Journal of Information Processing Vol.23 No.4 441–448 (July 2015)

[DOI: 10.2197/ipsjjip.23.441]

Regular Paper

Design and Implementation of Optimal Route Selection Mechanism for Outbound Connections on IPv6 Multihoming Environment

Yong Jin[1,a) Nariyoshi Yamai[2,b)] Kiyohiko Okayama[3,c)] Motonori Nakamura[4,d)]

Received: September 16, 2014, Accepted: March 4, 2015

Abstract: The Internet has been widely deployed as an infrastructure to provide various ICT (Information and Communication Technology) services today. Some typical services such as e-mail, SNS (Social Network Service) and WWW rely considerably on the Internet in terms of reliability and effectiveness. In this paper, we focus on the IPv6 site multihoming technology and its collaboration with route selection mechanism, which have been reported as one solution to accomplish these goals. Even if a host can easily obtain multiple IP addresses in IPv6 multihomed site, it has to select a proper site-exit router when sending out a packet in order to avoid ingress filtering. Especially, when an inside host initializes an outbound connection it can barely select a proper site-exit router based on its source IP address. To solve this problem, we propose an optimal route selection method for IPv6 multihomed site. With this method, a middleware will be deployed within each inside host so as to connect to the destination host through multiple site-exit router during the initialization phase simultaneously, and then use the first established one for data communication. We also embedded a kind of Network Address Translation (NAT) feature into the middleware to avoid the ingress filtering. By analyzing the results of our experiments on the prototype system we confirmed that the proposed method worked as well as we expected and the collaboration of the site multihoming technology and the proper route selection method can be one possible solution for IPv6 site multihoming in a real network environment.

Keywords: IPv6, site multihoming, outbound connection and route selection

1. Introduction

The development of the Internet not only benefits the users but also changes the way ICT service is provided. Nowadays, most ICT services are provided via the Internet due to its effectiveness and wide popularity. Accordingly, the conditions of the Internet such as delay, congestions and link breaks affect both the service providers and the end users. Thus, in order to provide reliable and effective ICT services, it is important for service providers to build up fault-tolerant and traffic-balanced sites. One approach that we considered in this paper is the collaboration of site multihoming technology and the proper route selection method. Site multihoming is capable of providing multiple connections to the Internet, which can contribute to fault-tolerance (reliability). Methods for the proper route selection can help end user select a proper route (low latency, high speed, etc.) and also can contribute to traffic balancing in the service provider site.

Considering that nowadays the Internet consists of mainly IPv4 networks and some IPv6 networks, we have already proposed several approaches related to site multihoming technology and route selection methods for IPv4 networks. Thus in this paper, we discuss the same topic for IPv6 networks. Focusing on IPv6 site multihoming technology, we also have proposed a route selection method for outbound connections [1][*1]. In this proposal, we introduce middleware into each host in the IPv6 multihomed site. When the host initializes an outbound connection it tries to establish a connection to the destination host through each site-exit router simultaneously. It then uses the first established one for data communication. As we mentioned above, site multihoming technology can help service providers provide fault-tolerant and traffic-balanced ICT services but these advantages can not be obtained without proper route selection. Thus if an organization builds up an IPv6 multihomed site and applies this approach, the inside hosts[*2] not only can use the proper route for the data communications but also can use an alternative route when one of the links is down. Moreover, considering that ingress filtering might be running on the backbone networks, middleware also has a Network Address Translation (NAT) feature to avoid being filtered. Thus it also works well under ingress filtering.

On the other hand, site multihoming technology of IPv4 is different from that of IPv6. In general, a Network Interface Card (NIC) is unable to be assigned multiple IPv4 addresses automat-

[1] Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[2] Tokyo University of Agriculture and Technology, Fuchu, Tokyo 183–8538, Japan
[3] Okayama University, Okayama 700–8530, Japan
[4] National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a)] yongj@gsic.titech.ac.jp
[b)] nyamai@cc.tuat.ac.jp
[c)] okayama@cc.okayama-u.ac.jp
[d)] motonori@nii.ac.jp

[*1] This paper is a revised version of Ref. [1].

[*2] In this paper, the inside host works as a server in a inbound connection and works as a client in an outbound connection.

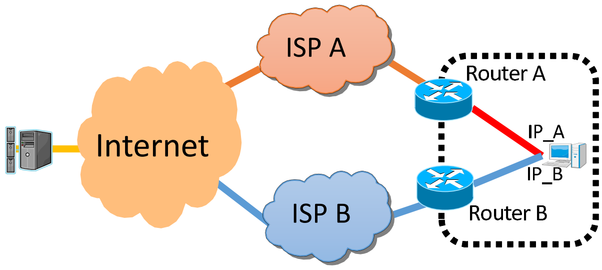

Fig. 1 Conventional IPv6 site multihoming.

ically (such as by DHCPv4 [2]), although it is possible to do so manually but it requires a lot of administrative work in large scale networks. Thus, in an IPv4 network, site multihoming is usually constructed using one site-exit router. We have proposed a route selection method for IPv4 multihomed networks [3]. In this method, a Network Address Translation (NAT) feature is introduced into the site-exit router for site multihoming. When an inside host initializes an outbound connection, the site-exit router duplicates the initialization packet and sends them to the destination host via different routes (NAT and global direct) simultaneously and then uses the first established connection for data communication. In an IPv4 network environment, this approach not only can be used for site multihoming but also can perform traffic balancing for multihomed sites. However, since there is only one single site-exit router in the site it can be the single point of failure. This makes the multihomed site unable to obtain the full advantages of the multihomed network.

Note that route selection for outgoing packets in multihomed sites needs to handle two kinds of communications in terms of directions: inbound connection (initiated from outside of the site) and outbound connection (initiated from inside of the site). For the inbound connection, the outgoing packet needs to be sent out via the site-exit router through which the incoming packet is sent in order to avoid ingress filtering [4]. For inbound connections the source IP address of the outgoing packet is automatically determined by the destination IP address of the incoming packet. Thus in this case only the site-exit route selection is necessary for selecting the proper route for the outgoing packet. For outbound connections, unfortunately both the source address and the site-exit router selections need to be handled when the inside host initializes an outbound connection. In order to provide optimal route selection in the multihoming, for inbound connections, we have proposed methods [5], [6] where a customized DNS server makes the host select the proper route by replying with the optimal IP address for each query based on the network conditions. However, these mechanisms are not applicable for outbound connections since the source address can not be selected by the DNS query and the DNS query can not analyze the network conditions of the route for the outgoing packet, either.

More importantly, in this paper, we propose an optimal route selection mechanism for outbound connections in an IPv6 multihomed network. Here, optimal route selection means that we are concerned about not only the proper source IP address selection but also the network conditions. We select the best route based on the criteria. With the proposed mechanism, we install a middleware into each inside host which attempts one connection through each site-exit router simultaneously when the host initializes an outbound connection by attaching the routing header appropriately. Consequently, the proposed method can solve the single point failure problem in an IPv4 network and can also contribute to achieving the full advantages of site multihoming technology.

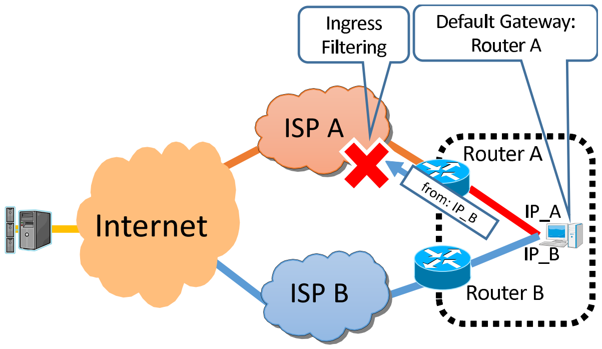

Fig. 2 Connection failure by ingress filtering.

2. IPv6 Site Multihoming and the Issues

2.1 Target Network Topology

Figure 1 shows a typical network topology of the IPv6 multihomed site we focus on in this paper. The host in the multihomed site is assigned an IP address from each ISP (ISP A and ISP B in this figure) and technically it can communicate with the Internet through any of the site-exit routers (Router A or Router B in this figure). An organization which constructs such a network topology implies that it has built up a multihomed site and what it needs next is a proper route selection mechanism. In this chapter, we consider a scenario where an inside host of a multihomed site with multiple IP addresses wants to initialize an outbound connection to the outside server. In this case, first, the host needs to select one of its IP addresses as the source IP address to establish the connection[*3]. Basically, the source IP address selection is mainly based on the longest match algorithm [7] and the host needs to select a proper site-exit router based on the selected source IP address. However, the longest match algorithm may cause some problems in a multihomed site and eventually the outbound connection attempt may be ended with a communication timeout. We show that the problems can be caused by the improper site-exit router selection in the following discussion.

2.2 Issues of the Site-exit Router Selection

According to the longest match algorithm [7], when a host has multiple IP addresses it uses the one which has the longest match with the destination IP address. We use **Fig. 2** to show an example of the communication failure caused by improper site-exit router selection when ingress filtering is running in the ISPs. In this example, we assume the IP address assigned by ISP B has the longest match with the destination host. On the other hand, we also assume the default gateway of the inside host is set as Router A. Finally, we put the deployment of the ingress filtering in both ISP A and ISP B as the prerequisite for this example. As a result, when the inside host initializes a connection to the

[*3] Note that in this paper we mainly discuss TCP communication since it is widely used for most applications in the Internet.

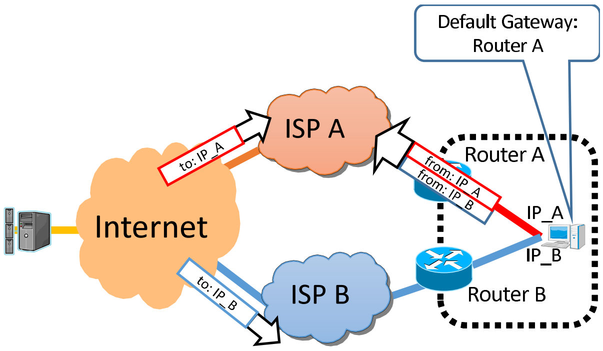

Fig. 3 Ineffective traffic balancing.

outside host, the IP address IP_B is selected as the source IP address based on the longest match algorithm and the initialization packet (SYN packet of TCP connection) is relayed to the default gateway Router A based on the routing algorithm. Then, ISP A captures the SYN packet and drops it since the ingress filtering judges the SYN packet has an improper source IP address. Consequently, the outbound connection can not be established and the communication may be ended with a communication timeout.

This problem not only happened with the outbound connection but also with the inbound connection. For the inbound connection, since the inside host has multiple IP addresses, the outside host can select one of them as the destination IP address for communication. If the default gateway of the inside host is not set to the proper site-exit router the same problem will occur. As a result, some kind of proper site-exit router selection mechanism needs to be introduced into IPv6 multihomed site.

2.3 Issue of Network Traffic Balancing

One of the great advantages of site multihoming is that the network traffic can be distributed across multiple ISPs. In the previous section we explained that ingress filtering may cause communications failure. However, even if we successfully negotiate with ISP A and ISP B to pass through all packets nomatter what source IP addresses they have, there will still be another problem with network traffic balancing. We show an example of ineffective network traffic balancing using **Fig. 3**. When ingress filtering is off in both ISPs, all packets from the inside hosts can be delivered via any of the site-exit routers. However, although the incoming packets can be distributed to multiple ISPs, all outgoing packets will go through the default gateway regardless of the available bandwidth and communication delay of the other route. In other words, all the outbound communications can be concentrated on the default route.

Furthermore, even if the default route problem is solved, we also need to consider the network conditions such as communication delay and available bandwidth in order to provide effective traffic balancing. For inbound connections, we have proposed dynamic route selection methods [5], [6] to meet these requirements. Here, we only explain the method proposed perviously [5] using **Fig. 4**. In what follows, the step number corresponds to the number illustrated in the figure.

(1) The outside host queries the IP address of the inside host to the inside DNS.

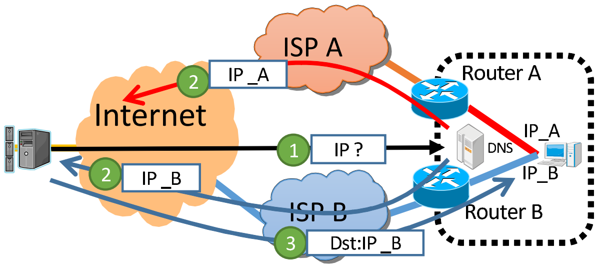

Fig. 4 The traffic balancing by multiple DNS responses.

(2) The inside DNS sends back different responses via each site-exit router simultaneously. The one sent via Router A indicates the IP address is IP_A and the one sent via Router B indicates the IP address is IP_B. In the DNS protocol, one response corresponds to one query. Thus in this example, the one that arrives first (response via Router B) is valid for the query and the other one will be simply discarded.

(3) The outside host uses IP_B as the destination IP address and begins communication.

This method is effective for route selection of the inbound connection in the multihomed site but it does not work well for the route selection of the outbound connection. Since the DNS protocol is mainly used for obtaining the destination IP address but not the source IP address, if we want to perform route selection for the outbound connection using the DNS protocol, we also need to customize the DNS server in the destination site which is not reasonable in reality.

2.4 Source Address Dependent (SAD) Routing Protocol

As related work, a solution based on the SAD routing protocol was considered for solving the site-exit router selection problem in case of ingress filtering. In the SAD routing protocol, the multihomed site needs to construct an SAD domain and the routers set in the SAD domain can forward packets based on the source IP address. This kind of source address-based routing feature can be realized by introducing PBR (Policy Based Routing) [9] into the host as well as routers. By deploying the SAD routing protocol into the multihomed site we can make the outbound route coincide with the inbound route in a specific TCP connection to solve the ingress filtering problem. However, in order to use the multihomed site effectively, we still need to detect the network conditions which are not included in the SAD routing protocol. One more concern is that the introduction of the SAD routing protocol involves all routers in the SAD domain. This can result in high administrative costs[*4] for large scale networks.

Considering the problems that may occur with the SAD routing protocol, we have proposed a route selection method by attaching a Routing Header that indicates the proper site-exit router according to the source IP address [10]. With this method, site-exit router selection based on the source IP address is done by middleware that we have developed and it needs to be installed into all inside hosts. However, middleware installation is similar to a kind of software deployment. Since it is comparatively easy to be introduced, this proposed method is applicable for the route se-

[*4] Although the deployment and administrative cost is not our main concern in this paper, we also consider the realization cost as a factor.

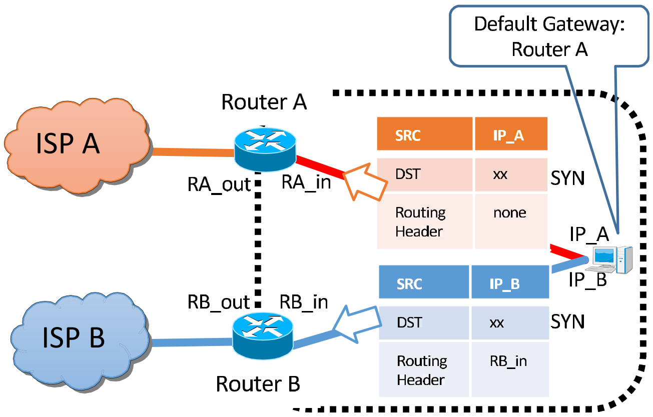

Fig. 5 The SYN packet process in the middleware.

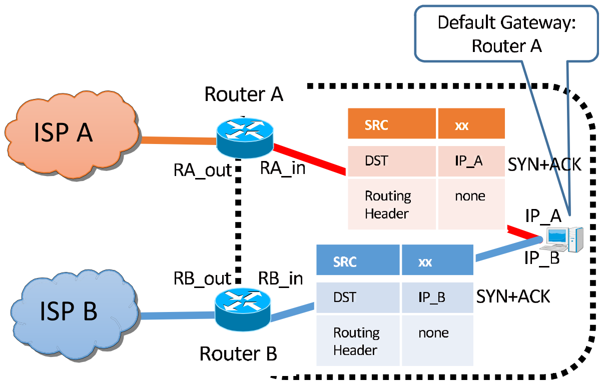

Fig. 6 The SYN+ACK packet process in the middleware.

lection of the outbound connection if the source IP address can be selected properly. However, one thing we need to clarify is that this proposed method still can not detect the network conditions when performing route selection for outbound connection. This is exactly the problem we want to solve in this paper.

3. An Optimal Route Selection Method for Outbound Connections

3.1 Basic Idea of the Proposal

The main purpose of this approach is to detect the network conditions before selecting a route for data communication in a multimode network. The basic idea is simple. When an inside host initializes an outbound connection from a multihomed network, it simultaneously attempts multiple routes and takes the fastest one for data communication. We have proposed a similar route selection method based on this idea for IPv4 network [3], but as we mentioned before, there is only one site-exit router in IPv4 multihomed network and it can be the single point of failure. Thus for IPv6 multihomed network, we consider a route selection mechanism under the case of having multiple site-exit routers.

In this approach, we add a middleware into the inside host and when it tries to initialize an outbound connection the middleware duplicates the SYN packet and sends them through different site-exit routers. Given that ingress filtering is running on the ISPs, the middleware also changes the source IP address if necessary and sends the packet via the proper site-exit router. In addition, we also add a middleware into the site-exit router so that it works as a non-default gateway which attaches a Routing Header to the incoming packet to make sure the return packet will be sent back to the proper site-exit router to be delivered to the outside.

When the outside host receives multiple SYN packets with different source IP addresses, it simply sends back a SYN+ACK packet for each one even if they are heading to the same inside host. Then the inside host takes the SYN+ACK packet that arrives first and establishes a connection. It then simply discards the remaining ones. As a result, the inside host uses the SYN packet to detect the network condition of each route and the fastest one (with lowest Round Trip Time) will be used for each data communication. Thus, this approach not only can solve the ingress filtering problem but also can perform proper traffic balancing among multiple routes.

3.2 Proper Source IP Address Selection

In this proposal, we detect the network condition using multiple SYN packets and make the inside host use the fastest route for data communication. Thus the inside host needs to process multiple SYN packets and SYN+ACK packets per connection. We show the processes in **Fig. 5** and **Fig. 6**, respectively.

In Fig. 5, when the middleware receives an outgoing SYN packet, it duplicates the packet and changes the source IP address appropriately before sending them out through the proper site-exit router. In this phase, the middleware installed in the host uses the Routing Header to forward the SYN packet to the non-default site-exit router. Note that this example shows a network topology with two ISPs but the proposal can also be applicable to a multihomed site connected with more ISPs.

In the next phase, in Fig. 6, when the outside host receives multiple SYN packets it simply sends back the SYN+ACK packets accordingly and the middleware in the inside host receives multiple SYN+ACK packets. Then the middleware forwards the first SYN+ACK packet to the OS and also changes the destination IP address properly if it is necessary.

Through the above processes, the inside host can select the proper source IP address and the corresponding site-exit router per connection for data communications.

3.3 Route Selection Using the Routing Header

In order to select a proper route for the outbound connection, with this approach we use a Routing Header [11] to indicate the proper site-exit router according to the selected source IP address. In this section, we describe the Routing Header-based route selection feature in more detail.

When the middleware installed in the inside host receives a TCP packet from the OS kernel, first, it checks if the packet is a SYN packet. If it is a SYN packet, the middleware duplicates it and changes its source IP address which is assigned from the non-default site-exit side. Then the middleware attaches a proper Routing Header to the IP header of the SYN packet to indicate the delivery route to the non-default site-exit router. Otherwise, since the optimal route for the received TCP (not SYN) packet has already been determined the middleware just translates the source IP address, if necessary, and attaches a proper Routing Header to the packet according to its source IP address.

On the other hand, the site-exit routers need to remove the

Routing Header since it is no longer needed out of the site. Otherwise the packet may be dropped somewhere in the Internet due to the lack of support for routing headers. Note that the Routing Header is not mandatory for all outbound connections. Thus if the packet without the Routing Header is guaranteed to go through the optimal route by the normal routing protocol, there is no need to attach the Routing Header to the packet. Also note that the Routing Header can be any type as long as all the inside hosts and site-exit routers use the same Routing Header.

With this approach, all the inside routers except for the site-exit routers are not required to be customized. But the new middleware needs to be installed into all inside hosts as well as the site-exit routers. Again, since the middleware installation is a kind of software deployment, it is relatively easy to perform, compared to the deployment of the SAD routing protocol.

3.4 How to Solve the Path MTU Problem?

As we described in the previous section, a 24-byte Routing Header needs to be attached to the packet with this approach. This causes an increase in packet size so that the packet size may exceed the MTU (Maximum Transfer Unit) of the inside host or the Path MTUs between the inside host and the site-exit routers. For the inside hosts, it is equivalent to a 24-byte reduction of the Path MTU. This means the inside host needs to adjust the payload of the outgoing packets according to the apparent Path MTU.

To avoid packet loss due to the Path MTU reduction, we additionally add two functions to the middleware installed in the inside host. First, if the packet size including the Routing Header exceeds the MTU of the network interface card then the middleware sends a Type2 ICMPv6 message [12] which indicates that the "Packet is Too Big" back to the OS kernel. With this message, the middleware also includes the apparent MTU which is 24 bytes smaller than the real one. Second, if the middleware receives the Type2 ICMPv6 message that indicates that the "Packet is Too Big" from somewhere other than the OS kernel, then it performs the following steps:

(1) The middleware checks if the original packet that caused the Type2 ICMPv6 message included the Routing Header by parsing the payload of the Type2 ICMPv6 message.
(2) If the original packet included the Routing Header then the middleware creates a new Type2 ICMPv6 message indicating the apparent MTU which is 24 bytes smaller than the real one in the received Type2 ICMPv6 message. It then sends this message to the OS kernel.
(3) If the original packet did not include the Routing Header then the middleware simply forwards it to the OS kernel, without any change.

With the above steps, it is possible to adjust the payload size of the outgoing packet when the size of the packet attached with the Routing Header exceeds the Path MTU.

4. Implementation of the Prototype System and Evaluations

4.1 Implementation of the Prototype System

Since the main purpose of the experiments is to confirm the effectiveness of the proposed method, we constructed an experimental network using several PCs and switching hubs. We also implemented the site-exit routers and the hosts using the middleware, which is coded based on the proposed method. For the operating sytems of the site-exit routers and the inside hosts, we selected OpenBSD since the Routing Header deletion function and the divert function for IPv6 is available in the OpenBSD [13]. Finally, for the Routing Header used in the proposed method, we selected the type 0 Routing Header of IPv6 protocol.

In this section, we describe the details of the middleware implementation, which is installed in the inside host. The following steps show the detailed procedures of the middleware when the inside host launches an outbound connection.

(1) Step 1: the inside host initializes an outbound connection
  (a) The middleware in the inside host receives a TCP packet via a divert socket.
  (b) Then the middleware checks if the destination IP address of the packet belongs to an internal site. If so, the middleware simply forwards the packet as usual.
  (c) If the destination IP address of the packet does not belong to an internal site then the middleware checks if the packet is a SYN packet. If so, the middleware duplicates the packet with an appropriate source IP address translation. After that, the middleware attaches a Routing Header to each SYN packet according to its source IP address, if necessary. (In case the source IP address belongs to the non-default site-exit router side, then the middleware attaches the Routing Header to the packet.) The middleware also keeps track of the 5-tuple of the connection, namely the original source IP address (srcIP), the destination IP address (dstIP), the source port number (srcPort), the destination port number (dstPort) and the IP address prefix of the optimal route (RP: Route Prefix). Every 5-tuple information is saved as one entry and all the entries for one connection will be saved until the optimal route is determined. After the optimal route is selected, all other entries for the connection will be deleted and the one for the optimal route will be saved until the communication is finished.
  (d) If the packet is not a SYN packet but a TCP packet, the middleware searches for the entry of the corresponding connection. If the prefix of the src IP address is different from the RP, the middleware performs the source IP address translation (i.e., change the source IP address to one with the same prefix as the RP).
  (e) Then the middleware attaches a Routing Header indicating a proper site-exit router to the packet according to its source IP address, if necessary.
  (f) Then the middleware attempts to send the packet out to the Internet. If the middleware fails to send the packet out due to the MTU problem, it generates a Type2 ICMPv6 message and sends it back to the OS kernel.
(2) Step 2: the inside host receives a connection request
  (a) The middleware installed in the inside host receives a packet via the divert socket.
  (b) Then the middleware checks if the packet is a SYN+ACK packet. If so, it searches for the entry of the

corresponding connection. If the entry exists, the middleware removes all other pending entries of the corresponding connection and sends the SYN+ACK packet using the continuous steps. If the entry is not found, it means the packet is not the first SYN+ACK packet. Thus the middleware sends a RST packet to the dstIP and discards the SYN+ACK packet.

(c) The middleware checks if the packet is an ICMPv6 message indicating that the "Packet is Too Big." If so, the middleware processes it based on the procedure described in Section 3.4.

(d) The middleware searches for the entry of the corresponding connection. If the entry is found and the destination IP address of the packet is different from the srcIP it is saved in the entry, then the middleware performs a destination IP address translation.

(e) Otherwise, the middleware simply forwards the packet to the OS kernel as usual.

4.2 Discussion of the Implementation and Deployment

As we mentioned before, most functions of the approach are done by a middleware instead of the OS kernel. Thus we do not need to customize the OS to introduce the proposal but only need to install a middleware into the inside host, which is comparatively easy. For normal inside hosts (without the middleware), they still can communicate with the Internet based on the conventional routing protocol via the default gateway (Router B) if they do not use a multihomed network. Thus the approach can be deployed step by step to a large scale network environment.

On the contrary, unlike the proposed method, some other approaches for site-exit router selection like the SAD routing protocol needs to customize the router OS, which is much more difficult than middleware installation. Moreover, the SAD routing protocol needs to be introduced to all the routers in the SAD domain to perform the site-exit router selection properly. Thus the deployment cost is very high on a large scale network environment. As we mentioned before, although it is not our main concern in this paper, we also need to consider the realization cost.

One more concern needs to be clear. And that is the validity of the IPv6 Type0 Routing Header (RH0) used in our approach. RH0 has been deprecated in an RFC [14] due to some security concerns. However, as we have already discussed [15], routers which are not listed in the RH0 will not check RH0 by default. Thus the proposal will work well as we expected. Furthermore, the site-exit router will delete the RH0 before sending out the packet according to the proposed method. Thus the usage of RH0 will not affect the communication security out of the site.

Finally, in this paper we targeted a multihomed site connected to two ISPs but the proposal is also applicable to multihomed sites including more than two site-exit routers. In those cases, the middleware needs to be updated about the newly added or deleted site-exit routers and so far this update needs to be performed manually. This update is very important and automatic middleware update will be more effective but this concern is beyond the scope of this paper.

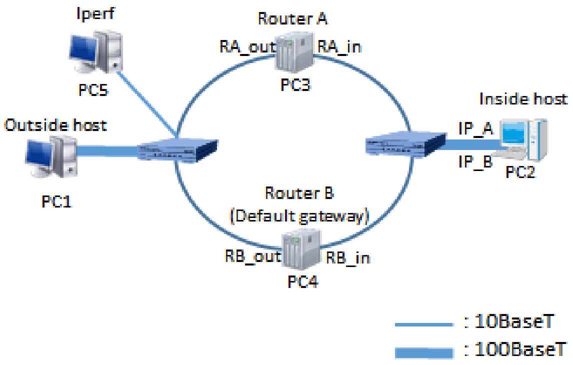

Fig. 7 The experimental network topology.

Table 1 Specifications of PCs.

PC	CPU, Main memory	OS
PC1	Core2 Duo 2.93 GHz, 2 GB	FreeBSD 8.2
PC2	Core2 Duo 2.93 GHz, 2 GB	OpenBSD 5.0
PC3	Core2 Duo 2.93 GHz, 2 GB	OpenBSD 5.0
PC4	Core2 Duo 2.93 GHz, 2 GB	FreeBSD 8.2
PC5	Intel Pentium 4 2.40 GHz, 1 GB	FreeBSD 8.2

4.3 Feature Evaluation

In order to evaluate the proposed method we constructed an experimental network including the prototype system described in Section 4.1. **Figure 7** shows the network topology and **Table 1** lists the specifications of the PCs used in the environment. We used 10BaseT for all links except for the one between PC1 and PC2 in order to create a congestion situation easily. In other words, the network traffic in a 10BaseT link can be affected by the background network traffic more easily than in a 100BaseT link.

In our experimental network, the default gateway of the inside host (PC2) is configured to the normal router (without any customization based on the proposed method) and dubbed Router B (PC4). On the outside host (PC1) the next-hops for destinations IP_A and IP_B are set to Router A (the prototype router with the proposed method) and Router B, respectively.

Using the above experimental environment, we first evaluated the features of the proposal to confirm that the inside host could communicate with the outside host using the optimal route. In this feature evaluation, we set up an HTTP server on the outside host and made the inside host access it via either of the site-exit routers. We also monitored the routes that the packets passed through due to the existence of the Type 0 Routing Header in the packet as well as the packet size on the site-exit routers. As a result, we confirmed that the SYN packet sent from the inside host was duplicated properly and the route through which the first SYN+ACK packet returned was selected for data communication. This means the inside host selects the optimal route based on the round trip time of the SYN packet and uses the route for data communication regardless of what source IP address is selected. We also confirmed that the source IP address translation function changed the source IP address to IP_A in Router A correctly for the outgoing packet as well as the reverse change for the response packet. Finally, we confirmed that the Type 0 Routing Header attachment and deletion functions on the Router

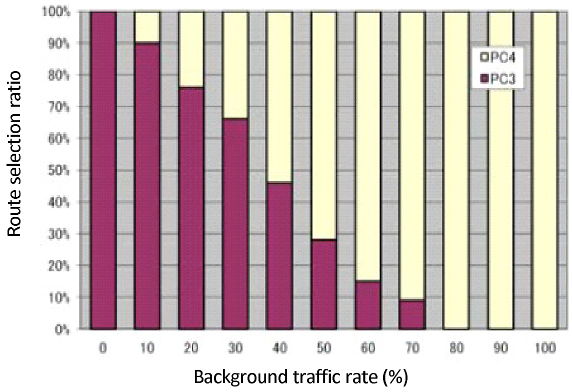

Fig. 8 The route selection ratio for various background traffic rate.

A worked as we expected and the Path MTU adjustment feature also worked properly on the inside host.

4.4 Performance Evaluation

After finishing the feature evaluations, we continued with the performance evaluations to verify the effectiveness of the network traffic balancing of the prototype system. In the performance evaluations, we measured the number of connections on each route when the congestion occurred on one of the routes. Additionally, since the proposal has some overhead caused by the attachment and deletion of the Type 0 Routing Header as well as the Path MTU reduction, we also measured the latency and throughput on the prototype system.

These performance evaluations were also performed in the same network environment as shown in Fig. 7. We describe the details with the evaluation results as follows:

(1) Evaluation of the network traffic balancing: First, we accessed the outside host (PC1) from the inside host (PC2) using the HTTP protocol once at a time without any background traffic and measured the number of connections for each route. As a result, all connections were established via Router A (PC3) since the RTT (Round Trip Time) of the route via Router A was about 15 microseconds shorter than via Router B. This is the original network condition of the experimental network and the proposal has a preference for the route with lowest latency.

Next, we measured the number of connections for each route when there was some background UDP traffic generated by "Iperf" between PC3 and PC5. **Figure 8** shows these results of the measurements. According to the results, we can see that the more the background traffic increases the more the route through PC4, which is less crowded than that through PC3, is selected for data communication. Accordingly, we confirmed that the proposal can perform traffic balancing according to network latency.

(2) The experiment of the throughput measurement: Then, we evaluated the overhead of the prototype system. In this experiment, we measured the throughputs of the TCP connections by "Iperf" in combination with the following cases: whether both PC2 and PC3 were of the prototype system or of a normal system, whether the traffic direction was incoming or outgoing, and whether the route passed through PC3 or PC4. The link speed of all the links was 100 Mbps. From the results of this experiment shown in **Table 2**, we confirmed that the overhead (shown as Difference) of the prototype system was small enough except for the outgoing throughput on the route through the PC3. However, this overhead is reasonable since the payload size would be reduced to as much as 24 bytes by adding the Routing Header.

Table 2 The result of throughput measurement.

Direction	System of PC2, PC3	Throughput	Difference
Incoming through PC3	Normal system	92.92 Mbps	-
	Prototype system	92.88 Mbps	0.05%
Outgoing through PC3	Normal system	92.94 Mbps	-
	Prototype system	91.30 Mbps	1.76%
Incoming through PC4	Normal system	92.70 Mbps	-
	Prototype system	92.68 Mbps	0.02%
Outgoing through PC4	Normal system	92.68 Mbps	-
	Prototype system	92.64 Mbps	0.04%

In the method proposed previously [3], the NAT router duplicates the SYN packet and it increases overhead to the communication. However, in the proposed method of this paper, the overheads are from the inside hosts and non-default site-exit routers instead, due to packet duplication and routing header attachment as well as deletion. Thus, it is necessary to make the selection of the non-default site-exit routers more difficult since they increase overhead to the communication. For example, we can add some delay to the packet with the routing header before delivery to control the route selection but such an approach is beyond the scope of this paper and we would like to consider as future work.

5. Conclusion

In this paper, we discussed IPv6 site multihoming technology in collaboration with a route selection mechanism in order to create a stable and effective architecture for ICT service providers. As one solution for outbound connections in a multihomed site, we proposed an optimal route selection method by duplicating the SYN packet and translating the source IP address. The feature evaluation results confirm that the proposed method worked as well as we expected and the performance evaluation results show that the overhead in the prototype system was small enough to be applied to a real network environment. For future work, we plan to evaluate the proposed method in a real network environment using several different applications. In this paper, we focused on TCP connections and we also would like to develop traffic balancing mechanism for UDP traffics.

References

[1] Yamaguchi, T., Yong, J., Yamai, N., Okayama, K., Okamoto, K. and Nakamura, M.: An Optimal Route Selection Mechanism for Outbound Connection on IPv6 Site Multihoming Environment, *Proc. IEEE 37th Annual Computer Software and Applications Conference Workshops* (*COMPSACW*), p.575, p.580 (July 2013).

[2] Droms, R.: Dynamic Host Configuration Protocol, RFC2131, IETF (1997).

[3] Yamai, N., Okayama, K., Shimamoto, H. and Okamoto, T.: A Dynamic Traffic Sharing with Minimal Administration on Multihomed Networks, *Proc. IEEE International Conference on Communications 2001*, Vol.5, pp.1506–1510 (June 2001).

[4] Baker, F. and Savola, P.: Ingress Filtering for Multihomed Networks, RFC3704, IETF (2004).

[5] Yong, J., Yamai, N., Okayama, K., Seike, T. and Nakamura, M.: A Dynamic Route Selection Method Using Multiple DNS Replies for Inbound E-mail Delivery on Multihomed Environment, *J. Inf. Process.*, Vol.51, No.3 (2010).

[6] Yong, J., Yamai, N., Okayama, K. and Nakamura, M.: An Adaptive Route Selection Mechanism Per Connection Based on Multipath DNS Round Trip Time on Multihomed Networks, *J. Inf. Process.*, Vol.20, No.2 (2010).

[7] Thaler, D. (Ed.), Draves, R., Matsumoto, A. and Chown, T.: Default Address Selection for Internet Protocol Version 6 (IPv6), RFC6724, IETF (2012).

[8] Bagnulo, M., Garcia-Martinez, A., Rodriguez, J. and Azcorra, A.: End-site routing support for IPv6 multihoming, *Computer Communications*, Vol.29, No.7, pp.893–899 (2006).

[9] Cisco Systems Inc.: *Policy-Routing*, available from ⟨http://www.cisco.com/warp/public/cc/pd/iosw/tech/plicy_wp.pdf⟩.

[10] Yong, J., Yamaguchi, T., Yamai, N., Okayama, K. and Nakamura, M.: A Site-Exit Router Selection Method Using Routing Header in IPv6 Site Multihoming, *J. Inf. Process.*, Vol.20, No.1, pp.1–9 (Jan. 2012).

[11] Deering, S. and Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification, RFC2460, IETF (1998).

[12] Conta, A., Deering, S. and Gupta, M. (Eds.): Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, RFC4443, IETF (2006).

[13] divert - kernel packet diversion mechanism, OpenBSD manual pages (online), available from ⟨http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man4/divert.4?query=divert⟩ (accessed 2014-08-20).

[14] Abley, J., Savola, P. and Neville-Neil, G.: Deprecation of Type 0 Routing Headers in IPv6, RFC5095, IETF (2007).

[15] Yong, J., Yamaguchi, T., Yamai, N., Okayama, K. and Nakamura, M.: A Site-Exit Router Selection Method Using Routing Header in IPv6 Site Multihoming, *J. Inf. Process.*, Vol.21, No.3 (2013).

Yong Jin received his M.E. degree in electronic and information systems engineering and Ph.D. degree in industrial innovation sciences from Okayama University, Japan in 2009 and 2012, respectively. In April 2012, he joined the Network Architecture Laboratory of the Photonic Network Research Institute in the National Institute of Information and Communications Technology, Japan, as a researcher. From October 2013, he joined the Global Scientific Information and Computing Center of Tokyo Institute of Technology as an assistant professor. His research interests include network architecture, traffic engineering and Internet technology. He is a member of IPSJ and IEICE.

Nariyoshi Yamai received his B.E. and M.E. degrees in electronic engineering and his Ph.D. degree in information and computer science from Osaka University, Osaka, Japan, in 1984, 1986 and 1993, respectively. In April 1988, he joined the Department of Information Engineering, Nara National College of Technology, as a research associate. From April 1990 to March 1994, he was an assistant professor in the same department. In April 1994, he joined the Education Center for Information Processing, Osaka University, as a research associate. In April 1995, he joined the Computation Center, Osaka University, as an assistant professor. From November 1997 to March 2006, he joined the Computer Center, Okayama University, as an associate professor. From April 2006 to March 2014, he was a professor in the Information Technology Center (at present, the Center for Information Technology and Management), Okayama University. Since April 2014, he has been a professor in the Institute of Engineering, Tokyo University of Agriculture and Technology. His research interests include distributed system, network architecture and Internet. He is a member of IEICE and IEEE.

Kiyohiko Okayama received his B.S., M.S. and Ph.D. degrees in information and computer sciences from Osaka University, Japan, in 1990, 1992 and 2001, respectively. After he has worked in the Department of Information System at Osaka University and in the Graduate School of Information Science at Nara Institute of Science and Technology as a research associate, he joined the Department of Communication Network Engineering at Okayama University in 2000. From 2005 to 2011, he joined the Information Technology Center at Okayama University. Since 2011, he has been an associate professor in Center for Information Technology and Management at Okayama University. His research interests include network design and network security. He is a member of IEICE.

Motonori Nakamura graduated from Kyoto University, Japan, where he received his B.E., M.E. and Ph.D. degrees in engineering in 1989, 1991 and 1996, respectively. From 1994, he was an assistant professor at Ritsumeikan University. From 1995, he was an associate professor at Kyoto University. Currently he is a professor at National Institute of Informatics, Japan (NII). His research interests are message transport network systems, network communications, next generation Internet and Identity & Access Management. He is a member of IEEE, ISOC, IEICE and JSSST.