Prototype Design for Scalable Support of Interdomain Routes in a Single AS

Akeo Masuda
NTT Network Service Systems Laboratories
NTT Corporation, Japan

Cristel Pelsser
Internet Initiative Japan*

Kohei Shiomoto
NTT Network Service Systems Laboratories
NTT Corporation, Japan

***Abstract*—In this paper, we show a prototype implementation for a new architecture of supporting interdomain routes. It is widely recognized that the rapid growth of Internet is forcing a scalability bottleneck to itself from the aspect of routing. 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 DHT so that the routers are freed from maintaining the full route in the Internet. Our new routing architecture tackles the scalability issue by reducing the number of routes needed to maintain in each router, and the number of control messages exchanged. We have developed a proof-of-concept prototype by designing modules that compose route servers and ASBRs that work in our proposed architecture. It will be used to confirm the improvement mentioned above through performance evaluation.**

I. Introduction

Recently, people started to discuss about designing the future Internet architecture through a clean slate approach [1], [2]. One of the major issues concerning today's Internet is the scalability of its routing protocols. In the last two decades, the number of routes to be supported by the routers has become very large. In September 2008, there were $125,125$ IPv4 entries inside the routeviews tables [3]. Moreover, predictions say that this number will continue to increase in the future. One research predicts that a router could have to hold 2 million FIB entries in 2021 [4]. In addition to the number of routes, the number of messages exchanged to distribute the routes has increased even faster. In [5] it is predicted that the routing entries may grow by a factor of two while the amount of BGP advertisements will increase by a factor of four, in five years time. The large number of routes and messages to be processed leads to the performance degradation on BGP convergence [6]. Therefore, Internet service providers may face a serious problem concerning large cost for network equipment and poor availability of the service.

Authors have previously proposed a novel routing architecture which can be deployed in a single AS to reduce the number of routing entries and the number of BGP messages exchanged between the routers in the AS, without forcing changes to current BGP protocols that are used between ASs [7]. Our solution is based on a simple principle of distributing the route calculation process and the route table to multiple entities. Distributed route servers (RS) calculate routes on behalf of the ASBRs. For this purpose, ASBRs relay eBGP messages to the RS who is responsible for the prefix indicated in the message. ASBRs do not maintain the database to store BGP routes learned or sent on the eBGP sessions. For a given destination prefix, RS selects the same route for all the ASBRs of a Point of Presence (PoP), and stores it to the Distributed Hash Table (DHT) that the ASBRs in the PoP are joining. This lets each ASBR of a PoP to maintain the portion of the Internet routes. In case if there is no entry for the arrived packet 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 have developed a proof-of-concept prototype that runs the proposed interdomain routing protocol. The prototype consists of RSs and RS clients. RS clients act as the ASBR to relay the eBGP messages learned by eBGP sessions, and resolve the route by means of DHT for forwarding the arrived packet. We have designed the ID/key structure and searching mechanism in the DHT to maintain information of interdomain routes. We aim to use this prototype to measure the improvement that our proposal gives, through evaluation concerning realistic conditions by using collected Internet routes and packet trace. This paper describes the design and implementation of our prototype. We also show some of the running examples retrieved from a deployment of a sample network.

In the next section, we describe the design of the distributed routing table. In section III, we present how we have implemented the required functions to the prototype. In section IV, we show the experimental deployment and show some running examples of the prototype. Finally we conclude the paper in the last section.

II. Design of the Distributed Routing Table

In this section, we describe the design of the distributed routing table. Fig. 1 shows how the BGP messages and the calculated route are stored and retrieved in our proposal. Our distributed routing table is achieved mainly by introducing two concepts: the per-PoP route selection and the route resolution based on distributed information storing/searching functions of DHT.

A. Per-PoP Route Selection

Current ASBRs calculate routes by themselves since the optimal route for a same destination may differ among the

*This work was done while the author was working at NTT.

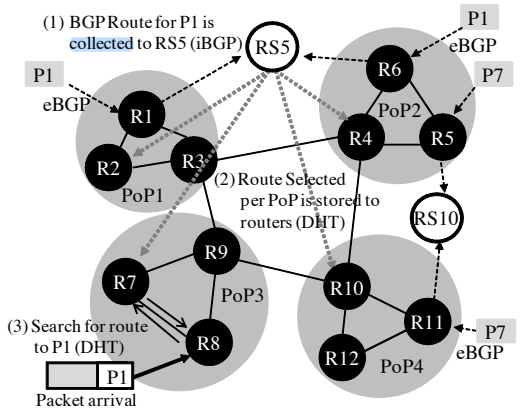

Fig. 1. Overview of the proposed protocol.

ASBRs in the AS. In our proposal, routes are calculated by RSs on behalf of the ASBRs. However, RSs do not calculate routes for each ASBR in the AS. Instead, for a given prefix, RSs calculate one route for all the ASBRs in a PoP. We call this a "per-PoP route selection". A PoP is a set of ASBRs that are present in the same location, for example in the same city or the same building.

In the common design of the currently deployed service provider's networks, the IGP cost of inter-PoP links is usually higher than intra-PoP link costs [8]. This ensures that traffic originated in and destined to the same PoP stays in the PoP. For this, inter-PoP path is consistent for all the ASBRs in a PoP. Thus, all the nodes in the PoP can use the same routes, and selecting one route per-PoP is enough for the RS. Furthermore, this enables the routing table to be shared among ASBRs in a PoP by way of DHT.

B. Route resolution based on DHT

Each ASBR of the PoP maintains only a portion of the routing table. When an ASBR needs a route that is not present locally, it requests the route from another node in its PoP. Since all the ASBRs in a PoP are in the same location, retrieving a route from a neighboring node is fast.

In our proposal, routes are stored in a DHT. DHT provide a framework for the distribution of the routes on multiple nodes. We choose to use Kademlia [9] due to the following properties of this DHT:

1) The search functionality in Kademlia is based on the XOR metric. As we will show in the next 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 [10].

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.

We observe that the XOR metric is appropriate for IP destination and prefix comparison. IP packets are 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 use the procedures of DHT for RSs and the ASBRs to work in our proposal. $findNODE$ is used for RS to find the appropriate ASBRs to deliver the calculated per-PoP routes. $STORE$ is used for RS to push those routes to ASBRs found by $findNODE$. $findVALUEandNODE$ is used for ASBRs where a packet has arrived to retrieve the route from other ASBRs in the same PoP. Here, $findVALUEandNODE$ is our modification to the $findVALUE$ procedure in order to let the ASBRs continue searching routes to look for the longest matching prefix.

After the route is retrieved for the arrived packet, the packet is encapsulated with the destination address of the egress ASBR in the AS. This enables the packet to be forwarded to the egress ASBR by IGP routing. This enables that ASBRs in a PoP do not need to maintain or retrieve interdomain routes of other PoPs for forwarding packets.

C. Discussions

Route table distribution that is enabled by these mechanisms provides scalability in terms of number of routing entries maintained in the routers. In [7] we have shown an analytical study on the number of entries required at the routers in each routing architectures. According to the numerical examples where the number of ASBRs in a PoP is assumed to be 20, ASBRs in the proposed architecture only needs to maintain 0.8 million entries, while legacy ASBRs with sparse iBGP topology needs to maintain 7.6 million entries.

However, since some of the routing retrieval depends on DHT, arrived packet might be forced to wait until the ASBR retrieves the route for that packet's destination. Since the retrieval time in DHT depends on the number of peers, this issue may be critical to ensure forwarding performance in some paticular configuration of the number of ASBRs in the PoP. For this issue, we plan to make measurements using a prototype implementation of the proposed protocol.

III. Prototype Implementation

In this section, we describe how we have implemented the required functionalities in the prototype.

A. Software architecture

The prototype consists of RSs and the RS clients. RS clients are the ASBRs in the PoPs that are capable of the proposed protocol. Some of the routers may work as an RS and an RS client at the same time.

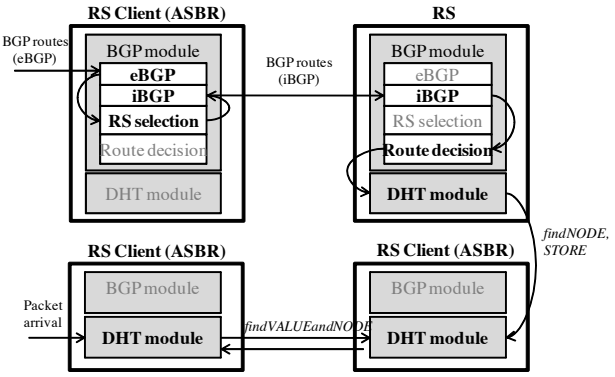

Fig. 2. Modules compose RS and RS Client

Usually, network entities negotiate what protocol to use between them by informing the functions they are capable. In our prototype, we use the capability feature of the BGP OPEN message to negotiate the "Route-Server" and "Route-Server client" capabilities [11]. We assign the code 250 to the "Route-Server Capability" and code 251 to the "Route-Server Client Capability". The RS ID and the PoP ID are set in the 32 bit field of the capability value of Route-Server Capability and the Route-Server Client Capability, respectively.

As shown in Fig. 2, RSs and the RS clients are composed of two modules that we have developed: the BGP module and the DHT module. Users can choose which to launch, as an RS or as an RS client in the PoP, via command line or configuration files. PoP ID and the RS ID (if it is a RS) are set by the parameter when it is launched.

The BGP module is developed based on the existing open source routing software, Quagga (ver. 0.99.11).The DHT module is developed based on one of the existing open source implementation of Kademlia, Entangled (release 0.1).

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

B. Implementation of RS

1) Collection of interdomain routes: RS receives the BGP route from ASBRs with the iBGP capability enabled by the function in Quagga. In our prototype, an RS needs to set up iBGP sessions with all the ASBRs in the AS. This is for collecting all the BGP routes that the RS is responsible.

2) Per-PoP route selection: RS computes the route for each prefix in the notified BGP route. For a given destination prefix, RS computes two best routes for each PoP. When a route is sent to the RS, it starts the BGP path decision process as the traditional BGP routers. Modification in this process is to enable "per-PoP route calculation". Since RSs calculate routes on behalf of the ASBRs, it concerns the location of the ASBRs. For this, we change the route decision rules implemented in Quagga. 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 ASBR in the topology. These rules ensure hot-potato routing. We have changed the 4th rule of the BGP decision process to: "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 [8], 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. According to this decision rules, the RS selects two best Next-Hop (NH) for the packet arrived at the ASBRs. Selecting second best NH in addition to the best NH is to allow fast restoration and load balancing of the traffic from a PoP on multiple ASBRs.

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

3) Delivery of the calculated routing information: Route calculated by the per-PoP route selection is delivered to the appropriate ASBRs in each PoP through the DHT store procedure. After the BGP module in the RS carries out the per-PoP route calculation, it handles the generated route information to the DHT module running in that RS in order to distribute it to the RS clients. In our prototype, the route information is generated as a file, and the DHT module periodically checks if the file is updated.

DHT module in the RS delivers the route information to appropriate ASBRs in each PoP through the "$findNODE$" and "$STORE$" procedure in Kademlia. This does not require changes to the store implementation of Entangled.

Nodes in Kademlia 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.

IDs and key are compared in order to determine the node to store the information, and search the node who stores the information. Criteria for determining which node to store the information is the distance of the ID and the key based on the XOR metric. Information will be stored in k nodes whose IDs are closest to the key of the information. k is called the replication parameter. In our prototype, RS uses this $STORE$ procedure to deliver the route information calculated for each PoP. RS finds k ASBRs that have the closest ID to the key of the calculated route information.

Distance based on the XOR metric is given as follows. We define Node ID and the key of a route information $k_r$as:

- $NodeID$::=$< PoPid >< hash >$
- k_r ::=$< PoPid >< prefix >< mask >< padding >$

$< PoP_{id} >$ identifies the PoP for which the route is destined. It is 32 bits long. $< hash >$ is 128 bits long. $< hash >$ is obtained by applying MD5 on a randomly generated number. $< prefix >$ 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.

Distance between two values based on XOR metric is indicated by the position of the first bit that differs. First, compare the bits starting from the left and stop once the two bits are different. Counting the position of these bits starting from the right gives the distance between the key and the node ID.

RS interacts with the ASBRs with the $findNODE$ RPC, and makes the $STORE$ RPC to the ASBRs that have the IDs that are the k closest. The procedure to store the information to the k closest nodes is not changed from the original Kademlia. 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.

Calculated route is encoded in the format of the value component as follows.

- $v ::=< version >< length >< NH_1 >< pref_1 > ... < NH_n >< pref_n >$

n corresponds to $< length >$. 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 our prototype, we consider $n = 2$.

C. Implementation of RS Client

1) Exchanging interdomain routes (eBGP): BGP module provides BGP capabilities for an ASBR to act as an RS client. RS client capable routers are able to set up eBGP sessions with peer ASBRs outside of the AS. eBGP capabilities are enabled by the functions of Quagga. However, for the concept of the distributed route calculation at the RSs in our proposal, we changed some functions of the Quagga implementation.

When the ASBRs learn BGP route on the eBGP session, it simply forwards the message. They do not run the BGP path selection, and they do not execute the update of FIB. They do not even create the adj-RIB-in/out database.

ASBRs only transfers the messages received from the peers to the RS, vice versa. It does not relay the messages from a peer to another peer. It neither relays the messages from an RS to another RS.

2) Forwarding interdomain routes to the RS (iBGP): RS clients that learn interdomain routes on eBGP sessions set up iBGP sessions with each RS in the AS. It forwards the eBGP message to the RS that is responsible for the destination prefix indicated in the message. Setting up iBGP sessions between ASBRs and RSs are also enabled by the function of Quagga. As mentioned before, RSs are responsible of the portion of the destination prefix space. Here we add a function to select the RS to send the BGP route. Prefixes are assigned to an 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 an RS. In the implementation, we defined RS IDs with a 32 bits IPv4 format. For the comparison, IDs may be converted into decimal numbers. For example, an RS ID $A.B.C.D$ will be converted to a decimal number $A * 256^3 + B * 256^2 + C * 256 + D$ and used for comparison. According to the result of this calculation, ASBRs sends the BGP route to the selected RS. Note that path decision is made by the RSs. Therefore, path attribute in the route information that ASBRs informs the RS should not be updated by the ASBRs.

3) Distributed routing tables: RS client can act as an ASBR to forward a packet that is injected to the AS. As explained above, routing information for forwarding the packet is generated and delivered from RSs. Set of routing information for the ASBRs in a PoP will be distributedly maintained among the ASBRs in a PoP. We use radix tree for the data store in the DHT module. Prefix and the mask in the k_r are used for the search key. Values stored for each key contains the best route (i.e. address of egress ASBR) for that prefix, second best route, last published time, originally published time, and the node ID of whom published it. Persistence of the key-value pairs are ensured by periodical re-publishment which provided by the original Kademlia function.

4) Routing information resolution for packet forwarding: When a packet to be forwarded was injected to the AS, DHT module of the RS client retrieves the routing information from the distributed data store in the PoP. This is basically provided by the existing functions of Entangled, but some of the procedures are added. First, if there is no routing entry found in its own data store, DHT module in the RS client searches its k-bucket, a list of contact nodes, to retrieve k nodes whose ID is closest to the key of the destination address of 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 as follows.

- $k_d ::=< PoPid >< DestinationIPaddress >< mask >< padding >$

$< PoP_{id} >$ 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.

The distance based on XOR metric is calculated the same way we explained above. Then it sends $findVALUE$ RPC to those k closest ASBRs. DHT module in the ASBRs that received this RPC will search for this key in its radix tree, and make a reply.

Since the key of the stored route and the key to look up for a route are different, we need to add some changes to the

original Kademlia's look up procedure based on the algorithm we have shown in [7], 1 and 2.

If the routing entry is found by its reply, the packet will be able to forward. In original Kademlia, nodes received $findVALUE$ RPC replies the node ID which is k closest to the key ONLY when it doesn't store the value to the key. For continuing search in the PoP in order to look for route information with much specific prefix, we change this process. In our prototype, nodes received $findVALUE$ RPC will reply the k closest nodes that it knows, even it has the value to answer. DHT module in the ASBR who sent the $findVALUE$ RPC will update its k-bucket according to the reply, and makes $findVALUE$ again to the nodes included in the k closest, if it has not been contacted yet. Therefore, we call this a $findVALUEandNODE$ RPC, instead of $findVALUE$ RPC.

In original Kademlia, in case no value was retrieved with the sequence of $findVALUE$, the node who requested the value will give up retrieving it. In our prototype, the ASBR where the packet arrived will continue searching with modifying the key. Each time the sequence of $findVALUEandNODE$ comes up with no result, it will change the last bit that is "1" at the destination address part in the key k_d.

Algorithm 1 retrieveRoute(k_d)

repeat
  {search for a less specific prefix if a matching route is not found for the initial k_d}
  N_{new}=N=a {first search in the local node}
  {Initialize N_{new} in order to not override N in the first execution of the loop}
  repeat
    {search for the most specific prefix for k_d}
    N=N_{new}
    (N_{new}, P)=findMatchingRouteInClosestNodes(N,k_d)
    if ($P \neq \emptyset$) **then**
      r=mostSpecificNewestRoute(P, r)
      installInFIB(r)
    end if
  until ($N_{new} \subseteq N$)
  k_d=resetLastIPAndMaskNonZeroBits(k_d)
until (defined r)
replicate(r)

Note that encapsulation and forwarding of the packet is not implemented in our prototype. Since this procedure relies on existing functions actually running in the routers, we believe it doesn't need to be shown that it is implementable.

5) Replication of the routing information: When an ASBR retrieves a value for the key, it stores the information not only in its data store, but also to other nodes that are desirable to store it. After the sequence of $findVALUEandNODE$ finishes, the requesting ASBR updates the k-bucket. If there were ASBRs that did not reply the value in spite that it is one of the k closest nodes, the requesting ASBR will send

Algorithm 2 findMatchingRouteInClosestNodes(N,k_d)

if ($|N| = 1$ **and** $N = a$) **then**
  N_{result}=getClosestNodes(k_d)
  P_{result}=getMostSpecificRoute(k_d)
else
  for ($n \in N$) **do**
    (N_{new}, P_{new})=sendFindRoute(k_d, n)
    N_{result}=$N_{result} \cup N_{new}$
    P_{result}=$P_{result} \cup P_{new}$
  end for
end if
return (N_{result}, P_{result})

$STORE$ RPC to make a replication of that information. Each stored routing entry has a timer. ASBRs refresh the timer when the information is used, and deletes the entry when the timer expires. This process works as an optimized cache that the entry that is frequently used will be maintained in many ASBRs. This leads to reduce the time required before forwarding the packet, by enabling the packet forwarding without the need of DHT message exchange.

D. Measurement functions

We have added to our prototype a capability to output various items of information that are recorded or measured while running the system. The major outputs are: trace logs, display of running configuration, and the measured statistics. Statistics show us the important values of performance for the evaluation of the proposed architecture. Items of statistics that the prototype can show are listed in table II. For example, by the item "memory consumption" we are able to know and compare the memory size required for routers running the original BGP, and those running our proposed protocol.

TABLE II
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

IV. EXPERIMENTAL DEPLOYMENT

A. Overview of the sample deployment

In this section, we describe a running example of the prototype. A RS or an RS client ASBR is deployed on a single Linux system. Linux systems are actually a virtual machine generated upon a single server hardware using Xen.We use CentOS 5.2 for the Linux, and the whole system is running

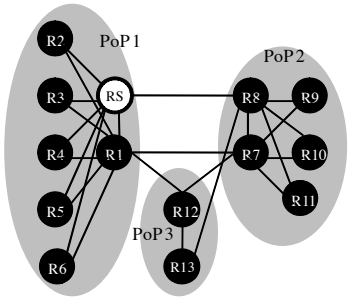

Fig. 3. Sample AS network topology consists of 13 ASBRs and an RS.

on a server which has two quad core processors and eight GBytes of memory.

As shown in Fig.3, 14 servers that act as an RS and 13 RS client ASBRs are set up. They are divided into three PoPs. BGP routes are injected on the eBGP sessions from independent servers running a BGP emulation software.

B. Measurement results

We have tested the prototype implementation described above by injecting 8 of sample BGP routes and also by emulating 15 packet arrival at the RS clients. We confirmed that the system successfully worked. RSs had calculated the routes and delivered it to the ASBRs in the PoPs. Route information had been retrieved by DHT process among the RS clients in a PoP. It sent average 3.0 $findVALUEandNODE$ RPCs within average 4.8 msec for the route retrieval for a packet forwarding. Some other measurement results are shown in Table. III. N_m, T_{avg} and $T_p$represent the number of messages sent, average and the 99% percentile for the time needed for each task, respectively. Messages include $JOIN$, $findNODE$, $STORE$, and $findVALUEandNODE$.

TABLE III
MEASUREMENT RESULTS

	N_m	T_{avg} (sec)	T_p
Join DHT	3.40	1.005	4.007
RS to client route delivery	7.95	0.039	0.080
Route retrieval	3.00	0.005	0.005

V. RELATED WORKS

For many years new architecture of the Internet has been discussed in order to liberate the routers from the load for interdomain routing. Concept of Locator-ID separation [12] aims to achieve the reduction of routing table size by separating the numbering space for locators from that for the end host IDs, and let the routing done only with the locator addresses. The difficulty of this approach lies in its deployment in the Internet. We cannot benefit from this approach until a large number of ASs at the border of the Internet adopt the solution. In contrast, our proposal gives scalability benefits immediately to the ISP even the deployment is limited to its AS. We believe this aspect is important considering the realistic, incremental deployment of a new Internet architecture. Compact routing [13] also is a solution for reducing the routing table size by aggregeating routes. This concept has a difficulty in terms of routing performance since it makes use of routes that are not necessarily the shortest path to the destination. RCP[14] is an interdomain routing architecture that lets one centralized node selects and provides the route for all the routers in an AS. This solution releases the routers from maintaining RIBs. Route selection at the route servers in our proposal resembles this approach, but it is different in terms of scalability that our route servers work distributedly in the AS. Our proposal of the distributed routing table is shown in [7], which addresses the protocol description, analytical evaluations that includes comparison between one of the major methods in compact routing. Based on this proposal, we have shown the design of a prorotype implementation in this paper.

VI. CONCLUSION

In this paper, we have shown a prototype design for scalable support of Internet routes in the routers that compose a single service provider network. We have shown how we designed the system in order to enable the features that our proposed protocol gives. Furthermore, we have shown a sample deployment using our prototype. For further work, we aim to evaluate our proposal using this prototype in a realistic condition emulating an AS in the Internet. Our future work using the prototype which we present in this paper will focus on the evaluation regarding the number of messages and time to retrieve a route via DHT. This is expected to be done by modeling a service provider network with a realistic topology, collected Internet routes and packet traces.

REFERENCES

[1] "Future INternet Design (FIND)," http://www.nsf.gov/pubs/2007/nsf07507/nsf07507.htm.
[2] "AKARI - Architecture Design Project for New Generation Network," http://akari-project.nict.go.jp.
[3] "Route Views Project," http://www.routeviews.org/.
[4] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," September 2007, RFC 4984.
[5] G. Huston and G. Armitage, "Projecting future IPv4 router requirements from trends in dynamic BGP behaviour," in *ATNAC*, Australia, December 2006.
[6] A. Feldmann, H. Kong, O. Maennel, and A. Tudor, "Measuring BGP pass-through times," in *PAM*, 2004, pp. 267–277.
[7] C. Pelsser, A. Masuda, and K. Shiomoto, "Scalable Support of Interdomain Routes in a Single AS," in *IEEE GLOBECOM 2009*, 2009.
[8] 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.
[9] 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.
[10] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proc. Internet Measurement Workshop*, 2002.
[11] R. Chandra and J. Scudder, "Capabilities advertisement with BGP-4," November 2002, RFC 3392.
[12] 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.
[13] D. Krioukov, k. c. claffy, K. Fall, and A. Brady, "On compact routing for the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 41–52, 2007.
[14] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe, "The Case for Separating Routing from Routers," in *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Portland, OR, September 2004.