Forwarding mechanism, routing tables and other details of the framework are also illustrated. Finally, some simple scenarios with both partial and full-upgrades are considered to illustrate possible gain in throughput by using the framework. The topologies described in Section 6.2 are used for the simulation (shown in Figures 7, 8).

Dest	*PathID*	*NextHop*	*VirtualHopWeight*
8	121	2	83
8	165	5	165
7	206	5	203

Table 3: Partial routing table at Node 1 for simulation run on Topology 1

7.1 Illustration of the framework

Consider the topology shown in Figure 8 (Topology 1). All links have 10Mbps bandwidth and 1ms delay with link weights as shown in the figure. To explain the working of the framework, we consider the paths taken by a packet from Node 1 to Node 8 (*Path-I*) and Node 1 to Node 7 (*Path-II*).

We consider a partial upgrade scenario where only nodes 1 and 2 (shown by dark circles) are multi-path capable. A subset of the entries in the routing table at Node 1, during a simulation run is presented in Table 3. The node computes the shortest path to a destination along with other available paths. Note that the table is indexed by two keys, namely, the destination address and the PathID. When a

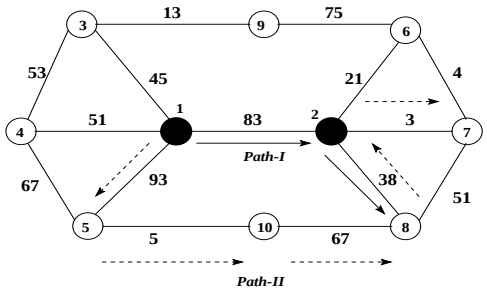

Figure 8: Figure demonstrating operation of BANANAS for Topology 1

packet arrives at node 1 with a valid PathID, the next hop is obtained from the table using the destination address and the PathID. The PathID on the packet is then decremented by the distance till the next multi-path node (indicated by *VirtualHopWeight*), before the packet is transmitted. So if a packet enters Node 1 with PathID 121 and intended for destination 8, it is forwarded to 2 with the PathID set to $(121 - 83) = 38$. Note that in this case (indicated as *Path-I* in the Figure 8) the next hop (node 2) is also a multi-path node.

Consider *Path-II* in Figure 8. Suppose a packet arrives at Node 1 with the PathID 206 (93+5+67+38+3) and destined to Node 7. The next hop is 5, a node that does not implement the multi-path algorithm. Node 1 subtracts the VirtualHopWeight (in this case 203=93+5+67+38). The packet reaches node 2 with a PathID 3 and is appropriately forwarded.

7.2 Throughput Gains with traffic splitting

With multiple paths available for routing traffic, a "source" can split traffic among those paths to achieve load balancing and gains in throughput. We use topologies shown in Figures 7, 8. The simulation consists of transmitting a 1 MB file between two nodes using CBR sources. These simulations assume the knowledge of residual capacity of a path.

Topology	*Shortest Path*	*Two Paths*	*Three Paths*
Topology 1	5.25	3.15	3.15
Topology 2	6.99	3.15	1.85

Table 4: Using multiple paths to obtain higher end-to-end bandwidth: Time taken (in seconds) for transfer of 1MB file

In an OSPF domain, this information can be obtained using opaque LSAs.

In Figure 8, Node 1 is the source and Node 8 is the destination. The nodes 1 and 2 are multi-path enabled. The shortest path from Node 1 to Node 2 is the one connecting nodes 1, 2 and 8, and is denoted as (1,2,8). Alternatives to the shortest path are (1,5,10,8) and (1,4,5,10,8). The path (5,10,8) has a constant background traffic of 4Mbps. The path (4,1,2,7) has a constant background traffic of 2Mbps. Thus the residual capacity on the paths would be 8Mbps on (1,2,8), 6Mbps on (1,4,5,10,8) and (1,5,10,8). Using the proportional BW heuristic discussed in Section 6.2, divide the traffic among the paths proportional to the residual capacities.The ratio for the traffic split would then be 8:3:3. Table 4 depicts the time taken to complete transmission of the file. Since the alternate paths (1,4,5,10,8) and (1,5,10,8), share the bottleneck link (10,8), the time to complete transmission with two and three paths is the same (3.15s). Thus increased number of paths may not necessarily mean an improvement in throughput.

For Topology 2 (10Mbps, 1ms links and link weights as shown in Figure 7), the source node is 18 and destination is 12. The multi-path nodes are 7 and 18. The shortest path is (18,15,14). Two alternate paths are (18,2,3,5,14) and (18,17,16,14). Note that the paths are independent. There is a constant background traffic of 2Mbps on paths (0,2,3,5), (15,14,12) and (15,14,13). Consequently the residual capacities on the paths are 10Mbps on (18,17,16,14), 8Mbps on (18,2,3,5,14) and 6Mbps on (18,15,14). Using the same heuristic as above to split traffic, we see the results in Table 4. Thus exploring alternative paths can yield higher throughput if residual capacities are higher on them.

8. RELATED WORK

There has been much recent interest in the area of connectionless traffic engineering intended for scenarios ranging from intra-domain, inter-domain, end-to-end, and overlay operation [31, 18, 8, 17, 10, 1].

In the intra-domain case, a large body of work centers around using current shortest path routing (OSPF) as the basis, and then achieving optimal routing by either managing link metrics [31, 11, 12, 21], using equal-cost multi-path extensions with static or dynamic local traffic splitting [15, 29], or extending intra-domain routing algorithms for multi-path support [24, 30, 5]. The problem of finding optimal OSPF link weights for a given traffic demand is an NP-hard problem [11, 3]. Moreover, traffic demand must be assumed to be characterized in a quasi-static manner, and there is an overhead of changing link-weights. Fortz and Thorup [12] use local-search algorithms, and optimize OSPF by changing weights of only few links. Wang et al [31] convert an

optimal routing problem in an overlay routing model (using VCs mapped to physical network) to a shortest path problem with appropriate link weights set to reflect the traffic demand. OSPF-ECMP [28, 15] allows traffic to be split equally among the multiple next hops for paths with equal weights to the destination. Weights have to be carefully engineered to achieve load-balancing. OSPF-OMP [29] uses ECMP, but instead of depending upon weight assignments, sample traffic load information and floods it via opaque LSAs. The information is used to tweak local load splitting decisions.

The performance of OSPF is limited by the underlying shortest path forwarding (or ECMP). Lorenz et al [21] show that OSPF routing performance can be as bad as $O(N)$ compared to the explicit source-based routing (eg: MPLS-based [3]). In [3, 11, 21] network topology and demand matrices have been constructed for which the best OSPF/IS-IS routing is worse than the best MPLS routing. Multi-path routing does not have this limitation.

The SNA protocol of IBM had one of the earliest multi-path routing algorithms [14]. Narvaez et al [24] and Vutukury et al [30] propose simple multi-path algorithms that can operate in DV or LS environments, but do not compute all possible paths. Chen et al[5] propose an interesting framework for multi-path forwarding and propose multi-path extensions to LS and DV routing. They develop a general concept of suffix matched path ID, that is a mathematical generalization of the *i-PathID* of the BANANAS framework. However, they propose a label-switched realization (hard to map to BGP), and expect *all* network routers to support multi-path forwarding. Most of these references do not consider the issue of source-based control over traffic splitting; and expect fully upgraded network. All these authors only consider a single, flat routing domain. These factors differentiate the BANANAS framework.

MPLS and ATM are the key protocols in the signaled traffic engineering protocols. MPLS-TE [3] offers signaled explicit label switched paths (LSPs) which can be set up using an arbitrary control algorithm. Traffic trunks can then be instantiated and mapped onto LSPs. In particular, a constraint-based routing problem can be defined (and solved with heuristics) where LSPs are set up to meet constraints in terms of both the traffic demands and the resources available. OSPF traffic engineering extensions [20] can be used to collect the information needed to setup this problem. Variations of the max-flow technique [19] or adaptive traffic splitting [9] can be used to map traffic onto the LSPs.

In the inter-domain area, the IRTF is considering requirements documents for a future inter-domain protocol, and traffic engineering figures in both the key proposals [8, 17] as an important problem. Inter-domain TE work has revolved around multi-homed AS's, in-bound/out-bound load-balancing between adjacent AS's using BGP [17], provider-selection and multi-homing issues considered with IPv6 development [25, 16], or map-distribution based approaches (NIMROD) [4].

Of these, only the NIMROD and IPv6/GSE proposals are comparable to BANANAS. NIMROD [4] is a hierarchical, map distribution-based architecture which allows explicit source-based path choice, through a signaled or datagram (i.e. IP source-route) method. NIMROD is incompatible with current BGP routing, does not have an equivalent of a PathID for forwarding, and may require signaling to setup explicit paths. IPv6 provides a routing option for provider selection, and elegant auto-configuration methods which easily accommodate site multi-homing. The provider-selection option was intended to allow choice of providers (potentially remote) in the forwarding path; which is not the same as a full route encoding. O'Dell's 8+8/GSE [25] proposes to break the IPv6 address into a locator (routing group) and a end-system designator (ESD) part. The locator could change after crossing autonomous systems (eg: providers). This would facilitate multi-homing, change of providers and be a vehicle for TE capabilities like provider-selection, path-selection. In contrast, BANANAS chooses not to encode the PathID into the address field and have a separate field which is interpreted along with the address field.

9. SUMMARY AND FUTURE WORK

This paper focuses on the BANANAS-TE framework and mapping aspects to contemporary routing protocols. The core PathID notion of encoding a path as either a sum of link-weights or a sum of ASNs (i.e. node IDs) is extremely simple and easy to understand. Moreover, PathID is especially attractive for incremental upgrades because PathID is a well-known *global* encoding for a network path in the connectionless routing model used by both BGP and OSPF. The nearest-PathID-match technique uses the global notion of PathID and extends the traditional IP longest-prefix match forwarding paradigm. This feature contrasts with the label-swapping technique that is based upon the notion of local IDs (labels), and requires either a signaling protocol or the distribution of labels for global consistency. A simple multi-path algorithm developed using Floyd-Warshall and DFS techniques allows the computation of all available multi-paths in partial upgraded networks. The BANANAS framework allows heterogeneous multi-path computation and forwarding capabilities at the upgraded nodes. A viable strategy for mapping BANANAS to hierarchical OSPF and BGP is discussed with several examples. It is also possible to map the framework (with some limitations) to distance vector protocols like RIP and EIGRP. Simulations and simple traffic splitting algorithms illustrate the potential of the framework. We believe BANANAS is an attractive framework to migrate today's Internet routing to ultimately support multi-domain TE capabilities.

Development of optimal algorithms (or heuristics) for multi-path computation, source-based path discovery and traffic splitting etc. is not the focus of this paper. In this paper, we do not consider incremental or asymptotically optimal multi-path computation, adaptive path discovery, path performance probing and load-splitting. We believe that the BANANAS framework is flexible enough to allow heterogeneity and evolution of these capabilities. We plan to work on such algorithms in future.

A key assumption in Section 2.1, is the uniqueness of the PathID field. The BANANAS framework is safe (i.e no loops) in the case of forwarding tuple collision, and pre-

scribes some diversity in assignment of link-weights to reduce tuple collision probability. The exact analysis of tuple-collision probability given the length of the PathID field and characteristics of the network graph is an interesting future problem. Finally, the introduction of traffic engineering capabilities in general leads to increase in dynamic complexity of routing (both in terms of control traffic and data traffic), which is another interesting topic for future study.

10. REFERENCES

[1] D.G. Andersen et al, "Resilient Overlay Networks," *ACM SOSP*, October 2001.

[2] D. Awaduche et al, "Overview and Principles of Internet Traffic Engineering," *IETF Internet Draft* draft-ietf-tewg-principles-02.txt, Work-in-progress, Jan 2002.

[3] D. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communications Magazine*, Vol. 37, No. 12, pp. 42-47, 1999.

[4] I. Castineyra, N. Chiappa, M. Steenstrup, "The Nimrod Routing Architecture," *IETF RFC 1992,* August 1996.

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

[6] D. Coppersmith and and S. Winograd, " Matrix Multiplication via Arithmetic Progression," *ACM Symp. on Theory of Computing (STOC)*, 1987, pp.1-6.

[7] T.H. Cormen, et. al., "Introduction to Algorithms", *The MIT Press, McGraw-Hill Book Company*, Second Edition, 2001.

[8] E. Davies et al, "Future Domain Routing Requirements," *IRTF Routing Research Draft* draft-davies-fdr-reqs-01.txt, Work-in-progress, July 2001.

[9] A. Elwalid, et al, "MATE: MPLS Adaptive Traffic Engineering," *INFOCOM'01*, April, 2001

[10] End-to-End Mailing List Thread, "Traffic engineering considered harmful," June 2001.

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

[12] B. Fortz, M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE JSAC*, to appear.

[13] R. G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Transactions on Communications*, Vol COM-25, No. 1, January 1977. pp. 73-85.

[14] J. Gray, T. McNeill, "SNA multiple-system networking," *IBM systems Journal*, Vol. 18, No. 2, 1979.

[15] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," *IETF RFC 2992*, 2000.

[16] C. Huitema, "IPv6: The New Internet Protocol," *Prentice Hall*, Second Edition, 1998.

[17] G.Huston, "Commentary on Inter-Domain Routing in the Internet," *IRTF Routing Research Draft* draft-iab-bgparch-02, Work-in-progress, Sept 2001.

[18] V. Jacobson, SIGCOMM 2001 Keynote Address.

[19] K. Kar, M. Kodialam, T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE JSAC*, Vol. 18, No. 12, December 2000.

[20] D. Katz, J. Cheung, K. Kompella, "Traffic Engineering Extensions to OSPF," *Internet draft* draft-katz-yeung-ospf-traffic-06.txt, Work-in-progress, Jan 2002.

[21] D.H. Lorenz, A.Orda, D.Raz, Y.Shavitt, "How good can IP routing be?," *DIMACS Technical Report 2001-17*, May 2001.

[22] J. Luciani et al "NMBA Next Hop Resolution Protocol (NHRP)", *IETF RFC 2332*, April 1998.

[23] J. Moy, "OSPF Version 2," *IETF RFC 2328*, April 1998

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

[25] M. O'Dell, "GSE – an alternate addressing architecture for IPv6," *Expired Internet Draft*, 1997.

[26] F. Roman, "Shortest Path Problem is Not Harder than Matrix Multiplication," *Information Processing Letters*, Vol. 11, 1980, 134-136.

[27] E. Rosen et al, "Multiprotocol Label Switching Architecture," *IETF RFC 3031*, January 2001

[28] D. Thaler, C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," *IETF RFC 2991*, 2000.

[29] C. Villamizar, "OSPF Optimized Multipath (OSPF-OMP)," *Expired Internet Draft*, 1999. Available: http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-ospf-omp-02.txt

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

[31] Z. Wang, Y. Wang, L. Zhang, "Internet Traffic Engineering without Full Mesh Overlaying," *INFOCOM'01*, April 2001.