EIGRP–A FAST ROUTING PROTOCOL BASED ON DISTANCE VECTORS

Bob Albrightson
Cisco Systems
Menlo Park, CA 94025
albright@cisco.com

J.J. Garcia-Luna-Aceves
University of California
Santa Cruz, CA 95064
jj@cse.ucsc.edu

Joanne Boyle
Cisco Systems
Menlo Park, CA 94025
boyle@cisco.com

Abstract

Early routing protocols were based on distance vectors; they were very simple and easy to implement but had the severe drawbacks of counting to infinity and routing loops. These problems were reduced using such techniques as split horizon and hold-downs; however, for these techniques to work in practice, long convergence times are introduced. Routing protocols based on link states have been implemented to address the problem of slow convergence in distance-vector protocols, but they add complexity in configuration and troubleshooting. We present a new distance-vector protocol that converges as quickly as current link-state protocols, while maintaining loop freedom at every instant. The protocol is based on three main elements: a transport algorithm that supports the reliable exchange of messages among routers, the diffusing update algorithm, which computes shortest paths distributedly, and modules that permit the operation of the new routing protocol in a multiprotocol environment.

1 INTRODUCTION

Today's intradomain routing protocols can be classified as distance-vector or link-state protocols. In a distance-vector protocol, a router knows the length of the shortest path from each neighbor node to every network destination, and uses this information to compute the shortest path and next router in the path to each destination. A router sends update messages to its neighbors, who in turn process the messages and send messages of their own, if needed. Each update message contains a vector of one or more entries, each of which specifies, as a minimum, the distance to a given destination. In contrast, in a link-state protocol a router must receive information about the entire topology to compute the shortest path to each destination using a local shortest-path algorithm such as Dijkstra's algorithm [1]. Each router broadcasts update messages, containing the state of each of the router's adjacent links, to every other router in the network.

The distance vector protocols used in the Internet thus far are based on variants of the distributed Bellman-Ford algorithm (DBF) for shortest-path computation [1]. The primary disadvantage of DBF is that incorrect entries in routing tables may form routing-table loops for one or more destinations whenever link costs increase [9]. Because a router chooses as its successor to a destination any neighbor router who appears to offer the shortest path to that destination, the router may choose paths that lead to loops for as long as those neighbor routers with viable paths to the destination offer path lengths longer than those paths leading to loops. The worst case of this problem is rather severe: when routers fail or the network partitions, a router can detect such events only after it has considered all possible path lengths to the one or more destinations that have become unreachable through any of its neighbors. Accordingly, this is referred to as the *counting-to-infinity problem.* Intradomain routing protocols approach this problem by requiring that a router employ one or more of the following techniques:

- Telling its neighbor routers whether it uses them as successors to destinations (a technique called *split horizon* [3] [12] [14] [15]
- Holding down the updating of its routing table for some period of time after detecting distance increases [2]
- Sending update messages reporting an infinite distance to a destination in order to force neighbor routers to choose successors other than the router itself [15]

However, none of these techniques solves DBF's performance problems satisfactorily in very large internets, because exceedingly long convergence times can be incurred.

Because of the slow convergence of such distance-vector protocols as RIP (routing information protocol) [7], link-state protocols (e.g., OSPF [11]) are considered by some as the only viable approach to routing in large internets. This paper introduces a distance-vector protocol for intradomain routing that is far superior than traditional distance-vector protocols. The new protocol is called the Enhanced Interior Gateway Routing Protocol (EIGRP). The basis of its operation is the Diffusing Update Algorithm (DUAL) [4, 5], which is used to compute shortest paths distributedly and without ever creating routing-table loops or incurring counting-to-infinity behavior.

DUAL has been shown to be free of routing-table loops at every instant, regardless of the type or number of changes in the network, and to converge to correct routing-table values within a finite time after the occurrence of an arbitrary sequence of link-cost or topological changes [5]. Because DUAL is loop free at every instant, it does not have any hop-count limitation, which is a necessity in RIP. Simulation studies [16] have shown that DUAL's average performance after link-cost changes or topology changes is far better than DBF, which is used in RIP, and similar to the performance of an ideal link-state algorithm with much less CPU overhead. Furthermore, more recent simulation results [6] show that using hierarchical routing as first proposed by McQuillan [10] in DUAL outperforms OSPF. However, using DUAL in a routing protocol requires

- Neighbor discovery and recovery mechanisms
- A reliable transport mechanism used to exchange update messages among routers that guarantees ordered delivery
- Protocol dependent modules that enable its operation in a multiprotocol environment

EIGRP, like IGRP [8] represents distances as a composite of available bandwidth, delay, load utilization, and link reliability; it is designed to be network-layer protocol independent, thereby allowing it to support multiple network-layer protocol suites.

The rest of this paper provides an overview of EIGRP and describes each of its four components in detail.

2 OVERVIEW OF EIGRP

EIGRP's link (neighbor[1]) discovery and recovery is the mechanism that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. This is achieved with low overhead by having each router send small Hello packets periodically. As long as Hello packets are received, a router can determine that a neighbor is alive and functioning. Once this is determined, the neighboring routers can exchange routing information.

The reliable transport mechanism of EIGRP is responsible for guaranteed, ordered delivery of packets to all neighbors. It supports inter-mixed transmission of multicast or unicast packets. Some packets must be transmitted reliably and others need not. For efficiency, reliability is provided only when necessary. For example, on a multi-access network that has multicast capabilities, such as Ethernet, it is not necessary to send Hellos reliably to all neighbors individually. So a single multicast Hello is sent with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets, such as Updates, require acknowledgment and this is indicated in the packet. The reliable transport has a provision to send multicast packets quickly when there are unacknowledged packets pending. This helps insure that convergence time remains low in the presence of varying speed links.

The Diffusing Update Algorithm (DUAL) implements the decision process for all route computations. It tracks all routes advertised by all neighbors. The distance information, known as a metric, is used by DUAL to select minimum cost loop free paths. DUAL selects routes to be inserted into a routing table based on feasible successors. A successor is a neighboring router used for packet forwarding that has a least cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors but there are neighbors advertising the destination, a recomputation must occur. This is the process where a new successor is determined. The amount of time it takes to recompute the route affects the convergence time. Even though the recomputation is not processor intensive, it is advantageous to avoid recomputation if it is not necessary. When a topology change occurs, DUAL will test for feasible successors. If there are feasible successors, it will use any it finds in order to avoid any unnecessary recomputation. From router i's standpoint, a feasible successor toward destination j is a neighbor router k that satisfies a *feasibility condition* that is a function of the router's own distance and its neighbor's distance to the destination.

The protocol dependent modules are responsible for network layer protocol specific requirements. For example, the IP module is responsible for sending and receiving DUAL packets that are encapsulated in IP. The IP module is responsible for parsing packets and informing DUAL of the new information received. The IP module asks DUAL to make routing decisions and the results of which are stored in the IP routing table. It is also responsible for leaking destinations learned by other IP routing protocols, as well as filtering and summarization.

[1] *link* and *neighbor* are used interchangeably throughout this paper

3 LINK DISCOVERY

Link (neighbor) discovery and dead detection is used in determining when links appear and disappear. The transport mechanism uses the link discovery and dead detection, while dual only uses the dead detection case.

Small hello messages are sent periodically. They contain the hold time for the link and factors used in the metric computation. The latter is used only as verification of consistency. When a router receives a hello, it sets a timer to expire after the advertised hold time interval. Each time a hello is received, the timer is reset.

A link is discovered when the first hello packet is received. The state of this link (up) is recorded and a request is made of DUAL to send a full update to the link. When the transport is requested to send multicasts, it uses the link information to determine from whom to expect acknowledgements.

When the hold timer expires, the link is declared down, and dual is informed with a link-down event. Other link-down events are triggered by interfaces going down, a maximum packet retransmission threshold exceeded, and various configuration changes that would disable routing of one or several links.

DUAL's treatment of a link-down event is described below.

4 DUAL

DUAL is specified in detail in References [4, 5]; this section provides only a summary of its operation as it applies to EIGRP.

Each router maintains a vector with its distance to every known destination in the routing table. Routing information is exchanged only between neighbors by means of update messages; this is done after routers detect changes in the cost or status of links. Each update message contains a distance vector of one or more entries, and each entry specifies the length of the selected path to a given destination, as well as an indication of whether the entry constitutes an update, a query, or a reply to a previous query. A router also maintains a topology table (also called distance table [5]) containing the distance reported by selected neighbor routers to each known destination. Information for the routing table is taken from the topology table. In addition, a link-state table is maintained that contains a list of all neighbors heard. This is used by the transport as well as DUAL. There are two other tables that are used. They are the query-origin table, and the reply-status table. These are described below.

For a given destination, a router updates its routing table differently depending on whether it is *passive* or *active* for that destination. A router that is passive for a given destination can update the routing-table entry for that destination independently of any other routers, and simply chooses as its new distance to the destination to be the shortest distance to that destination among all neighbors, and as its new successor to that destination to be any neighbor through whom the shortest distance is achieved. In contrast, a router that is or becomes active for a given destination must synchronize the updating of its routing-table entry with other routers. A router is active if it is waiting for at least one neighbor to send a reply to a query already sent by the router, and is passive otherwise. Furthermore, a router is initialized in passive state for all known destinations with a 0 distance to itself and a finite distance to other destinations that are directly attached to an adjacent link. Passive destinations with infinite distances are removed from the topology table.

When a router is passive and needs to update its routing table for a given destination j after it processes an update message from a neighbor or detects a change in the cost or availability of a link, it tries to obtain a *feasible successor*. From router i's standpoint, a feasible successor toward destination j is a neighbor router k that satisfies the feasibility condition given by the following two equations:

$$D_j = D_{jk} + c_k = Min\{D_{jp} + c_p \mid p \;\; is\ a\ neighbor\}$$

$$D_{jk} < FD_j$$

where:
D_j = current distance from router i to destination j
D_{jk} = distance to destination j reported by neighbor k as known by router i
c_k = cost of the link to neighbor k as known by router i
FD_j = *feasible distance* for destination j, and equal to the minimum value obtained for D_j since the last time router i transitioned from active to passive state for destination j.

If router i finds a feasible successor, it remains passive and updates its routing table entry as in DBF [1]. Alternatively, if router i cannot find a feasible successor, it first sets its distance equal to the addition of the distance reported by its current successor plus the cost of the link to that neighbor. The router also sets its feasible distance equal to its new distance. After performing these updates, the router becomes active by sending a query in an update message to all its neighbors; such a query specifies the router's new distance through its current successor. It then sets the destination's reply-status table entry for each link to one, indicating that it expects a reply from each neighbor for that destination.

Once active, a router cannot change its successor, its feasible distance, the value of the distance it reports to its neighbors, or its entry in the routing table, until it receives all the replies to its query. A reply received from a neighbor indicates that such a neighbor has processed the query and has either obtained a feasible successor to the destination, or determined that it cannot reach the destination. Once Node i obtains all the replies to its query, it computes a new distance and successor to destination j, updates its feasible distance to equal its new distance, and sends an update to all its neighbors.

Multiple changes in link cost or availability are handled by ensuring that a given node is waiting to complete the processing of at most one query at any given time. The mechanism used to accomplish this is specified in [5], and is such that a node can be either passive or in one of four active states. These four active states are kept track of in the query-origin table:

- $o_j^i = 0$ indicates that node i becomes active without receiving a query from its successor, and it has experienced at least one distance increase after becoming active.
- $o_j^i = 1$ indicates that node i becomes active after processing an input event other than a query from its successor, and experiences no distance increase and receives no query from its successor while it is active.
- $o_j^i = 2$ indicates that node i has received a query from its successor and has detected at least one more input event that caused its distance to increase.
- $o_j^i = 3$ indicates that node i becomes active by processing a query from its successor, and experiences no distance increases after becoming active.

When node i transitions from active to passive state and $o_j^i = 1$ or 3, then it simply chooses as its successor a neighbor that offers a shortest path. In contrast, when node i becomes passive and $o_j^i = 0$ or

2, it uses the value of FD^i_j at the time it became active to determine whether or not a neighbor satisfies the feasibility condition before computing D^i_j and s^i_j. Hence, node i processes any pending update or distance increases that occurred while it was active.

Ensuring that updates will stop being sent in the network when some destination is unreachable is easily done. If node i has set $D^i_j = \infty$ already and receives an input event (a change in cost or status of link $(i,\ k)$, or an update or query from node k) such that $D^i_{jk} + l^i_k = \infty$, then node i simply updates D^i_{jk} or l^i_k, and sends a reply to node k with $RD^i_j = \infty$ if the input event is a query from node k. When an active node i has an infinite feasible distance and receives all the replies to its query such that every neighbor offers an infinite distance to the destination, the node simply becomes passive with an infinite distance.

When node i establishes a link with a neighbor k, it updates the value of l^i_k and assumes that node k has reported infinite distances to all destinations and has replied to any query for which node i is active. Furthermore, if node k is a previously unknown destination, node i sets $o^i_k = 1$, $s^i_k = null$, and $D^i_k = RD^i_k = FD^i_k = \infty$. Node i also sends to its new neighbor k an update for each destination for which it has a finite distance.

When node i is passive and detects that link $(i,\ k)$ has failed, it sets $l^i_k = \infty$ and $D^i_{jk} = \infty$. After that, node i carries out the same steps used for the reception of a link-cost change in the passive state.

Because a router can become active in only one diffusing computation per destination at a time, it can expect at most one reply from each neighbor. Accordingly, when an active node i loses connectivity with a neighbor n, node i can set $r^i_{jn} = 0$ and $D^i_{jn} = \infty$, i.e., assume that its neighbor n sent any required reply reporting an infinite distance. If node n is s^i_j, node i also sets $o^i_j = 0$. When node i becomes passive again and $o^i_j = 0$, it cannot simply choose a shortest distance; rather, it must find a neighbor that satisfies the feasibility condition using the value of FD^i_j set at the time node i became active in the first place.

Figures 1 and 2 present simple example networks illustrating the loop-freedom property of DUAL's operation; they focus on the routing decisions made by the nodes of such networks regarding destination node a. In the example of Figure 1, all links are assumed to have unit cost; the example in Figure 2 shows a network with links that have costs different than 1 and are assumed to be the same in both directions of the links. Arrowheads on the links indicate the choice of successor for the node toward node a. When link (d, b) fails, node d has no neighbor that has reported a distance smaller than 2, its feasible distance, and must send a query to all its remaining neighbors. When node e processes node d's query, it finds that node c has reported a distance to a equal to 2, which is smaller than its feasible distance of 3; accordingly, node e sends a reply to node d reporting a distance of 3 and adopts node c as its new successor. Node d is able to adopt node e as its new successor to node a as soon as it processes node e's reply.

In the example shown in Figure 2, the costs of the links are such that, when link (d, b) fails, node d is again forced to send a query to its neighbors during Step B when it is unable to find a neighbor with a reported distance smaller than its feasible distance of 2. Node e must forward the query during Step C because its feasible distance is 3 and none of its neighbors is feasible (i.e., has reported a distance smaller than 3); during the same step, node c sends a reply to node d with a distance of 3. During step D, nodes d and c both send a reply to node e's query; node c does it because it has a feasible successor (its current successor) and node d sends its reply because it is active and node e is not its current successor. Once node e processes the replies from its two neighbors, it sends its own reply to node d and chooses node c as its new successor and obtains a distance of 4. During Step F, node d processes the last reply it needs and is able to choose node c as its new successor and obtain a distance of 4 too.

5 TRANSPORT MECHANISM

The transport mechanism reliably delivers a mix of multicasts and unicasts in the order requested, and is also capable of sending datagrams unreliably. It has a window size of one, i.e., it only allows one outstanding packet per link (neighbor) when transmitting multiple packets.

The complexity of the transport mechanism lies in its ability to deliver multicasts and unicasts in order to all receivers, while delivering multicast packets even when a link is unable to quickly respond to unicasts. To achieve this, the transport will send out a special (unreliable) packet that lists all links that are not permitted to receive the next multicast packet. When a router receives this special packet, and it is not in the list, it will enter what is known as conditionally receive mode. It will not throw away the next multicast packet. The multicast packet that follows has a bit set in the header, called the conditionally receive bit (CR bit). If this bit is set, and the receiver is in conditionally receive mode, it will accept the multicast, otherwise it will throw it away. On the other hand, if the CR bit is not set, all receivers will unconditionally receive and acknowledge the multicast packet.

When reliable multicasts are sent, the sender expects an acknowledgement from every neighbor. The state of every neighbor is stored in the link-state table. If a retransmission must occur, it is sent as a unicast. Although a scheme based on multicasting retransmissions could have been used, that would have increased the complexity of the protocol, and would have also increased the load on receivers that have already acknowledged the packet and forced them to needlessly return another acknowledgement that would be ignored anyway.

As stated previously, a link-down event is sent to DUAL when hello packets are no longer received or when the user changes the configuration such that one or more links are no longer available for exchanging routing information. In addition, the transport mechanism will generate a link-down event when the retransmission count for any given datagram exceeds a preselected threshold. This seems to fix the problem case where a router can hear its neighbor's hello packets but they cannot hear the router. An alternative could be to implement a state machine based on a three way handshake before a link is declared up, but this seemed to be needlessly complex.

In order to compute a reasonable retransmission interval, a round trip estimator had to be implemented. We decided to use a similar approach to the one used in TCP [13].

6 PROTOCOL-DEPENDENT MODULES

The network-layer protocol-dependent modules (PDM) are responsible for initialization, building and decoding packets, interfacing dual to the routing table, providing protocol-dependent support routines for DUAL, and configuration and runtime command support.

Initialization is of course done when a process is started. It handles setting up the fixed data block where entries such as function jump tables, and global DUAL state are stored. It also sets up various queues for packets going into and going out of the system as well as internal processing queues.

There are four generic packet types supported. They are update, query, reply, and hello. DUAL has

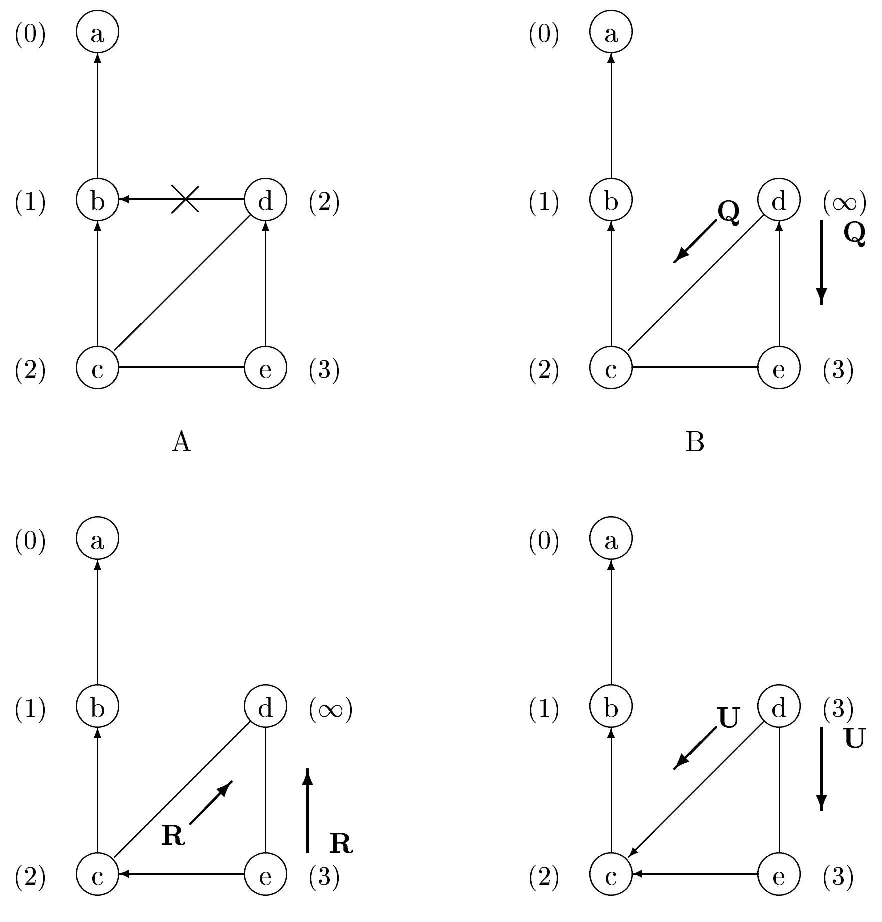

Figure 1: Diffusing Computation

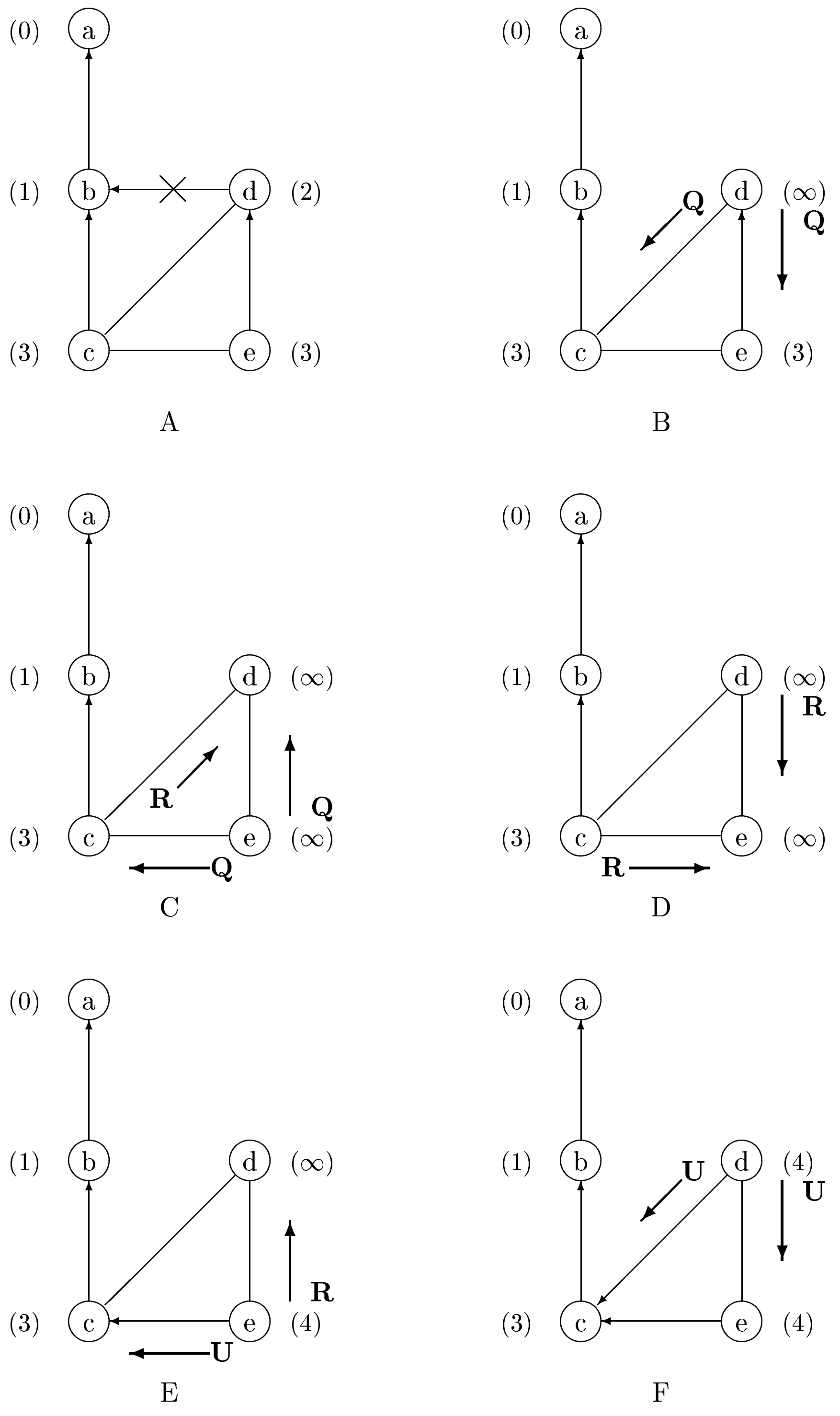

Figure 2: Diffusing Computation

no knowledge of packets. Therefore it is up to the PDM to handle the encoding and decoding of the various packets. When an update, query, or reply is received, the PDM pulls out the type (U, Q, or R), destination, source of the information, metric and any external data (if it is an external destination) computes the metric and composite metric and passes this along to DUAL. When hello packets are received they are passed along, as is, to the link discovery/recovery protocol engine.

The protocol-dependent modules handle the interface between the routing table and DUAL. When DUAL wants to insert a destination into the routing table, it calls the PDM which then either accepts or rejects the destination based filtering or its own redistribution rules.

The routing table can also insert information into the topology table. This sort of action occurs when routes are being redistributed (leaked) from another routing protocol. This also occurs when an interface that is considered "connected" to the router from DUAL's point of view, either comes up, or is newly configured. When either of these events occurs, DUAL is notified and marks the entry as connected or redistributed. In the redistributed case, there is usually an external data block generated by the PDM that is then tagged in the topology table. This block contains information such as the routing protocol and metric where the destination originated, the ID of the redistributing router, an arbitrary configurable tag, and some miscellaneous flags. When destinations tagged with external data are sent to neighboring routers, the external information rides along as well.

The router ID field in the external data is used to determine if the destination should be ignored. The router, when receiving an external destination, compares the router ID field against its own router ID. If they match, the destination is ignored. This is used to prevent routing loops between routing domains that use dissimilar protocols, thereby making it impossible to compare the metrics. The same problem recurs when multiple routers span the boarder between the two domains. To get around this the tag field is used. The network administrator can configure an arbitrary tag for all destinations comming from a particular protocol. Then, if an external destination is received with the configured tag, a filter configured to ignore anything with that tag will prevent the loop.

Another looping situation can occur if a router chooses an external DUAL route over an internal. If a given destination is in both an internal and external routing domain, the metric coming from the external domain may not be accurate with respect to its actual cost through the external path. To avoid looping in this case, EIGRP always chooses an internal path over an external one regardless of the metric.

Because DUAL is a protocol-independent algorithm, it must rely on the PDM to provide some support routines. Some of these include network and address comparison, copying, and printing, and hash bucket calculation. There are also some utility functions in the link discovery code that handle the assignment and lookup of neighbor identifiers that are used by DUAL in the reply-status table.

6.1 Metric handling

Metric comparison between EIGRP and other routing protocols, and the initial setting of the metric in the router where the destinations are leaked from a dissimilar protocol into DUAL, are handled differently between the various network-layer protocols. This is due to differing requirements in the user community. In IP, a model already existed for leaking destinations between dissimilar routing protocols. Metric comparisons between dissimilar IP routing protocols is quite simple; it is just is not done.

IPX does automatic redistribution between RIP and EIGRP. When leaking destinations from IPX RIP into EIGRP a fixed bandwidth is used at the redistribution point between RIP and EIGRP. The redistributed delay is based on the delay field[2] passed around in the IPX RIP packet. Using the RIP ticks value allows some differentiation in metrics when two redistributed paths are compared. When a destination is redistributed from RIP to EIGRP it's RIP hopcount and delay are preserved.

When leaking destinations from EIGRP into RIP, the original RIP hop count (the EIGRP external hop count) is incremented and used as the new RIP metric. This allows an IPX network to expand beyond the 16 hop limitation of RIP. To accommodate data packets which in IPX will be dropped after 15 hops, the router uses a special algorithm which allows it to only increment the transport control field of the IPX packet once for each 16 hops it transverses. The resultant maximum width of the network when running EIGRP is 224 hops.

When IPX EIGRP compares different router advertisements to determine which to chose, it always prefers an Internal EIGRP route over an External one regardless of EIGRP metric. This prevents looping do to non-symmetric metric translation during automatic redistribution. When deciding between a RIP and an EIGRP route, the router generally prefers the EIGRP route.

In Appletalk, when redistributing destinations from RTMP into DUAL, the RTMP hop count is multiplied by a constant and used as the EIGRP bandwidth. The interface delay value is used as the EIGRP delay. When selecting between an RTMP route and a DUAL route for the same destination, the same RTMP to DUAL translation is done, and the composite metrics are compared directly. The lower resultant metric will be chosen except in the case where the EIGRP hop count exceeds the RTMP hopcount. In that case the RTMP route will be chosen regardless of EIGRP metric,

In IP, metric handling when leaking routes from other routing protocols is handled differently depending on the source of information. If the destinations are leaked from other than IGRP or EIGRP, the metric is taken from a user configured default metric. If there is no configured metric and the source of information is static, then it will take the metric from the associated interface. Because IGRP and EIGRP have a similar metric, they are effectively translatable. When choosing a path between dissimilar routing sources for a given destination, the notion of administrative distance is used. Administrative distance is checked first. If the distances are different, it picks the lower of the two. If they are equal, it assumes the sources are from similar routing protocols and compares the metrics directly.

In all cases leaking destinations from DUAL into these other dynamic protocols is done in a symmetrical fashion.

6.2 Other PDM tasks

There are other protocol-dependent tasks that the PDM must provide. Among these are route summarization (i.e, aggregation), which is a means of collapsing contiguous chunks of address space in order to conserve routing table space. This is currently only supported in the IP PDM.

The IPX PDM provides incremental SAP updates to it's EIGRP neighbors, thus conserving substantial network bandwidth. This is possible because of EIGRP's reliable transport mechanism.

Another task of the PDM is filtering. The user has the option to filter destinations on input or output. However, filtering doesn't always mean a complete absence of input or output information for any given destination that is filtered. For example, when a query is received for a destination that is filtered on output, a reply must still be sent. The only difference between filtered and unfiltered destinations is that

[2]This delay field is known as ticks

in the filtered case, the metric is set to infinity regardless of the actual reported distance (RD).

7 CONCLUSIONS

EIGRP is the first internet routing protocol that provides loop-freedom at every instant and convergence times comparable to those obtained with standard link-state protocols. Furthermore, EIGRP provides multiple paths to every destination that may have different weights.

As of this writing, EIGRP is deployed in part of cisco System's engineering network. It has shown that, as expected, it does provide loop freedom and quick convergence in a medium-scale network. In addition, many sites have run EIGRP in a laboratory environment. Brian Jemes at Hewlett Packard has simulated the HP backbone and injected thousands of destinations into the topology yielding good results. Other sites have, on not as grand a scale, also shown that EIGRP stands up to the punishment one expects in a real-world internetwork.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Second Edition, Prentice-Hall, 1992.

[2] L. Bosack, "Method and Apparatus for Routing Communications among Computer Networks," U.S. Patent assigned to Cisco Systems, Inc., Menlo Park, California, February 1992.

[3] T. Cegrell, "A Routing Procedure for the TIDAS Message-Switching Network," *IEEE Trans. Commun.*, Vol. COM-23, No. 6, June 1975, pp. 575-585.

[4] J.J. Garcia-Luna-Aceves, "Method and Apparatus for Distributed Loop-Free Routing in Computer Networks," U.S. Patent application, SRI International, Menlo Park, California, 1993.

[5] J.J. Garcia-Luna-Aceves, "Loop-Free Routing Using Diffusing Computations," *IEEE/ACM Trans. Networking*, Vol. 1, No. 1, February 1993.

[6] J.J. Garcia-Luna-Aceves and W.T. Zaumen, "Area-Based, Loop-Free Internet Routing," *Proc. IEEE INFOCOM 94*, Toronto, Ontario, Canada, June 1994.

[7] C. Hedrick, "Routing Information Protocol," RFC 1058, June 1988.

[8] C. Hedrick, "An Introduction to IGRP," Rutgers - The State University of New Jersey Technical Publication, Laboratory for Computer Science Research, August 1991.

[9] J.M. Jaffe and F.M. Moss, "A Responsive Routing Algorithm for Computer Networks," *IEEE Trans. Commun.*, Vol. COM-30, No. 7, July 1982, pp. 1758-1762.

[10] J. McQuillan, "Adaptive Routing Algorithms for Distributed Computer Networks," BBN Rep. 2831, Bolt Beranek and Newman Inc., Cambridge MA, May 1974.

[11] J. Moy, "OSPF Version 2," *Internet Draft*, November 1992.

[12] W.E. Naylor "A Loop-Free Adaptive Routing Algorithm for Packet Switched Networks," *Proc. Fourth Data Communications Symposium*, Quebec City, Canada, October 1975, pp. 7.9-7.15.

[13] J.B. Postel, “Transmission Control Protocol,” RFC 793, September 1981.

[14] K.G. Shin and M. Chen, “Performance Analysis of Distributed Routing Strategies Free of Ping-Pong-Type Looping,” *IEEE Trans. Computers*, Vol. COMP-36, No. 2, February 1987, pp. 129-137.

[15] T.E. Stern, “An Improved Routing Algorithm for Distributed Computer Networks,” *IEEE International Symposium on Circuits and Systems*, Workshop on Large-Scale Networks and Systems, Houston, TX, April 1980.

[16] W.T. Zaumen and J.J. Garcia-Luna-Aceves, “Dynamics of Link-State and Loop-Free Distance-Vector Routing Algorithms,” *Journal of Internetworking*, Wiley, Vol. 3, December 1992, pp. 161-188.