An Introduction to IGRP

Charles L. Hedrick
RUTGERS
The State University of New Jersey
Center for Computers and Information Services
Laboratory for Computer Science Research

22 August 1991

Acknowledgements

This paper really should show Len Bosack of cisco Systems as co-author, and possibly should also list an unidentified lawyer at Townsend and Townsend. Most of the ideas behind IGRP are Len's. However since he has't even seen this paper, it would seem misleading to list him as an author.

This paper is based on the patent application for IGRP, and on an examination of the code implementing IGRP in the cisco gateways. Unlike previous drafts, this paper follows the code more closely than the patent application. The exceptions are a couple of features which are not yet implemented, but which are planned to be. The background and evaluative comments are my own. However when it came to giving equations and detailed algorithmic descriptions, it seemed to make more sense simply to lift them from the patent application, to the extent that they were consistent with the way the code actually works. I have tried to translate them from legalese into wording more appropriate for technical discussions, but inevitably there are going to be similarities in both organization and wording between this paper and the patent application. Unfortunately patent applications do not follow the usual conventions of scholarly attribution. But it is quite clear that most of the ideas in it came from Len Bosack, with some wording contributed by Townsend and Townsend. But of course I am responsible for the final form, and should bear the blame for any errors that may appear.

This paper is an introduction to IGRP, cisco System’s Inter-Gateway Routing Protocol. It has two purposes. One is to form an introduction to the IGRP technology, for those who are interested in using, evaluating, and possibly implementing it. The other is to give wider exposure to some interesting ideas that are embodied in IGRP. I believe even those who are not interested in using IGRP per se may find some of these ideas useful.

This draft is intended for discussion in the open distance-vector group within the IETF. Thus some attempt has been made to give protocol details. It is not clear whether there are enough details to do a complete implementation or not. However that is the intention.

This document has been checked carefully against the source code for cisco’s Gateway Server, release 8.0(3). Where possible, the description is consistent with that code. However no attempt has been made to describe all of the internal data structures or algorithms, since this is a protocol description, not an implementation design. In particular, here are some areas in which this description can be expected to differ from cisco’s implementation:

- cisco does not yet implement multiple types of service, nor tests that depend upon the hop count. However they do maintain and propagate the necessary information. The description here indicates how I believe that information will be used in future releases.
- cisco has a number of administrative controls, allowing filters and modifications of various sorts on routing information. Since these apply to all routing protocols implemented by cisco, I have chosen to view these controls as features of the implementation, and not part of the protocol. They are not reflected in any of the data exchanged between gateways. Thus there is nothing to prevent other implementors from choosing different sorts of controls. In previous releases of the cisco software, "administrative distance" appeared to be part of IGRP. However in 8.0, administrative distance is not propagated by IGRP, but is purely local to the gateway. I do not consider it be part of the protocol, and do not describe it here.
- cisco supplies several ways to define default routes. Again, I have chosen to regard this as a feature of the implementation rather than the protocol. However the concept of "exterior routes" is reflected in the protocol, and will be described here.

This edition has been updated based on Cisco’s release 8.2(5). A full comparison against the source code has not been done. However changes have been made to reflect the fact that variance is no longer implemented, and that holddowns may now be disabled.

The technology described here is the subject of a patent application by cisco. This paper is being circulated with cisco’s permission. However that permission should not be construed as giving any sort of license. The legal implications of implementing the ideas described in this paper are between you, your lawyer, and cisco.

1. Goals for IGRP

IGRP is a protocol that allows a number of gateways to coordinate their routing. Its goals are

- stable routing even in very large or complex networks. No routing loops should occur, even as transients.
- fast response to changes in network topology
- low overhead. That is, IGRP itself should not use more bandwidth than what is actually needed for its task.
- splitting traffic among several parallel routes when they are of roughly equal desirability.
- taking into account error rates and level of traffic on different paths
- the ability to handle multiple "types of service" with a single set of information.

The current implemention of IGRP handles routing for TCP/IP. However the basic design is intended to be able to handle a variety of protocols.

During the last few years, routing has suddenly become a more difficult problem than it used to be. A few years ago, protocols such as RIP were sufficient to handle most real networks. However growth in the Internet, and decentralization of control of its structure, have now resulted in a system of networks that is nearly beyond our capabilities to manage. Similar situations are occuring in large corporate networks as well. IGRP is one tool intended to help attack this problem.

No one tool is going to solve all routing problems. Conventionally the routing problem is broken into several pieces. Protocols such as IGRP are called "internal gateway protocols" (IGP's). They are intended for use within a single set of networks, either under a single management or closely coordinated managements. Such sets of networks are connected by "external gateway protocols" (EGP's). An IGP is designed to keep track of a good deal of detail about network topology. Priority in designing an IGP is placed on producing optimal routes and responding quickly to changes. An EGP is intended to protect one system of networks against errors or intentional misrepresentation by other systems. Priority in designing an EGP is on stability and administrative controls. Often it is sufficient for an EGP to produce a reasonable route, rather than the optimal route. In fact, there are features in cisco's implementation that allow IGRP to be used as an EGP in some circumstances. However the emphasis in its design is on use as an IGP.

IGRP has some similarities to older protocols such as Xerox's Routing Information Protocol, Berkeley's RIP, and Dave Mills' Hello. It differs from these protocols primilarly in being designed for larger and more complex networks. Section 4 gives a more detailed comparison with RIP, which is the most widely used of the older generation of protocols.

Like these older protocols, IGRP is a distance vector protocol. In such a protocol, gateways exchange routing information only with adjacent gateways. This routing information contains a summary of information about the rest of the network. It can be shown mathematically that all of the gateways taken together are solving an optimization problem by what amounts to a distributed algorithm. Each gateway only needs to solve part of the problem, and it only has to receive a portion of the total data.

The major alterative is a class of algorithms referred to as SPF (shortest-path first). These are based on a flooding technique, where every gateway is kept up to date about the status of every interface on every other gateway. Each gateway independently solves the optimization problem from its point of view using data for the entire network. There are advantages to each approach. In some circumstances SPF may be able to respond to changes more quickly. In order to prevent routing loops, IGRP has to ignore new data for a few minutes after certain kinds of changes. Because SPF has information directly from each gateway, it is able to avoid these routing loops. Thus it can act on new information immediately. However SPF has to deal with substantially more data than IGRP, both in internal data structures and in messages between gateways. Thus SPF implementations can be expected to have higher overhead than IGRP implementations, other things being equal.

2. The Routing Problem

IGRP is intended for use in gateways connecting several networks. We assume that the networks use packet-based technology. In effect the gateways act as packet switches. When a system connected to one network wants to send a packet to a system on a different network, it addresses the packet to a gateway. If the destination is on one of the networks connected to the gateway, the gateway will forward the packet to the destination. If the destination is more distant, the gateway will forward the packet to another gateway that is closer to the destination. Gateways use routing tables to help them decide what to do with packets. Here is a simple example routing table. (Addresses used in the examples are IP addresses taken from Rutgers University. Note that the basic routing problem is similar for other protocols as well, but this description will assume that IGRP is being used for routing IP.)

```
network    gateway     interface

128.6.4    none        ethernet 0
128.6.5    none        ethernet 1
128.6.21   128.6.4.1   ethernet 0
128.121    128.6.5.4   ethernet 1
10         128.6.5.4   ethernet 1
```

(Actual IGRP routing tables have additional information for each gateway, as we will see.) This gateway is connected to two Ethernets, called 0 and 1. They have been given IP network numbers (actually subnet numbers) 128.6.4 and 128.6.5. Thus packets addressed for these specific networks can be sent directly to the destination, simply by using the appropriate Ethernet interface. There are two nearby gateways, 128.6.4.1 and 128.6.5.4. Packets for networks other than 128.6.4 and 128.6.5 will be forwarded to one or the other of those gateways. The routing table indicates which gateway should be used for which network. For example, packets addressed to a host on network 10 should be forwarded to gateway 128.6.5.4. One hopes that this gateway is closer to network 10, i.e. that the best path to network 10 goes through this gateway. The primary purpose of IGRP is allow the gateways to build and maintain routing tables like this.

3. Summary of IGRP

As mentioned above, IGRP is a protocol that allows gateways to build up their routing table by exchanging information with other gateways. A gateway starts out with entries for all of the networks that are directly connected to it. It gets information about other networks by exchanging routing updates with adjacent gateways. In the simplest case, the gateway will find one path that represents the best way to get to each network. A path is characterized by the next gateway to which packets should be sent, the network interface that should be used, and metric information. Metric information is a set of numbers that characterize how good the path is. This allows the gateway to compare paths that it has heard from various gateways and decide which one to use. There are often cases where it makes sense to split traffic between two or more paths. IGRP will do this whenever two or more paths are equally good. The user can also configure it to split traffic when paths are almost equally good. In this case more traffic will be sent along the path with the better metric. The intent is that traffic can be split between a 9600 bps line and a 19200 bps line, and the 19200 line will get roughly twice as much traffic as the 9600 bps line.

The metric used by IGRP includes

- the topological delay time
- the bandwidth of the narrowest bandwidth segment of the path
- the channel occupancy of the path
- the reliability of the path.

Topological delay time is the amount of time it would take to get to the destination along that path, assuming an unloaded network. Of course there is additional delay when the network is loaded. However load is accounted for by using the channel occupany figure, not by attempting to measure actual delays. The path bandwidth is simply the bandwidth in bits per second of the slowest link in the path. Channel occupancy indicates how much of that bandwidth is currently in use. It is measured, and will change with load. Reliability indicates the current error rate. It is the fraction of packets that arrive at the destination undamaged. It is measured.

Although they are not used as part of the metric, two addition pieces of information are passed with it: hop count and MTU. The hop count is simply the number of gateways that a packet will have to go through to get to the destination. MTU is the maximum packet size that can be sent along the entire path without fragmentation. (That is, it is the minimum of the MTU's of all the networks involved in the path.)

Based on the metric information, a single "composite metric" is calculated for the path. The composite metric combines the effect of the various metric components into a single number representing the "goodness" of that path. It is the composite metric that is actually used to decide on the best path.

Periodically each gateway broadcasts its entire routing table (with some censoring because of the split horizon rule) to all adjacent gateways. When a gateway gets this broadcast from another gateway, it compares the table with its existing table. Any new destinations and paths are added to the gateway's routing table. Paths in the broadcast are compared with existing paths. If a new path is better, it may replace the existing one. Information in the broadcast is also used to update channel occupancy and other information about existing paths. This general procedure is similar to that used by all distance vector protocols. It is referred to in the mathematical literature as the Bellman-Ford algorithm. For a detailed development of the basic procedure, see RFC 1058, which describes RIP, an older distance vector protocol.

In IGRP, the general Bellman-Ford algorithm is modified in three critical aspects. First, instead of a simple metric, a vector of metrics is used to characterize paths. Second, instead of picking a single path with the smallest metric, traffic is split among several paths, whose metrics fall into a specified range. Third, several features are introduced to provide stability in situations where the topology is changing.

The best path is selected based on a composite metric

[(K1 / Be) + (K2 * Dc)] r

Where:

K1, K2 = constants;

Be = unloaded path bandwidth x (1 - channel occupancy)

Dc = topological delay; and

r = reliability.

The path having the smallest composite metric will be the best path. Where there are multiple paths to the same destination, the gateway can route the packets over more than one path. This is done in accordance with the composite metric for each data path. For instance, if one path has a composite metric of 1 and another path has a composite metric of 3, three times as many packets will be sent over the data path having the composite metric of 1. However, only paths whose composite metrics are with a certain range of the smallest composite metric will be used. K1 and K2 indicate the weight to be assigned to bandwidth and delay. These will depend upon the "type of service". For example, interactive traffic would normally place a higher weight on delay, and file transfer on bandwidth.

There are two advantages to using a vector of metric information. The first is that it provides the ability to support multiple types of service from the same set of data. The second advantage is improved accuracy. When a single metric is used, it is normally treated as if it were a delay. Each link in the path is added to the total metric. If there is a link with a low bandwidth, it is normally represented by a large delay. However bandwidth limitations don't really cumulate the way delays do. By treating bandwidth as a separate component, it can be handled correctly. Similarly, load can be handled by a separate channel occupancy number.

IGRP provides a system for interconnecting computer networks which can stably handle a general graph topology including loops. The system maintains full path metric information, i.e., it knows the path parameters to all other networks to which any gateway is connected. Traffic can be distributed over parallel paths and multiple path parameters can be simultaneously computed over the entire network.

4. Comparison with RIP

This section compares IGRP with RIP. This comparison is useful because RIP is used widely for purposes similar to IGRP. However doing this is not entirely fair. RIP was not intended to meet all of the same goals as IGRP. RIP was intended for use in small networks with reasonably uniform technology. In such applications it is generally adequate.

The most basic difference between IGRP and RIP is the structure of their metrics. Unfortunately this is not a change that can simply be retrofitted to RIP. It requires the new algorithms and data structures present in IGRP.

RIP uses a simple "hop count" metric to describe the network. Unlike IGRP, where every path is described by a delay, bandwidth, etc., in RIP it is described by a number from 1 to 15. Normally this number is used to represent how many gateways the path goes through before getting to the destination. This means that no distinction is made between a slow serial line and an Ethernet. In some implementations of RIP, it is possible for the system administrator to specify that a given hop should be counted more than once. Slow networks can be represented by a large hop count. But since the maximum is 15, this can’t be done very much. E.g. if an Ethernet is represented by 1 and a 56Kb line by 3, there can be at most 5 56Kb lines in a path, or the maximum of 15 is exceeded. In order to represent the full range of available network speeds, and allow for a large network, studies done by cisco suggest that a 24-bit metric is needed. If the maximum metric is too small, the system administrator is presented with an unpleasant choice: either he can’t distinguish between fast and slow routes, or he can’t fit his whole network into the limit. In fact a number of national networks are now large enough that RIP can’t handle them even if every hop is counted only once. RIP simply can’t be used for such networks.

The obvious response would be to modify RIP to allow a larger metric. Unfortunately, this won’t work. Like all distance vector protocols, RIP has the problem of "counting to infinity". This is described in more detail in RFC 1058. When topology changes, spurious routes will be introduced. The metrics associated with these spurious routes slowly increase until they reach 15, at which point the routes are removed. 15 is a small enough maximum that this process will converge fairly quickly, assuming that triggered updates are used. If RIP were modified to allow a 24-bit metric, loops would persist long enough for the metric to be counted up to 2**24. This is not tolerable. IGRP has features designed to prevent spurious routes from being introduced. These are discussed below in section 5.2. It is not practical to handle complex networks without introducing such features or changing to a protocol such as SPF.

IGRP does a bit more than simply increase the range of allowable metrics. It restructures the metric to describe delay, bandwidth, reliability, and load. It is possible to represent such considerations in a single metric such as RIP’s. However the approach taken by IGRP is potentially more accurate. For example, with a single metric, several successive fast links will appear to be equivalent to a single slow one. This may be the case for interactive traffic, where delay is the primary concern. However for bulk data transfer, the primary concern is bandwidth, and adding metrics together is not the right approach there. IGRP handles delay and bandwidth separately, cumulating delays, but taking the minimum of the bandwidths. It is not easy to see how to incorporate the effects of reliability and load into a single-component metric.

In my opinion, one of the big advantages of IGRP is ease of configuration. It can directly represent quantities that have physical meaning. This means that it can be set up automatically, based on interface type, line speed, etc. With a single-component metric, the metric is more likely to have to be "cooked" to incorporate effects of several different things.

Other innovations are more a matter of algorithms and data structures than of the routing protocol. For example, IGRP specifies algorithms and data structures that support splitting traffic among several routes. It is certainly possible to design an implementation of RIP that does this. However once routing is being reimplemented, there is no reason to stick with RIP.

So far I have described "generic IGRP", a technology which could support routing for any network protocol. However in this section it is worth mentioning a bit more about the specific TCP/IP implementation. That is the implementation that is going to be compared with RIP.

RIP update messages simply contain snapshots of the routing table. That is, they have a number of destinations and metric values, and little else. The IP implementation of IGRP has additional structure. First, the update message is identified by an "autonomous system number." This terminology comes out of the Arpanet tradition, and has specific meaning there. However for most networks what it means is that you can run several different routing systems on the same network. This is useful for places where networks from several organizations converge. Each organization can maintain its own routing. Because each update is labelled, gateways can be configured to pay attention only to the right one. Certain gateways are configured to receive updates from several autonomous systems. They pass information between the systems in a controlled manner. Note that this is not a complete solution to problems of routing security. Any gateway can be configured to listen to updates from any autonomous system. However it is still a very useful tool in implementing routing policies where is a reasonable degree of trust between the network administrators.

The second structural feature about IGRP update messages affects the way default routes are handled by IGRP. Most routing protocols have a concept of default route. It is often not practical for routing updates to list every network in the world. Typically a set of gateways need detailed routing information for networks within their organization. All traffic for destinations outside their organization can be sent to one of a few boundary gateways. Those boundary gateways may have more complete information. The route to the best boundary gateway is a "default route". It’s a default in the sense that it is used to get to any destination that is not listed specifically in the internal routing updates. RIP, and some other routing protocols, circulate information about the default route as if it were a real network. IGRP takes a different approach. Rather than a single fake entry for the default route, IGRP allows real networks to be flagged as candidates for being a default. This is implemented by placing information about those networks in a special exterior section of the update message. However it might as well be thought of as turning on a bit associated with those networks. Periodically IGRP scans all the candidate default routes and chooses the one with the lowest metric as the actual default route.

Potentially this approach to defaults is somewhat more flexible than the approach taken by most RIP implementations. Most typically RIP gateways can be set to generate a default route with a certain specified metric. The intention is that this would be done at boundary gateways. However there are situations where this isn’t really good enough. Suppose that Rutgers University is directly connected to two regional networks, say NYSERnet at 56kbps and JvNCnet at T1. Normally the link to JvNCnet would be the best default. Using RIP we would configure our gateway to JvNCnet to advertise a default route with a lower metric than our gateway to NYSERnet. However suppose something happens within JvNCnet so that its access to the national networks is either cut off or becomes much worse (e.g. it might end up passing traffic through a backup link to some other regional). In that case it might be better to use NYSERnet. The typical RIP implementations would not adapt to that change, because the boundary gateway is generating a default with a fixed metric. With IGRP we would configure our gateways so that the NSFnet national backbone was flagged as a candidate default route. In that case, we would choose the default route at each gateway based on metric information that takes into account the entire path back to the national backbone. So the default would change to take into account events within our regional networks, assuming that they pass full IGRP metric information to us.

5. Detailed Description

5.1 Overall Description

When a gateway is first turned on, its routing table is initialized. This may be done by an operator from a console terminal, or by reading information from configuration files. A description of each network connected to the gateway is provided, including the topological delay along the link (i.e., how long it takes a single bit to transverse the link) and the bandwidth of the link.

```
network 1      network 2          network 3     network 4
128.6.5         128.6.4            128.6.21      128.121
 =====  ======================   ==========   ================
    |     |          |         |    |        |    |         |
  __|_____|__   _____|_____  __|____|__  __|____|_____  ___|________
  128.6.4.2     128.6.4.3    128.6.4.1   128.6.21.1    128.121.50.2
  128.6.5.1                  128.6.21.2  128.121.50.1
  __________    __________   __________  ____________  ____________
   gateway R    computer A    gateway S    gateway T    computer B

 Fig 1.  A simple example network
```

For instance, in Fig. 1, gateway S would be told that it is connected to networks 2 and 3 via the corresponding interfaces. Thus, initially, gateway 2 only knows that it can reach any destination computer in networks 2 and 3. All the gateways are programmed to periodically transmit to their neighboring gateways the information that they have been initialized with, as well as information gathered from other gateways. Thus, gateway S would receive updates from gateways R and T and learn that it can reach computers in network 1 through gateway R and computers in network 4 through gateway T. Since gateway S sends its entire routing table, in the next cycle gateway T will learn that it can get to network 1 through gateway S. It is easy to see that information about every network in the system will eventually reach every gateway in the system, providing only that the network is fully connected.

```
               ________ network 1
                   |
                 gw A --nw2-- gw C
                   |           / |
                   |         /   |
                 nw3    nw4   nw5
                   |    /       |
                   |  /         |
                 gw B         gw D
               ____|____________|____  network 6

     Fig 2.  Example of alternate paths
```

Each gateway computes a composite metric to determine the desirability of the data paths to destination computers. For instance, in Fig 2, for a destination in network 6, gateway A would compute metric functions for two paths, via gateways B and C. Note that paths are defined simply by the next hop. There are actually three possible routes from A to network 6:

- direct to B
- to C and then to B
- to C and then to D

However gateway A need not choose between the two routes involving C. The routing table in A has a single entry representing the path to C. Its metric represents the best way of getting from C to the final destination. If A sends a packet to C, it is up to C to decide whether to use B or D.

The composite metric function computed for each data path is as follows:

$$[(K1 / Be) + (K2 * Dc)]\, r \qquad \text{Eq. 1}$$

Where:

r = fractional reliability
(% of transmissions that are sucessfully received at the next hop.)

Dc = composite delay;

Be = effective bandwidth: unloaded bandwidth x (1 - channel occupancy)

K1, K2 = constants.

In principle the composite delay, Dc, could be determined as follows:

$$Dc = Ds + Dcir + Dt \qquad \text{Eq. 2}$$

Where:

Ds = switching delay;

Dcir = circuit delay (propagation delay of 1 bit); and

Dt = transmission delay (no-load delay for a 1500 bit message).

However in practice a standard delay figure is used for each type of network technology. E.g there will be a standard delay figure for Ethernet, and for serial lines at any particular bit rate.

Here is an example of how gateway A's routing table might look in the case of Fig 2. (Note that individual components of the metric vector are not shown, for simplicity.)

	interface	next gateway	metric
network 1	nw 1	none	directly connected
network 2	nw 2	none	directly connected
network 3	nw 3	none	directly connected
network 4	nw 2	C	1270
	nw 3	B	1180
network 5	nw 2	C	1270
	nw 3	B	2130
network 6	nw 2	C	2040
	nw 3	B	1180

Fig 3. An example routing table

The basic process of building up a routing table by exchanging information with neighbors is described by the Bellman-Ford algorithm. The algorithm has been used in earlier protocols such as RIP (RFC 1058). In order to deal with more complex networks, IGRP adds three features to the basic Bellman-Ford algorithm:

1. Instead of a simple metric, a vector of metrics is used to characterize paths. A single composite metric can be computed from this vector according to equation 1. Use of a vector allows the gateway to accomodate different types of service, by using several different coefficients in Eq. 1. It also allows a more accurate representation of the characteristics of the network than a single metric.

2. Instead of picking a single path with the smallest metric, traffic is split among several paths with metrics falling into a specified range. This allows several routes to be used in parallel, providing a greater effective bandwidth than any single route. A variance V is specified by the network administrator. All paths with minimal composite metric M are kept. In addition, all paths whose metric is less than V x M are kept. Traffic is distributed among multiple paths in inverse proportion to the composite metrics.

3. There are some problems with this concept of variance. It is difficult to come up with strategies that make use of variance values greater than 1, and do not also lead to packets looping. In Cisco release 8.2, the variance feature is not implemented. (I am not sure in what release the feature was removed.) The effect of this is to set the variance permanently to 1.

4. Several features are introduced to provide stability in situations where the topology is changing. These features are intended to prevent routing loops and "counting to infinity," which have characterized previous attempts to use Ford-type algorithms for this type of application. The primary stability features are "holddowns", "triggered updates", "split horizon," and "poisoning". These will be discussed in more detail below.

Traffic splitting (point 2) raises a rather subtle danger. The variance V is designed to allow gateways to use parallel paths of different speeds. For example, there might be a 9600 bps line running in parallel with a 19200 bps line, for redundancy. If the variance V is 1, only the best path will be used. So the 9600 bps line will not be used if the 19200 bps line has a reasonable reliability. (However if several paths are the same, the load will be shared among them.) By raising the variance, we can allow traffic to be split between the best route and other routes that are nearly as good. With a large enough variance, traffic will be split between the two lines. The danger is that with a large enough variance, paths become allowed that aren't just slower, but are actually "in the wrong direction". Thus there should be an additional rule to prevent traffic from being sent "upstream": No traffic is sent along paths whose remote composite metric (the composite metric calculated at the next hop) is greater than the composite metric calculated at the gateway. In general system administrators are encouraged not to set the variance above 1 except in specific situations where parallel paths need to be used. In this case, the variance is carefully set to provide the "right" results.

IGRP is intended to handle multiple "types of service," and multiple protocols. Type of service is a specification in a data packet that modifies the way paths are to be evaluted. For example, the TCP/IP protocol allows the packet to specify the relative importance of high bandwidth, low delay, or high reliability. Generally, interactive applications will specify low delay, whereas bulk transfer applications will specify high bandwidth. These requirements determine the relative values of K1 and K2 that are appropriate for use in Eq. 1. Each combination of specifications in the packet that is to be supported is referred to as a "type of service". For each type of service, a set of parameters K1 and K2 must be chosen. A routing table is kept for each type of service. This is done because paths are selected and ordered according to the composite metric defined by Eq. 1. This is different for each type of service. Information from all of these routing tables is combined to produce the routing update messages exchanged by the gateways, as described in Fig. 7.

5.2 Stability Features

This section describes holddowns, triggered updates, split horizon, and poisoning. These features are designed to prevent gateways from picking up erroneous routes. As described in RFC 1058, this can happen when a route becomes unusable, due to failure of a gateway or a network. In principle, the adjacent gateways detect failures. They then send routing updates that show the old route as unusable. However it is possible for updates not to reach some parts of the network at all, or to be delayed in reaching certain gateways. A gateway that still believes the old route is good can continue spreading that information, thus reentering the failed route into the system. Eventually this information will propagate through the network and come back to the gateway that reinjected it. The result is a circular route.

In fact there is some redundancy among the countermeasures. In principle, holddowns and triggered updates should be sufficient to prevent erroneous routes in the first place. However in practice, communications failures of various kinds can cause them to be be insufficient. Split horizon and route poisoning are intended to prevent routing loops in any case.

Normally, new routing tables are sent to neighboring gateways on a regular basis (every 90 seconds by default, although this can be adjusted by the system administrator). A triggered update is a new routing table that is sent immediately, in response to some change. The most important change is removal of a route. This can happen because a timeout has expired (probably a neighboring gateway or line has gone down), or because an update message from the next gateway in the path shows that the path is no longer usable. When a gateway G detects that a route is no longer usable, it triggers an update immediately. This update will show that route as unusable. Consider what happens when this update reaches the neighboring gateways. If the neighbor's route pointed back to G, the neighbor must remove the route. This causes the neighbor to trigger an update, etc. Thus a failure will trigger a wave of update messages. This wave will propagate throughout that portion of the network in which routes went through the failed gateway or network.

Triggered updates would be sufficent if we could guarantee that the wave of updates reached every appropriate gateway immediately. However there are two problems. First, packets containing the update message can be dropped or corrupted by some link in the network. Second, the triggered updates don't happen instantaneously. It is possible that a gateway that has not yet gotten the triggered update will issue a regular update at just the wrong time, causing the bad route to be reinserted in a neighbor that had already gotten the triggered update. Holddowns are designed to get around these problems. The holddown rule says that when a route is removed, no new route will be accepted for the same destination for some period of time. This gives the triggered updates time to get to all other gateways, so that we can be sure any new routes we get aren't just some gateway reinserting the old one. The holddown period must be long enough to allow for the wave of triggered updates to go throughout the network. In addition, it should include a couple of regular broadcast cycles, to handle dropped packets. Consider what happens if one of the triggered updates is dropped or corrupted. The gateway that issued that update will issue another update at the next regular update. This will restart the wave of triggered updates at neighbors that missed the initial wave.

The combination of triggered updates and holddowns should be sufficient to get rid of expired routes and prevent them from being reinserted. However some additional precautions are worth doing anyway. They allow for very lossy networks, and networks that have become partitioned. The additional precautions called for by IGRP are split horizon and route poisoning. Split horizon arises from the observation that it never makes sense to send a route back in the direction from which it came. Consider the following situation:

```
 network 1       network 2
-------------X----------------X
        gateway A         gateway B
```

Gateway A will tell B that it has a route to network 1. When B sends updates to A, there is never any reason for it to mention network 1. Since A is closer to 1, there is no reason for it to consider going via B. The split horizon rule says a separate update message should be generated for each neighbor (actually each neighboring network). The update for a given neighbor should omit routes that point to that neighbor. This rule prevents loops between adjacent gateways. E.g. suppose A's interface to network 1 fails. Without the split horizon rule, B would be telling A that it can get to 1. Since it no longer has a real route, A might pick up that route. In this case, A and B would both have routes to 1. But A would point to B and B would point to A. Of course triggered updates and holddowns should prevent this from happening. But since there's no reason to send information back to the place it came from, split horizon is worth doing anyway. In addition to its role in preventing loops, split horizon keeps down the size of update messages.

Split horizon should prevent loops between adjacent gateways. Route poisoning is intended to break larger loops. The rule is that when an update shows the metric for an existing route to have increased sufficiently, there is a loop. The route should be removed and put into holddown. Currently the rule is that a route is removed if the composite metric increases more than a factor of 1.1. It is not safe for just any increase in composite metric to trigger removal of the route, since small metric changes can occur due to changes in channel occupancy or reliability. So the factor of 1.1 is just a heuristic. The exact value isn't critical. We expect this rule only to be needed to break very large loops, since small ones will be prevented by triggered updates and holddowns.

5.2.1 Disabling holddowns

As of release 8.2, Cisco's code provides an option to disable holddowns. The disadvantage of holddowns is that they delay adoption of a new route when an old route fails. With default parameters, it can take several minutes before a router adopts a new route after a change. However for the reasons explained above, it is not safe simply to remove holddowns. The result would be count to infinity, as described in RFC 1058. We conjecture, but cannot prove, that with a stronger version of route poisoning, holddowns are no longer needed to sto count to infinity. Thus disabling holddowns enables this stronger form of route poisoning. Note that split horizon and triggered updates are still in effect.

The stronger form of route poisoning is based on a hop count. If the hop count for a path increases, the route is removed. This will obviously remove routes that are still valid. If something elsewhere in the network changes so that the path now goes through one more gateway, the hop count will increase. In this case, the route is still valid. However there is no completely safe way to distinguish this case from routing loops (count to infinity). Thus the safest approach is to remove the route whenever the hop count increases. If the route is still legitimate, it will be reinstalled by the next update, and that will cause a triggered update that will reinstall the route elsewhere in the system.

In general, distance vector algorithms adopt new routes easily. The problem is completely purging old ones from the system. Thus a rule that is overly aggressive about removing suspicious routes should be safe.

5.3 Details of the Update Process

The set of processes described in Fig. 4-8 are intended to handle a single network protocol, e.g. TCP/IP, DECnet, or the ISO/OSI protocol. However protocol details will be given only for TCP/IP. A single gateway may process data that follows more than one protocol. Because each protocol has different addressing structures and packet formats, the computer code used to implement Figs. 4-8 will generally be different for each protocol. The process described in Fig. 4 will vary the most, as described in the detailed notes for Fig. 4. The processes described in Fig. 5-8 will have the same general structure. The primary difference from protocol to protocol will be the format of the routing update packet, which must be designed to be compatible with a specific protocol.

Note that the definition of a destination may vary from protocol to protocol. The method described here can be used for routing to individual hosts, to networks, or for more complex hierachical address schemes. Which type of routing is used will depend upon the addressing structure of the protocol. The current TCP/IP implementation supports only routing to IP networks. Thus "destination" really means IP network or subnet number. Subnet information is only kept for connected networks.

Figures 4-7 show pseudo-code for various pieces of the routing process used by the gateways. At the start of the program, acceptable protocols and parameters describing each interface are entered.

The gateway will only handle certain protocols which are listed. Any communication from a system using a protocol not on the list will be ignored. The data inputs are

- the networks to which the gateway is connected
- the unloaded bandwidth of each networks
- topological delay of each network
- reliability of each network
- channel occupancy of each network
- MTU of each network

The metric function for each data path is then computed according to Eq. 1. Note that the first three items are reasonably permanent. They are a function of the underlying network technology, and do not depend upon load. They could be set from a configuration file or by direct operator input. Note that IGRP does not use measured delay. Both theory and experience suggest that it is very difficult for protocols that use measured delay to maintain stable routing. There are two measured parameters: reliability and channel occupancy. Reliability is based on error rates reported by the network interface hardware or firmware.

In addition these inputs, the routing algorithm requires a value for several routing parameters. This includes timer values, variance, and whether holddowns are enabled. This would normally be specified by a configuration file or operator input. (As of Cisco release 8.2, the variance is permanently set to 1.)

Once initial information is entered, operations in the gateway are triggered by events - either the arrival of a data packet at one of the network interfaces, or expiration of a timer. The processes described in Figs. 4-7 are triggered as follows:

When a packet arrives, it is processed according to Fig 4. This results in the packet being sent out another interface, discarded, or accepted for further processing.

When a packet is accepted by the gateway for further processing, it is analyzed in a protocol-specific fashion not described in this specification. If the packet is a routing update, it is processed according to Fig. 5.

Figure 6 shows events triggered by a timer. The timer is set to generate an interrupt once per second. When the interrupt occurs, the process shown in Fig. 6 is executed.

Figure 7 shows a routing update subroutine. Calls to this subroutine are shown in Figs 5 and 6.

In addition, Fig. 8 shows details of metric computations, referred to in Fig. 5 and 7.

There are 4 critical time constants that control route propagation and expiration. These time constants may be set by the system administrator. However there are default values. These time constants are

- broadcast time - updates are broadcast by all gateways on all connected interfaces this often. The default is once every 90 sec.
- invalid time - if no update has been received for a given path within this amount of time, it is considered to have timed out. It should be several times the broadcast time, in order to allow for the possibility that packets containing an update could be dropped by the network. The default is 3 times the broadcast time.
- hold time - when a destination has become unreachable (or the metric has increased enough to cause poisoning), the destination goes into "holddown". During this state, no new path will be accepted for the same destination for this amount of time. The hold time indicates how long this state should last. It should be several times the broadcast time. The default value is 3 times the broadcast time plus 10 sec. (As described in section 5.2.1, it is possible to disable holddowns.)
- flush time - if no update has been received for a given destination within this amount of time, the entry for it is removed from the routing table. Note the difference between invalid time and flush time: After the invalid time a path is timed out and removed. If there are no remaining paths to a destination, the destination is now unreachable. However the database entry for the destination remains. It has to remain in order to enforce the holddown. After the flush time the database entry is removed from the table. It should be somewhat longer than the invalid time plus the holddown time. The default is 7 times the broadcast time.

These figures presuppose the following major data structures. A separate set of these data structures is kept for each protocol supported by the gateway. Within each protocol, a separate set of data structures is kept for each type of service to be supported.

For each destination known to the system, there is a (possibly null) list of paths to the destination, a holddown expiration time, and a last update time. The last update time indicates the last time any path for this destination was included in an update from another gateway. Note that there are also update times kept for each path. Holddowns are discussed in section 5.2. When the last path to a destination is removed, the destination is put into holddown, unless holddowns are disabled (See section 5.2.1). The holddown expiration time indicates the time at which the holddown expires. The fact that it is non-zero indicates that the destination is in holddown. In order to save calculation time, it is also a good idea to keep a "best metric" for each destination. This is simply the minimum of the composite metrics for all the paths to the destination.

For each path to a destination, there is the address of the next hop in the path, the interface to be used, a vector of metrics characterizing the path, including topological delay, bandwidth, reliability, and channel occupancy. Other information is also associated with each path, including hop count, MTU, source of information, the remote composite metric, and a composite metric calculated from these numbers according to equation 1. There is also a last update time. The source of information indicates where the most recent update for that path came from. In practice this is the same as the address of the next hop. The last update time is simply the time at which the most recent update arrived for this path. It is used to expire timed-out paths.

Note that an IGRP update message has three portions: interior, system (meaning "this autonomous system" but not interior), and exterior. The interior section is for routes to subnets. Not all subnet information is included. Only subnets of one network are included. This is the network associated with the address to which the update is being sent. Normally updates are broadcast on each interface, so this is simply the network on which the broadcast is being sent. (Other cases arise for responses to an IGRP request and point to point IGRP.) Major networks (i.e. non-subnets) are put into the system portion of the update message unless they are specifically flagged as exterior.

A network will be flagged as exterior if it was learned from another gateway and the information arrived in the exterior portion of the update message. cisco’s implementation also allows the system administrator to declare specific networks as exterior. Exterior routes are also referred to as "candidate default". They are routes that go to or through gateways that are considered to be appropriate as defaults, to be used when there is no explicit route to a destination. For example at Rutgers we configure the gateway that connects Rutgers to our regional network so that it flags the route to the NSFnet backbone as exterior. cisco’s implementation chooses a default route by picking that exterior route with the smallest metric.

The following sections are intended to clarify certain portions of Fig. 4-8.

5.4 Packet Routing

Fig 4 describes overall processing of input packets. This is used simply to clarify terminology. Obviously this is not a complete description of what an IP gateway does.

This process uses the list of supported protocols and the information about the interfaces entered when the gateway is initialized. Details of the packet processing depend upon the protocol used by the packet. This is determined in Step A. Step A is the only portion of Fig. 4 which is shared by all protocols. Once the protocol type is known, the implementation of Fig. 4 appropriate to the protocol type is used. Details of the packet contents are described by the specifications of the protocol. The specifications of a protocol include a procedure for determining the destination of a packet, a procedure for comparing the destination with the gateway’s own addresses to determine whether the gateway itself is the destination, a procedure for determining whether a packet is a broadcast, and a procedure for determining whether the destination is part of a specified network. These procedures are used in steps B and C of Fig. 4. The test in step D requires a search of the destinations listed in the routing table. The test is satisfied if there is an entry in the routing table for the destination, and that destination has associated with it at least one usable path. Note that the destination and path data used in this and the next step are maintained separately for each type of service supported. Thus this step begins by determining the type of service specified by the packet, and selecting the corresponding set of data structures to use for this and the next step.

A path is usable for the purposes of steps D and E if its remote composite metric is less than its composite metric. A path whose remote composite metric is greater than its composite metric is a path whose next hop is "farther away" from the destination, as measured by the metric. This is referred to as an "upstream path." Normally one would expect that the use of metrics would prevent upstream paths from being chosen. It is easy to see that an upstream path can never be the best one. However if a large variance is allowed, paths other than the best one can be used. Some of those could be upstream.

Step E computes the path to use. Paths whose remote composite metric is not less than their composite metrics are not considered. If more than one path is acceptable, such paths are used in a weighted form of round-robin alternation. The frequency with which a path is used is inversely proportional to its composite metric.

5.5 Reception Of Routing Updates

Fig 5 describes the processing of a routing update received from a neighboring gateway. Such updates consist of a list of entries, each of which gives information for a single destination. More than one entry for the same destination can occur in a single routing update, to accomodate multiple types of service. Each of these entries is processed individually, as described in Fig. 5. If an entry is in the exterior section of the update, the exterior flag will be set for the destination if it is added as a result of this process.

The entire process described in figure 5 must be repeated once for each type of service supported by the gateway, using the set of destination / path information associated with that type of service. This is shown in the outermost loop in Fig. 5. The entire routing update must be processed once for each type of service. (Note that the current implementation of IGRP does not support multiple types of service. So the outermost loop is not actually implemented.)

In step A, basic acceptability tests are done on the path. This should include reasonableness tests for the destination. Impossible ("Martian") network numbers should be rejected. (See RFC 1009 and RFC 1122 for details.) Updates are also rejected if the destination they refer to is in holddown, i.e. the holddown expiration time is non-zero and later than the current time.

In steps B the routing table is searched to see whether this entry describes a path that is already known. A path in the routing table is defined by the destination with which it is associated, the next hop listed as part of the path, the output interface to be used for the path, and the information source (the address from which the update came -- in practice normally the same as the next hop). The entry from the update packet describes a path whose destination is listed in the entry, whose output interface is the interface that the update came in, and whose next hop and information source are the address of the gateway that sent the update (the "source" S).

In steps H and T, the update process described in Fig. 7 is scheduled. This process will actually run after the entire process described in Fig. 5 has been finished. That is, the update process described in Fig. 7 will only happen once, even if it is triggered several times during the processing described in Fig. 5. Furthermore, precautions must be taken to keep updates from being issued too often, if the network is changing rapidly.

Steps K is done if the destination described by the current entry in the update packet already exists in the routing table. K compares the new composite metric computed from data in the update packet with the best composite metric for the destination. Note that the best composite metric is not recomputed at this time. So if the path being considered is already in the routing table, this test may compare new and old metrics for the same path.

Step L is performed for the paths that are worse than the existing best composite metric. This includes both new paths that are worse than existing ones and existing paths whose composite metric has increased. Step L tests whether the new path is acceptable. Note that this test implements both the test for whether a new path is good enough to keep, and route poisoning. In order to be acceptable, the delay value must not be the special value that indicates an unreachable destination (for the current IP implementation, all ones in a 24 bit field), and the composite metric (calculated as specified in Fig. 8) must be acceptable. To determine whether the composite metric is acceptable, compare it with the composite metrics of all other paths to the destination. Let M be the minimum of these. The new path is acceptable if it is $< V \times M$, where V is the variance set when the gateway was initialized. If $V = 1$ (which is always true as of Cisco release 8.2), then a metric any worse than the existing one is not acceptable. There is one exception to this: If the path already exists and is the only path to the destination, the path will be retained if the metric has not increased by more than 10% (or where holddowns are disabled, if the hop count has not increased).

Step V is done when the new information for a path indicates that the composite metric will be decreased. The composite metrics of all paths to destination D are compared. In this comparison, the new composite metric for P is used, rather than the one appearing in routing table. The minimum composite metric M is calculated. Then all paths to D are examined again. If the composite metric for any path > M x V, that path is removed. V is the variance, entered when the gateway was initialized. (As of Cisco release 8.2, the variance is permanently set to 1.)

5.6 Periodic Processing

The process described in Figure 6 is triggered once a second. It examines various timers in the routing table, to see if any has expired. These timers are described above.

In step U, the process described in Fig. 7 is activated.

Steps R and S are necessary because the composite metrics stored in the routing table depend upon the channel occupancy, which changes over time, based on measurements. Periodically the channel occupancy is recalculated, using a moving average of measured traffic through the interface. If the newly-calculated value differs from the existing one, all composite metrics involving that interface must be adjusted. Every path shown in the routing table is examined. Any path whose next hop uses interface "I" has its composite metric recalculated. This is done in accordance with eq. 1, using as the channel occupancy the maximum of the value stored in routing table as part of the path's metric, and the newly calculated channel occupancy of the interface.

5.7 Generating Update Messages

Figure 7 describes how the gateway generates update messages to be sent to other gateways. A separate message is generated for each network interface attached to the gateway. That message is then sent to all other gateways that are reachable through the interface (step J). Generally this is done by sending the message as a broadcast. However, if the network technology or protocol does not allow broadcasts, it may be necessary to send the message individually to each gateway.

In general, the message is built up by adding an entry for each destination in the routing table, in step G. Note that the destination / path data associated with each type of service must be used. In the worst case, a new entry is added to the update for each destination for each type of service. However, before adding an entry to the update message in step G, the entries already added are scanned. If the new entry is already present in the update message, it is not added again. A new entry duplicates an existing one when the destinations and next hop gateways are the same.

For the sake of simplicity, the pseudo-code omits one thing: IGRP update messages have three parts: interior, system, and exterior. Thus there are actually three loops over destinations. The first includes only subnets of the network to which the update is being sent. The second includes all major networks (i.e. non-subnets) that are not flagged as exterior. The third includes all major networks that are flagged as exterior.

Step E implements the split horizon test. In the normal case, this test fails for routes whose best path goes out the same interface that the update is being sent out. However if the update is being sent to a specific destination (e.g. in response to an IGRP request from another gateway, or as part of "point to point IGRP"), split horizon fails only if the best path originally came from that destination (its "information source" is the same as the destination) and its output interface is the same as the one the request came in from.

5.8 Computing Metric Information

Fig. 8 describes how the metric information is processed from update messages received by the gateway, and how it is generated for update messages being sent by the gateway. Note that the entry is based on one particular path to the destination. If there is more than one path to the destination, a path whose composite metric is minimum is chosen. If more than one path has the minimum composite metric, an arbitrary tie-breaking rule is used. (For most protocols, this will be based on the address of the next hop gateway.).

Fig 4. Processing incoming packets

```
       Data packet arrives using interface I

A      Determine protocol used by packet

       If protocol is not supported
           then discard packet

B      If destination address matches any of gateway’s addresses
       or the broadcast address
           then process packet in protocol-specific way

C      If destination is on a directly-connected network
           then send packet direct to the destination, using
                the encapsulation appropriate to the protocol
                and link type

D      If there are no paths to the destination in the routing
       table, or all paths are upstream
           then send protocol-specific error message and
                discard the packet

E      Choose the next path to use.  If there are more than
       one, alternate round-robin with frequency proportional
       to inverse of composite metric.

       Get next hop from path chosen in previous step.

       Send packet to next hop, using encapsulation appropriate
       to protocol and data link type.
```

Fig 5. Processing incoming routing updates

```
        Routing update arrives from source S

        For each type of service supported by gateway

            Use routing data associated with this type of service

            For each destination D shown in update

A               If D is unacceptable or in holddown
                    then ignore this entry and continue loop
                         with next destination D

B               Compute metrics for path P to D via S  (see Fig 8)

                If destination D is not already in the routing table
                then Begin

                    Add path P to the routing table, setting last
                    update times for P and D to current time.

H                   Trigger an update

                    Set composite metric for D and P to new composite
                    metric computed in step B.

                    End
```

```
          Else begin  (dest. D is already in routing table)

K              Compare the new composite metric for P with best
               existing metric for D.

                 New > old:

L                    If D is shown as unreachable in the update,
                     or holddowns are enabled and
                        the new composite metric >
                           (the existing metric for D) * V
                           [use 1.1 instead of V if V = 1,
                            as it is as of Cisco release 8.2]
O                    or holddowns are disabled and
                        P has a new hop count > old hop count
                     then Begin

                        Remove P from routing table if present

                        If P was the last route to D
                            then Unless holddowns are disabled
                                   Set holddown time for D to
                                     current time + holddown time
T                                and Trigger an update

                        End

                     else Begin

                        Compute new best composite metric for D

                        Put the new metric information into the
                        entry for P in the routing table

                        Add path P to the routing table if it
                        was not present.

                        Set last update times for P and D to
                        current time.

                        End

                 New <= old:

V                    Set composite metric for D and P to new
                     composite metric computed in step B.

                     If any other paths to D are now outside the
                     variance, remove them.

                     Put the new metric information into the
                     entry for P in the routing table

                     Set last update times for P and D to
                     current time.

               End

          End of for

      End of for
```

Fig 6. Periodic processing

```
        Process is activated by regular clock, e.g. once per second

        For each path P in the routing table (except directly
        connected interfaces)

            If current time < P’s last update time + invalid time
                then continue with the next path P

            Remove P from routing table

            If P was the last route to D
                then Set metric for D to inaccessible
                     Unless holddowns are disabled,
                         Start holddown timer for D and
                     Trigger an update

                else Recompute the best metric for D

            End of for

        For each destination D in the routing table

            If D’s metric is inaccessible
                then Begin

                Clear all paths to D

                If current time >= D’s last update time + flush time
                    then Remove entry for D

                End

            End of for


        For each network interface I attached to the gateway

R           Recompute channel occupancy and error rate

S           If channel occupancy or error rate has changed,
                then recompute metrics

            End of for

        At intevals of broadcast time

U           Trigger update
```

Fig 7. Generate update

```
       Process is caused by "trigger update"

       For each network interface I attached to the gateway

           Create empty update message

           For each type of service S supported

               Use path/destination data for S

               For each destination D

E                  If any paths to D have a next hop reached through I
                       then continue with the next destination

                   If any paths to D with minimal composite metric are
                   already in the update message
                       then continue with the next destination

G                  Create an entry for D in the update message, using
                   metric information from a path with minimal
                   composite metric (see Fig. 8)

                   End of for

               End of for

J          If there are any entries in the update message
              then send it out interface I

           End of for
```

Fig 8. Details of metric computations

This section describes the procedure for computing metrics and hop counts from an arriving routing update.

The input to this function is the entry for a specific destination in a routing update packet. The output is a vector of metrics which can be used to compute the composite metric, and a hop count. If this path is added to the routing table, the entire vector of metrics is entered in the table. The interface parameters used in the following definitions are those set when the gateway was initialized, for the interface on which the routing update arrived, except that the channel occupancy and reliability are based on a moving average of measured traffic through the interface.

delay = delay from packet + interface topological delay

bandwidth = max (bandwidth from packet, interface bandwidth)

reliability = min (reliability from packet, interface reliability)

channel occupancy = max (channel occupancy from packet,
interface channel occupancy)

(Max is used for bandwidth because the bandwidth metric is stored in inverse form. Conceptually, we want the minimum bandwidth.) Note that the original channel occupancy from the packet must be saved, since it will be needed to recompute the effective channel occupancy whenever the interface channel occupancy changes.

The following are not part of the metric vector, but are also kept in the routing table as characteristics of the path:

hop count = hop count from packet

MTU = min (MTU from packet, interface MTU)

remote composite metric - calculated from Eq. 1 using the metric values from the packet. That is, the metric components are those from the packet, and are not updated as shown above. Obviously this must be calculated before the adjustments shown above are done.

composite metric- calculated from Eq. 1 using the metric values calculated as described in this section

This section describes the procedure for computing metrics and hop count for routing updates to be sent:

This function determines the metric information and hop count to be put into an outgoing update packet. It is based on a specific path to a destination, if there are any usable paths. If there are no paths, or the paths are all upstream, the destination is called "inaccesible".

If destination is inaccessible, this is indicated by using a specific value in the delay field. This value is chosen to be larger than the largest valid delay. For the IP implementation this is all ones in a 24-bit field.

If destination is directly reachable through one of the interfaces, use the delay, bandwidth, reliability, and channel occupancy of the interface. Set hop count to 0.

Otherwise, use the vector of metrics associated with the path in the routing table. Add one to the hop count from the path in the routing table.

6. Details of the IP implementation

This section gives a brief description of the packet formats used by cisco's IGRP. IGRP is sent using IP datagrams with IP protocol 9 (IGP). The packet begins with a header. It starts immediately after the IP header.

```
unsigned version: 4;    /* protocol version number */
unsigned opcode: 4;     /* opcode */
uchar edition;          /* edition number */
ushort asystem;         /* autonomous system number */
ushort ninterior;       /* number of subnets in local net */
ushort nsystem;         /* number of networks in AS */
ushort nexterior;       /* number of networks outside AS */
ushort checksum;        /* checksum of IGRP header and data */
```

For update messages, routing information follows immediately after the header.

The version number is currently 1. Packets having other version numbers are ignored.

The opcode is either

1	update
2	request

This indicates the type of message. The format of the two message types will be given below.

Edition is a serial number which is incremented whenever there is a change in the routing table. (This is done in those conditions in which the pseudo-code above says to trigger a routing update.) The edition number allows gateways to avoid processing updates containing information that they have already seen. (This is not currently implemented. That is, the edition number is generated correctly, but it is ignored on input. Because it is possible for packets to be dropped, it is not clear that the edition number is sufficient to avoid duplicate processing. It would be necessary to make sure that all of the packets associated with the edition had been processed.)

Asystem is the autonomous system number. In cisco's implementation a gateway can participate in more than one autonomous system. Each such system runs its own IGRP protocol. Conceptually, there are completely separate routing tables for each autonomous system. Routes that arrive via IGRP from one autonomous system are sent only in updates for that AS. This field allows the gateway to select which set of routing tables to use for processing this message. If the gateway receives an IGRP message for an AS that it is not configured for, it is ignored. In fact cisco's implementation allows information to be "leaked" from one AS to another. However I regard that as an administrative tool, and not part of the protocol.

Ninterior, nsystem, and nexterior indicate the number of entries in each of the three sections of update messages. These sections have been described above. There is no other demarcation between the sections. The first ninterior entries are taken to be interior, the next nsystem entries as being system, and the final nexterior as exterior.

Checksum is an IP checksum, computed using the same checksum algorithm as a UDP checksum. The checksum is computed on the IGRP header and any routing information that follows it. The checksum field is set to zero when computing the checksum. The checksum does not include the IP header, nor is there any virtual header as in UDP and TCP.

6.1 Requests

An IGRP request asks the recipient to send its routing table. The request message has only a header. Only the version, opcode, and asystem fields are used. All other fields are zero. The recipient is expected to send a normal IGRP update message to the requester.

6.2 Updates

An IGRP update message contains a header, followed immediately by routing entries. As many routing entries are included as will fit into a 1500-byte datagram (including IP header). With current structure declarations, this allows up to 104 entries. If more entries are needed, several update messages are sent. Since update messages are simply processed entry by entry, there is no advantage to using a single fragmented message rather than several independent ones.

Here is the structure of a routing entry:

```
        uchar number[3];     /* 3 significant octets of IP address */
        uchar delay[3];      /* delay, in tens of microseconds */
        uchar bandwidth[3];  /* bandwidth, in units of 1 Kbit/sec */
        uchar mtu[2];        /* MTU, in octets */
        uchar reliability;   /* percent packets successfully tx/rx */
        uchar load;          /* percent of channel occupied */
        uchar hopcount;      /* hop count */
```

The fields defined uchar[2] and uchar[3] are simply 16 and 24 bit binary integers, in normal IP network order.

Number defines the destination being described. It is an IP address. To save space, only the first 3 bytes of the IP address are given, except in the interior section. In the interior section, the last 3 bytes are given. For system and exterior routes, no subnets are possible, so the low-order byte is always zero. Interior routes are always subnets of a known network, so the first byte of that network number is supplied.

Delay is in units of 10 microseconds. This gives a range of 10 microseconds to 168 seconds, which seems sufficient. A delay of all ones indicates that the network is unreachable.

Bandwidth is inverse bandwidth in bits per sec scaled by a factor of 1.0e10. The range is from a 1200 bps line to 10 Gbps. (That is, if the bandwidth is N kbps, the number used is 10000000 / N.)

MTU is in bytes.

Reliability is given as a fraction of 255. That is, 255 is 100%.

Load is given as a fraction of 255.

Hopcount is a simple count.

Because of the somewhat wierd units used for bandwidth and delay, some examples seem in order. These are the default values used for several common media.

```
                Delay            Bandwidth
Satellite    200,000 (2 sec)            20 (500 Mbit)
Ethernet         100 (1 ms)          1,000
1.544 Mbit      2000 (20 ms)         6,476
64 Kbit         2000               156,250
56 Kbit         2000               178,571
10 Kbit         2000             1,000,000
1 Kbit          2000            10,000,000
```

6.3 Metric computations

Here is a description of the way the composite metric is actually computed in cisco version 8.0(3).

```
metric = [K1*bandwidth + (K2*bandwidth)/(256 - load) + K3*delay] *
                                          [K5/(reliability + K4)]

If K5 == 0, the reliability term is not included.

The default version of IGRP has K1 == K3 == 1, K2 == K4 == K5 == 0
```