Network Working Group
Request for Comments: 2080
Category: Standards Track

G. Malkin
Xylogics
R. Minnear
Ipsilon Networks
January 1997

RIPng for IPv6

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies a routing protocol for an IPv6 internet. It is based on protocols and algorithms currently in wide use in the IPv4 Internet.

This specification represents the minimum change to the Routing Information Protocol (RIP), as specified in RFC 1058 [1] and RFC 1723 [2], necessary for operation over IPv6 [3].

Acknowledgements

This document is a modified version of RFC 1058, written by Chuck Hedrick [1]. The modifications reflect RIP-2 and IPv6 enhancements, but the original wording is his.

We'd like to thank Dennis Ferguson and Thomas Narten for their input.

Table of Contents

1. Introduction 2
1.1 Theoretical Underpinnings 3
1.2 Limitations of the Protocol 3
2. Protocol Specification 4
2.1 Message Format 5
2.1.1 Next Hop 7
2.2 Addressing Considerations 8
2.3 Timers 9
2.4 Input Processing 10
2.4.1 Request Messages 10
2.4.2 Response Messages 11

2.5 Output Processing 14
2.5.1 Triggered Updates 14
2.5.2 Generating Response Messages 15
2.6 Split Horizon 16
3. Control Functions 17
4. Security Considerations. 18
References 18
Authors' Addresses 19

1. Introduction

This memo describes one protocol in a series of routing protocols based on the Bellman-Ford (or distance vector) algorithm. This algorithm has been used for routing computations in computer networks since the early days of the ARPANET. The particular packet formats and protocol described here are based on the program "routed," which is included with the Berkeley distribution of Unix.

In an international network, such as the Internet, it is very unlikely that a single routing protocol will used for the entire network. Rather, the network will be organized as a collection of Autonomous Systems (AS), each of which will, in general, be administered by a single entity. Each AS will have its own routing technology, which may differ among AS's. The routing protocol used within an AS is referred to as an Interior Gateway Protocol (IGP). A separate protocol, called an Exterior Gateway Protocol (EGP), is used to transfer routing information among the AS's. RIPng was designed to work as an IGP in moderate-size AS's. It is not intended for use in more complex environments. For information on the context into which RIP version 1 (RIP-1) is expected to fit, see Braden and Postel [6].

RIPng is one of a class of algorithms known as Distance Vector algorithms. The earliest description of this class of algorithms known to the author is in Ford and Fulkerson [8]. Because of this, they are sometimes known as Ford-Fulkerson algorithms. The term Bellman-Ford is also used, and derives from the fact that the formulation is based on Bellman's equation [4]. The presentation in this document is closely based on [5]. This document contains a protocol specification. For an introduction to the mathematics of routing algorithms, see [1]. The basic algorithms used by this protocol were used in computer routing as early as 1969 in the ARPANET. However, the specific ancestry of this protocol is within the Xerox network protocols. The PUP protocols [7] used the Gateway Information Protocol to exchange routing information. A somewhat updated version of this protocol was adopted for the Xerox Network Systems (XNS) architecture, with the name Routing Information Protocol [9]. Berkeley's routed is largely the same as the Routing

Information Protocol, with XNS addresses replaced by a more general address format capable of handling IPv4 and other types of address, and with routing updates limited to one every 30 seconds. Because of this similarity, the term Routing Information Protocol (or just RIP) is used to refer to both the XNS protocol and the protocol used by routed.

1.1 Theoretical Underpinnings

An introduction to the theory and math behind Distance Vector protocols is provided in [1]. It has not been incorporated in this document for the sake of brevity.

1.2 Limitations of the Protocol

This protocol does not solve every possible routing problem. As mentioned above, it is primarily intended for use as an IGP in networks of moderate size. In addition, the following specific limitations are be mentioned:

- The protocol is limited to networks whose longest path (the network's diameter) is 15 hops. The designers believe that the basic protocol design is inappropriate for larger networks. Note that this statement of the limit assumes that a cost of 1 is used for each network. This is the way RIPng is normally configured. If the system administrator chooses to use larger costs, the upper bound of 15 can easily become a problem.

- The protocol depends upon "counting to infinity" to resolve certain unusual situations (see section 2.2 in [1]). If the system of networks has several hundred networks, and a routing loop was formed involving all of them, the resolution of the loop would require either much time (if the frequency of routing updates were limited) or bandwidth (if updates were sent whenever changes were detected). Such a loop would consume a large amount of network bandwidth before the loop was corrected. We believe that in realistic cases, this will not be a problem except on slow lines. Even then, the problem will be fairly unusual, since various precautions are taken that should prevent these problems in most cases.

- This protocol uses fixed "metrics" to compare alternative routes. It is not appropriate for situations where routes need to be chosen based on real-time parameters such a measured delay, reliability, or load. The obvious extensions to allow metrics of this type are likely to introduce instabilities of a sort that the protocol is not designed to handle.

2. Protocol Specification

RIPng is intended to allow routers to exchange information for computing routes through an IPv6-based network. RIPng is a distance vector protocol, as described in [1]. RIPng should be implemented only in routers; IPv6 provides other mechanisms for router discovery [10]. Any router that uses RIPng is assumed to have interfaces to one or more networks, otherwise it isn't really a router. These are referred to as its directly-connected networks. The protocol relies on access to certain information about each of these networks, the most important of which is its metric. The RIPng metric of a network is an integer between 1 and 15, inclusive. It is set in some manner not specified in this protocol; however, given the maximum path limit of 15, a value of 1 is usually used. Implementations should allow the system administrator to set the metric of each network. In addition to the metric, each network will have an IPv6 destination address prefix and prefix length associated with it. These are to be set by the system administrator in a manner not specified in this protocol.

Each router that implements RIPng is assumed to have a routing table. This table has one entry for every destination that is reachable throughout the system operating RIPng. Each entry contains at least the following information:

- The IPv6 prefix of the destination.

- A metric, which represents the total cost of getting a datagram from the router to that destination. This metric is the sum of the costs associated with the networks that would be traversed to get to the destination.

- The IPv6 address of the next router along the path to the destination (i.e., the next hop). If the destination is on one of the directly-connected networks, this item is not needed.

- A flag to indicate that information about the route has changed recently. This will be referred to as the "route change flag."

- Various timers associated with the route. See section 2.3 for more details on timers.

The entries for the directly-connected networks are set up by the router using information gathered by means not specified in this protocol. The metric for a directly-connected network is set to the cost of that network. As mentioned, 1 is the usual cost. In that case, the RIPng metric reduces to a simple hop-count. More complex metrics may be used when it is desirable to show preference for some

networks over others (e.g., to indicate of differences in bandwidth or reliability).

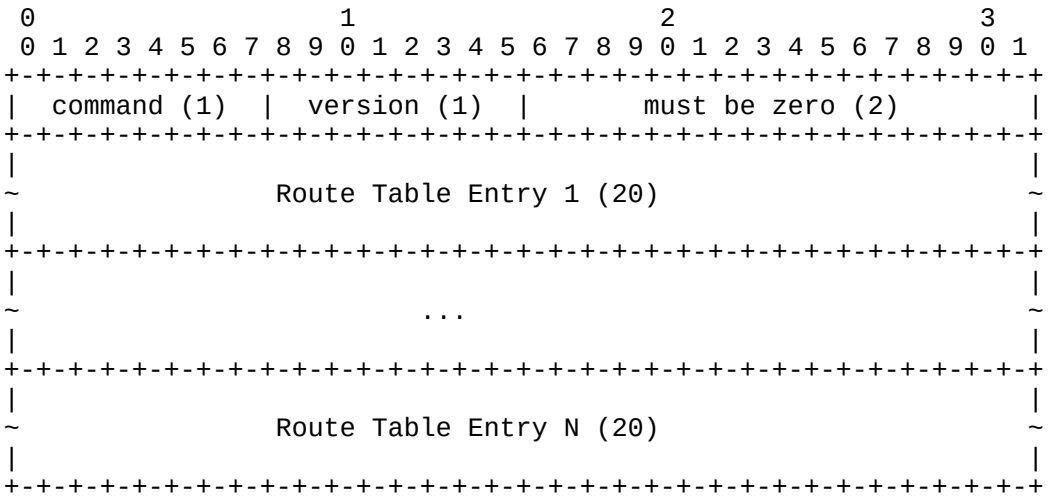
Implementors may also choose to allow the system administrator to enter additional routes. These would most likely be routes to hosts or networks outside the scope of the routing system. They are referred to as "static routes." Entries for destinations other than these initial ones are added and updated by the algorithms described in the following sections.

In order for the protocol to provide complete information on routing, every router in the AS must participate in the protocol. In cases where multiple IGPs are in use, there must be at least one router which can leak routing information between the protocols.

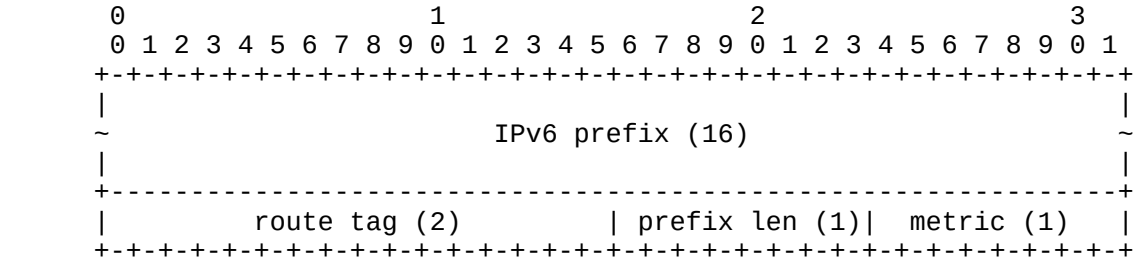
2.1 Message Format

RIPng is a UDP-based protocol. Each router that uses RIPng has a routing process that sends and receives datagrams on UDP port number 521, the RIPng port. All communications intended for another router's RIPng process are sent to the RIPng port. All routing update messages are sent from the RIPng port. Unsolicited routing update messages have both the source and destination port equal to the RIPng port. Those sent in response to a request are sent to the port from which the request came. Specific queries may be sent from ports other than the RIPng port, but they must be directed to the RIPng port on the target machine.

The RIPng packet format is:



where each Route Table Entry (RTE) has the following format:



The maximum number of RTEs is defined below.

Field sizes are given in octets. Unless otherwise specified, fields contain binary integers, in network byte order, with the most-significant octet first (big-endian). Each tick mark represents one bit.

Every message contains a RIPng header which consists of a command and a version number. This document describes version 1 of the protocol (see section 2.4). The command field is used to specify the purpose of this message. The commands implemented in version 1 are:

1 - request A request for the responding system to send all or part of its routing table.

2 - response A message containing all or part of the sender's routing table. This message may be sent in response to a request, or it may be an unsolicited routing update generated by the sender.

For each of these message types, the remainder of the datagram contains a list of RTEs. Each RTE in this list contains a destination prefix, the number of significant bits in the prefix, and the cost to reach that destination (metric).

The destination prefix is the usual 128-bit, IPv6 address prefix stored as 16 octets in network byte order.

The route tag field is an attribute assigned to a route which must be preserved and readvertised with a route. The intended use of the route tag is to provide a method of separating "internal" RIPng routes (routes for networks within the RIPng routing domain) from "external" RIPng routes, which may have been imported from an EGP or another IGP.

Routers supporting protocols other than RIPng should be configurable to allow the route tag to be configured for routes imported from different sources. For example, routes imported from an EGP should be able to have their route tag either set to an arbitrary value, or at least to the number of the Autonomous System from which the routes were learned.

Other uses of the route tag are valid, as long as all routers in the RIPng domain use it consistently.

The prefix length field is the length in bits of the significant part of the prefix (a value between 0 and 128 inclusive) starting from the left of the prefix.

The metric field contains a value between 1 and 15 inclusive, specifying the current metric for the destination; or the value 16 (infinity), which indicates that the destination is not reachable.

The maximum datagram size is limited by the MTU of the medium over which the protocol is being used. Since an unsolicited RIPng update is never propagated across a router, there is no danger of an MTU mismatch. The determination of the number of RTEs which may be put into a given message is a function of the medium's MTU, the number of octets of header information preceeding the RIPng message, the size of the RIPng header, and the size of an RTE. The formula is:

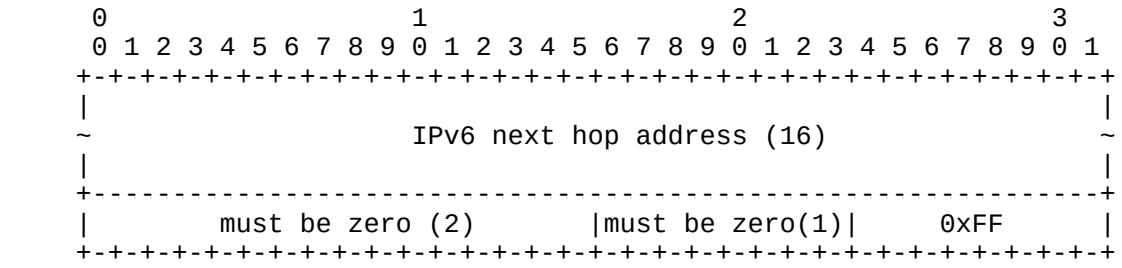
```
                +-                                                  -+
                | MTU - sizeof(IPv6_hdrs) - UDP_hdrlen - RIPng_hdrlen |
#RTEs = INT     | ---------------------------------------------------- |
                |                      RTE_size                       |
                +-                                                  -+
```

2.1.1 Next Hop

RIPng provides the ability to specify the immediate next hop IPv6 address to which packets to a destination specified by a route table entry (RTE) should be forwarded in much the same way as RIP-2 [2]. In RIP-2, each route table entry has a next hop field. Including a next hop field for each RTE in RIPng would nearly double the size of the RTE. Therefore, in RIPng, the next hop is specified by a special RTE and applies to all of the address RTEs following the next hop RTE until the end of the message or until another next hop RTE is encountered.

A next hop RTE is identified by a value of 0xFF in the metric field of an RTE. The prefix field specifies the IPv6 address of the next hop. The route tag and prefix length in the next hop RTE must be set to zero on sending and ignored on receiption.

The next hop Route Table Entry (RTE) has the following format:



Specifying a value of 0:0:0:0:0:0:0:0 in the prefix field of a next hop RTE indicates that the next hop address should be the originator of the RIPng advertisement. An address specified as a next hop must be a link-local address.

The purpose of the next hop RTE is to eliminate packets being routed through extra hops in the system. It is particularly useful when RIPng is not being run on all of the routers on a network. Note that next hop RTE is "advisory". That is, if the provided information is ignored, a possibly sub-optimal, but absolutely valid, route may be taken. If the received next hop address is not a link-local address, it should be treated as 0:0:0:0:0:0:0:0.

2.2 Addressing Considerations

The distinction between network, subnet and host routes does not need to be made for RIPng because an IPv6 address prefix is unambiguous.

Any prefix with a prefix length of zero is used to designate a default route. It is suggested that the prefix 0:0:0:0:0:0:0:0 be used when specifying the default route, though the prefix is essentially ignored. A default route is used when it is not convenient to list every possible network in the RIPng updates, and when one or more routers in the system are prepared to handle traffic to the networks that are not explicitly listed. These "default routers" use the default route as a path for all datagrams for which they have no explicit route. The decision as to how a router becomes a default router (i.e., how a default route entry is created) is left to the implementor. In general, the system administrator will be provided with a way to specify which routers should create and advertise default route entries. If this mechanism is used, the implementation should allow the network administrator to choose the metric associated with the default route advertisement. This will make it possible to establish a precedence amoung multiple default routers. The default route entries are handled by RIPng in exactly the same manner as any other destination prefix. System

administrators should take care to make sure that default routes do not propagate further than is intended. Generally, each AS has its own preferred default router. Therefore, default routes should generally not leave the boundary of an AS. The mechanisms for enforcing this restriction are not specified in this document.

2.3 Timers

This section describes all events that are triggered by timers.

Every 30 seconds, the RIPng process is awakened to send an unsolicited Response message, containing the complete routing table (see section 2.6 on Split Horizon), to every neighboring router. When there are many routers on a single network, there is a tendency for them to synchronize with each other such that they all issue updates at the same time. This can happen whenever the 30 second timer is affected by the processing load on the system. It is undesirable for the update messages to become synchronized, since it can lead to unnecessary collisions on broadcast networks (see [13] for more details). Therefore, implementations are required to take one of two precautions:

- The 30-second updates are triggered by a clock whose rate is not affected by system load or the time required to service the previous update timer.

- The 30-second timer is offset by a small random time (+/- 0 to 15 seconds) each time it is set. The offset is derived from: 0.5 * the update period (i.e. 30).

There are two timers associated with each route, a "timeout" and a "garbage-collection time." Upon expiration of the timeout, the route is no longer valid; however, it is retained in the routing table for a short time so that neighbors can be notified that the route has been dropped. Upon expiration of the garbage-collection timer, the route is finally removed from the routing table.

The timeout is initialized when a route is established, and any time an update message is received for the route. If 180 seconds elapse from the last time the timeout was initialized, the route is considered to have expired, and the deletion process described below begins for that route.

Deletions can occur for one of two reasons: the timeout expires, or the metric is set to 16 because of an update received from the current router (see section 2.4.2 for a discussion of processing updates from other routers). In either case, the following events happen:

- The garbage-collection timer is set for 120 seconds.

- The metric for the route is set to 16 (infinity). This causes the route to be removed from service.

- The route change flag is to indicate that this entry has been changed.

- The output process is signalled to trigger a response.

Until the garbage-collection timer expires, the route is included in all updates sent by this router. When the garbage-collection timer expires, the route is deleted from the routing table.

Should a new route to this network be established while the garbage-collection timer is running, the new route will replace the one that is about to be deleted. In this case the garbage-collection timer must be cleared.

Triggered updates also use a small timer; however, this is best described in section 2.5.1.

2.4 Input Processing

This section will describe the handling of datagrams received on the RIPng port. Processing will depend upon the value in the command field. Version 1 supports only two commands: Request and Response.

2.4.1 Request Messages

A Request is used to ask for a response containing all or part of a router's routing table. Normally, Requests are sent as multicasts, from the RIPng port, by routers which have just come up and are seeking to fill in their routing tables as quickly as possible. However, there may be situations (e.g., router monitoring) where the routing table of only a single router is needed. In this case, the Request should be sent directly to that router from a UDP port other than the RIPng port. If such a Request is received, the router responds directly to the requestor's address and port with a globally valid source address since the requestor may not reside on the directly attached network.

The Request is processed entry by entry. If there are no entries, no response is given. There is one special case. If there is exactly one entry in the request, and it has a destination prefix of zero, a prefix length of zero, and a metric of infinity (i.e., 16), then this is a request to send the entire routing table. In that case, a call is made to the output process to send the routing table to the requesting address/port. Except for this special case, processing is quite simple. Examine the list of RTEs in the Request one by one. For each entry, look up the destination in the router's routing database and, if there is a route, put that route's metric in the metric field of the RTE. If there is no explicit route to the specified destination, put infinity in the metric field. Once all the entries have been filled in, change the command from Request to Response and send the datagram back to the requestor.

Note that there is a difference in metric handling for specific and whole-table requests. If the request is for a complete routing table, normal output processing is done, including Split Horizon (see section 2.6 on Split Horizon). If the request is for specific entries, they are looked up in the routing table and the information is returned as is; no Split Horizon processing is done. The reason for this distinction is the expectation that these requests are likely to be used for different purposes. When a router first comes up, it multicasts a Request on every connected network asking for a complete routing table. It is assumed that these complete routing tables are to be used to update the requestor's routing table. For this reason, Split Horizon must be done. It is further assumed that a Request for specific networks is made only by diagnostic software, and is not used for routing. In this case, the requester would want to know the exact contents of the routing table and would not want any information hidden or modified.

2.4.2 Response Messages

A Response can be received for one of several different reasons:

- response to a specific query
- regular update (unsolicited response)
- triggered update caused by a route change

Processing is the same no matter why the Response was generated.

Because processing of a Response may update the router's routing table, the Response must be checked carefully for validity. The Response must be ignored if it is not from the RIPng port. The datagram's IPv6 source address should be checked to see whether the datagram is from a valid neighbor; the source of the datagram must be a link-local address. It is also worth checking to see whether the

response is from one of the router's own addresses. Interfaces on broadcast networks may receive copies of their own multicasts immediately. If a router processes its own output as new input, confusion is likely, and such datagrams must be ignored. As an additional check, periodic advertisements must have their hop counts set to 255, and inbound, multicast packets sent from the RIPng port (i.e. periodic advertisement or triggered update packets) must be examined to ensure that the hop count is 255. This absolutely guarantees that a packet is from a neighbor, because any intermediate node would have decremented the hop count. Queries and their responses may still cross intermediate nodes and therefore do not require the hop count test to be done.

Once the datagram as a whole has been validated, process the RTEs in the Response one by one. Again, start by doing validation. Incorrect metrics and other format errors usually indicate misbehaving neighbors and should probably be brought to the administrator's attention. For example, if the metric is greater than infinity, ignore the entry but log the event. The basic validation tests are:

- is the destination prefix valid (e.g., not a multicast prefix and not a link-local address) A link-local address should never be present in an RTE.
- is the prefix length valid (i.e., between 0 and 128, inclusive)
- is the metric valid (i.e., between 1 and 16, inclusive)

If any check fails, ignore that entry and proceed to the next. Again, logging the error is probably a good idea.

Once the entry has been validated, update the metric by adding the cost of the network on which the message arrived. If the result is greater than infinity, use infinity. That is,

```
metric = MIN (metric + cost, infinity)
```

Now, check to see whether there is already an explicit route for the destination prefix. If there is no such route, add this route to the routing table, unless the metric is infinity (there is no point in adding a route which unusable). Adding a route to the routing table consists of:

- Setting the destination prefix and length to those in the RTE.

- Setting the metric to the newly calculated metric (as described above).

- Set the next hop address to be the address of the router from which the datagram came or the next hop address specified by a next hop RTE.

- Initialize the timeout for the route. If the garbage-collection timer is running for this route, stop it (see section 2.3 for a discussion of the timers).

- Set the route change flag.

- Signal the output process to trigger an update (see section 2.5).

If there is an existing route, compare the next hop address to the address of the router from which the datagram came. If this datagram is from the same router as the existing route, reinitialize the timeout. Next, compare the metrics. If the datagram is from the same router as the existing route, and the new metric is different than the old one; or, if the new metric is lower than the old one; do the following actions:

- Adopt the route from the datagram. That is, put the new metric in, and adjust the next hop address (if necessary).

- Set the route change flag and signal the output process to trigger an update.

- If the new metric is infinity, start the deletion process (described above); otherwise, re-initialize the timeout.

If the new metric is infinity, the deletion process begins for the route, which is no longer used for routing packets. Note that the deletion process is started only when the metric is first set to infinity. If the metric was already infinity, then a new deletion process is not started.

If the new metric is the same as the old one, it is simplest to do nothing further (beyond reinitializing the timeout, as specified above); but, there is a heuristic which could be applied. Normally, it is senseless to replace a route if the new route has the same metric as the existing route; this would cause the route to bounce back and forth, which would generate an intolerable number of triggered updates. However, if the existing route is showing signs of timing out, it may be better to switch to an equally-good alternative route immediately, rather than waiting for the timeout to happen. Therefore, if the new metric is the same as the old one, examine the timeout for the existing route. If it is at least halfway to the expiration point, switch to the new route. This heuristic is optional, but highly recommended.

Any entry that fails these tests is ignored, as it is no better than the current route.

2.5 Output Processing

This section describes the processing used to create response messages that contain all or part of the routing table. This processing may be triggered in any of the following ways:

- By input processing, when a Request is received. In this case, the Response is sent to only one destination (i.e. the unicast address of the requestor).

- By the regular routing update. Every 30 seconds, a Response containing the whole routing table is sent to every neighboring router.

- By triggered updates. Whenever the metric for a route is changed, an update is triggered.

The special processing required for a Request is described in section 2.4.1.

When a Response is to be sent to all neighbors (i.e., a regular or triggered update), a Response message is multicast to the multicast group FF02::9, the all-rip-routers multicast group, on all connected networks that support broadcasting or are point-to-point links. RIPng handles point-to-point links just like multicast links as multicasting can be trivially provided on such links. Thus, one Response is prepared for each directly-connected network, and sent to the all-rip-routers multicast group. In most cases, this reaches all neighboring routers. However, there are some cases where this may not be good enough. This may involve a network that is not a broadcast network (e.g., the ARPANET), or a situation involving dumb routers. In such cases, it may be necessary to specify an actual list of neighboring routers and send a datagram to each one explicitly. It is left to the implementor to determine whether such a mechanism is needed, and to define how the list is specified.

2.5.1 Triggered Updates

Triggered updates require special handling for two reasons. First, experience shows that triggered updates can cause excessive loads on networks with limited capacity or networks with many routers on them. Therefore, the protocol requires that implementors include provisions to limit the frequency of triggered updates. After a triggered update is sent, a timer should be set for a random interval between 1 and 5 seconds. If other changes that would trigger updates occur

before the timer expires, a single update is triggered when the timer expires. The timer is then reset to another random value between 1 and 5 seconds. Triggered updates may be suppressed if a regular update is due by the time the triggered update would be sent.

Second, triggered updates do not need to include the entire routing table. In principle, only those routes which have changed need to be included. Therefore messages generated as part of a triggered update must include at least those routes that have their route change flag set. They may include additional routes, at the discretion of the implementor; however, sending complete routing updates is strongly discouraged. When a triggered update is processed, messages should be generated for every directly-connected network. Split Horizon processing is done when generating triggered updates as well as normal updates (see section 2.6). If, after Split Horizon processing for a given network, a changed route will appear unchanged on that network (e.g., it appears with an infinite metric), the route need not be sent. If no routes need be sent on that network, the update may be omitted. Once all of the triggered updates have been generated, the route change flags should be cleared.

If input processing is allowed while output is being generated, appropriate interlocking must be done. The route change flags should not be changed as a result of processing input while a triggered update message is being generated.

The only difference between a triggered update and other update messages is the possible omission of routes that have not changed. The remaining mechanisms, described in the next section, must be applied to all updates.

2.5.2 Generating Response Messages

This section describes how a Response message is generated for a particular directly-connected network:

The IPv6 source address must be a link-local address of the possible addresses of the sending router’s interface, except when replying to a unicast Request Message from a port other than the RIPng port. In the latter case, the source address must be a globaly valid address. In the former case, it is important to use a link-local address because the source address is put into routing tables (as the next hop) in the routers which receive this Response. If an incorrect source address is used, other routers may be unable to route datagrams. Sometimes routers are set up with multiple IPv6 addresses on a single physical interface. Normally, this means that several logical IPv6 networks are being carried over one physical medium. It is possible that a router may have multiple link-local addresses for

a single interface. In this case, the router must only originate a single Response message with a source address of the designated link-local address for a given interface. The choice of which link-local address to use should only change when the current choice is no longer valid. This is necessary because nodes receiving Response messages use the source address to identify the sender. If multiple packets from the same router contain different source addresses, nodes will assume they come from different routers, leading to undesirable behavior.

Set the version number to the current version of RIPng. The version described in this document is version 1. Set the command to Response. Set the bytes labeled "must be zero" to zero. Start filling in RTEs. Recall that the maximum datagram size is limited by the network's MTU. When there is no more space in the datagram, send the current Response and start a new one.

To fill in the RTEs, examine each route in the routing table. Routes to link-local addresses must never be included in an RTE. If a triggered update is being generated, only entries whose route change flags are set need be included. If, after Split Horizon processing, the route should not be included, skip it. If the route is to be included, then the destination prefix, prefix length, and metric are put into the RTE. The route tag is filled in as defined in section 2.1. Routes must be included in the datagram even if their metrics are infinite.

2.6 Split Horizon

Split Horizon is a algorithm for avoiding problems caused by including routes in updates sent to the gateway from which they were learned. The basic split horizon algorithm omits routes learned from one neighbor in updates sent to that neighbor. In the case of a broadcast network, all routes learned from any neighbor on that network are omitted from updates sent on that network.

Split Horizon with Poisoned Reverse (more simply, Poison Reverse) does include such routes in updates, but sets their metrics to infinity. In effect, advertising the fact that there routes are not reachable. This is the preferred method of operation; however, implementations should provide a per-interface control allowing no horizoning, split horizoning, and poisoned reverse to be selected.

For a theoretical discussion of Split Horizon and Poison Reverse, and why they are needed, see section 2.1.1 of [1].

3. Control Functions

This section describes administrative controls. These are not part of the protocol per se; however, experience with existing networks suggests that they are important. Because they are not a necessary part of the protocol, they are considered optional. However, it is strongly recommend that at least some of them be included in every implementation. These controls are intended primarily to allow RIPng to be connected to networks whose routing may be unstable or subject to errors. Here are some examples:

- It is sometimes desirable to restrict the routers from which updates will be accepted, or to which updates will be sent. This is usually done for administrative, routing policy reasons.

- A number of sites limit the set of networks that they allow in Response messages. Organization A may have a connection to organization B that they use for direct communication. For security or performance reasons A may not be willing to give other organizations access to that connection. In such a case, A should not include B's networks in updates that A sends to third parties.

Here are some typical controls. Note, however, that the RIPng protocol does not require these or any other controls.

- A neighbor list which allows the network administrator to be able to define a list of neighbors for each router. A router would accept response messages only from routers on its list of neighbors. A similar list for target routers should also be available to the administrator. By default, no restrictions are defined.

- A filter for specific destinations would permit the network administrator to be able to specify a list of destination prefixes to allow or disallow. The list would be associated with a particular interface in the incoming and/or outgoing directions. Only allowed networks would be mentioned in Response messages going out or processed in Response messages coming in. If a list of allowed prefixes is specified, all other prefixes are disallowed. If a list of disallowed prefixes is specified, all other prefixes are allowed. By default, no filters are applied.

4. Security Considerations

Since RIPng runs over IPv6, RIPng relies on the IP Authentication Header (see [11]) and the IP Encapsulating Security Payload (see [12]) to ensure integrity and authentication/confidentiality of routing exchanges.

References

[1] Hedrick, C., "Routing Information Protocol", RFC 1058, Rutgers University, June 1988.

[2] Malkin, G., "RIP Version 2 - Carrying Additional Information", RFC 1723, Xylogics, Inc., November, 1994.

[3] Hinden, R., "IP Next Generation Overview", Work in Progress.

[4] Bellman, R., "Dynamic Programming", Princeton University Press, Princeton, N.J., 1957.

[5] Bertsekas, D. P., and Gallaher, R. G., "Data Networks", Prentice-Hall, Englewood Cliffs, N.J., 1987.

[6] Braden, R., and J. Postel, "Requirements for Internet Gateways", USC/Information Sciences Institute, STD 4, RFC 1009, June 1987.

[7] Boggs, D. R., Shoch, J. F., Taft, E. A., and Metcalfe, R. M., "Pup: An Internetwork Architecture", IEEE Transactions on Communications, April 1980.

[8] Ford, L. R. Jr., and Fulkerson, D. R., "Flows in Networks", Princeton University Press, Princeton, N.J., 1962.

[9] Xerox Corp., "Internet Transport Protocols", Xerox System Integration Standard XSIS 028112, December 1981.

[10] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 1970, August 1996.

[11] Atkinson, R., "IP Authentication Header", RFC 1826 Naval Research Laboratory, August 1995.

[12] Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC 1827, Naval Research Laboratory, August 1995.

[13] Floyd, S., and Jacobson, V., "The Synchronization of Periodic Routing Messages", Proceedings of ACM SIGCOMM '93, September 1993.

Authors' Addresses

Gary Scott Malkin
Xylogics, Inc.
53 Third Avenue
Burlington, MA 01803

Phone: (617) 272-8140
EMail: gmalkin@Xylogics.COM

Robert E. Minnear
Ipsilon Networks, Inc.
2191 E. Bayshore Road, Suite 100
Palo Alto, CA 94303

Phone: (415) 846-4614
EMail: minnear@ipsilon.com