Routing in a network

- Focus is small to medium size networks, not yet the Internet
- Overview
  - Distance vector algorithm (RIP)
  - Link state algorithm (OSPF)
- Then
  - Talk about routing more generally
  - E.g., cost metrics, stability, multi-path, scalability, …

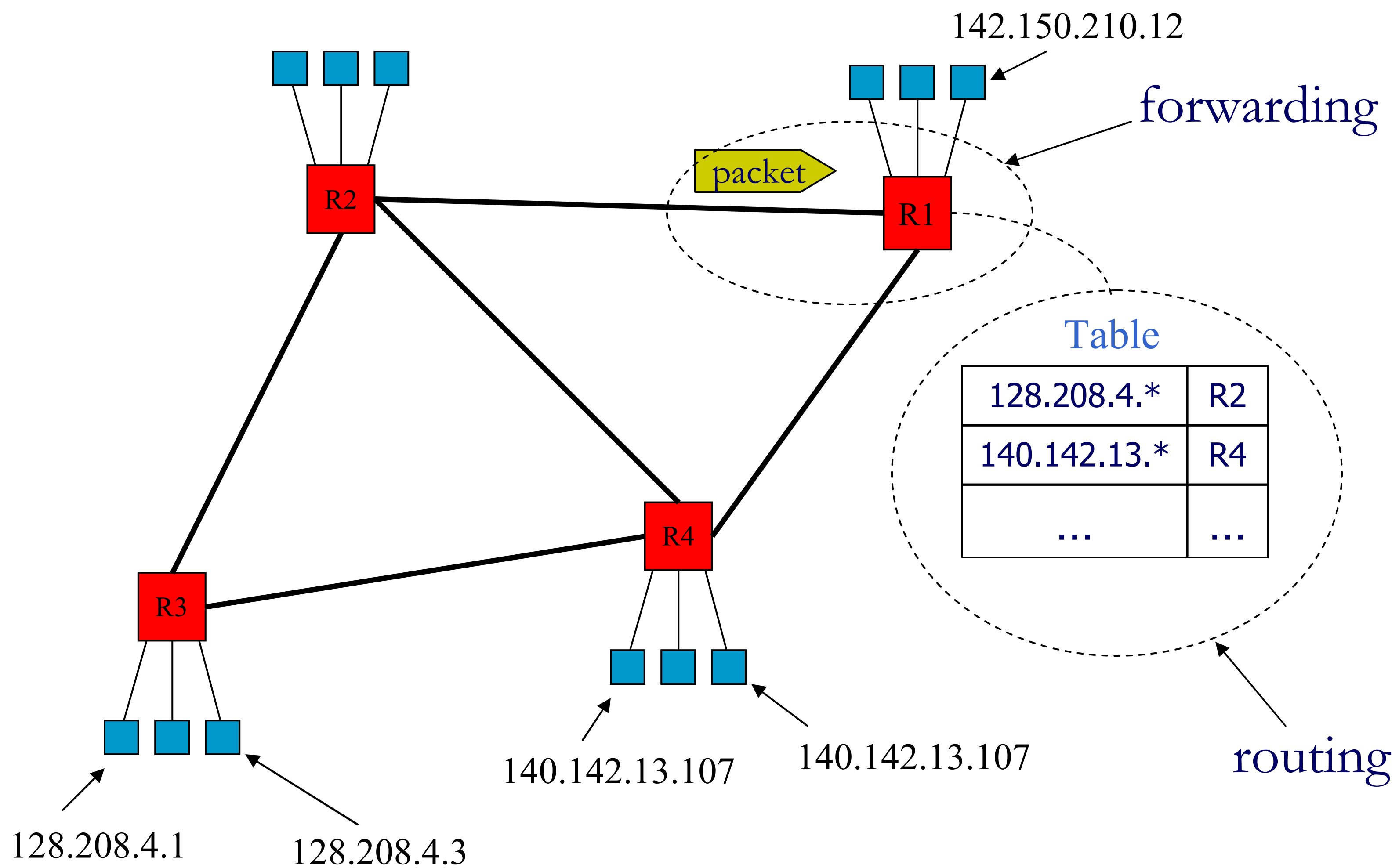
Application
Presentation
Session
Transport
Network
Data Link
Physical

Routing versus Forwarding

- Routing is the process by which all nodes exchange control messages to calculate the *routes* packets will follow
  - Involves *global* decisions; emphasis is *correctness*
  - Nodes build a routing table that models the global network

- Forwarding is the process by which a node examines packets and sends them along their *paths* through the network
  - Involves *local* decisions; emphasis is *efficiency*
  - Nodes distill a forwarding table from their routing table keyed by packet attributes, e.g., address

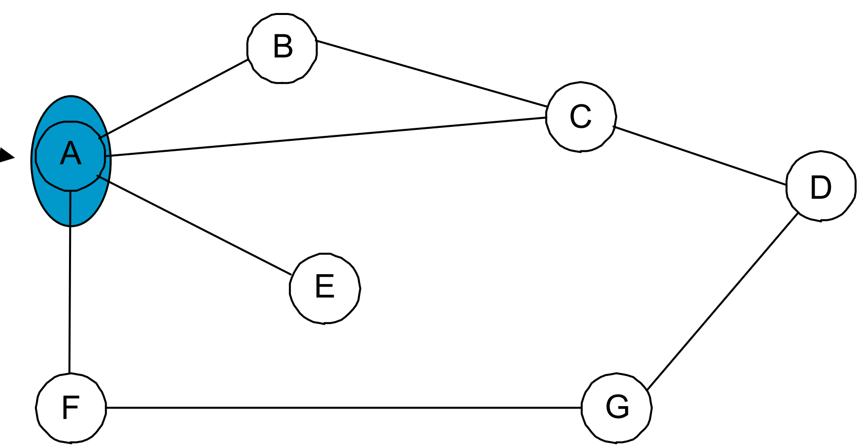
Routing versus Forwarding



A Simple "Routing Table"

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

Dest	Next Hop
B	B
C	C
D	C
E	E
F	F
G	F



- Other possible keys:
  - Destination and source; path identifier; service class

Some Pitfalls

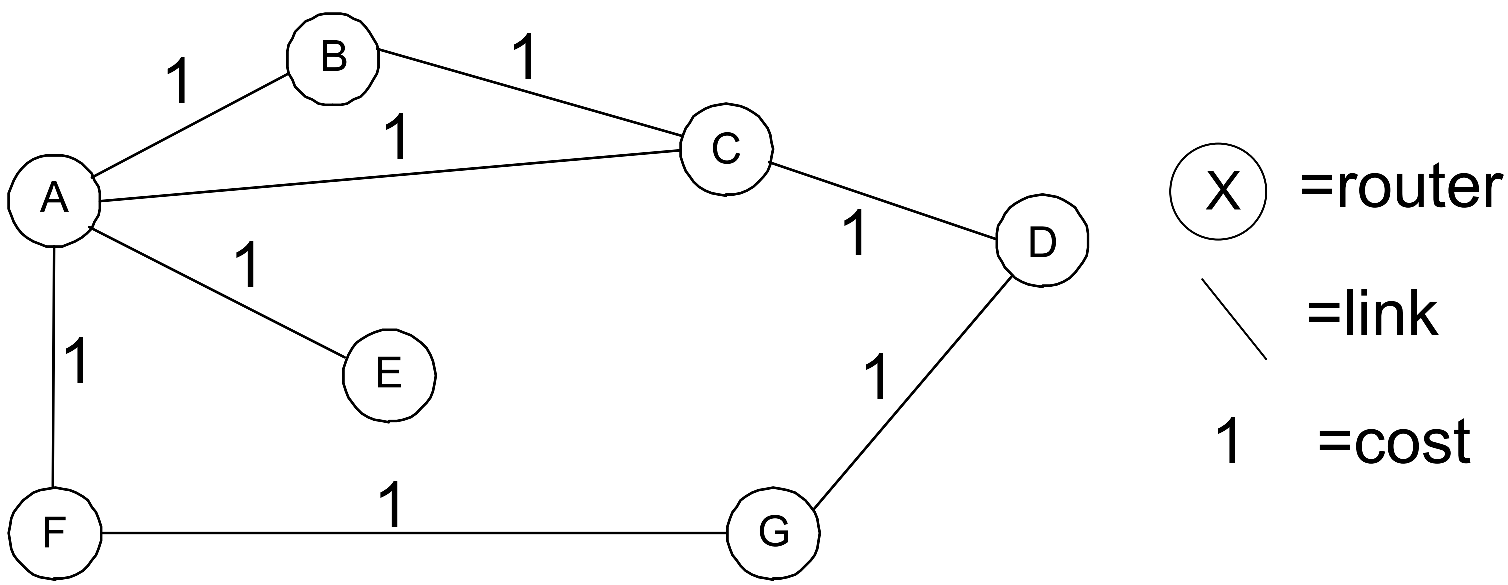
- Remember the network is a distributed system
  - No central authority just tells nodes what to do

- 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  - Can be out-of-date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. Remember Bellman-Ford Single-Source Shortest Path?



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

- All nodes do this in parallel; its a distributed routing computation

DV Algorithm

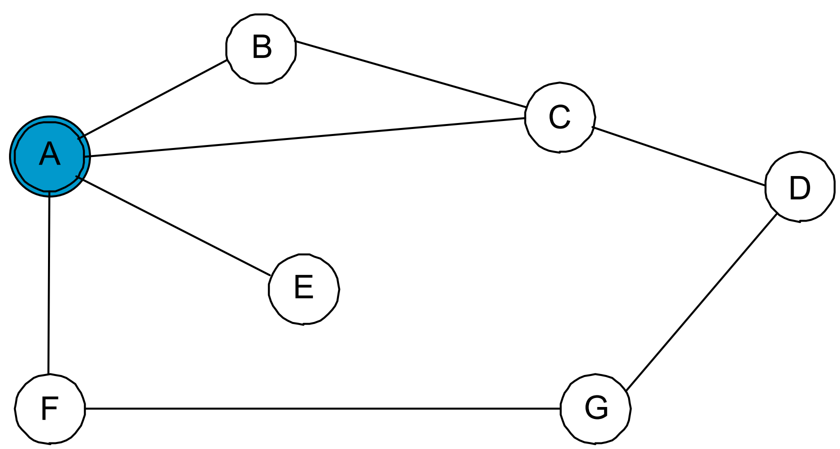
- Each router maintains a vector of costs to all destinations as well as routing table
  - Initialize neighbors with known cost, others with infinity

- Periodically send copy of distance vector to neighbors

- On reception of a vector, if your neighbor's path to a destination plus cost to that neighbor cost is better
  - Update the cost and next-hop in your outgoing vectors

- Assuming no changes, will converge to shortest paths
  - But what happens if there are changes?

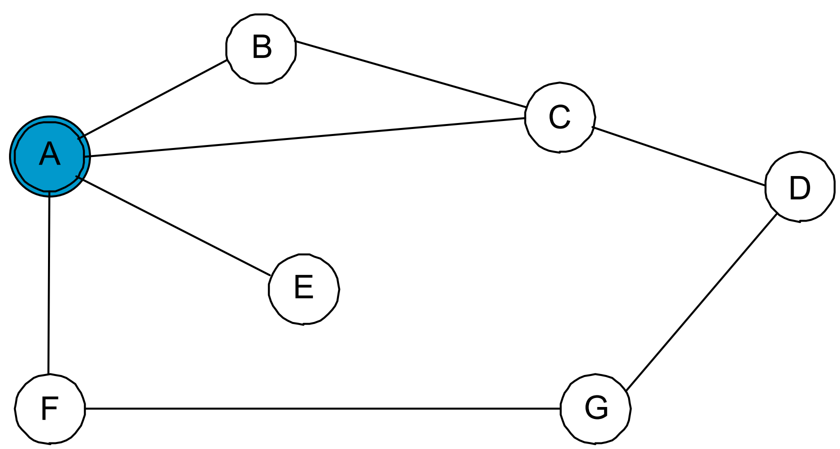
DV Example – Initial Table at A



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

DV Example – Final Table at A

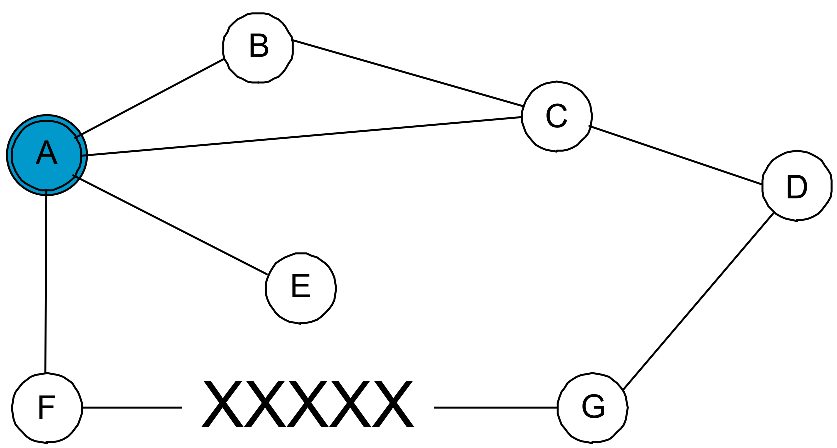
This simple example converges after one iteration



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

What if there are changes?

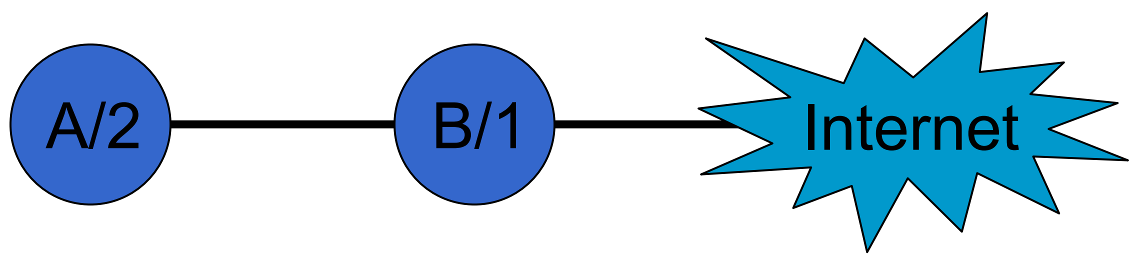
- One scenario: Suppose link between F and G fails
  1. F notices failure, sets its cost to G to infinity and tells A
  2. A sets its cost to G to infinity too, since it learned it from F
  3. A learns route from C with cost 2 and adopts it



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

But failures can cause problems …

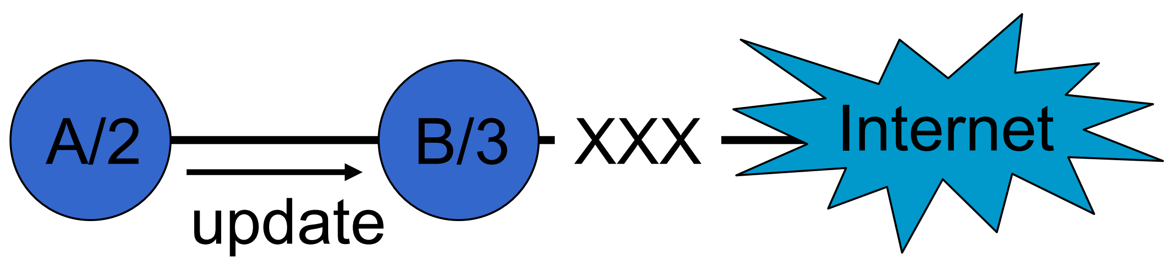
- Imagine two nodes want to maintain routes to the Internet.



What happens when the 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



- (work through the next few steps …)

Only partial solutions

- "Split Horizon" solves trivial count-to-infinity problem
  - Router never advertises the cost of a destination back 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 kinds of transient problems in more complicated scenarios
  - Many enhancements suggested
  - Fundamentally distributed computation is hard

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 than DV:
  - Tell all routers the topology and have each compute best paths
  - Two phases:
    1. Topology dissemination (flooding)
    2. Shortest-path calculation (Dijkstra’s algorithm)

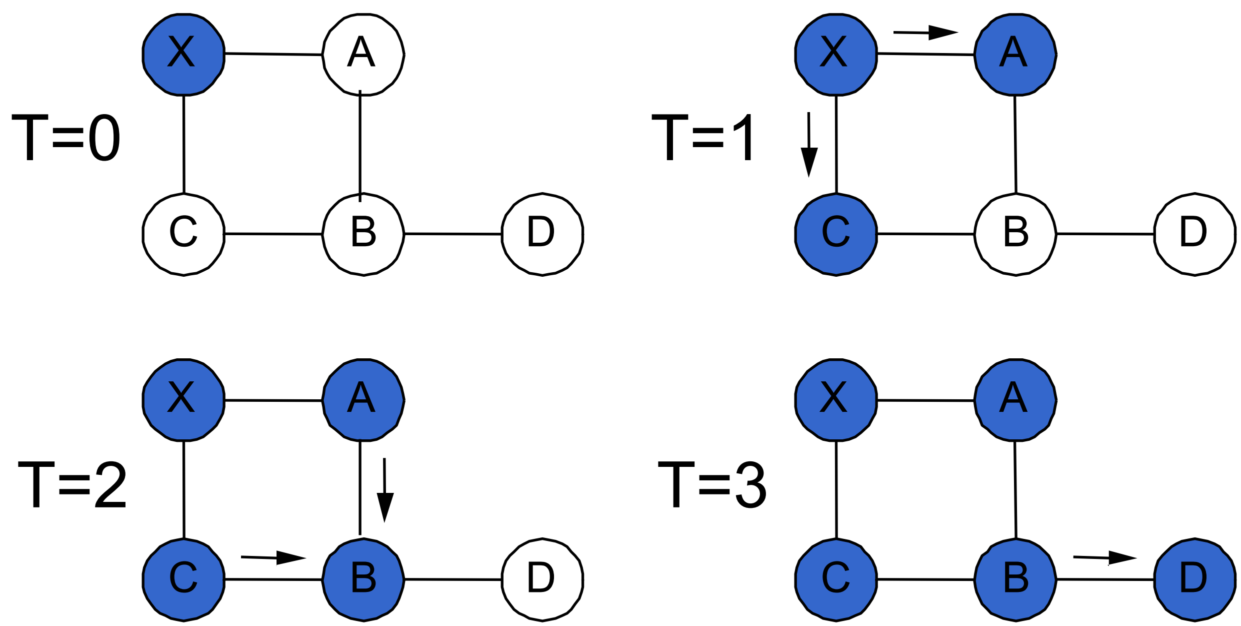
- Why?
  - In DV, routers hide their computation, making it difficult to decide what to use when there are changes
  - With LS, faster convergence and hopefully better stability
  - It is more complex though …

Flooding

- Each router maintains link state database and periodically sends link state packets (LSPs) to neighbor
  - 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 when a router fails and restarts?
  - What sequence number should it use? Don't want data ignored.
  - One option: age LSPs and send with "TTL 0" to purge
- What happens if the network is partitioned and heals?
  - Different LS databases must be synchronized
  - A version number is used!

A deeper look at sequence numbers …

- Complications highlight robustness issues …

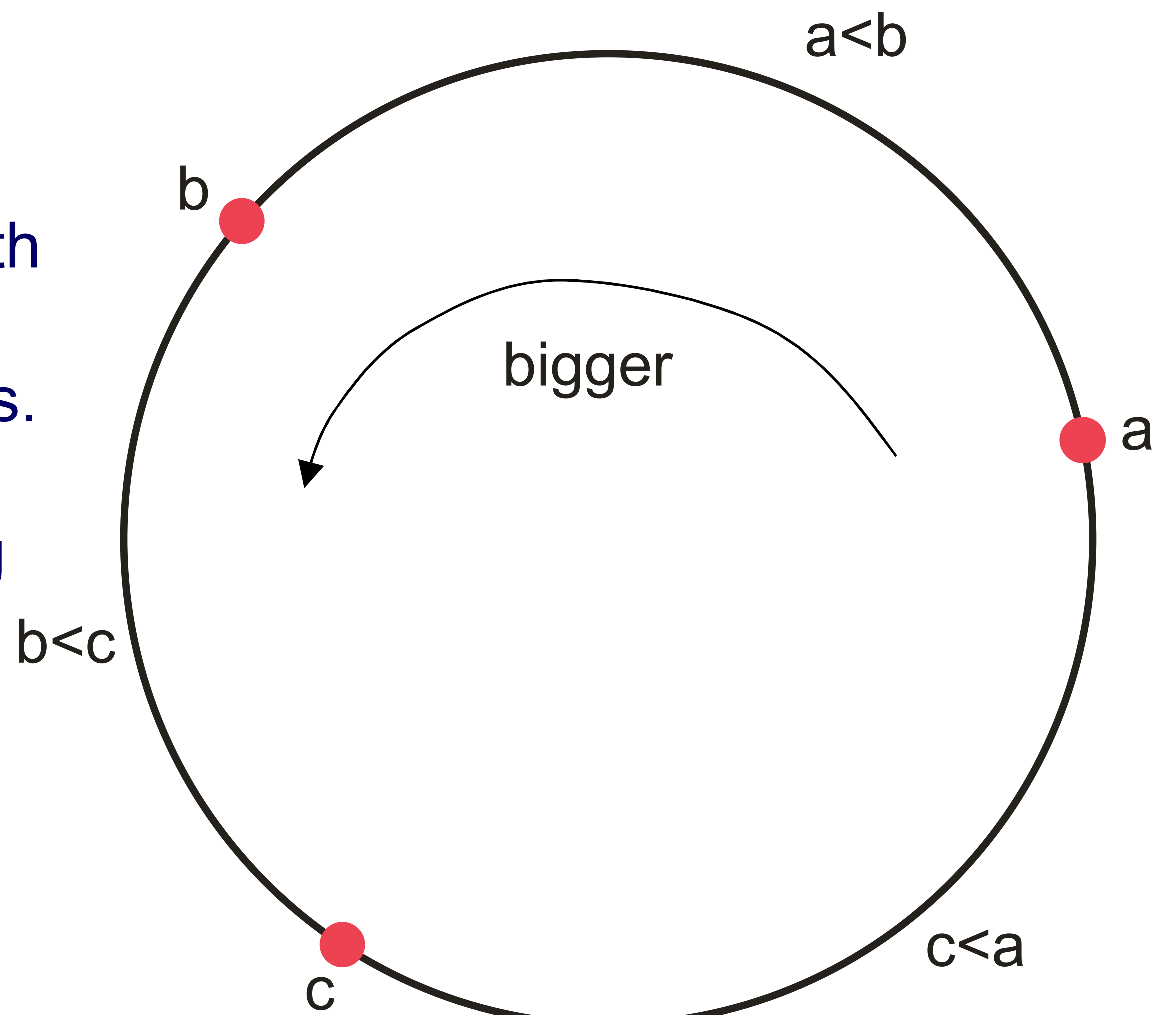
- How do we handle sequence number exhaustion?

- Make sequence number space very large?
  - There's a problem …

- Use module arithmetic?
  - There's a problem …

ARPANet failed in 1981, because…

A dying router emitted 3 LSPs with 3 very unlucky sequence numbers. Soon, the entire network was doing nothing but propagating these same three LSPs everywhere.



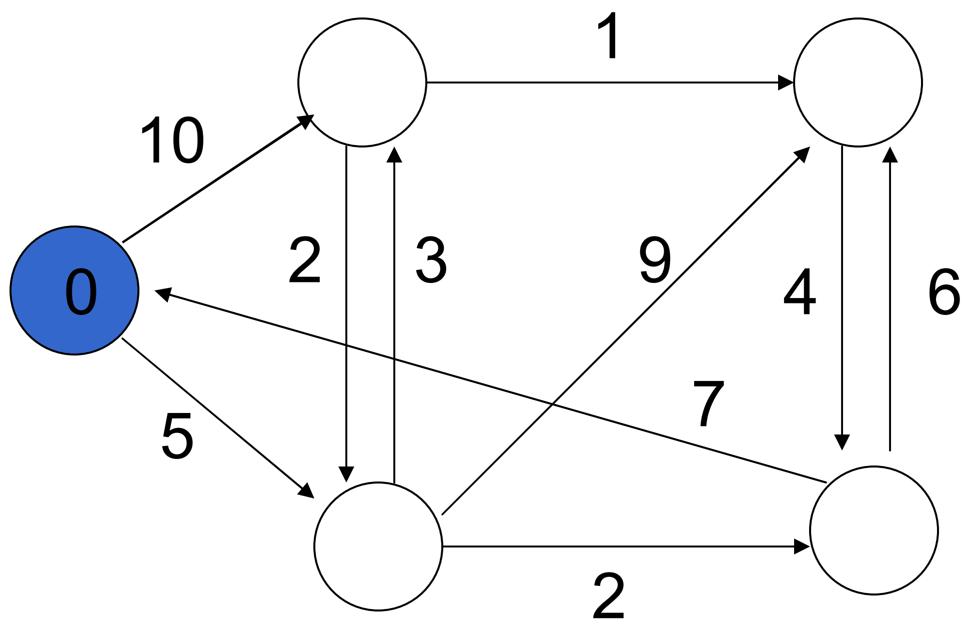
Shortest Paths: Dijkstra's Algorithm

- Graph algorithm for single-source shortest path

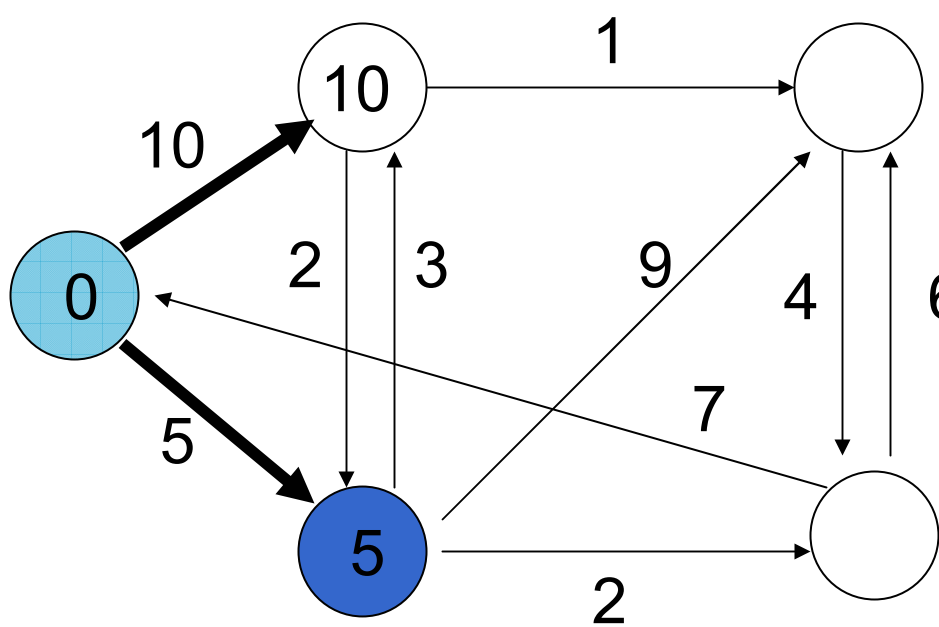
```
S ← {}
Q ← <all nodes keyed by distance>
While Q != {}
        u ← extract-min(Q)
        S ← S plus {u}             ←u is done, add to shortest paths
        for each node v adjacent to u
                "relax" the cost of v
```

←u is done, add to shortest paths

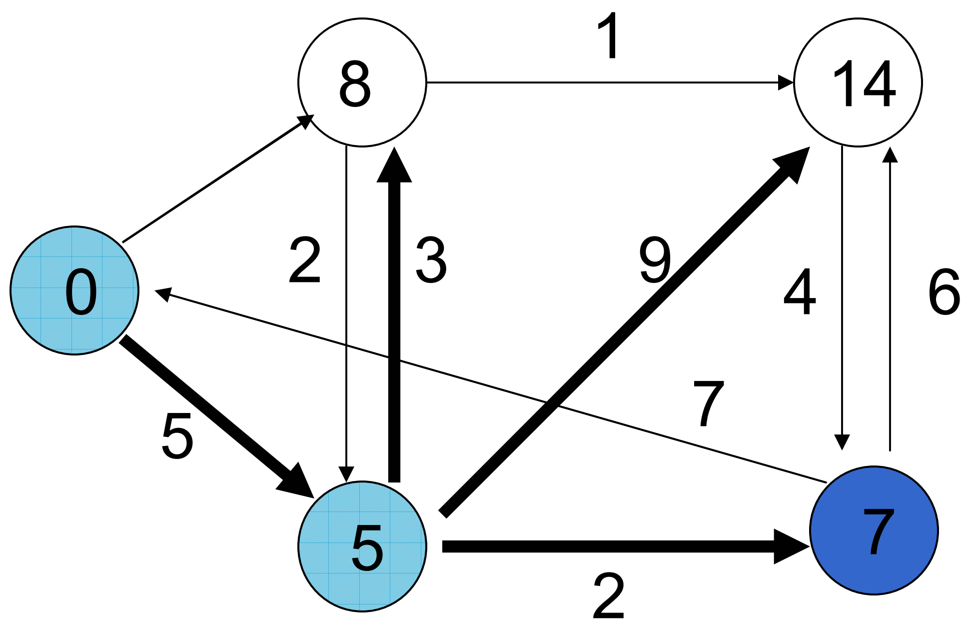
Dijkstra Example – Step 1



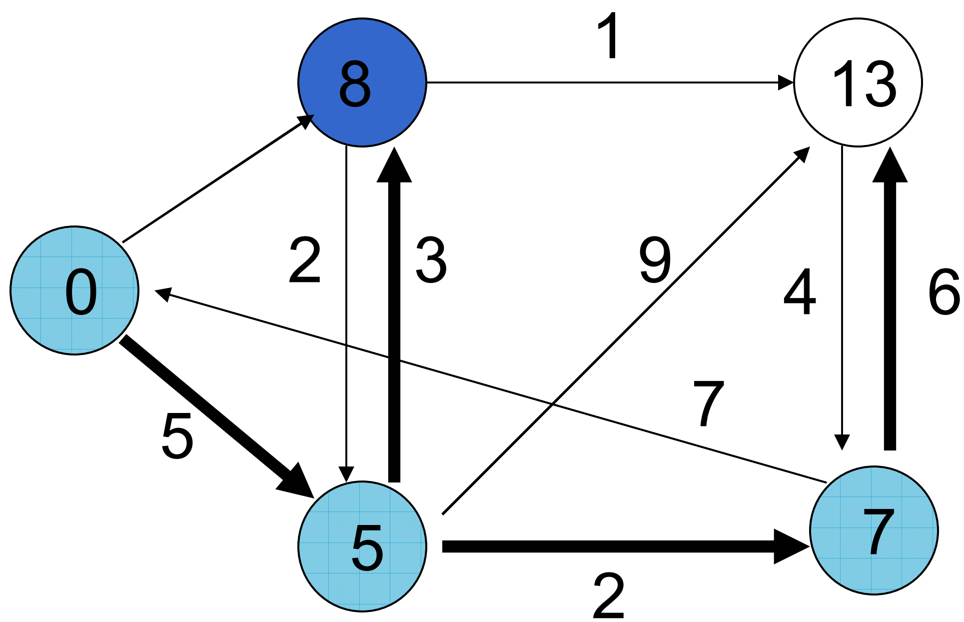
Dijkstra Example – Step 2



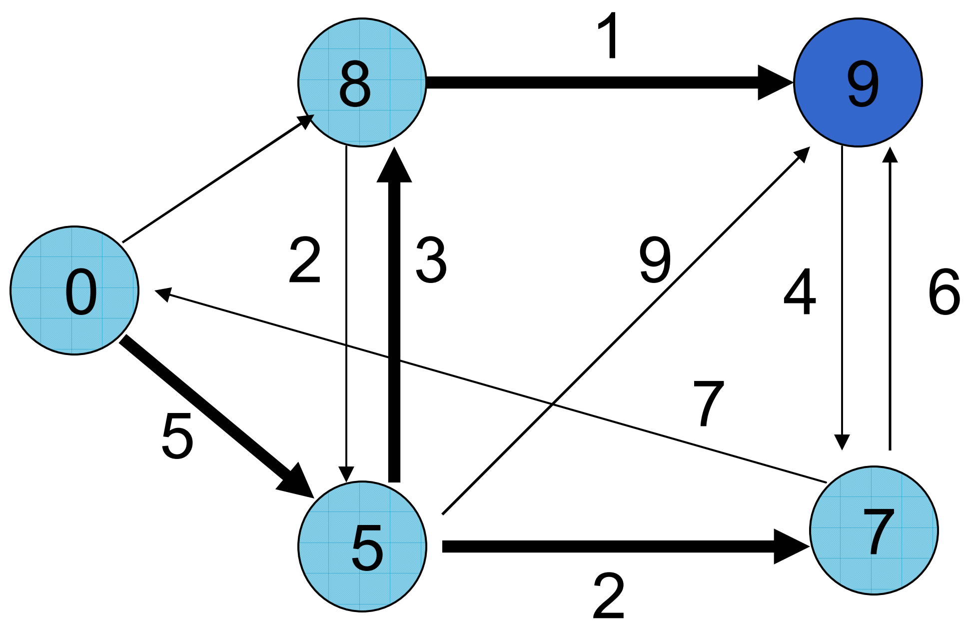
Dijkstra Example – Step 3



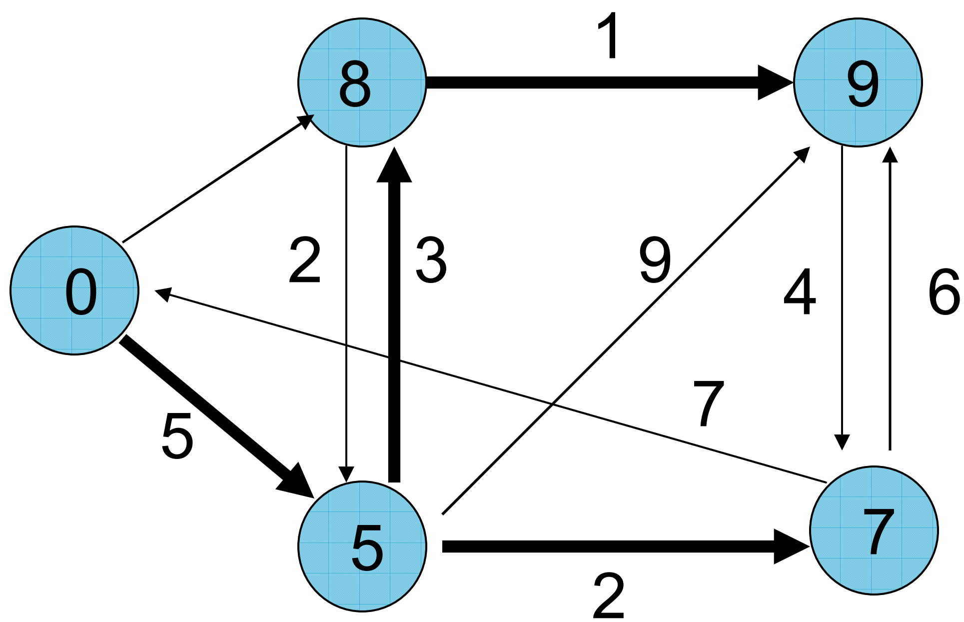
Dijkstra Example – Step 4



Dijkstra Example – Step 5



Dijkstra Example – Done



Open Shortest Path First (OSPF)

- Widely-used Link State protocol today; see also ISIS

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

Contrast of DV and LS

- DV: “Tell your neighbors about the world.”
  - Easy to get confused
  - Simple but limited, chatty and slow
    - Number of hops might be limited
    - Periodic broadcasts of large tables
    - Slow convergence due to ripples and hold down

- LS: “Tell the world about your neighbors.”
  - Harder to get confused
  - More complex and more computation
    - Faster convergence and better stability
    - Able to impose global policies in a globally consistent way
      - e.g., load balancing

- LS is used today except in resource limited situations

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
- Fixed or load adaptive route selection

- Internet is mostly to the left ☺

How do we choose the best path? (What is "best" anyway?)

- Can define "best" as lowest cost. But how to choose cost?
  - To get high bandwidth, low delay or low loss?
  - Do they depend on the load, e.g., to avoid congestion?

- Can define "best" for the network, not individual paths.
  - Routing is no longer solely on destination, e.g., MPLS
  - Typically an optimization problem that involves traffic

- Routing within a network today
  - Cost-based with fixed costs mostly based on latency
    - Direct paths are good for everyone in most networks
    - Manually adjust costs to "engineer" the network
  - Path-based (MPLS in ISPs) for greater engineering/control

What didn't work: 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

- Today we have slow-scale traffic engineering at ISPs