TDTS04 Computer Networks and Distributed Systems

Distance Vector Routing Lab

Authors
Maximilian Bragazzi Ihrén, maxbr431@student.liu.se
Markus Lewin, marle943@student.liu.se

VT1
Version 1.0

1. Audit Descriptions

Ver.	Audit Description	Date
1.0	Written for report.	150303

2. How distance vector routing works

The distance vector routing protocol operates by having all the nodes in the system periodically send updates of it's cheapest costs to all it's neighbors.

The actual term “distance vector” refers to the fact that each node houses a vector containing it's costs to other nodes in the system, which it then uses to determine which path a packet should be sent.

Each node waits and listens for any cost changes in the system, at which point it updates it's cheapest costs, and sends an update to its neighbors (which in turn updates and then sends to its neighbors, a sort of recursive solution). This process is called “routing by rumor”, since all routers have to assume that its neighbors are telling the truth.

The actual calculation is based upon the Bellman-Ford equation, which has a fair share of problems, such as routing loops and “count-to-infinity” problems. This in turn is worked around (at least in the RIP protocol) by poisoned reverse.

“Count-to-infinity” refers to when a node receives an update from another node, and sees a cheaper path. Since it does not know whether or not that path is routing through itself, it might choose that path, which would lead to the two nodes sending updates back and forth, incrementing with the cost of their link every time, until they find a cheaper path through some other node.

Poisoned reverse solves the count-to-infinity problem (sort of, see point 4). It does this by, when sending an update, checking whether or not any routes in your “cheapest paths” are routing through the node you are currently sending to, and if it does it sets that value to infinity (in the packet, not for itself).

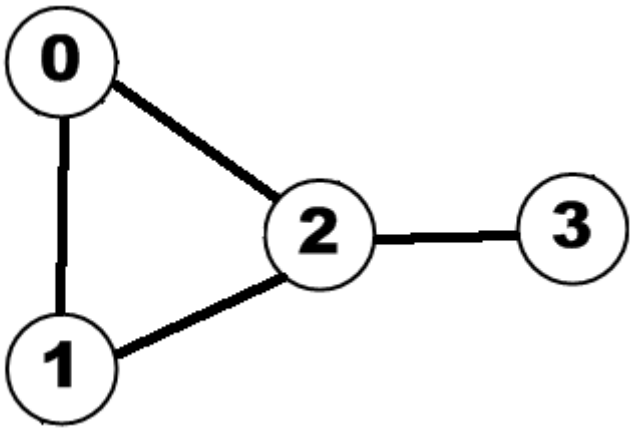
3. How we tested the algorithms

We tested the algorithms by first drawing up the network in every simulation and calculating the shortest path ourselves, and then ran the simulation and checked that against our calculations (our code actually corrected our personal calculations from time to time!).

4. When poisoned reverse may fail

Poisoned reverse only works for loops concerning two nodes, once the loop concerns three or more nodes, poisoned reverse fails.

For example:



Assume that the 2-3 connection fails.

The optimal path from 1 to 3 is 1-0-2-3, which means that (through poisoned revere), 1 will not send that path as infinite to 2.

2 can in this case choose to go through 1 as the next hop to 3, since 1 has no idea that 2-3 failed.

This will in turn create a loop.

5. A solution to poisoned reverse failing

A solution to above would be that 1 would advertise it's smallest cost to 3 along with the path for the same connection. That way, 2 could check if it is in the path somewhere, and therefore not choose that path.

Or, alternatively, that 1 saves the entire path it has to 3, and checks if the node it is sending to is in that path at all, and in that case sends infinite. Of course, this would mean that the node that connected to 1 would have to send it's entire path to 1.

The only difference between the two solutions is where the check is performed.

This of course increases the size of the packets sent, which would increase the cost of packet sending (for the protocol).