IDN-2

Internet Access Protocol

Part 2

Eric Rosen

August 1981

This message continues my free associating about the design of the internet gateway.

First, a little "off-the-wall" idea. In my last message, I proposed having a checksum (possibly optional) as part of the internet access protocol. Ordinarily, packets with bad checksums just have to be discarded. They can't be NAKed, since there is no way of knowing whether the addressing information has been corrupted. However, in the internet, even if we know that the Network Access Protocol might have been corrupted, we might still be sure that the Pathway Access Protocol has not been corrupted, and the Pathway Access Protocol might contain enough addressing information to enable us to NAK the corrupted packet. For

example, if an ARPANET gateway receives an internet datagram with a bad internet protocol checksum, the gateway can still be sure that the 1822 leader is uncorrupted (barring line hits on the 1822 line itself, which of course "never happens"). The 1822 leader of course contains all the addressing information needed to NAK the datagram, providing of course that the information is still around at the time when we look at the internet checksum. This ability to NAK corrupted packets might be very useful for debugging host software or for detecting flakey memory in hosts.

I've been thinking a little bit about interoperability of different gateway systems, and it still seems to me that there is no real problem. In the general case, when a gateway receives a packet over some particular Pathway access line, it knows automatically what the Pathway Access Protocol is. (E.g., anything received from an ARPANET access line must be wrapped in an 1822 envelope.) It strips off the Pathway Access Protocol envelope, and finds the Network Access Protocol envelope, which it then processes directly.

In some cases, it may not be necessary to have a separate Pathway access protocol. Consider the case of a gateway which is on an ethernet. It is possible, I suppose, to design the

ethernet interfaces so that they use the internet access protocol to access the ethernet. That is, the addressing information which each host must place in a packet in order to have it properly handled in the ethernet might be placed in a format either identical or at least compatible with the internet protocol. Then the gateway would not have to strip off any outer protocol envelope. The gateway would look at the destination address, determine whether the destination is on the ethernet (and is up), and take the packet only if use of the internet is necessary. I don't know if we will ever actually see this configuration, but it seems easy to handle. Of course, it assumes a compatible addressing scheme between the local net and the internet.

If the ethernet protocol is not identical to the internet protocol, then the gateway may have to strip off the ethernet protocol and put on the internet protocol before forwarding it further along the internet. The reverse transformation would be done in the reverse direction. A further optimization is also easy: if the destination host can be reached directly over one of the other networks that the gateway is connected to, the internet protocol need never be put on. Only the Pathway access protocol

for the "destination network" would have to be put on. (Unfortunately, the fact that most TCPs seem to require IP limits the utility of such an optimization. I guess no internet users are worried about things like volume-based tariffs.)

Now let's consider the case where a host on internet A needs to communicate with a host on internet B. For this to work, there must be some network C such that a gateway GA of internet A and a gateway GB of internet B are both hosts on network C. Now GA can be regarded as the destination gateway of internet A, since as far as GA is concerned, internet B is a Pathway (no internal structure) to the destination host. Similarly, gateway GB can be regarded as the source gateway for internet B, since it regards the internet A as a Pathway to the source host. When gateway GA gets some data for the destination host it only needs to strip off the network access protocol of internet A, replace it with the network access protocol of internet B, wrap it in the access protocol for network C (which is GA's Pathway to gateway GB, hence to internet B), and then send it to gateway GB via network C. That is, GA access internet B just as if it is an ordinary host on internet B. However, instead of giving its own name as the source address in the internet protocol, it gives the

name of the original source host. Of course, some simplification is possible if both internets use the same access protocol and the same logical addressing scheme, since then it is not necessary to remove one protocol envelope and replace it with another. Gateway GA will also have to have some way of knowing that certain addresses can only be reached through the other internet; that information is properly stored in the logical-to-physical translation tables.

This all seems pretty simple to me, though it seems like it ought to be much more complex. I can see some subtle problems with stuff like flow control. If gateway GB wants to send flow control messages back to the source host, GA might want to let those messages go right back to the actual source host, or it might want to intercept them, and use them as input to its own flow control mechanism. From our perspective, we only have to worry about what to do when one of our gateways receives flow control messages from another internet system which are destined for a host on our own internet system. I think it would be a good idea to intercept these and feed them into our congestion control system so that our own source gateway can merge this with our own internal congestion control information, and present a

single flow control interface to the source host. Of course, if one of our gateways encounters source quench messages, or messages which we don't understand which come from another internet, maybe we should just throw them away.

All that's really needed in the protocols to make all this possible is a protocol version field which is in the same place in every protocol. Can anyone think of any other problems which I have totally neglected?

I still have the feeling that treating an internet as a Pathway will result in some difficulties in fault isolation. However, if we routinely keep track of the rate at which we receive data on each "connection", and if we can gather this information routinely and reduce it in some useful way, we will at least be able to say whether someone's data has arrived at our internet, or whether it must have gotten lost before reaching our internet. Within our own internet, we can use some of the instrumentation bits in the packet headers (that I proposed in my last message) for similar purposes. If some particular source host claims that his data is not getting through, we can set an instrumentation bit in each packet that we see from him. Each of our gateways that sees a packet with this bit set can increment a

count. This would help us to understand whether all packets entering our internet are also leaving it, i.e., whether they are getting lost within some individual network. Keeping such counts in the intermediate gateways too would enable us to determine which individual network is at fault.

This suggests another feature that it would be nice to have in the gateways. If particular networks have, in their own access protocols, bits which can be used for instrumentation within that particular network, we need to be able to turn them on selectively when sending packets into that network. For example, the gateways ought to be able to select certain packets going into the ARPANET (say, every nth packet from a certain source to a certain destination) and turn on the ARPANET's trace bit in those packets. This could be very valuable for fault isolation.

I want to give you some idea of the way I would like to approach congestion control. Basically, I would have each gateway determine, for each of its Pathways to neighboring gateways, whether that Pathway is "overloaded", "underloaded", or "fully loaded". Given this determination, it is not difficult to modify the SPF algorithm so that it determines, for every route

between a source and destination gateway (where a route is just an ordered sequence of Pathways), whether that route is "overloaded" (i.e., has at least one overloaded Pathway"), "underloaded" (i.e., consists exclusively of underloaded Pathways), or "maximally loaded", (i.e., consists of at least one maximally loaded Pathway but no overloaded Pathways.) That is, at a given gateway G, the routing algorithm would tell it that the route from itself to another gateway G' was in one of these three states. In my last message, I proposed host-host connection blocks which would have throughput limits for that connection, which the source gateway could use to throttle the source host. I would like to have the throughput limits vary dynamically, as follows:

a) If a host-host connection goes to a gateway along an overloaded route, the throughput limits for that connection should be reduced periodically by a certain (small) amount, until a minimum is reached, or until the route to that gateway is no longer overloaded.

b) If a host-host connection goes to a gateway along an underloaded route, the throughput limits should be increased periodically by a certain (small) amount, until the limits reach

infinity or the route ceases to be underloaded.

c) Host-host connections using "maximally loaded" routes should have their throughput limits remain the same.

I really will have to write this up into a 25 page paper (at least) to make it really clear, but maybe you get the idea. The basic notion is to have a disciplined and enforced throttling of input into the internet, and in particular, to throttle all and only that traffic which would travel a congested route. Throttling limits would increase and decrease slowly until an appropriate level of load is reached. Note that controls are applied not merely to hosts which happen to get their packets dropped due to congestion, but rather to all hosts which are using congested Pathways.

If we routinely keep throughput statistics in the connection blocks, we can play some games to increase fairness. When lowering the throughput limits, for example, we can decrease the maximum allowable throughput by a percentage of the actual measured throughput, so that the heaviest users get throttled the most. This might tend to lead to an equalization of flows when the internet is congested, which is an important fairness consideration.

Of course, this sort of congestion control scheme would be experimental; I don't know if any similar scheme has been tried out (or even simulated) anywhere, although we will try to get such a scheme in our own ARPANET simulator, and a somewhat simpler version of this scheme will probably get into the ARPANET when we get rid of all the 16K IMPs. Of course, the guys at ISI will probably say that they thought of it first.

I glossed over the issue of how a gateway can determine the congestion status of its outgoing Pathways. I think we will have to investigate this issue experimentally. Congestion in the individual networks will tend to cause certain symptoms (such as queues getting longer than a threshold and staying longer for a certain amount of time, or an excessively high loss rate) and we will have to determine experimentally what these symptoms actually are. There will also be internal gateway congestion to worry about, which would show up as buffer shortage or high CPU utilization or something like that (perhaps the C70 microcode will enable us to determine exactly what the CPU utilization is.) When the congestion is internal to the gateway, all Pathways leaving that gateway should be declared congested, since the congestion affects them all equally.

Note that in order to determine whether to change the throughput limitations applied to a certain connection, we must know what the destination gateway of that connection is. This means that connection blocks must be individuated not only by source and destination host, but also by destination gateway. (That is, if we are load splitting by alternating the traffic for a certain destination host to different destination gateways, then we need separate connection blocks.) Also, if different classes of service cause traffic for the same destination gateway to take two different routes, we will need a different connection block for each type of service that requires a different route.

To have any hope of making this work efficiently, we must be able to efficiently access the set of connection blocks that are associated with a particular destination gateway. This suggests having a queue of connection blocks for each destination gateway. But what if a routing change causes data for destination host H to go via gateway G′ where it previously went via gateway G. We now need to move the connection blocks for destination host H (for a given type of service) from the G queue to the G′ queue. This means we need an efficient way to access all connection blocks associated with a given destination host. Since there is

nothing to prevent a block from being on two queues simultaneously, we can also have each block on a queue organized by destination host and type of service. This should allow efficient access of the blocks in either of two ways.

Maybe we should talk about this in person.

I want to raise an issue about the application of my logical addressing stuff to the internet. I have defined the notions of a logical address being "authorized" and of its being "effective." As I would like this to work in the ARPANET, when an IMP first comes up, the logical addresses of all directly connected hosts would be "ineffective" by default. When the connection between the IMP and the directly attached host first comes up, the host must send in LAD messages to indicate which of its logical addresses should become effective, and the IMP, if these logical addresses are authorized, will make them effective and will acknowledge this fact to the host. This works out nicely because host and IMP can easily tell when the other has gone down (or more precisely, when the connection between them has gone down.) However, I think this procedure would be too strict to apply to the internet. That is, I don't think the gateways will be able to keep track of all the host up/downs as a

matter of course; rather, they will only be able to get this information on an exception basis, as needed. On the other side of the coin, we certainly can't require every host to redeclare its logical addresses after every gateway crash. Therefore we will probably have to have all logical addresses be effective by default.

However, we can give the hosts the ability, if we want, to declare certain logical addresses to be ineffective, or to declare that it only wants certain of its logical addresses to be effective, and the others to be ineffective. That is, we can allow the hosts to use LAD messages as an option. The gateways receiving LAD messages would acknowledge them (possibly negatively, if a host asks to be addressed with an "unauthorized" name). A LAD message would retain its effect until superseded by another LAD message from the same host, or until the gateway restarts, in which case all logical addresses would become effective again, by default. This does mean that hosts may sometimes receive messages that are not intended for them (i.e., misdelivery of data.) This can happen if two hosts share a port in some network, but each has a different logical address. Since both logical addresses are effective by default, the internet has

no way of knowing automatically when one host is disconnected and the other one connected in its place. If one of these hosts receives data intended for another, we must rely on its willingness to inform the internet that as to which one of the logical addresses is to be effective at that time.

This is not such a problem if the network itself has logical addressing, since then we can rely on the network to tell us which host is really there, and to prevent misdelivery.

It is interesting to consider the particular case of a gateway on a local net. In this case, because of the extremely high bandwidth and low delay of the network, it might indeed be feasible to have the gateway keep track in real time of the up/down status of each host, and of the effectiveness of each authorized logical address for the local net. When a gateway comes up, all the logical addresses can be marked ineffective by default, and the gateway can query each host to see what logical addresses it is to be known by at that time. This suggests that we define a protocol by which a gateway can demand this information from the hosts on its network, and the hosts are required to respond if they want to be able to receive internet communications. (If a host sends a packet to the internet with a

particular source logical address, we might want to regard that as an implicit declaration that it is willing to receive data at that address. Unless wee want to allow logical addresses that can only send, but not receive??) This is an implication of optimizing the gateways for the particular nets that they are connected to ("the importance of Pathway characteristics"). Gateways on local nets would be expected to keep strict track of the hosts on the local net, but gateways on PR nets would not. This means our protocols need to be general enough to handle both cases.

These considerations suggest a possible way of connecting local nets to long haul nets like the ARPANET. We could have one of our gateways be a host on both the ARPANET and the local net. From the perspective of the ARPANET, the gateway would appear to be a single host with many logical addresses. That is, the ARPANET logical addresses of all hosts on the local net would map to the gateway's physical address. The gateway could keep track of which hosts on the local net are up or down, and keep the ARPANET informed of this. The gateway could handle all protocol needed to interface between the ARPANET and the local net. In this way, hosts on the local net could be treated as ordinary

ARPANET hosts, as well as being treated as internet hosts. Seems like this ought to be good for something.

It occurs to me that we ought to have some analogue of the ARPANET's "host going down" and "imp going down" messages. That is, maybe hosts on a network should be able to inform all the gateways on that network that they are going to be down for awhile, indicating a reason and a duration of the outage. Any time a host on some other network tries to communicate with a host that has declared itself to be down in this way, the destination gateway can send the source gateway a control packet with the reason for and duration of the outage, and the source gateway can feed this back to the source host. Once a host has declared itself down by sending a "host going down" message to a particular gateway, that gateway should consider the host to be down until it hears otherwise from the host.

More on up/down later.