                               RIP Version 2

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 an extension of the Routing Information
   Protocol (RIP), as defined in [1], to expand the amount of useful
   information carried in RIP messages and to add a measure of security.

   A companion document will define the SNMP MIB objects for RIP-2 [2].
   An additional document will define cryptographic security
   improvements for RIP-2 [3].

Table of Contents

1. Justification

   With the advent of OSPF and IS-IS, there are those who believe that
   RIP is obsolete. While it is true that the newer IGP routing
   protocols are far superior to RIP, RIP does have some advantages.
   Primarily, in a small network, RIP has very little overhead in terms
   of bandwidth used and configuration and management time. RIP is also
   very easy to implement, especially in relation to the newer IGPs.

   Additionally, there are many, many more RIP implementations in the
   field than OSPF and IS-IS combined. It is likely to remain that way
   for some years yet.

   Given that RIP will be useful in many environments for some period of
   time, it is reasonable to increase RIP's usefulness. This is
   especially true since the gain is far greater than the expense of the
   change.

2. Current RIP

   The current RIP-1 message contains the minimal amount of information
   necessary for routers to route messages through a network. It also
   contains a large amount of unused space, owing to its origins.

   The current RIP-1 protocol does not consider autonomous systems and
   IGP/EGP interactions, subnetting [11], and authentication since
   implementations of these postdate RIP-1. The lack of subnet masks is
   a particularly serious problem for routers since they need a subnet
   mask to know how to determine a route. If a RIP-1 route is a network
   route (all non-network bits 0), the subnet mask equals the network
   mask. However, if some of the non-network bits are set, the router
   cannot determine the subnet mask. Worse still, the router cannot
   determine if the RIP-1 route is a subnet route or a host route.
   Currently, some routers simply choose the subnet mask of the
   interface over which the route was learned and determine the route
   type from that.

3. Basic Protocol

3.1 Introduction

   RIP is a routing protocol 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. RIP 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-1 is expected to fit, see Braden and Postel [6].

RIP uses one of a class of routing 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.

RIP is intended for use within the IP-based Internet. The Internet
is organized into a number of networks connected by special purpose
gateways known as routers. The networks may be either point-to-point
links or more complex networks such as Ethernet or token ring. Hosts
and routers are presented with IP datagrams addressed to some host.
Routing is the method by which the host or router decides where to
send the datagram. It may be able to send the datagram directly to
the destination, if that destination is on one of the networks that
are directly connected to the host or router. However, the
interesting case is when the destination is not directly reachable.

   In this case, the host or router attempts to send the datagram to a
   router that is nearer the destination. The goal of a routing
   protocol is very simple: It is to supply the information that is
   needed to do routing.

3.2 Limitations of the Protocol

   This protocol does not solve every possible routing problem. As
   mentioned above, it is primary 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 RIP 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. (This will be explained in the next section.)
     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.

3.3. Organization of this document

   The main body of this document is organized into two parts, which
   occupy the next two sections:

      A conceptual development and justification of distance vector
      algorithms in general.

      The actual protocol description.

   Each of these two sections can largely stand on its own. Section 3.4
   attempts to give an informal presentation of the mathematical
   underpinnings of the algorithm. Note that the presentation follows a
   "spiral" method. An initial, fairly simple algorithm is described.
   Then refinements are added to it in successive sections. Section 3.5
   is the actual protocol description. Except where specific references
   are made to section 3.4, it should be possible to implement RIP
   entirely from the specifications given in section 3.5.

3.4 Distance Vector Algorithms

   Routing is the task of finding a path from a sender to a desired
   destination. In the IP "Internet model" this reduces primarily to a
   matter of finding a series of routers between the source and
   destination networks. As long as a message or datagram remains on a
   single network or subnet, any forwarding problems are the
   responsibility of technology that is specific to the network. For
   example, Ethernet and the ARPANET each define a way in which any
   sender can talk to any specified destination within that one network.
   IP routing comes in primarily when messages must go from a sender on
   one network to a destination on a different one. In that case, the
   message must pass through one or more routers connecting the
   networks. If the networks are not adjacent, the message may pass
   through several intervening networks, and the routers connecting
   them. Once the message gets to a router that is on the same network
   as the destination, that network's own technology is used to get to
   the destination.

   Throughout this section, the term "network" is used generically to
   cover a single broadcast network (e.g., an Ethernet), a point to
   point line, or the ARPANET. The critical point is that a network is
   treated as a single entity by IP. Either no forwarding decision is
   necessary (as with a point to point line), or that forwarding is done
   in a manner that is transparent to IP, allowing IP to treat the
   entire network as a single fully-connected system (as with an
   Ethernet or the ARPANET). Note that the term "network" is used in a
   somewhat different way in discussions of IP addressing. We are using
   the term "network" here to refer to subnets in cases where subnet

addressing is in use.

A number of different approaches for finding routes between networks
are possible. One useful way of categorizing these approaches is on
the basis of the type of information the routers need to exchange in
order to be able to find routes. Distance vector algorithms are
based on the exchange of only a small amount of information. Each
entity (router or host) that participates in the routing protocol is
assumed to keep information about all of the destinations within the
system. Generally, information about all entities connected to one
network is summarized by a single entry, which describes the route to
all destinations on that network. This summarization is possible
because as far as IP is concerned, routing within a network is
invisible. Each entry in this routing database includes the next
router to which datagrams destined for the entity should be sent. In
addition, it includes a "metric" measuring the total distance to the
entity. Distance is a somewhat generalized concept, which may cover
the time delay in getting messages to the entity, the dollar cost of
sending messages to it, etc. Distance vector algorithms get their
name from the fact that it is possible to compute optimal routes when
the only information exchanged is the list of these distances.
Furthermore, information is only exchanged among entities that are
adjacent, that is, entities that share a common network.

Although routing is most commonly based on information about
networks, it is sometimes necessary to keep track of the routes to
individual hosts. The RIP protocol makes no formal distinction
between networks and hosts. It simply describes exchange of
information about destinations, which may be either networks or
hosts. (Note however, that it is possible for an implementor to
choose not to support host routes. See section 3.2.) In fact, the
mathematical developments are most conveniently thought of in terms
of routes from one host or router to another. When discussing the
algorithm in abstract terms, it is best to think of a routing entry
for a network as an abbreviation for routing entries for all of the
entities connected to that network. This sort of abbreviation makes
sense only because we think of networks as having no internal
structure that is visible at the IP level. Thus, we will generally
assign the same distance to every entity in a given network.

We said above that each entity keeps a routing database with one
entry for every possible destination in the system. An actual
implementation is likely to need to keep the following information
about each destination:

- address: in IP implementations of these algorithms, this will be
  the IP address of the host or network.

- router: the first router along the route to the destination.

- interface: the physical network which must be used to reach the
  first router.

- metric: a number, indicating the distance to the destination.

- timer: the amount of time since the entry was last updated.

In addition, various flags and other internal information will
probably be included. This database is initialized with a
description of the entities that are directly connected to the
system. It is updated according to information received in messages
from neighboring routers.

The most important information exchanged by the hosts and routers is
carried in update messages. Each entity that participates in the
routing scheme sends update messages that describe the routing
database as it currently exists in that entity. It is possible to
maintain optimal routes for the entire system by using only
information obtained from neighboring entities. The algorithm used
for that will be described in the next section.

As we mentioned above, the purpose of routing is to find a way to get
datagrams to their ultimate destinations. Distance vector algorithms
are based on a table in each router listing the best route to every
destination in the system. Of course, in order to define which route
is best, we have to have some way of measuring goodness. This is
referred to as the "metric".

In simple networks, it is common to use a metric that simply counts
how many routers a message must go through. In more complex
networks, a metric is chosen to represent the total amount of delay
that the message suffers, the cost of sending it, or some other
quantity which may be minimized. The main requirement is that it
must be possible to represent the metric as a sum of "costs" for
individual hops.

Formally, if it is possible to get from entity i to entity j directly
(i.e., without passing through another router between), then a cost,
$d(i,j)$, is associated with the hop between i and j. In the normal
case where all entities on a given network are considered to be the
same, $d(i,j)$ is the same for all destinations on a given network, and
represents the cost of using that network. To get the metric of a
complete route, one just adds up the costs of the individual hops

that make up the route. For the purposes of this memo, we assume
that the costs are positive integers.

Let D(i,j) represent the metric of the best route from entity i to
entity j. It should be defined for every pair of entities. d(i,j)
represents the costs of the individual steps. Formally, let d(i,j)
represent the cost of going directly from entity i to entity j. It
is infinite if i and j are not immediate neighbors. (Note that d(i,i)
is infinite. That is, we don't consider there to be a direct
connection from a node to itself.) Since costs are additive, it is
easy to show that the best metric must be described by

```
   D(i,i) = 0,                          all i
   D(i,j) = min [d(i,k) + D(k,j)],   otherwise
                 k
```

and that the best routes start by going from i to those neighbors k
for which d(i,k) + D(k,j) has the minimum value. (These things can
be shown by induction on the number of steps in the routes.) Note
that we can limit the second equation to k's that are immediate
neighbors of i. For the others, d(i,k) is infinite, so the term
involving them can never be the minimum.

It turns out that one can compute the metric by a simple algorithm
based on this. Entity i gets its neighbors k to send it their
estimates of their distances to the destination j. When i gets the
estimates from k, it adds d(i,k) to each of the numbers. This is
simply the cost of traversing the network between i and k. Now and
then i compares the values from all of its neighbors and picks the
smallest.

A proof is given in [2] that this algorithm will converge to the
correct estimates of D(i,j) in finite time in the absence of topology
changes. The authors make very few assumptions about the order in
which the entities send each other their information, or when the min
is recomputed. Basically, entities just can't stop sending updates
or recomputing metrics, and the networks can't delay messages
forever. (Crash of a routing entity is a topology change.) Also,
their proof does not make any assumptions about the initial estimates
of D(i,j), except that they must be non-negative. The fact that
these fairly weak assumptions are good enough is important. Because
we don't have to make assumptions about when updates are sent, it is
safe to run the algorithm asynchronously. That is, each entity can
send updates according to its own clock. Updates can be dropped by
the network, as long as they don't all get dropped. Because we don't
have to make assumptions about the starting condition, the algorithm
can handle changes. When the system changes, the routing algorithm
starts moving to a new equilibrium, using the old one as its starting
point. It is important that the algorithm will converge in finite

time no matter what the starting point. Otherwise certain kinds of
changes might lead to non-convergent behavior.

The statement of the algorithm given above (and the proof) assumes
that each entity keeps copies of the estimates that come from each of
its neighbors, and now and then does a min over all of the neighbors.
In fact real implementations don't necessarily do that. They simply
remember the best metric seen so far, and the identity of the
neighbor that sent it. They replace this information whenever they
see a better (smaller) metric. This allows them to compute the
minimum incrementally, without having to store data from all of the
neighbors.

There is one other difference between the algorithm as described in
texts and those used in real protocols such as RIP: the description
above would have each entity include an entry for itself, showing a
distance of zero. In fact this is not generally done. Recall that
all entities on a network are normally summarized by a single entry
for the network. Consider the situation of a host or router G that
is connected to network A. C represents the cost of using network A
(usually a metric of one). (Recall that we are assuming that the
internal structure of a network is not visible to IP, and thus the
cost of going between any two entities on it is the same.) In
principle, G should get a message from every other entity H on
network A, showing a cost of 0 to get from that entity to itself. G
would then compute C + 0 as the distance to H. Rather than having G
look at all of these identical messages, it simply starts out by
making an entry for network A in its table, and assigning it a metric
of C. This entry for network A should be thought of as summarizing
the entries for all other entities on network A. The only entity on
A that can't be summarized by that common entry is G itself, since
the cost of going from G to G is 0, not C. But since we never need
those 0 entries, we can safely get along with just the single entry
for network A. Note one other implication of this strategy: because
we don't need to use the 0 entries for anything, hosts that do not
function as routers don't need to send any update messages. Clearly
hosts that don't function as routers (i.e., hosts that are connected
to only one network) can have no useful information to contribute
other than their own entry $D(i,i) = 0$. As they have only the one
interface, it is easy to see that a route to any other network
through them will simply go in that interface and then come right
back out it. Thus the cost of such a route will be greater than the
best cost by at least C. Since we don't need the 0 entries, non-
routers need not participate in the routing protocol at all.

Let us summarize what a host or router G does. For each destination
in the system, G will keep a current estimate of the metric for that
destination (i.e., the total cost of getting to it) and the identity

of the neighboring router on whose data that metric is based. If the
destination is on a network that is directly connected to G, then G
simply uses an entry that shows the cost of using the network, and
the fact that no router is needed to get to the destination. It is
easy to show that once the computation has converged to the correct
metrics, the neighbor that is recorded by this technique is in fact
the first router on the path to the destination. (If there are
several equally good paths, it is the first router on one of them.)
This combination of destination, metric, and router is typically
referred to as a route to the destination with that metric, using
that router.

4.ne The method so far only has a way to lower the metric, as the
existing metric is kept until a smaller one shows up. It is possible
that the initial estimate might be too low. Thus, there must be a
way to increase the metric. It turns out to be sufficient to use the
following rule: suppose the current route to a destination has metric
D and uses router G. If a new set of information arrived from some
source other than G, only update the route if the new metric is
better than D. But if a new set of information arrives from G
itself, always update D to the new value. It is easy to show that
with this rule, the incremental update process produces the same
routes as a calculation that remembers the latest information from
all the neighbors and does an explicit minimum. (Note that the
discussion so far assumes that the network configuration is static.
It does not allow for the possibility that a system might fail.)

To summarize, here is the basic distance vector algorithm as it has
been developed so far. (Note that this is not a statement of the RIP
protocol. There are several refinements still to be added.) The
following procedure is carried out by every entity that participates
in the routing protocol. This must include all of the routers in the
system. Hosts that are not routers may participate as well.

- Keep a table with an entry for every possible destination in the
  system. The entry contains the distance D to the destination, and
  the first router G on the route to that network. Conceptually,
  there should be an entry for the entity itself, with metric 0, but
  this is not actually included.

- Periodically, send a routing update to every neighbor. The update
  is a set of messages that contain all of the information from the
  routing table. It contains an entry for each destination, with the
  distance shown to that destination.

- When a routing update arrives from a neighbor G', add the cost
  associated with the network that is shared with G'. (This should
  be the network over which the update arrived.) Call the resulting

distance D'. Compare the resulting distances with the current
routing table entries. If the new distance D' for N is smaller
than the existing value D, adopt the new route. That is, change
the table entry for N to have metric D' and router G'. If G' is
the router from which the existing route came, i.e., G' = G, then
use the new metric even if it is larger than the old one.

3.4.1 Dealing with changes in topology

   The discussion above assumes that the topology of the network is
   fixed. In practice, routers and lines often fail and come back up.
   To handle this possibility, we need to modify the algorithm slightly.

   The theoretical version of the algorithm involved a minimum over all
   immediate neighbors. If the topology changes, the set of neighbors
   changes. Therefore, the next time the calculation is done, the
   change will be reflected. However, as mentioned above, actual
   implementations use an incremental version of the minimization. Only
   the best route to any given destination is remembered. If the router
   involved in that route should crash, or the network connection to it
   break, the calculation might never reflect the change. The algorithm
   as shown so far depends upon a router notifying its neighbors if its
   metrics change. If the router crashes, then it has no way of
   notifying neighbors of a change.

   In order to handle problems of this kind, distance vector protocols
   must make some provision for timing out routes. The details depend
   upon the specific protocol. As an example, in RIP every router that
   participates in routing sends an update message to all its neighbors
   once every 30 seconds. Suppose the current route for network N uses
   router G. If we don't hear from G for 180 seconds, we can assume
   that either the router has crashed or the network connecting us to it
   has become unusable. Thus, we mark the route as invalid. When we
   hear from another neighbor that has a valid route to N, the valid
   route will replace the invalid one. Note that we wait for 180
   seconds before timing out a route even though we expect to hear from
   each neighbor every 30 seconds. Unfortunately, messages are
   occasionally lost by networks. Thus, it is probably not a good idea
   to invalidate a route based on a single missed message.

   As we will see below, it is useful to have a way to notify neighbors
   that there currently isn't a valid route to some network. RIP, along
   with several other protocols of this class, does this through a
   normal update message, by marking that network as unreachable. A
   specific metric value is chosen to indicate an unreachable
   destination; that metric value is larger than the largest valid
   metric that we expect to see. In the existing implementation of RIP,
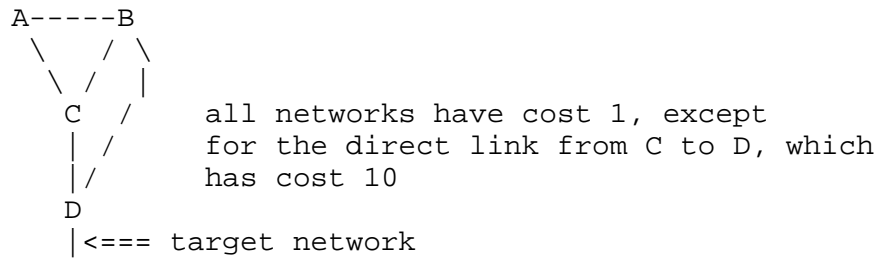   16 is used. This value is normally referred to as "infinity", since

it is larger than the largest valid metric. 16 may look like a
surprisingly small number. It is chosen to be this small for reasons
that we will see shortly. In most implementations, the same
convention is used internally to flag a route as invalid.

3.4.2 Preventing instability

The algorithm as presented up to this point will always allow a host
or router to calculate a correct routing table. However, that is
still not quite enough to make it useful in practice. The proofs
referred to above only show that the routing tables will converge to
the correct values in finite time. They do not guarantee that this
time will be small enough to be useful, nor do they say what will
happen to the metrics for networks that become inaccessible.

It is easy enough to extend the mathematics to handle routes becoming
inaccessible. The convention suggested above will do that. We
choose a large metric value to represent "infinity". This value must
be large enough that no real metric would ever get that large. For
the purposes of this example, we will use the value 16. Suppose a
network becomes inaccessible. All of the immediately neighboring
routers time out and set the metric for that network to 16. For
purposes of analysis, we can assume that all the neighboring routers
have gotten a new piece of hardware that connects them directly to
the vanished network, with a cost of 16. Since that is the only
connection to the vanished network, all the other routers in the
system will converge to new routes that go through one of those
routers. It is easy to see that once convergence has happened, all
the routers will have metrics of at least 16 for the vanished
network. Routers one hop away from the original neighbors would end
up with metrics of at least 17; routers two hops away would end up
with at least 18, etc. As these metrics are larger than the maximum
metric value, they are all set to 16. It is obvious that the system
will now converge to a metric of 16 for the vanished network at all
routers.

Unfortunately, the question of how long convergence will take is not
amenable to quite so simple an answer. Before going any further, it
will be useful to look at an example (taken from [2]). Note that
what we are about to show will not happen with a correct
implementation of RIP. We are trying to show why certain features
are needed. In the following example the letters correspond to
routers, and the lines to networks.

```
    A-----B
     \   / \
      \ /   |
       C   /     all networks have cost 1, except
       | /        for the direct link from C to D, which
       |/         has cost 10
       D
       |<=== target network
```

Each router will have a table showing a route to each network.

However, for purposes of this illustration, we show only the routes
from each router to the network marked at the bottom of the diagram.

```
        D:  directly connected, metric 1
        B:  route via D, metric 2
        C:  route via B, metric 3
        A:  route via B, metric 3
```

Now suppose that the link from B to D fails. The routes should now
adjust to use the link from C to D. Unfortunately, it will take a
while for this to this to happen. The routing changes start when B
notices that the route to D is no longer usable. For simplicity, the
chart below assumes that all routers send updates at the same time.
The chart shows the metric for the target network, as it appears in
the routing table at each router.

```
        time ------>

        D: dir, 1    dir, 1    dir, 1    dir, 1  ...  dir, 1    dir, 1
        B: unreach   C,   4    C,   5    C,   6       C,   11   C,   12
        C: B,   3    A,   4    A,   5    A,   6        A,   11   D,   11
        A: B,   3    C,   4    C,   5    C,   6        C,   11   C,   12

        dir = directly connected
        unreach = unreachable
```

Here's the problem: B is able to get rid of its failed route using a
timeout mechanism, but vestiges of that route persist in the system
for a long time. Initially, A and C still think they can get to D
via B. So, they keep sending updates listing metrics of 3. In the
next iteration, B will then claim that it can get to D via either A
or C. Of course, it can't. The routes being claimed by A and C are
now gone, but they have no way of knowing that yet. And even when
they discover that their routes via B have gone away, they each think
there is a route available via the other. Eventually the system
converges, as all the mathematics claims it must. But it can take
some time to do so. The worst case is when a network becomes

completely inaccessible from some part of the system. In that case,
the metrics may increase slowly in a pattern like the one above until
they finally reach infinity. For this reason, the problem is called
"counting to infinity".

You should now see why "infinity" is chosen to be as small as
possible. If a network becomes completely inaccessible, we want
counting to infinity to be stopped as soon as possible. Infinity
must be large enough that no real route is that big. But it
shouldn't be any bigger than required. Thus the choice of infinity
is a tradeoff between network size and speed of convergence in case
counting to infinity happens. The designers of RIP believed that the
protocol was unlikely to be practical for networks with a diameter
larger than 15.

There are several things that can be done to prevent problems like
this. The ones used by RIP are called "split horizon with poisoned
reverse", and "triggered updates".

3.4.3 Split horizon

Note that some of the problem above is caused by the fact that A and
C are engaged in a pattern of mutual deception. Each claims to be
able to get to D via the other. This can be prevented by being a bit
more careful about where information is sent. In particular, it is
never useful to claim reachability for a destination network to the
neighbor(s) from which the route was learned. "Split horizon" is a
scheme for avoiding problems caused by including routes in updates
sent to the router from which they were learned. The "simple split
horizon" scheme omits routes learned from one neighbor in updates
sent to that neighbor. "Split horizon with poisoned reverse"
includes such routes in updates, but sets their metrics to infinity.

If A thinks it can get to D via C, its messages to C should indicate
that D is unreachable. If the route through C is real, then C either
has a direct connection to D, or a connection through some other
router. C's route can't possibly go back to A, since that forms a
loop. By telling C that D is unreachable, A simply guards against
the possibility that C might get confused and believe that there is a
route through A. This is obvious for a point to point line. But
consider the possibility that A and C are connected by a broadcast
network such as an Ethernet, and there are other routers on that
network. If A has a route through C, it should indicate that D is
unreachable when talking to any other router on that network. The
other routers on the network can get to C themselves. They would
never need to get to C via A. If A's best route is really through C,
no other router on that network needs to know that A can reach D.
This is fortunate, because it means that the same update message that

is used for C can be used for all other routers on the same network.
Thus, update messages can be sent by broadcast.

In general, split horizon with poisoned reverse is safer than simple
split horizon. If two routers have routes pointing at each other,
advertising reverse routes with a metric of 16 will break the loop
immediately. If the reverse routes are simply not advertised, the
erroneous routes will have to be eliminated by waiting for a timeout.
However, poisoned reverse does have a disadvantage: it increases the
size of the routing messages. Consider the case of a campus backbone
connecting a number of different buildings. In each building, there
is a router connecting the backbone to a local network. Consider
what routing updates those routers should broadcast on the backbone
network. All that the rest of the network really needs to know about
each router is what local networks it is connected to. Using simple
split horizon, only those routes would appear in update messages sent
by the router to the backbone network. If split horizon with
poisoned reverse is used, the router must mention all routes that it
learns from the backbone, with metrics of 16. If the system is
large, this can result in a large update message, almost all of whose
entries indicate unreachable networks.

In a static sense, advertising reverse routes with a metric of 16
provides no additional information. If there are many routers on one
broadcast network, these extra entries can use significant bandwidth.
The reason they are there is to improve dynamic behavior. When
topology changes, mentioning routes that should not go through the
router as well as those that should can speed up convergence.
However, in some situations, network managers may prefer to accept
somewhat slower convergence in order to minimize routing overhead.
Thus implementors may at their option implement simple split horizon
rather than split horizon with poisoned reverse, or they may provide
a configuration option that allows the network manager to choose
which behavior to use. It is also permissible to implement hybrid
schemes that advertise some reverse routes with a metric of 16 and
omit others. An example of such a scheme would be to use a metric of
16 for reverse routes for a certain period of time after routing
changes involving them, and thereafter omitting them from updates.

The router requirements RFC [11] specifies that all implementation of
RIP must use split horizon and should also use split horizon with
poisoned reverse, although there may be a knob to disable poisoned
reverse.

3.4.4 Triggered updates

   Split horizon with poisoned reverse will prevent any routing loops
   that involve only two routers. However, it is still possible to end
   up with patterns in which three routers are engaged in mutual
   deception. For example, A may believe it has a route through B, B
   through C, and C through A. Split horizon cannot stop such a loop.
   This loop will only be resolved when the metric reaches infinity and
   the network involved is then declared unreachable. Triggered updates
   are an attempt to speed up this convergence. To get triggered
   updates, we simply add a rule that whenever a router changes the
   metric for a route, it is required to send update messages almost
   immediately, even if it is not yet time for one of the regular update
   message. (The timing details will differ from protocol to protocol.
   Some distance vector protocols, including RIP, specify a small time
   delay, in order to avoid having triggered updates generate excessive
   network traffic.) Note how this combines with the rules for
   computing new metrics. Suppose a router's route to destination N
   goes through router G. If an update arrives from G itself, the
   receiving router is required to believe the new information, whether
   the new metric is higher or lower than the old one. If the result is
   a change in metric, then the receiving router will send triggered
   updates to all the hosts and routers directly connected to it. They
   in turn may each send updates to their neighbors. The result is a
   cascade of triggered updates. It is easy to show which routers and
   hosts are involved in the cascade. Suppose a router G times out a
   route to destination N. G will send triggered updates to all of its
   neighbors. However, the only neighbors who will believe the new
   information are those whose routes for N go through G. The other
   routers and hosts will see this as information about a new route that
   is worse than the one they are already using, and ignore it. The
   neighbors whose routes go through G will update their metrics and
   send triggered updates to all of their neighbors. Again, only those
   neighbors whose routes go through them will pay attention. Thus, the
   triggered updates will propagate backwards along all paths leading to
   router G, updating the metrics to infinity. This propagation will
   stop as soon as it reaches a portion of the network whose route to
   destination N takes some other path.

   If the system could be made to sit still while the cascade of
   triggered updates happens, it would be possible to prove that
   counting to infinity will never happen. Bad routes would always be
   removed immediately, and so no routing loops could form.

   Unfortunately, things are not so nice. While the triggered updates
   are being sent, regular updates may be happening at the same time.
   Routers that haven't received the triggered update yet will still be
   sending out information based on the route that no longer exists. It

is possible that after the triggered update has gone through a
router, it might receive a normal update from one of these routers
that hasn't yet gotten the word. This could reestablish an orphaned
remnant of the faulty route. If triggered updates happen quickly
enough, this is very unlikely. However, counting to infinity is
still possible.

The router requirements RFC [11] specifies that all implementation of
RIP must implement triggered update for deleted routes and may
implement triggered updates for new routes or change of routes. RIP
implementations must also limit the rate which of triggered updates
may be trandmitted. (see section 3.10.1)

3.5 Protocol Specification

RIP is intended to allow routers to exchange information for
computing routes through an IPv4-based network. Any router that uses
RIP 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 RIP 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 IPv4 destination address and subnet
mask associated with it. These are to be set by the system
administrator in a manner not specified in this protocol.

Any host that uses RIP is assumed to have interfaces to one or more
networks. 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 is its metric or
"cost". The metric of a network is an integer between 1 and 15
inclusive. It is set in some manner not specified in this protocol.
Most existing implementations always use a metric of 1. New
implementations should allow the system administrator to set the cost
of each network. In addition to the cost, each network will have an
IPv4 network number and a subnet mask associated with it. These are
to be set by the system administrator in a manner not specified in
this protocol.

Note that the rules specified in section 3.7 assume that there is a
single subnet mask applying to each IPv4 network, and that only the
subnet masks for directly-connected networks are known. There may be
systems that use different subnet masks for different subnets within
a single network. There may also be instances where it is desirable

for a system to know the subnets masks of distant networks. Network-
wide distribution of routing information which contains different
subnet masks is permitted if all routers in the network are running
the extensions presented in this document. However, if all routers in
the network are not running these extensions distribution of routing
information containing different subnet masks must be limited to
avoid interoperability problems. See sections 3.7 and 4.3 for the
rules governing subnet distribution.

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

- The IPv4 address 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 IPv4 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 3.6 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 RIP 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).

To support the extensions detailed in this document, each entry must
additionally contain a subnet mask. The subnet mask allows the router
(along with the IPv4 address of the destination) to identify the
different subnets within a single network as well as the subnets
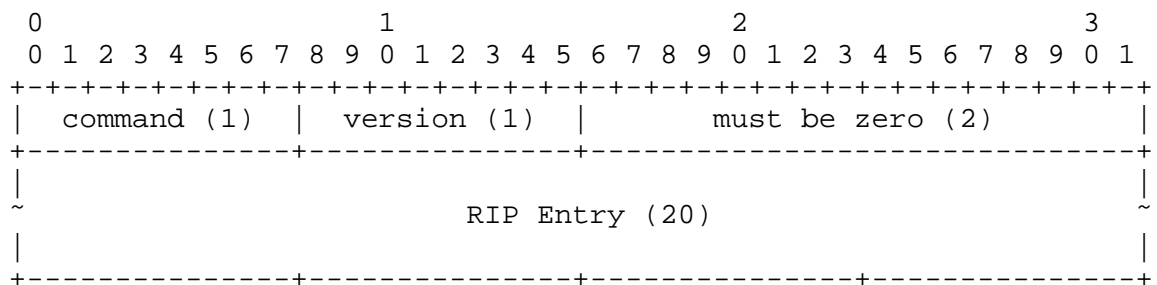masks of distant networks.

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.

3.6 Message Format

RIP is a UDP-based protocol. Each router that uses RIP has a routing
process that sends and receives datagrams on UDP port number 520, the
RIP-1/RIP-2 port. All communications intended for another routers's
RIP process are sent to the RIP port. All routing update messages
are sent from the RIP port. Unsolicited routing update messages have
both the source and destination port equal to the RIP port. Update
messages 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 RIP port, but they must be directed to the RIP port on
the target machine.

The RIP packet format is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  command (1)  |  version (1)  |       must be zero (2)        |
+---------------+---------------+-------------------------------+
|                                                               |
~                         RIP Entry (20)                        ~
|                                                               |
+---------------+---------------+---------------+---------------+
```

There may be between 1 and 25 (inclusive) RIP entries. A RIP-1 entry
has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| address family identifier (2) |       must be zero (2)        |
+-------------------------------+-------------------------------+
|                        IPv4 address (4)                       |
+---------------------------------------------------------------+
|                        must be zero (4)                       |
+---------------------------------------------------------------+
|                        must be zero (4)                       |
+---------------------------------------------------------------+
|                          metric (4)                           |
+---------------------------------------------------------------+
```

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 RIP header which consists of a command and a
version number. This section of the document describes version 1 of
the protocol; section 4 describes the version 2 extensions. The
command field is used to specify the purpose of this message. The
commands implemented in version 1 and 2 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, in version 1, the remainder of the
datagram contains a list of Route Entries (RTEs). Each RTE in this
list contains an Address Family Identifier (AFI), destination IPv4
address, and the cost to reach that destination (metric).

The AFI is the type of address. For RIP-1, only AF_INET (2) is
generally supported.

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

3.7 Addressing Considerations

   Distance vector routing can be used to describe routes to individual
   hosts or to networks. The RIP protocol allows either of these
   possibilities. The destinations appearing in request and response
   messages can be networks, hosts, or a special code used to indicate a
   default address. In general, the kinds of routes actually used will
   depend upon the routing strategy used for the particular network.
   Many networks are set up so that routing information for individual
   hosts is not needed. If every node on a given network or subnet is
   accessible through the same routers, then there is no reason to
   mention individual hosts in the routing tables. However, networks
   that include point-to-point lines sometimes require routers to keep
   track of routes to certain nodes. Whether this feature is required
   depends upon the addressing and routing approach used in the system.
   Thus, some implementations may choose not to support host routes. If
   host routes are not supported, they are to be dropped when they are
   received in response messages (see section 3.7.2).

   The RIP-1 packet format does not distinguish among various types of
   address. Fields that are labeled "address" can contain any of the
   following:

   host address subnet number network number zero (default route)

   Entities which use RIP-1 are assumed to use the most specific
   information available when routing a datagram. That is, when routing
   a datagram, its destination address must first be checked against the
   list of node addresses. Then it must be checked to see whether it
   matches any known subnet or network number. Finally, if none of
   these match, the default route is used.

   When a node evaluates information that it receives via RIP-1, its
   interpretation of an address depends upon whether it knows the subnet
   mask that applies to the net. If so, then it is possible to
   determine the meaning of the address. For example, consider net
   128.6. It has a subnet mask of 255.255.255.0. Thus 128.6.0.0 is a
   network number, 128.6.4.0 is a subnet number, and 128.6.4.1 is a node
   address. However, if the node does not know the subnet mask,
   evaluation of an address may be ambiguous. If there is a non-zero
   node part, there is no clear way to determine whether the address
   represents a subnet number or a node address. As a subnet number
   would be useless without the subnet mask, addresses are assumed to
   represent nodes in this situation. In order to avoid this sort of
   ambiguity, when using version 1, nodes must not send subnet routes to
   nodes that cannot be expected to know the appropriate subnet mask.
   Normally hosts only know the subnet masks for directly-connected
   networks. Therefore, unless special provisions have been made,

routes to a subnet must not be sent outside the network of which the
subnet is a part. RIP-2 (see section 4) eliminates the subnet/host
ambiguity by including the subnet mask in the routing entry.

This "subnet filtering" is carried out by the routers at the "border"
of the subnetted network. These are routers which connect that
network with some other network. Within the subnetted network, each
subnet is treated as an individual network. Routing entries for each
subnet are circulated by RIP. However, border routers send only a
single entry for the network as a whole to nodes in other networks.
This means that a border router will send different information to
different neighbors. For neighbors connected to the subnetted
network, it generates a list of all subnets to which it is directly
connected, using the subnet number. For neighbors connected to other
networks, it makes a single entry for the network as a whole, showing
the metric associated with that network. This metric would normally
be the smallest metric for the subnets to which the router is
attached.

Similarly, border routers must not mention host routes for nodes
within one of the directly-connected networks in messages to other
networks. Those routes will be subsumed by the single entry for the
network as a whole.

The router requirements RFC [11] specifies that all implementation of
RIP should support host routes but if they do not then they must
ignore any received host routes.

The special address 0.0.0.0 is used to describe a default route. A
default route is used when it is not convenient to list every
possible network in the RIP updates, and when one or more closely-
connected routers in the system are prepared to handle traffic to the
networks that are not listed explicitly. These routers should create
RIP entries for the address 0.0.0.0, just as if it were a network to
which they are connected. The decision as to how routers create
entries for 0.0.0.0 is left to the implementor. Most commonly, the
system administrator will be provided with a way to specify which
routers should create entries for 0.0.0.0; however, other mechanisms
are possible. For example, an implementor might decide that any
router which speaks BGP should be declared to be a default router.
It may be useful to allow the network administrator to choose the
metric to be used in these entries. If there is more than one
default router, this will make it possible to express a preference
for one over the other. The entries for 0.0.0.0 are handled by RIP
in exactly the same manner as if there were an actual network with
this address. System administrators should take care to make sure
that routes to 0.0.0.0 do not propagate further than is intended.
Generally, each autonomous system has its own preferred default

router. Thus, routes involving 0.0.0.0 should generally not leave
the boundary of an autonomous system. The mechanisms for enforcing
this are not specified in this document.

3.8 Timers

This section describes all events that are triggered by timers.

Every 30 seconds, the RIP process is awakened to send an unsolicited
Response message containing the complete routing table (see section
3.9 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. 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 5
  seconds) each time it is set. (Implementors may wish to consider
  even larger variation in the light of recent research results [10])

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 3.7.2 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 set 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 3.9.1.

3.9 Input Processing

    This section will describe the handling of datagrams received on the
    RIP port. Processing will depend upon the value in the command
    field.

    See sections 4.6 and 5.1 for details on handling version numbers.

3.9.1 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 broadcasts
    (multicasts for RIP-2), from the RIP 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 RIP port. If such a Request is received, the
    router responds directly to the requestor's address and port.

    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 an address family identifier 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 3.9 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.

3.9.2 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 RIP port. The
datagram's IPv4 source address should be checked to see whether the
datagram is from a valid neighbor; the source of the datagram must be
on a directly-connected network. 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
broadcasts/multicasts immediately. If a router processes its own
output as new input, confusion is likely so such datagrams must be
ignored.

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 address valid (e.g., unicast; not net 0 or 127)
- 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 address. 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 is unusable). Adding a route to the routing
table consists of:

- Setting the destination address to the destination address 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

- Initialize the timeout for the route. If the garbage-collection
  timer is running for this route, stop it (see section 3.6 for a
  discussion of the timers)

- Set the route change flag

- Signal the output process to trigger an update (see section 3.8.1)

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 (i.e., 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 re-initializing 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.

3.10 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 (this Response is
        unicast to the requestor; see section 3.7.1)

      - By the regular routing update (broadcast/multicast every 30
        seconds) router.

      - By triggered updates (broadcast/multicast when a route changes)

When a Response is to be sent to all neighbors (i.e., a regular or
triggered update), a Response message is directed to the router at
the far end of each connected point-to-point link, and is broadcast
(multicast for RIP-2) on all connected networks which support
broadcasting. Thus, one Response is prepared for each directly-
connected network, and sent to the appropriate address (direct or
broadcast/multicast). 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.

3.10.1 Triggered Updates

Triggered updates require special handling for two reasons. First,
experience shows that triggered updates can cause excessive load 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. A triggered update should 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 3.9). 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.

3.10.2 Generating Response Messages

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

Set the version number to either 1 or 2. The mechanism for deciding
which version to send is implementation specific; however, if this is
the Response to a Request, the Response version should match the
Request version. Set the command to Response. Set the bytes labeled
"must be zero" to zero. Start filling in RTEs. Recall that there is
a limit of 25 RTEs to a Response; if there are more, send the current
Response and start a new one. There is no defined limit to the
number of datagrams which make up a Response.

To fill in the RTEs, examine each route in the routing table. 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 address and metric are put into the
RTE. Routes must be included in the datagram even if their metrics
are infinite.

4. Protocol Extensions

   This section does not change the RIP protocol per se. Rather, it
   provides extensions to the message format which allows routers to
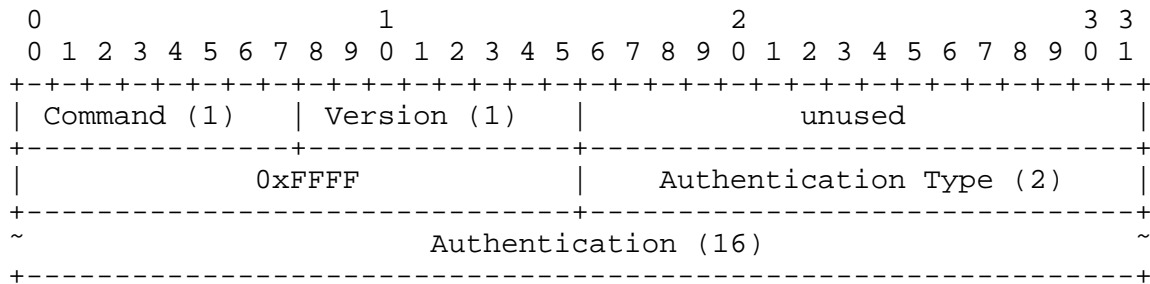   share important additional information.

   The same header format is used for RIP-1 and RIP-2 messages (see
   section 3.4). The format for the 20-octet route entry (RTE) for
   RIP-2 is:

```
 0                   1                   2                   3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Address Family Identifier (2) |        Route Tag (2)          |
+-------------------------------+-------------------------------+
|                         IP Address (4)                        |
+---------------------------------------------------------------+
|                        Subnet Mask (4)                        |
+---------------------------------------------------------------+
|                         Next Hop (4)                          |
+---------------------------------------------------------------+
|                          Metric (4)                           |
+---------------------------------------------------------------+
```

   The Address Family Identifier, IP Address, and Metric all have the
   meanings defined in section 3.4. The Version field will specify
   version number 2 for RIP messages which use authentication or carry
   information in any of the newly defined fields.

4.1 Authentication

   Since authentication is a per message function, and since there is
   only one 2-octet field available in the message header, and since any
   reasonable authentication scheme will require more than two octets,
   the authentication scheme for RIP version 2 will use the space of an
   entire RIP entry. If the Address Family Identifier of the first (and
   only the first) entry in the message is 0xFFFF, then the remainder of
   the entry contains the authentication. This means that there can be,
   at most, 24 RIP entries in the remainder of the message. If
   authentication is not in use, then no entries in the message should
   have an Address Family Identifier of 0xFFFF. A RIP message which
   contains an authentication entry would begin with the following
   format:

```
 0                   1                   2                   3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Command (1)   | Version (1)   |            unused             |
+---------------+---------------+-------------------------------+
|            0xFFFF             |    Authentication Type (2)    |
+-------------------------------+-------------------------------+
~                        Authentication (16)                    ~
+---------------------------------------------------------------+
```

   Currently, the only Authentication Type is simple password and it is
   type 2. The remaining 16 octets contain the plain text password. If
   the password is under 16 octets, it must be left-justified and padded
   to the right with nulls (0x00).

4.2 Route Tag

   The Route Tag (RT) 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" RIP
   routes (routes for networks within the RIP routing domain) from
   "external" RIP routes, which may have been imported from an EGP or
   another IGP.

   Routers supporting protocols other than RIP should be configurable to
   allow the Route Tag to be configured for routes imported from
   different sources. For example, routes imported from EGP or BGP
   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
   RIP domain use it consistently. This allows for the possibility of a
   BGP-RIP protocol interactions document, which would describe methods
   for synchronizing routing in a transit network.

4.3 Subnet mask

   The Subnet Mask field contains the subnet mask which is applied to
   the IP address to yield the non-host portion of the address. If this
   field is zero, then no subnet mask has been included for this entry.

   On an interface where a RIP-1 router may hear and operate on the
   information in a RIP-2 routing entry the following rules apply:

   1) information internal to one network must never be advertised into
      another network,

2) information about a more specific subnet may not be advertised
   where RIP-1 routers would consider it a host route, and

3) supernet routes (routes with a netmask less specific than the
   "natural" network mask) must not be advertised where they could be
   misinterpreted by RIP-1 routers.

4.4 Next Hop

The immediate next hop IP address to which packets to the destination
specified by this route entry should be forwarded. Specifying a
value of 0.0.0.0 in this field indicates that routing should be via
the originator of the RIP advertisement. An address specified as a
next hop must, per force, be directly reachable on the logical subnet
over which the advertisement is made.

The purpose of the Next Hop field is to eliminate packets being
routed through extra hops in the system. It is particularly useful
when RIP is not being run on all of the routers on a network. A
simple example is given in Appendix A. Note that Next Hop is an
"advisory" field. That is, if the provided information is ignored, a
possibly sub-optimal, but absolutely valid, route may be taken. If
the received Next Hop is not directly reachable, it should be treated
as 0.0.0.0.

4.5 Multicasting

In order to reduce unnecessary load on those hosts which are not
listening to RIP-2 messages, an IP multicast address will be used for
periodic broadcasts. The IP multicast address is 224.0.0.9. Note
that IGMP is not needed since these are inter-router messages which
are not forwarded.

On NBMA networks, unicast addressing may be used. However, if a
response addressed to the RIP-2 multicast address is received, it
should be accepted.

In order to maintain backwards compatibility, the use of the
multicast address will be configurable, as described in section 5.1.
If multicasting is used, it should be used on all interfaces which
support it.

4.6 Queries

If a RIP-2 router receives a RIP-1 Request, it should respond with a
RIP-1 Response. If the router is configured to send only RIP-2
messages, it should not respond to a RIP-1 Request.

5. Compatibility

   RFC [1] showed considerable forethought in its specification of the
   handling of version numbers. It specifies that RIP messages of
   version 0 are to be discarded, that RIP messages of version 1 are to
   be discarded if any Must Be Zero (MBZ) field is non-zero, and that
   RIP messages of any version greater than 1 should not be discarded
   simply because an MBZ field contains a value other than zero. This
   means that the new version of RIP is totally backwards compatible
   with existing RIP implementations which adhere to this part of the
   specification.

5.1 Compatibility Switch

   A compatibility switch is necessary for two reasons. First, there
   are implementations of RIP-1 in the field which do not follow RFC [1]
   as described above. Second, the use of multicasting would prevent
   RIP-1 systems from receiving RIP-2 updates (which may be a desired
   feature in some cases). This switch should be configurable on a
   per-interface basis.

   The switch has four settings: RIP-1, in which only RIP-1 messages are
   sent; RIP-1 compatibility, in which RIP-2 messages are broadcast;
   RIP-2, in which RIP-2 messages are multicast; and "none", which
   disables the sending of RIP messages. It is recommended that the
   default setting be either RIP-1 or RIP-2, but not RIP-1
   compatibility. This is because of the potential problems which can
   occur on some topologies. RIP-1 compatibility should only be used
   when all of the consequences of its use are well understood by the
   network administrator.

   For completeness, routers should also implement a receive control
   switch which would determine whether to accept, RIP-1 only, RIP-2
   only, both, or none. It should also be configurable on a per-
   interface basis. It is recommended that the default be compatible
   with the default chosen for sending updates.

5.2 Authentication

   The following algorithm should be used to authenticate a RIP message.
   If the router is not configured to authenticate RIP-2 messages, then
   RIP-1 and unauthenticated RIP-2 messages will be accepted;
   authenticated RIP-2 messages shall be discarded. If the router is
   configured to authenticate RIP-2 messages, then RIP-1 messages and
   RIP-2 messages which pass authentication testing shall be accepted;
   unauthenticated and failed authentication RIP-2 messages shall be
   discarded. For maximum security, RIP-1 messages should be ignored

when authentication is in use (see section 4.1); otherwise, the
routing information from authenticated messages will be propagated by
RIP-1 routers in an unauthenticated manner.

Since an authentication entry is marked with an Address Family
Identifier of 0xFFFF, a RIP-1 system would ignore this entry since it
would belong to an address family other than IP. It should be noted,
therefore, that use of authentication will not prevent RIP-1 systems
from seeing RIP-2 messages. If desired, this may be done using
multicasting, as described in sections 4.5 and 5.1.

5.3 Larger Infinity

While on the subject of compatibility, there is one item which people
have requested: increasing infinity. The primary reason that this
cannot be done is that it would violate backwards compatibility. A
larger infinity would obviously confuse older versions of rip. At
best, they would ignore the route as they would ignore a metric of
16. There was also a proposal to make the Metric a single octet and
reuse the high three octets, but this would break any implementations
which treat the metric as a 4-octet entity.

5.4 Addressless Links

As in RIP-1, addressless links will not be supported by RIP-2.

6. Interaction between version 1 and version 2

Because version 1 packets do not contain subnet information, the
semantics employed by routers on networks that contain both version 1
and version 2 networks should be limited to that of version 1.
Otherwise it is possible either to create blackhole routes (i.e.,
routes for networks that do not exist) or to create excessive routing
information in a version 1 environment.

Some implementations attempt to automatically summarize groups of
adjacent routes into single entries, the goal being to reduce the
total number of entries. This is called auto-summarization.

Specifically, when using both version 1 and version 2 within a
network, a single subnet mask should be used throughout the network.
In addition, auto-summarization mechanisms should be disabled for
such networks, and implementations must provide mechanisms to disable
auto-summarization.

7. Security Considerations

    The basic RIP protocol is not a secure protocol. To bring RIP-2 in
    line with more modern routing protocols, an extensible authentication
    mechanism has been incorporated into the protocol enhancements. This
    mechanism is described in sections 4.1 and 5.2. Security is further
    enhanced by the mechanism described in [3].

Appendix A

   This is a simple example of the use of the next hop field in a rip
   entry.

```
      -----   -----   -----           -----   -----   -----
     |IR1|   |IR2|   |IR3|           |XR1|   |XR2|   |XR3|
      --+--   --+--   --+--           --+--   --+--   --+--
       |       |       |               |       |       |
      --+-------+-------+--------------+-------+-------+--
       <------------RIP-2------------>
```

   Assume that IR1, IR2, and IR3 are all "internal" routers which are
   under one administration (e.g. a campus) which has elected to use
   RIP-2 as its IGP. XR1, XR2, and XR3, on the other hand, are under
   separate administration (e.g. a regional network, of which the campus
   is a member) and are using some other routing protocol (e.g. OSPF).
   XR1, XR2, and XR3 exchange routing information among themselves such
   that they know that the best routes to networks N1 and N2 are via
   XR1, to N3, N4, and N5 are via XR2, and to N6 and N7 are via XR3. By
   setting the Next Hop field correctly (to XR2 for N3/N4/N5, to XR3 for
   N6/N7), only XR1 need exchange RIP-2 routes with IR1/IR2/IR3 for
   routing to occur without additional hops through XR1. Without the
   Next Hop (for example, if RIP-1 were used) it would be necessary for
   XR2 and XR3 to also participate in the RIP-2 protocol to eliminate
   extra hops.

References

   [1] Hedrick, C., "Routing Information Protocol", STD 34, RFC 1058,
       Rutgers University, June 1988.

   [2] Malkin, G., and F. Baker, "RIP Version 2 MIB Extension", RFC
       1389, January 1993.

   [3] Baker, F., and R. Atkinson, "RIP-II MD5 Authentication", RFC
       2082, January 1997.

   [4] Bellman, R. E., "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 Postel, J., "Requirements for Internet Gateways",
       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] Floyd, S., and V. Jacobson, "The synchronization of Periodic
        Routing Messages," ACM Sigcom '93 symposium, September 1993.

   [11] Baker, F., "Requirements for IP Version 4 Routers." RFC 1812,
        June 1995.

Author's Address

   Gary Scott Malkin
   Bay Networks
   8 Federal Street
   Billerica, MA 01821

   Phone: (978) 916-4237
   EMail: gmalkin@baynetworks.com

Full Copyright Statement