Computer Networks (CS425)

Instructor: Dr. Dheeraj Sanghi

Prev| Next| Index

Routing Algorithms

Non-Hierarchical Routing

In this type of routing, interconnected networks are viewed as a single network, where bridges, routers and gateways are just additional nodes.

- Every node keeps information about every other node in the network
- In case of adaptive routing, the routing calculations are done and updated for all the nodes.

The above two are also the disadvantages of non-hierarchical routing, since the table sizes and the routing calculations become too large as the networks get bigger. So this type of routing is feasible only for small networks.

Hierarchical Routing

This is essentially a 'Divide and Conquer' strategy. The network is divided into different regions and a router for a particular region knows only about its own domain and other routers. Thus, the network is viewed at two levels:

1. The Sub-network level, where each node in a region has information about its peers in the same region and about the region's interface with other regions. Different regions may have different 'local' routing algorithms. Each local algorithm handles the traffic between nodes of the same region and also directs the outgoing packets to the appropriate interface.
2. The Network Level, where each region is considered as a single node connected to its interface nodes. The routing algorithms at this level handle the routing of packets between two interface nodes, and is isolated from intra-regional transfer.

Networks can be organized in hierarchies of many levels; e.g. local networks of a city at one level, the cities of a country at a level above it, and finally the network of all nations.

In Hierarchical routing, the interfaces need to store information about:

- All nodes in its region which are at one level below it.
- Its peer interfaces.
- At least one interface at a level above it, for outgoing packages.

Advantages of Hierarchical Routing :

- Smaller sizes of routing tables.
- Substantially lesser calculations and updates of routing tables.

Disadvantage :

- Once the hierarchy is imposed on the network, it is followed and possibility of direct paths is ignored. This may lead to sub optimal routing.

Source Routing

Source routing is similar in concept to virtual circuit routing. It is implemented as under:

- Initially, a path between nodes wishing to communicate is found out, either by flooding or by any other suitable method.
- This route is then specified in the header of each packet routed between these two nodes. A route may also be specified partially, or in terms of some intermediate hops.

Advantages:

- Bridges do not need to lookup their routing tables since the path is already specified in the packet itself.
- The throughput of the bridges is higher, and this may lead to better utilization of bandwidth, once a route is established.

Disadvantages:

- Establishing the route at first needs an expensive search method like flooding.
- To cope up with dynamic relocation of nodes in a network, frequent updates of tables are required, else all packets would be sent in wrong direction. This too is expensive.

Policy Based Routing

In this type of routing, certain restrictions are put on the type of packets accepted and sent. e.g.. The IIT- K router may decide to handle traffic pertaining to its departments only, and reject packets from other routes. This kind of routing is used for links with very low capacity or for security purposes.

Shortest Path Routing

Here, the central question dealt with is 'How to determine the optimal path for routing ?' Various algorithms are used to determine the optimal routes with respect to some predetermined criteria. A network is represented as a graph, with its terminals as nodes and the links as edges. A 'length' is

associated with each edge, which represents the cost of using the link for transmission. Lower the cost, more suitable is the link. The cost is determined depending upon the criteria to be optimized. Some of the important ways of determining the cost are:

- **Minimum number of hops:** If each link is given a unit cost, the shortest path is the one with minimum number of hops. Such a route is easily obtained by a breadth first search method. This is easy to implement but ignores load, link capacity etc.
- **Transmission and Propagation Delays:** If the cost is fixed as a function of transmission and propagation delays, it will reflect the link capacities and the geographical distances. However these costs are essentially static and do not consider the varying load conditions.
- **Queuing Delays:** If the cost of a link is determined through its queuing delays, it takes care of the varying load conditions, but not of the propagation delays.

Ideally, the cost parameter should consider all the above mentioned factors, and it should be updated periodically to reflect the changes in the loading conditions. However, if the routes are changed according to the load, the load changes again. This feedback effect between routing and load can lead to undesirable oscillations and sudden swings.

Routing Algorithms

As mentioned above, the shortest paths are calculated using suitable algorithms on the graph representations of the networks. Let the network be represented by graph G (V, E) and let the number of nodes be 'N'. For all the algorithms discussed below, the costs associated with the links are assumed to be positive. A node has zero cost w.r.t itself. Further, all the links are assumed to be symmetric, i.e. if $d_{i,j}$ = cost of link from node i to node j, then $d_{i,j} = d_{j,i}$. The graph is assumed to be complete. If there exists no edge between two nodes, then a link of infinite cost is assumed. The algorithms given below find costs of the paths from all nodes to a particular node; the problem is equivalent to finding the cost of paths from a source to all destinations.

Bellman-Ford Algorithm

This algorithm iterates on the number of edges in a path to obtain the shortest path. Since the number of hops possible is limited (cycles are implicitly not allowed), the algorithm terminates giving the shortest path.

Notation:

$d_{i,j}$ = Length of path between nodes i and j, indicating the cost of the link.
h = Number of hops.
D[i,h] = Shortest path length from node i to node 1, with upto 'h' hops.
D[1,h] = 0 for all h .

Algorithm :

Initial condition : D[i, 0] = infinity, for all i (i != 1)

Iteration : D[i, h+1] = min { $d_{i,j}$ + D[j,h] } over all values of j .

Termination : The algorithm terminates when

D[i, h] = D [i, h+1] for all i .

Principle:
For zero hops, the minimum length path has length of infinity, for every node. For one hop the shortest-path length associated with a node is equal to the length of the edge between that node and node 1. Hereafter, we increment the number of hops allowed, (from h to h+1) and find out whether a shorter path exists through each of the other nodes. If it exists, say through node 'j', then its length must be the sum of the lengths between these two nodes (i.e. $d_{i,j}$) and the shortest path between j and 1 obtainable in upto h paths. If such a path doesn't exist, then the path length remains the same. The algorithm is guaranteed to terminate, since there are utmost N nodes, and so N-1 paths. It has time complexity of O (N^3) .

Dijkstra's Algorithm

Notation:
D_i = Length of shortest path from node 'i' to node 1.
$d_{i,j}$ = Length of path between nodes i and j .

Algorithm
Each node j is labeled with Dj, which is an estimate of cost of path from node j to node 1. Initially, let the estimates be infinity, indicating that nothing is known about the paths. We now iterate on the length of paths, each time revising our estimate to lower values, as we obtain them. Actually, we divide the nodes into two groups ; the first one, called set P contains the nodes whose shortest distances have been found, and the other Q containing all the remaining nodes. Initially P contains only the node 1. At each step, we select the node that has minimum cost path to node 1. This node is transferred to set P. At the first step, this corresponds to shifting the node closest to 1 in P. Its minimum cost to node 1 is now known. At the next step, select the next closest node from set Q and update the labels corresponding to each node using :

$$\mathbf{D_j = min [D_j , D_i + d_{j,i}]}$$

Finally, after N-1 iterations, the shortest paths for all nodes are known, and the algorithm terminates.

Principle

Let the closest node to 1 at some step be i. Then i is shifted to P. Now, for each node j , the closest path to 1 either passes through i or it doesn't. In the first case Dj remains the same. In the second case, the revised estimate of D_j is the sum $D_i + d_{i,j}$. So we take the minimum of these two cases and update D_j accordingly. As each of the nodes get transferred to set P, the estimates get closer to the lowest possible value. When a node is transferred, its shortest path length is known. So finally all the nodes are in P and the D_j 's represent the minimum costs. The algorithm is guaranteed to terminate in N-1 iterations and its complexity is O(N^2).

The Floyd Warshall Algorithm

This algorithm iterates on the set of nodes that can be used as intermediate nodes on paths. This set grows from a single node (say node 1) at start to finally all the nodes of the graph. At each iteration, we find the shortest path using given set of nodes as intermediate nodes, so that finally all the shortest paths are obtained.

Notation
$D_{i,j}$ [n] = Length of shortest path between the nodes i and j using only the nodes 1,2,....n as intermediate nodes.

Initial Condition
Di,j[0] = $d_{i,j}$ for all nodes i,j .

Algorithm
Initially, n = 0. At each iteration, add next node to n. i.e. For n = 1,2, ...N-1 ,

Di,j[n + 1] = min { $D_{i,j}$[n] , $D_{i,n+1}$[n] + $D_{n+1,j}$[n] }

Principle
Suppose the shortest path between i and j using nodes 1,2,...n is known. Now, if node n+1 is allowed to be an intermediate node, then the shortest path under new conditions either passes through node n+1 or it doesn't. If it does not pass through the node n+1, then $D_{i,j}$[n+1] is same as $D_{i,j}$[n] . Else, we find the cost of the new route, which is obtained from the sum, $D_{i,n+1}$[n] + $D_{n+1,j}$[n]. So we take the minimum of these two cases at each step. After adding all the nodes to the set of intermediate nodes, we obtain the shortest paths between all pairs of nodes together. The complexity of Floyd-Warshall algorithm is O (N^3).

It is observed that all the three algorithms mentioned above give comparable performance, depending upon the exact topology of the network.

---

back to top
Prev| Next | Index