IEN #86

EXTENDED INTERNET ROUTING

Radia Perlman
Bolt, Beranek, and Newman
April 5, 1979

INTRODUCTION

The catenet differs from most networks because in the catenet not all links are functionally identical. Because, for various reasons, various types of packets should not be routed on certain of the links, no single topology defines the connectivity state of the catenet for all types of traffic.

There are three factors to the design of catenet routing to meet these needs:
1) categorizing a packet
2) routing a categorized packet
3) preventing spoofing
These will be defined and discussed in this paper.

CATEGORIZING PACKETS

A packet's category is a number that is n bits long, where n is the number of networks plus the number of other factors that should be considered, such as the delay class or reliability desired. A "1" for a bit means the packet is allowed to traverse that network, or the factor (such as delay) corresponding to that bit is of importance.

In this paper we will assume that packets are routed according to:
1) delay
2) reliability
3) authorization.

Authorization can be based on such things as:
1) source net
2) destination net
3) source host
4) destination host
5) "stamp of approval".

The "stamp of approval" would be some sort of code given out by an access controller for a network that wishes to restrict traffic into or through itself (and does not wish to rely solely on other information in the internet header). A user who wishes to use one of these networks must contact the relevant access controller and receive the code. Either access controllers would be cooperative, and a single access controller could give the stamp of approval for several networks at once, or the user would have to contact an access controller for each fussy network he wished to use.

constrained into bursts. For instance if the window is 6 packets from gateway B to gateway A, B can send 6 packets, can receive an acknowledgment that the first 3 were received, so that B can be sending 3 more packets while the acknowledgment after the 6th packet can be delivered. (Otherwise, if A sent acknowledgments only after 6 packets, then there would be a packet round trip delay time during which B could not send any packets--the time from launching the 6th packet to the time of receipt of the acknowledgment.) This automatically limits the amount of traffic down each path to what can be handled by the next gateway. To ensure that a lost acknowledgment will not deadlock flow between two gateways, a gateway waiting for an acknowledgment of its last group should prod the next gateway with null packets requesting acknowledgments. There must also be a method for identifying acknowledgments, so that the receiver of an acknowledgment can determine when the acknowledgment was sent with respect to the packet stream. Also, the window lasts only a short time. If the first gateway sends k-1 packets (where acknowledgments are usually sent after every k packets), then sends no packets for a while (like several seconds), when there are more packets to send it should send an entire window's worth, not wait for an acknowledgment of its first packet, because by then the old packets have presumably been dealt with successfully or dropped. Window schemes obviously require increased control traffic overhead, and increased processing time by the gateway. They must also be very carefully designed, as there are all sorts of potential phase problems.

SOURCE NOTIFICATION

No matter how elaborate a scheme of load splitting is implemented, there is a limit to the rate of traffic the catenet can handle, and if the source sends packets faster than that, packets will be dropped and the throughput curve will go over into the falling range.

Thus there must be a method of providing feedback to the source that it is sending packets too quickly. One method is a windowing scheme, as described above for the gateway to gateway case. The window would be in effect between the source host and the first gateway. This has problems because the number of hosts on a network is much larger than the number of neighbor gateways a gateway has. If many hosts sent a window's worth of packets to a single gateway at once, the vast majority of packets would get dropped.

Another method is a rate feedback scheme. In this type of scheme, a gateway becoming congested would send messages back to packet sources indicating that the packet source should cut down on its rate to that destination net. This mechanism gives a source faster feedback than a mechanism that expects congestion to get reported from gateway to gateway until the congestion

propagates all the way back to the source. (That would be the case if a windowing scheme were implemented throughout the catenet, including from the source to the first gateway, or if rate messages were sent only one hop.) The steady state would consist of a source gradually increasing its rate until it starts receiving warning messages. Then the source would back down the rate, wait a while, and if it stops receiving warning messages, it will gradually increase its rate again. If it continues receiving warning messages, it will back down some more, etc.

This scheme leads to implementation problems in the source, though. One question is what protocol level should respond to the message. There is the half gateway, the higher level protocol (such as TCP, telnet, XNET, etc.), and ultimately the user (person). Obviously the person shouldn't be expected to notice a message and type slower, or whatever. If the protocol is expected to respond, then all the code for implementing that would have to be duplicated in every module that sends packets. The most efficient mechanism is for the half gateway to respond. This means that the half gateway must keep tables of destination networks and rates. It must keep track, per destination net, of the rate that packets are being sent. If it receives more packets for a destination network than the allowed rate, it must either accept packets more slowly from the processes that are generating the packets (which has the disadvantage of slowing down all packets from those processes--not just to the slow destination), refuse packets from them selectively (a great deal of overhead), or drop relevant packets.

FAVORING THROUGH TRAFFIC OVER SOURCE TRAFFIC

Intuitively, it is more expensive to drop a packet that has travelled many hops than a new packet, because if the dropped packet must be retransmitted, all the hops it had traversed before are wasted. This is rigorously described in the paper "Congestion Control of Store-and-Forward Networks by Input Buffer Limits--An Analysis", by Simon Lam and Martin Reiser.

This scheme can be implemented in the internet context. What it would consist of is that gateways would place a limit on the number of buffers (say half of its buffers) that can be devoted to packets from sources (not from gateways). There would be no limit on through traffic (traffic from other gateways). In the heavily congested case this would cause no new traffic to be allowed to enter the catenet. In the very lightly loaded case there is no decrease in performance because of the buffer limit. This scheme does not seem fair, however. A very large packet stream traversing some network could completely lock out traffic from that network from entering the catenet. This scheme should not be implemented unless it is firmly believed that other flow control measures would ensure that a catenet state in which a gateway was so congested that it had no buffers left over for

source traffic would be a short-lived state. The scheme could also be modified to place a buffer limit on through traffic (that limit would be higher than the input buffer limit). This would provide that, even when the catenet was very congested, a very small amount of source traffic could be injected into the catenet. Although the overall cost function would degrade with this modification (since some high cost packets would be dropped in favor of low cost, new packets), fairness is just as important a consideration.