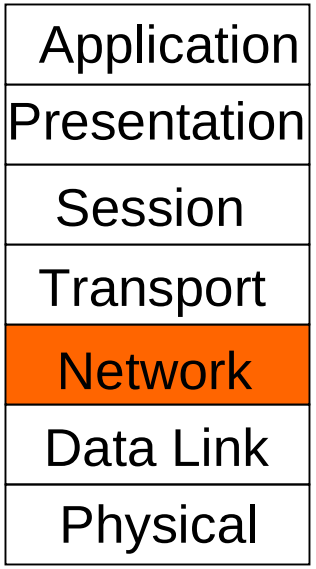
CSE/EE 461 – Lecture 7

David Wetherall

djw@cs.washington.edu

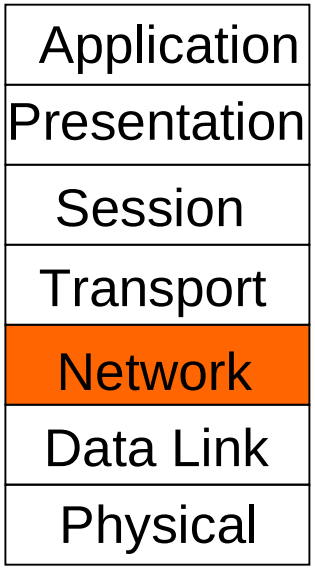
Last Time

- Introduction to the Network layer
  - Internetworks
  - Datagram and virtual circuit services
  - Internet Protocol (IP) packet format
- The Network layer
  - Provides end-to-end data delivery between networks, solves the problem of routing
  - Issues of scale and heterogeneity



This Time

- Focus
  - How do we calculate routes for packets?
- Routing Algorithms
  - Introductory material
  - Distance Vector routing (RIP)
  - Link State routing (OSPF)
  - Cost Metrics



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

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

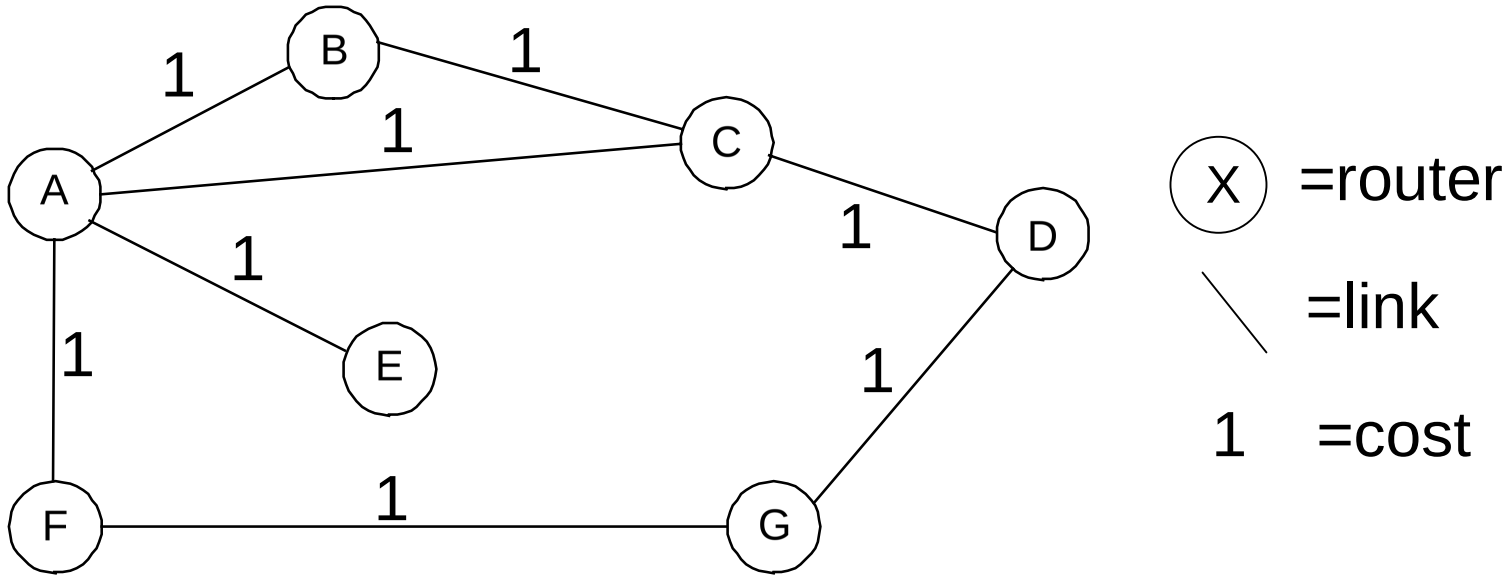
- How to choose best path?
  - Defining “best” is slippery
- How to scale to millions of users?
  - Minimize control messages and routing table size
- How to adapt to failures or changes?
  - Node and link failures, plus message loss
  - We will use distributed algorithms

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  - Can be out-of-date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



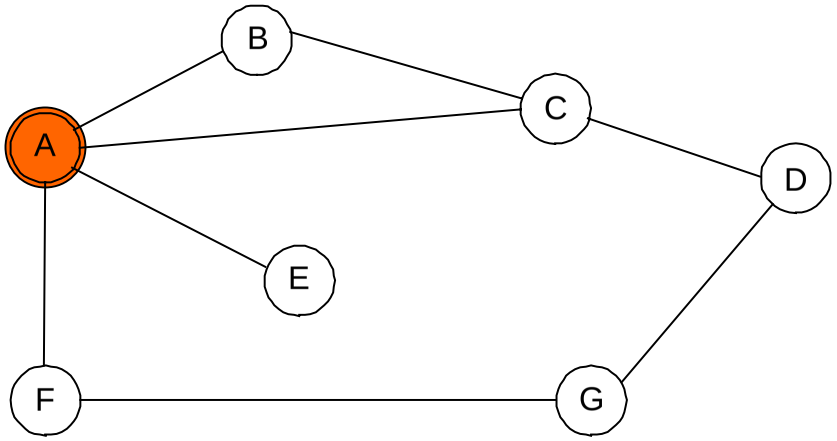
Distance Vector Routing

- 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?

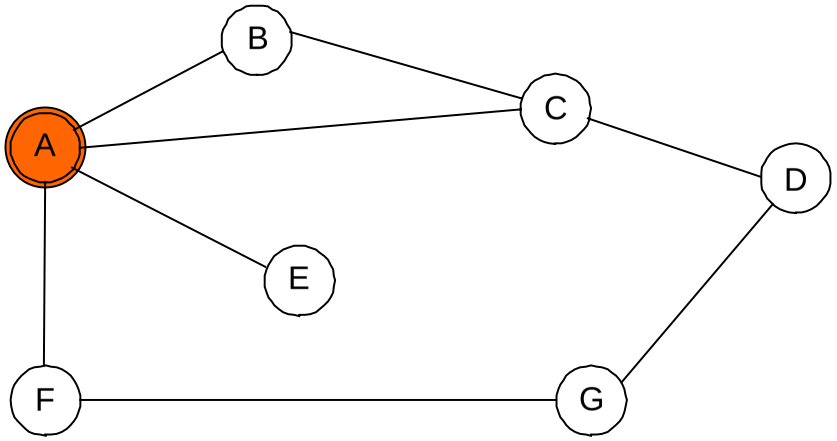
DV Example – Initial Table at A



Dest	Cost	Next
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	E
G	∞	-

DV Example – Final Table at A

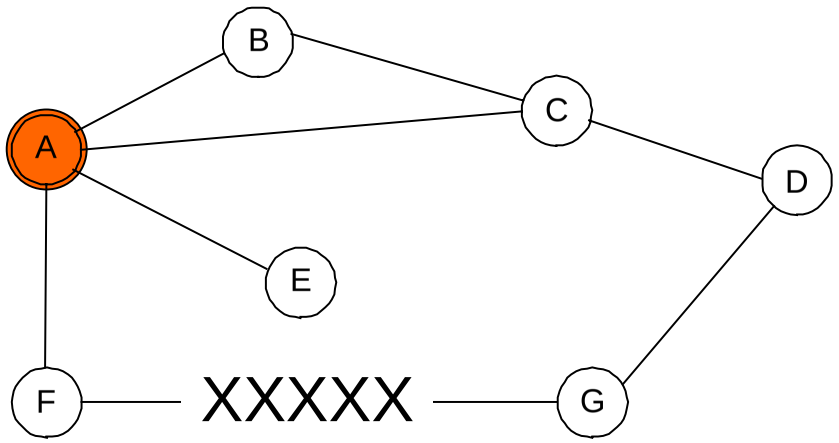
- Reached in a single iteration ... simple example



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

What if there are changes?

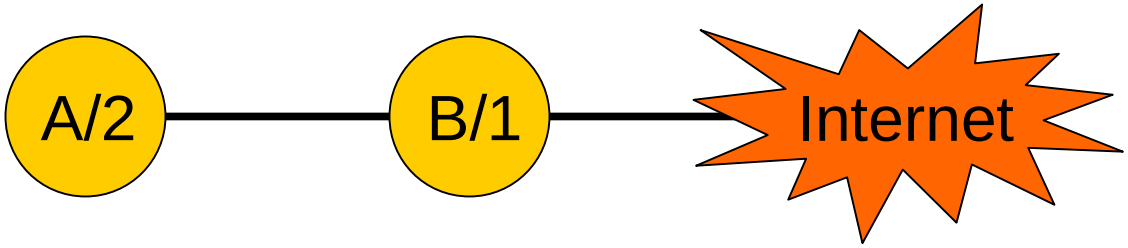
- One scenario: Suppose link between F and G fails
  - F notices failure, sets its cost to G to infinity and tells A
  - A sets its cost to G to infinity too, since it learned it from F
  - A learns route from C with cost 2 and adopts it



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

Count To Infinity Problem

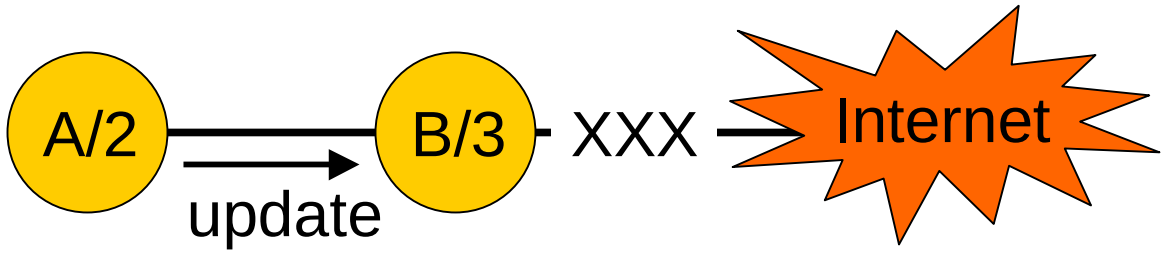
- Simple example
  - Costs in nodes are to reach Internet



- Now link between B and Internet fails ...

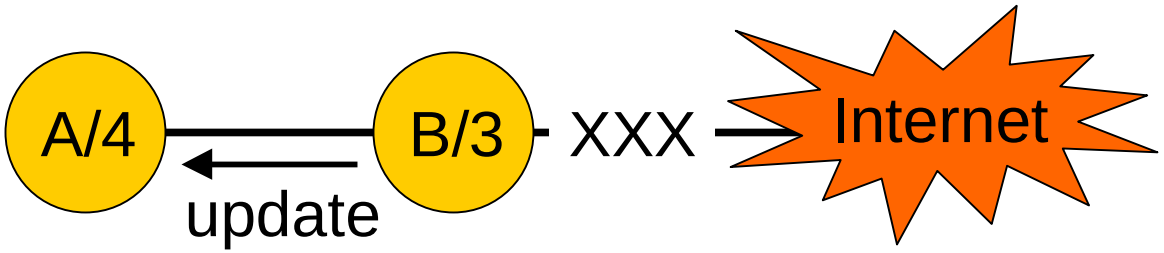
Count To Infinity Problem

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



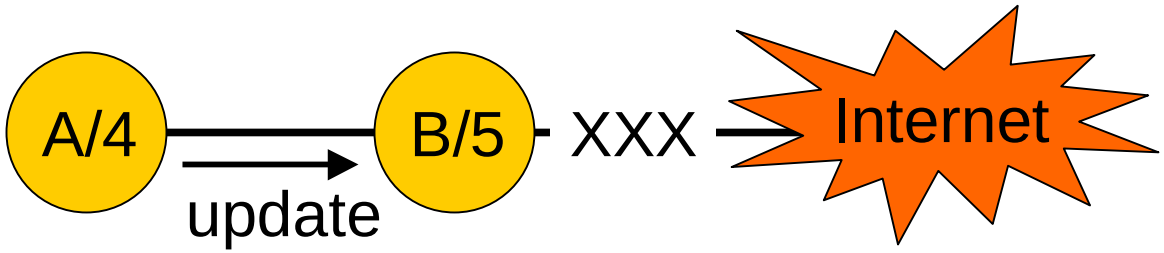
Count To Infinity Problem

- 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

Link State Routing

- Same assumptions/goals, but different idea:
  - Tell all routers the topology and have each compute best paths
  - Two phases:
    - Topology dissemination (flooding)
    - Shortest-path calculation (Dijkstra's algorithm)

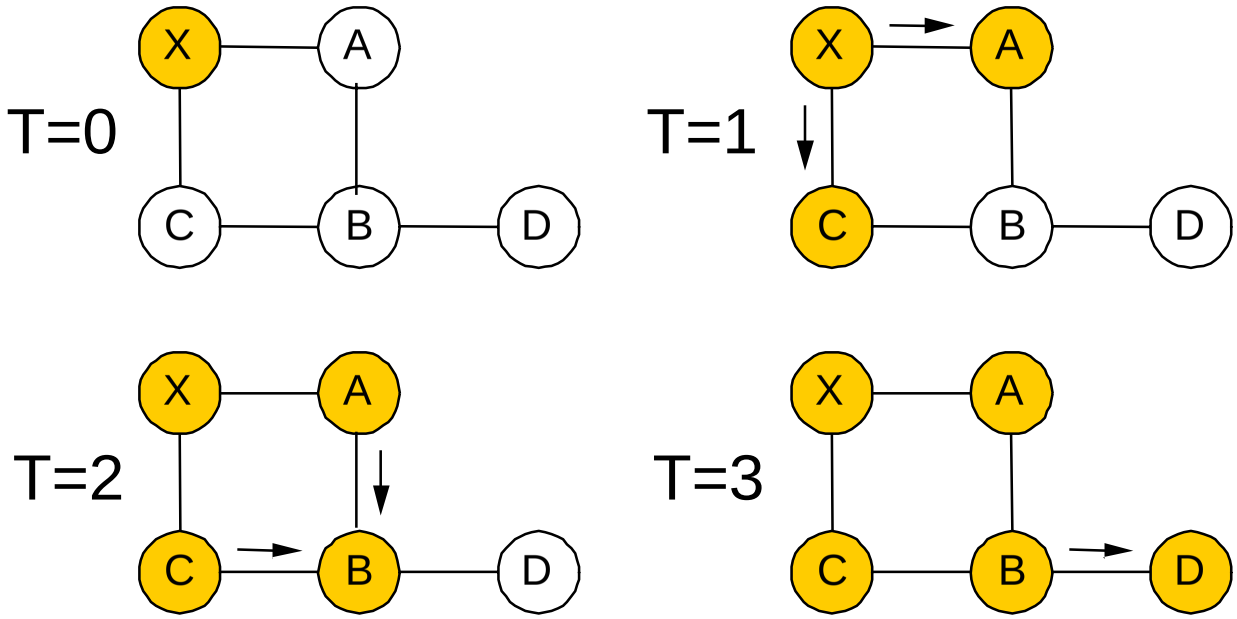
- Why?
  - In DV, routers hide their computation, making it difficult to decide what to use when there are changes
  - With LS, faster convergence and hopefully better stability
  - It is more complex though

Flooding

- Each router maintains link state database and periodically sends link state packets (LSPs) to neighbor
  - 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



Complication: Sequence Numbers

- 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 when a router fails and restarts?
  - What sequence number should it use? Don't want data ignored.
  - One option: age LSPs and send with "cost 0" to purge
    - Router can listen at startup to learn right sequence number
- What happens if the network is partitioned and heals?
  - Different LS databases must be synchronized
  - A version number is used!

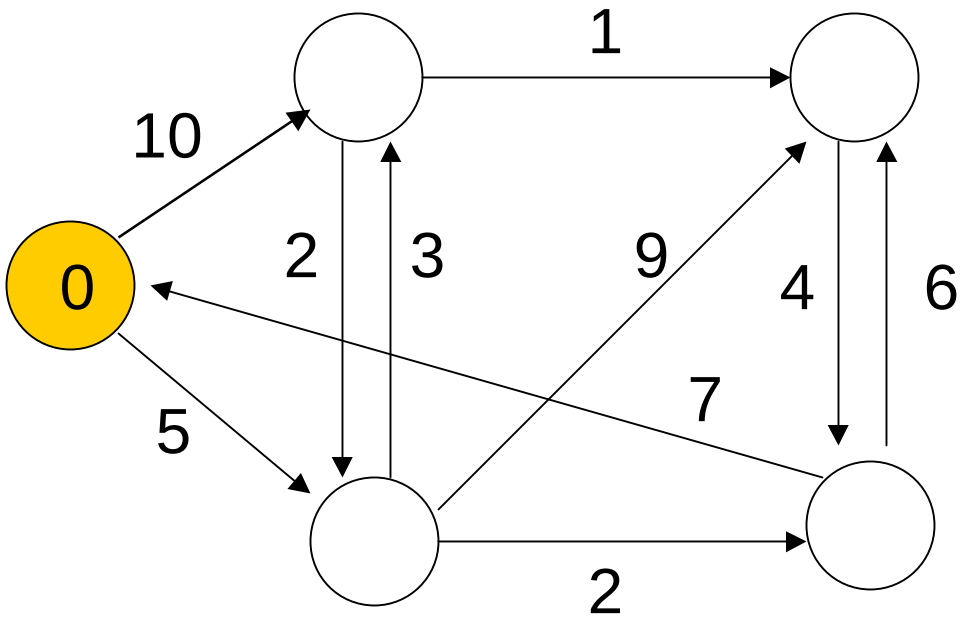
Dijkstra's Algorithm

- Graph algorithm for single-source shortest path

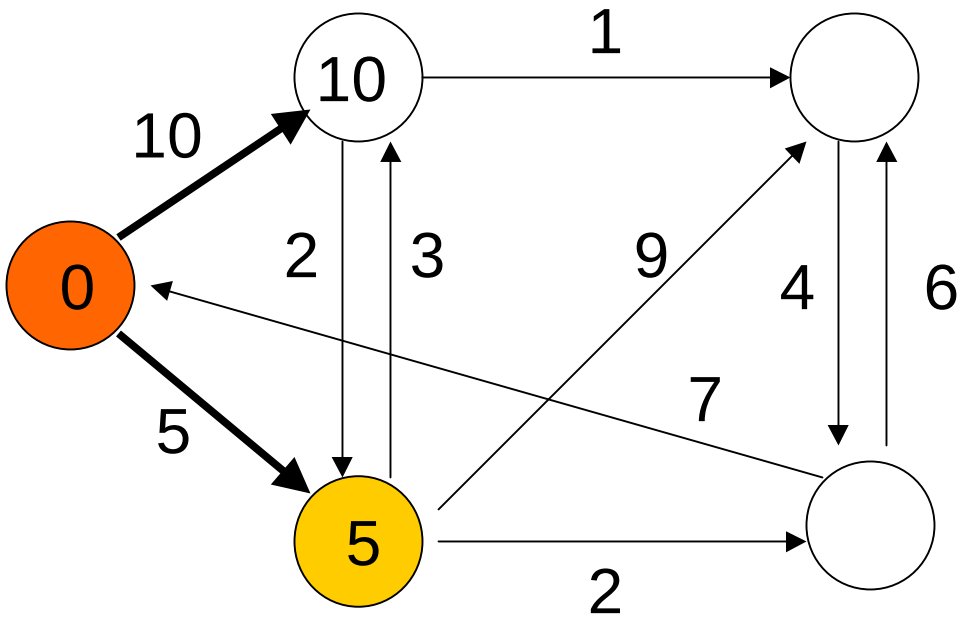
```
S ← {}
Q ← <all nodes keyed by distance>
While Q != {}
        u ← extract-min(Q)
        S ← S plus {u}
        for each node v adjacent to u
                "relax" the cost of v
```

← u is done

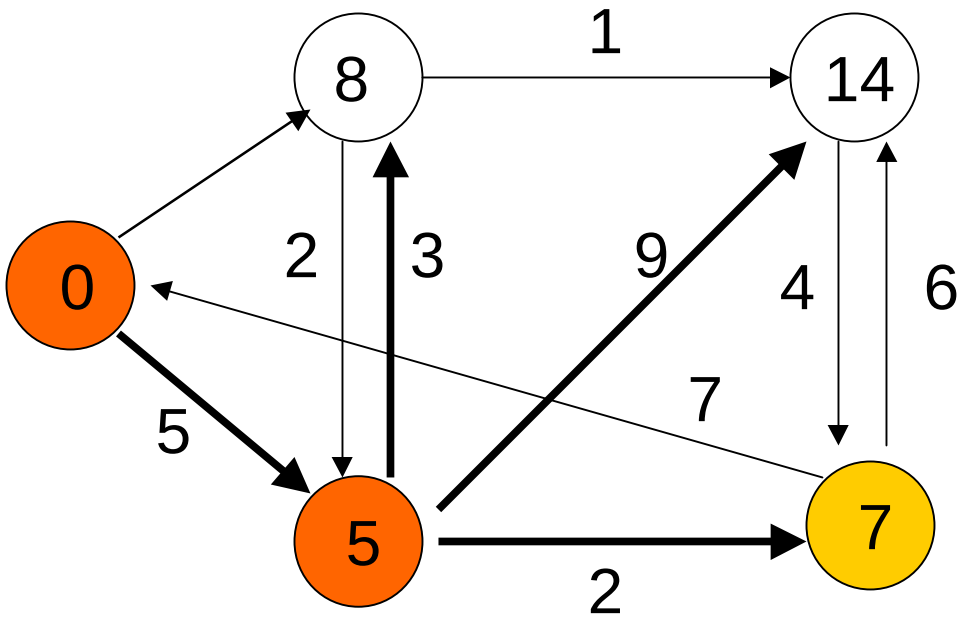
Dijkstra Example – Step 1



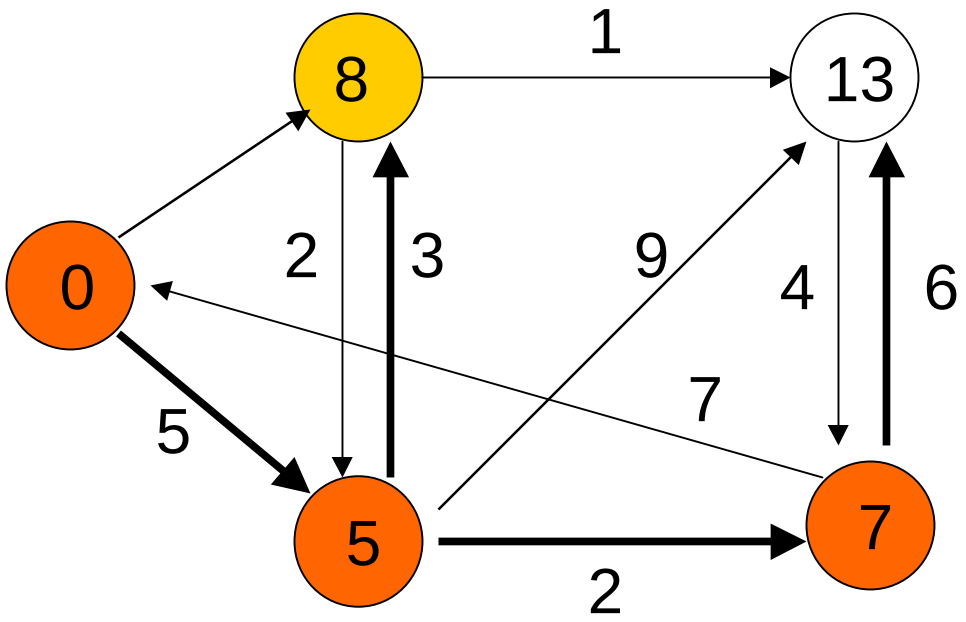
Dijkstra Example – Step 2



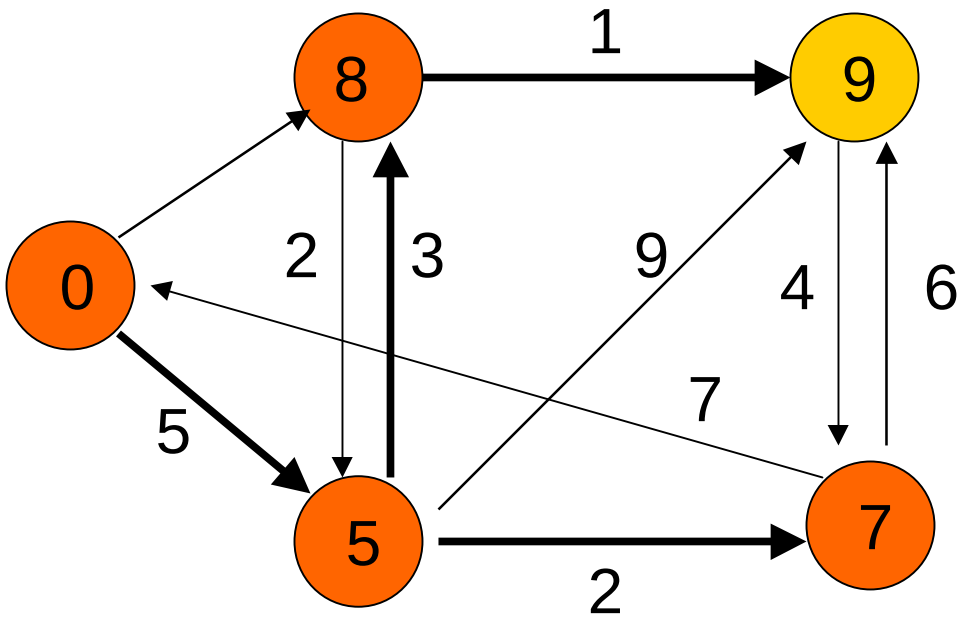
Dijkstra Example – Step 3



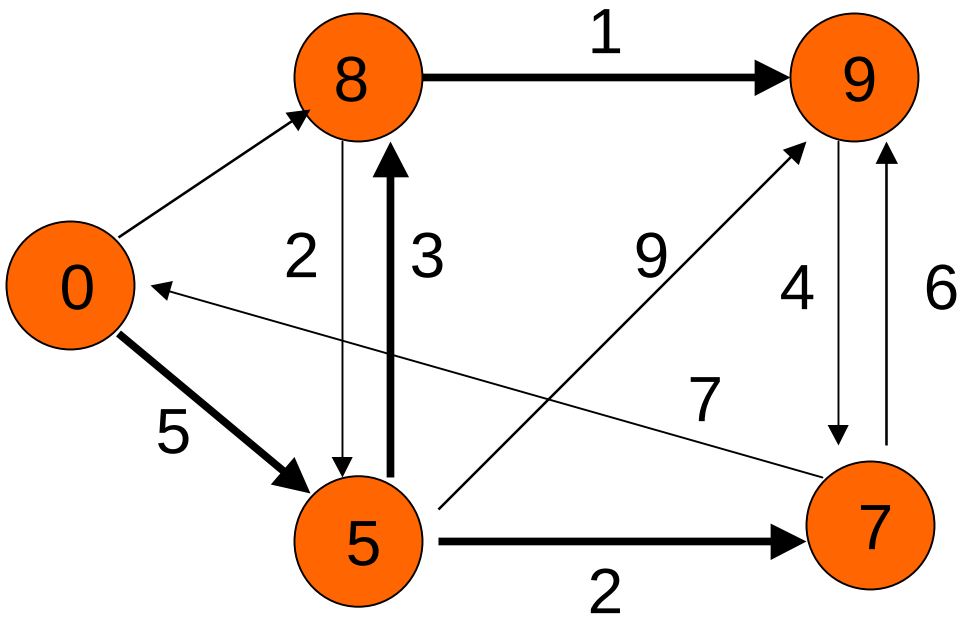
Dijkstra Example – Step 4



Dijkstra Example – Step 5



Dijkstra Example – Done



Open Shortest Path First (OSPF)

- Most widely-used Link State protocol today

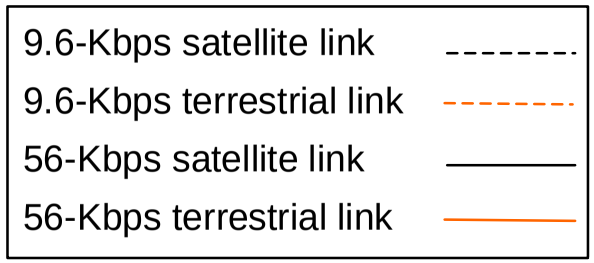
- Basic link state algorithms plus many features:
  - Authentication of routing messages
  - Extra hierarchy: partition into routing areas
  - Load balancing: multiple equal cost routes

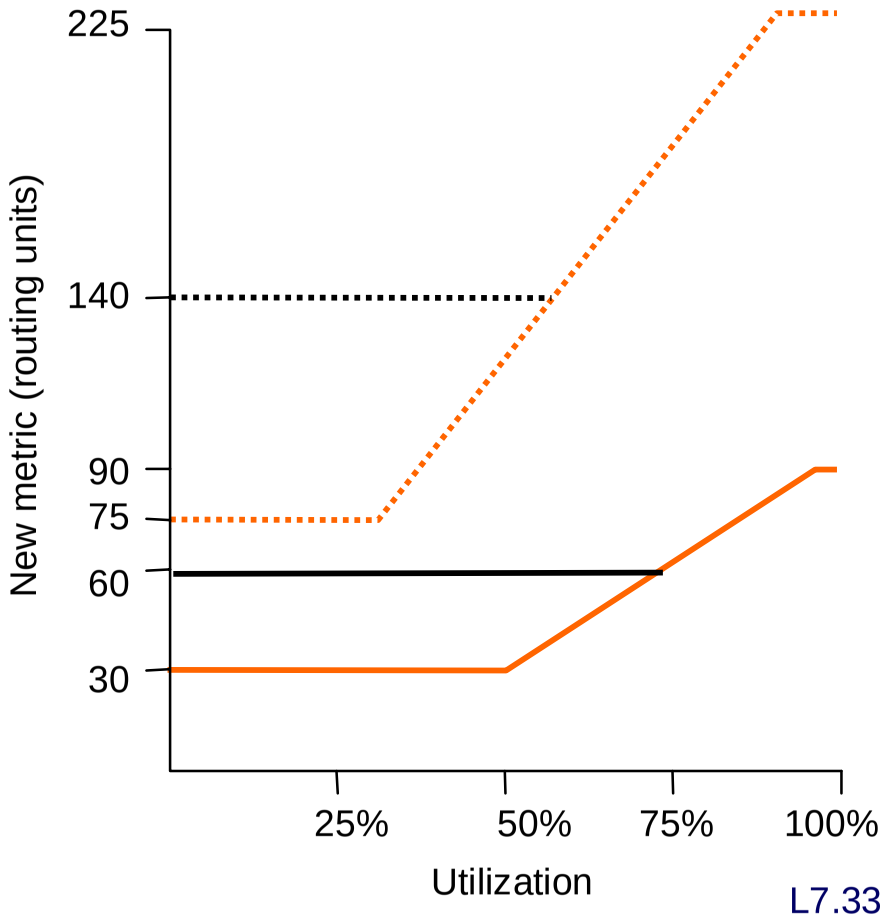
Cost Metrics

- 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 DV (RIP) and LS (OSPF)
- Challenges:
  - Handling failures/changes
  - Scaling to millions of users
  - Defining “best” paths