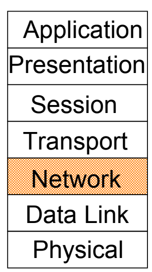
CSC 458 -- Lecture 5
Routing Protocols

Administrivia

- Projects:
  - #2 due today
  - #3 out today
- Homework:
  - #2 due next week
- Midterm:
  - One more class next week before midterm

This Time

- Focus
  - How do we calculate routes for packets?
  - Routing is a network layer function
- Routing Algorithms
  - Distance Vector routing (RIP)



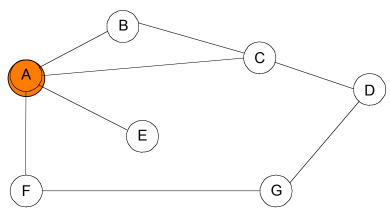
Forwarding and Routing

- Forwarding is the process that each router goes through for every packet to send it on its way
  - Involves local decisions
- Routing is the process that all routers go through to calculate the routing tables
  - Involves global decisions

What's in a Routing Table?

- The routing table at A, for example, lists at a minimum the next hops for the different destinations

Dest	Next Hop
B	
C	
D	
E	
F	
G	



Kinds of Routing Schemes

- Many routing schemes have been proposed/explored!

- Distributed or centralized
- Hop-by-hop or source-based
- Deterministic or stochastic
- Single or multi-path
- Static or dynamic route selection

- Internet is to the left ☺

Routing Questions

- How to choose best path?
  - Defining "best" is slippery
- How to scale to millions of users?
  - Minimize control messages and routing table size
- How to adapt to failures or changes?
  - Node and link failures, plus message loss
  - We will use distributed algorithms

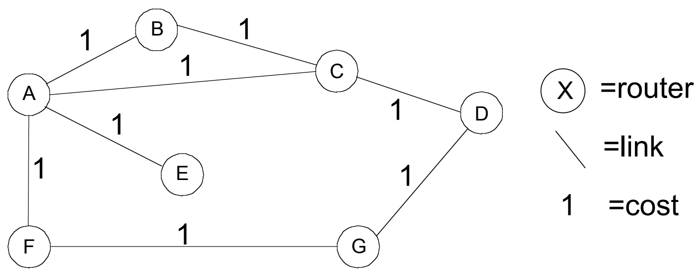
Some Pitfalls

- Using global knowledge is challenging
  - Hard to collect
  - Can be out-of-date
  - Needs to summarize in a locally-relevant way

- Inconsistencies in local /global knowledge can cause:
  - Loops (black holes)
  - Oscillations, esp. when adapting to load

Network as a Graph

- Routing is essentially a problem in graph theory



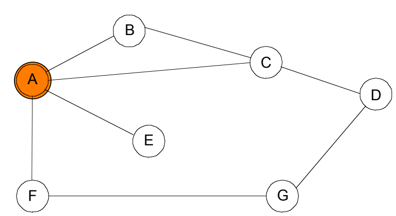
Distance Vector Routing

- Assume:
  - Each router knows only address/cost of neighbors
- Goal:
  - Calculate routing table of next hop information for each destination at each router
- Idea:
  - Tell neighbors about learned distances to all destinations

DV Algorithm

- Each router maintains a vector of costs to all destinations as well as routing table
  - Initialize neighbors with known cost, others with infinity
- Periodically send copy of distance vector to neighbors
  - On reception of a vector, if neighbors path to a destination plus neighbor cost is better, then switch to better path
    - update cost in vector and next hop in routing table
- Assuming no changes, will converge to shortest paths
  - But what happens if there are changes?

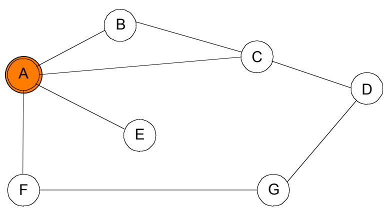
DV Example – Initial Table at A



Dest	Cost	Next
B		
C		
D		
E		
F		
G		

DV Example – Final Table at A

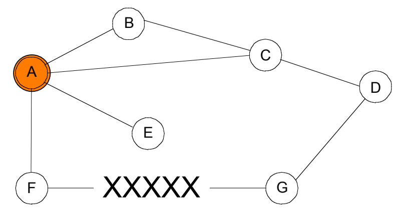
- Reached in a single iteration … simple example



Dest	Cost	Next
B		
C		
D		
E		
F		
G		

What if there are changes?

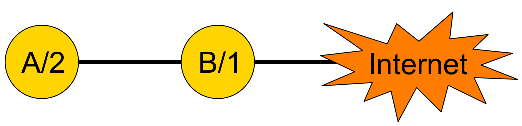
- One scenario: Suppose link between F and G fails
  1. F notices failure, sets its cost to G to infinity and tells A
  2. A sets its cost to G to infinity too, since it learned it from F
  3. A learns route from C with cost 2 and adopts it



Dest	Cost	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	3	C

Count To Infinity Problem

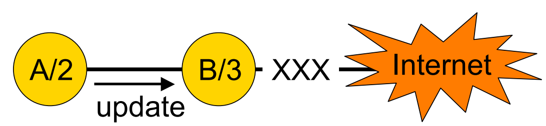
- Simple example
  - Costs in nodes are to reach Internet



- Now link between B and Internet fails …

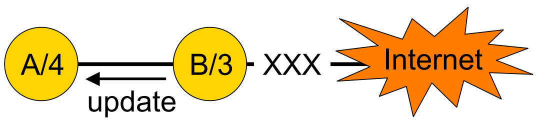
Count To Infinity Problem

- B hears of a route to the Internet via A with cost 2
- So B switches to the "better" (but wrong!) route



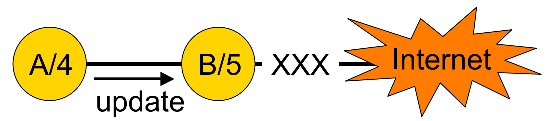
Count To Infinity Problem

- A hears from B and increases its cost



Count To Infinity Problem

- B hears from A and (surprise) increases its cost
- Cycle continues and we "count to infinity"



- Packets caught in the crossfire loop between A and B

Split Horizon

- Solves trivial count-to-infinity problem
- Router never advertises the cost of a destination back to to its next hop – that's where it learned it from!
- Poison reverse: go even further – advertise back infinity
- However, DV protocols still subject to the same problem with more complicated topologies
  - Many enhancements suggested

Routing Information Protocol (RIP)

- DV protocol with hop count as metric
  - Infinity value is 16 hops; limits network size
  - Includes split horizon with poison reverse
- Routers send vectors every 30 seconds
  - With triggered updates for link failures
  - Time-out in 180 seconds to detect failures
- RIPv1 specified in RFC1058
  - www.ietf.org/rfc/rfc1058.txt
- RIPv2 (adds authentication etc.) in RFC1388
  - www.ietf.org/rfc/rfc1388.txt

RIP is an “Interior Gateway Protocol”

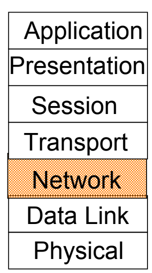
- Suitable for small- to medium-sized networks
  - such as within a campus, business, or ISP
- Unsuitable for Internet-scale routing
  - hop count metric poor for heterogeneous links
  - 16-hop limit places max diameter on network
- Later, we’ll talk about “Exterior Gateway Protocols”
  - used between organizations to route across Internet

Key Concepts

- Routing is a global process, forwarding is local one
- The Distance Vector algorithm and RIP
  - Simple and distributed exchange of shortest paths.
  - Weak at adapting to changes (loops, count to infinity)

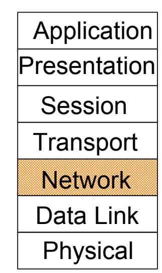
Last Time …

- 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

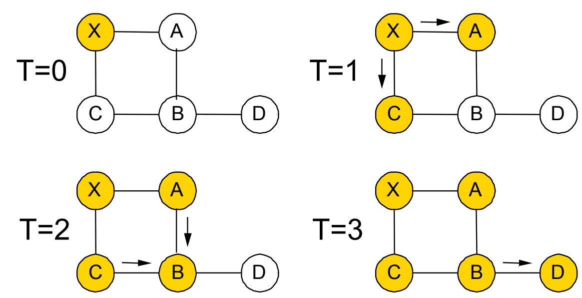
- Same assumptions/goals, but different idea than DV:
  - Tell all routers the topology and have each compute best paths
  - Two phases:
    1. Topology dissemination (flooding)
       - New News travels fast.
       - Old News should eventually be forgotten
    2. Shortest-path calculation (Dijkstra’s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

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



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!

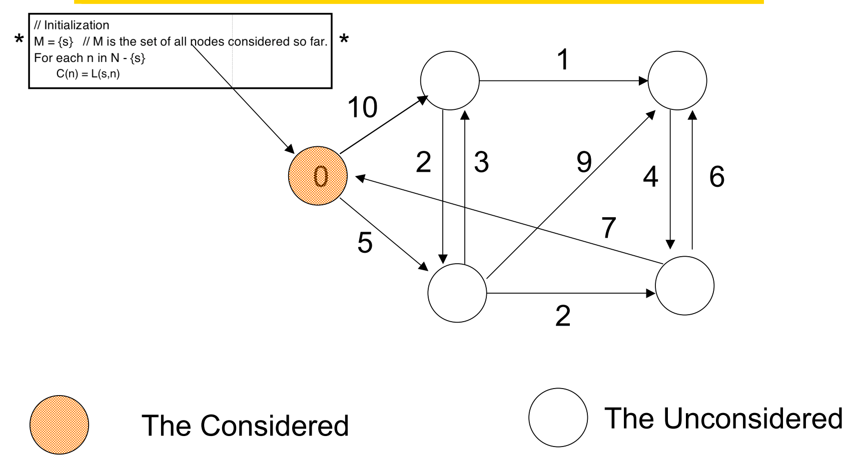
Shortest Paths: Dijkstra's Algorithm

- *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(n) 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

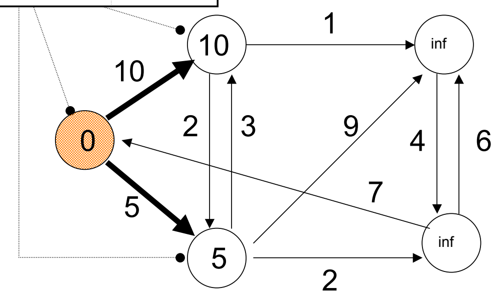
```
// Initialization
M = {s}   // M is the set of all nodes considered so far.
For each n in N - {s}
      C(n) = L(s,n)

// 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



Dijkstra Example – Post Initialization

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

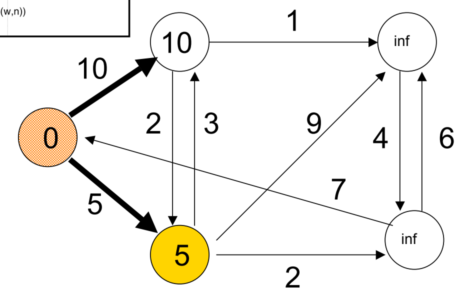


The Considered

The Unconsidered.

Considering a Node

```
// 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))
}
```



Cost updates of 8,14, and 7

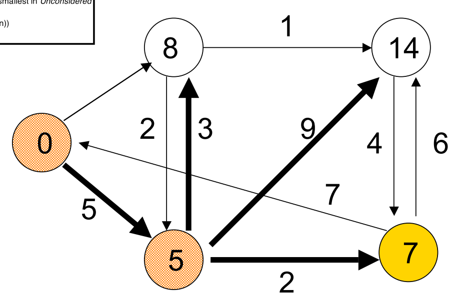
The Considered

The Unconsidered.

The Under Consideration *(w)*.

Pushing out the horizon

```
// 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))
}
```



Cost updates of 13

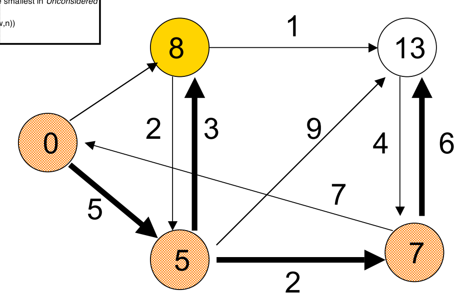
The Considered

The Unconsidered.

The Under Consideration *(w)*.

Next Phase

```
// 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))
}
```



Cost updates of 9

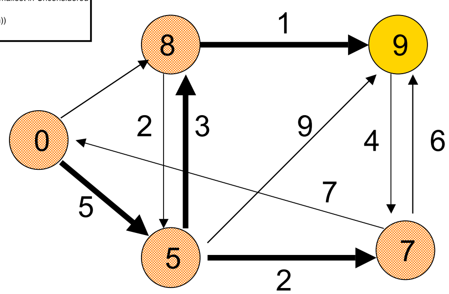
The Considered

The Unconsidered.

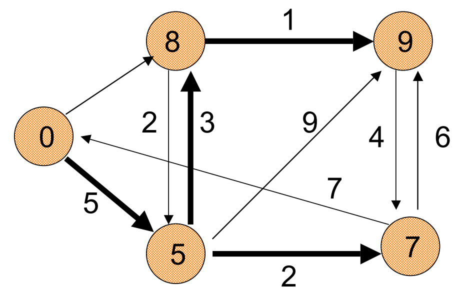
The Under Consideration *(w)*.

Considering the last node

```
// 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 – Done



Open Shortest Path First (OSPF)

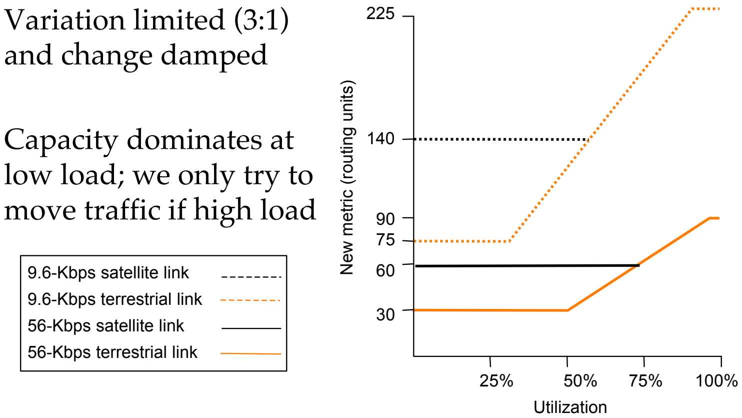
- Most widely-used Link State protocol today
- Basic link state algorithms plus many features:
  - Authentication of routing messages
  - Extra hierarchy: partition into routing areas
    - Only bordering routers send link state information to another area
      - Reduces chatter.
      - Border router "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) same
  - 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