3.6 Reverse Path Forwarding

We now describe another broadcast routing algorithm that forwards packets based on their source of broadcast. The simple version of the algorithm does not use a forwarding table like Table 3.2, but only the routing table that a node would normally use to route packets. This simple scheme is suboptimal in the number of packet copies transmitted, and so we extend it to use a table similar to Table 3.2. This table is constructed dynamically and therefore the algorithm is adaptive to changing network conditions. Note that even the suboptimal algorithm would be adaptive since it makes use of the routing table, which in most communication networks is dynamically updated. We will describe the algorithm in context of the routing algorithm and tables used by the IMPs in the ARPANET. Appendix C briefly reviews this routing algorithm.

The idea behind the algorithm was first proposed by Metcalfe (in a private communication to the author). The broadcast packets are not forwarded along the tree of shortest paths that connect the source to the destinations, but rather the tree of shortest paths that connect the destinations to the source. Hence, the name reverse path forwarding. In the event that communication costs in either direction of an edge are same, the two trees will be identical.

3.6.1 The Simple Scheme

When a broadcast packet from a particular source (with destination "all") arrives on a particular communication link, the node determines

from its routing table whether it would route a packet to that source along the same link. If so, the packet is delivered to all hosts connected to the node, and forwarded along all links except the one on which it arrived. The node does this because it concludes that it lies on the shortest path that connects some of the destinations to the source. If the packet did not arrive on the correct link, then it is discarded.

This simple algorithm guarantees prevention of network flooding, because the paths along which packets are accepted for delivery and forwarding are cycle free. We show why this is so. A packet is accepted on a link if the link is part of the paths that connect the destinations to the source by the shortest paths as determined by the normal routing algorithm. If the normal routing algorithm creates, at any time, a unique route between two nodes, then the shortest paths from the destinations to the source will be a tree that is cycle free. These paths will also be cycle free if the normal routing algorithm creates cycle free routes between any two nodes. Routing algorithms, in general, satisfy both these properties.

This scheme requires no modification to the packet format currently used in the ARPANET. The IMP code would, however, have to be enhanced to perform the forwarding function. Abstractly, in an ALGOL-like language, the program in the IMP would be:

```
if PACKET.DEST = ALL
then begin
     if INCOMING_LINK = ROUTING_TABLE(PACKET.SOURCE)
     then [deliver to all hosts, and forward along all other edges]
     else [discard the packet]
     end
else begin
     [route this normal packet];
     end;
```

The number of packets transmitted is, however, larger than the minimum. The delay for propagating the packets may be larger than the optimal value if the reverse path tree is very much different from the shortest path tree. Figure 3.4 illustrates how packets would be forwarded along the reverse paths from node 8. The notation for labelling packets is identical to that used in section 3.4. Note, again, that the packets that are accepted by the nodes for delivery to the hosts connected to them, arrive on branches of a tree that connect the destinations to the source by the shortest path.

3.6.2 The Optimal Scheme

Notice from figure 3.4 that the total number of packet copies transmitted is 22, while the minimum number is N-1=8. The total number is equal to the sum of the connectivities of each node minus (N-1), since each node forwards a packet along all but one link incident to it, except for the source which transmits the packet on all links incident to it. For most network topologies this number is much larger than the minimum.

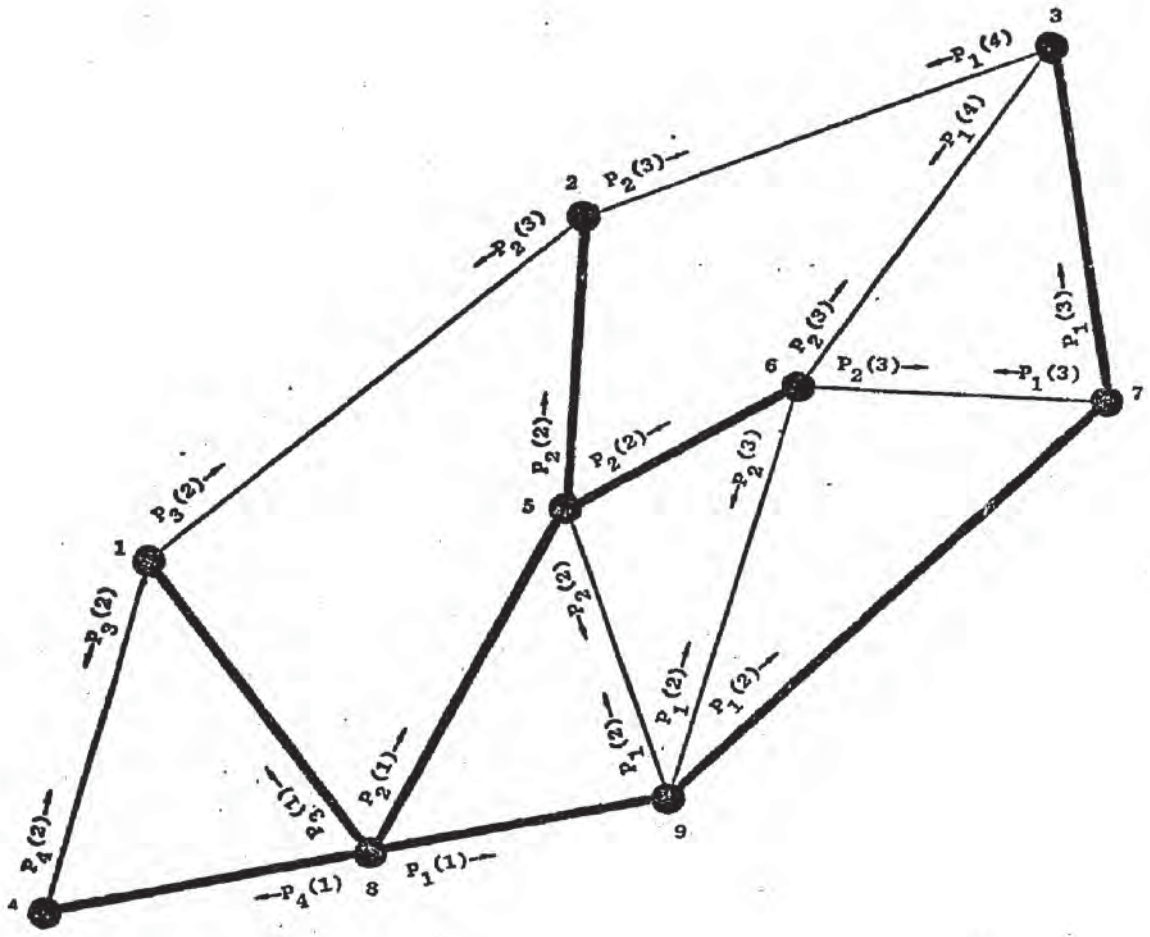

Figure 3.4. SUBOPTIMAL REVERSE PATH FORWARDING INITIATED FROM NODE 8. (Note: host addressing synonymous with node addressing.)

If each node knew along which links to forward the packet (as determined by columns of Table 3.2), then the number of packet copies transmitted would be equal to N-1. Such a BROADCAST_FORWARDING_TABLE at node 6, of the network in figure 3.2, is illustrated in figure 3.5

In terms of the reverse path forwarding model, in order to construct such a forwarding table, a node must know to which destinations a given link will be used by its neighboring nodes to transmit a packet. Hence, if a broadcast packet arrived from one of those "destination", then the node will know along which links to forward it. Periodically, each node will transmit to its neighbors a LINK_USAGE_TABLE indicating to which destinations the node will use this link for transmitting packets. When a node receives such a table, it is written over the appropriate column of its BROADCAST_ROUTING_TABLE. The row to be written over is the row corresponding to the link that the arriving LINK_USAGE_TABLE came in on. The LINK_USAGE_TABLE can easily be derived from the ROUTING_TABLE at a node. The LINK_USAGE_TABLEs are the inverse of the ROUTING_TABLE, i.e. they indicate for which destinations a particular link will be used. Figure 3.5 also illustrates the ROUTING_TABLE from which the LINK_USAGE_TABLE for link d was derived. This LINK_USAGE_TABLE will be transmitted to node 5.

The LINK_USAGE_TABLE could be transmitted every 2/3 of a second along with the MINIMUM_DELAY_TABLE. The BROADCAST_ROUTING_TABLE and LINK_USAGE_TABLEs could be stored as bit maps since their entries are logical values.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.