Scalable Support of Interdomain Routes in a Single AS

Cristel Pelsser*, Akeo Masuda and Kohei Shiomoto
NTT Network Service Systems Laboratories, NTT Corporation, Japan

***Abstract*— The Internet has grown extremely fast in the last two decades. The number of routes to be supported by the routers has become very large. Moreover, the number of messages exchanged to distribute the routes has increased even faster. To keep up with the increase, network operators regularly have to perform costly upgrades of the routers. It is unclear whether advances in hardware will be able to keep up with the increasing routing load. More importantly, the large number of routes and iBGP messages negatively impacts iBGP convergence time leading to long connectivity losses.**

In this paper, we propose a scalable way to support the Internet routes in a Service Provider network. We make use of distributed servers that select routes on behalf of the routers. Then, routes are stored in a Distributed Hash Table (DHT). We adapted the concept of DHT for that purpose. Each router maintains its share of Internet routes in addition to a cache of routes currently in use to forward the Internet traffic. We call our proposal SpliTable. We show that our proposal is more scalable in the number of routes supported in each router than current iBGP route distribution solutions. Moreover, the number of control messages exchanged with our proposal is bounded contrary to current sparse iBGP route distribution solutions which may never converge.

I. Introduction

The Internet is divided into multiple domains also called Autonomous Systems (ASs). An AS is a network administered by a single entity such as an Internet Service Provider (SP) or a university, for example. The Border Gateway Protocol (BGP) is the routing protocol used to distribute the Internet routes between neighboring ASs and to distribute these routes inside an AS itself.

In the last two decades, the number of routes to be supported by the routers has become very large. Moreover, predictions say that this number will continue to increase in the future.

In addition to the number of routes, the number of messages exchanged to distribute the routes has increased even faster. In [1], Huston and Armitage have studied the increase in BGP routing entries and in number of BGP messages. They predicted a growth of the routing entries by a factor 2 while the amount of BGP advertisements will increase by a factor 4, in 5 years time.

The large number of routes and messages to be processed has an impact on BGP convergence. Feldmann et al. [2] have shown that BGP processing time is largely affected by the rate of BGP update messages and the number of BGP peers. They have shown that BGP processing time plays a significant role in BGP convergence time. Long BGP convergence times in turn affect the traffic [3].

Inside a SP network, the Internet routes are redistributed to the routers via iBGP. Traditionally, a full-mesh of iBGP sessions where established for that purpose. In order to reduce the load of the routers, solutions requiring the establishment of fewer iBGP sessions are now used. These solutions make use of Route-Reflectors (RRs) and confederations. They are called sparse iBGP topologies. These topologies are more scalable in the number of routes stored in the routers than an iBGP full-mesh. However, this number is still large. To be able to forward packets to every possible destination, each router maintains all the Internet routes in its Forwarding Information Base (FIB). Secondly, BGP requires that each router also maintains tables with the routes received from and sent to each of its BGP peers. These tables are called the Adj-RIB-Ins and Adj-RIB-outs. Their number increases with the number of BGP peers.

In addition to maintaining large routing tables, the number of BGP messages exchanged with a sparse iBGP topology is considerable. In some situations, the amount of messages required to distribute the routes is not even bounded. The iBGP protocol may not converge [4].

In this paper, we propose a solution to reduce the number of routing entries in the routers of a SP network and the number of BGP messages exchanged between the routers of the SP network. In our proposal, the distribution of the Internet routes inside the local AS is sure to converge. We solve the scalability issue of iBGP. By focussing on the single AS case, our solution provides a mean for network administrators to deploy a scalable Internet routing solution in their network without requiring changes in their customer, peer and provider networks, and, a fortiori to the global Internet. The adoption of our proposal in an AS is transparent to the external ASs.

In essence, we propose to split the Internet routing table on multiple routers of the AS. Each router does not maintain routes for every prefix locally, anymore. We call our proposal SpliTable (ST). We propose to use distributed servers and a method to distribute the load on these servers. The servers select routes on behalf of the routers. Normal routers do not have to maintain Adj-RIB-Ins and Adj-RIB-outs anymore. The same routes are selected for all the routers of a Point of Presence (PoP). For a prefix, two routes are selected for each PoP. With two routes per PoP, our solution is resilient in the face of the failure of one of the routes. Once the routes are selected at the servers, they are stored in a DHT. We adapted the concept of DHT for that purpose. Each router of a PoP

Cristel Pelsser now works at Internet Initiative Japan (IIJ).

maintains a portion of the Internet routes selected for the PoP. In addition, it maintains a cache of routes that are currently in use to forward the traffic. Once a packet arrives, if there is no entry for its flow in the cache, the corresponding route is retrieved from the DHT. Since the routes of a PoP are stored in the PoP itself, quick route retrieval upon packet arrival is ensured.

In the next section, we introduce the work that is related to our proposal. Then, in section III, we present SpliTable, our proposal. In section IV, we analyze the scalability of SpliTable with regard to the size of the routing tables and the amount of control messages required to distribute the routes. Finally, we conclude the paper and highlight future directions for this work.

II. Related work

Solutions that tackle the scalability issue of BGP are currently studied in the Routing Research Group (RRG) of the IRTF. Among the solutions considered, the most popular approach is the Locator/ID Separation Protocol (LISP) [5]. The authors propose to stop advertising in BGP the prefixes allocated to the ASs located at the border of the Internet. The hosts in these ASs are assigned an identifier. There is a mapping function that associates an IP address to a host's identifier. This IP address can be reached based on the BGP routes. Packets destined to the host are encapsulated to reach the node with the IP returned by the mapping function. The node with this IP address knows how to reach the destination host. It decapsulates the packets and forwards them to the host. The difficulty of this approach lies in its deployment in the Internet. A large number of ASs at the border of the Internet has to adopt the solution to see an impact on the scalability. Moreover, ASs that adopt the solution still have to be reachable from hosts in ASs that do not adopt it.

In [6], Krioukov et al. study the trade-offs in using compact routing in the Internet. In compact routing, the size of the routing table is reduced by aggregating the routes. This leads to longer routes and, thus, an increase in resource consumption. This phenomenon is called path stretch.

Virtual Aggregates [7] is an example of a compact routing solution applied to a single AS[7]. We describe VA and compare it to our proposal in section IV.

Finally, [8] propose the use of Route Servers (RSs) that select a BGP route on behalf of other routers in the AS. The motivation of these works is to be able to perform smarter BGP route selection than in the routers. We also rely in this RS concept. We see an additional benefit in this concept. It may relieve the routers from the necessity of storing a large number of routes in their own tables. Our contribution with regard to the RSs is that we propose in this paper a means to distribute the routing load among the RSs.

III. SpliTable (ST) Architecture

In this section, we describe our proposal for the scalable redistribution of Internet routes inside an AS. Our proposal, SpliTable, relies on a distributed route selection mechanism and a DHT to split the routing table on the routers of the AS.

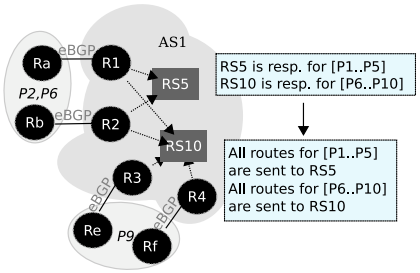

Fig. 1. Route Servers prefix allocation.

A. Distributed Route Selection

We introduce Route Servers (RSs) in the AS. Each server is responsible of BGP path selection for a subset of the external prefixes. It receives all the external BGP messages for these prefixes. Prefixes are assigned to a RS as follows. Each RS is assigned an ID. The set of RS IDs is noted R. There is a function that maps each prefix to a key. Route servers' identifiers and prefix's keys belong to the same domain K. Thus, $R \subseteq K$. Each RS with ID r_i is responsible for prefixes with key k comprised in $r_j < k \leq r_i$, where $r_j, r_i \in R$ and r_j is the largest ID, that is smaller than r_i, assigned to a RS.

In order for the AS Border Routers (ASBRs) to send all the routes they learn on eBGP sessions to the appropriate RSs, each ASBR has an iBGP session with each RS. The ASBRs discover the RSs that are present in the AS, as well as their ID, by means of the IGP.

In figure 1, the ID of a route server is the number used in its label. Thus, the ID of $RS5$ is 5. Moreover, the key of a prefix is the index used in its label. For example, $P2$ has key 2. In figure 1, $RS5$ receives all the routes for prefix $P2$. The advertisements concerning $P6$ and $P9$ are sent to $RS10$.

Traditionally, the BGP routes are redistributed to the routers of an AS on iBGP sessions. Then, each router performs the selection of its own routes. This changes with our proposal. Routers do not perform their own selection anymore. Thus, the current route selection procedure has to be adapted.

The current BGP Decision Process (DP) is shown in table I. The 4th and the 5th rules of the DP make use of the location of the router in the topology. These rules ensure hot-potato routing. Hot-potato routing consists in sending traffic as soon as possible outside of the AS to reduce the internal cost of carrying interdomain traffic. We provide a network of reference for illustrating the BGP route selection in Figure 2. The considered AS is divided into Points of Presence (PoPs). A PoP is a set of routers that are present in the same location, for example in the same city or the same building. Figure 3 illustrates the traditional BGP route selection at the routers in PoP "PoP-SW" assuming a full-mesh of iBGP sessions. Routes with Rd and Rc as Next-Hop (NH) are eliminated at the 5th rule of the BGP decision process. The cost to reach the NH is higher for these routes than for routes with NHs Ra and Rb. Then, when comparing routes with NHs Ra and Rb, Rb has a higher ID than Ra. Thus, it is eliminated and Ra is selected as best route for $R7$, $R8$ and $R9$.

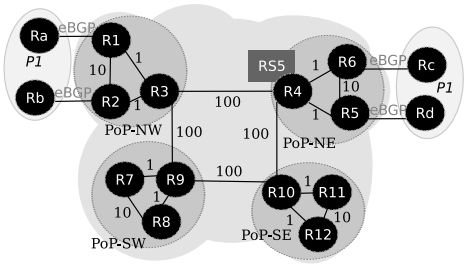

Fig. 2. Reference network.

Traditional route selection for P1 at R7, R8, R9 (assuming an iBGP full-mesh)

Prefix: *P1* NH: Ra	Prefix: *P1* NH: Rb	Prefix: *P1* NH: Rd	Prefix: *P1* NH: Rc
1)Loc_pref: 50	1)Loc_pref: 50	1)Loc_pref: 50	1)Loc_pref: 50
2)AS_path length: 2	2)AS_path length: 2	2)AS_path length: 2	2)AS_path length: 2
3)MED: max value	3)MED: max value	3)MED: max value	3)MED: max value
4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP
5)IGP cost:	5)IGP cost:	5)IGP cost:	5)IGP cost:
from R7: 102	from R7: 102	from R7: 202	from R7: 202
from R8: 102	from R8: 102	from R8: 202	from R8: 202
from R9: 101	from R9: 101	from R9: 201	from R9: 201
6)Tie break:	6)Tie break:	6)Tie break:	6)Tie break:
lowest router id	high router id	not considered	not considered

Fig. 3. Traditional route selection.

We note that, as in Figure 2, in deployed SP networks, the IGP cost of inter-PoP links is usually higher than intra-PoP link costs. This common design [9] ensures that traffic originated in and destined to the same PoP stays in the PoP. This design has the following consequence: *all the routers in the PoP perform a consistent route selection*. If an external route is received at a router of the PoP, this route will be selected. Otherwise, all the routers of the PoP select the same route, learned on an eBGP session in a different PoP. In Figure 3, all the routers select a route in PoP "PoP-NW" for $P1$. Similarly, all the routers in "PoP-SE" select the route with Rc as NH, according to the traditional route selection process.

TABLE I
SIMPLIFIED BGP DECISION PROCESS (DP)

Sequence of rules			
1	Highest `Loc_pref`	4	eBGP over iBGP
2	Shortest `AS-path`	5	Lowest IGP cost to NH
3	Lowest `MED`	6	Tie-break

In our proposal, route selection is done at the RSs instead of at the individual routers. However, in order to favor hot-potato routing, the location of the router that will use the route is considered in the route selection process.

Route selection is consistent for all the routers in a PoP because they are in the same location. Thus, it is sufficient for the RS to select the same route for all the routers in the PoP. The RS only computes routes for one router of the PoP. We call this selection "per PoP route selection". In our proposal, each router advertises the identifier of its PoP in the IGP. This enables the RSs to identify the nodes of a PoP.

First, there are no changes concerning the rules 1 to 3, in table I. The 4th rule of the BGP decision process becomes: "If the NH of the route is directly connected to one of the routers of the PoP for which the selection is performed, select this route". In the 5th rule, the route server will keep the routes with NHs that are the closest to the considered PoP. Since intra-PoP paths have lower IGP cost than inter-PoP paths [9], the 5th rule of the DP can rely on the IGP cost from any node in the PoP to the route's NH. This cost is computed from the link costs distributed by the IGP. Finally, there are no changes in the application of the tie-breaking rules.

Route selection for P1 at RS5
Route selection for PoP-SW

Prefix: *P1* NH: Ra	Prefix: *P1* NH: Rb	Prefix: *P1* NH: Rd	Prefix: *P1* NH: Rc
1)Loc_pref: 50	1)Loc_pref: 50	1)Loc_pref: 50	1)Loc_pref: 50
2)AS_path length: 2	2)AS_path length: 2	2)AS_path length: 2	2)AS_path length: 2
3)MED: max value	3)MED: max value	3)MED: max value	3)MED: max value
4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP
5)IGP cost:	5)IGP cost:	5)IGP cost:	5)IGP cost:
from R7: 102	from R7: 102	from R7: 202	from R7: 202
6)Tie break:	6)Tie break:	6)Tie break:	6)Tie break:
lowest router id	high router id	not considered	not considered

Route selection for P1 at RS5
Route selection for PoP-NW

Prefix: *P1* NH: Ra	Prefix: *P1* NH: Rb	Prefix: *P1* NH: Rd	Prefix: *P1* NH: Rc
1)Loc_pref: 50	1)Loc_pref: 50	1)Loc_pref: 50	1)Loc_pref: 50
2)AS_path length: 2	2)AS_path length: 2	2)AS_path length: 2	2)AS_path length: 2
3)MED: max value	3)MED: max value	3)MED: max value	3)MED: max value
4)eBGP/iBGP: eBGP	4)eBGP/iBGP: eBGP	4)eBGP/iBGP: iBGP	4)eBGP/iBGP: iBGP
5)IGP cost:	5)IGP cost:	5)IGP cost:	5)IGP cost:
from R2: 2	from R2: 1	not considered	not considered
6)Tie break:	6)Tie break:	6)Tie break:	6)Tie break:
lowest router id	high router id	not considered	not considered

Fig. 4. Per PoP route selection.

Figure 4 illustrates the per PoP route selection performed at route server $RS5$ for prefix $P1$ in the network of Figure 2.

As it is already widely recommended in traditional iBGP topologies, in order to avoid forwarding loops[10], encapsulation is also required to avoid such loops with our proposal.

In order to allow fast restoration and load balancing of the traffic from a PoP on multiple ASBRs, we enable the RSs to select for each PoP multiple routes to a given prefix. We observe here that multiple routes' selection per PoP enables the routers of "PoP-NW" in figure 2 to use both Ra and Rb for the traffic destined to $P1$, should the RS advertise the two best NHs to the routers of the PoP.

With per PoP route selection, all the nodes in the PoP use the same routes. Thus, per PoP route selection makes possible a split of a PoP's routing table. Each router of the PoP maintains only a portion of the routing table. When a router needs a route that is not present locally, it requests it from another node in its PoP. Since all the routers in a PoP are in the same location, retrieving a route from a neighboring node is fast. In the next section, we will describe how routes are distributed in a PoP and how route retrieval is performed.

B. Distributed Routing Tables

In this section, we present our proposal for maintaining distributed routing tables inside a PoP. We also describe how routing information is retrieved from this distributed route repository.

First, we list the set of properties that is desired from a routing system. We will show in section IV that our proposal verifies these properties.

- Maintaining routes in the routers should consume less memory than current BGP's routing tables' sizes.
- Distributing the routes on the routers and retrieving the routes necessary to forward traffic should consume less bandwidth and CPU than current iBGP route distribution.
- Retrieving a routing entry upon packet arrival should only take a few tens of milliseconds.

In the development of our proposal, we rely on the following assumptions:

- A large amount of the traffic received at an ASBR is destined to a small number of destinations.

This assumption is motivated by the findings expressed in [11] and [12].Due to this property of the traffic, we first note that the amount of active routes in the ASBRs is small. Thus, an ASBR does not need to maintain all the routes locally. In our proposal, ASBRs only maintain their share of routes and the subset of active routes in their tables. Second, there may be some benefit in replicating routes for popular destinations in order to retrieve this information quickly. Our second assumption is:

- The routes to popular destinations are stable.

This assumption is confirmed by the work of Rexford et al. [12]. This assumption implies that the routes active for a long period at an ASBR are not likely to change often. They can be maintained in a cache. There is no need for regular pulling of these routes at short timescales to check for changes.

In our proposal, we choose to store the routes in a Distributed Hash Table (DHT). DHTs provide a framework for the distribution of the routes on multiple nodes. We choose to use Kademlia [13] due to the following properties of this DHT:

1) The search functionality in Kademlia is based on the XOR metric. As we will show later in this section, this metric is suitable for searching for a prefix.
2) The information is replicated on multiple nodes. Replication provides robustness in the face of the failure of a DHT element. In our case, routes are still available if a network element fails, if the DHT graph is not partitioned.
3) In Kademlia, replication increases with the popularity of the information. It requires less messages and it takes less time to retrieve popular information. This property is suitable to the popular destinations trends observed in the Internet traffic [12].

Kademlia is currently used as a file tracker in BitTorrent, the widely deployed peer-to-peer network for file sharing. It enables to quickly find a list of nodes that participate in the torrent for the distribution of the searched file.

In Kademlia, nodes are assigned an identifier (ID). Moreover, each piece of information is assigned a key. Keys and node IDs belong to the same domain, the set of naturals that can be represented in 160 bits. The key identifies a piece of information. It is used to retrieve the information from the DHT. In our proposal, a piece of information is a route to be used in a given PoP.

The key is used to locate the nodes that will store the related piece of information. In Kademlia, node IDs and keys are compared to determine in which node to store the information. A piece of information will be stored in k nodes. k is called the replication parameter. These nodes have the IDs that are closest to the key based on the XOR metric. Figure 5 illustrates the storing action of a (key, value) pair in Kademlia as well

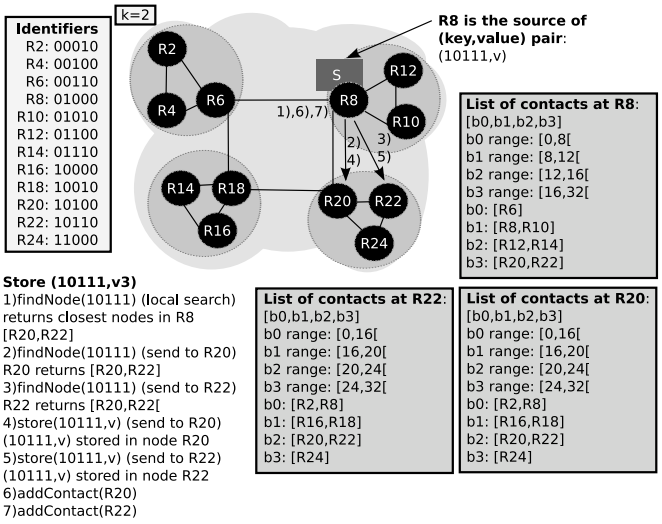

Fig. 5. Store a (key,value) pair in Kademlia.

as the use of the XOR metric. Each node maintains a list of contacts. For each contact, it knows the node ID, its IP address and the port on which it can be reached.

In figure 5, node $R8$ wants to store v with key 10111 in the system. Let assume in this example that $k = 2$. $R8$ looks in its own contact data structure for the closest nodes to 10111 (23 in decimal representation). The closest nodes to key 23 are $R20$ and $R22$, in the $b3$ list. According to the XOR metric, $R22$ is the closest node to key 23 since only the last bit of its ID, $1011\underline{\mathbf{0}}$ differs from the key, $1011\underline{\mathbf{1}}$. Similarly, $R20$ is the second closest node. The third bit of its ID differs from key 23. As $k = 2$, $R8$ contacts these two nodes. $R8$ asks these nodes if they know closer nodes to the key 23. This is done by sending a $findNode$ request. These nodes do not know closer nodes. They know the existence of $R24$. However, $R24$ is farther away than $R20$ and $R22$. The second bit of $R24$'s ID already differs from the key. Both nodes return $R20$ and $R22$ as closest nodes. Thus, $R8$ asks them to store v, by sending them a $store$ request. Finally, the closest nodes are added to the contacts of $R8$. Here no new nodes are added since they were already known. However, timers for these contacts are reset.

Now, assume that $R20$, in figure 5 wants to retrieve the value for key 23. It first looks for the (key,value) pair locally. If this fails, it searches for the k closest nodes to 23 in its contacts. These are nodes $R20$ and $R22$ according to the XOR metric. Thus, it asks $R22$ for the value with key 23. $R22$ looks in its own data structure. If it finds the value, it returns it. Otherwise it returns the k closest nodes to 23, which are $R20$ and $R22$. In the latter case, the search failed. The searched information is not present in the system.

We observe that the XOR metric is appropriate for IP destination and prefix comparison. IP packets are currently routed according to the longest match prefix. Let's assume that an ASBR has routes for $66/8$ and $66.249/16$. Today, when an IP packet with destination address $66.249.89.147$ is received, the ASBR routes the packet according to the route for prefix

$66.249/16$. The XOR metric is consistent with this practice. With the XOR metric, the longest match prefix $66.249/16$ will be closer to the IP destination than the other prefix. It will thus be preferred.

We will now explain how we propose to adapt Kademlia to our purpose. In our proposal, S is a RS and $R20$ is an ASBR, in figure 5. First, we consider the node ID assignment. We also define the keys to store a route in the system and to retrieve it. These concepts are particularly important as they determine the efficacy of the store and retrieval procedures.

In Kademlia, the ID of a node is 160 bits long. In our proposal, we set the ID of a node to its PoP identifier to which a hash value is concatenated.

$$< node_{id} >::=< PoP_{id} >< hash >$$

where the PoP_{id} is 32 bits long and $< hash >$ is 128 bits long. $< hash >$ is obtained by applying MD5 on a randomly generated number.

In this paper, the DHT is used to store routes. A route is a (key, value) pair where the key is 160 bits long. We denote the key of a route as k_r. The format of this key is:

$$k_r ::=< PoP_{id} >< prefix >< mask >< padding > .$$

$< PoP_{id} >$ identifies the PoP for which the route is destined. It is 32 bits long. $< prefix >$ is a 32 bits IPv4 prefix and $< mask >$ is the mask for the IPv4 prefix. The mask is also 32 bits long. $< padding >$ is 64 bits long. All these 64 bits are set to 0.

The format of the value component is $v ::=< version >< length >< NH_1 >< pref_1 > ... < NH_n >< pref_n >$ where $< length >::= n$. The value may consist of multiple NHs to which the traffic destined to the prefix may be sent. A preference is assigned to each NH. Setting multiple NHs in a route enables an ASBR to perform restoration upon a failure as well as to load balance the traffic on multiple paths during normal network operation. In this paper, we consider $n = 2$.

We do not apply any modifications to Kademlia's store procedure. Our definitions of k_r and $node_{id}$ ensure that routes destined to a PoP will preferably be stored in nodes of the PoP. This is because for a node in a PoP and a route destined to this PoP, the first 32 bits of the node ID and the route's key are identical.

Upon a packet arrival, the ASBR (for example $R20$ in figure 5) has to retrieve a matching route in order to forward the packet. However, the matching prefixes for the packet's destination is not known in advance. A fortiori, the most specific matching prefix is not known. Thus, k_r cannot be built. We define a different key to retrieve a route from the system. This key is denoted k_d:

$$k_d ::=< PoP_{id} >< IP >< mask >< padding > .$$

$< PoP_{id} >$ is the identifier of the PoP in which the packet is received. $< IP >$ is the packet's IP destination address. It is 32 bits long. The mask is composed of 32 bits set to 1. Finally, the $< padding >$ is 64 bits long. All these 64 bits are set to 0. The presence of $< PoP_{id} >$ at the head of the key enables to contain route search in the PoP and, thus, quickly find the route computed for the PoP.

Since the key of the stored route and the key to look up for a route are different, we cannot use Kademlia's look up procedure. Our proposed route retrieval method is shown in algorithm 1.

Let a be the ASBR receiving a packet, in algorithm 1 and 2. Moreover, N is the set of the next nodes to contact to search for the route.

Algorithm 1 retrieveRoute(k_d)

```
1:  repeat
2:    {search for a less specific prefix if a matching route is
      not found for the initial k_d}
3:    N_new=N=a {first search in the local node}
4:    {Initialize N_new in order to not override N in the first
      execution of the loop}
5:    repeat
6:      {search for the most specific prefix for k_d}
7:      N=N_new
8:      (N_new, P)=findMatchingRouteInClosestNodes(N,k_d)
9:      if (P ≠ ∅) then
10:       r=mostSpecificNewestRoute(P, r)
11:       installInFIB(r)
12:     end if
13:   until (N_new ⊆ N)
14:   k_d=resetLastIPAndMaskNonZeroBits(k_d)
15: until (defined r)
16: replicate(r)
```

In algorithm 1, we first search for a prefix matching the key k_d. If such a specific route cannot be found, we search for a less specific route. This is done by setting the last non zero bit of the IP address and the corresponding tailing bits of the mask to 0, in line 14. This is the purpose of the outer most loop (lines 1-15).

For a given k_d, in algorithm 1, lines 5-13, we first look in the local node for the most specific and newest matching route for k_d (line 8). We also search for the closest nodes to the key k_d (line 8). If a matching prefix is found, it is installed in the Forwarding Information Base (FIB) (line 11). Thus, we note that as soon as a route for a matching prefix is found it is installed in the FIB. This enables the ASBR to forward packets quickly even though the route is not the final, most specific route. The search continues to find the most specific route. For that purpose, the k closest nodes known locally, the nodes in N, are contacted in the next run of the inner loop (line 8). These nodes in turn locally search for the most specific and newest route. In addition, they respond with a list of k closest nodes. Details for `findMatchingRouteInClosestNodes` are provided in algorithm 2.

During a route search, when routes are received from multiple nodes, line 8 in Alg. 1, the set of "most specific

Algorithm 2 findMatchingRouteInClosestNodes(N,k_d)

1: **if** ($|N| = 1$ **and** $N = a$) **then**
2: N_{result}=getClosestNodes(k_d)
3: P_{result}=getMostSpecificRoute(k_d)
4: **else**
5: **for** ($n \in N$) **do**
6: (N_{new}, P_{new})=sendFindRoute(k_d, n)
7: N_{result}=$N_{result} \cup N_{new}$
8: P_{result}=$P_{result} \cup P_{new}$
9: **end for**
10: **end if**
11: **return** (N_{result}, P_{result})

matching routes" is identified. From this set, the method `mostSpecificNewestRoute` only returns the route with the newest version number (line 10). The version number of a route is important here. Each time a RS computes a route, it attaches a version number to the route. Popular routes are replicated in the DHT. When the RS recomputes a route, it deletes the old routes present in the k closest nodes. However, it is not able to delete the other replications. These copies timeout. In the mean time, the version number ensures that the stale routes are not used to forward traffic once the route search is terminated.

In this section, we presented our proposal, SpliTable. We showed how to distribute the load on multiple RSs while ensuring full visibility of the routes for a prefix at a RS. Together with encapsulation, this ensures correctness [4]. Our route distribution proposal converges and there are no forwarding loops. Then, we presented our modifications to Kademlia in order to support the storage of IP routes and their retrieval upon packet arrivals.

IV. Scalability analysis

In this section, we compare traditional iBGP route distribution and the Virtual Aggregate (VA) solution proposed by Francis et al. [7] to our solution. Concerning traditional iBGP, we consider both full-mesh iBGP topologies and sparse iBGP topologies. For each of these techniques, we quantify the routing information stored in each node and provide an upper bound on the number of control messages exchanged to distribute the routing information.

We define the following variables: p is the number of external prefixes learned by the AS, n is the number of nodes in the AS, q is the average number of iBGP peers of a node in a sparse iBGP topology. And, s is the number of RSs in the AS, with our proposal. It is assumed that $q < s$ and $n > s$. p is an upper bound for the number of prefix advertisement received at each ASBR. An overview of the variables used to compute the upper bound on the number of routes in the tables and control messages exchanged is provided in table II. We assign a value to each variable. This are the values used for the generation of figures 6 and 7. These values are only provided as an example of the values that may be observed in a real SP network.

TABLE II
Variables

Notation	Example value	Semantic
n	200	number of nodes in the AS
p	200000	number of external prefixes
q	16	average number of iBGP peers per node (in sparse iBGP topology)
r	3	average number of eBGP peers per ASBR
s	200	number of RS
a	160	number of ASBRs
l	20	number of PoPs
m	8400	number of eBGP messages (per hour)
k	2	replication factor in the DHT
t	3 min	cache timeout (in minutes)
d	18000	maximum number of destinations per "t" interval
c	4050	maximum number of cache misses per "t" interval

We computed the average number of iBGP peers q in table II based on a iBGP topology composed of 2 RRs per PoP. Each node in the PoP has an iBGP session with the RRs of its PoP. There is a full-mesh of iBGP sessions in the PoP. And, all the RRs of the AS are connected in a full-mesh of iBGP sessions. This design follows one of the recommendations in [14]. m encompasses the eBGP churn. The value assigned to m in table II, is the average number of route changes observed per hour at the routers participating in linx interconnection point. We computed this average based on one day's BGP updates data (March 24th, 2009), collected by the routeviews project (http://archive.routeviews.org/bgpdata/). The values for t, d, c are assigned according to the findings of Iannone et al. in [11]. The authors of [11] assumed that the Internet traffic of a university is routed according to the entries of a cache. For each flow, they installed a routing entry in the cache. For different timeouts, they determined the number of routing entries required in the cache to route all the traffic. They also measured to number of cache miss for each timeout value. In our proposal, the ASBRs request a route entry from the DHT upon a packet arrival. Once the route entry is received it is stored in a cache and can be used to route subsequent packets. The study in [11] gives us an idea of the number of route requests generated at an ASBR when a routing cache is used at the ASBR.

In VA, a router may act as Aggregation Points Routers (APR) for a Virtual Prefix (VP). A VP is an aggregate of prefixes. It is not necessarily part of the Internet routes. The APR advertises the VP in iBGP. The APR installs in its FIB the routes for the prefixes that are more specific than its VP. However, the other routers do not. They only install the VP in their FIB. They will thus route the traffic to the APR. The APR will then relay the route along the most specific route. This enables to reduce the size of the FIB in the routers. They do not have to install all the most specific routes in their FIB. However, it does not enable to reduce the size of the Adj-RIB-Ins and Adj-RIB-outs as a router still have to advertise all the Internet routes to its BGP peers. Since the traffic is first sent to an APR and then along the most specific route, traffic may follow a longer path than in traditional iBGP. This is called

the path stretch. If there are APRs for each VP in each PoP as suggested in [15], the path stretch is limited. In order to avoid forwarding loops, packets are encapsulated at the APRs. The destination of the encapsulating tunnel is the NH of most specific route.

A. Number of routes

In this section, we look at the number of routes stored in the nodes with a full-mesh, a sparse iBGP topology, VA and our proposal (SpliTable). First, we show in table III the formulas used to compute the number of routers stored in the nodes of an AS for each of the techniques.

TABLE III
TABLE SIZES

BGP table sizes		
Technique	Adj-RIB (in+out)	FIB
Full-mesh	$p(2r+2n-3)$	p
Sparse	$p(2r+2q-1)$	p
VA	$p(2r+2q-1)$	$(2lp/n)$ $+2(127-2*127(l/n))$
SpliTable (at RS)	$(p/s)(ar+2l)$	–

DHT table sizes (SpliTable)		
	DHT table	Routing cache
At ASBR	$pkl/n+(a/l)dl/n$ $+(a/l)(1440/t)cl/n$	d
At RS	$pkl/n+(a/l)dl/n$ $+(a/l)(1440/t)cl/n$	–

We see in table III that the number of routes in the Adj-RIBs with a full-mesh is proportional to the number of nodes in the AS. This is not scalable. Sparse iBGP topologies and VA are much more scalable. We also observe that for VA, only the size of the FIB is reduced compared to traditional iBGP topologies.

In SpliTable, only the RSs maintain BGP routes. The ASBRs do not maintain a complete FIB. They maintain a cache with only the most recent useful routing entries. The ASBRs and RSs maintain their share of the routing entries due to their participation in the DHT. This share is inversely proportional to the number of nodes in a PoP. The more nodes there is in a PoP, the less entries each node has to maintain, assuming a constant number of Internet prefixes.

Figure 6 illustrates the total number of routes stored in a node for sparse iBGP topologies, VA and SpliTable. We use the values listed in table II. We see that the gain of VA is very limited. For a network of 200 nodes, a node still maintains 98% of the routes stored in a node with the sparse iBGP topology. This is because the largest number of entries is stored in the Adj-RIB, not the FIB. VA aims only at reducing the size of the FIB. The number of routes stored in a RS with our proposal corresponds to 76% of the routes maintained in a node in the sparse iBGP topology, for a network of 200 nodes. At the ASBRs, this number drops to 21%. We are able to effectively reduce the size of the routing tables. We also note that the higher the number of nodes n, the less entries there are in the nodes with our proposal. This is because the number of nodes per PoP increases, n/l. Thus, there are more nodes on which the PoP's routing table can be split.

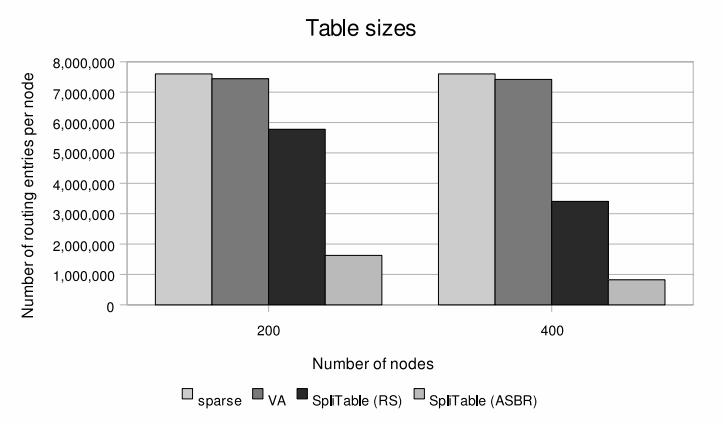

Fig. 6. Comparison of the number of routes.

B. Amount of control messages

In a full-mesh of iBGP sessions, the arrival of an external BGP message generates either 0 update messages, if the new route is not the best BGP route for the advertised prefixes or $n-1$ messages otherwise.

In the case of a sparse iBGP topology, it is not possible to predict the number of BGP messages following the reception of an eBGP update message. In some configurations, the reception of an eBGP message may generate an infinite number of iBGP messages. That is, the iBGP protocol may not converge in some situations. We refer the reader to [4] for an example of such a situation.

When making use of Virtual Aggregates to reduce FIB size, all the prefixes are still advertised in iBGP. Thus, the same number of iBGP messages is generated with VA as in the same iBGP topology without VA enabled.

For our proposal, we can provide maximum bounds on the number of messages required to distribute the routes to the route servers (first column in table IV), to store, update and remove entries from the DHT (second column) and to retrieve the routing entries from the DHT (third column). In table IV, we show the upper bound on the number of control messages that may be exchanged in one day in an AS.

TABLE IV
CONTROL MESSAGES

Technique	BGP	DHT (store, update, remove)	DHT (retrieve)
Full-mesh	$24m(n-1)$	0	0
Sparse	$infinite$	0	0
VA	$infinite$	0	0
SpliTable	$24m$	$48ml(min((n/l)+k, 129k))$	$a(d+(1440/t)c)(2*min(n/l, 24*128k)+1)$

Figure 7 shows the upper bound on the number of control messages for a full-mesh of iBGP session and for our proposal. With sparse iBGP topologies and VA making use of a sparse iBGP topology, it is not possible to provide upper bounds independently of a specific configuration. One of the merits of our proposal is that we can provide an upper bound on

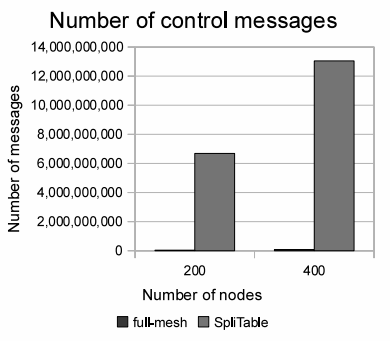

Fig. 7. Comparison of the number of control messages.

the number of control messages. Convergence of our route distribution solution is ensured.

We see in figure 7, that the maximum number of messages is much higher with our proposal than in a full-mesh. Moreover, it increases with the size of the network. As n increases, the number of nodes in a PoP increases. Therefore, the total number of nodes that can be contacted to store, update, delete or retrieve a route entry increases. However, we believe that the real number of control messages is far below this upper bound. The node discovery mechanism of Kademlia and the XOR metric ensure that the k closest nodes to a key are found after a very short exchange of messages. Usually, it requires of the exchange of at most 2 request, 2 response and 1 store messages to retrieve information in Kademlia. This is far below the upper bound of 21 messages for the retrieval operation. When taking these estimations into account, the number of control messages can be divided by a factor 4 for the AS of 200 nodes and by a factor 8 for the 400 nodes case.

The time required to retrieve a route from the DHT depends on the number of messages that are exchanged to retrieve the route. Retrieving a route requires at most $2 * min(n/l, 24 * 128k) + 1$ messages. This corresponds to a maximum bound of 21 messages for the network described in table II. Again, even though the upper bound on the number of messages is high, the real number of exchanged messages should be far lower than this upper bound. Moreover, these messages are all contained in the PoP of the requesting nodes. They are exchanged between nodes that are in the same location. Thus, the delay of exchanging these messages is expected to be low.

V. Conclusion and further work

In this paper, we proposed a solution to the scalable distribution of Internet routes in a SP network. We rely on distributed Route Servers (RSs) for the selection of routes. We proposed a way to distribute the load on multiple RSs. After the route selection, the routes are stored in a DHT. In a DHT, the same key is used to store information and then to retrieve it. We have shown that this is not the case when the information is a route entry and it has to be retrieved upon packet arrival. We adapted a DHT, Kademlia, to our needs.

We provided upper bounds on the routing table sizes and number of messages exchanged to estimate the scalability of our proposal. We compared our proposal to traditional iBGP topologies and the Virtual Aggregates (VA) proposal. We have shown that our proposal is very effective to reduce the size of the routing tables. The routers store much less routes than in a traditional sparse iBGP topology. Depending on their role (RS or ASBR), they only maintain 76% or 21% of the amount of routes present in the tables of a node in a sparse iBGP topology. The upper bound on the number of messages required to distribute the routes in the DHT and to retrieve them from the DHT is very high. However, we are convinced that this upper bound largely overestimates the number of control messages that will be observed in real network operation.

The next steps of our work will consist in developing a prototype of our proposal. We will verify our conviction with regard to the number of control messages and time to retrieve a mapping by developing a model of a SP network, its traffic and its external routes.

References

[1] G. Huston and G. Armitage, "Projecting future IPv4 router requirements from trends in dynamic BGP behaviour," in *ATNAC*, Australia, December 2006.

[2] A. Feldmann, H. Kong, O. Maennel, and A. Tudor, "Measuring BGP pass-through times," in *PAM*, 2004, pp. 267–277.

[3] F. Wang, Z. Mao, J. Wang, L. Gao, and R. Bush, "A measurement study on the impact of routing events on end-to-end internet path performance," in *ACM SIGCOMM 2006*, September 2006.

[4] T. Griffin and G. Wilfong, "On the correctness of iBGP configuration," in *ACM SIGCOMM 2002*, August 2002.

[5] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)," March 2009, internet draft, draft-farinacci-lisp-12.txt, work in progress.

[6] D. Krioukov, K. Claffy, K. Fall, and A. Brady, "On compact routing for the internet," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 37, no. 3, 2007.

[7] P. Francis, X. Xu, and H. Ballani, "FIB suppression with Virtual Aggregation," February 2009, internet Draft, draft-francis-intra-va-00.txt, work in progress.

[8] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *Networked Systems Design and Implementation (NSDI)*, May 2005.

[9] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier 1 backbone," *IEEE Network*, vol. 18, no. 2, pp. 13–19, Mar-Apr 2004.

[10] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving sub-50 milliseconds recovery upon BGP peering link failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1123 – 1135, October 2007.

[11] L. Iannone and O. Bonaventure, "On the cost of caching locator/ID mappings," in *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, December 2007.

[12] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "Bgp routing stability of popular destinations," in *Proc. Internet Measurement Workshop*, 2002.

[13] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *International workshop on Peer-To-Peer Systems (IPTPS 2002)*, March 2002.

[14] R. Zhang and M. Bartell, *BGP Design and Implementation*, 1st ed. Cisco Press, December 2003.

[15] P. Francis, "A configuration-only approach to shrinking FIBs," February 2008, presentation at NANOG42' meeting.