Design, Implementation and Evaluation of Routing Protocols for IPv6 Anycast Communication

Satoshi Doi
Osaka University
s-doi@anarg.jp

Shingo Ata
Osaka City University
ata@info.eng.osaka-cu.ac.jp

Hiroshi Kitamura
NEC Corporation
kitamura@da.jp.nec.com

Masayuki Murata
Osaka University
murata@ist.osaka-u.ac.jp

Abstract

Anycast is a new IPv6 feature that supports service–oriented address assignments in IPv6 networks. Because there are no protocol standards or even consensus on routing protocols, inter–segment anycast communications are not yet available. In this paper, we propose two routing protocols for inter–segment anycast to support anycast–oriented communication. Our proposed architecture (1) achieves the advantages of anycast communications, (2) takes the deployment scenario into the existing unicast network into consideration, and (3) maintains scalability. We also implement the proposed routing protocols in an experimental environment and verify that they work correctly.

1 Introduction

Anycast [1] is one of the new IPv6 (IP version 6 [2]) features that supports service-oriented address assignments in IPv6 networks. An anycast address is not determined by the location of the node, but by the type of service offered at the node. In anycast communications, the client can automatically obtain the appropriate node corresponding to a specific service without knowledge of the location of the server.

Like a multicast address, a single anycast address is assigned to multiple nodes (called *anycast membership*), but unlike multicasting, only one member of the assigned anycast address communicates with the originator at a time.

The basic idea behind anycast communication is to separate the logical service identifier from the physical host equipment, i.e., the anycast address is assigned on a type-of-service basis, which enables the network service to act as a logical host.

However, IPv6 anycasting still has several problems that need to be clarified within the context of the current specifications. In our previous work, we showed some applications suitable to anycasting and provided some advantages of anycasting [3].

Another problem with IPv6–based anycasting is that a routing protocol has not been included in its specifications, which is indispensable in making anycasting more widespread. There are several challenging issues that need to be resolved in designing anycast routing protocols [3].

1. Scalability issue

   The routing entries for anycast addresses should be stored individually on the router. It is easy to imagine explosions in routing tables as anycast addresses get to be more widely used.

2. Criteria for selecting anycast membership

   Anycast routing is required to transfer an anycast packet to an *appropriate* anycast node, but the meaning of *appropriate* needs to differ among applications. The criteria for anycast routing strongly affects anycast communication capabilities.

Based on these findings, we designed routing protocols for inter-segment anycast communication that we will present after the next section.

The rest of this paper is organized as follows. In Section 2, we describe the specification of our architecture and we test and evaluate our proposed protocols in Section 3. Finally, we summarize our work and describe our future research topics in Section 4.

2 Routing Protocol Design

We focused on the difference between anycasting and unicasting/multicasting to develop an anycast routing protocol through existing unicast/multicast routing protocols. Anycasting and unicasting/multicasting have many similar characteristics while they also have some differences.

Several protocols for unicast or multicast routing are currently available. As we can see in Table 1, these can be classified into three types, i.e., a (1) distance vector, (2) link state, and (3) core-based tree.

Since each routing protocol has both advantages and disadvantages, we defined the anycast routing protocol based on all of these, i.e., (1) the *Anycast* extensions to RIP

Table 1. Classification of Routing Protocols

	Distance-Vector	Link-State	Core-Based-Tree
Unicast	RIPng [4]	OSPFv3 [5]	
Multicast	DVMRP [6]	MOSPF [7]	PIM-SM [8]
Anycast	ARIP	AOSPF	PIA-SM [9]

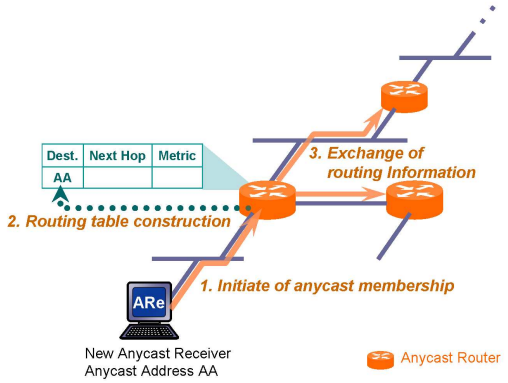

Figure 1. Overview of Anycast Routing Protocol

(ARIP), (2) the *Anycast* extensions to OSPF (AOSPF), and (3) the Protocol Independent *Anycast* Sparse Mode (PIA-SM). In our previous work, we designed the (3) PIA-SM [9], and Matsunaga [10] provides a good description of the implementation method for this. (1) ARIP and (2) AOSPF are presented in turn in the subsections that follow.

In terms of functionality, a routing protocol for anycast communication consists of the following three steps (see also Figure 1) and the difference between the two above–mentioned routing protocols are in Step 2.

1. Initiate anycast membership
2. Construct and update routing table
3. Forward anycast packets

The anycast router forwards anycast packets based on the routing table constructed in Step 2. Note again that Step 3 is the same as unicast routing. Each anycast router simply checks the unicast routing table to find an entry regarding the destination address of the packet. In what follows, we detail Step 1 and Step 2 separately.

2.1 Initiating Anycast Membership

Like multicasting, the host participating in (or leaving from) anycast membership must have the capability of notifying the nearest anycast router of the status (joining/leaving). The method of finding a host participating in anycast membership (called *anycast host* below) is different and is based on the location of the anycast host. If the anycast receiver and the anycast router are on the same segment, an extended version of MLD (Multicast Listener Discovery) is used [11]. This is called ARD (Anycast Receiver Discovery). Basically, an anycast host generates an ARD report message to the anycast router after the anycast host receives an ARD query message from the anycast router. The anycast host can additionally send the ARD report message if it cannot receive the ARD query message. However, the anycast host sends an ARD done message prior to leaving membership. Because the destination address field of ARD packets is set to one of the link-local addresses, e.g., the link-scope all-nodes (`FF02::1`) or the link-scope all-routers (`FF02::2`), this method can only be applied where all hosts and routers reside within the same segment.

2.2 Constructing and Updating Routing Table

If the type of routing entry advertised by the anycast receiver is only the receiver metric, the processes of constructing and updating the routing table are common to the ARIP and AOSPF. We call these procedures as the *Advertising Receiver Metric*, which we present first. This is followed by an explanation on constructing and updating routing tables supporting the link metric.

2.2.1 Advertising Receiver Metric

Figure 2 outlines the constructing and updating routing tables when anycast routers only consider the receiver metric. If anycast routers only consider the receiver metric, they can use unicast routing information to describe the topology of routers. Each anycast receiver becomes just like a *leaf* attached to a tree constructed through the topology of anycast routers.

Before describing the procedure, we define some routing related nodes.

- **Selected anycast receiver** is the anycast receiver which has the minimum metric among the same anycast membership.
- **Alternate anycast receiver** is the anycast receiver which has the second minimum metric among the same anycast membership.
- **Selected anycast router** is the anycast router physically or virtually connected to the *selected anycast receiver*.
- **Alternate anycast router** is the anycast router physically or virtually connected to the *selected anycast receiver*.

- **Adjacent anycast router** is the anycast router connected physically or virtually.

Figure 2 shows the basic operations for the Advertising Receiver Metric which are following.

1. Notify the membership information by exchanging ARD query/report All anycast receivers send the ARD report indicating their membership information and metric in response to the ARD query sent from the anycast router periodically. If the anycast receiver cannot receive the ARD query, they can send the unsolicited ARD report.

After receiving the ARD report, the anycast router creates/updates the entry in the local database called ARDB (Anycast Receiver Database). Each entry in the ARDB is stored with three items: the anycast address itself, the receiver metric, and the unicast address of the anycast router.

2. Send the information of new anycast receiver. If the anycast router receives the ARD report, it sends the information on the new anycast receiver (i.e., three items registered in the ARDB) to the adjacent anycast routers.

3. Constructing the routing table and the ARDB After receiving the entry of ARDB, the anycast router lookups the routing entry for the anycast address specified in the ARD report, and compares the metric in the ARD report with the metric in the matched routing entry. If the metric in the ARD report is smaller than the metric in the routing entry, the anycast router replaces the metric to the smaller one. Then, the anycast router forwards the entry to all the adjacent anycast routers except the router from which the ARD report arrives. By propagating the ARD report hop-by-hop basis, all anycast routers can obtain the minimum value of the metric and its forwarding direction. Then, all the packets sent to the anycast address are transferred to the *selected anycast receiver*.

Additionally, the anycast router connected to the anycast receiver can store the entry of ARDB if the anycast address of attached anycast receiver is the same as the anycast address of new anycast receiver. This stored ARDB entry is used when the metric is updated. If the *selected anycast router* does not have the ARDB entry of other anycast receivers and it detects the overload of *selected anycast receiver* by using *threshold exceeding* message as follows, the anycast router will send a large number of message to discover other anycast receivers. Moreover, if the anycast router receives this message, it also does not know other anycast receiver. Then, each anycast router sends the reply message respectively. It consumes much of traffic.

Therefore, each anycast router stores the receiving entry in the ARDB if the metric is more than the value of the attached anycast receiver's metric.

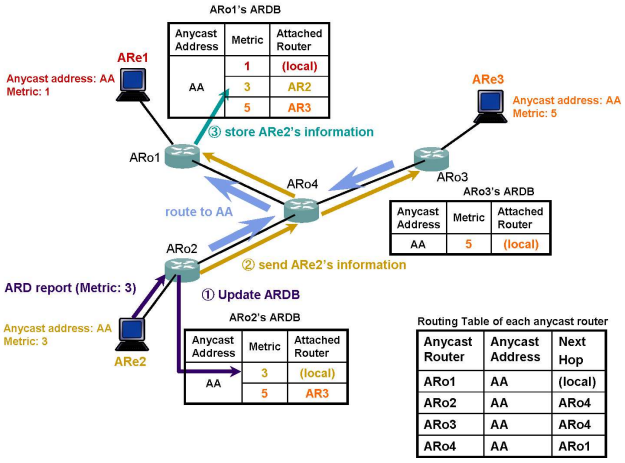

Figure 2. Basic Operation of Advertising Receiver Metric

Basically, all requests from the client are forwarded to the anycast receiver with the lowest metric (called the *selected anycast receiver*). If the condition of the *selected anycast receiver* changes, the metric of the receiver changes. When the *selected anycast receiver* does not have the lowest metric, another anycast receiver (called *alternate anycast receiver*) is selected as a new *selected anycast receiver*.

To discover the *alternate anycast receiver*, the *selected anycast router* picks out an entry with the lowest metric among all entries in the ARDB except for the current *selected anycast receiver*. More details about this updating mechanism can be found in [12].

2.2.2 Supporting Link Metric - ARIP

The ARIP and the AOSPF use different mechanism to collect the link metric. In this paper, we only describe the ARIP due to space limitation. The AOSPF is designed in the same manner as ARIP by modifying OSPF. More details about AOSPF can be found in [12].

Figure 3 has an example of constructing/updating a routing table with ARIP. ARIP works as follows.

1. Notify the membership information by exchanging ARD query/report All anycast receivers send the ARD report indicating their membership information (i.e., anycast address) in response to the ARD query the anycast router sent periodically. If the anycast receiver cannot receive the ARD query, they can send the unsolicited ARD query at certain interval (e.g., every 30 seconds). The periodical update by the anycast router is triggered by the ARD report from the anycast receiver.

2. Send the ARI message After receiving the ARD report, the anycast router creates an *Anycast Route Information* (ARI) message which consists of at least (anycast address, metric) pair. Then, the anycast router sends it to the adjacent anycast routers. When the anycast router sends the ARI message to adjacent anycast routers, it overwrites the metric of ARI message by adding the link metric associated with the output interface. This is because that the link metric in the direction from the anycast receiver is more important. The anycast receiver acts as a server, then much data will be transferred from the anycast receiver to the clients.

3. Receive the ARI message and update the routing table and/or Blocking list When an anycast router receives the ARI message, the anycast router first checks whether the anycast address of the ARI message has already been stored in the routing table. If the anycast address is not in the routing table on the anycast router, the anycast router registers the anycast address into the routing table. Then, the anycast router overwrites the metric of ARI message and forwards it to the adjacent anycast routers except in the direction of its source. Otherwise, it compares the metric of the ARI message with the metric of existing routing entry.

After receiving the entry of ARI message, the anycast router lookups the routing entry for the anycast address specified in the ARI message, and compares the metric in the ARI message with the metric in the matched routing entry. If the metric in the ARI message is smaller than the metric in the routing entry, the anycast router replaces the metric to the smaller one. Then, the anycast router forwards the entry to all the adjacent anycast routers except the router from which the ARI message arrives. When the anycast router sends the ARI message to adjacent anycast routers, it overwrites the metric of ARI message by adding the link metric associated with the output interface. By propagating the ARI message hop-by-hop basis, all anycast routers can obtain the minimum value of the metric and its forwarding direction. Then, all the packets sent to the anycast address are transferred to the *selected anycast receiver*.

Otherwise, the anycast router checks the direction where the anycast router receives the ARI message. If the output interface of the existing routing entry is different from the interface which receives the ARI message, this ARI message means the existence of another anycast receiver, which is not the *selected anycast receiver*. The anycast router stores the (anycast address, metric) pair in the *Blocking List*. This entry stored in the *Blocking List* is used when the metric of existing routing entry increases and it is no longer the entry with minimum metric. Otherwise, the ARI message means the update of the existing routing entry. Therefore, the metric of the existing routing entry increases, and the anycast router may keep another entry which has less metric than the existing entry in the *Blocking List*. Then, the anycast router stores this alternate entry in the routing table, and moves the existing entry to the *Blocking List*.

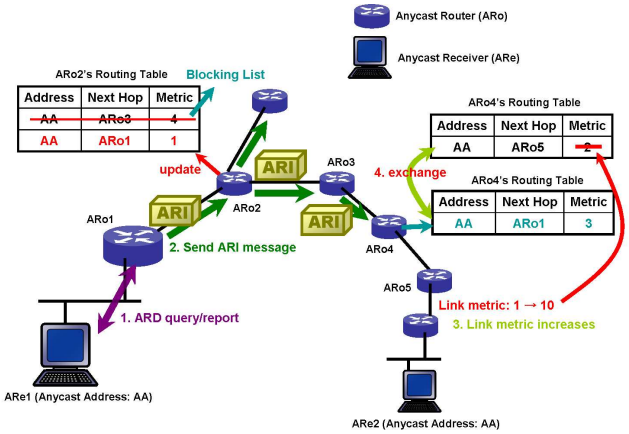

Figure 3. ARIP

3 Implementation and Evaluation

In this section, we will first describe implementation issues that arose with the proposed routing protocols. Then, we will prove that they worked correctly through some experiments. Moreover, we will compare the proposed protocols, ARIP and AOSPF, described in Section 2.

3.1 Implementation of Routing Protocols

3.1.1 ARD

To transfer metrics from anycast receivers, we used a similar method to the extended version of MLD (we refer as ARD throughout this paper) proposed by Haberman and Thaler [11]. We made an additional extension to transfer a metric.

We consequently extended the ARD packet format to transfer metrics from anycast receivers. The embedded information of *metric* includes *metric type* and *metric value*. The *metric type* is used to support multiple selection criteria for the anycast address. We defined two *metrics* to classify the *metric types*: 1) a receiver metric, and 2) a link metric. The ARD report message may have multiple sets of header fields and an extendable value field. If one anycast receiver has multiple anycast addresses, it should send multiple membership and metric information to the anycast router. To reduce the number of messages and improve efficiency, the anycast receiver can set multiple entries in the one ARD report message.

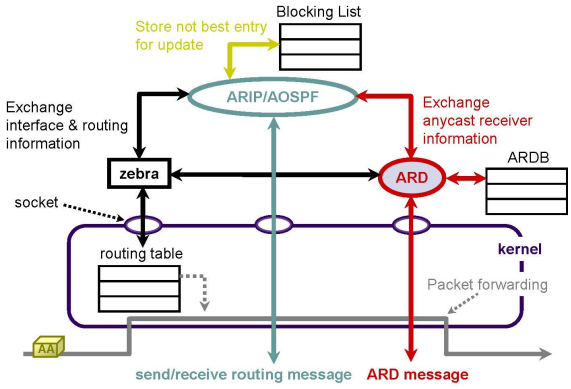

Figure 4. Implementation Overview

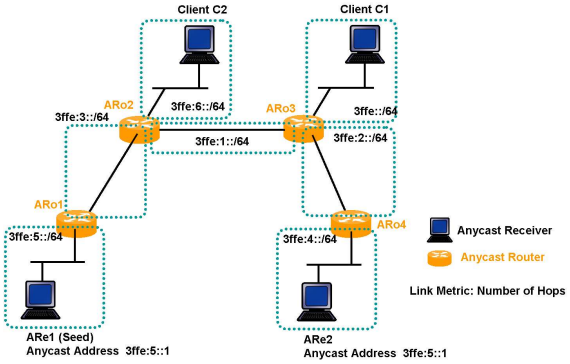

Figure 5. Network Environment for Implementation Experiment

3.1.2 ARIP and AOSPF

We modified the GNU Zebra(`http://www.zebra.org`) to support the anycast routing protocols described in Section 2. GNU Zebra is free software that manages TCP/IP-based routing protocols. It supports multiple routing protocols including RIPng [4] and OSPFv3 [5]. Each routing protocol (e.g., RIPng and OSPFv3) is implemented as an independent daemon process (e.g., `ripngd` for RIPng and `ospf6d` for OSPFv3). Each routing daemon communicates with the zebra daemon, which obtains the interface and route information from the system kernel. Due to the multiprocessing nature of Zebra software, it can easily be upgraded. Each routing protocol can be upgraded separately, without modifying the other protocols.

Figure 4 is an overview of the implementation. The RIPng and AOSPF were modified to ARIP and AOSPF, which support both unicasting and anycasting. We added a new process to deal with ARD packets and manage locally attached anycast receivers. Both ARIP and AOSPF communicate with the ARD daemon and obtain information on anycast receivers attached to the anycast router. Both ARIP and AOSPF use a specific message to handle anycast address (i.e., ARI, AM-LSA), and routing information for the anycast address is transferred to other anycast routers as described in Section 2. If the routing daemon receives a routing message, it determines whether it will transfer the information in the message to the Zebra daemon or the *Blocking List*. The routing information in the *Blocking List* is useful in updating the metric. After collecting routing information from the routing daemon or the ARD process, the Zebra daemon adds routing information to and deletes it from the routing table in the system kernel. Packets are forwarded by the kernel according to the routing table constructed by the Zebra daemon.

3.2 Experimental Results

Figure 5 outlines our experimental network topology where we have assumed that the criteria of node selection is the number of hops, i.e., the smaller hop count is the more appropriate. In Figure 5, there are four anycast routers and two anycast receivers. Each anycast router is connected to multiple network segments. Anycast receivers are placed on the different network segments. Additionally, the client C 1 is connected to the anycast router ARo 3, and the other client C 2 is connected to the anycast router ARo 2. We first chose a *seed node* [3] from the anycast membership, and then assigned the anycast address of this membership to be the unicast address of the *seed node*. Here, we selected ARe 1 as the seed node of the anycast membership. That is, the anycast address of the membership is set to `3ffe:5::1`, which is the unicast address of node ARe 1. The other node ARe2 in a different network (`3ffe:4::/64`) has the same anycast address.

To verify that our routing protocol works correctly, we examine two simple tests. We first check routes from both clients (C1 and C2) to the anycast address `3ffe:5::1` in the case where all of routers are unicast routers. Since there is no anycast router in this case, all anycast packets are expected to be forwarded to ARe1. We then run programs of anycast routing protocols on anycast nodes (i.e., ARIP on anycast routers, ARD on anycast receivers). After running routing programs, we again check routes to the anycast address. In the latter case, anycast packets from C2 are delivered to ARe1, while packets from C1 are forwarded to ARe2.

We use `traceroute6` command to check the route to the anycast address. `traceroute6` shows intermediate nodes from the source node to the destination node with the round trip delay.

We first execute `traceroute6` with specifying the

anycast address `3ffe:5::1` on C1. The result is following:

```
C1> traceroute6 3ffe:5::1
traceroute6 to 3ffe:5::1 (3ffe:5::1) from
3ffe::210:f3ff:fe01:e242, 64 hops max,
12 byte packets
 1  ARo3  0.500 ms  0.289 ms  0.202 ms
 2  ARo2  0.534 ms  0.403 ms  0.363 ms
 3  ARo1  0.731 ms  0.573 ms  0.649 ms
 4  ARe1  1.029 ms  0.851 ms  0.800 ms
```

Next is the result of `traceroute6` on client C2.

```
C2> traceroute6 3ffe:5::1
traceroute6 to 3ffe:5::1 (3ffe:5::1) from
3ffe:6::210:f3ff:fe01:e23b, 64 hops max,
12 byte packets
 1  ARo2  0.384 ms  0.192 ms  0.191 ms
 2  ARo1  0.416 ms  0.490 ms  0.495 ms
 3  ARe1  0.871 ms  0.545 ms  0.650 ms
```

Even if the anycast routers do not execute the anycast routing protocol, the packets sent to the anycast address `3ffe:5::1` can be reached to the anycast receiver ARe1.

However, the anycast receiver ARe2 is more appropriate for the client C1 because the number of hops from C1 to ARe2 is smaller than the one from C1 to ARe1.

We next run anycast routing protocols on both anycast routers and receivers. Then, we execute the same command as above. The result of `traceroute6` on C1 is following.

```
C1> traceroute6 3ffe:5::1
traceroute6 to 3ffe:5::1 (3ffe:5::1) from
3ffe::210:f3ff:fe01:e242, 64 hops max,
12 byte packets
 1  ARo3  0.411 ms  0.243 ms  0.159 ms
 2  ARo4  0.405 ms  0.397 ms  0.363 ms
 3  ARe2  0.737 ms  0.562 ms  0.658 ms
```

Next is the result of `traceroute6` on client C2.

```
C2> traceroute6 3ffe:5::1
traceroute6 to 3ffe:5::1 (3ffe:5::1) from
3ffe:6::210:f3ff:fe01:e23b, 64 hops max,
12 byte packets
 1  ARo2  0.446 ms  0.311 ms  0.190 ms
 2  ARo1  0.526 ms  0.503 ms  0.374 ms
 3  ARe1  0.868 ms  0.707 ms  0.627 ms
```

These results show that packets from C1 to the anycast address `3ffe:5::1` are reached to the appropriate anycast receiver ARe2. Moreover, packets from C2 still reach the anycast receiver ARe1. For the client C2, ARe1 is more appropriate than ARe2. From the above results, we can see that our proposed routing protocols work as expected.

4 Conclusion and Future Works

In this paper, we have discussed our analysis and design of new anycast routing control mechanisms for inter–segment anycast communications. Then we have implemented our routing mechanisms by modifying existing routing software. Our experiments revealed the feasibility of these protocols

It is necessary to evaluate our proposed protocols on several real networks (e.g., unstable and large networks) in future research, and confirm their efficiency. Another approach from an entirely different viewpoint is to design a completely new routing protocol, which would provide one possible solution, and the knowledge derived from our research should be useful in designing this.

References

[1] R. Hinden and S. Deering, "IP version 6 addressing architecture," *RFC3513*, April 2003.

[2] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," *RFC2460*, Dec. 1998.

[3] S. Doi, S. Ata, H. Kitamura, and M. Murata, "IPv6 anycast for simple and effective communications," *IEEE Communications Magazine*, vol. 42, pp. 163–171, May 2004.

[4] G. S. Malkin, "RIPng for IPv6," *RFC2080*, Jan. 1997.

[5] R. Coltun, D. Ferguson, and J. Moy, "OSPF for IPv6," *RFC2740*, Dec. 1999.

[6] D. Waitzman, C. Partridge, and S. Deering, "Distance vector multicast routing protocol," *RFC1075*, Nov. 1988.

[7] J. Moy, "Multicast extensions to OSPF," *RFC1584*, Mar. 1994.

[8] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. gung Liu, P. Sharma, and L. Wei, "Protocol independent multicast-sparse mode (PIM-SM): Protocol specification," *RFC2117*, June 1998.

[9] S. Doi, S. Ata, H. Kitamura, and M. Murata, "Protocol design for anycast communication in IPv6 network," in *Proceedings of 2003 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'03)*, pp. 470–473, Aug. 2003.

[10] S. Matsunaga, "PIA-SM: Design and implementation of core-based routing protocol for ipv6 anycast," *Bachelor's thesis, School of Engineering Science, Osaka University*, Feb. 2004.

[11] B. Haberman and D. Thaler, "Host-based anycast using MLD," *Internet draft* `draft-haberman-ipngwg-host-anycast-01.txt`, May 2002. (Expired November 2002).

[12] S. Doi, "Design, implementation and evaluation of routing protocols for IPv6 anycast communication," Master's thesis, Graduate School of Information Science and Technology, Feb. 2004.