BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet

Hema Tahilramani Kaur, Shivkumar Kalyanaraman, Andreas Weiss, Shifalika Kanwar, Ayesha Gandhi

***Abstract*—**

Today the Internet offers a single path between end-systems even though it intrinsically has a large multiplicity of paths. This paper proposes an evolutionary architectural framework "BANANAS" aimed at simplifying the introduction of multipath routing in the Internet. The framework starts with the observation that a path can be encoded as a short hash ("PathID") of a sequence of globally known identifiers. The PathID therefore has global significance (unlike MPLS or ATM labels). This property allows multipath capable nodes to *autonomously* compute PathIDs in a partially upgraded network without requiring an explicit signaling protocol for path setup. We show that this framework allows the introduction of sophisticated explicit routing and multipath capabilities within the context of widely deployed connectionless routing protocols (e.g. OSPF, IS-IS, BGP) or overlay networks. We establish these characteristics through the development of PathID encoding and route-computation schemes. The BANANAS framework also allows considerable flexibility in terms of architectural function placement and complexity management. To illustrate this feature, we develop an efficient variable-length hashing scheme that moves control-plane complexity and state overheads to network edges, allowing a very simple interior node design. All the schemes have been evaluated using both sizable SSFNet simulations and Linux/Zebra implementation evaluated on Utah's Emulab testbed facility.

I. Introduction

Today's Internet routing protocols like OSPF and BGP were designed to provide one primary end-to-end service: "best effort reachability." These protocols realize the "best-effort" concept by offering a single-path to destination subnets. Our work is motivated by the fact that the notion of "best-effort" does not rule out the incorporation of end-to-end path multiplicity at the routing level. Indeed the Internet topology has an intrinsic multiplicity of paths: hosts have multiple potential network interfaces and autonomous systems (both enterprises and ISPs of various sizes) are multi-homed [1], [2], [3], [4]. It is interesting to ponder: *why have we not significantly exploited this intrinsic path multiplicity in the Internet ?* The answer to this question is clearly *not* the lack of algorithms and protocols. Indeed there have been several proposals for multipath route-computation [5], [6], [7], [8], Internet signaling architectures [9], [10], [11], [12], [13], novel overlay routing methods [14], [15] and transport-level approaches [16], [17]. The fact that these developments have not triggered widespread deployment suggests that the core problem is an architectural one [1]. The Internet lacks an evolutionary framework that admits incremental deployment of path multiplicity, while providing sufficient flexibility in terms of architectural function-placement and management of complexity. This paper proposes to fill that void with a framework called "BANANAS" [2].

At the highest level, BANANAS proposes a simple extension of Internet operation to admit and leverage end-to-end path-multiplicity (PM). In this model, source-hosts initiate one or more end-to-end "flows" and map flows to local network interfaces. The "network" provides one or more end-to-end paths through the independent upgrades of a *subset* of network nodes, possibly situated in multiple administrative domains. A subset of these upgraded nodes (e.g. selected edge-nodes) may also map "flows" to available "paths" [3]. Source-hosts may arbitrarily map "packets" to "flows." Observe that today's single-path model is a special case of this PM-model. The PM model also allows a subset of source-hosts and routers to be independently upgraded within the scope of usual administrative boundaries. Upgraded node may "see" only a subset of available paths within appropriate administrative boundaries. This high-level model is a *best-effort path multiplicity* model, clearly different from IPv4/IPv6 connectionless loose-source-routing model [18], [19] and from end-to-end signaled source-route models used in ATM networks (e.g. PNNI [20]) or MPLS networks [21].

BANANAS provides a set of concepts and building blocks to realize this high-level PM model. A core abstract idea in BANANAS is that a path can be efficiently encoded as a short hash (called the "PathID") of a sequence of globally-known identifiers (e.g. router IDs, link interface IDs, link weights, AS numbers etc.). This concept has some very important advantages. First, a hash-based data-plane encoding is more efficient than IPv4/IPv6's loose-source-routing encoding [18], [19] that is an uncompressed string of IP addresses. Second, since the PathID is a function of globally-known quantities, it inherits their global significance, i.e., it can be computed and

[1]Another key problem involves incentives; but incentives depend upon attributes of the underlying architectural framework.

[2]BANANAS is not an acronymn! It is adapted from the car racing comedy movie title *Herbie goes Bananas*

[3]Eg: Packets from TCP connections would be mapped single "path" to avoid out-of-order packets

interpreted within the same scope of visibility. This "global" scope may refer to a single routing domain if router/link IDs are involved; or may refer to the universe of BGP-4 routers if AS numbers are used. The global PathID semantics allows any upgraded multipath capable (MPC) node to autonomously compute the PathID without any changes in legacy single-path capable nodes. It also removes the need for an explicit out-of-band signaling protocol as a path-setup mechanism. Note that one purpose of signaling in ATM and MPLS is to map global IDs (global addresses, path specifications) to *locally* assigned IDs (labels). The global PathID semantics allow the mapping of BANANAS in an *incremental* manner to *connectionless* Internet routing protocols (e.g. OSPF, BGP-4).

In addition, the BANANAS framework allows considerable flexibility in terms of architectural function placement and complexity management. These intangible aspects are crucial for tailoring the proposed building blocks and establishing the appropriate incentives for adoption by vendors and ISPs. For example, the framework allows considerable flexibility in the choice of multipath route-computation algorithms. It also provides a distributed validation procedure to ensure the validity of computed PathIDs, i.e. to check if forwarding exists in all downstream routers for the PathIDs. As another example of architectural flexibility, we propose an efficient variable-length hash realization of the abstract framework: this scheme moves control-plane complexity and state overheads to network *edges*, allowing a very simple interior node design. The proposed scheme realizations are evaluated using integrated OSPF/BGP simulations in sizable topologies and Linux/Zebra implementation run on Utah's Emulab emulation testbed facility.

The BANANAS framework is not just restricted to legacy OSPF or BGP networks. It can be applied to exciting new contexts such as overlay networks, peer-to-peer networks (e.g. dynamically instantiated overlays using a peer-to-peer lookup infrastructure to support video-conferencing) and last-mile multi-hop fixed-wireless networks. We are currently initiating the deployment of the BANANAS framework on the PlanetLab infrastructure [22] as an public experimental wide-area network overlay service. We are also building a medium-sized multi-hop 802.11 community wireless network on which this framework will be deployed. We believe that the mere *expectation* of multiple end-to-end paths will trigger application innovation in areas such as bandwidth aggregation [17], resilience [14], [15], [23] and security strategies (e.g. protecting data integrity using multipaths).

The rest of the paper is organized as follows. Section II develops the abstract framework and concepts. Section III explores the architectural flexibility in BANANAS by considering an alternate index-based PathID encoding. Section IV summarizes the intra-domain routing extensions for link-state protocols, OSPF and IS-IS. Section V develops the inter-domain ideas of BANANAS in the context of BGP-4. Section VI presents both simulation and linux-based implementation results to illustrate the architectural features of BANANAS. Related work is surveyed in Section VII, followed by summary and concluding remarks in Section VIII.

II. The BANANAS Framework

A. PathID: Abstract Concept

Consider a network modeled 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. Let N denote the number of nodes in the network, i.e. the cardinality of the set V. Each link $(i, j) \in E$ has an identifier associated with it, denoted by $l_{i,j}$. Each node i also has an identifier denoted by n_i. Consider a path $P_{i,j}$ from node i to node j, which passes through nodes $i, 1, 2, ..., m-1, j$. This path can be represented as a sequence of globally-known node and link identifiers $[n_i, l_{i,1}, n_1, l_{1,2}, n_2, ..., l_{m-1,j}, n_j]$. This path sequence can be compactly represented by a *hash* of its elements. A path identifier (or, in short "PathID") is defined as a hash of the *above sequence or any non-null subsequence* derived from it. Observe that the IP destination address (j), the uncompressed IPv4/v6 loose-source-routes [18], [19], the XOR of router IDs proposed in LIRA [11], or a hash of the subsequence of link weights are all examples of valid PathIDs, obviously with differing characteristics. Therefore the particular subsequence and PathID encoding function chosen is crucial in determining the utility of the PathID. These abstract concepts are illustrated in Figure 1.

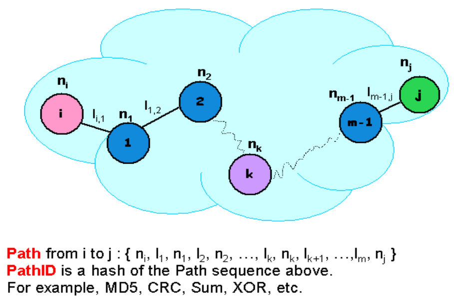


Fig. 1. Path and PathID Concepts

A desirable hash is compact, easy to compute and has a low collision probability (i.e. high uniqueness probability). This demands a *hash function* that offers low collision probabilities. A simple hash of the path sequence may be obtained by using the sum or XOR function (suggested in LIRA [11]). While these are simple and fast, it may lead to non-unique PathIDs. Our canonical hash function choice is a 128-bit MD5 hash followed by a 32-bit CRC of the 128 bit MD5 hash (resulting in a final 32-bit hash value). We use the notation (MD5 + CRC32) hash to represent the above two-step hashing process. This value is used in conjunction with the destination address (j) available anyway in IP packets, leading to a two-tuple hash: [j, PathID][4]. The collision probability now depends only on the number of paths to any given destination prefix, and the nature of the path subsequence on which the MD5+CRC32 function is applied. Assuming a random bit-string and all the 2^{32} outputs to be equally likely, the probability for collision for k paths can be approximated by $\frac{k}{2^{32}}$, which is adequately small for our purposes.

For the underlying *path sequence*, we could use a sequence of well-known link interface IDs, router IDs and link weights (in OSPF or IS-IS) on the path. However, link-weights are usually non-unique, chosen from a narrow range and may be dynamic (to implement traffic engineering/ traffic-sensitive routing), whereas router IDs and link interface IDs are unique identifiers. Our canonical choice is the subsequence of all node IDs on the path (generalizes to a sequence of AS numbers in BGP-4). Section III develops an alternative hash function that is a concatenation of well-known link ID indices at nodes.

B. Packet Forwarding

This section describes the forwarding table structure and forwarding algorithm corresponding to our canonical choice of hash function and path subsequence made in Section II-A. Section III develops an alternative forwarding algorithm (for OSPF/IS-IS) that does not require a large forwarding table at interior nodes.

IP forwarding tables essentially contain two-tuple entries of the form **[destination prefix, outgoing interface]**. A longest-prefix-match lookup procedure is employed. At upgraded routers we propose to use four-tuple entries of the form **[destination prefix, incoming PathID, outgoing interface, outgoing PathID]**. The "incoming PathID" field represents the hash of the explicit path from the current router to the destination prefix. The "outgoing PathID" field is the hash of the corresponding path suffix from the *next upgraded router* to the destination.

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. If an exact match is not found (i.e. errant hash value in packet), then the hash value in the packet is set to zero, and the packet is sent on the default path (i.e. shortest path in OSPF/IS-IS or default policy route in BGP-4). The hash value may also be set to zero if the next-hop is the destination itself, or there are no upgraded routers in the path specified by the incoming PathID. A non-upgraded router simply ignores the PathID field and forwards the packet on the shortest path. Observe that this procedure is a *hybrid* of IP longest prefix match and label swapping used in MPLS (or ATM) networks. A key difference is our use of globally known PathIDs instead of locally meaningful MPLS labels. The global PathIDs may be computed at each router with minor modifications to OSPF LSAs (See Section IV), avoiding an explicit out-of-band signaling protocol.

[4]We will continue to refer to the second tuple value as the PathID for convenience.

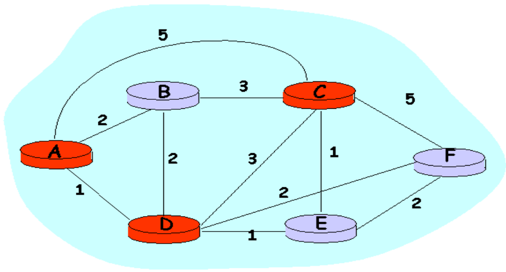


Fig. 2. Multi-Path Forwarding with Partial Upgrades

Figure 2 shows a partially upgraded network. Nodes A, C and D are multipath capable (MPC). Assume 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 etc 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.

C. Path and PathID Computation

The BANANAS framework not only supports upgrades of a subset of nodes, but also allows heterogeneity in multipath computation algorithms used at different upgraded routers. The fundamental tradeoff in link-state protocols (given our canonical choice of PathID hashing method) is route-computation and space complexity incurred at upgraded routers to avoid signaling.

In link-state protocols each router has a complete map of the network in the form of link-state database. We propose to first annotate this "map" at an upgraded node with the knowledge of other upgraded nodes. In Figure 2, upgraded node A will know that nodes C and D are upgraded and vice versa. We propose that upgraded multi-path capable (MPC) nodes set a bit (a.k.a. MPC-bit) in their link state advertisements (LSAs) to inform other upgraded nodes. Non-upgraded routers are oblivious to this bit and ignore it.

Using the link-state database ("map") and knowledge of upgraded routers, every router can locally compute available network paths. The simplest model that admits the largest number of paths is where each upgraded router can forward to any neighbor. The paths can be computed as follows. Perform a Floyd-Warshall [24] (all-pairs shortest path) to identify next-hops to destinations at non-upgraded nodes. Perform a depth-first-search (DFS) [24] that traverses every neighbor of upgraded nodes and the shortest-path neighbor at non-upgraded nodes. This results in a table containing next-hops for all paths to a destination under the constraint of a known subset of MPC nodes. We refer to this strategy as DFS under partial upgrade constraints or DFS-PU for shorthand. This simple approach is obviously expensive in both computational and storage terms, especially as the number of MPC nodes grows.

The BANANAS framework allows an upgraded router to compute and store only a *valid subset of available paths* under partial constraints. Indeed, we can choose from a wide variety of multipath computation algorithms available in the literature, for example k-shortest-paths, all k-hop paths, k-disjoint paths (see [5] and references within), DFS with constrained depth ([7] uses a depth-constraint of 1-hop) etc. The only constraint is that the algorithm should also compute the shortest (default) path. These algorithms may be adapted for the MPC constraint, i.e. there is a known subset of upgraded nodes.

However, there is a second, more subtle problem: if different routers compute and store different sets of paths, it is possible that the path computed by one upgraded node may not be supported by another upgraded or non-upgraded node that lies downstream on this path. We term such paths as *"invalid"*, i.e., forwarding support for the path does not exist at some downstream node.

To solve the above problem, we propose a *distributed validation algorithm* that ensures validity of chosen paths. The main idea behind the validation algorithm is that a path is valid (i.e. forwarding for a path exists) if all its path suffixes are valid. This suggests a mathematical induction style algorithm approach. We know that all one-hop paths are always valid because they represents a direct link. A two-hop path is valid if its one-hop path *suffix* is valid.

The proposed algorithm (see Algorithm 1) has two phases. In the first phase a node computes the paths using the chosen algorithm. For example, let us assume that node i uses a k_i-shortest-path algorithm. The k_i paths computed to each destination are input into a map data structure [25] that is ordered by hop-count. The node may start using these paths (in its forwarding table) with the understanding that some of them may be invalidated in phase 2. To prepare for phase 2, the node also needs to know the path computation algorithm and parameters used by other upgraded nodes. In our example, node i needs to know the k_j parameter associated with each upgraded node j. With this knowledge, it can compute the k_j paths for node j and input it into the hop-count ordered map data-structure (lines 2-5 in Algorithm 1) . At non-upgraded nodes, k_j is 1 (lines 6-9 in Algorithm 1). Essentially we have computed all potentially *available* paths in phase 1.

Phase 2 operates similar to mathematical induction: All one-hop paths in the map are declared as valid. For each 2-hop path, the algorithm simply searches for the 1-hop path suffix in the just-validated set. If a match is not found, the path is invalid and is discarded. If the path (i.e. the corresponding PathID entry) exists in the forwarding table, it is removed. In this process, validating an m-hop path entry implies looking up its (m-1)-hop path suffix in the just-validated set of (m-1)-hop paths and finding a match (the variable temp_pair and the lines 16,17 in Algorithm 1 are used to find a suffix match in the Routing_Map structure). By mathematical induction, when the entire map has been linearly traversed, the remaining paths are valid.

The computational complexity of this approach can be estimated as follows. In a N-node network with u upgraded routers, the complexity of first phase is given $uC(k) + (N - u)C(1)$ where, $C(k)$ denotes the complexity of computing k-shortest paths, $C(1)$ denotes the complexity of Dijkstra's algorithm. The total number of paths, T, computed at the end of first phase is equal to $(N-1)((N-u) + \sum_{i=1}^{i=u} k_i)$. The complexity of the validation phase is $O(Tlog(T)\bar{h})$ where, $\bar{h}$ is the average hop count for the paths. The log(T) term arises due to searching for a suffix in the Map (see Algorithm 1, line 18). The validation algorithm may be optimized or be eliminated for special cases: e.g. if all nodes are upgraded and use the same value of k.

In summary, Algorithm 1 is a general 2-phase validation procedure that can be applied to validate paths computed using *any* deterministic path computation algorithm at MPC routers that also computes the default shortest path.

III. Architectural Flexibility in BANANAS

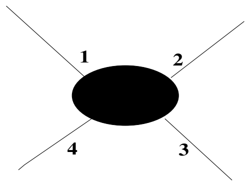
A general concern with the canonical description so far is the increase in complexity at upgraded nodes (both edge and core nodes). An interesting question is whether we can use an alternative hashing method that leads to overall complexity reduction and a more attractive division of functions between the edge and core, and between data-plane and control-plane. To demonstrate the affirmative answer, we develop a new *index-based encoding scheme* that moves complexity to network edges, and simplifies core node operations. The tradeoff is to use a variable-length PathID encoding instead of the canonical

Algorithm 1 Algorithm for validating paths at a router in a partially upgraded network

```
Let NU and U denote the set of all non-upgraded and upgraded
  nodes respectively
for all u ∈ U do
  newPaths ← Compute paths using u's advertised algorithm
  Routing_Map.append(newPaths)
end for
for all n ∈ NU do
  newPaths ← Compute shortest path using Dijkstra's algorithm
  Routing_Map.append(newPaths)
end for
All 1-hop paths are valid
Initialize suffixLength ← 2
while suffixLength < maxHops do
  for all path ∈ Routing_Map do
    if hop count of path ≥ suffixLength then
      temp_pair.hopcount ← suffixLength-1;
      temp_pair.PathString ← last suffixLength nodes in path;
      if Routing_Map.find(temp_pair) == FALSE then
        delete path
      end if
    end if
  end for
  suffixLength++;
end while
```

32-bit fixed length encoding. Also the scheme is only applicable to link-state protocols, where the neighbor relationships do not change often. Specifically, the index-based scheme is *not* applicable to path-vector based protocols like BGP-4, or mobile ad-hoc networks where neighbor relationships change rapidly.

A. Index-based Scheme: PathID Encoding



IP Address	Index
128.12.25.2	1
128.12.25.3	2
128.12.25.4	3
128.12.25.5	4

Fig. 3. Example of How Interfaces are Indexed

To motivate the scheme, consider an example. An upgraded node orders its link interface IDs and represents each link by its index in this ordering (see Figure 3). The new link ID (i.e. index) can now be efficiently encoded. For example, a router with 15 interfaces will need 4-bit link indices. In general, the link or interface IDs of a node may be *locally hashed using a globally-known hash function*. Since every node knows the global hash function and it operates on globally-known link IDs (e.g. IP addresses of interfaces) each node can independently compute the hashes of any other node and hence avoid signaling.

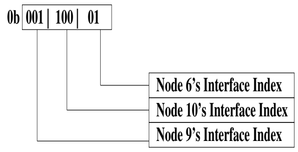


Fig. 4. Explanation of the Index-Based Encoding

A path can now be specified as a concatenation of such link-indices (see Figure 4). Moreover, this PathID encoding is guaranteed to be unique (unlike the earlier MD5+CRC32 encoding which had a very small residual non-uniqueness probability). For a reasonable maximum bit-budget in the packet header (e.g. 128 bits), and an average of 15 interfaces per router, up to 32-hop paths can be encoded with this technique. The limitation of 32-hops is not too restrictive (in [26], authors find that the average number of hops to reach a destination in the Internet is 19); it applies only within a single area or a domain. The PathID is re-initialized by the first upgraded router after crossing any area or domain boundary. The concatenation operation used here is an example of a *reversible* or perfect hash, i.e., the local hash (i.e. next-hop information) can be extracted from the overall PathID without needing a per-path table entry. The state needed at interior nodes is a small; only a table mapping link indices to link-IDs is needed. For example, at a router with 15 interfaces, a 15 entry index-table is needed irrespective of network size. No other control-plane computation or state-complexity is required at interior nodes. Since the interior nodes can forward to any neighbor now, a large number of network paths may be supported. Edge-nodes can compute paths using heterogeneous algorithms, and use a simpler validation algorithm (see Section III-C).

To summarize the impact in terms of function placement and complexity management, the index-based scheme uses Per-hop *PathID processing* in the data-plane instead of a table-driven per-hop *PathID swapping* strategy. Only edge routers need to compute the multipaths and their PathIDs using a simplified validation procedure.

B. Index-Based Scheme: Packet Forwarding

Upgraded interior routers maintain an index table that maps the interface index to the link interface IP address. On receiving a packet, an upgraded interior router extracts the interface index of the outgoing interface (next-hop) from the PathID field in the packet header and uses the interface index table to forward the packet on the appropriate link (see Figure 5).

Figure 6 shows a packet being sent from node S to node 7 along the path S-6-2-4-3-7, the PathID at various points and various interface indices. Only nodes S, 6 and 4 are upgraded.

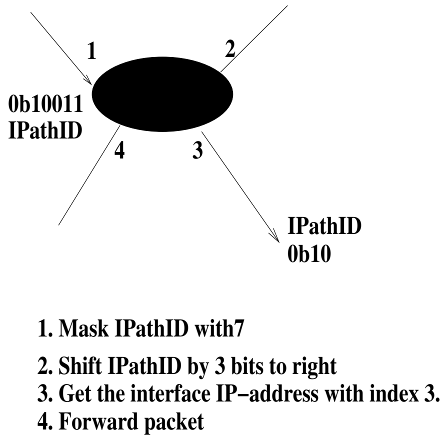


Fig. 5. Index-Based Forwarding At An Upgraded Router

Node S has complete map of the network from the link-state database and knows that node 6 has two interfaces and the next-hop index at node 6 is 2, encoded using two-bits. Note that the interface indexing starts from 1 because PathID of zero still refers to the default (shortest) path. Likewise, the index at node 4 for this path is 3, encoded using three bits. The PathID of the packet sent from node S is $0...01110_2$ = 14, indicating an index ($10_2 = 2$ for node 6 and $011_2 = 3$ for node 4). Node 6 has an index table with 2 entries mapping the link indices to the interface IP addresses. On receiving a packet with PathID in the routing header, it extracts the last two bits and then looks up its index table. The PathID is also right-shifted by two bits in this operation so that the next upgraded router can extract its index from the last bits of the PathID. Similarly, node 4 will extract three bits from the PathID and right shifts it by the same number before forwarding it. The remaining PathID will now be zero. The non-upgraded routers merely forward packets along the default shortest paths, oblivious of the PathID field.

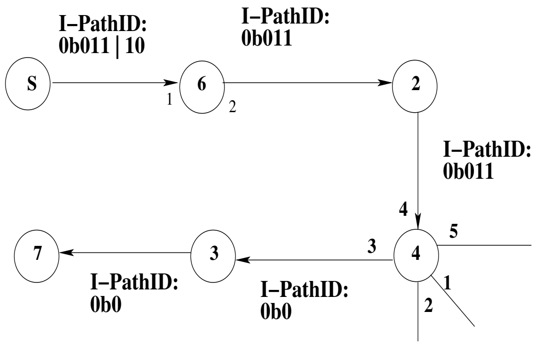


Fig. 6. Forwarding With the Index-based PathID Encoding Scheme (Note: "0b" indicates binary encoding)

C. *Index-based Scheme: Path Computation*

In this scheme "source" (or edge routers) can independently use any multipath computation algorithm to find a subset of available paths, similar to the discussion in Section II-C. The only information needed is the knowledge of which routers in the network are upgraded (available with the MPC-bit in LSAs).

Path validation is only necessary to impose the constraint that non-upgraded nodes can forward packets only on their default shortest paths. Algorithm 2 shows the pseudo-code of a generic validation algorithm for edge routers. The idea is very simple: validate only those paths where the next-hop of the non-upgraded routers corresponds to the shortest path next-hop in those nodes. Again the validation algorithm consists of two phases. First phase deals with the computation of shortest paths for non-upgraded nodes (lines 4-6 in Algorithm 2) and computation of multiple paths using any desired multipath computation algorithm. In second phase, the paths are checked for passing through non-upgraded nodes. If a path passes through a non-upgraded node, the next-hop must be same as the next-hop in the pre-computed shortest path. A path is *invalid* if this condition is not met (lines 14-16). In a N-node network with u upgraded routers, the complexity of first phase is given $C(k) + (N - u)C(1)$ where, $C(k)$ denotes the complexity of computing k paths (assuming the upgraded router keeps k paths), $C(1)$ denotes the complexity of Dijkstra's single-shortest-path algorithm. The complexity of the second phase of the validation algorithm is $O(k \times (N - 1) \times (N - u))$, where k is the maximum number of paths for each destination to be stored in the forwarding table. Note that the validation phase in the index-based path encoding scheme is simpler compared to the validation phase in Algorithm 1. This is because the upgraded routers can forward packets to any of their interfaces. Recall that in Algorithm 1, the validation phase also needed to ensure that the downstream *upgraded* nodes of a path would indeed provide forwarding for that path (i.e. have a forwarding table entry for that path).

IV. BANANAS EXTENSIONS FOR INTRA-DOMAIN PROTOCOLS

In this section, we summarize the extensions to OSPF/IS-IS to support the BANANAS framework. A 32-bit PathID field is required in the packet header, that can be implemented as a new *routing option*, called *i-PathID*[5]. The route computation algorithm (Dijkstra's algorithm) at upgraded routers must be extended to compute multiple paths (e.g. DFS under partial upgrade constraints (DFS-PU), k-shortest paths [5] etc), and a validation algorithm (Algorithm 1). The upgraded nodes must

[5]From here on, we use the words PathID and i-PathID interchangeably in the context of intra-domain routing

Algorithm 2 Algorithm for validating paths in new Scheme

Let $\mathcal{N}$ denote the set of nodes in a network and $\mathcal{NU}$ denote the set of non-upgraded nodes
Compute multiple paths using desired multipath computation algorithm
Let $\mathcal{P}(dst)$ denote the set of paths to destination dst
for $n \in \mathcal{NU}$ **do**
  Compute Dijkstra
end for
for $dst \in \mathcal{N}$ **do**
  Compute the desired paths to destination dst using any of k-shortest paths, k-disjoint paths, all paths upto k-hops etc.
  for $path \in \mathcal{P}(dst)$ **do**
    for $n \in \mathcal{NU}$ **do**
      if $path$.find(n)==TRUE **then**
        // nextHopSP is the next-hop in the shortest path from n to dst
        // nextHop($path$) denotes the next-hop of n in the $path$
        if nextHop($path$) $! =$ nextHopSP **then**
          delete $path$
        end if
      end if
    end for
  end for
end for

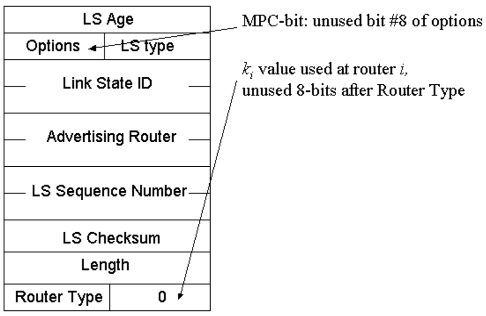


Fig. 7. Proposed Modifications to OSPF Link State Advertisements (LSAs)

compute the shortest path as the default path. Incoming packets with erroneous PathIDs are forwarded on the shortest paths and the PathID field set to zero. The intra-domain forwarding tables at upgraded routers would have tuples *(destination prefix, incoming PathID, outgoing interface (next-hop), outgoing PathID)*. As indicated in Figure 7, one bit in the OSPF Link State Advertisements (LSAs) [27] must be used to indicate that the router is multipath capable (MPC). In the Linux/Zebra based implementation as well as in the SSFNet simulations, we have used the eighth bit in the *LSA options* field of the router-LSA as the MPC bit.

Also, if we allow different upgraded routers to compute paths using different algorithms, 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). In our Zebra-based implementation, we have assumed that upgraded nodes implement the k-shortest-path algorithm with different values of k. Therefore, we leverage the currently unused 8-bits after the router type field in the LSA to indicate the value of k.

For the alternative index-based path encoding scheme, the concatenation of indices is done from the lower-order-bits to the high-order-bits. Each router simply shifts the PathID to the right by the number of bits needed to encode its interface index. This allows upgraded *interior* routers to extract the next-hop index from the lowest-order-bits without knowing its position within the path, i.e. without the knowledge of how many upgraded nodes are on the path. The upgraded interior routers only need to set the MPC bit in their LSA and need not advertise the route computation algorithm. Each upgraded router must maintain an ordered list of its own interfaces and the corresponding index. For example, a router with 15 interfaces needs 4 bits to store the index, i.e., the size of the index table is just 540 bits (15 x (32 bits + 4 bits)). The upgraded *edge* routers can use any multipath algorithm to compute multiple paths. However, they need to validate the paths using the validation algorithm (Algorithm 2). Like today's non-upgraded routers, upgraded routers must always compute the default shortest paths to all destinations. This is necessary in order to forward packets with no PathID option, zero or erroneous PathID.

A. *Forwarding Across Multiple Areas*

Large OSPF and IS-IS networks support hierarchical routing with up to two levels of hierarchy. Our approach is to view each area as a flat routing domain for the purpose of multipath computation. Multiple paths are found locally within areas, and crossing areas are view as crossing to a new multipath routing domain, i.e. we re-use the i-PathID field. For example, if a source needs to send a packet outside an area, it chooses one of the multipaths to the area border router (ABR). Then, the ABR may choose among the several multipaths within area 0 to other ABRs. The i-PathID field is re-initialized by the first ABR at the area-boundary.

V. BANANAS EXTENSIONS TO BGP

A. *Motivation and Goals*

BGP-4 [28] is *the* inter-domain routing protocol in the Internet. BGP uses a path vector and policy routing approach to announce a subset of actively used paths to its neighbors. Load-balancing and traffic engineering in BGP are becoming important as operators attempt to deploy services like virtual private networks (VPNs), and optimize on complex peering agreements [1], [29], [30], [31]. Enterprises are also increasingly multi-homed and are increasingly active in managing their inbound and outbound traffic [1], [32].

BGP-4 designers have provided a number of parametric "hooks" [28] (e.g. MED, LOCAL_PREF, AS-PATH, community attributes etc) to "tweak" BGP to achieve such traffic management goals. However, the parameter setting process today is largely manual, error prone and is likely to get harder as the protocol is overloaded to serve more functions [33], [30], [34].

While BANANAS is not designed to address this vast multitude of problems, it does provide a set of building blocks to enable fine-grained BGP traffic engineering both within and across domains. Fine-grained policy routing has been a motivating factor for some overlay networks [14].

In particular, BANANAS introduces two new capabilities: *explicit exit* forwarding and *explicit AS-PATH* forwarding. *Explicit-exit forwarding* refers to the capability where traffic aggregates can be explicitly mapped to AS exits (i.e. specific border routers). *Explicit AS-PATH forwarding* is similar to the explicit path routing introduced for OSPF/IS-IS, except that we now refer to explicit AS-PATHs rather than a sequence of contiguous routers and links. We examine these aspects further in the following sections.

B. Explicit-Exit Forwarding

The idea of explicit-exit routing is quite simple. The overall objective is to define a traffic aggregate and then map it to a chosen exit router (ASBR). Traffic aggregates may be chosen at per-packet, per-flow or per-prefix granularities by the upgraded EBGP or IBGP routers, i.e., ISPs can define fine-grained bundles of outbound traffic. This explicit exit capability is superior to the LOCAL_PREF mechanism in BGP-4 [28]. LOCAL_PREF is like a priority mechanism that maps all traffic to a destination prefix to the same exit. The explicit exit capability can map traffic for the same destination prefix to multiple exits (based upon the autonomous decisions at upgraded IBGP nodes).

The explicit exit mechanism works as follows. An upgraded IBGP router chooses an arbitrary exit AS border router (ASBR) for a given traffic aggregate (e.g. a flow or all traffic to a destination prefix). It then "pushes" the destination address into a *"address stack"* field, and replaces the destination address with the exit ASBR address (adjusting the checksum appropriately). Now, intermediate routers forward the packet to the exit-ASBR to which it is addressed. The exit-ASBR then simply "pops" the address from the address-stack field back into the destination address field (and adjusts the checksum) before forwarding it along to the next AS.

The upgraded IBGP node would hence have table entries of the form: **[Dest-Prefix Exit-ASBR Next-Hop-to-Exit-ASBR]** and **[Dest-Prefix Default-Next-Hop]**. The second tuple is the regular IBGP-defined default policy route for the destination prefix: this forwarding entry is used for all traffic for which this IBGP router does not decide the exit router. The first 3-tuple is applied only to the traffic aggregates for which this IBGP router chooses an explicit exit. This kind of operation is important to avoid conflicting exit routing decisions by upgraded IBGP routers.

Observe that only *a subset* of IBGP routers and exit ASBRs (eBGP) routers need to be upgraded. All BGP routers synchronize on their default policy routes as usual [28]. In addition, the upgraded exit ASBRs should also synchronize with the upgraded IBGP routers so that they know which exits are available for any given prefix. This additional synchronization between upgraded nodes can be accomplished as a simple extension to current full-mesh IBGP operation.

The explicit-exit mechanisms proposed are similar in spirit to the label-stacking (multi-level tunneling) ideas in MPLS[21]. A key difference is that BANANAS proposes only a single-level address stack, whereas MPLS can have multiple levels in its label-stack. Also the "labels" used in BANANAS are global identifiers (IP addresses) whereas MPLS uses locally meaningful labels. Another interpretation of the explicit exit functionality is that it is a simple and efficient connectionless tunneling mechanism that does not require manual configuration. Also, from an abstract framework perspective, the explicit exit routing is a special case of explicit path routing introduced in earlier sections. The PathID "hash" in this case is simply the exit ASBR IP address. This address stacking procedure operates in the fast processing path at all routers (both upgraded and non-upgraded), unlike IP loose-source-routing that defaults to the slow-processing path because it is an IP option.

C. Explicit AS-PATH Forwarding

The goal of explicit AS-PATH forwarding is to provide a distributed mechanism to send packets along an arbitrary, but validated AS-PATH. The idea is similar to the explicit path routing introduced for OSPF/IS-IS, except that we now refer to explicit AS-PATHs rather than a sequence of contiguous routers and links. In particular, we propose a separate hash field called external-PathID or e-PathID in packets for this function. The e-PathID is the hash of the desired AS-PATH, i.e., hash of the sequence of AS numbers.

The e-PathID hash is processed as follows. First, in an upgraded AS, we assume that at least the entry and exit AS border routers (ASBRs) are upgraded to support the explicit AS-PATH function. Assume that a border router (called the entry ASBR) receives a packet with a non-zero, valid e-PathID. The incoming e-PathID is used by the entry ASBR to determine an appropriate exit ASBR. The packet is then explicitly sent to this exit ASBR using the mechanisms described in the earlier section, i.e. address-stacking. Indeed, once the address is stacked, the i-PathID may also be explicitly chosen to indicate a specific route to that exit ASBR. We require however that the e-PathID is *not* swapped at the entry ASBR. The outgoing e-PathID (for the AS-PATH suffix) replaces the incoming e-PathID only at the *exit* ASBR. This convention is required because the autonomous system is an atomic entity (similar to a node) as far as the e-PathID is concerned. However, the AS physically breaks up into an entry- and exit-ASBR (similar to input and output interfaces of a node). If we imagine that the abstract PathID swapping happens at the output interface, that corresponds to our convention of swapping the e-PathID at the exit ASBR. Observe, that we have required only EBGP routers to be aware of the multi-AS-PATH feature, and do not require upgrades in selected IBGP routers (unlike the explicit exit case discussed earlier).

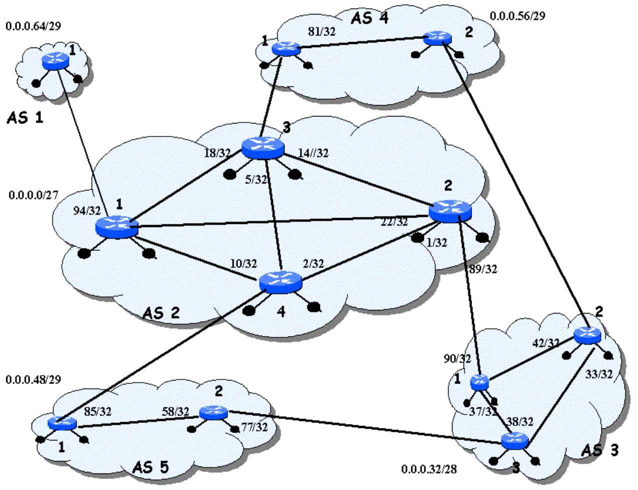


Fig. 8. Topology for illustrating explicit AS-PATH forwarding

To illustrate the explicit AS-PATH feature, we consider the AS-graph topology in Figure 8, and assume that we would like to send traffic from AS1 to AS5, i.e. to the IP prefix 0.0.0.48 along AS-PATH AS1-AS2-AS3-AS5, represented as (1 2 3 5). The AS-PATHs available are AS1-AS2-AS5, AS1-AS2-AS4-AS3-AS5, AS1-AS2-AS3-AS5. The explicit path (1 2 3 5) is chosen at router 1; the suffix AS-PATH is (2 3 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 AS-PATH (at Router 1 in AS 2) is learned from IBGP, the e-PathID is left unchanged at the entry ASBR (i.e., at Router 1 in AS2). The e-PathID will be swapped only at the exit ASBR (i.e. Router 2 in AS2). At this exit router, the exact match of the prefix and e-PathID results in a next hop in AS3. A similar sequence of events occurs in AS3 involving entry ASBR (router 1) and exit ASBR (router 3) before the packet is forwarded to AS5. The outgoing e-PathID from AS3 will be set to 0 because AS5 is the destination AS.

In spite of these apparent reductions in upgrade complexity, BGP's path-vector nature poses a more important problem. Specifically, a new AS-PATH is unknown to an upstream AS unless the intervening AS explicitly advertises it (after internal synchronization). In other words, even if ISPs were interested in AS-PATH multiplicity, increased control traffic is necessary to advertise the existence of multiple AS-PATHs to neighbor AS'es. Recall that such excess control traffic was not required in link-state algorithms (we merely piggybacked LSAs with minimal information). On the other hand, the path-vector nature of BGP-4 also implies that no path computation is necessary once the multiple AS-PATHs have been received and filtered for acceptance.

We recognize that this increased control traffic requirement poses a significant disincentive for ISPs against adopting multi-AS-PATH capabilities en masse. Given the scalability and instability issues with adding control traffic, we expect that ISPs may choose to advertise only a small set of multiple AS-PATHs to their neighbor AS'es. For example, some AS'es may collaborate to allow forwarding along multiple paths to certain destination prefixes and advertise this as a non-transitive attribute to certain AS'es only.

D. BANANAS Extensions for Inter-Domain Routing (BGP-4)

In summary, we propose two capabilities in the context of inter-domain routing: *explicit exit routing and explicit AS-PATH routing*. For the former, we propose a 32-bit "address stack" field in the routing header into which the destination IP address will be *"pushed"*. The destination field in the IP header is overwritten with the exit ASBR's IP address. The Exit ASBR will simply *"pop"* the destination address back from the "address stack" to the destination IP address. This address stacking procedure (similar to MPLS) operates in the fast processing path unlike the IP loose source routing option. Moreover, it allows flexibility for only a subset of BGP routers to be upgraded to support such explicit exit choice.

For explicit AS-PATH forwarding we propose a new 32-bit field in the packet routing header called the external PathID or e-PathID. This field stores a hash of the sequence of ASNs along the desired explicit AS-PATH. ISPs may choose to only advertise a small set of multiple AS-PATHs to their selected neighbor AS'es. In a multi AS-PATH capable AS, only the entry ASBRs and exit ASBRs (i.e. only the EBGP routers) need to be upgraded and synchronized on the available multiple AS paths. The incoming ePathID hash is swapped with the outgoing AS-PATH suffix hash only at the exit AS border router. The forwarding from the entry ASBR to the exit ASBR uses the explicit exit mechanisms described above. Multiple paths between the entry and exit ASBRs are possible using the i-PathID mechanism described earlier for intra-domain routing.

VI. Implementation and Simulation Results

In this section, we illustrate the working of the proposed framework. We have implemented the BANANAS framework schemes in the Linux kernel: we use MIT's Click Modular Router package [35] (data-plane) and GNU Zebra routing sofware version 0.92a [36] (control-plane). These implementations are tested on Utah's Emulab testbed [37] to emulate sizable topologies running real implementation code. In particular, we test three cases: a) when an upgraded router keeps all available paths (as computed by the DFS-PU strategy), b) when upgraded nodes compute k-shortest paths, with heterogeneous values of k at different nodes, and c) the index-based scheme to illustrate architectural flexibility.

We use SSFNet [38] for larger integrated BGP/OSPF simulations. These SSFNet simulations illustrate the framework in larger network topologies that integrate both OSPF and BGP BANANAS functionalities. Note that in this section, we have intentionally preferred simplicity in terms of topology/test-case choices. We have performed a larger set of SSFNet simulations and Emulab runs in more complex scenarios, all of which support our assertions. These results will be reported in a detailed technical report.

A. Linux Implementation Results

Figure 9 shows the topology of a simple validation experiment conducted on Utah's Emulab [37] testbed with the Linux Zebra version 0.92a of OSPF (i.e. control-plane) upgraded with our BANANAS building blocks. The forwarding plane was implemented in Linux using MIT's Click Modular Router package [35]. Note that this is a partially upgraded network: only nodes 1 and 2 (the dark colored nodes) are upgraded in this configuration. Figure 9 also indicates the IP addresses of various router interfaces and the link weights. The router ID is statically defined to be the smallest interface IP address.

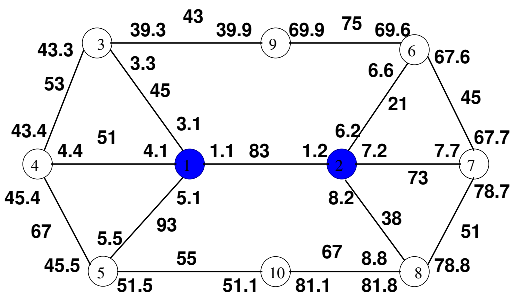


Fig. 9. Experimental Topology on Utah Emulab using Linux Zebra/Click Platforms (Note: only dark colored nodes are multi-path capable)

1) All Paths with Partial Upgrades (DFS-PU Algorithm): Table I illustrates a partial forwarding table computed at node 1 (IP address 192.168.1.1) for destination 3 (192.186.3.3). Note that the path string shown in Table I 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 outgoing path ID is the hash of the suffix path formed after omitting 192.168.1.1. If the path goes through other nodes which are not upgraded (e.g. 1-4-3), the outgoing 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	Incoming PathID	Outgoing PathID
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 I

Partial routing table at 192.168.1.1 for destination 192.186.3.3

2) k-Shortest Paths with Partial Upgrades: In this section we illustrate, using the Linux implementation, the case when the upgraded routers compute upto k-shortest paths, and different upgraded routers using different values of k.

Consider the 10-node topology shown in Figure 9. This topology was setup in the Emulab network. We assume that the routers 192.168.1.1 and 192.168.1.2 are upgraded with k equal to 3 and 2 respectively. The results are presented to verify the correctness of the "validation phase" (Algorithm 2). Tables II, III show respectively part of the routing tables at 198.168.1.1 for destinations 198.168.6.6 and 198.168.8.8 respectively. Tables IV, V show the corresponding entries at router 198.168.2.2. For destination 198.168.6.6 the router 198.168.1.1 finds 3 paths, all of which are valid as two paths have next-hop 198.168.2.2 and router 198.168.2.2 keeps 2 shortest paths. For destination 198.168.8.8, the router 198.168.1.1 computes 3-paths, 1-2-8, 1-2-6-7-8, 1-2-7-8. The path 1-2-7-8 is invalidated in the "validation phase" as router 198.168.2.2 only keeps 2 paths (2-8, 2-6-7-8). Note that the *Path* string is shown in Tables II-V for the purpose of explanation.

B. Evaluation of Index-based Path Encoding Scheme

The alternative index-based PathID encoding scheme was implemented in the Linux kernel (MIT's Click Router platform) and simulated in SSFNet. We present our simulation results in this section on a sizeable topology that corresponds to the old MCI topology of 1995 [39].

Path	Incoming PathID	Next-hop	Outgoing PathID
1-2-6	1989316858	192.168.1.2	3491782861
1-2-7-6	656924081	192.168.1.2	3645081405
1-3-9-6	534784006	192.168.3.3	0

TABLE II
PART OF ROUTING TABLE AT 192.168.1.1 FOR DESTINATION 192.186.6.6

Path	Incoming PathID	Next-hop	Outgoing PathID
1-2-8	3654096761	192.168.1.2	1973392862
1-2-7-6-8	1777786090	192.168.1.2	2123671348

TABLE III
PART OF ROUTING TABLE AT 192.168.1.1 FOR DESTINATION 192.186.8.8

In this configuration, only nodes 4, 6, 7, 9, 10 are upgraded. The source node in this simulation is node 6. Observe that node 6 is the only node that computes the k-shortest-paths (k = 5) for all destinations and runs the validation algorithm (Algorithm 2). All other upgraded nodes merely keep an index table as described in Section III-A). Table VI shows a part of the forwarding table at node 6 (only those paths for destination node 7), and the i-PathIDs using index-based encodings. The node 6 may choose any one of these paths for a packet to node 7. We have verified that the progression of i-PathIDs through the network follows the description given in Section III-B.

C. *Integrated OSPF/BGP SSFNet Simulation*

In this section we use SSFNet simulation results to illustrate the integrated operation of proposed framework in the Internet. This example demonstrates both the intra-domain (OSPF) and inter-domain (BGP-4) operation of the framework with explicit AS-PATH as well as explicit exit forwarding.

Figure 11 shows the topology used for the results presented in this section. The topology has eight (8) autonomous systems (AS'es). Four of these AS'es, namely AS1, AS2, AS5 and AS6, have been upgraded to support explicit AS-PATH forwarding. Even within these upgraded autonomous systems, only a *subset* of routers are upgraded to support the explicit

Path	Incoming PathID	Next-hop	Outgoing PathID
2-6	1973392862	0.0.0.0	1973392862
2-7-6	2123671348	192.168.7.7	2123671348

TABLE IV
PART OF ROUTING TABLE AT 192.168.2.2 FOR DESTINATION 192.186.6.6

Path	Incoming PathID	Next-hop	Outgoing PathID
2-8	3491782861	0.0.0.0	0
2-6-7-8	3645081405	192.168.6.6	0

TABLE V
PART OF ROUTING TABLE AT 192.168.2.2 FOR DESTINATION 192.186.8.8

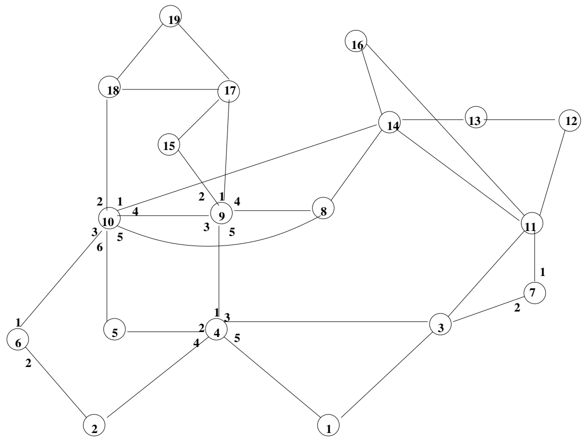


Fig. 10. Old MCI Topology: Used for Testing the Index-Based Scheme (Only Nodes 4, 6, 7, 9, 10 are upgraded)

Path	Next-Hop	i-PathID
6-2-4-3-7	2	0b01110
6-10-9-17-16-11-7	10	0b00110001
6-10-14-11-7	10	0b00101
6-10-9-4-3-7	10	0b01110110001

TABLE VI
PATHS AT NODE 6 FOR DESTINATION NODE 7 (NOTE: 0B INDICATES BINARY ENCODING)

Forwarding Table of AS1 at Router 1					
Dest	NextHop	In e-PathID	AS-PATH	Out e-PathID	Exit ASBR
0.57/28	**2.93/32**	**2025862315**	**2-4-8**	**3535826417**	**0.91/32**
0.57/28	2.93/32	4160716901	2-5-6-7-8	1248156781	0.91/32
0.57/28	2.93/32	669121903	2-5-6-4-8	2630971039	0.91/32

TABLE VII
INTEGRATED OSPF/BGP SIMULATION: FORWARDING TABLE OF THE BORDER ROUTER IN AS1 (NOTE: 0.57/28 REFERS TO IP ADDRESS 0.0.0.57/28 ETC)

Forwarding Table of AS2 at Router 5					
Dest	NextHop	In e-PathID	ASPATH	Out e-PathID	Exit ASBR
0.57/28	**2.97/32**	**3535826417**	**2-4-8**	**3535826417**	**2.107/32**
0.57/28	2.113/32	3535826417	2-4-8	3535826417	2.107/32
0.57/28	2.97/32	1248156781	2-5-6-7-8	1248156781	2.24/32
0.57/28	2.113/32	1248156781	2-5-6-7-8	1248156781	2.24/32

TABLE VIII
INTEGRATED OSPF/BGP SIMULATION: FORWARDING TABLE ROUTER 5 IN AS2 (SEE FIGURE 12)

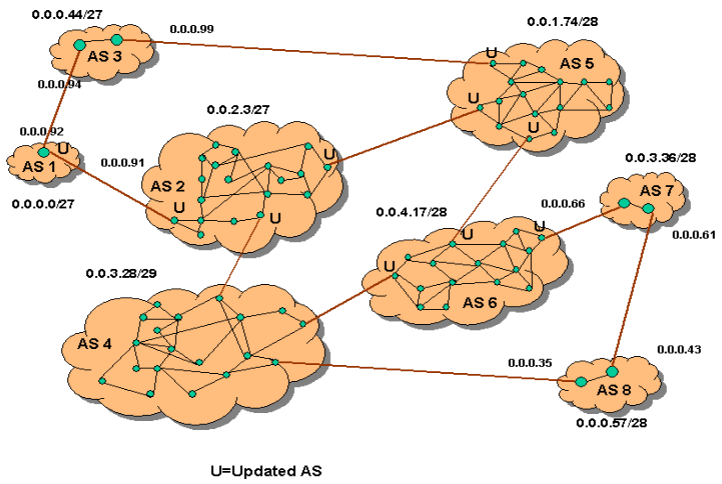


Fig. 11. Topology used for integrated SSFNet simulation

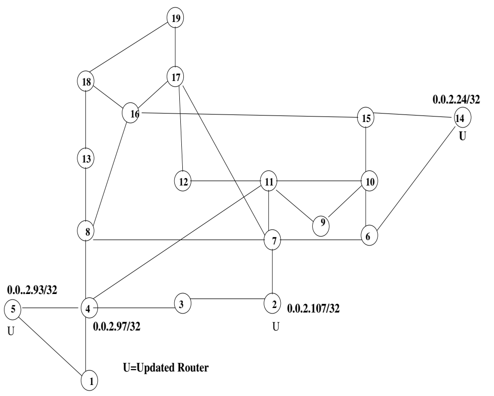


Fig. 12. Blow-up of AS2's Internal Topology in the Integrated OSPF/BGP Simulation (Figure 11)

Destination	Path	i-PathID
0.0.2.107/32	5-4-3-2	17
0.0.2.107/32	5-1-4-3-2	18
0.0.2.107/32	**5-4-11-7-2**	**1669**
0.0.2.107/32	5-4-8-7-2	201
0.0.2.24/32	5-4-11-10-15-14	69
0.0.2.24/32	5-4-8-7-6-14	169
0.0.2.24/32	5-4-8-16-15-14	105
0.0.2.24/32	5-1-4-8-16-15-14	106
0.0.2.24/32	5-4-11-9-10-15-14	101
0.0.2.24/32	5-1-4-11-9-10-15-14	102

TABLE IX
FORWARDING TABLE AT ROUTER 5 IN AS2 (FIGURE 12): K SHORTEST PATHS (K = 7)

AS-PATH and explicit exit routing as described in Sections V-C and V-B. The upgraded routers have been marked with a "U" in Figure 11. A blow-up of the internal topology of AS2 is shown in Figure 12; the upgraded routers are again indicated with "U"

Consider forwarding of a packet from AS1 to AS8 (see Figure 11). Given the constraints that only a partial set of AS'es are upgraded, the following AS-PATHs may be used from AS1 to reach AS8: AS2-AS4-AS8, AS2-AS5-AS6-AS7-AS8 and AS2-AS5-AS6-AS4-AS8. These AS-PATHs and their corresponding e-PathIDs are indicated in Table VII, which is a part of the routing table at the AS border router in AS1. Note that the AS-PATH AS2-AS4-AS6-AS7-AS8 is not available because AS4 is not upgraded, and uses a default AS-PATH of AS4-AS8. Also in this simulation, we assumed that the upgraded routers do not do any further filtering, i.e., they re-advertise all their available AS-PATHs to their neighboring AS'es.

In our example simulation, the border router of AS1 chooses the AS-PATH AS2-AS4-AS8, which corresponds to the e-PathID of 3535826417 (see the first row of Table VII). When the packet arrives at router 5 of AS2 (the entry ASBR), its header looks like Figure 13(A). This entry ASBR (i.e. router 5) of AS2 examines the incoming e-PathID to find the exit ASBR to be node 2 with IP address 0.0.2.107 (see first row of Table VIII). Note that it *does not* swap the e-PathID field, because this will be done at the exit ASBR. To emphasize this point, observe that the outgoing e-PathID column in Table VIII is the same as the incoming e-PathID for the destination prefix 0.0.0.57/28.

The entry ASBR (router 5) now "pushes" the destination IP address (i.e. 0.0.0.57) into the address stack field and replaces it with the exit ASBR IP address. The entry ASBR also chooses a path within the AS to the exit ASBR. Table IX shows the intra-domain paths available to reach exit ASBR (router 2). In this simulation, we have integrated the index-based PathID encoding scheme as well as the k-shortest path route computation scheme (k=7) with the OSPF protocol running in AS2. In particular, the path 5-4-11-7-2 within the AS is chosen that corresponds to a i-PathID of 1669 (see the third row of Table IX). The header fields of the packet at this stage are shown in Figure 13(B).

The packet proceeds on the explicit intra-domain path (as described in earlier sections) to reach the exit router 2 with an i-PathID value of 0. At this router, the destination address (0.0.0.57) is "popped" back from the address stack. The e-PathID is also replaced with the outgoing e-PathID of 1895667324 (see Figure 13(C)). Now the packet is sent to AS4, which is not upgraded, but sends the packet on its default policy AS-PATH, i.e., directly to AS8. In summary, we have shown how a distributed set of upgraded and non-upgraded nodes, with explicit paths independently selected within upgraded AS'es can honor an explicit AS-PATH request of the source AS.

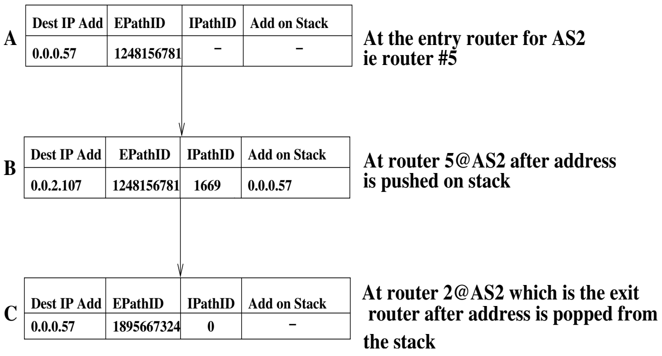


Fig. 13. Diagram Showing How e-PathID, i-PathID and Destination Address Change in the Integrated OSPF/BGP Simulation

VII. Related Work

Most related work for multipath routing have been done in the context of intra-domain protocols. OSPF, the most common intra-domain routing protocol used in the Internet today is based on single shortest path with equal splitting between next-hops of equal cost paths. Lorenz et al [40] show that OSPF routing performance could be improved by $O(N)$ if traffic-matrix aware explicit source-based multipath routing is used (e.g. MPLS-based [42], [43]).

Protocol extensions to support multipath routing (both in RIP and OSPF) have been studied by Narvaez et al [7], Chen et al [6] and Vutukury et al [8]. In [7], authors propose to find loop-free multipaths only by concatenating the shortest paths of their neighbors with their link to the neighbors. This approach essentially uses a depth first search with a depth of 1, whereas we allow arbitrary depth in our DFS-PU algorithm. Chen et al and Vutukury et al [6], [8] propose more general multipath computations, but their schemes require the co-operation and upgrade of *all* the routers in the network. Chen et al present a general concept of suffix-matched path identifier to allow multipath computation using distributed computation, but they use *local labels* to realize the path like in ATM networks [20] or MPLS [21]. Therefore, they require a signaling protocol to map a global path specification to locally assigned labels at each node.

The proposed BANANAS framework allows source-based multipath routing using a “PathID”. The use of a *globally significant path hash* allows multipath capabilities *without signaling* (i.e. in a connectionless manner) even in a *partially upgraded* network. The signaling requirement for source-routing is seen in protocols like ATM networks, MPLS networks [21] and NIMROD [12] routing (a link-state approach to inter-domain routing). IPv4 [18] and IPv6 [19], [13] provide a variable-length loose-source-routing option that may be considered “data-plane” signaling. But IPv4/v6 uses a uncompressed string of IP addresses in contrast to our efficient PathID encoding schemes.

Even though MPLS has gained popularity in some large ISPs, a significant fraction of ISPs [44] favor using OSPF/IS-IS to enable multipath and traffic engineering capabilities. This is due to the widespread deployment and operational experience available with OSPF/IS-IS. Our approach extends the OSPF/IS-IS to allow such capabilities even in partially upgraded networks. Our index-based scheme offers significant reduction of state complexity in comparison to MPLS label tables. Our computations can also be further optimized using incremental k-shortest path algorithms similar to those suggested for OSPF’s Dijkstra algorithm [45], [46].

In LIRA [11], Stoica et al briefly propose a forwarding scheme which they suggest could replace MPLS. A path is encoded as the XOR of router IDs along the path, and is processed along the path using a series of XOR operations. The work in LIRA is a special case of the BANANAS framework. In particular, the authors do not consider the larger architectural issues of partial upgrades, route-computation, state-computation trade-offs, inter-domain operation etc. The focus in their paper was also different: a framework for service differentiation.

VIII. Summary and Concluding Remarks

The key contributions in this paper are:

- Identification of abstract multipath architectural concepts (global PathID semantics, efficient path hashing) that are crucial to avoiding the need for signaling and allowing incremental network upgrades in connectionless routing protocols.
- Canonical multipath and explicit path realizations in the context of legacy routing protocols: OSPF, BGP-4.
- Demonstration of significant architectural flexibility: alternative PathID encodings, alternative route-computation algorithms (DFS-PU, k_i-shortest paths), movement of complexity to edges, division of functions between data-plane and control-plane, development of distributed validation algorithms etc.
- Linux implementation results and integrated OSPF/BGP simulation results to validate various options

These building blocks can be used in two broad ways. First, they could be used in the context of traffic engineering within a partially upgraded legacy network. An operator may want to emulate signaled capabilities in a connectionless network (e.g. see [43], [40], [41]) or might desire fine-grained traffic management control hard to extract from parameter tweaking (e.g. see [31], [30], [32], [33]). The building blocks may be mixed and matched in a limited number of ways. For example, one could select a MD5+CRC32 encoding for BGP-4 (i.e. e-PathIDs) and a index-based encoding for OSPF (i-PathID). Obviously, a common encoding must be chosen across ISPs for the explicit AS-PATH case.

Second, and perhaps more important, the BANANAS framework building blocks could form the long-term basis for a best-effort end-to-end path multiplicity model. Through the independent partial upgrades of nodes in different autonomous systems, end-systems can have a growing *expectation* of multiple end-to-end paths. We strongly believe that such a mere *expectation* of end-to-end path multiplicity will trigger substantial application innovation. To test this hypothesis, we plan to deploy the BANANAS framework on the PlanetLab infrastructure [22] as a public experimental wide-area network overlay service by Fall 2003.

References

[1] G. Huston, "Commentary on inter-domain routing in the internet," RFC 3221, December 2001.

[2] D. Meyer, "RouteViews Project," http://www.routeviews.org

[3] N. Spring, R. Mahajan, D. Wetherall, "Measuring ISP Topologies with Rocketfuel," *SIGCOMM 2002*, Pittsburg PA, August 2002.

[4] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, W. Willinger, Network Topology Generators – Structural vs. Degree-Based, Proceedings of the ACM SIGCOMM, August 2002.

[5] D. Eppstein, "Finding the k shortest Paths," Proceedings of 35th IEEE Symposium on Foundations on Computer Science (FOCS), pp. 154-165, 1994.

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

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

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

[9] D. O. Awduche, L. Berger, D. Gan, T. Li, G. Swallow, V. Srinivasan, "RSVP-TE: Extensions to RSVP for LSP Tunnels," *IETF RFC 3209*, December 2001.

[10] L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, "Label Distribution Protocol Specification," *IETF RFC 3036*, January 2001.

[11] I. Stoica, H. Zhang, "LIRA: An Approach for Service Differentiation in the Internet," in *Proceedings of NOSSDAV'98*, Cambridge, England, July 1998, pp. 115-128.

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

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

[14] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, R. Morris, "Resilient Overlay Networks," in *Proceedings of 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[15] S. Savage et al, "Detour: A Case for Informed Internet Routing and Transport," *IEEE Micro*, volume 19, no. 1, January 1999.

[16] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, "Stream Control Transmission Protocol," *IETF RFC 2960*, October 2000.

[17] H-Y. Hsieh, R. Sivakumar, "A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts," *Proceedings of ACM Mobicom 2002*, Atlanta, GA, September 2002.

[18] DARPA INTERNET PROGRAM, "Internet Protocol," *IETF RFC 791*, September 1981.

[19] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *IETF RFC 1883*, 1995.

[20] U. Black, "ATM, Volume I: Foundation for Broadband Networks,", *Prentice Hall*, 2nd Edition, 1999.

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

[22] L. Peterson, T. Anderson, D. Culler, T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proceedings of the First ACM Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, NJ, October 2002.

[23] W. Xu, S. S. Hemami, "Efficient Partitioning of Unequal Error Protected MPEG Video Streams for Multiple Channel Transmission," *Proceedings of IEEE International Conf. on Image Processing*, Rochester, NY, Sept. 2002. Available from: http://foulard.ee.cornell.edu/publications.html

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms," *The MIT Press, McGraw Hill Book Company* Second Edition, 2001.

[25] D. R. Musser, G. J. Derge, A. Saini, "STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library," *Addison-Wesley Pubs, 2nd edition*, March 27, 2001.

[26] Rka Albert, Hawoong Jeong, Albert-Lszl Barabsi, "Diameter of the World-Wide Web," Department of Physics, University of Notre Dame, Notre Dame, Indiana 46556, USA, appeared in *NATURE, VOL 401,9*, SEPTEMBER 1999.

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

[28] J. W. Stewart, "BGP-4 Inter-Domain Routing in the Internet," *Addison Wesley*, 1999.

[29] W. Norton, "Internet Service Providers and Peering," White Paper, 2002. http://www.equinix.com

[30] T. Griffin, G. Wilfong, "Analysis of the MED oscillation problem in BGP," *Proceedings of ICNP 2002*, Paris, France, November 2002.

[31] N. Feamster, J. Borkenhagen, and J. Rexford, "Controlling the impact of BGP policy changes on IP traffic," *AT&T Research Technical Report 011106-02*, November 2001. Available at: http://www.research.att.com/ jrex/papers/tm011106.02.ps

[32] S. Hares et al, "Smart Routing Technologies," *NANOG Panel*, Toronto, June 2002. http://www.nanog.org/mtg-0206/smart.html

[33] R. Mahajan, D. Wetherall, T. Anderson, "Understanding BGP Misconfiguration," *In Proceedings of ACM SIGCOMM*, 2002.

[34] T. Griffin, G. Wilfong, "On the Correctness of IBGP Configuration," *Proceedings of ACM SIGCOMM 2002*, Pittsburg PA, 2002.

[35] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems*, Vol. 18, No. 3, August 2000, pages 263-297.

[36] GNU Zebra Open-Source Routing Software, http://www.zebra.org/

[37] J. Lepreau, "The Utah Emulab Network Testbed," http://www.emulab.net/

[38] Scalable Simulation Framework (SSF) Network Models, available from *http://www.ssfnet.org*.

[39] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees," *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, Atlanta, GA, October 1997.

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

[41] H. Suzuki and F. A. Tobagi, "Fat bandwidth reservation scheme with multi-link and multi-path routing in ATM networks," In *Proceedings of IEEE INFOCOM*, 1992.

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

[43] A. Elwalid, C. Jin and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," *In Proceedings of INFOCOM'01*, April 2001.

[44] North American Network Operators Group (NANOG) attendees, *Private communication*, June 2002.

[45] G. Ramalingam, and T. Reps, "An incremental algorithm for a generalization of the shortest-path problem," *Journal of Algorithms,* Vol.21, 1996, pp. 267-305.

[46] P. Narvaez, K.-Y. Siu and H.-Y. Tzeng, "New Dynamic Algorithms for Shortest Path Tree Computation," *IEEE Transactions on Networking,* Vol. 8, No. 6, Dec. 2000.