PAPER

SpliTable: Toward Routing Scalability Through Distributed BGP Routing Tables

Akeo MASUDA[†], *Member*, **Cristel PELSSER**[††], *Nonmember*, *and* **Kohei SHIOMOTO**[†], *Fellow*

SUMMARY

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. In this paper, we propose SpliTable, a scalable way to support the Internet routes in a Service Provider network. In our proposal, BGP route selection is done by distributed servers on behalf of the routers. They are called route selection servers. The selected routes are then stored in distributed routing tables. Each router maintains only its share of Internet routes, not the routes for each Internet prefix as it is the case today. We adapted the concept of Distributed Hash Tables (DHT) for that purpose.

We show analytically 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. We confirm these findings in an evaluation of a prototype implementation.

key words: *Scalable routing, BGP, DHT*

1. Introduction

The rapid growth of the Internet impacts the scalability of its routing infrastructure. 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[1]. In addition to the number of routes, the number of messages exchanged to distribute the routes has increased even faster. In [2], 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 negative impact on BGP convergence time leading to long connectivity losses. Feldmann et al. [3] 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 [4].

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. We call them 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. The number of adjacencies routing tables increases with the number of BGP peers thus leading to an increased number of routes stored per Internet prefix in the router. 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 load and capacity.

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 [5].

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 focusing 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, whose first concept was addressed in [6]. 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. We use a DHT, called Kademlia,

Manuscript received January 1, 2008.
Manuscript revised January 1, 2008.
Final manuscript received January 1, 2008.
[†]NTT Corporation, Musashino-Shi, 180-8585 Japan.
[††]Internet Initiative Japan, Chiyoda-ku, 101-0051 Japan.
DOI: 10.1587/trans.E0.??.1

to implement the distributed routing table concept. Once the routes are selected at the servers, they are stored in the DHT. Each router of a PoP 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.

We developed prototypes to conceptually prove the feasibility of the proposed architecture and measure its benefits. We implemented prototypes for the Route Selection Servers (RSS), ASBRs and internal routers. ASBRs relay the eBGP messages learned on eBGP sessions to the RSSs. RSSs perform per PoP BGP route selection and inject the resulting routes in the DHT. ASBRs resolve the route for a packet by means of the DHT and then forward the packet. RSS, internal routers and ASBRs participate in the DHT. They store a portion of the routes computed for their PoP. We use the prototype to measure the scalability improvements made possible by our proposal. In this paper, we describe the design and implementation of our prototypes. We also show some preliminary performance results of our proposal with the prototypes deployed in a sample network.

The paper is structured as follows. First, we introduce the related work. Then, in section 3, we present SpliTable, our proposal. We describe our prototype implementation in section 4. 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, in section 5. We show some preliminary results concerning the scalability of the proposal in a simple network in section 6. Finally, we conclude the paper and highlight future directions for this work.

2. Related work

One of the most popular approach to tackle the scalability issue of BGP is the Locator/ID Separation Protocol (LISP) [7]. 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 contrast, our proposal is designed to give scalability benefits immediately to the SP even the deployment is limited to its AS. We believe this aspect is important considering the realistic, incremental deployment of a new Internet architecture.

In [8], 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 [9] is an example of a compact routing solution applied to a single AS[9]. We describe VA and compare it to our proposal in section 5.

In [10] the use of Route Servers (RSs) is proposed. RSs 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 on 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 our RSSs is that we propose in this paper a means to distribute the routing selection and storage load among the RSSs.

There are several proposals of the routing architecture that make use of DHT technology. LISP-DHT[11] also aims to tackle the interdomain routing scalability [illegible] It is different in the sense that DHT is used for the r[illegible]tion of the RLOC-EID mapping information required in LISP, and it is designed to work in the global Internet. Our proposal uses DHT to share the route itself, and it works in a single AS. ROFL[12] is a suggestion of a routing based on addresses that only contain end-IDs. Here, routing is done with a mechanism similar to Chord ring. However ROFL does not address scalability issues. Authors admit that in present, performance including scalability is far from ideal.

3. SpliTable 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.

3.1 Distributed Route Selection

We introduce Route Selection Servers (RSSs) 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 RSS as follows. Each RSS is assigned an ID. The set of RSS IDs is noted R. There is a function that maps each prefix to a key. Route selection servers' identifiers and prefix's keys belong to the same domain K. Thus, $R \subseteq K$. Each RSS 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 RSS.

In order for the AS Border Routers (ASBRs) to send all the routes they learn on eBGP sessions to the appropriate RSSs, each ASBR has an iBGP session with each RSS. The ASBRs discover the RSSs that are present in the AS, as well

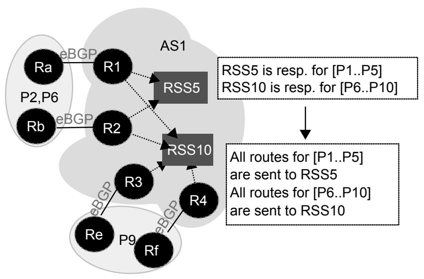


Fig. 1 Route Selection Servers prefix allocation.

as their ID, by means of the IGP.

In figure 1, the ID of a route selection server is the number used in its label. Thus, the ID of $RSS5$ is 5. Moreover, the key of a prefix is the index used in its label. For example, $P2$ has key 2. In figure 1, $RSS5$ receives all the routes for prefix $P2$. The advertisements concerning $P6$ and $P9$ are sent to $RSS10$.

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 is changed 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 1. 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. Here we show how the best route to prefix P1 will be selected among the four routes that the AS has learned. Each column represents a BGP route candidate identified by the NH of the route. Each line represents an attribute of the route that is compared based on each of the six rules of the DP. The last column shows the routes that remain after each rule of the DP. In the first four process, no routes were evicted since the values are the same.Routes with Rd and Rc as Next-Hop (NH) are eliminated at the 5th rule of the DP. 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 based on the 6th rule, Rb has a higher ID than Ra. Thus, it is eliminated and Ra is selected as best route for $R7$, $R8$ and $R9$.

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 [13] 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

Table 1 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

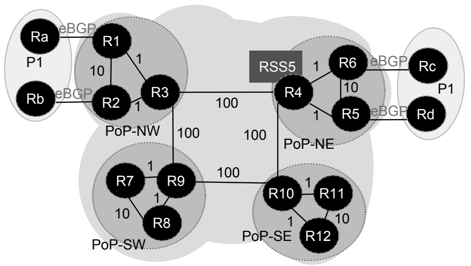
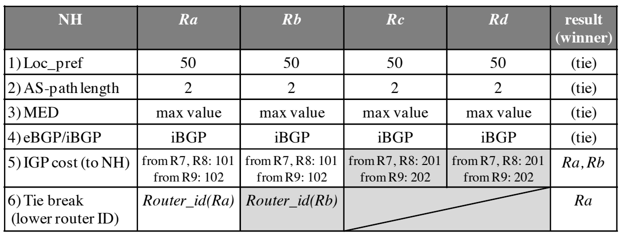


Fig. 2 Reference network.



NH	*Ra*	*Rb*	*Rc*	*Rd*	result (winner)
1) Loc_pref	50	50	50	50	(tie)
2) AS-path length	2	2	2	2	(tie)
3) MED	max value	max value	max value	max value	(tie)
4) eBGP/iBGP	iBGP	iBGP	iBGP	iBGP	(tie)

5) IGP cost (to NH)	from R7, R8: 101
	from R9: 102
		from R7, R8: 101
		from R9: 102
			from R7, R8: 201
			from R9: 202
				from R7, R8: 201
				from R9: 202
					Ra, *Rb*

6) Tie break (lower router ID)	*Router_id(Ra)*	*Router_id(Rb)*			*Ra*

Fig. 3 Traditional route selection.

be selected. Otherwise, all the routers of the PoP select the same route, learned on an eBGP session in a different PoP. In this example of the route selection for the packest arrived at the routers in the PoP-SW, all the routers select the same route, Ra as the NH for $P1$. Similarly, all the routers in "PoP-SE" select the route with Rc as NH, according to the traditional route selection process.

In our proposal, route selection is done at the RSSs 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 RSS to select the same route for all the routers in the PoP. The RSS 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 RSSs to identify the nodes of a PoP.

First, there are no changes concerning the rules 1 to 3, in table 1. 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 selection 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 [13], 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.

Figure 4 illustrates the per PoP route selection per-

Route selection done by RSS for routers in PoP-SW (R7, R8, R9) for P1

NH	*Ra*	*Rb*	*Rc*	*Rd*	result (winner)
1) Loc_pref	50	50	50	50	(tie)
2) AS-path length	2	2	2	2	(tie)
3) MED	max value	max value	max value	max value	(tie)
4) Direct connection to the PoP	n/a	n/a	n/a	n/a	(tie)
5) IGP cost(to NH)	from R7: 102	from R7: 102	from R7: 202	from R7: 202	*Ra*, *Rb*
6) Tie break (lower router ID)	*Router_id(Ra)*	*Router_id(Rb)*			*Ra*

Route selection done by RSS for routers in PoP-NW (R1, R2, R3) for P1

NH	*Ra*	*Rb*	*Rc*	*Rd*	result (winner)
1) Loc_pref	50	50	50	50	(tie)
2) AS-path length	2	2	2	2	(tie)
3) MED	max value	max value	max value	max value	(tie)
4) Direct connection to the PoP	available	available	n/a	n/a	*Ra*, *Rb*
5) IGP cost(to NH)	from R2: 2	from R2: 1			*Rb*

Fig. 4 Per PoP route selection.

formed at route sele[illegible] server $RSS5$ for prefix $P1$ in the network of Figure 2. Here we show per PoP route selection for PoP "PoP-SW" and "PoP-NW". In the route selection for "PoP-NW", first elimination of the candidate happens with the 4th rule. Ra and Rb are directly connected to the routers in the PoP, while the others are not. Finally, Rb remains to be the best route to P1, by the 5th rule.

As it is already widely recommended in traditional iBGP topologies, in order to avoid forwarding loops[14], 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 RSSs 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 RSS 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.

3.2 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 5 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 [15] and [16].

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. [16]. 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 [17] 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 [16].

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

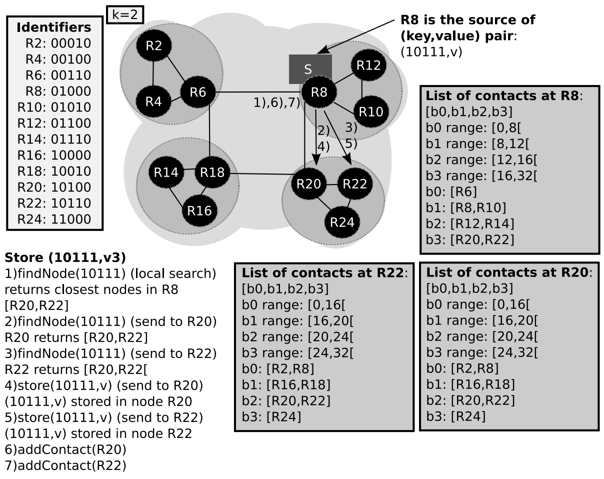


Fig. 5 Store a (key,value) pair in Kademlia.

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 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$\underline{0}$ 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 RSS 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 are 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:

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 ($P \neq \emptyset$) then
10:            $r$=mostSpecificNewestRoute($P$, $r$)
11:            installInFIB($r$)
12:        end if
13:    until ($N_{new} \subseteq N$)
14:    $k_d$=resetLastIPAndMaskNonZeroBits($k_d$)
15: until (defined $r$)
16: replicate($r$)
```

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

$< PoP_{id} >$ 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.

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 matching routes" is identified. From this set, the

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 ($N_{result}, P_{result}$)
```

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 RSS computes a route, it attaches a version number to the route. Popular routes are replicated in the DHT. When the RSS 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 RSSs while ensuring full visibility of the routes for a prefix at a RSS. Together with encapsulation, this ensures correctness [5]. Then, we presented our modifications to Kademlia in order to support the distributed storage of IP routes and their retrieval upon packet arrivals. Now, we will describe our prototype implementation. We describe the functions that we implemented in the different nodes, the RSSs and the ASBRs. We show the feasibility of our proposal.

4. Prototype Implementation

In this section, we show how we have implemented the required functionalities in the RSS and ASBR prototypes. Running example of the prototype will be shown later in section 6.

4.1 Software architecture

The prototype consists of RSSs and the RSS clients. RSS clients are ASBRs that are capable of the protocol proposed in this paper. Some of the routers may work as an RSS and an ASBR at the same time.

As shown in Fig. 6, RSSs and the ASBRs are composed of two modules that we have developed: the BGP module and the DHT module. The BGP module is a slight modification of an existing open source routing software, Quagga (ver. 0.99.11) [18]. The DHT module is developed based on one of the existing open source implementation of Kademlia, Entangled (release 0.1) [19].

Usually, network entities negotiate which protocol to use between them by advertising the functions they are capable of to their neighbor. In our prototype, we use the capability feature of the BGP OPEN message to negotiate the

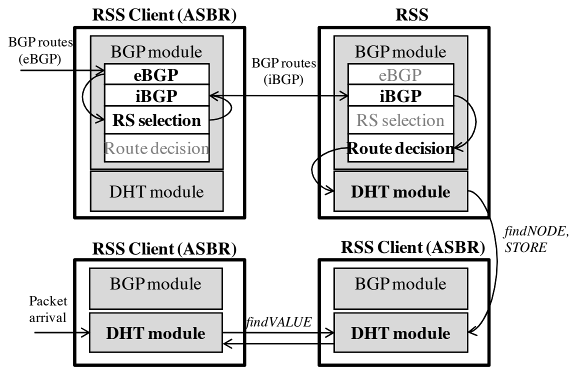


Fig. 6 Prototype architecture: RSS and RSS Client modules

"RSS" and "RSS client" capabilities [20]. We assign the code 250 to the "RSS Capability" and code 251 to the "RSS Client Capability". The RSS ID and the PoP ID are set in the 32 bit field of the capability value of RSS Capability and the RSS Client Capability, respectively.

Network operators can choose which nodes to launch as an RSS or as an ASBR via command line or configuration files. The PoP ID and the RSS ID, in the case of an RSS, are set by the operator.

Here we describe how the functionalities of RSSs and ASBRs are achieved by the development of BGP and DHT module.

4.2 Implementation of a RSS

An RSS sets up iBGP sessions with all the ASBRs in the AS in order to collect all the BGP routes that the RSS is responsible for. The Quagga implementation of the ASBR is changed to relay BGP routes to the right RSS. ASBRs select the RSS responsible for the prefix advertised and simply send the BGP advertisement to this RSS. There is no BGP route selection at the ASBR anymore.

An RSS computes routes for each prefix, as is explained in 3.1. Per-PoP route selection is performed by the RSS. Then, the routes are delivered to the ASBRs of the PoP. This is done by storing the routes in the DHT. The ID of the ASBRs and the key of the routes ensure storage of the routes in their respective PoP. After the BGP module in the RSS carries out the per-PoP route calculation, it handles the generated route information to the DHT module running in that RSS in order to distribute it to the ASBRs. In our prototype, the external BGP routes are generated in a file, and the DHT module of the ASBRs periodically checks if the file has been updated. The DHT module of the RSS delivers the route information to some ASBRs of the appropriate PoP through the Kademlia's "$findNODE$" and "$STORE$" procedures. This does not require changes to the store implementation of Entangled.

4.3 Implementation of an ASBR

The BGP module provides the capabilities for an ASBR to set up eBGP sessions with peer ASBRs outside of the AS. This is already provided by Quagga. However, we changed the following functions of the Quagga implementation, to support the distributed route calculation at the RSSs.

When an ASBR learns a BGP route on the eBGP session, it determines the RSS in charge of the prefix and it simply forwards the message to this RSS. ASBRs do not run the BGP path selection, and they do not store the route in the FIB. There is no FIB in the ASBRs anymore, only a cache with the routes currently in use for packet forwarding. They do not even create the adj-RIB-in/out database. ASBRs only transfers the messages received from the peers to the RSSs, and vice versa for external BGP route distribution.

The support of iBGP sessions between ASBRs and RSSs is also provided by the standart Quagga implementation. As mentioned in 3.1, RSSs are assigned an ID. The ID is used when the ASBRs determine which RSS the BGP route should be sent. In the implementation, we defined the RSS ID as being a 32 bits IPv4 address. The decimal representation of prefix keys and RSS IDs is used to compare them, in order to identify the RSS in charge of a given prefix. For example, an RSS ID $A.B.C.D$ will be converted to a decimal number $A{*}256^3{+}B{*}256^2{+}C{*}256{+}D$. According to the result of this calculation, ASBRs sends the BGP route to the selected RSS. Note that path selection is made by the RSSs. Therefore, the attributes of the BGP routes are transmitted unchanged to the RSS.

4.3.1 Distributed routing tables

Ingress ASBRs forward packets that are received from neighboring ASs to the egress ASBRs of the AS. As explained above, routing information for packet forwarding is computed by the RSSs and stored in the DHT by the RSSs. The set of routes for the ASBRs in a PoP will be distributedly maintained among the ASBRs of the PoP. We use a radix tree as the data structure, in the DHT module. The route key k_r is composed of the identifier of the PoP for which the route was selected, the route's prefix and its mask. This key is used to determine the set of nodes in the DHT that will store the route for the PoP. These nodes are in the PoP. The value stored for each route's key contains the Next-Hop (NH) of the best route (i.e. the address of the egress ASBR for the best route) for that prefix, the NH of the second best route, the last publication time, the original publication time and, the node ID of the RSS who published the route. The original publication time indicates the time when the route was selected. Thus, when different routes were found for a destination prefix, it will be used to determine which route is the newer selection that should be used. The persistence of the key-value pairs is ensured through periodical re-publication of the pairs by the RSSs. The last publication time corresponds to the latest re-publication. This is provided by the original Kademlia implementation.

When a packet arrives at the border of the AS, the DHT module of the ASBR retrieves the routing information from the DHT. We modified the implementation of Entangled to perform our route retrieval algorithm (Algorithms 1 and 2).

Table 2 Items of statistics

BGP module	Memory consumption for each information
	Number of iBGP sessions
	Number of iBGP messages sent per session
	Number of total iBGP messages sent/received
	Number of UPDATE messages on iBGP session
	Number of UPDATE messages on eBGP session
	Route decision result
DHT module	Number of sent RPCs for each DHT procedure
	Elapsed time for each DHT procedure
	Number of routes maintained (total and per radix tree)
	Memory size consumed to maintain the routes (total and per radix tree)
	Number of entries in the k-buckets
	Memory size consumed by k-buckets

Modifications were required to enable the retrieval of the route for the most specific prefix that matches a packets destination, instead of a strict match provided by standard Kademlia implementations.

4.4 Measurement functions

Our aim is not only to prove that our concept is feasible and implementable, but also to carry out measurements in the future. We thus have added to our prototype a capability to output various items of information that are recorded or measured while running the system. The generated statistics show us important performance criteria that can be used for the evaluation of the proposed architecture. Items of statistics that the prototype can show are listed in table 2. Note that we modified Quagga to output these statistics for both our prototype implementation and for the regular BGP behavior. For example, because the prototype monitors the “memory consumption” in the RSS, the RSSs clients and the standard BGP implementation, we are able to know and compare the memory size required for routers running the original BGP, and those running our proposed protocol.

Measurements are important to validate our proposal. Since in our proposal ASBRs do not maintain locally a route for every prefix, they need some time to retrieve the route from the DHT. Thus, packet might be forced to wait until the ASBR retrieves the route for that packet’s destination. As the retrieval time in DHT depends on the number of peers, this issue may be critical to ensure good forwarding performance with a large number of ASBRs in the PoP. In the future, we plan to evaluate the importance of this issue by using our prototype implementation.

In this section, we have shown the implementation of the prototype. The prototype was implemented in order not only to make a proof of the concept, but also to evaluate and validate our proposal in terms of performance. In the next section we will give analytical evalutation that clarifies how much we can benifit from our proposal. Running example and sample measurement results are addressed in section 6.

5. Scalability analysis

In this section, we compare traditional iBGP route distribution and the Virtual Aggregate (VA) solution proposed by Francis et al. [9] 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.

Table 3 Variables and example value for each notation.

n	200 (20)	number of nodes in the AS
p	320000	number of external prefixes
q	16 (3)	average number of iBGP peers per node (in sparse iBGP topology)
r	3 (1.9)	average number of eBGP peers per ASBR
s	20 (3)	number of RSS
a	160 (20)	number of ASBRs
l	20 (3)	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 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 RSSs 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 3. We assign a value to each variable. These are the values used for the generation of figures 7 and 8. These values are only provided as an example of the values that may be observed in a real SP network. Numbers in the brackets are used for evaluation assuming a much practical case with a sample topology modeling an network that is actually in service. Evaluation is addressed in section6.

We computed the average number of iBGP peers q in table 3 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 [21]. m encompasses the eBGP churn. The value assigned to m in table 3, 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 assumed according to the findings of Iannone et al. in [15]. The authors assumed that the Internet traffic at the bound 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

Table 4 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 RSS)	$(p/s)(ar+2l)$	–
DHT table sizes (SpliTable)		
	DHT table	Routing cache
At ASBR	$pkl/n+(a/l)dl/n$ $+(a/l)(1440/t)cl/n$	d
At RSS	$pkl/n+(a/l)dl/n$ $+(a/l)(1440/t)cl/n$	–

entry from the DHT up[illegible]acket arrival. Once the route entry is received it is st[illegible]n a cache and can be used to route subsequent packets. As the study in [15] analyses the packet trace collected at the egress point of a campus, this can be seen as a behavior of a single stub network from the SP point of view. Thus, if we assume a simplified model that one stub network is connected to each of the ASBRs, we can rely on the numbers observed in[15] for assigning numbers of d and c, the destination addresses arrived and cache misses at an 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 [22], 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.

5.1 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 4 the formulas used to compute the number of routers stored in the nodes of an AS for each of the techniques.

We see in table 4 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.

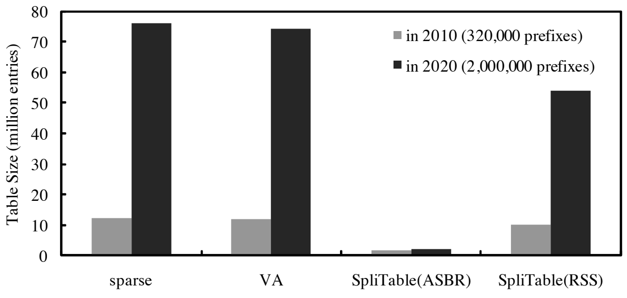


Fig. 7 Comparison of the table sizes.

In SpliTable, only the RSSs 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 RSSs 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 are in a PoP, the less entries each node has to maintain, assuming a constant number of Internet prefixes.

Figure 7 illustrates the total number of routes stored in a node for sparse iBGP topologies, VA and SpliTable. We set the number of propagated prefixes (p) observed in [23] at present (in year 2010) and that of an assumed value based on the prediction which says a router could have to hold 2 million FIB entries in about ten years later [1].For other values we use them listed in table 3. We see that the gain of VA is very limited. 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 RSS with our proposal corresponds to 82% and 71% of the routes maintained in a node in the sparse iBGP topology, in 2010 and 2020, respectively. At the ASBRs, this number drops to 14% and 3%. We can see that with our proposal, the number of entries held inside the nodes can be dramatically reduced. The effect will be much significant when we consider the future Internet. 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.

5.2 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 [5] for an example of such a situation.

Table 5 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	$24m$	$48ml(min((n/l) +k, 129k))$	$a(d+(1440/t)c) *(2*min(n/l, 24*128k)+1)$

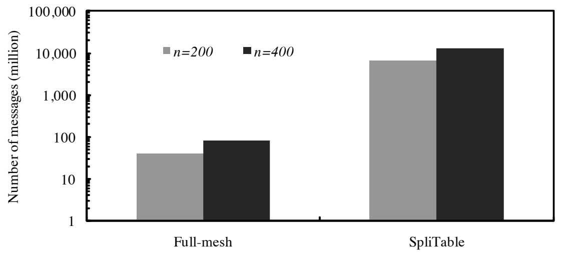


Fig. 8 Comparison of the number of control messages per day.

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 selection servers (first column in table 5), to store, update and remove entries from the DHT (second column) and to retrieve the routing entries from the DHT (third column). In table 5, we show the upper bound on the number of messages that may be exchanged in one day in an AS.

Figure 8 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 the number of control messages. Convergence of our route distribution solution is ensured.

We see in figure 8, 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 nodes that maintain the requested key are found after a very short exchange of messages. Actually we have verified this through the prototype evaluation addressed in the next section. The average number of DHT messages that was counted for the route retrieval upon a packet arrival with no available cache was 3.4, where the number of nodes in the PoP is 9. This is far below the 21 messages that is counted in the analysis of the upper bound.

In kademlia, when a node maintains all the contact information of the nodes inside a PoP, it can directly access the nodes with the k closest IDs to the key. The key should be found in these nodes, since those are selected to be the node to store the key, by having the k closest IDs to the key. The only case the requesting node needs to make multiple cycles of searching is when the k-bucket (a list of nodes that has the same XOR metric index) of that node was full, and was unable to store the entry of a node who is really the k closest to the key. This hardly happens when we use Kademlia in SpliTable, since the number of nodes in a PoP is assumed to be at most several tens. k-bucket has a capacity to maintain 160 * k nodes.

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 3. 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.

In this section, we have analyzed the scalability benefits that our proposal can give. We showed that it can give significant benefit in terms of number of routing entries that a router should maintain. Also we showed it has an upper bound of the number of messages in contrary to the common configuration of sparse iBGP where the number of messages can be infinite. However, number of messages and the time to retrieve a route in the DHT are issues that are very difficult to examine by calculation. Implementation of the prototype is done not only for the proof of concept, but also for the purpose that we should evaluate these factors emulating a realistic topology of an AS in the real world. Next section shows that the proposed routing architecture has worked successfully with providing measurement results.

6. Prototype evaluation

In this section, we address the proof of concept of our proposal by running the prototype implementation which described in section4. For evaluation assuming a practical use in SP network, we have configured a single AS topology imitating a network that is actually deployed and in service. Furthermore, we have injected real BGP routes collected in the Internet. We show the performance evaluation in terms of number of routes stored in the nodes, number of DHT messages and the time required for the retrieval of the route in case it was not found in the cache. These results are shown in order to give support to the scalability analysis addressed in section5 from the aspect of assuming a practical case.

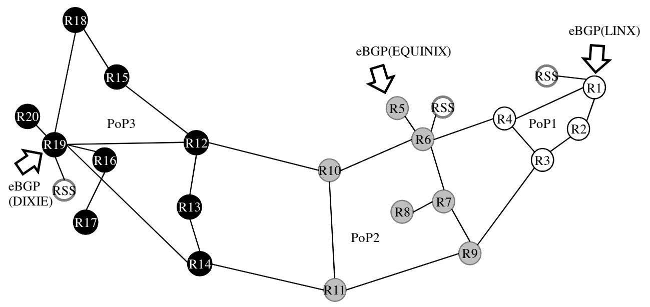


Fig. 9 Sample AS network topology consists of 20 ASBRs and 3 RSS.

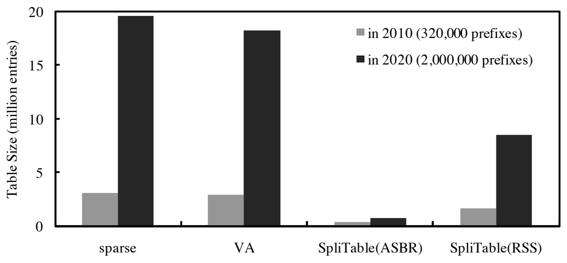


Fig. 10 Comparison of the number of table sizes of the routers in the sample topology.

6.1 Sample Deployment of the Prototype

The RSS and ASBRs are installed on a single Linux system. Linux systems are actually a virtual machine generated upon a single server hardware using Xen [24]. We use CentOS 5.2 for the Linux, and the whole system is running on a server which has 2 quad core processors and 8 GBytes of memory.

As shown in Fig.9, we set up 23 nodes. 3 of these nodes act as RSS; 20 act as routers. The routers are connected to model the layer-3 topology of ESnet[25]. They form the topology of a single AS. They are divided into three PoPs. In the figure, each PoP is in a different shade of grey.BGP routes are injected on the eBGP sessions from independent servers running a BGP emulation software. We used BGP routes collected by the routeviews project at three IXs: DIXIE, EQUINIX, and LINX in midnight on Jan. 1st, 2010. Through those IXs, we can emulate 4, 9, 26 eBGP sessions, respectively. For each IX, we used a single file containing updates collected within 15 minutes. As a result, about 7,800 unique IPv4 prefixes remained in the DHT.

6.2 Scalability Analysis with the sample topology

As done in the previous section, we have calculated the table size of the routers and RSSs when each technique were deployed in the sample topology. Parameters depends on the topology has changed from the calculation done in the previous section. They are also shown in table4 with brackets. We can see the scalability benefits of adopting SpliTable, as shown in Fig.10.

Table 6 Measurement results

	N_m			T_r [msec]		
	average	max	min	average	max	min
R3 (PoP1)	2.01	7	1	1.95	6.67	0.82
R12 (PoP3)	3.40	9	1	2.95	6.70	0.83

6.3 Measurement results

We have injected test packets into routers in two PoPs that has the largest (PoP3: 9 nodes) and smallest (PoP1: 4 nodes) number of nodes. Since we aim to verify the required number of messages and the time to retrieve routes only in the case that the packet arrivals trigger search in the DHT, i.e. the routes are not found in the cache, destinations of the packets are assigned to be unique and not to overlap between each other. We have injected 576 packets to R3, and 269 packets to R12.

We confirmed that the system successfully worked. RSSs had calculated the routes and delivered it to the ASBRs in the PoPs. Route information had been retrieved by DHT process among the ASBRs in a PoP. Measurement results are shown in table.6. N_m represent the number of DHT messages sent upon packet arrival, andT_r represent the time. We show the average, maximum, and the minimum for them. Results indicate that the time to retieve a route below is 3 [msec] in average, and it is larger when the number of nodes in the PoP is larger. In PoP1, almost all of the route retrieval had completed with 5 or 1 message(s). Only one of them required 7. With $k = 2$, route retrieval with 5 messages indicates it has needed only one cycle of search. It contains a request and a reply message between the k closest nodes, and a store message in order for the replication. That with 1 message indicates that the route was found locally and it sent only the store message. This fact proves the performance of the Kademlia's search mechanism based on XOR metric.

7. 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 Selection Servers (RSSs) for the selection of routes. We proposed a way to distribute the load on multiple RSSs. 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 (RSS or ASBR), they only maintain 82% or 14% 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.

We have also shown a prototype design and an experimental deployment of our proposal. We confirmed the proposal had successfully worked, and through the experiment we retrieved some measurement results such as number of messages and time required for a retrieval of the route in case the destination was not found in the router's cache.

Our future work using the prototype which we described in this paper will focus on the evaluation regarding the overall number of control messages and time to resolve a route.This is expected to be done by taking consideration of the performance of the cache, using packet traces collected in the Internet.

Acknowledgments

We would like to thank Akinori Isogai at NTT Network Service Systems Laboratories for his support to this work in the prototype evaluation.

References

[1] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," RFC 4984, September 2007.

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

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

[4] 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," ACM SIGCOMM 2006, September 2006.

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

[6] C. Pelsser, A. Masuda, and K. Shiomoto, "Scalable Support of Interdomain Routes in a Single AS," IEEE GLOBECOM 2009, 2009.

[7] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)," January 2010. Internet draft, draft-ietf-lisp-06.txt, work in progress.

[8] 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.

[9] P. Francis, X. Xu, H. Ballani, D. Jen, R. Raszuk, and L. Zhang, "FIB suppression with Virtual Aggregation," Internet Draft, draft-ietf-grow-va-02.txt, work in progress, March 2010.

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

[11] L. Mathy and L. Iannone, "LISP-DHT: towards a DHT to map identifiers onto locators," CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference, New York, NY, USA, pp.1–6, ACM, 2008.

[12] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: routing on flat labels," SIGCOMM Comput. Commun. Rev., vol.36, no.4, pp.363–374, 2006.

[13] 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.

[14] 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.

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

[16] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," Proc. Internet Measurement Workshop, 2002.

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

[18] "Quagga Routing Software Suite." `http://www.quagga.net`.

[19] "Entangled - DHT and Tuple Space based on Kademlia." `http://entangled.sourceforge.net/`.

[20] R. Chandra and J. Scudder, "Capabilities advertisement with BGP-4," RFC 3392, November 2002.

[21] R. Zhang and M. Bartell, BGP Design and Implementation, first ed., Cisco Press, December 2003.

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

[23] "BGP Routing Table Analysis Reports." `http://bgp.potaroo.net/`.

[24] "Xen hypervisor." `http://www.xen.org/`.

[25] "Energy Sciences Network." `http://www.es.net/`.

Akeo Masuda received the B.S degree from the University of Tokyo, Japan, in 1997. He received the M. Science and Ph. D degree from Waseda University, Japan. He is with the Network Service Systems Laboratories in NTT Corporation since 1997, and is a visiting researcher at Waseda University. His research interests include IP-QoS, optical networks, network virtualization, inter-domain routing, and wireless access protocols. He is a member of IEICE.

Cristel Pelsser received her Master degree in computer science from the FUNDP in Belgium in 2001. She then obtained her PhD in applied sciences from the UCL, in Belgium, in 2006. From 2007 to 2009, she held a post-doctorate position at NTT Network Service Systems Laboratories in Japan. She is now a researcher at Internet Initiative Japan (IIJ). Her current research interests are in Internet routing and web applications scaling.

Kohei Shiomoto is a Senior Research Engineer, Supervisor, Group Leader at NTT Network Service Systems Laboratories, Tokyo, Japan. He joined the Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan in April 1989. He has been engaged in R&D of high-speed networking including ATM, IP, (G)MPLS, and IP+Optical networking in NTT labs. From August 1996 to September 1997 he was a visiting scholar at Washington University in St. Louis, MO, USA. Since April 2006, he has been leading the IP Optical Networking Research Group in NTT Network Service Systems Laboratories. He received the B.E., M.E., and Ph.D degrees in information and computer sciences from Osaka University, Osaka in 1987 1989, and 1998, respectively. He is a Fellow of IEICE, a member of IEEE, and ACM.