ECE 333: Introduction to Communication Networks
Fall 2001

Lecture 23: Routing and Addressing II

- Link state routing
- Hierarchical routing

Link state Routing

In Lecture 22, we consider distance vector routing. In distance vector routing, each node maintains a list of the distance to each destination and periodically broadcasts this list to its neighbors. Each node uses the Bellman-Ford Algorithm to calculate the shortest path.

Next we consider another type of shortest-path routing algorithm - ***Link state routing.***

In link state routing, each router must:

1. Discover its neighbors and learn their addresses.
2. Measure cost to each neighbor, this is called the "link state".
3. Broadcast (e.g. flood) a packet containing the link state to **all** other routers
4. Use the information it receives to compute the shortest path to all other routers.

Each node will receive a link state packet from every other node in the network. Thus, each node can reconstruct a graph of the entire network topology and use this to calculate the shortest paths to every node. Each node will individually calculate the shortest paths, based on this common information. To calculate shortest paths, ***Dijkstra's Algorithm*** is used.

Dijkstra's Algorithm

Given a graph for the entire network, Dijkstra's Algorithm can be used to find the shortest paths from a given node to ***every*** other node. The basic idea of Dijkstra's algorithm is to first find the closest node, then the second closest node, etc. For a given node, s, notice the first closest node must be a neighbor of s and the second closest node must either be a neighbor of s or a neighbor of the first closest node, and so on.

Why?

Dijkstra's Algorithm

Let $C(i)$ be the cost from the source node, s , to node i.

Let $d_{i,j}$ again be the link cost from node i to j. (Set $d_{i,j} = \infty$ if no link between i and j.)

To describe Dijkstra's algorithm, we introduce a set N, which will contain the nodes for which the shortest path is known at each step of the algorithm. The algorithm will finish when every node is in this set.

Dijkstra's Algorithm:

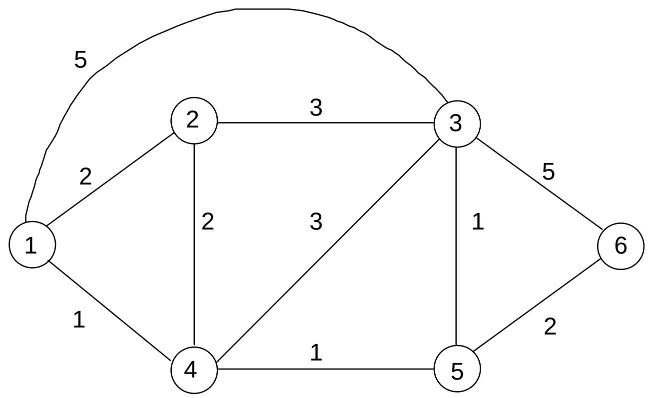
Initialization: Let $N = \{s\}$, for each $j \notin N$, set $C(j) = d_{s,j}$

Until all nodes are in N:

1. Choose $j \notin N$ with minimal $C(j)$ (use any rule to break ties).
2. Set $N = N \cup \{j\}$.
3. Update $C(i)$: For all $i \notin N$ set $C(i) = min\ [C(i),\ C(j) + d_{i,j}]$

Stop.

Example of Dijkstra's Algorithm



Suppose we wish to find the minimum cost path to each node from node 6.

Initialization: $N = \{6\}$ $C(1) = \infty$ $C(2) = \infty$ $C(3) = 5$ $C(4) = \infty$ $C(5) = 2$

Step 1: Add node 5 to N ($N = \{6, 5\}$)

Update costs as $C(i) = \min \{ C(i), C(5) + d_{i,5} \}$

We get $C(1) = \infty$ $C(2) = \infty$, $C(3) = 3$, $C(4) = 3$

Note: $C(5)$ not updated

$C(3) = \min [C(3), C(5) + d_{i,5}] = \min [5, 2 + 1]$

$= 3$

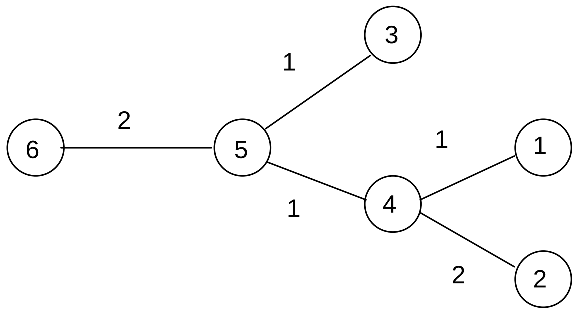
Continuing we get:

Step	N	*C(1)*	*C(2)*	*C(3)*	*C(4)*	*C(5)*
init	6	∞	∞	5	∞	2
1	6,5	∞	∞	3	3	2
2	6,5,3	8	6	3	3	2
3	6,5,3,4	4	5	3	3	2
4	6,5,3,4,1	4	5	3	3	2
5	6,5,3,4,1,2	4	5	3	3	2

Note as each node is added to N, we also keep track of which "link" is used in calculating the cost.

When the algorithm is finished we can construct a tree, which is used for routing from node 6.

For the above example this tree is:



Thus any packet departing from node 6 would be routed on the link towards node 5.

Link State vs. Distance Vector

- Distance Vector Routing: each router communicates locally (only to its neighbors), and sends global information (distances to all nodes)
- Link State Routing: each router communicates globally, but only sends local information (distances to neighbors).

- Link state routing generally converges faster than distance vector approaches.
- Link state routing is also more robust to bad information and does not exhibit the count-to-infinity problems associated with distance vector approaches.
- However link state routing does not *scale* as well as distance vector approaches due to need for broadcasting link state packets to all nodes.

Some issues in adaptive routing:

We have described both the link state and distance vector algorithms for a network with a given set of link costs. In an adaptive routing algorithm, these link costs may change based on congestion, delay, etc. In designing an adaptive routing algorithm several issues need to be addressed, including:

1. How often to send routing updates?
2. How to deal with the possibility of errored or delayed routing updates?
3. What is the interaction or routing and link costs

   e.g., if link cost reflect delay, then by changing routes, can change delays, which can lead to further changes in routes, etc.

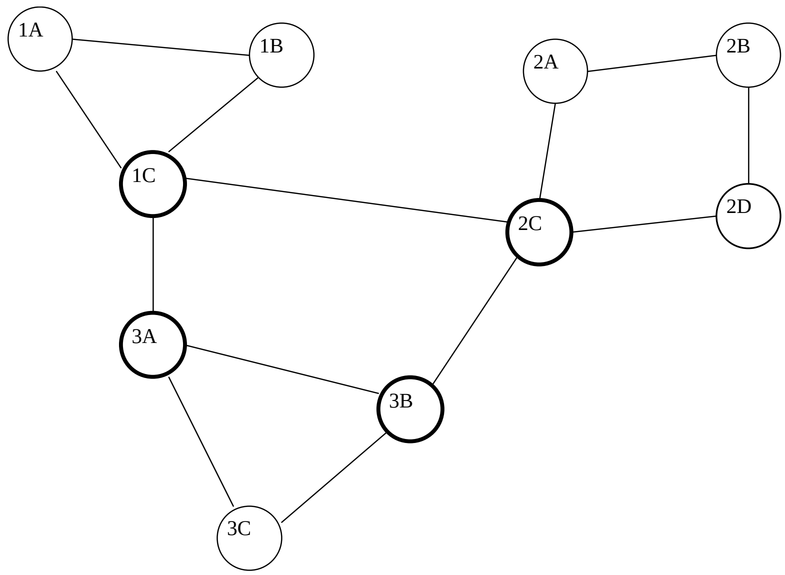
Hierarchical Routing

For large networks, **scaling problems** arise in any routing algorithm, such as:

- Routing tables become huge - adds memory cost and slows forwarding due to increased look-up time.
- Routing information traffic becomes huge; in link state routing this leads to many packets needed to be flooded, in distance vector approaches, this leads to larger distance vectors.

Hierarchical Routing provides one solution to these scaling problems. The idea behind hierarchical routing is to somehow partition the network into regions. All nodes within a region need to calculate routes for all other nodes in the region. Traffic to nodes in other regions will only be routed based on the region of the other node. Inter-region traffic is often handled by special routers, these routers can again run shortest-paths algorithms to find the shortest paths between regions. Implementing hierarchical routing requires ***hierarchical addressing***, so that the region of a destination can be easily identified. A well-known example of this is the area code in telephone numbers.

Hierarchical routing example:



In the above example, the network is divided into 3 regions as indicated by the number identifying each node. The routers shown in bold handle the inter-region traffic.

Notice the routing table at node 1A without hierarchy would contain 9 entries, with hierarchy contains only 3 entries.

Such an approach can result in sub-optimal route selection, but the added benefit in terms of better manageability and scalability are worth this cost.

The idea of hierarchical routing can also be extend to multiple (>2) levels of hierarchy.