Design and Implementation of Routing Protocols for IPv6 Global Anycast Communications

Satoshi Doi
Graduate School of Information Science and Technology, Osaka University, 1–5 Yamadaoka, Suita, Osaka 565–0871, Japan

Shingo Ata
Graduate School of Engineering, Osaka City University, 3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan

Hiroshi Kitamura
Solution Development Laboratories, NEC Corporation, 2–11–5 Shibaura, Minato-Ku, Tokyo 108–8557, Japan

Masayuki Murata
Graduate School of Information Science and Technology, Osaka University, 1–5 Yamadaoka, Suita, Osaka 565–0871, Japan

Abstract: Anycast is a new IPv6 feature that supports service–oriented address assignments in IPv6 networks. However, because there are no protocol standards or even consensus on routing protocols, inter–segment anycast communications are not yet available. In this paper, we first discuss problems and solutions on inter–segment anycast communications. Based on our findings, we propose two routing protocols for inter–segment anycast to support anycast–oriented communication. We also implement the proposed routing protocols in an experimental environment and verify that they work correctly. We also compare the proposed routing protocols and discuss which is most suited to reducing overheads as much as possible.

Keywords: IPv6, Anycast Communication, Network–layer Anycast, Anycast Application, Anycast Routing

Reference to this paper should be made as follows: Doi, S., Ata, S., Kitamura, H. and Murata, M. (2005) 'Design and Implementation of Routing Protocols for IPv6 Global Anycast Communications', *International Journal of Internet Protocol Technology*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Satoshi Doi is currently with Sony Corporation, Japan. He received B.E. and M.E. degrees in information science and technology from Osaka University in 2002 and 2004, respectively. His research work is in the area of designing protocols related to IPv6 anycasting.

Shingo Ata is a lecturer in the Graduate School of Engineering at Osaka City University, Japan. He received M.E. and Ph.D. degrees in informatics and mathematical science from Osaka University in 1998 and 2000, respectively. His research includes design of communication protocols and performance modeling of communication networks.

Hiroshi Kitamura works at NEC Corporation since 1990. He also works as a Visiting Associate Professor at University of Electro-Communication since 2004. He received B.S. and M.S. degree from Nagoya University in 1988 and 1990, respectively. He also received a Ph.D. degree in informatics and mathematical science from Osaka University in 2002. He has been engaged in research and development of Internet protocols. He currently focuses on research and development of IPv6, Mobile IPv6, Plug and Play, and Security.

Masayuki Murata is a professor in the graduate school of information

Table 1: IPv6 address types.

	unicast	multicast	anycast
communication form	point to point	point to multipoint	point to point
target of address	node	group	service type
number of membership	single	multiple	multiple
roles in C/S model	both	client (listner)	server

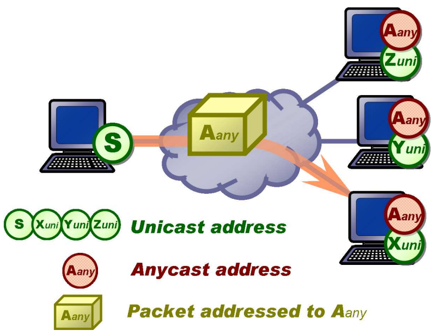

Figure 1: Anycast communication.

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. Anycast, which is defined in the IPv6, is a new networking paradigm supporting service–oriented addresses and an identical address can be assigned to multiple nodes providing a specific service. An anycast packet (i.e., one with an anycast destination address) is delivered to one of these nodes with the same anycast address. The idea of anycast was first described in RFC 1546 [3], which stated that the motivation for anycasting was to drastically simplify the task of finding an appropriate server on the Internet.

The Internet Protocol version 6 (IPv6) has three types of IP addresses, i.e., unicast and multicast addresses as in IPv4, and an anycast address that is the subject of the current paper. Table 1 summarizes the forms of communication for these addresses. A unicast address is a unique identifier for each network interface, and multiple interfaces must not be assigned the same unicast address. Packets with the same destination address are sent to the same node. A multicast address, on the other hand, is assigned to a group of nodes, i.e., all group members have the same multicast address and packets for this address are sent to all members simultaneously.

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.

Figure 1 has an example of anycast communication. There are three nodes associated with the anycast address A_{any}. When the source node sends a packet, where the destination address is A_{any}, the packet is sent to one of three nodes (X_{uni} in this figure), not to all hosts. The advantage of anycasting is that the source node can receive a specific service without knowledge on current conditions in service nodes and/or networks. When host X_{uni} goes down, the packet for A_{any} can be sent to another host (Y_{uni} or Z_{uni}) (Fig. 1). How appropriately the destination node is chosen from anycast membership depends on the anycast routing protocol.

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 [4].

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 [4].

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.

We also need to identify how stateful applications utilize anycasting in designing their routing protocols. Internet applications using all TCP-based or some UDP-based protocols are *stateful*, i.e., end hosts establish the conditions of communication with each other and assume that their partners are identical during the exchange. This is very important because the current definition of anycasting is essentially *stateless*, i.e., the destination host should be determined on a packet-by-packet basis by the routers. In our previous work, we have proposed Anycast Address Resolving Protocol (AARP) to establish TCP connections with a specific anycast address [4].

The rest of this paper is organized as follows. The next section discusses the proposed anycast routing architecture. In Section 3, we describe the specification of our architecture and we test and evaluate our proposed protocols in Section 4. Finally, we summarize our work and describe our future research topics in Section 5.

2 ARCHITECTURAL DESIGN

The advantage of anycast communication from the application's view is that the packet is automatically forwarded to the appropriate node according to network and/or node conditions. It is therefore important to maintain the routing information of anycast addresses. Because of this, we will propose a new anycast routing architecture, which we describe in this section.

2.1 Design Choices and Models

The design choices we made in our anycast routing protocol are as follows.

2.1.1 Using Existing Address Space

We allowed unicast and anycast addresses within the same space and to do this we chose a *seed node* from anycast membership before assigning an anycast address. We then established the anycast address of membership to be the unicast address of the *seed node*. The anycast router forwards an anycast packet to an *appropriate* node within the anycast membership. However, the unicast router only tries to forward the anycast packet to the *seed node*. An anycast packet leaving an arbitrary node is at the very least sent to the seed node. Any packet destined for the anycast address is guaranteed to be sent to at least one destination node.

2.1.2 Gradual Deployment

We envision the gradual deployment of anycasting and the protocol works correctly in our architecture and offers advantages even if there is only one anycast router between the sender and seed node. Its impact increases as more anycast routers are deployed.

2.1.3 Modifying Existing Routing Protocols

We adopted an approach that modifies existing routing protocols to the anycast routing protocol to reduce the complexity of implementation.

2.1.4 Packet-by-Packet Basis Forwarding

Each anycast router forwards anycast packets to only one node on a packet-by-packet basis.

As previously discussed, anycast routers should have both node selection criteria and knowledge to select one entry. We introduce a value called *metric* for this purpose. Each anycast router selects one *appropriate* node based on the *metric*. The meaning of *appropriateness* differs due to the kinds of applications, and only a node with an anycast address can know what application it provides. Then, the *metric* is advertised by the node having the anycast address. Moreover, we assume that the *metric* is non-negative with an integer value that simplifies the operation of anycast routers. All anycast routers select *appropriate* a node from multiple nodes by simply comparing the *metric* (e.g., the anycast router chooses one routing entry with a minimum value for the metric). We define two types of *metric* (called *metric type*):

- receiver metric: the preference value of a node with an anycast address (e.g., CPU load).

  The receiver metric can only be set by the anycast receiver, and must not be updated by the anycast router. This type of metric is suitable for notifying of the availability of resources in the anycast receiver (e.g., CPU resources and number of acceptable requests).

- link metric: the preference value of a link among two anycast routers (e.g., propagation delay).

  If the metric is link metric, the anycast router overwrites the metric value in the control message by adding the value of link metric associated with the anycast router. The link metric is useful in describing the end-to-end performance (e.g., round trip delay and number of hops). However, this link metric should be configured based on the metric of unicast routing because the anycast packets traverse the route that unicast routing uses.

2.2 Proposed Architecture

Figure 2 is an overview of the routing architecture we propose and there are two types of routing topologies. The *unicast network* is the existing network topology where both unicast and anycast packets are forwarded on the basis of a unicast address. In the *anycast network*, anycast-aware routers (called *anycast routers*) are connected to one another and only anycast packets are forwarded by treating their addresses as anycast addresses. The anycast network can thus be considered as a logical overlay network over the unicast network.

In an anycast network, nodes are not physically (i.e., directly) connected, but are connected via various kinds of logical peer-to-peer connections (e.g., virtual path, tunneling, or encapsulation). An anycast router is upper-compatible, does anycast routing functions, and has the capabilities of unicast routers. An anycast router has extra routing entries (called *anycast routing entries*) in the unicast routing table to handle anycast addresses. For each anycast address, the anycast router registers only one *anycast routing entry* as a *host entry*. The *host entry* means the routing entry with 128–bit length address prefix. If the anycast router knows there are multiple nodes having same anycast address, it selects one node and registers it in the routing table. When a packet arrives at the anycast router, it checks the unicast routing table to find an entry regarding the destination address of the packet in the same fashion as existing unicast routing which uses the longest prefix matching. As a result of the longest prefix matching, the anycast routing entry must be chosen if the anycast router has its entry. Then, the anycast router can find an entry by using the destination address. After it finds this, the packet is treated as an *anycast packet* and forwarded to the next anycast router according to the routing table. Otherwise, it is forwarded through the unicast routing mechanism.

Figure 2 has an example of anycast routing where we have assumed that the node selection criterion is the number of hops. A smaller count is more appropriate here. In Figure 2, the short cylinders represent routers and the one labeled "ARo" is an anycast router. The other short cylinders (i.e., non-labeled cylinders) are unicast routers. There are two anycast members for the anycast address `3ffe:5::5`. Note here that `3ffe:5::5` is also the unicast address of anycast receiver ARe1. Here, node

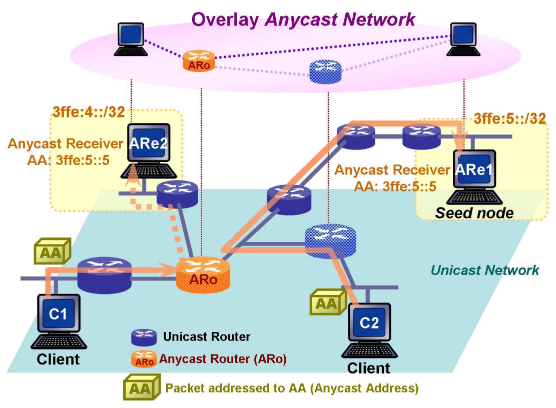

Figure 2: Proposed Architecture

ARe1 is the *seed node* of anycast membership for `3ffe:5::5`. The other node ARe2 is in a different network (`3ffe:4::/32`). Let us now consider where two nodes (C1 and C2) send packets destined for anycast addresses `3ffe:5::5`. The difference is whether there is an anycast router on the route to seed node ARe1. C1 first forwards the packet to router ARo through unicast routing (solid arrow). Intermediate router ARo is an anycast router and can detect that the packet is also an anycast packet.

According to anycast routing (dashed arrow), anycast router ARo then forwards it to node ARe2, which is the node nearest C1. However, since there is no anycast router between C2 and ARe1, the packet is simply forwarded to ARe1 through unicast routing only. Note that there is a more appropriate node (ARe2) in this network. For example, if we replace the router next to C2 (short-checked cylinder) with an anycast router, the packet could be transmitted to the more appropriate ARe2 node through anycast routing.

The above description reveals that our anycast routing protocol works appropriately even when there are a limited number of anycast routers. If these are increased, better routing is achieved. When all routers in the network are anycast, flexible routing adopting a control policy using various metrics will be possible.

3 ROUTING PROTOCOL DESIGN

This section describes the routing protocols for the anycast routing architecture we propose. Note again that our basic motivation in supporting anycasting was to minimize the overheads or implementation involved in deploying it as much as possible.

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. Our first step in designing the anycast routing protocol is to clarify these similarities and then find how to modify the existing routing protocols to support anycast routing.

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

In the distance vector algorithm, a router has a list of routers which are directly connected to it. By exchanging the list with other adjacent routers, it can identify all routers capable of connecting to an arbitrary destination. The link state algorithm utilizes a list of connected links instead of a list of routers. Through exchanging the list of links, the router can identify the entire topology of the network. The router then prepares a shortest path tree (SPT) with Dijkstra's shortest path first algorithm [5]. Based on the SPT, the router finally constructs the routing table. The core-based tree is a kind of hierarchical algorithm and it first chooses one or more *core* routers from all routers. The *core* router centralizes all routing information on behalf of the other routers. One of these other routers only holds the routing information for where it belongs. Each router only sends a packet to the *core* router and only it can decide the route for the destination address.

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 (ARIP), (2) the *Anycast* exten-

Table 2: Classification of Routing Protocols

	Distance Vector	Link State	Core Based Tree
Unicast	RIPng [6]	OSPFv3 [7]	
Multicast	DVMRP [8]	MOSPF [9]	PIM-SM [10]
Anycast	ARIP	AOSPF	PIA-SM [11]

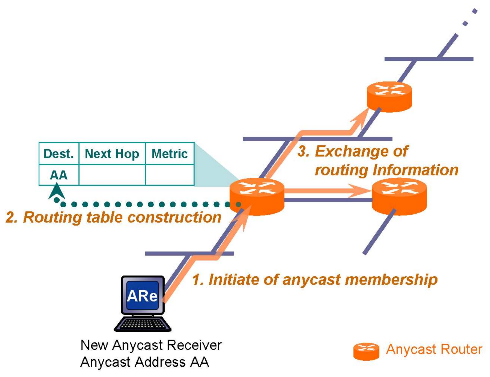
Dest.
Next Hop
Metric
AA
3. Exchange of routing Information
2. Routing table construction
1. Initiate of anycast membership
ARe
New Anycast Receiver
Anycast Address AA
Anycast Router

Figure 3: Overview of Anycast Routing Protocol

sions to OSPF (AOSPF), and (3) the Protocol Independent *Anycast* Sparse Mode (PIA-SM). In our previous work, we designed the (3) PIA-SM [11], and Matsunaga [12] provides a 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 3) 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.

3.1 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 [13]. 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.

3.2 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.

3.2.1 Advertising Receiver Metric

Figure 4 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 4 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.

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.* Then, the fol-

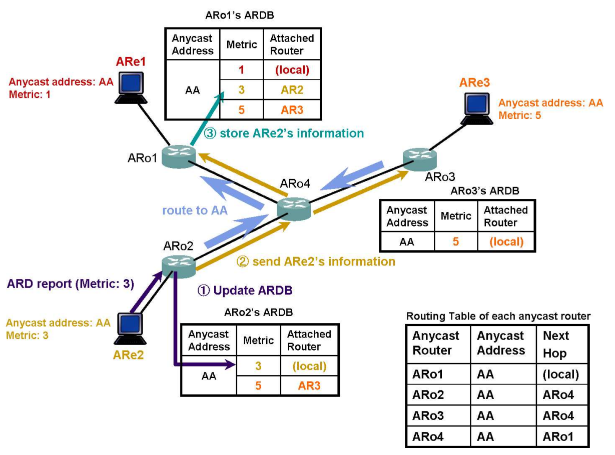

Figure 4: Basic Operation of Advertising Receiver Metric

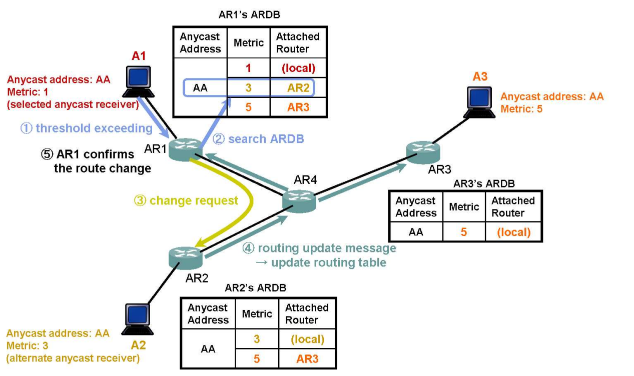
AR1's ARDB
Anycast Address | Metric | Attached Router
AA | 1 | (local)
3 | AR2
5 | AR3
A1
Anycast address: AA
Metric: 1
(selected anycast receiver)
① threshold exceeding
② search ARDB
⑤ AR1 confirms the route change
AR1
A3
Anycast address: AA
Metric: 5
AR3
AR4
AR3's ARDB
Anycast Address | Metric | Attached Router
AA | 5 | (local)
③ change request
④ routing update message → update routing table
AR2
AR2's ARDB
Anycast Address | Metric | Attached Router
AA | 3 | (local)
5 | AR3
Anycast address: AA
Metric: 3
(alternate anycast receiver)
A2

Figure 5: Route Update of Advertising Receiver Metric

lowing process is done to update the routing entry (See Figure 5).

1. The *selected anycast receiver* sends a *threshold exceeding* message when the metric value exceeds the threshold.

2. When the *selected anycast router* receives the *threshold exceeding* message, it lookups the entries which has the minimum metric for the anycast address specified in the *threshold exceeding* message. Then, it selects an *alternate anycast receiver*, and finds the *alternate anycast router* in the ARDB.

3. After finding the *alternate anycast receiver*, the *selected anycast router* sends a *change request* message to the *alternate anycast router*. The *alternate anycast router* is identified by the unicast address of anycast router stored in the ARDB.

4. After receiving the *change request* message, the *alternate anycast router* sends a *routing update* message to all anycast routers by using flooding.

5. Then, each anycast router updates its routing table when it receives the *routing update* message. The *selected anycast router* can recognize that the *alternate anycast router* changes its routing information based on the change route request. If the *selected anycast router* cannot receive a *routing update* message, it wonders that the *change request* message might be dropped. The *selected anycast router* remove the entry which has the minimum metric for the anycast address in the ARDB, and back to Step 2.

We designed this mechanism assuming that the *selected anycast receiver* would change according to the condition of anycast receivers. In unicast routing, packets during transitions in the routing path may be dropped, but it does not occur frequently. However, with anycasting, because route changes occur more frequently than with unicasting, the frequency of dropping packets in anycasting is relatively higher than the case of unicasting. Consequently, more fail–safe mechanisms are needed to prevent anycast packets from being dropped because of route changes. With our design, the anycast router which sent the *change request* message confirms that the *selected anycast receiver* is replaced by the *alternate anycast receiver* by receiving the *routing update* message. For a more complete fail–safe mechanism, we introduced following procedure.

- During route changes, the routing information for all anycast routers is unstable. Some anycast routers have changed to the new route while others have not done yet. Because the new route information is distributed from the *alternate anycast router*, anycast packets may be forwarded to either the *selected anycast receiver* or the *alternate anycast receiver*. Since the *routing update* message is transferred from the *alternate anycast router*, the anycast routers near the *alternate anycast router* can update the routing table early. However, anycast routers far from the *alternate anycast router* may forward the anycast packet to the *selected anycast receiver*. In that case, the *selected anycast router* redirects the anycast packet toward the *alternate anycast router*. This prevents packets from dropping.

This mechanism is also useful when anycast receivers fail. If a *selected anycast receiver* suddenly goes down without sending a *threshold exceeding* message and the attached anycast router detects this, the router attached to the *selected anycast receiver* sends a *change request* message to the router attached to the *alternate anycast receiver*. Additionally, during route changes, it redirects anycast packets toward the *alternate anycast receiver*.

3.2.2 Supporting Link Metric

The ARIP and the AOSPF use different mechanism to collect the link metric. In what follows, we describe these mechanisms separately.

ARIP

Figure 6 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*.

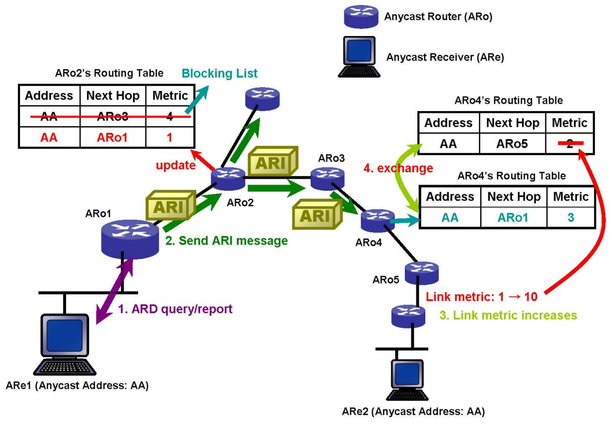

Figure 6: ARIP

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*.

AOSPF

Figure 7 has an example of updating the routing table with the AOSPF, which is operated as follows.

1. Constructing topology of anycast routers

   Each anycast router creates a Router/Network LSAs (Link State Advertisement), which shows the information for the attached link of the anycast router. This process is the same as for OSPF [7]. After the anycast router receives LSAs from all the other anycast routers and it stores them in the topological database. The anycast router can then obtain a corresponding graph showing the topology for anycast routers by using Dijkstra's shortest path first algorithm.

2. Notify the membership information by exchanging ARD query/report

   All anycast receivers send the ARD report indicating their membership information 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.

3. Create and send the *Anycast Membership LSA*

   After receiving the ARD report, the anycast router creates an *Anycast Membership LSA* (AM-LSA) packet which consists of the anycast address. Then, the anycast router sends it to the adjacent anycast routers.

4. Receive the AM-LSA and update the routing table and/or *Blocking list*

   When an anycast router receives the AM-LSA message, the anycast router first checks whether the anycast address of the AM-LSA 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 forwards it to the adjacent anycast routers except in the direction of its source.

   Otherwise, it evaluates the *appropriateness* of the entry specified in the AM-LSA message as follows. The anycast router first calculates the total link cost from the router itself to the originator of the AM-LSA by using the topology of the anycast routers generated in Step 1. Then, the anycast router can obtain the receiving entry's metric. The anycast router compares the metric of the receiving entry and the metric of existing routing entry. If the AM-LSA's metric is smaller than the metric in the routing entry, the anycast router updates the routing entry and forwards the AM-LSA packet to adjacent anycast

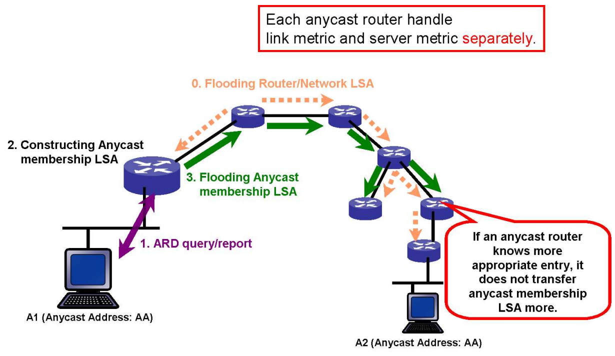
Each anycast router handle
link metric and server metric separately.
0. Flooding Router/Network LSA
2. Constructing Anycast membership LSA
3. Flooding Anycast membership LSA
1. ARD query/report
If an anycast router knows more appropriate entry, it does not transfer anycast membership LSA more.
A1 (Anycast Address: AA)
A2 (Anycast Address: AA)

Figure 7: AOSPF

routers except the router from which the AM-LSA packet arrives. If the ARI packet's metric is equal to or greater than the existing metric, the anycast router does not forward it and registers it in the *Blocking List* to update the metric. The 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.

When the anycast router detects the change of the condition of the attached link, it generates a Router/Network LSA showing the current link status and sends this advertisement to the adjacent anycast routers. Then, the receiving anycast router recalculates the routing table. If the anycast router has a more appropriate entry in the *Blocking List*, it sends the AM-LSA message saved in the *Blocking List* to all anycast routers except the router from which the AM-LSA packet arrives.

3.3 Packet Forwarding

The packet forwarding procedure is straightforward. When an anycast router receives an anycast packet, it first checks its unicast routing table. Each routing entry for the anycast address is registered as a *host entry*, and consequently the anycast router can find the routing entry for the anycast receiver by matching the longest prefix if the entry exists.

4 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 2.2.

4.1 Validating Anycast Routing Protocols

We designed the routing protocols based on existing versions (i.e., RIP and OSPF). We then implemented them by modifying the existing routing software that is currently available and widely used in IPv6 networks.

4.2 Implementation of Routing Protocols

4.2.1 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 [13]. They proposed that anycast receivers could join anycast membership by generating and sending an extended MLD report message. The anycast receiver set the multicast address field for the extended MLD report message to the anycast address it wished to join. All other fields were the same as regular (i.e., multicasting) MLD report messages. This simple extension enabled an anycast receiver to notify the anycast router only about membership information. 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

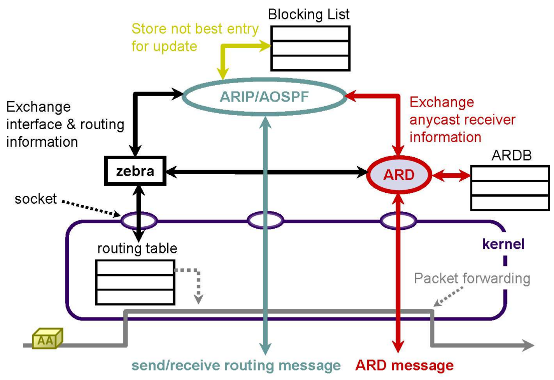
Blocking List
Store not best entry for update
Exchange interface & routing information
ARIP/AOSPF
Exchange anycast receiver information
ARDB
zebra
ARD
socket
routing table
kernel
Packet forwarding
AA
send/receive routing message
ARD message

Figure 8: Implementation Overview

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.

4.2.2 ARIP and AOSPF

We modified the GNU Zebra(`http://www.zebra.org`) to support the anycast routing protocols described in Section 2.2. GNU Zebra is free software that manages TCP/IP-based routing protocols. It supports multiple routing protocols including RIPng [6] and OSPFv3 [7]. 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 8 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 2.2. 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.

4.3 Experimental Results

Figure 9 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 9, 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. As described in Section , we first chose a *seed node* [4] 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

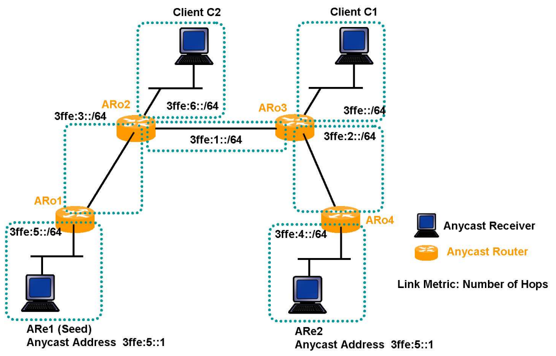

Figure 9: Environment for Experiment

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.

4.4 Comparisons of Anycast Routing Protocols

Let us now compare our proposed protocols, i.e., ARIP and AOSPF. We had the following three objectives in mind for the comparison.

- Convergence time due to membership changes
- Protocol overheads (e.g., CPU load, memory consumption)

- The amount of bandwidth consumed by the packet, which the routing protocol itself uses
- Load of router (e.g., CPU load and memory consumption)

- Convergence time due to membership changes

  All routing protocols determine the route through certain communication between routers. All routing protocols have different convergence times, which means the time until the exchange of routing information stops. This convergence time strongly affects the stability of the network. This convergence characteristic is one of the factors determining the scalability of the protocol specifying it. Therefore, if the convergence time for a routing protocol is long, routing-protocol scalability is low.

- Protocol overheads (e.g., CPU load, memory consumption)

  Executing any protocol will consume some network resources, such as router memory for storing the routing table, and network bandwidth for transmitting the routing-control message. A good routing protocol should reduce these overheads and keep resource consumption to a minimum. We considered the following items in evaluating the overheads.

  - The amount of bandwidth consumed by the packet, which the routing protocol itself uses

    The packet a routing protocol uses to update routing information is actually unnecessary in static routing. Therefore, traffic by this packet must be kept to a minimum. However, these packets to update routing information needs to be exchanged when routing information comes over or route time-out occurs. To reflect the actual network conditions in a routing table, it is best for routers to exchange routing information frequently. There is a trade-off between the amount of bandwidth consumed by the packet, which the routing protocol also uses, and the accuracy of the routing protocol.

  - Load of router (e.g., CPU load and memory consumption)

    If a routing protocol is running in a network, routers in the network incur some load in running it. If the routing protocol require some calculations to the router, its CPU load is consumed. If the router needs to retain more routing information, its memory consumption increases.

Table 3: Comparisons of Anycast Routing Protocols

	ARIP	AOSPF
convergence	slow (hop by hop)	fast (flooding)
traffic consumption	$O(m)$	$O(l)$
calculation	$O(1)$	$O(l * log(r))$
number of entries	$O(g)$	$O(g)$

These two measures to limit overheads increase as the scale of a network increases. In the following, we describe the protocol overheads for ARIP and AOSPF. The overheads for all anycast routers in these protocols are summarized in Table 3.

m means the number of nodes that share the same anycast address, l means the total number of links, r means the total number of anycast routers, and g means the number of anycast groups (number of anycast addresses).

ARIP takes a long time for routes to converge because the anycast router transfers ARI packets hop-by-hop. AOSPF, on the other hand, takes less time to converge. This is because the anycast router knows the current conditions of all links and they only process anycast receivers as a leaf that is attached to a tree. However, if there are more anycast receivers in the network, the time for ARIP to converge will decrease. The whole network is divided into multiple areas, and the number of routers the ARI packets traverse will decrease.

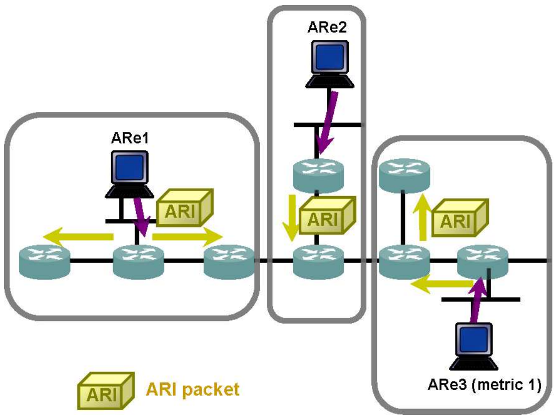

Figure 10: ARIP: Reduce the Time to Converge

With respect to protocol overheads, both ARIP and AOSPF have better performance than unicast routing protocols (i.e., RIP and OSPF) except for the number of routing entries. As anycast addresses cannot inherently be aggregated, their routing entries are stored in the routing table for each address.

5 CONCLUSION AND FUTURE WORKS

In this paper, we pointed out there were no routing protocols to handle anycast addresses, and that inter–segment anycast communications had not been achieved. we have discussed our analysis and design of new anycast routing control mechanisms for inter–segment anycast communications. The design approach was modifying existing routing protocols to reduce the complexity of implementation. Furthermore, the proposed mechanisms were aimed at gradually deploying anycasting and scalable design. Then we have implemented our routing mechanisms by modifying existing routing software. Our experiments revealed the feasibility of these protocols, and comparisons with routing protocols proved that our proposal could significantly reduce overheads.

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.

ACKNOWLEDGEMENTS

This work is partly supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE-I) from the Ministry of Internal Affairs and Communications of Japan.

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 C. Patridge, T. Mendez, and W. Milliken, “Host anycasting service,” *RFC1546*, Nov. 1993.

4 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.

5 E. W. Dijkstra, “A note on two problems in connection with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

6 G. S. Malkin, “RIPng for IPv6,” *RFC2080*, Jan. 1997.

7 R. Coltun, D. Ferguson, and J. Moy, “OSPF for IPv6,” *RFC2740*, Dec. 1999.

8 D. Waitzman, C. Partridge, and S. Deering, “Distance vector multicast routing protocol,” *RFC1075*, Nov. 1988.

9 J. Moy, “Multicast extensions to OSPF,” *RFC1584*, Mar. 1994.

10 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.

11 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.

12 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.

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