CS 456 Hao Pan
Salahuddin, Mohammad
Spring 2018

Chapter 5.2 Routing Protcols

5.2.1 Overview

- The goal of **routing protocols** is to determine *"good"* paths/routes from a sending host to the receiving host, through a network of routers.
- A **path** is a sequence of routers which packets will travel through while going from the source host to the destination host.
- Whether or not a path is *good* can be define in different ways, such as having the least cost, is the fasted, or is the least congested.
- See an example of graph abstraction from a network on **slide 5-9**.
- The *cost* of a path could be defined in different ways depending on preference. The cost could always be 1 (meaning shortest path wins), inversely related to bandwidth, or inversely related to congestion (take speed into account).

5.2.2 Routing Algorithm Classifications

- Routers can have *global* or *decentralized* information:
  - **Global**: All routers have complete a complete topology and the link cost info. This is the kind of information that *"link state" algorithms* provide.
  - **Decentralized**: Routers only know their physically-connected neighbours and the link cost to those neighbours. It is an iterative process where they exchange info with neighbours. This is the info that *"distance vector" algorithms* provide.
- Router information can be *static* or *dynamic*:
  - **Static** routes change slowly over time.
  - **Dynamic** routes change more quickly in response to changes to link costs. They also update periodically.

5.2.3 A Link-State Routing Algorithm

- **Dijkstra's algorithm** is commonly used to compute the net topology and link costs of a network.
- It is guaranteed to compute the least cost paths from one node to all other nodes.
- Using Dijkstra's algorithm, we know that after k iterations, we have the least cost paths to k destinations.
- See pseudocode for Dijkstra's algorithm on **slide 5-14**. See examples of it in action on **slide 5-15** to **slide 5-17**.
- Dijkstra's algorithm has a complexity of $O(n^2)$, where n is the number of nodes. A more efficient algorithm can reduce the complexity to $O(n \log n)$.

5.2.4 Distance Vector Algorithm

- The **Bellman-Ford equation** (dynamic programming) is often used.
- The algorithm can be found on **slide 5-20**.
- The algorithm essentially uses dynamic programming to compute the least cost path by checking the cost to reach each of the target node's neighbours.
- In other words, suppose we are trying to find the shortest path from node x to node y. Suppose y has neighbours v. The Bellman-Ford algorithm will find the shortest path from x to each v, then use that to compute the smallest of $D(x, v) + D(v, y)$. Do note that it will recursively go back by checking the neighbours of v when looking for the shortest path from x to v.
- An example can be seen on **slide 5-21**.
- The key idea of this algorithm is that from time-to-time, each node will send its own distance vector (DV) estimate to its neighbours. When a node receives new DV estimates, it will update its own DV using bellman-ford.
- The process is iterative, and each iteration is caused by either *a change in local link costs* or *a DV update message from a neighbour*. Note that each node only notifies neighbours if its DV changes, which causes their neighbours to notify their neighbours if there is change.
- See examples of this algorithm in action as well as the related node tables on **slides 5-25** and **slide 5-26**.
- If a link cost changes, a node will, when it detects the link cost change, update its own routing info and recalculate its distance vectors. If the DV changes, it will notify its neighbours.
- Note that *bad news travels slow* because the routers do not know the next optimal path if one link cost increases. This is known as the **count-to-infinity problem**.
- Under normal conditions, the distance vectors will converge to the actual least cost distance.

5.2.5 Comparison of LS and DV algorithms

- Message complexity:
  - **LS**: with n nodes and E links, $O(nE)$ messages are sent.
  - **DV**: Exchanges are only done between neighbours. Thus the convergence time varies.
- Speed of convergence:
  - **LS**: $O(n^2)$ algorithm requires $O(nE)$ messages.
  - **EV**: Convergence time varies because there may be routing loops, and the count-to-infinity problem may occur.
- Robustness (what happens if router malfunctions):
  - **LS**: Node can advertise incorrect link cost. It won't affect other nodes cause nodes only compute their own routing table.
  - **EV**: DV nodes can advertise incorrect path cost. Each node's table is used by others, so it could cause errors throughout the network.