

Lecture 15:

Link state routing

CSE 123: Computer Networks

Aaron Schulman





Lecture 15 Overview

- Intra-domain routing
 - Introduction to the concept
 - Link-state routing



Introduction to Routing

- Forwarding
 - ◆ Move packet from input link to the appropriate output link
 - ◆ Purely local computation
 - ◆ Must go be very fast (executed for every packet)
- Routing
 - ◆ Make sure that the next hop leads **toward** the destination
 - ◆ Global decisions; distributed computation and communication
 - ◆ Can go slower (only important when topology changes)



Routing Options

- Source routing
 - ◆ Complete path listed in packet
- Virtual circuits
 - ◆ Set up path out-of-band and store path identifier in routers
 - ◆ Local path identifier in packet
- Destination-based routing
 - ◆ Router looks up address in forwarding table
 - ◆ Forwarding table contains (address, next-hop) tuples



Source Routing

- Routing
 - ◆ Host computes path
 - » Must know global topology and detect failures
 - ◆ Packet contains complete ordered path information
 - » I.e. node A then D then X then J...
 - ◆ Requires variable length path header
- Forwarding
 - ◆ Router looks up next hop in packet header, strips it off and forwards remaining packet
 - » Very quick forwarding, no lookup required
- In practice
 - ◆ ad hoc networks (DSR), some HPC networks (Myrinet), and for debugging on the Internet (LSR,SSR)



Virtual Circuits

- Routing
 - ◆ Hosts sets up path out-of-band, requires connection setup
 - ◆ Write (input id, output id, next hop) in table on each router
 - Flexible (one path per flow)
- Forwarding
 - ◆ Send packet with path id
 - ◆ Router looks up input, swaps for output, forwards on next hop
 - ◆ Repeat until reach destination
 - ◆ Table lookup for forwarding (why faster than IP lookup?)
- In practice
 - ◆ ATM: fixed VC identifiers and separate signaling code
 - ◆ MPLS: ATM meets the IP world (why? *traffic engineering*)



Destination-based Routing

- Routing
 - ◆ All addresses are globally known
 - » No connection setup
 - ◆ Host sends packet with destination address in header
 - » No path state; only routers need to worry about failure
 - ◆ Distributed routing protocol that populates routing tables
- Forwarding
 - ◆ Router looks up destination in forwarding table
 - » Must keep state proportional to destinations rather than connections
 - ◆ Lookup address, send packet to next-hop link
 - » All packets follow same path to destination
- In practice **this is what we use for Internet routing**



Intra-domain IP Routing

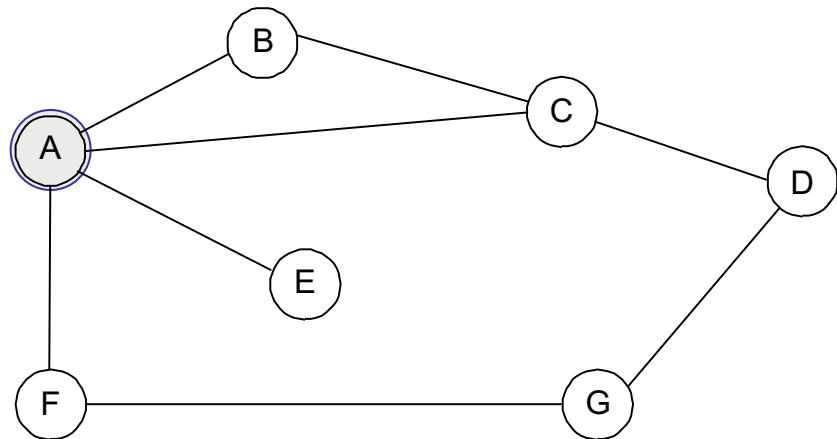
- Routing **within** a network/organization
 - ◆ A single administrative domain
 - ◆ The administrator can set edge costs
- Overall goals
 - ◆ Provide **intra**-network IP router connectivity
 - ◆ Adapt quickly to failures or topology changes
 - ◆ Optimize use of network resources
- Non-goals
 - ◆ Extreme scalability
 - ◆ Lying, and/or disagreements about edge costs
 - ◆ It's simpler, we will start by discussing it



Routing Tables

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

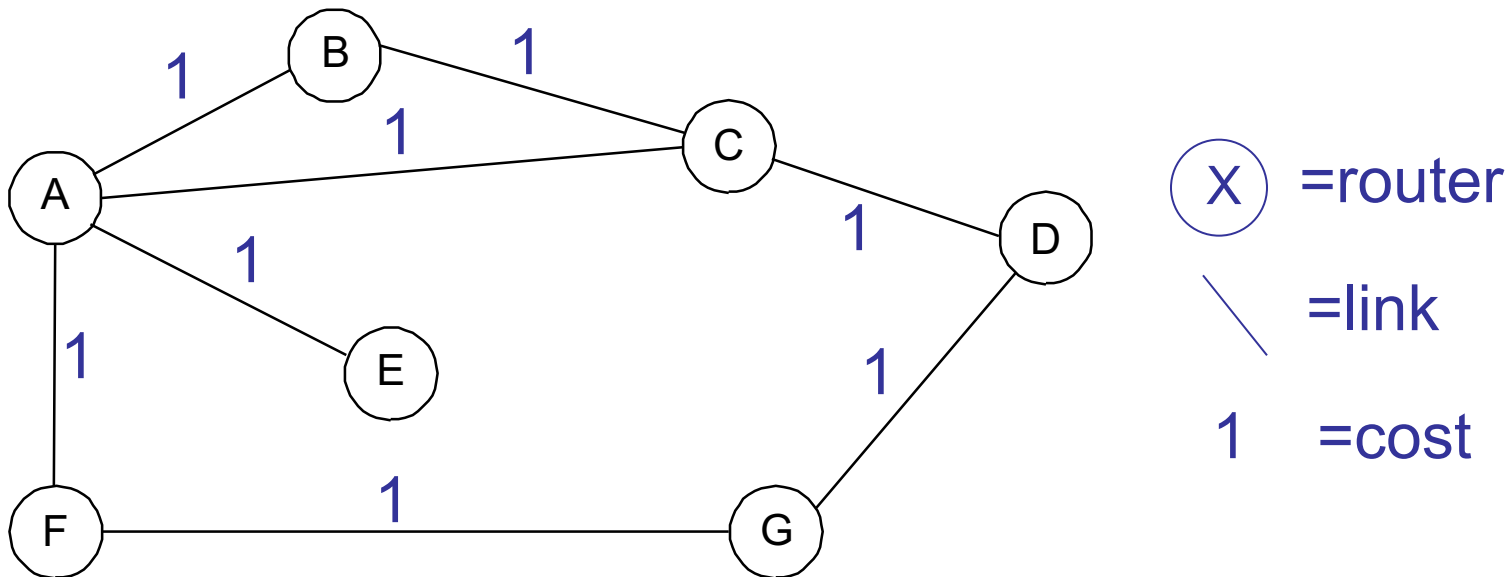
Dest	Next Hop
B	B
C	C
D	C
E	E
F	F
G	F





Routing on a Graph

- Essentially a graph theory problem
 - ◆ Network is a directed graph; routers are vertices
- Find “best” path between every pair of vertices
 - ◆ In the simplest case, best path is the shortest path





Routing Challenges

- How to choose best path?
 - ◆ Defining “best” can be slippery
- How to scale to millions of destinations (addresses)?
 - ◆ Minimize control messages sent between routers and routing table size in each router
- How to adapt to failures or changes?
 - ◆ Node and link failures, plus message loss



Intra-domain Routing

- Routing within a network/organization
 - ◆ A single administrative domain
 - ◆ The administrator can set edge costs
- Overall goals
 - ◆ Provide intra-network connectivity
 - ◆ Adapt quickly to failures or topology changes
 - ◆ Optimize use of network resources
- Non-goals
 - ◆ Extreme scalability
 - ◆ Lying, and/or disagreements about edge costs
 - ◆ Simpler, so we will start with it



Basic Approaches

- Static
 - ◆ Type in the right routes and hope they are always true
 - ◆ ...So far
- **Link state**
 - ◆ Tell **everyone** what you know about your **neighbors**
 - ◆ Today's lecture!
- Distance vector
 - ◆ Tell your **neighbors** what you know about **everyone**
 - ◆ Next time...



Link-state Routing

- Two steps
 - ◆ Reliable flooding
 - » Tell all routers what you know about your local topology
 - ◆ Path calculation (Dijkstra's algorithm)
 - » Each router computes best path over complete network
- Motivation
 - ◆ Global information allows optimal route computation
 - ◆ Straightforward to implement and verify



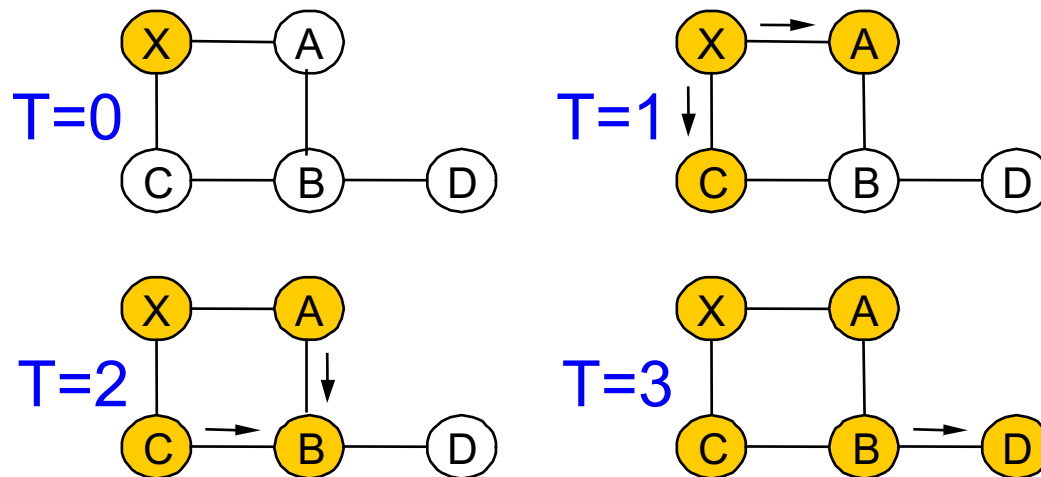
Broadcasting Link State

- Reliable flooding
 - ◆ Each router transmits a **Link State Packet** (LSP) on all links
 - ◆ A neighboring router forwards out all links except incoming
 - » Keep a copy locally; don't forward previously-seen LSPs
- Challenges
 - ◆ Packet loss
 - ◆ Out-of-order arrival
- Solutions
 - ◆ Acknowledgments and retransmissions
 - ◆ Sequence numbers
 - ◆ Time-to-live for each packet



Flooding Example

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



- Entire process repeats with LSPs for A, B, C, ...
 - ◆ Actually in runs in parallel

Dijkstra's Shortest Path Tree



- So you have all of these LSPs. Now what?
- Graph algorithm for single-source shortest path tree (find best route to all nodes)

$S \leftarrow \{\}$

$Q \leftarrow \text{<remaining nodes keyed by distance>}$

While $Q \neq \{\}$

$u \leftarrow \text{extract-min}(Q)$ *$u = \text{node with lowest cost}$*

$S \leftarrow S \text{ plus } \{u\}$ *$\leftarrow u \text{ is done}$*

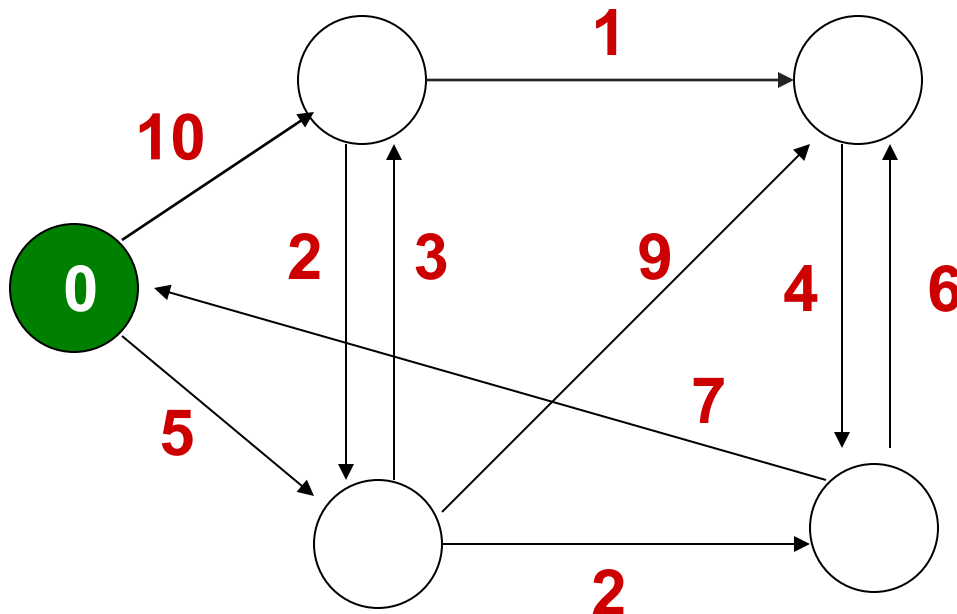
Within Q :

for each node v adjacent to u

“relax” the cost of v *is it cheaper to go through u ?*



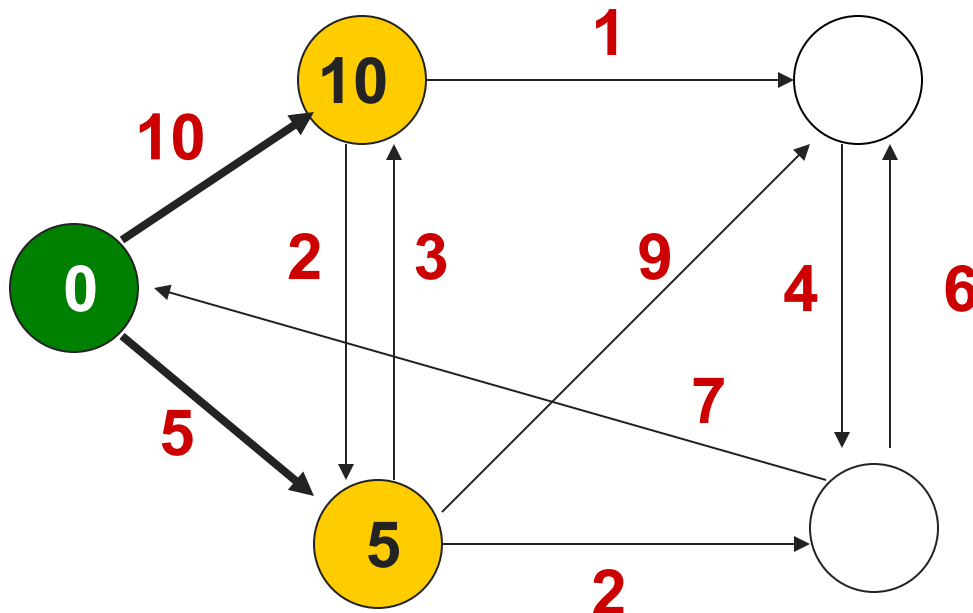
Dijkstra Example – Step 1



- **Green** nodes are “confirmed”
- **Yellow** nodes are “tentative”
- We can add ourselves to “confirmed”



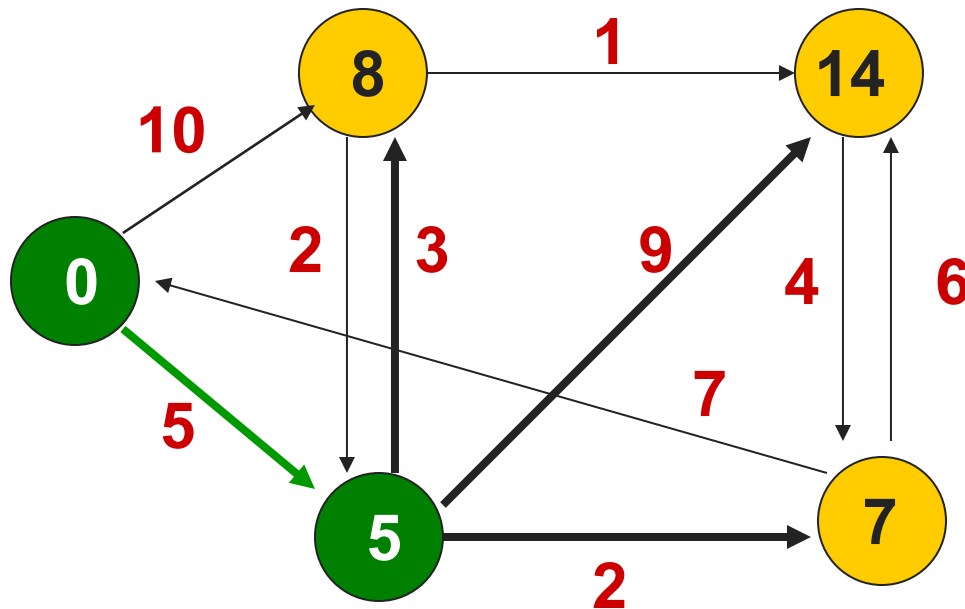
Example – Step 2



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- First look at neighbors
- “5” is cheaper than “10”
- We can confirm path with cost “5”



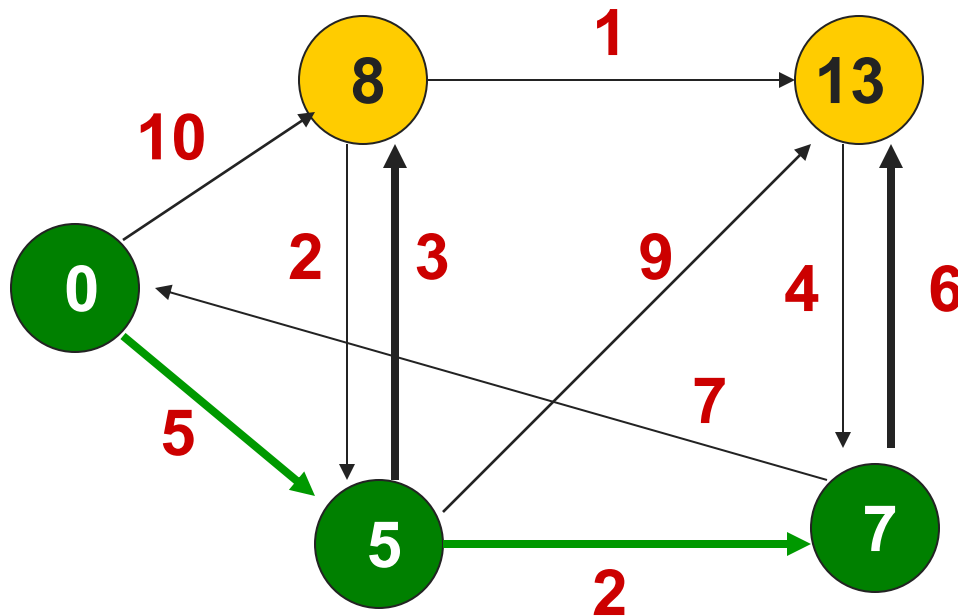
Example – Step 3



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- Update costs
- Look at 5’s neighbors
- 7 is cheapest
- We can confirm path with cost 7



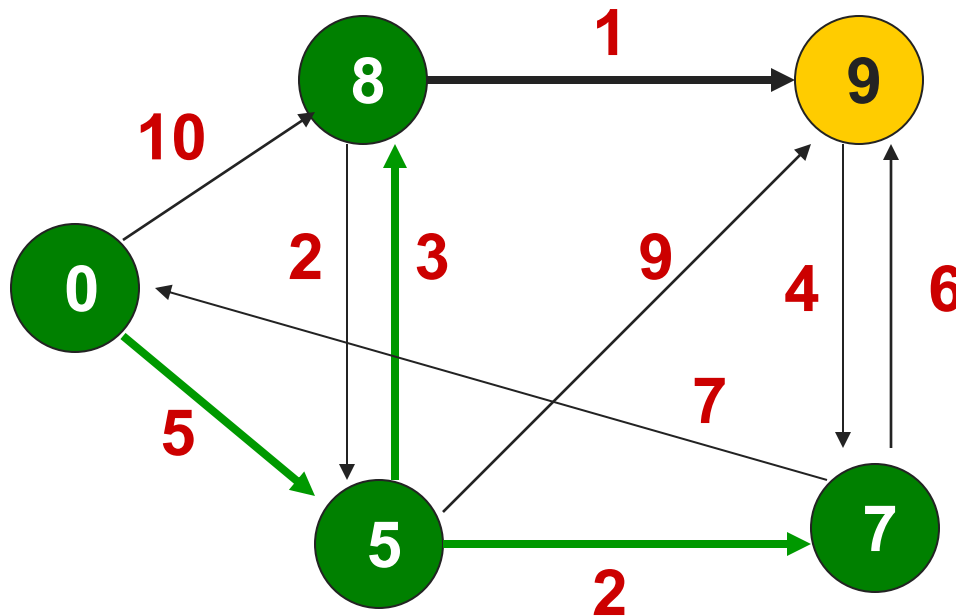
Example – Step 4



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- Update costs
- 7 has no new neighbors
- 8 is cheapest
- We can confirm 8

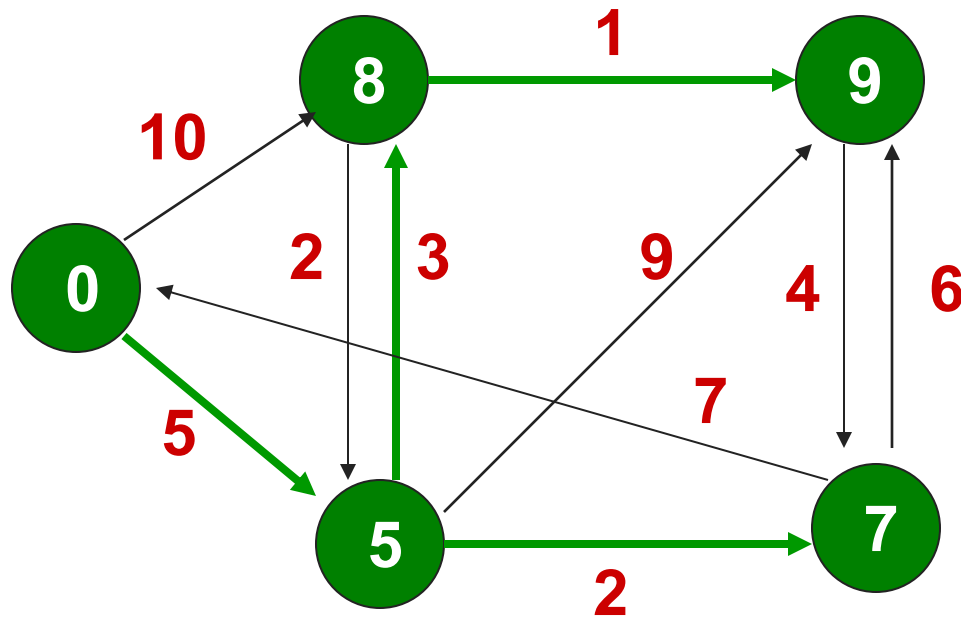


Example – Step 5



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- Update costs
- No new neighbors
- 9 is cheapest
- We can confirm path with cost 9

Example – Done



For next time...



- Read Chapter 3.4 in P&D (Distance vector)