CSE/EE 461

Link State Routing

Last Time …

• Routing Algorithms
  – Introduction
  – Distance Vector routing (RIP)

This Lecture

• Routing Algorithms
  – Link State routing (OSPF)
Why have two protocols?

- DV: “Tell your neighbors about the world.”
  - Easy to get confused ("the telephone game")
  - Simple but limited, costly and slow
    - 15 hops is all you get. (makes it faster to loop to infinity)
    - Periodic broadcasts of large tables
    - Slow convergence due to ripples and hold down
- LS: “Tell the world about your neighbors.”
  - Harder to get confused ("the nightly news")
  - More complicated
    - As many hops as you want
    - Faster convergence (instantaneous update of link state changes)
    - Able to impose global policies in a globally consistent way
      - Richer cost model, load balancing

Link State Routing

- Same assumptions/goals, but different idea than DV:
  - Tell all routers the topology and have each compute best paths
  - Two phases:
    1. Topology dissemination (flooding)
      - New News travels fast.
      - Old News should eventually be forgotten
    2. Shortest-path calculation (Dijkstra’s algorithm)
      - $n \log n$

Flooding

- Each router maintains link state database and periodically sends link state packets (LSPs) to neighbor
  - LSPs contain [router, neighbors, costs]
- Each router forwards LSPs not already in its database on all ports except where received
  - Each LSP will travel over the same link at most once in each direction
- Flooding is fast, and can be made reliable with acknowledgments
Example

- LSP generated by X at T=0
- Nodes become yellow as they receive it

![Graph showing LSP updates]

Complications

- When link/router fails need to remove old data. How?
  - LSPs carry sequence numbers to determine new data
  - Send a new LSP with cost infinity to signal a link down
- What happens if the network is partitioned and heals?
  - Different LS databases must be synchronized
  - A version number is used!

Shortest Paths: Dijkstra’s Algorithm

- N: Set of all nodes
- M: Set of nodes for which we think we have a shortest path
- s: The node executing the algorithm
- L(i,j): cost of edge (i,j) (inf if no edge connects)
- C(i): Cost of the path from ME to i.
- Two phases:
  - Initialize C(i) according to received link states
  - Compute shortest path to all nodes from s

- As link costs are symmetric, shortest path from A to B is also the shortest path from B to A.
**The Algorithm**

```plaintext
// Initialization
M = {s} // M is the set of all nodes considered so far.
For each n in N - {s}
    C(n) = L(s,n)

// Find Shortest paths
Forever {
    Unconsidered = N-M
    If Unconsidered == {} break
    M = M + {w} such that C(w) is the smallest in Unconsidered
    For each n in Unconsidered
        C(n) = MIN(C(n), C(w) + L(w,n))
}
```

**Dijkstra Example – After the flood**

The considered nodes are marked with a circle.

**Dijkstra Example – Post Initialization**

The considered nodes are marked with a circle.
Considering a Node

Cost updates of 8, 14, and 7

Pushing out the horizon

Cost updates of 13

Next Phase

Cost updates of 9
Considering the last node

Dijkstra Example – Done

Open Shortest Path First (OSPF)

- Most widely-used Link State protocol today
- Basic link state algorithms plus many features:
  - Authentication of routing messages
  - Extra hierarchy: partition into routing areas
    - Only bordering routers send link state information to another area
      - Reduces chatter.
    - Border router “summarizes” network costs within an area by making it appear as though it is directly connected to all interior routers
  - Load balancing
Cost Metrics

- How should we choose cost?
  - To get high bandwidth, low delay or low loss?
  - Do they depend on the load?

- Static Metrics
  - Hopcount is easy but treats OC3 (155 Mbps) and T1 (1.5 Mbps)
  - Can tweak result with manually assigned costs

- Dynamic Metrics
  - Depend on load; try to avoid hotspots (congestion)
  - But can lead to oscillations (damping needed)

Revised ARPANET Cost Metric

- Based on load and link
- Variation limited (3:1) and change damped
- Capacity dominates at low load; we only try to move traffic if high load

Key Concepts

- Routing uses global knowledge; forwarding is local
- Many different algorithms address the routing problem
  - We have looked at two classes: DV (RIP) and LS (OSPF)
- Challenges:
  - Handling failures / changes
  - Defining "best" paths
  - Scaling to millions of users