IDN-8

Routing in Next Generation Gateway

Eric Rosen

October 1983

This note just consists of some thoughts on routing in the next generation of gateway. It is preliminary to writing a functional spec, and even more preliminary to writing a design document.

In the current gateway system, information gathered by EGP about exterior networks is merged directly into the interior (GGP) routing updates, and GGP is the only means whereby one gateway informs another of the connectivitiy of exterior nets. To my mind, this is a mere expedient; looking back at my original paper on "autonomous systems", I see that I never intended this sort of merging of the exterior and interior information. Rather, I had in mind that totally differnet mechanisms could be used to disseminate the two different kinds of information. Interior routing information would be disseminated in much the manner in which it is done in the ARPANET, and exterior information in a different manner which would have lower overhead and be less responsive.

The interior routing still does need to know how to reach an exterior net. My model of how exterior nets appear to the interior routing algorithm is the following. If interior gateway IG has an exterior neighbor (direct or indirect) EG, such that EG is an appropriate first hop to net N, then the interior routing algorithm should model net N as being directly connected to gateway IG. That is, the interior routing algorithm does not need to know anything about exterior gateway EG; it only needs to know that to get a packet to net N, it is sufficient to route it through interior gateway IG. Of course, IG itself needs to know that the packet has to go through yet another (exterior) gateway, but as far as the routing algorithm is concerned, IG is the exit gateway, just as if IG had an interface directly on net N.

I propose to use some form of SPF routing. This implies that each gateway must create updates showing its connectivity, and disseminate these updates to ALL other gateways in the same autonomous system. Please remember that this is ALL other gateways, not just neighboring gateways. To perform SPF routing, each gateway must be able to form (through the receipt of updates from all other gateways) an undirected graph representing the network topology. Then each gateway runs Dijkstra's algorithm in order to compute a spanning tree which gives the shortest path from that gateway to each other gateway. From this tree one can derive a "routing table", which is just a map from the ultimate destination of a packet to the packet's next hop.

The current gateway system uses a form of routing which we can call "network-directed", in that the routing table maps the network number of a packet's IP destination address into the number of the next network that the packet needs to traverse in order to reach its destination netowrk. The routing algorithm in a particular gateway tells that gateway only which network a packet must be sent to next. If the gateway has several neighbors on that next net, the routing algorithm does not choose among them; some other means of choosing must be developed.

An alternative would be "gateway-directed" routing, in which the "entry gateway" of an autonomous system would map a packet's IP destination address into an "exit gateway", and the routing tables in all the gateways would map the exit gateway into the next gateway that the packet needs to traverse on its way to the exit gateway. If there are several networks connecting the two neighboring gateways, the routing algorithm leaves the choice of network as a local decision.

Let's look at what would be required to do network-directed routing. The routing updates from each gateway would have to contain the id numbers of the networks to which that gateway is interfaced. This is all that is needed to enable all the gateways to form their network connectivity graph. These updates would be quite small, in general, since gateways will rarely have more than a few interfaces. Suppose a gateway receives a packet destined for net N. It looks up net N in its routing table, and sees that the next hop for that packet is net M. It then needs to know which of its neighbors is connected to net M, as well as how to address that neighbor over the network to which they are both connected. With this information, it can forward the packet to its next hop. If there are several neighbors with interfaces on net M, it needs to make a choice.

Note that, since our routing updates are purely internal to our autonomous system, if we think we may never have more than 256 networks in one system, we can define a mapping of network numbers into a single byte, and carry around this short, private identifier in the routing updates, instead of the longer class B or C numbers. This would allow some savings in overhead.

Gateway-directed routing is somewhat more complicated. First, the entry gateway (first gateway in the autonomous system to see the packet) must choose an exit gateway. To do this, it must be able to determine which networks each other gateway is connected to. This implies that the routing updates created by a gateway must contain the id numbers of each network that the gateway is connected to, just as for network-directed routing. However, gateway-directed routing requires considerably more information. The routing updates created by a given gateway also need to provide a list of neighboring gateways. Neighboring gateways do not have to be identified by internet address, however, but merely by an assigned number (which could probably be just 1 byte long).

What are the relative advantages and dis-advantages of each kind of routing. Network-directed routing has a number of apparent advantages:

1 - Routing updates are smaller, since a given gateway will generally have interfaces to only two, three, or four networks, but may have dozens of neighbors.

2 - Unless we want gateways to have to fragment and reassemble routing updates, the minimum maximum packet size of 128 bytes may put an untenable restriction on the number of neighbors a gateway can have, if gateway-directed routing is used.

3 - Gateway-directed routing requires the entry gateway to choose an exit gateway, and by implication, to communicate this choice to all intermediate gateways along the path to the exit gateway. But the internet protocol provides no means for gateways to communicate this information to each other. Thus transit traffic which is travelling between gateways must carry additional header information which is not necessary if network-directed routing is used.

However, network-directed routing also has an important disadvantage. There are situations in which one wants to route to a particular exit gateway, rather than just to a destination net. For example, we want to be able eventually to accomodate the situation in which a given host on net N is reachable from one net N gateway but not from another (partitioned net), and network-directed routing is poorly suited to handling this sort of situation. It is also easy to imagine other situations in which routing towards a particular exit gateway is useful. This might be useful eventually for load sharing among exit gateways. It would certainly be useful if we went to a more connection-oriented internet protocol. I think this is a real possibility in the DoD world, if there is a trend to more stuff like access control, which is inherently connection-oriented. Certainly, if we ever have a commercial gateway product, it is more than a possibility.

We want our routing algorithm to be able to do more than just produce the route that spans the least number of "hops", whether that means the least number of gateways or the least number of networks. We would like to have the capability of dynamically assigning a "distance" to each hop, e.g., the delay a packet would experience when making that hop. Network-directed routing is poorly suited to this, because it provides no way to indicate that the delay (or whatever other metric of distance we may choose) incurred travelling from one network to a neighboring network may be very different, depending on which pair of gateways is actually used. In particular, it is poorly suited to adapting to the situation in which a particular gateway, rather than a particular network itself, is a bottleneck. Gateway-directed routing, on the other hand, is well-suited to adapting to bottlenecks which may either be gateways or networks; if a network is generally a bottleneck, this can be made to show up as a large distance between every pair of gateways on the network.

Of course, in order to provide such a function, the routing updates sent by a given gateway need to indicate not only which other gateways are neighbors, but also what the "distance" to each neighbor is.

There should be no problem with having fixed distances in our initial implementation, and switching later to dynamically measured distances. The way the distance is computed is transparent to the rest of the routing algorithm, and to all other gateways except the one doing the computing.

One of the functions we definitely want our routing to be able to handle is that of providing different routes for different types of service. In particular, we would like to be able to provide a type of service which prevents certain types of traffic from traversing certain networks, or from traversing certain networks unless there is no alternative path. A routing algorithm which is based on laying a minimum spanning tree over a complete picture of the internet topology makes this service easy to provide, since the spanning tree can be computed on the (factually false) assumption that the prohibited network is unreachable, thereby causing data to be routed as if that network did not exist. It is a simple matter to compute a number of spanning trees, under a variety of assumptions, and use different ones for routing different service classes. (Of course, we should never compute a spanning tree under the assumption that some really unreachable network or gateway is up.)

This is a function which can be easily handled with network-directed routing. However, it is also easily handled with gateway-directed routing, PROVIDED that the routing updates sent by a given gateway indicate not only its neighbors, but also the network(s) (if any) over which each neighbor is a neighbor.

Thus far, this is the only requirement for having the routing updates of a gateway-directed routing scheme indicate the network with respect to which a given neighbor is a neighbor.

A related feature which is easily handled with gateway-directed routing, but not so easily with network-directed routing is that of having certain types of service be prohibited from using certain gateways. One can imagine potential applications of this facility.

We have not yet shown any requirement to have the full internet addresses of the neighbors appear in the routing updates. Since this is not required, it becomes easy to accomodate the case of two gateways connected over a direct line (like UCL and RSRE are now) without having to assign a network number to a purely fictitious network.

(Of course, this only means that we don't need the "null network" number for the purposes of routing. We still would have to consider how to address a gateway which is connected to another via a direct line. Perhaps each autonomous system should have a network number of its own to be used for direct gateway addressing. Then address 193.193.193.1 (for example) might be gateway number 1 in a particular system. If this network number is advertised via EGP, then any host in the internet can address any of our gateways by number, without having to know the internet address of any of its interfaces. We might consider this a first step towards logical addressing.)

It is probably a good idea, though, for the routing updates to carry the internet addresses of the neighbors (if the neighbors have internet addresses), even if this is not strictly necessary.

Note that if we do this properly, it is easy to allow routing to a destination which is on a net that we do not wish to use as a transit net. We could, for example, compute a spanning tree on the (factually false) assumption that a particular net N gateway has no reachable neighbors, but is still reachable itself (i.e., has incoming but not outgoing pathways). This allows traffic for net N to be routed to it as an exit gateway, but prevents traffic for other nets from being sent through net N. This is a function which would be difficult to handle with network-directed routing, but is easily handled with gateway-directed routing.

Another type of service we ought to be able to handle with our routing is "expressway" service. The main problem here is defining just what the requirements of this service are. Perhaps when the entry gateway sees a packet which requires a certain type of service, we need to give it some sort of special handling like the following:

a) choose a gateway on the wideband network, and send the packet to that gateway along a route which does not include the network from which the packet arrived at the entry gateway

b) when the packet arrives at that gateway on the wideband net, send it over the wideband net to another wideband net gateway (presumably the one which will result in the shortest trip from the wideband net to the ultimate destination)

c) when it gets to that second wideband net gateway, route it to its destination in the ordinary manner.

I'm not sure whether a peculiar type of service like this is really needed for anything, but the gateway-directed routing seems general enough to be able to handle it.

(Note that some of the features which seem well suited to gateway- directed routing could be handled by network-directed routing if a gateway address is placed in the source route option field of the IP header. But if the gateways add stuff to the option field, then they are modifying the user's header, thereby eliminating one of the advantages which network-directed routing is supposed to have over gateway-directed routing.)

Routing based on type of service will require that the routing updates sent by a gateway not only list its neighbors, and a distance to each neighbor, but perhaps a different distance to each neighbor for each type of service. If the reported distances are the result of some sort of measurement, it is obvious that they need to be carried in the routing updates. But even if they are fixed, tabled values, it is still best to carry them around in the routing updates. Then the distance between gateways G1 and G2 need only be tabled in the config area of those two gateways; all other gateways learn about it from the routing updates. This minimizes the configuration control problems, and provides a flexible scheme which makes it relatively easy to experiment with either tabled or measured distances.

Another function which is much more easily handled with gateway-directed routing than with network-directed routing is that of incorporating a small number of selected service hosts (such as the NIC, name servers, etc.) into routing by treating them as singly-connected gateways. That is, these hosts would never send routing updates, but some gateways on their nets would send updates implying that these hosts are neighboring gateways. This would enable the routing algorithm to determine whether the special hosts are up or down, enabling the quickest possible feedback to oter hosts. If several different name servers, for example, are given the same identifier in the routing algorithm, we can easily implement a simple form of logical addressing wherein our routing algorithm directs certain specially addressed packets to the closest name server. Gateway-directed routing gives us this capability without much extra cost; some day we'll probably find a use for it.

In general, gateway-directed routing is more natural and suitable if we want to think of an autonomous system of gateways as a single integrated network, with source and destination hosts accessing it via entry and exit gateways. Everything that can be done with network-directed routing can also be done with gateway-directed routing, but the latter gives us more control over the routing and allows the incorporation of additional functions in a unified (non-kludgy) manner. Thus I would recommend the use of gateway-directed routing. However, this does bring along with it the difficulties I have cited above. However, I think these difficultes can be solved (though not without some effort.)

One difficulty is the large size of updates which would have to be sent from gateways which have lots of neighbors, such as the mail bridges. This could result not only in a large amount of routing overhead, but could even push us up against the minimum maximum packet size limit. (Remember that every update must go to every gateway, which means that even packet radio gateways must be able to receive updates from the ARPANET-MILNET gateways.) I think that under EGP, these gateways will probably have fewer (interior) neighbors than they do at present. Many of their current neighbors are dumb gateways which would become exterior neighbors, and hence would not be mentioned in the interior routing updates. Other neighbors, while they are macro-11 gateways, are connected to stub nets or to systems of nets with no other connetion to the ARPANET or MILNET, and hence could be relegated to other autonomous systems and made into exterior neighbors. Even so, it is possible that some gateways will have a large number of neighbors. However, it is possible to reduce the routing overhead by using a concept which we might call the "next door neighbor" concept.

On many networks, the relationship "is a neighbor of" is transitive; if A is a neighbor of B, and B is a neighbor of C, then A is also a neighbor of C. (Packet radio networks might be the only kind of net in which this doesn't apply.) We can make use of this fact in the routing algorithm by having each gateway report, in its routing updates, only a subset of its neighbors (its "next door neighbors".) By knowing which gateways are next door neighbors on which nets, the routing algorithm could figure out which gateways are really neighbors on those nets. Thus relatively small updates with only a few neighbor entries could carry information from which one could determine the true interconnectivity of the gateway system.

Note that this procedure does NOT amount to assuming that every gateway with an interface to net N is a neighbor of every other such gateway. Rather, if net N partitions, the procedure makes it possible to determine which of the net N gateways are still neighbors of each other, and which are not.

This same sort of technique could be used to reduce the overhead associated with the "neighbor up/down" protocol, wherein all gateways have to ping all their neighbors. We could have the gateways just ping their "next door neighbors". The routing updates from neighbors and neighbors of neighbors could indicate to a gateway which other gateways it has as neighbors but not next door neighbors. (Remembering, of course, that neighborliness is only transitive when restricted to a particular network; it is not transitive in general, or each gateway would be a neighbor of all others.) The only distinction between being a next door neighbor and an ordinary neighbor has to do with the information placed in routing updates, and the pinging that is needed; the distinction is of no import as far as the routing of data packets goes. If G1 and G2 are next door neighbors on net N, as are G2 and G3, then G1 and G3 are neighbors and data packets can be routed from one to another without the need for any intermediate hop through G2.

Of course, the "next door neighbor" approach does introduce complications in the area of neighbor discovery and acquisition which we will have to deal with.

The "next door neighbor" approach can generate sub-optimalities, though, since it is no longer possible for each gateway, in its routing updates, to indicate its distance to ALL its neighbors. Rather, only distance to the next door neighbors is given. Yet the routing algorithm needs to know the distance to all neighbors. Well, if we really think it is important to have the routing algorithm know the precise measured distance between a certain pair of neighbors, we may just have to make them next door neighbors. Otherwise, we can use one of two approximations:

- Assume that all neighbors are equidistant. This might be appropriate for a broadcast network, like SATNET or an ethernet.

- If G1 and G3 are neighbors because each is a next door neighbor of G2 and all three are on the same net, assume that the distance between G1 and G3 is the sum of the distances from G1 to G2 and G2 to G3. This might be appropriate on a network like the ARPANET, if we tend to make a gateway's next door neighbors be the ones it is closest to on the ARPANET.

The routing updates from a particular gateway would have to indicate which approximation to use; this would have to be part of that gateway's config.

Having discussed the "next door neighbor" concept at some length, it is worth pointing out that most networks will probably have only some small number of gateways (1-3); it is only when we have networks with lots of neighboring gateways that we need to worry about this.

There are other methods of holding down the overhead of this sort of routing algorithm. One of the more exciting possibilities for the future is that of using a hierarchical routing strategy. An example of hierarchical would be to make the connectivity of the ARPANET gateways totally transparent to the routing algorithm executed by non-ARPANET gateways. For example, UCL and ISI could construct routing updates which imply that they connect to the ARPANET and are neighbors over it. Routing from distant portions of the internet could be done with this simplified picture of the internet topology. Of course, a separate, lower-level (in the hierarchy) routing algorithm would be needed to enable UCL and ISI to determine that they are considered neighbors in the higher-level (in the hierarchy) routing algorithm, and to tell them how to actually deliver traffic to each other.

I think we will really need this technique ultimately to handle a large internet to with a rich connectivity. This would provide a much better routing than could ever be gotten with EGP, but would still require that we have our own gateways all over the place.

We should also consider allowing two gateways to be neighbors if they are not on the same network, but rather are connected via someone else's autonomous system. This could allow us to provide a reasonable global routing algorithm in which our picture of the internet has other autonomous systems as pathways. It would still require that we have our gateways in lots of distant places, but would not require that every gateway a packet passes through be one of ours. (There are a number of issues in actually making this work, but it does fit nicely into the general architecture.)

Let's talk now about the dissemination of routing updates. I think we should use more or less the same procedure as is used in the ARPANET, with a few differences to make it more appropriate to the different environment. A gateway will generate an update as a result of neighbor up/downs, or as a result of measured distance changes. We will need to come up with a neighbor up/down protocol which is relatively immune to flapping, so that updates don't get generated too frequently for no real reason. (It should probably take at least a minute to declare a neighbor up, and we shouldn't be too quick to declare a neighbor down. Of course, if we have hard evidence that a gateway is down, like an 1822 dead, we should declare it down immediately, but still be somewhat slow to bring it back up. A gateway in its loader should give an "I am dead" response, to make things simpler.)

People who are steeped in GGP routing often seem to forget that in SPF routing, each gateway generates routing updates which contain information only about itself and its immediate neighbors, but which travel UNCHANGED to all other gateways, no matter how distant. The following discussion will be very confusing if one does not keep this in the forefront of one's mind.

The gateway which generates an update assigns it a sequence number. We should have a reasonably large sequence number space, so that wraparound is not a real problem. It should take at least several hours to wrap around, even with updates being generated at the maximum rate. Probably 16 bits is sufficient. Updates should be flooded to all gateways; flooding is probably the only technique reliable enough in a generally lossy and unreliable environment like the internet, but it does mean that generally, a gateway will get a copy of each update from each of its neighbors. (Flooding means that each gateway sends a copy of each update it receives to each neighbor. This is always the fastest way to get the updates around, in addition to being themost reliable.) Of course, a gateway doesn't need to process the duplicate copies of the upate; the sequence number allows for duplicate detection.

What stops the flooding? When a gateway gets an update it has already seen, we can assume we have already flooded it, and just ignore it.

One thing I am not too sure of right now is whether we can restrict the flooding to next door neighbors, or whether it really ought to involve all neighbors.

Routing updates will contain complete, rather than incremental, information. That is, every routing update from gateway G mentions all the (next door) neighbors for gateway G which are up at the time of the update. Neighbors which are down need not be mentioned. Neighbors which are not mentioned can be assumed to be down. (Actually, what matters is not whether the neighbor of G is up or down, but whether the direct path from G to that neighbor is up or down.) This ensures that the right thing happens if G goes down and then comes up with a different set of neighbors; there is no need for G to figure out who its neighbors were before so it can tell everyone else that they are no longer its neighbors.

The sequence number allows both duplicate detection and out-of-order detection. Since updates contain complete information, it is always sufficient to process the most recent update, even if some prior updates have not yet been seen. The information from out of order updates should never be used to modify the gateways's picture of the internet connectivity or the shortest path tree. This does mean that each gateway must keep track of the sequence number of the latest update it receives from each other gateway. (Of course, if it receives updates out of order, it keeps track of the sequence number of the update which was created most recently, not the one it received most recently.)

What if a gateway restarts and doesn't remember its the last sequence number it used? We can avoid this case since the gateways will have a store of easily and rapidly accessed non-volatile memory. We should use it to store the last sequence number used. Then when a gateway restarts, it can proceed with the next number in the sequence (we might want to skip a number for extra safety). But what if the non-volatile memory is lost? I think we can assume that a gateway which needs to be restarted after its non-volatile memory is wiped out will require some relatively long period of time (probably more than ten minutes?) to be reloaded. Thus we can say that if no update from gateway G has been received in more than 10 minutes, assume it has been reloaded and believe the sequence number of the next update you get.

In a generally lossy environment, just sending an update to a neighbor is not enough to ensure that he gets it. Unless one can determine that the neighbor has really received the update, one must retransmit it. (We need to define a retransmission interval, which will have to depend on the characteristics of the transmission medium between the neighbors.) There are two ways in which one can be sure that the neighbor has received the update:

a - He sends an explcit acknowledgment for it.

b - You receive the a copy of the same update from him. (In fact, there is no need to send a particular neighbor an update if he has sent you a copy of that same update.)

Note that, in general, if you receive a copy of an update from a neighbor after you have sent a copy of that same update to him, you need not acknowledge it explicitly, because of clause b.

There is, of course, one exception. If a neighbor retransmits an update to you, you should always ack the retransmission explicitly, or he will probably retransmit forever. This means that retransmissions must be marked as such.

One should never retransmit routing updates (or transmit them in the first place) to a neighbor who appears to be down. When the neighbor up/down protocol determines that a neighbor is up, one should send it all updates which it hadn't acknowledged. In the case of a new gateway, or one which has been down for a long time, this means one must send copies of all updates. To cut down on the amount of traffic this can cause, it may be desirable to be able to send multiple updates in a single datagram.

One may wonder why there is this need for two gateways that see each other come up to send all the routing information to each other. If one of the gateways was down, it is clear that it needs to get all the routing information before it can start sending and receiving data traffic. What is perhaps less obvious is the need for this big routing exchange if both gateways were up. However, it is important to realize that when two operating gateways are able to establish communications with each other after a period of not being able to talk to each other, it is possible that a partition of the internet is coming to an end. During the duration of the partition, there might have been many updates circulated in one segment which did not reach the other. In order to recombine the two segments, each needs to obtain all the routing information which is extant in the other. This technique assures that routing information from each segment will be sent around the other when the partition ends, since a gateway getting updates it hasn't seen before will flood them. Note that if the partition was not very severe, the routing updates will not flow far, since gateways that have seen them before will not flood them.

In the ARPANET, IMPs do not obtain information from their neighbors in this manner when a line come up. Rather, each IMP is required to re-issue a routing update every minute, and a line which is ready to come up is put into a special "waiting" state for a minute. While the line is in "waiting" state, routing updates are sent over it, but the line appears down to routing. This enables routing updates to flow across the erstwhile border of a partition before the line appears in the connectivity picture. This scheme has the advantage of not requiring any additional overhad when a line comes up. However, it does require every node to generate an update every minute, and it does assume that one minute is plenty of time for updates to get around. I don't know if this is an appropriate time constant in the internet. It is hard to use this sort of scheme with a longer time period, since this scheme has the consequence that when a node come up, it cannot begin participating in routing until that time period has elapsed. So we couldn't use a scheme like this while having the gateways send routing at least every 10 minutes, because we couldn't require that a gateway wait as much as 10 minutes before coming on line. Having gateways obtain routing information from their neighbors enables us to eliminate any dependence on time constants, at the cost of having a relatively large spike in the overhead. We should discuss the trade-offs here.

The need to send lots of updates to a neighbor when one first sees it come up can result in quite a bit of overhead. This is one reason why it is important to avoid flapping in the neighbor up/down protocol. To prevent an enormous spike in the overhead, two neighbors which see each other come up can send each other routing at some controlled rate, taking, say, a minute to send each other all the latest updates. Each gateway should wait until the minute elapses before creating and sending an update from itself indicating that its neighbor is up. This provides some time for routing information to circulate before the gateway begins to participate in routing.

In the ARPANET, there are no explicit acks for routing updates. An IMP acks a routing update by "echoing" back a copy of the same update. However, in the gateways, I would suggest we have an explicit acknowledgment packet. In certain cases, this might be much shorter than the routing update it acks. Also, it allows us to send multiple acks in a single packet. If we are sometimes going to have multiple updates in a single packet, it could be important to be able to have multiple acks in a single packet. As a reliability measure, we might want to be able to piggyback routing update acks in a hello (neighbor probe) packet; maybe each neighbor probe packet could contain the last n acks sent to that neighbor, or something like that.

Another slight complication: since each update carries complete information about its source gateway, later updates from a given source gateway obsolete earlier ones. Once one is in possesion of a later update from gateway G, one ought to stop retransmitting the earlier one from gateway G, since there is no longer any reason for the neighbor to get the earlier one.

This all implies that each gateway must keep track of which of its neighbors have received which updates from which source gateways. (This will certainly be more manageable if updates are flooded only to next door neighbors. But if we adopt that strategy, we have to make sure that there are not race conditions in the acquisition of next door neighbors which might cause some next door neighbor not to get some update.)

As added protection against the impossible case of an update's running around forever in circles, update packets should carry a hop count, as well as some indication of how long it has been since they were created. Note that the former should not change when an update is retransmitted, but the latter should.

A neighboring gateway that fails to ack routing updates is going to have to be considered down at some point. We need to give further consideration to the interaction between up/down and routing; perhaps some piggybacking of one function on the other is in order.