Master's Thesis

Title

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

Supervisor

Prof. Hideo Miyahara

Author

Satoshi Doi

February 13th, 2004

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

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

Satoshi Doi

Abstract

Anycast is a new IPv6 feature that supports service–oriented address assignments in IPv6 networks. However, the current anycast definitions are still unclear in several respects, so the use of anycast addresses is quite limited. Also, because there are no protocol standards or even consensus on routing protocols, inter–segment anycast communications are not yet available. In this thesis, we first review IPv6–based anycast communications. At present, there are several possible applications that are suitable for anycasting. We then raise several problems and provide possible solutions to these. Based on our findings, we propose two routing protocols for inter–segment anycast to support anycast–oriented communication. Our proposed architecture has the following characteristics: It (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. We also compare the proposed routing protocols and discuss which is most suited to reducing overheads as much as possible.

Keywords

Anycast Communication, Network–layer Anycast, Anycast Application, Anycast Routing

Contents

List of Figures

List of Tables

1 Introduction

Recently, use of the Internet typified by the WWW (World Wide Web) and e-mail have spread extensively. The IPv4 (Internet Protocol version 4) [1] supporting the current Internet was designed in the 1970s, and has been extended to satisfy several requirements that have changed with the times (e.g., multicasting support). IPv4 has been remarkably resilient, despite its age, but it is beginning to develop problems. More importantly, there is a growing shortage of IPv4 addresses to assign all new machines that are being added to the Internet. Some technologies, such as CIDR (Classless Inter-Domain Routing) [2] and NAT (Network Address Translator) [3], have been proposed to rectify this situation. However, these technologies cannot solve the problem fundamental of a lack of addresses.

With the trigger of the IPv4 address space problem, research on a next-generation Internet Protocol started around 1994. After long discussions by the IETF (Internet Engineering Task Force), the IPv6 (Internet Protocol version 6) [4] became a proposed standard as the next-generation Internet Protocol from the three major candidate protocols that were proposed. Because the IPv6 has too wide an address space (2^{128}) it is not necessary to use NAT technology in the network for mitigating the address exhaustion. Also, all IPv6 nodes can have a unique IPv6 address, and real end-to-end communications can be achieved in the network.

The IPv6 solves a number of problems raised by IPv4 (e.g., security issues), in addition to the limited number of available IP addresses. It also introduces enhanced new features (e.g., autoconfiguration). The situation with the Internet has been changing dramatically, and the areas of application in the Internet are spreading widely. The IPv6 is being used not only on traditional communication nodes but also on new types of nodes (e.g., mobile nodes, automobiles, and home appliances).

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 [5], which stated that the motivation for anycasting was to drastically simplify the task of finding an appropriate server on the Internet. 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.

The notion of anycasting is not only limited to the network (i.e., IP) layer, but can also be achieved in other (e.g., application) layers. As will be discussed in Section 2, network– and application–layer anycasting have both strengths and weakness, and we will focus on network–layer anycasting in this thesis.

However, IPv6 anycasting still has several problems that need to be clarified within the context of the current specifications. First, we should have clear answers to "what kinds of applications are suited to using anycasting?" and "what are the advantages/disadvantages of using anycasting for applications?"

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. The router should play an active role in deciding the destination network so that anycast packets can be appropriately forwarded. We need to design and implement an anycast routing protocol that is suited to anycast applications. We also need a migration scenario where the Internet will gradually be able to support anycasting. For example, anycast routing should work adequately (though not necessarily optimally) even when only a few nodes and/or routers support anycast routing within the Internet. We will discuss problems with anycast routing and present our proposal in Section 3.

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 [6].

Of course, security considerations are also very important in anycast communications. A malicious node may *hijack* communication by announcing a spoofed advertisement through which it receives packets for the anycast address. Thus, some authentication mechanism is necessary to actually validate anycasting. Public-key-based authentication is one promising approach to counteract this problem, but further considerations on security are beyond the scope of this thesis.

The rest of this thesis is organized as follows. The next section discusses applications suitable (and not suitable) to network–layer anycast communications, and lists what kinds of functionalities are required to realize these applications. Section 3 discusses the proposed anycast routing architecture. In Section 4, we describe the specification of our architecture and we test and evaluate our proposed protocols in Section 5. Finally, we summarize our work and describe our future research topics in Section 6.

2 Anycast Communication

In this section, we will first present the characteristics of anycast addresses and several scenarios where it is suitable to use anycasting. We will then discuss some of its advantages. After that, we will list what kinds of functionalities are required to achieve these scenarios.

2.1 IPv6 Addressing Architecture

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 thesis. 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}) (Figure 1). How appropriately the destination node is chosen from anycast membership depends on the anycast routing protocol, which will be discussed more in Section 3.

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 (listener)	server

2.2 Application Scenarios

In our previous work, we showed some applications suitable to anycasting [7]. This subsection reviews what kinds of applications are suited to anycast communication. An excellent survey on IPv6 anycast addresses can be found in [8], where they introduced several applications that can be achieved through anycasting. One important example is *server location*, through which the sender host can choose one of many functionally identical hosts. As a result, load distribution among anycast hosts can be achieved if we utilize some appropriate anycast routing method, where anycast requests are evenly distributed to hosts.

Another example is *service location* [8], where the sender host can communicate with an optimal (e.g., minimum delay or largest throughput) host chosen from multiple anycast hosts by specifying the anycast address. This is especially useful in dynamically changing environments such as mobile ad hoc networks. While this kind of service can be obtained through application–layer anycasting, the node can communicate automatically with an appropriate (e.g., nearest) server through network–layer anycasting.

In summary, the advantage of network–layer anycasting lies essentially in that it provides a simple mechanism where the source node can receive a specific service without the knowledge of service nodes and/or networks. However, this immediately implies that it is difficult to ob-

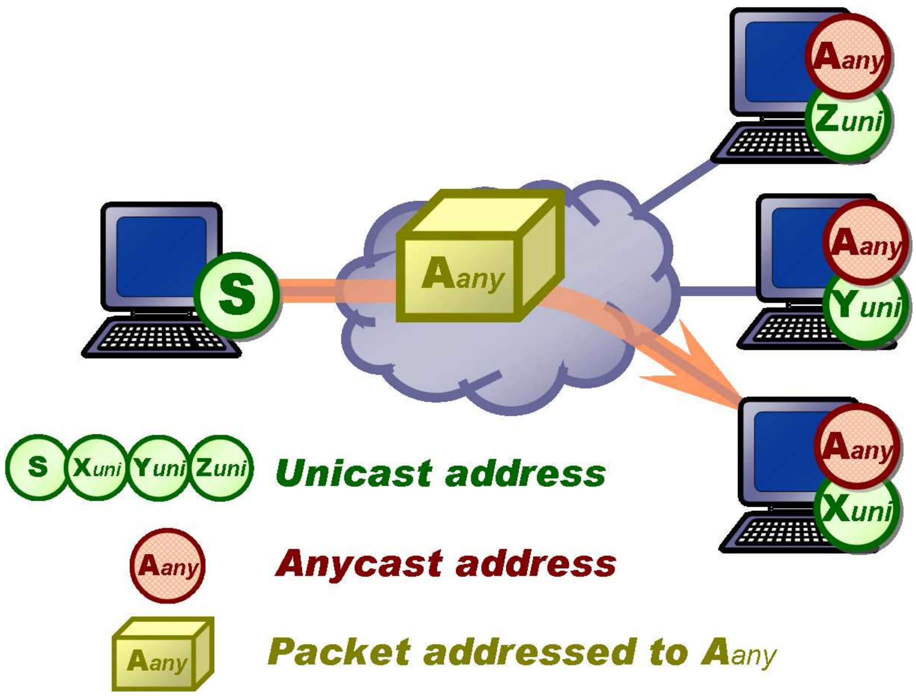


Figure 1: Anycast Communication

tain rich functionalities. Moreover, the additional anycast examples that are listed below clearly demonstrate this.

2.2.1 Host Auto-Configuration (Plug&Play)

By defining and assigning a well known anycast address to widely used applications (e.g., Domain Name Services (DNS), and proxy services), the user can reach these without knowing the location (i.e., their unicast address) of the server [5]. Moreover, the user can utilize these applications everywhere by specifying a well known anycast address (Figure 2). DNS resolvers would no longer have to be configured with the IP addresses of their DNS servers, and it would be sufficient to send a query to a well known DNS anycast address. This functionality can also be used for plug&play. Auto configuration through anycasting is quite effective during the primitive setup phase (e.g., DNS server cannot be used). When a host is plugged in, its IPv6 address is configured, automatically. However, to achieve true plug&play, various settings are necessary (e.g., configuring unicast

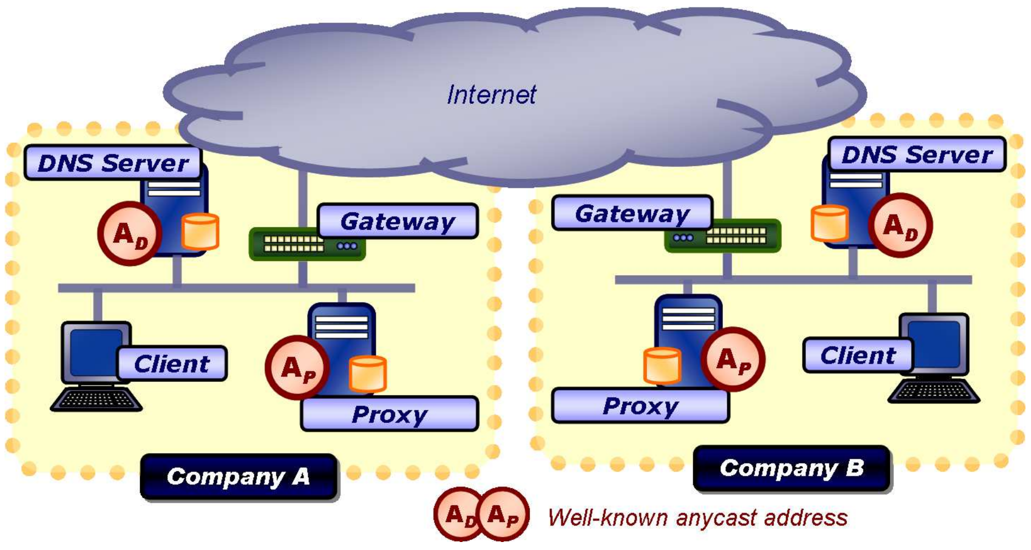


Figure 2: Host Auto-Configuration (Plug&Play)

addresses of DNS and proxy servers). If a well-known anycast address is installed in the hardware beforehand, end users can utilize these services without configuration.

Since auto configuration of hosts is often done to discover locally-provided servers (e.g., DNS, SMTP, or proxy servers), anycast routing protocol crossing segments are not mandatory, which means that the scalability problem is not an issue here. One remaining problem is that if the application uses stateful protocols like TCP, an additional mechanism as will be discussed in Subsection 2.4 is necessary.

2.2.2 Gate to Overlay Network

The *gate to overlay network* is another example of an anycast application. A distributed application like a P2P (Peer-to-Peer) service constructs a logical network topology among nodes participating in the service. However, the peer needs to know the address to connect the logical network prior to participating in the service. Each peer only specifies the anycast address in order to participate in the logical network and one of the participating peers becomes the *gate of the logical network* for the new node, which should be determined by the anycast routing protocol (Figure 3). The

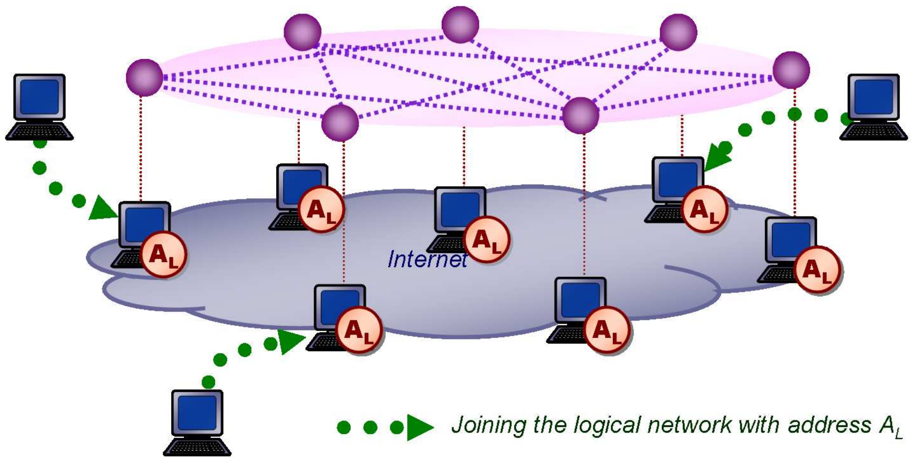


Figure 3: Gate to Overlay Network

advantage of this model is that all processes are completed within its own protocol. Furthermore, even when the connected peer leaves the logical network, it is possible to continue participating via another peer, which is automatically changed by the anycast routing protocol. This cannot be attained with any of the existing technologies.

It has been preferable in this application for the topology of the logical overlay network to reflect that of the underlying physical network. Anycasting makes this possible if the routing protocol of network–layer anycasting is appropriately operated based on the underlying network topology. When this occurs and a peer wants to connect to its nearest peer to participate in the logical network, the range of advertisements for routing information on the anycast address can be restricted, which relaxes scalability problems.

2.2.3 Improving System Reliability

Anycasting permits multiple number of hosts with the same address and by increasing the number of hosts, system reliability can be improved because it still works even when some hosts fail (Figure 4).

Through a mechanism that was similar to this, we would have services that had better proper-

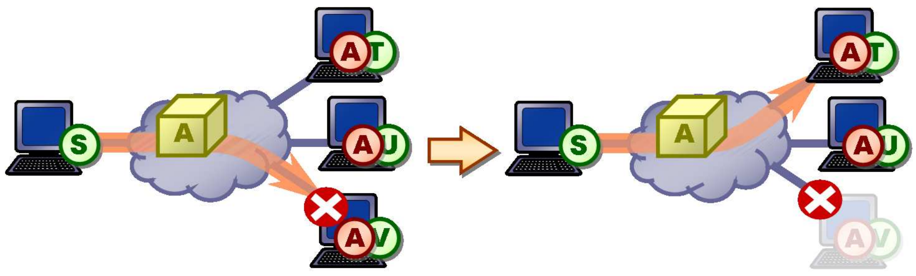


Figure 4: Improving System Reliability

ties to tolerate faults. However, if anycast packets are destined for other segments, we again need anycast routing.

2.2.4 Local Information Service

We can consider other new services and those for local information such as “Emergency Calls (e.g., call for an ambulance)” can be introduced by defining a common anycast address to assign that certain service. This means a client can reach a nearby service offered by the server by using that address. By assigning a common anycast address to servers providing “restaurant information,” the client can know the location of nearby restaurants by only communicating with the anycast address. Particularly by combining it with mobile communication, this kind of local information can be provided without reconfiguration and services like “Emergency Calls” can be provided at the IP–layer.

2.3 Advantages of Anycasting

As previously discussed, only anycasting has advantages that existing application-layer technologies cannot achieve and we will describe these in the following.

2.3.1 Transparency

Anycast addresses are allocated from unicast address space. Thus, they are syntactically indistinguishable from unicast addresses. Therefore, the clients do not need to know whether the destination address is unicast or anycast. Only servers that have anycast addresses are explicitly configured to know that it is an anycast address. This obviates the need to change client and server applications.

Moreover, in application-layer anycasting, to access an appropriate server providing a specific service, service location tools (i.e., directory server) are necessary. However, even if the server works correctly, trouble with the directory server may make the server unreachable.

Network-layer anycasting, on the other hand, can help in selecting an appropriate server by only specifying an IP address. The directory server is no longer necessary to find services.

2.3.2 Separation of Application and Routing Policy

Each application has different levels of appropriateness. In application-layer anycasting, this application-dependent information managed for each application is required to attain appropriate server selection. Therefore, a different routing architecture is required for each application to support different levels of appropriateness.

Network-layer anycasting, on the other hand, reduces this onerous task and can be used for all applications. Each level of appropriateness can be mapped to one criterion, and the routing architecture for network-layer anycasting works based on this criterion. Each application can then use the routing architecture to support different levels of appropriateness. Anycasting relaxes the routing architecture from application-dependent information.

2.4 Problems and Possible Solutions

This subsection discusses problems that remain with the current specifications for IPv6 anycasting because they contain too few definitions. It also reviews several solutions.

2.4.1 How a host announces its participation in anycast membership

There are no standards for nodes to announce that they can receive anycast packets except for publishing routing information for anycast addresses, i.e., the node must be a *router* in the IPv6 specifications. This implies that a host (i.e., not a router) that intends to participate in (or leave) anycast membership must have a different capability of notifying the nearest anycast router of the status (joining/leaving). The method of finding the host participating in anycast membership (called *anycast host* below) differs depending on the location of the anycast host. Haberman and Thaler [9] proposed an extended version of MLD (Multicast Listener Discovery) if the anycast host and router were located within the same segment. A case where they are located in different segments is discussed in Section 4.

2.4.2 How an upper–layer stateful protocol is supported

Anycasting has a stateless nature, where it cannot assure that all packets belonging to the same anycast address will go to the same destination node. The destination may change during communications under changing network conditions, unless the routing protocol corrects for this. It is easy to think that the router can maintain the per–flow state in anycasting, where all anycast packets belonging to the same flow can be forwarded to the same destination host, but clearly this must be avoided to achieve a robust and scalable routing protocol. Also, to expand anycasting to have wider applications, complexities with both implementation and deployment should be simplified as much as possible. We need to introduce a situation where there are minimum modifications and/or replacements to existing applications and protocols. Thus, anycast routing is required to determine the destination networks of anycast packets on a packet-by-packet basis.

However, this leads to serious problems in that stateful protocols like TCP cannot be supported. When a host initiates a TCP connection to an anycast address, the receiving host cannot set its own anycast address as the source address for the acknowledgement packet. The IPv6 specifications prohibit the anycast address from being set into the source address field of the packet header. This is basically because an IPv6 anycast address does not identify a single source node. During anycast communication, there is the possibility that the target host associated with the anycast address may change because of changing network conditions. If the protocol allowed the anycast address to be set into the source address of the packet, the receiving host could not be sure that all packets sent during the communication had come from the same host. This means that the host cannot receive acknowledgement for a packet sent to an anycast address. Also, successive packets cannot be delivered to the same node. Some solutions to this problem have been proposed [8], but they require modifications to existing applications and/or upper–layer protocols. Weber and Cheng [8] recently discussed the *anycast address mapper*, which had been proposed by Oe and Yamaguchi [10]. It translates anycast addresses to corresponding unicast addresses at the host receiving anycast packets and this is done prior to anycast communication. However, each application must be modified in using the anycast address mapper to map an anycast address before communication begins. In our previous work, we have proposed AARP [6]. It is implemented as a kind of dynamic linkable library overwrites original APIs, which resolves the anycast address into the corresponding unicast address prior to establishing the communication. It is similar to *anycast address mapper*, but there is no need to modify upper–layer protocols or applications.

2.4.3 How anycast routing is achieved

The current anycast standard does not define the routing protocol, and there are several challenging issues that need to be resolved in designing anycast routing protocols.

1. Scalability issue

A conventional IP router aggregates multiple entries that have the same prefix for the destination address (or network), and the same output interface (i.e., the same direction) to reduce the size of the routing table. However, the routing entries for anycast addresses cannot be aggregated because anycast membership locations are widespread regardless of their actual prefix. Hence, 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. For example, if an application requires a faster response, the propagation delay between the source node and anycast node is extremely important, i.e., the nearest node for anycast membership should be chosen. The criteria for anycast routing strongly affects anycast communication capabilities.

3. Security issues

   Maintaining anycast membership is particularly important. The easiest way for a host that intends to gain membership is for it to simply advertise the routing entry for the associated anycast address to the router. However, such an approach can sometimes lead to serious security problems in that the anycast host can freely add or delete anycast entries in the routing table.

One important feature of anycast addresses is that they should be assigned from the same address space as a unicast address and are thus syntactically indistinguishable from unicast addresses. RFC1546 originally recommended assigning anycasting its own address space because it greatly expected this to reduce the risk of applications mistakenly failing to recognize anycast addresses. However, when we consider the deployment of anycast routers, it is very likely that some routers on the Internet will not be able to process anycast addresses. If these addresses are allocated in a unicast address space, it is not necessary for legacy routers to deploy special operations for communication: i.e., these simply pass on anycast packets through unicast forwarding, expecting packets can be reached. As it is difficult for an anycast router to decide whether the receiving packet's destination address is anycast or unicast, designing an anycast routing protocol is problematic.

There have been several proposals for an anycast routing protocol [8, 11], but to the best of our knowledge, none of these have conformed to IPv6 anycast specifications and anycast addresses are allocated in their own address space, which is different from the unicast address space. However, the routing protocol we proposed in Section 3 allows the same space to be used for both unicast and anycast addresses.

2.4.4 How a well known anycast address is decided

Although it is not a technical issue, a well known anycast address should be defined in advance to achieve auto-configuration or plug&play. Currently, the only well known anycast address is the Mobile IPv6 Home-Agents Anycast Address, which has been defined [12].

2.5 Requirements for Network Components

As we described in our previous work [13], we listed what kinds of functionalities were required to achieve the anycast applications described in Subsection 2.2.

2.5.1 Basic Functionality

Before considering each application, we will discuss the basic (i.e., minimum) functionalities to achieve anycast communications.

- Supporting unicast forwarding

  Anycast addresses are syntactically indistinguishable from unicast addresses. Then,

  - if the destination address of the receiving packet is not an anycast address, each router must forward the packet as a unicast packet.

  Moreover, this functionality is necessary to retain backward compatibility.Therefore,

  - if we replace existing routers with new routers that support the routing mechanism for anycasting, they must do all the tasks existing routers do.

- Reaching server

  To ensure the server can be reached, it is necessary for routers to forward the packet destined for the anycast address. To forward packets,

  - routers should have the routing entry for the receiving packet destined for the anycast address.

  If the router does not have the routing entry for the receiving packet in the routing table, it simply discards the packet.

  Moreover, only nodes with anycast addresses can identify the anycast address, because it is not identified by the address itself. Therefore, routers cannot identify anycast addresses unless the nodes notify them. Then,

  - each node having an anycast address must notify a neighbor router of its membership information.

  These two functionalities are not necessary when the address prefix of the anycast address is determined based on the node's unicast address prefix. With unicast, routers forward packets by prefix routing. Therefore, packets with a unicast address as the destination address can reach the last hop router through unicast. When the last hop router receives anycast packets, the anycast receiver that actually communicates with the client is determined by the Neighbor Cache (ND) [14] of the last hop router.

2.5.2 Functionalities to Gain Anycasting Advantages

We will next discuss what kinds of functionalities are required to gain anycasting advantages.

Transparency

The following functionalities are required to achieve transparency.

- If a router receives a packet destined for an anycast address, the router must forward it to one node that has the anycast address.

  If multiple packets are sent to an anycast address, multiple reply packets may be delivered to the client. This does not give much transparency to the client because the client application cannot decide the appropriate server.

  Of course, if the router only keeps one routing entry for the receiving packet, it simply forwards the packet according to the entry. However, if there are multiple candidate entries for receiving packets, following the functionalities are necessary.

  - Routers should have node selection criteria.
  - Routers should have information to select one entry.
  - Servers may notify a neighboring router of its preference value (e.g., metric).

  These are unique functionalities for anycast communication.

- Well-known anycast address may be defined.

  A well known anycast address needs to be defined for each service to find specific servers without directory servers. This can be done with the IANA (Internet Assigned Numbers Authority) [15].

Separation of Application and Routing Policy

The following functionality is required to achieve this advantage.

- Application-dependent information should be hiden behind the servers as much as possible.

  If routers provide a common routing architecture among different applications, servers should not tell some application-dependent information to the routers. The servers providing same service (i.e., having same anycast address) only notify the level of appropriateness mapped to one criterion. Then, following the functionalities are necessary.

  – Routers can forward anycast packets based on the preference value which servers specified.

  – Servers may notify a neighboring router of its preference value (e.g., metric).

2.5.3 Other Functionality

Moreover, the following functionality is required to achieve anycast communication.

Introducing Scope

As described in [16], using anycast addresses may limit scalability. To improve routing-protocol scalability, it is necessary to limit the range of sending-route information to an anycast address. Then,

- routers can handle the scope to limit the range of sending-route information.

2.5.4 Summary

The following functionalities are required for each network component.

Server functionalities (requirements)

- Servers must notify a neighboring router of membership information.
- Servers may notify a neighboring router of the preference value (e.g., metric).

Router functionalities (requirements)

- Routers must support unicast forwarding
- Routers should have the routing entry for the receiving packet destined for the anycast address.
- Routers should have node selection criteria.
- Routers should have information to select one entry.
- Routers may handle scope.

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

3 Design of Anycast Routing Architecture

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.

3.1 Design Choices and Models

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

3.1.1 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.

3.1.2 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.

3.1.3 Modifying Existing Routing Protocols

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

Additionally, we choose following models when we design our routing protocols to satisfy the functionalities described in Section 2.

3.1.4 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.

There are three scenarios using the metrics above:

- receiver metric only
- link metric only
- path metric

The link and receiver metrics can be combinated. The path metric is the combination of the receiver and link metrics. In the combination case the path metric is the link metric when the value of the receiver metric is 0.

3.2 Proposed Architecture

Figure 5 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

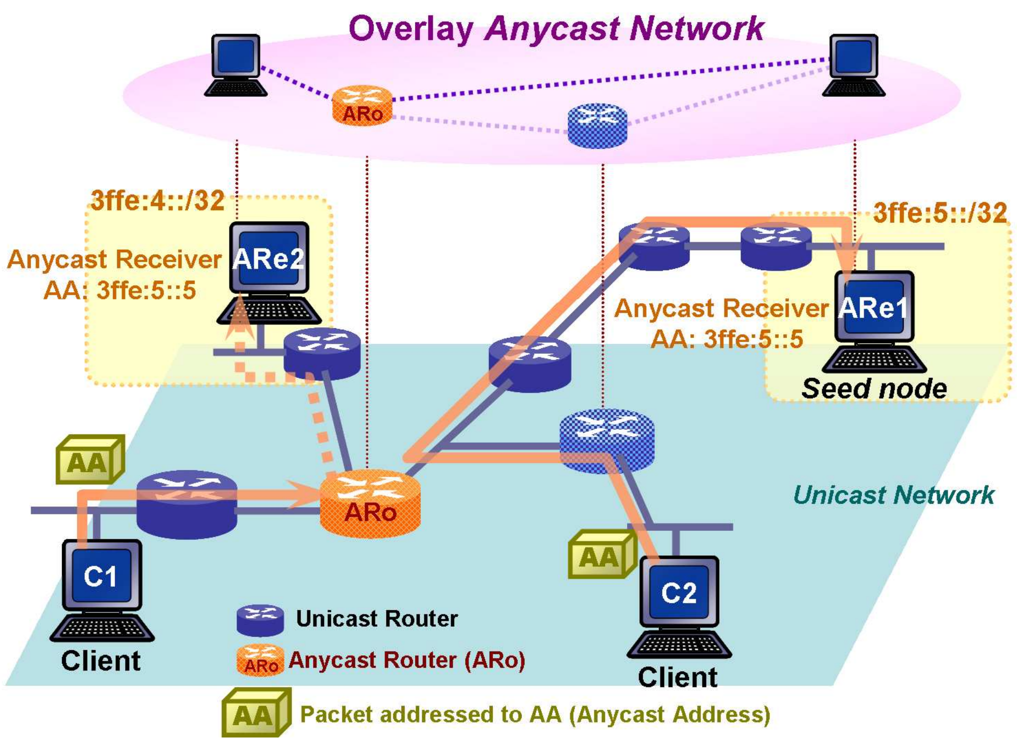


Figure 5: Proposed Architecture

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 5 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 5, 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 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.

4 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 therefore 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 [17]. 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* extensions to OSPF (AOSPF), and (3) the Protocol Independent *Anycast* Sparse Mode

Table 2: Classification of Routing Protocols

	Distance-Vector	Link-State	Core-Based-Tree
Unicast	RIPng [18]	OSPFv3 [19]	
Multicast	DVMRP [20]	MOSPF [21]	PIM-SM [22]
Anycast	ARIP	AOSPF	PIA-SM

(PIA-SM). In our previous work, we designed the (3) PIA-SM [6], and Matsunaga [23] 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 6) and the difference between the two above–mentioned routing protocols are in Step 2.

1. Initiate anycast membership (Subsection 4.1)

   The anycast router collects information on nodes that intend to join anycast memberships.

2. Construct and update routing table (Subsection 4.2)

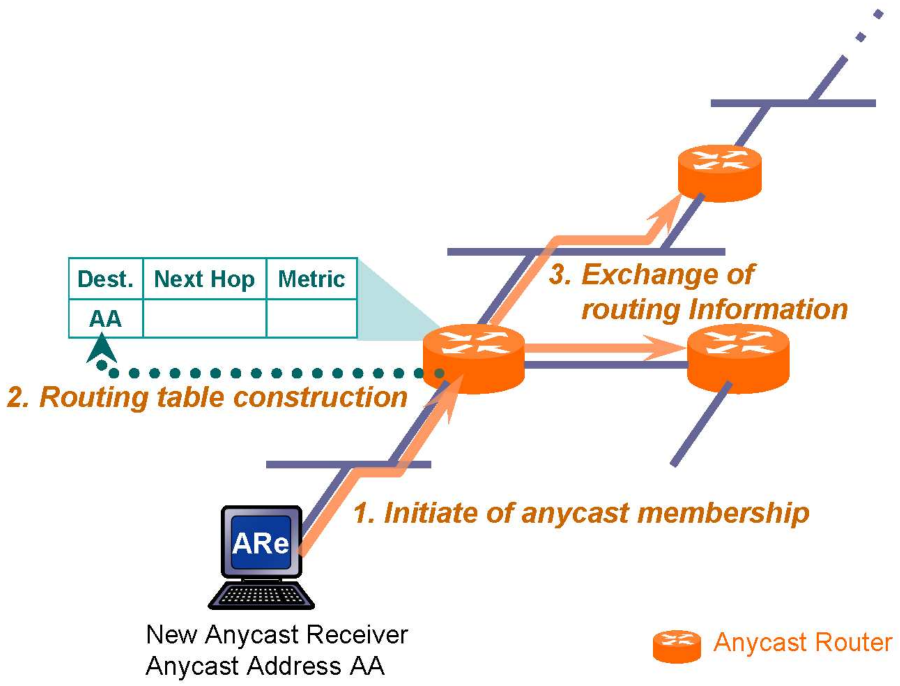
   According to the information collected, anycast routers construct their own routing tables and then exchange routing information with one another to reconfigure these.

3. Forward anycast packets (Subsection 4.3)

   The anycast router then forwards the arriving anycast packet based on its own routing table. If it has multiple entries associated with a specific anycast address, the anycast router needs to select an em appropriate entry from them.

   The anycast router forwards anycast packets based on the routing table constructed in Step 2. Note again that this process 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 the above steps separately.



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 6: Overview of Anycast Routing Protocol

4.1 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 [9]. 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.

All edge routers must become anycast routers with the capability of ARD to enable inter-segment anycast routing. However, this is unrealistic in the early stages of anycasting. One possible solution is to locate the authorization node on the same segment as the anycast router. A node with the capability of forwarding an anycast packet establishes a tunneling path to the authorization node. After establishing the tunneling path, the authorization node advertises anycast address information to the anycast router through ARD. The tunneling path is only used to announce anycast routing information. As will be discussed in Subsection 4.5, we can use another effective approach to this problem if we can slightly change the existing unicast routing protocol.

Again note that the method by which anycast hosts are collected sometimes leads to serious security problems. The anycast router should have some mechanism that prevents illegal and/or spoofed anycast host notifications.

4.2 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.

4.2.1 Advertising Receiver Metric

Figure 7 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 7 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*.

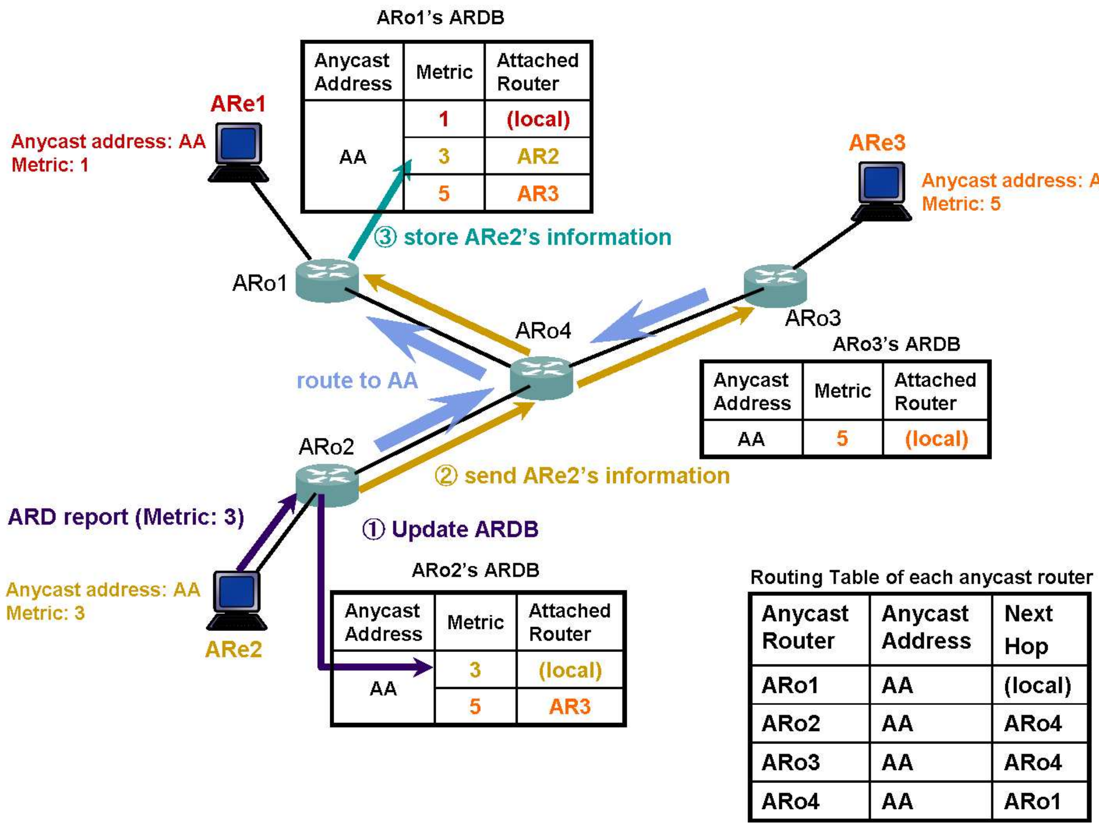


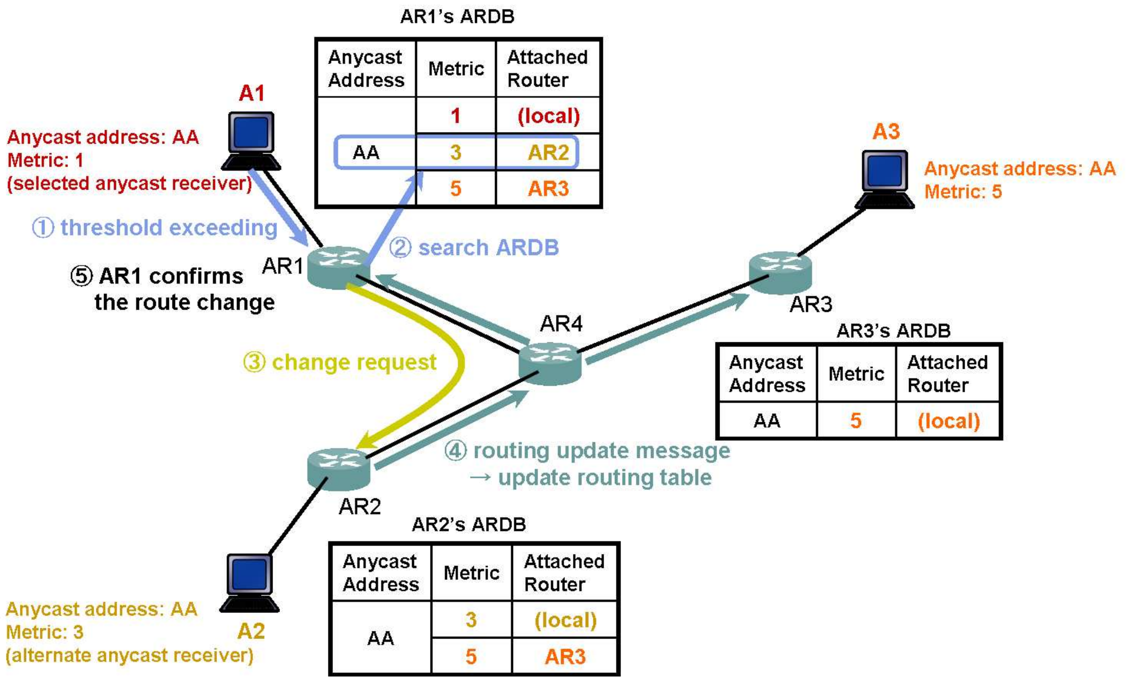
Figure 7: Basic Operation of Advertising Receiver Metric

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 following process is done to update the routing entry (See Figure 8).

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*



AR1's ARDB
Anycast Address | Metric | Attached Router
1 | (local)
AA | 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
AR4
③ change request
A3
Anycast address: AA
Metric: 5
AR3
AR3's ARDB
Anycast Address | Metric | Attached Router
AA | 5 | (local)
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 8: Route Update of Advertising Receiver Metric

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*.

4.2.2 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 9 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*.

AOSPF

Figure 10 has an example of updating the routing table with the AOSPF, which is operated as

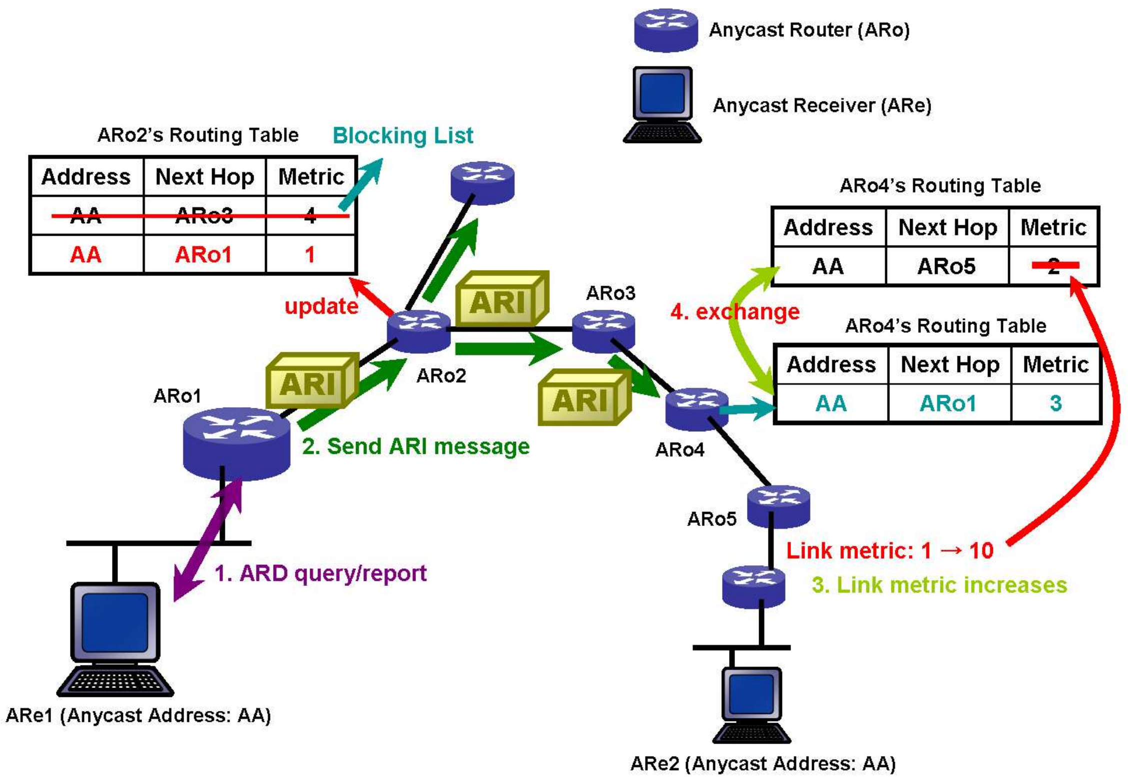


Figure 9: ARIP

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 [19]. 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 [17].

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 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

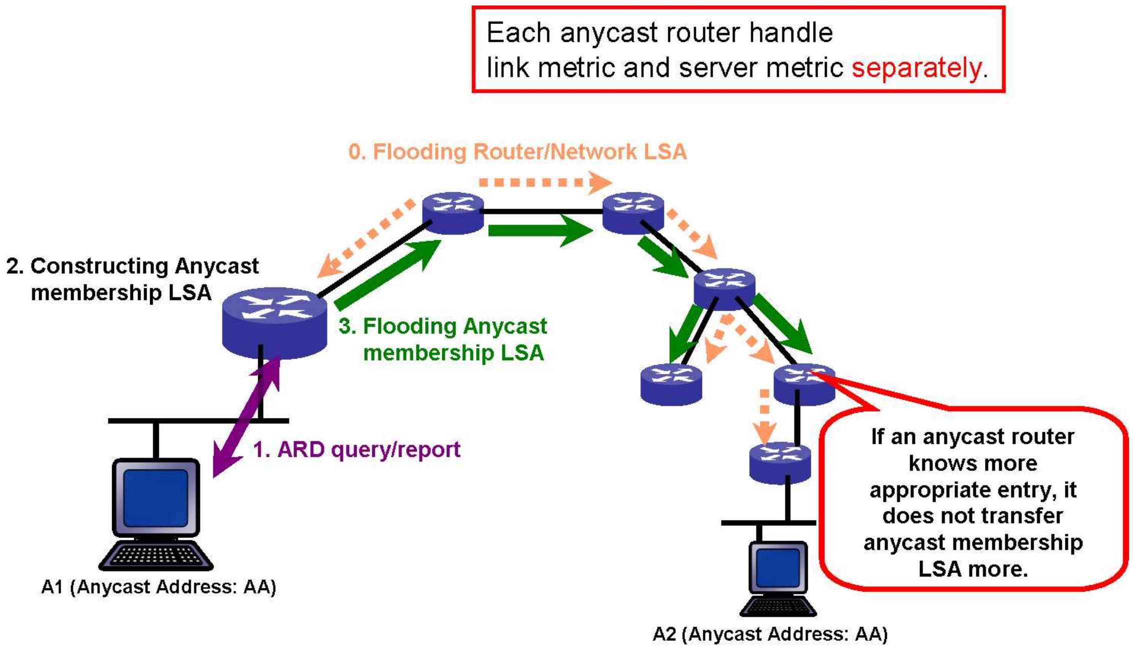

Each anycast router handle link metric and server metric separately.
0. Flooding Router/Network LSA
2. Constructing Anycast membership LSA
3. Flooding Anycast membership LSA
1. ARD query/report
If an anycast router knows more appropriate entry, it does not transfer anycast membership LSA more.
A1 (Anycast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

Figure 10: AOSPF

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.

4.3 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.

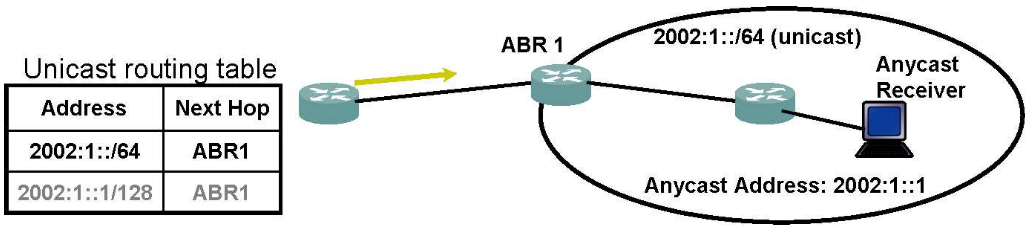


Figure 11: How to Reduce the Entries of Routing Table

4.4 Inter-Area Routing

The Advertising Receiver Metric procedure and the AOSPF use flooding to notify anycast receivers of information. These protocols may be useful in a small or limited size of network. However, they lack scalability if we deploy them in a large network, such as the Internet.

For more widespread use of anycasting, we need to maintain scalability. Therefore, we introduce the concept of inter-area routing to reduce the number of routing entries. Figure 11 is an overview of a method to reduce routing entries by using information on unicast routing and an inter-area routing mechanism.

The circle in Figure 11 indicates the boundary for an area. An anycast router existing on the boundary of the area is called as area border router (ABR). Anycast routers outside the area have this aggregated routing information (`2002:1::/64`). There is an anycast receiver whose anycast address is `2002:1::1` in Figure 11. The ABR may advertise this address and unicast routing information separately, but the anycast router outside the circle does not need to register information on the anycast address. This is because the anycast packet destined for `2002:1::1` can be delivered to the anycast receiver by only using unicast routing.

Thus, if the direction of the anycast routing entry is the same as the direction of the existing unicast routing entry, it is not necessary to retain the anycast address in the routing table. Packet sent to this anycast address can be transferred by the unicast routing.

We also introduce inter-area routing like OSPF [24] into AOSPF. We divided a large network (e.g., the network of a specific Autonomous System) into several small areas. In each area, the

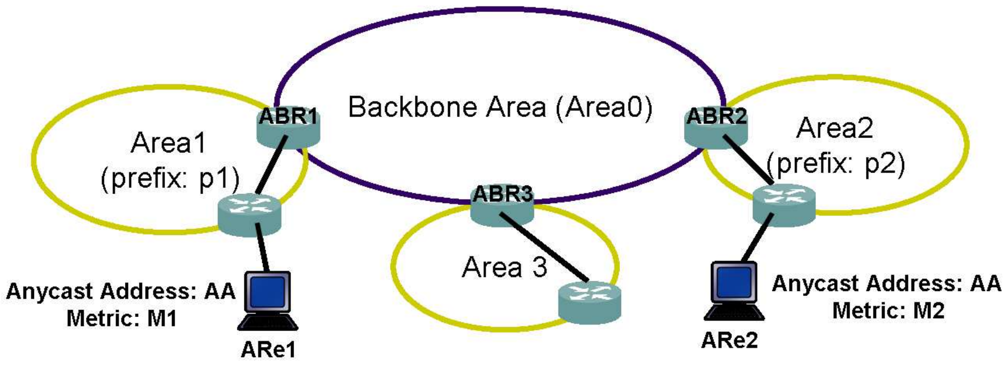


Figure 12: Inter-Area Routing (Advertising Receiver Metric)

AOSPF processes were executed as previously described. This meant that each area had its own topological database and corresponding graph.

The topology of an area is invisible from the outside. Conversely, routers internal to a given area know nothing about the detailed topology outside the area. This isolation enables the protocol to markedly reduce the routing traffic compared to treating the entire AS as a single SPF domain.

There are four areas in Figure 12 and the backbone is connected to the other three areas. The anycast routers in each area communicate with other anycast routers in other areas via the ABRs in the backbone. In this figure, we have assumed that anycast address AA is within the range of the address prefix of Area 1. The following cases occur according to the metrics of ARe 1 and ARe 2.

Case 1: $M1 < M2$

The routing information for anycast receiver ARe 1 is flooded into Area 1. The ABR 1 then sends this to all ABRs in the backbone area. If each ABR in the backbone area has a routing entry for anycast receiver ARe 2, it deletes the entry for ARe 2. If each anycast router in Area 2 has a routing entry for anycast receiver ARe 2, it deletes the entry for ARe 2.

After that, all anycast routers in the Area 1 keep the host entry for ARe 1. However, all anycast routers in other areas (e.g., ABR 2, ABR 3, and other non-ABR anycast routers in

Area 2 and Area 3) do not need to take the anycast address AA into consideration. They simply use unicast routing to forward the anycast packet destined for the AA.

Case 2: $M1 > M2$

The routing information for anycast receiver ARe 2 is flooded in Area 2. The ABR2 sends it to all ABRs in the backbone area. The ABR 1 can then receive the routing information for anycast receiver ARe 2. As a result, the ABR 1 sends the routing information for ARe 2 into Area 1.

After that, all anycast routers in Area 1, Area 2, and the backbone area have a host entry for ARe 2. Anycast routers in other areas (e.g., Area 3) do not need to take anycast address AA into consideration.

4.5 Transition to Fully Anycast-enable Network via Tunneling

As described in Section 2, it is very likely that some routers on the Internet will not be able to process anycast addresses. Therefore, we should consider the deployment scenario for anycast routers. As we have already mentioned in Section 3, an anycast router is connected to another anycast router via various kinds of logical peer-to-peer connections. In this subsection, we discuss an effective method using tunneling to gradually deploy anycast routers.

To design the transition mechanism, we first started with existing tunneling mechanisms. To the best of our knowledge, existing tunneling mechanisms encapsulate/decapsulate packets. The gateway of a specific network attaches an additional IPv6 header to the original packet, and sends it to the external network.

The attached IPv6 header contains the address of the tunnel end-point node. Then, it can be handled by the routers in the external network. Thus, this encapsulated packet can be delivered to the tunnel end-point node. The tunnel end-point node then decapsulates the packet and obtains the original packet. After this, the original packet is delivered to the destination node by checking the address for the original destination.

To use this encapsulation mechanism, the gateway node should know the address for the tunnel end-point node. In multicast, static configuration is used generally. The administrator configures the network interface for the gateway node with the address of the tunnel end-point. The interface is then connected to the tunnel end-point as a virtual link.

Another approach automatically finds the address of the tunnel end-point. There are some mechanisms that deploy IPv6 networks [25, 26] to achieve this. However, these automatic tunnel establishing mechanisms use the IPv4 address of the tunnel end-point to identify it. They also use the well-known address prefix for the gateway to recognize that the receiving packet is an encapsulated packet. This can be done because the IPv4 address of tunnel end-point can be embedded in the original IPv6 destination address. When the gateway receives this encapsulated packet, it removes the well-known address prefix indicating that the packet is encapsulated one. Then, the gateway can extract the IPv4 tunnel end-point address automatically. Consequently, we cannot use the same approach to find existing anycast routers. However, a few parts of the mechanism described in [25] can be used. It uses a well-known anycast address.

We introduced a well known anycast address which identifies the anycast router. This anycast address is called the AllAnycastRouters address. All anycast routers have the AllAnycastRouters address and advertise this address to unicast routers. As a result of exchanging routing information, all unicast routers have the routing entry for this address. Therefore, packets destined for this address can be delivered to one of the anycast routers according to unicast routing. We also introduced a new message, called a *tunneling request* message, which is used by a new anycast router that wants to join the anycast network. The processes for the proposed tunnel establishing mechanism are as follows (See Figure 13):

1. If a new anycast router cannot connect to another router directly, it sends a *tunneling request* message to the AllAnycastRouters address to establish the tunnel between the new anycast

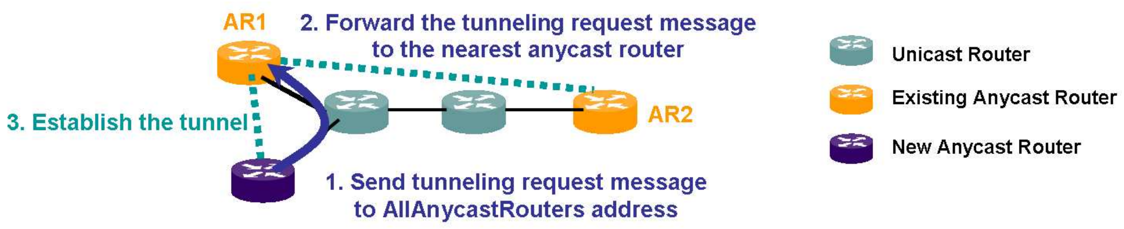

AR1
2. Forward the tunneling request message to the nearest anycast router
3. Establish the tunnel
AR2
1. Send tunneling request message to AllAnycastRouters address
Unicast Router
Existing Anycast Router
New Anycast Router

Figure 13: How to Establish a Tunnel

router and existing one.

2. Unicast routers forward this message to the nearest existing anycast router.

3. After receiving the *tunneling request* message, the existing anycast router replies an acknowledgement message to the new anycast router. From this confirmation, the new anycast router can know the existence of the anycast router, and then connect to the anycast router and establish the tunnel. Next, the existing anycast router set the tunnel interface. After that, both the new anycast router and the existing anycast router establish a tunnel by creating a tunnel interface.

After establishing the tunnel, the new anycast router communicates with the existing anycast router and obtains the route information the existing anycast router has.

However, there is no guarantee that the route established with the above mechanism is optimal for all routes which the existing anycast router knows. Consequently, we slightly changed this simple mechanism to optimize the route for anycast routers. Figure 14 describes a method of reconfiguring a tunnel between two anycast routers. This operation is done as follows:

1. As previously described, a new anycast router establishes a tunnel with its nearest anycast router (called tunnel end-point router).

2. If the tunnel end-point router has routing entries whose output interface is the same as the tunnel interface, it sends a *tunnel reconfiguration request* message to the new anycast router.

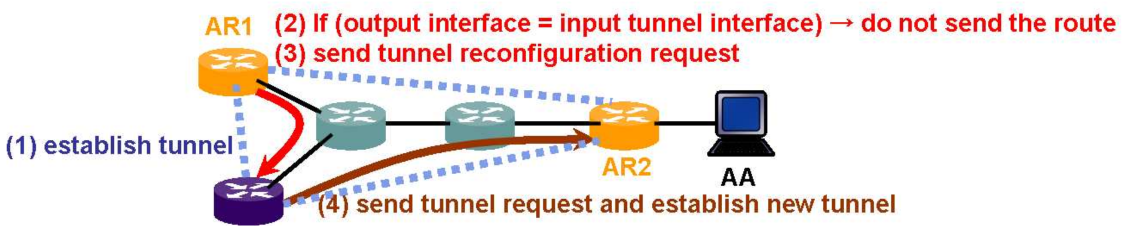

AR1
(2) If (output interface = input tunnel interface) → do not send the route
(3) send tunnel reconfiguration request
(1) establish tunnel
AR2
AA
(4) send tunnel request and establish new tunnel

Figure 14: Tunnel Route Optimization

3. If the new anycast router has not yet established a tunnel to the interface which the *tunnel reconfiguration request* message indicates, it sets up a tunnel to this interface by sending the *tunnel reconfiguration request* message

This tunneling mechanism is also useful to anycast receivers. If there are no anycast routers on the segment where the anycast receiver is, it can send an ARD report to the AllAnycastRouters address to communicate with the nearest anycast router.

5 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 4.

5.1 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.

5.1.1 Implementation of Routing Protocols

ARD

To transfer metrics from anycast receivers, we used a similar method to the extended version of MLD proposed by Haberman and Thaler [9]. 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. However, an additional extension was required to transfer a metric.

We consequently extended the ARD packet format to transfer metrics from anycast receivers. Figure 15 outlines the format for an ARD packet. The metric consists of both a 32-bit header field and an extendable value field. The header field includes *metric type* (2 bits) and *metric length* (16 bits). The metric field also has fourteen reserved bits for future extensions. The value field can be set up with an arbitrary length indicated by the length field in the header field.

The *metric type* field is used to support multiple selection criteria for the anycast address. As

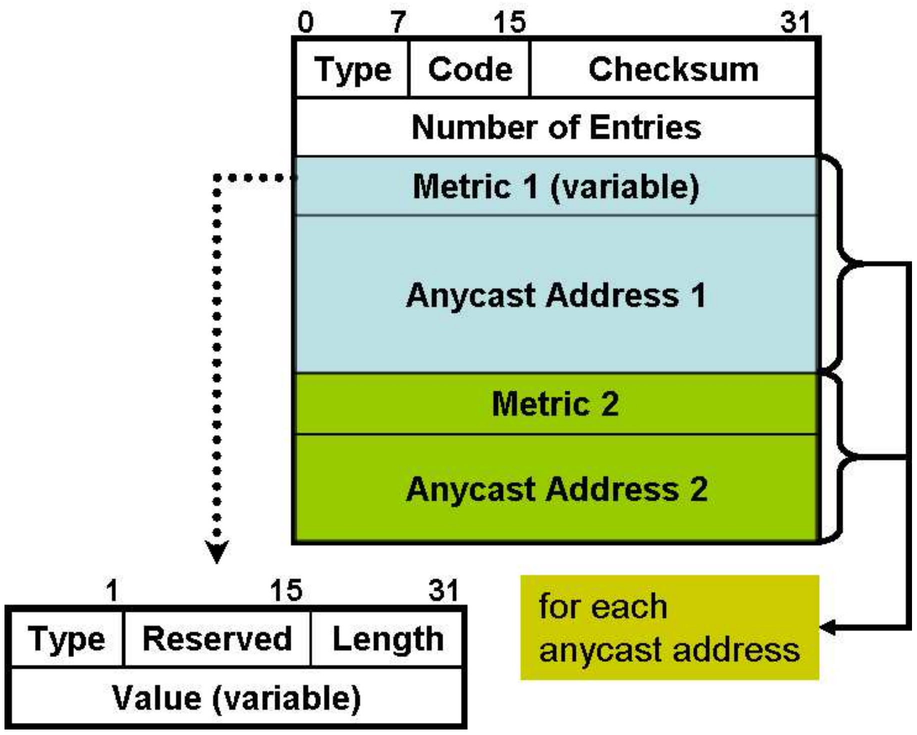


Figure 15: ARD Report Packet Format

Table 3: Metric Types

	Type	Feature
Node metric	00	Set by only anycast receiver
Path metric	01	Overwritten by each anycast router

defined in Section 3, we defined two *metrics* to classify the *metric types*: 1) a receiver metric, and 2) a link metric (See. Table 3).

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 as described in Figure 15.

ARIP and AOSPF

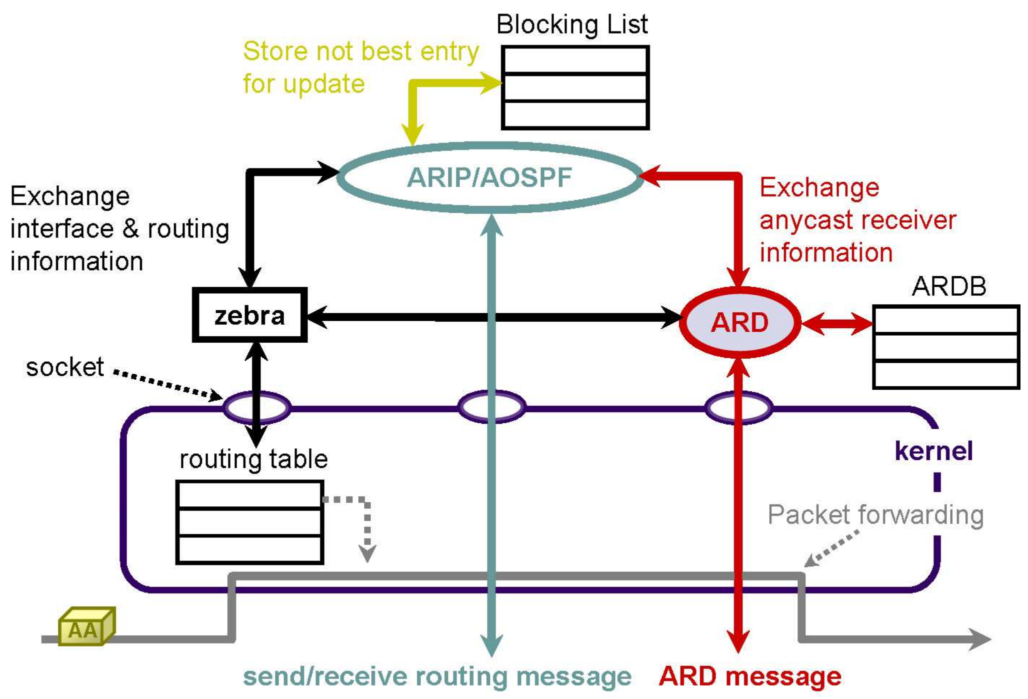

Blocking List
Store not best entry for update
ARIP/AOSPF
Exchange interface & routing information
Exchange anycast receiver information
zebra
ARD
ARDB
socket
routing table
kernel
Packet forwarding
AA
send/receive routing message
ARD message

Figure 16: Implementation Overview

We modified the GNU Zebra [27] to support the anycast routing protocols described in Section 4. GNU Zebra is free software that manages TCP/IP-based routing protocols. It supports multiple routing protocols including RIPng [18] and OSPFv3 [19]. 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 16 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 4. 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.

5.1.2 Experimental Results

Figure 17 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 17, 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 3, we first chose a *seed node* 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 (C 1 and C 2) 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 ARe 1. We then run programs of anycast routing protocols on anycast nodes (i.e., ARIP on anycast routers, ARD on anycast receivers). After running routing

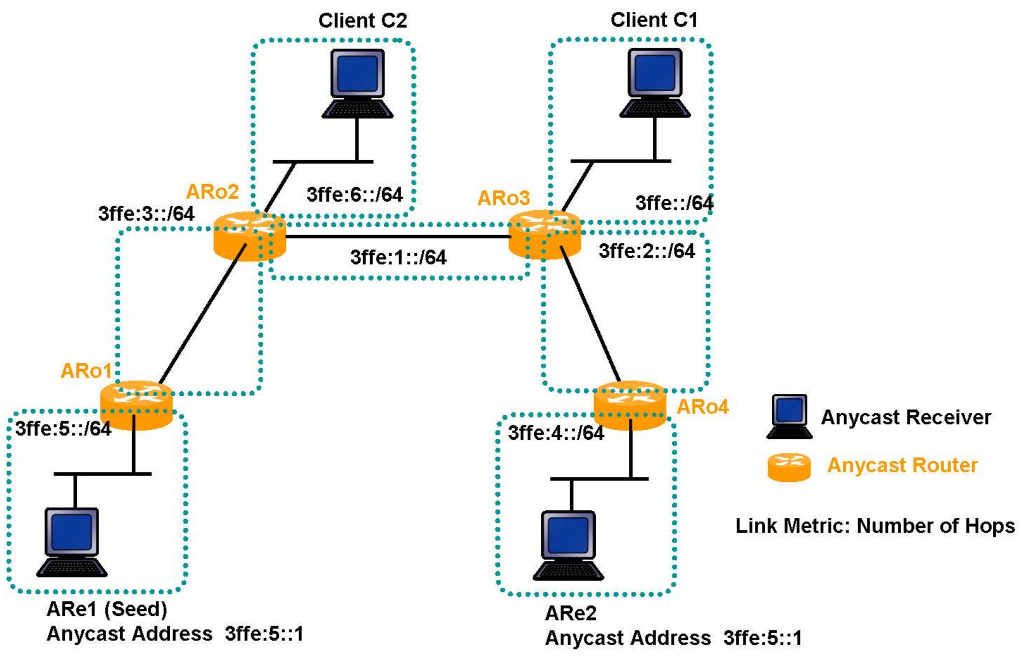


Figure 17: Network Environment for Implementation Experiment

programs, we again check routes to the anycast address. In the latter case, anycast packets from C 2 are delivered to ARe 1, while packets from C 1 are forwarded to ARe 2.

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 C 1. 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 ARe 1.

However, the anycast receiver ARe 2 is more appropriate for the client C 1 because the number of hops from C 1 to ARe 2 is smaller than the one from C 1 to ARe 1.

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 C 1 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 C 1 to the anycast address `3ffe:5::1` are reached to the appropriate anycast receiver ARe 2. Moreover, packets from C 2 still reach the anycast receiver ARe 1. For the client C 2, ARe 1 is more appropriate than ARe2. From the above results, we can conclude that our proposed routing protocols work as expected.

5.2 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

  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

Table 4: 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	$O(g)$	$O(g)$

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.

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 4.

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).

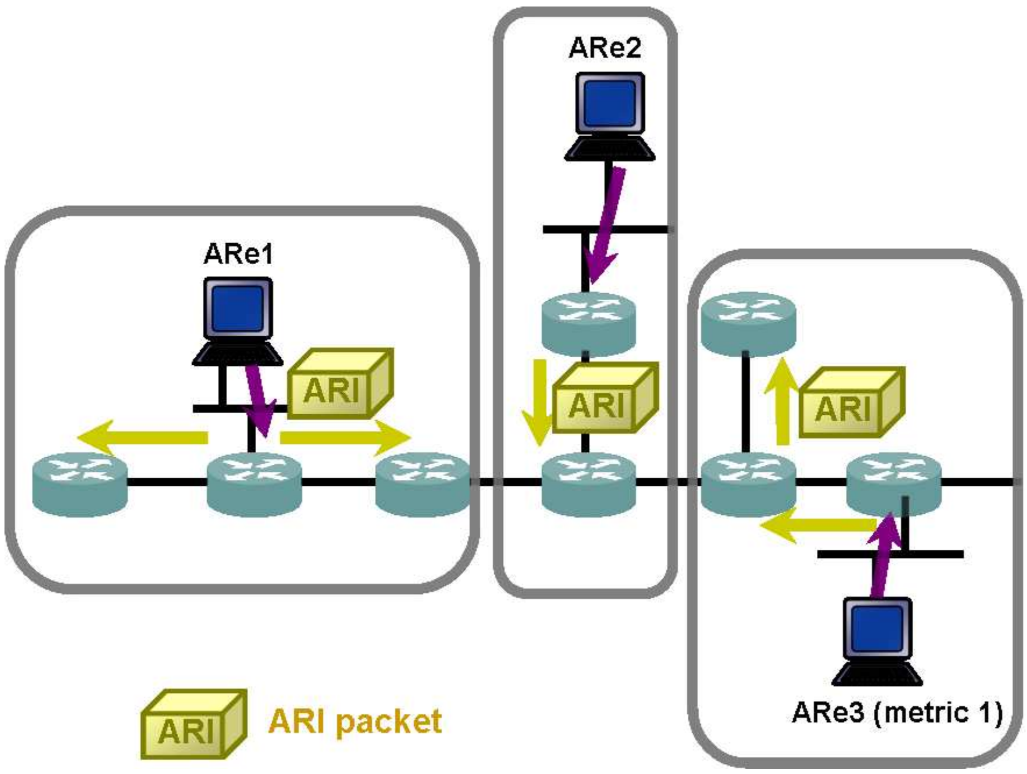


Figure 18: ARIP: Reduce the Time to Converge

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 18.

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. However, as described in 4.4, the number of routing entries can be reduced by introducing the inter-area routing.

6 Concluding Remarks

In this thesis, we discussed the characteristics of anycast addresses and several applications of anycasting. We then described current problems with anycasting. We particularly pointed out there were no routing protocols to handle anycast addresses, and that inter–segment anycast communications had not been achieved.

We next 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.

Moreover, we 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

I would like to express my sincere appreciation and deep gratitude to Professor Hideo Miyahara of Osaka University, who was my supervisor, for his tremendous help and valuable comments.

I would like to express my sincere appreciation to Professor Masayuki Murata, who introduced me to the area of computer networks and gave me advice throughout my studies and in the preparation of this thesis.

My deepest thanks also go to Dr. Hiroshi Kitamura of NEC Corporation who I sincerely wish to thank for his advice and assistance.

The work described in this thesis would not have been possible without the support of Lecturer Shingo Ata of Osaka City University. He constantly provided me with appropriate guidance and invaluable advice.

I would like to extend my appriciation to Dr. Ibrahim Khalil of Osaka University for his valuable comments and English support.

I am also indebted to Professor Shinji Shimojo, Associate Professor Naoki Wakamiya, Associate Professor Hiroyuki Ohsaki, Associate Professor Go Hasegawa, and Research Assistant Shin'ichi Arakawa and Research Assistant Ichinoshin Maki for their valuable support and advice.

Furthermore, I would like to express my gratitude to a number of friends and colleagues at the Graduate School of Information Science and Technology of Osaka University for their generous support, encouragement, and valuable advice.

Finally, I wish to thank my parents and all my friends for their spiritual support.

References

[1] J. Postel, "Internet protocol," *RFC791*, Sept. 1981.

[2] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless inter-domain routing (CIDR): an address assignment and aggregation strategy," *RFC1519*, Sept. 1993.

[3] P. Srisuresh and K. B. Egevang, "Traditional IP network address translator (traditional NAT)," *RFC3022*, Jan. 2001.

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

[5] C. Patridge, T. Mendez, and W. Milliken, "Host anycasting service," *RFC1546*, Nov. 1993.

[6] 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.

[7] S. Doi, S. Ata, H. Kitamura, and M. Murata, "IPv6 anycast for simple and effective communications," *to appear in IEEE Communications Magazine Internet Technology Series*, May 2004.

[8] S. Weber and L. Cheng, "A survey of anycast in IPv6 networks," *IEEE Communications Magazine*, vol. 42, Jan. 2004.

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

[10] M. Oe and S. Yamaguchi, "Implementation and evaluation of IPv6 anycast," in *Proceedings of the 10th Annual Internet Society Conference*, 2000.

[11] D. Xuan, W. Jia, W. Zhao, and H. Zhu, "A routing protocol for anycast messages," in *Proceedings of IEEE Transactions on Parallel and Distributed Systems*, pp. 571–588, June 2000.

[12] D. Johnson and S. Deering, "Reserved IPv6 subnet anycast addresses," *RFC2526*, Mar. 1999.

[13] S. Doi, I. Khalil, S. Ata, H. Kitamura, and M. Murata, "IPv6 anycast functionality/terminology definition," *Internet Draft* `draft-doi-anycast-func-term-01.txt`, Feb. 2004. work in progress.

[14] T. Narten, E. Nordmark, and W. Simpson, "Neighbor discovery for IP version 6 (IPv6)," *RFC2461*, Dec. 1998.

[15] IANA Home Page, "available at `http://www.iana.org/`,"

[16] J. Hagino and K. Ettikan, "An analysis of IPv6 anycast," *Internet draft* `draft-ietf-ipngwg-ipv6-anycast-analysis-02.txt`, June 2003. work in progress.

[17] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

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

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

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

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

[22] 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.

[23] 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.

[24] J. Moy, “OSPF version 2,” *RFC2328*, Apr. 1998.

[25] C. Huitema, “An anycast prefix for 6to4 relay routers,” *RFC3068*, June 2001.

[26] F. L. Templin, T. Gleeson, M. Talwar, and D. Thaler, “Intra-site automatic tunnel addressing protocol (ISATAP),” *Internet Draft* *`draft-ietf-ngtrans-isatap-17.txt`*, Jan. 2004. work in progress.

[27] GNU Zebra,, “available at `http://www.zebra.org/`,”

[28] D. B. Johnson, C. E. Perkins, and J. Arkko, “Mobility support in IPv6,” *Internet Draft* *`draft-ietf-mobileip-ipv6-24.txt`*, June 2003. work in progress.

[29] S. Deering, W. Fenner, and B. Haberman, “Multicast listener discovery (MLD) for IPv6,” *RFC2710*, Oct. 1999.

[30] G. S. Malkin, “RIP version 2,” *RFC2453*, Nov. 1998.

[31] A. Conta and S. Deering, “Generic packet tunneling in IPv6 specification,” *RFC2473*, Dec. 1998.