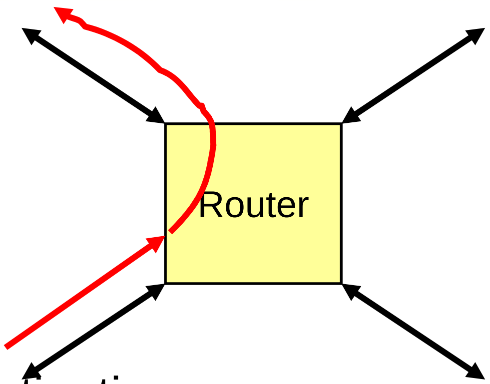
15-441 Computer Networks

Link State Routing

Professor Hui Zhang

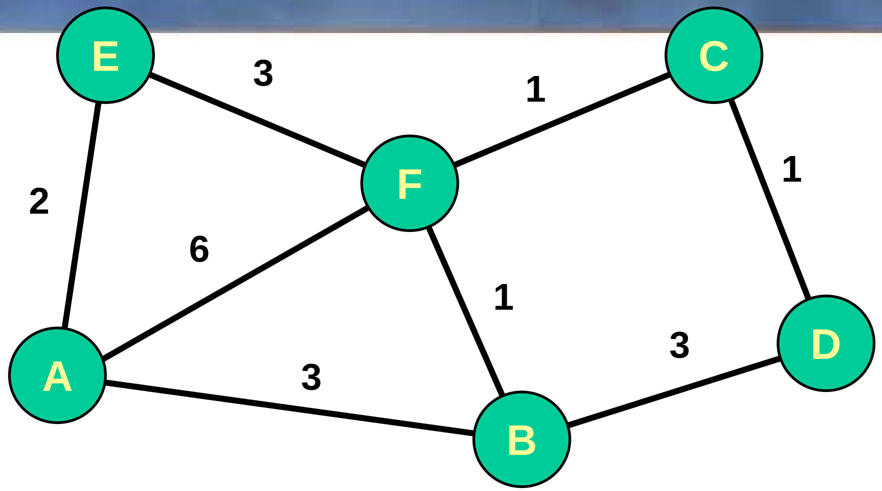
hzhang@cs.cmu.edu

Forwarding and Routing



- **Forwarding**
  - Examine header to determine intended destination
  - *Look up in table to determine next hop in path*
  - Send packet out appropriate port
- **Routing**
  - Each router *forwards* packet to next router
  - Overall goal is to *route* packet from source to destination
    - Requires consistent forwarding tables at different nodes
    - Distributed computation in dynamic environments

Graph Model



- Represent each router as node
- Direct link between routers represented by edge
  - asymmetric links ⇒ directed graph
- Edge “cost” c(x,y) denotes measure of difficulty of using link

❖ Task

- Determine least cost path from every node to every other node
  - Path cost d(x,y) = sum of link costs

Shortest Path Routing Table

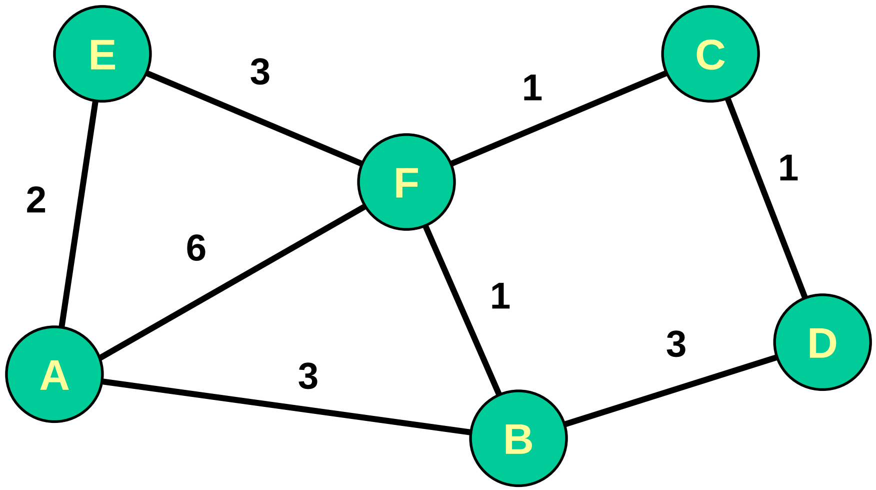


Table for A		
Dst	Cst	Hop
A	0	A
B	3	B
C	5	B
D	6	B
E	2	E
F	4	B

Table for B		
Dst	Cst	Hop
A	3	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

Table for C		
Dst	Cst	Hop
A	5	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	6	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	4	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Think Out of the Box

- **What are the limitations of the architecture with the following?**
  - Destination-based forwarding
  - Shortest-path routing

Ways to Compute Shortest Paths

- **Centralized**
  - Collect graph structure in one place
  - Use standard graph algorithm
  - Disseminate routing tables
- **Partially Distributed**
  - Every node collects complete graph structure
  - Each computes shortest paths from it
  - Each generates own routing table
  - "Link-state" algorithm
- **Fully Distributed**
  - No one has copy of graph
  - Nodes construct their own tables iteratively
  - Each sends information about its table to neighbors
  - "Distance-Vector" algorithm

Issues to Think About in Your Project

- How to tell a link or node fails?
- How to send control packets to neighbors?

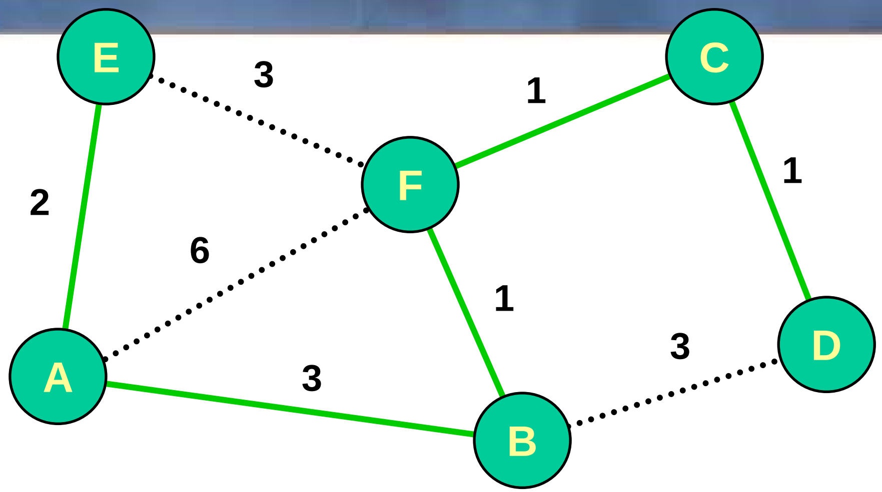
Link State Protocol Concept

- **Every Node Gets Complete Copy of Graph**
  - Every node “floods” network with data about its outgoing links
- **Every Node Computes Routes to Every Other Node**
  - Using single-source, shortest-path algorithm
- **Every Node Updates Own Routing Table**
- **Process Performed Whenever Needed**
  - When connections die / reappear
  - Periodically

Hui Zhang

Least Cost Routes from Node A

Table for A		
Dest	Cost	Next Hop
A	0	A
B	3	B
C	5	B
D	6	B
E	2	E
F	4	B



- **Properties**
  - Some set of shortest paths forms tree
    - Shortest path spanning tree
  - Solution not unique
    - E.g., A-B-D also has cost 6

Dijkstra's Algorithm

- **Edsgar Dijkstra (1930--2002)**
  - Pioneer in understanding mathematical basis for computer science
  - Fundamental ideas in concurrency (e.g., semaphores)
- **Given**
  - Graph with source node s and edge costs c(u,v)
  - Determine least cost path from s to every node v
- **Shortest Path First Algorithm**
  - Traverse graph in order of least cost from source

A Link State Routing Algorithm

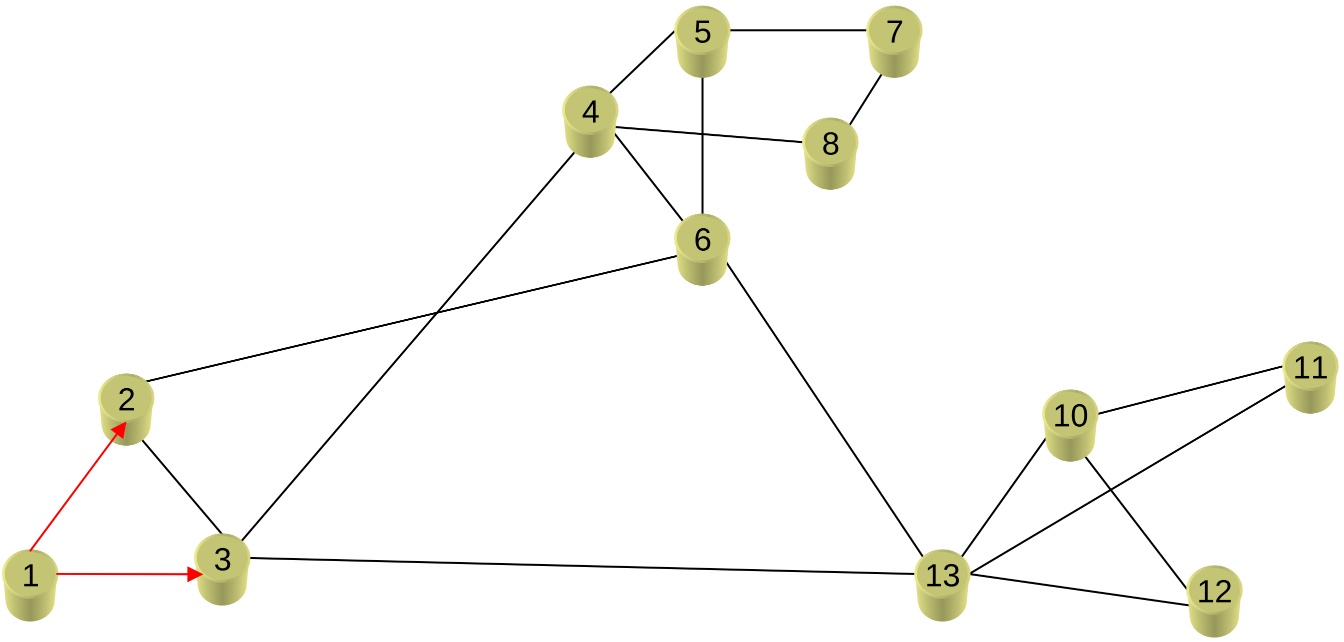
Dijkstra's algorithm

- **Net topology, link costs known to all nodes**
  - Accomplished via "link state flooding"
  - All nodes have same info
- **Compute least cost paths from one node ('source") to all other nodes**
- **Iterative: after *k* iterations, know least cost paths to *k* closest destinations**

Notations

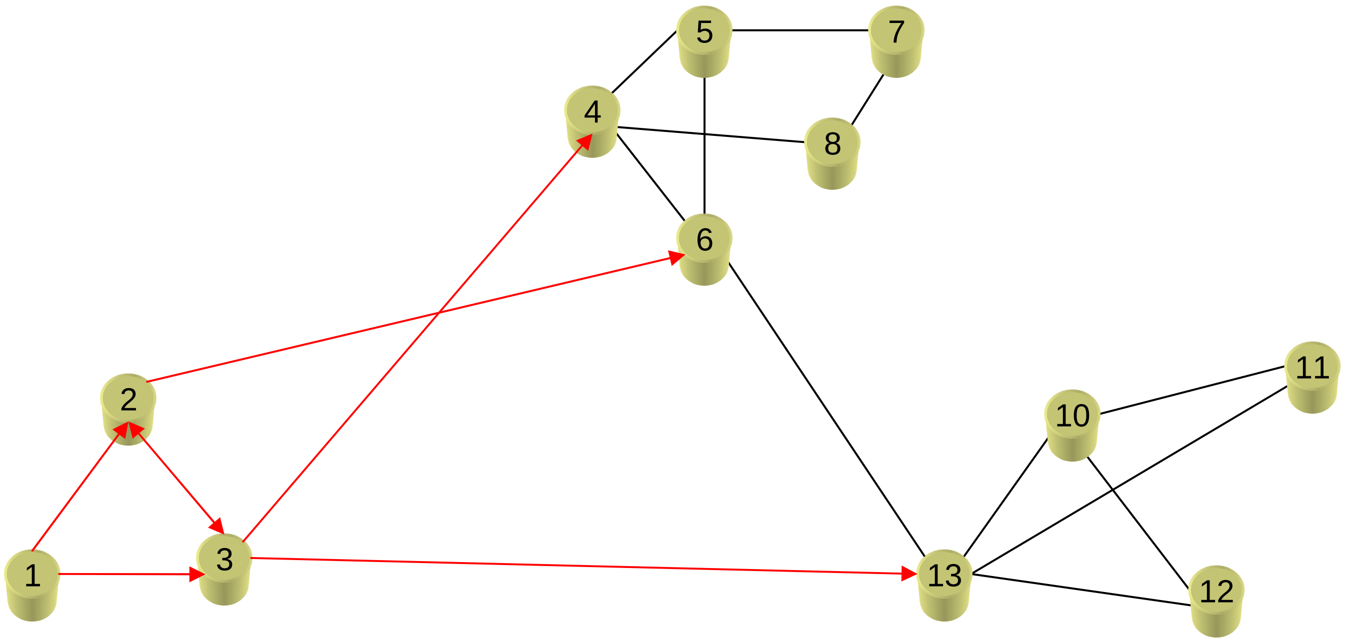
- **c(i,j): link cost from node *i* to *j*; cost infinite if not direct neighbors**
- **D(v): current value of cost of path from source to destination *v***
- **p(v): predecessor node along path from source to *v*, that is next to *v***
- **S: set of nodes whose least cost path definitively known**

Link State Flooding Example



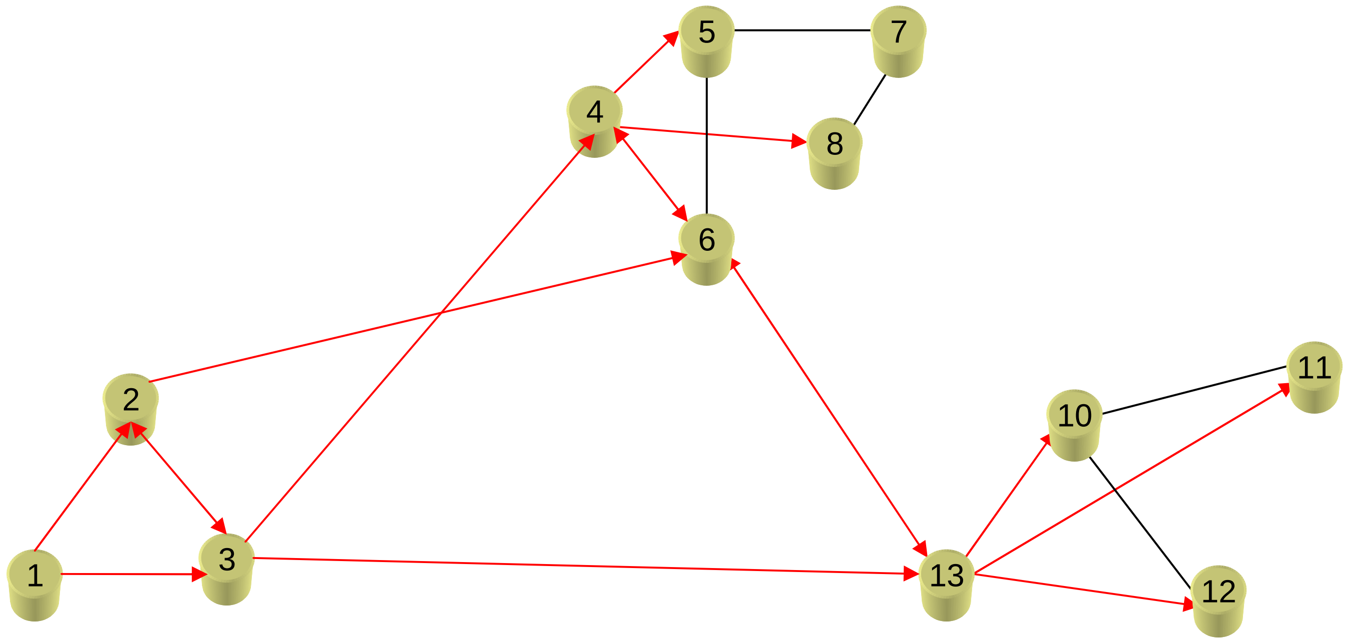
Hui Zhang

Link State Flooding Example



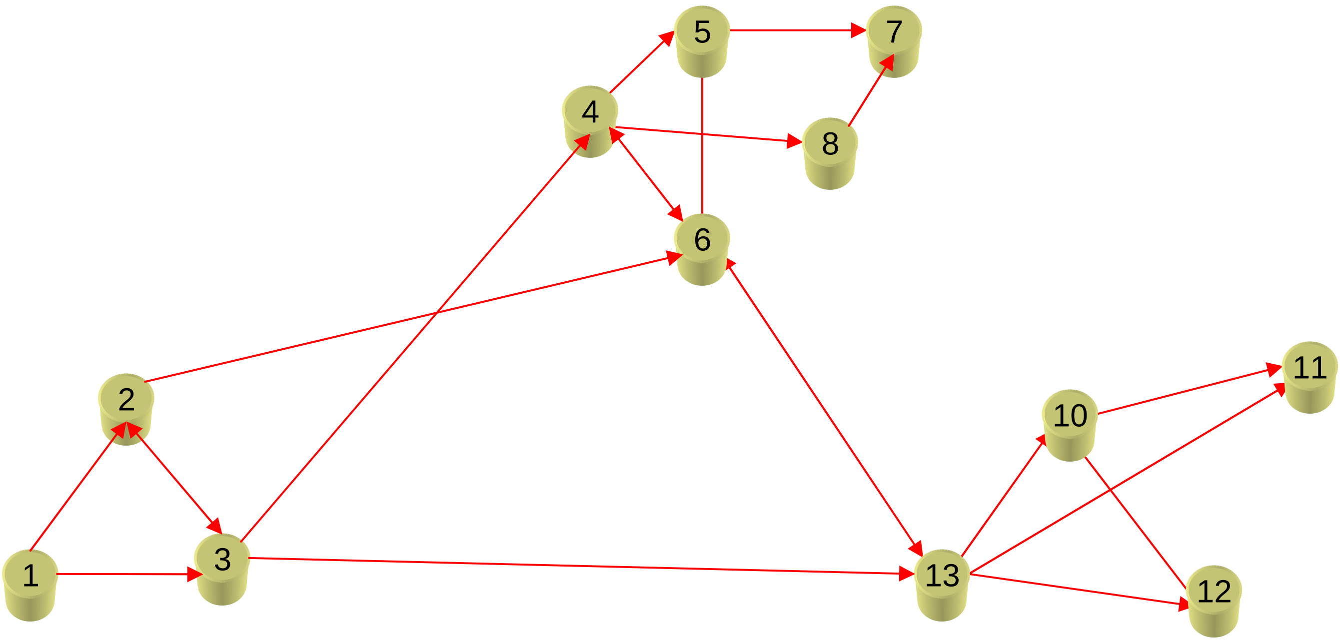
Hui Zhang

Link State Flooding Example



Hui Zhang

Link State Flooding Example



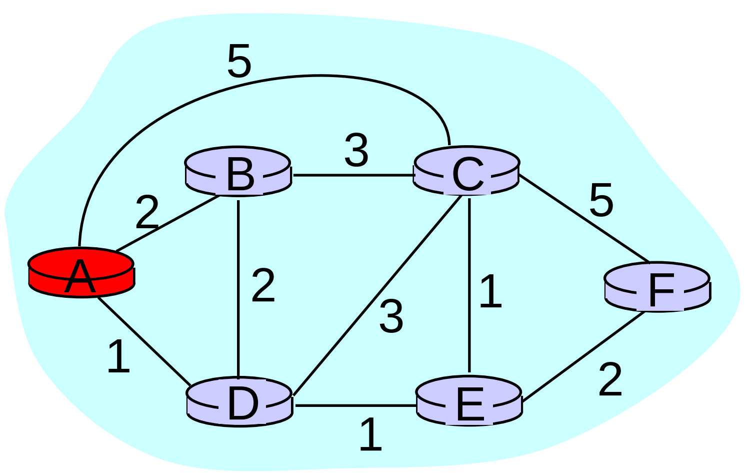
Dijsktra's Algorithm

```
Initialization:
  S = {A};
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v);
      else D(v) = ∞;

Loop
    find w not in S such that D(w) is a minimum;
    add w to S;
    update D(v) for all v adjacent to w and not in S:
       D(v) = min( D(v), D(w) + c(w,v) );
       // new cost to v is either old cost to v or known
       // shortest path cost to w plus cost from w to v
until all nodes in S;
```

Example: Dijkstra’s Algorithm

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						

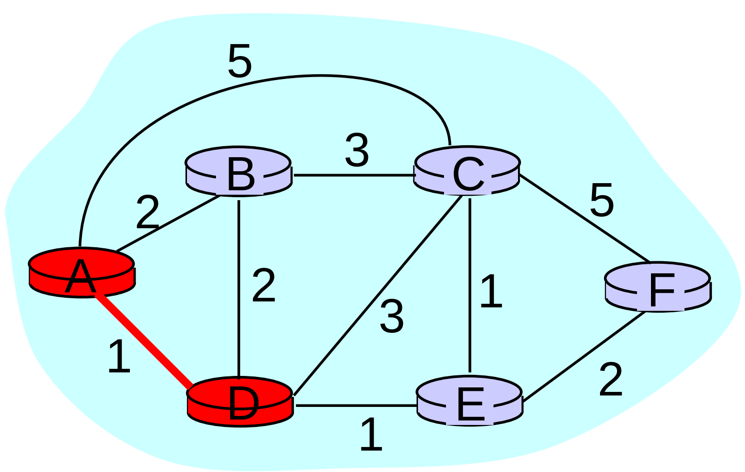


Initialization:
  S = {A};
  for all nodes *v*
    if *v* adjacent to *A*
      then D(v) = c(A,v);
      else D(v) = ∞;
…

Hui Zhang

Example: Dijkstra's Algorithm

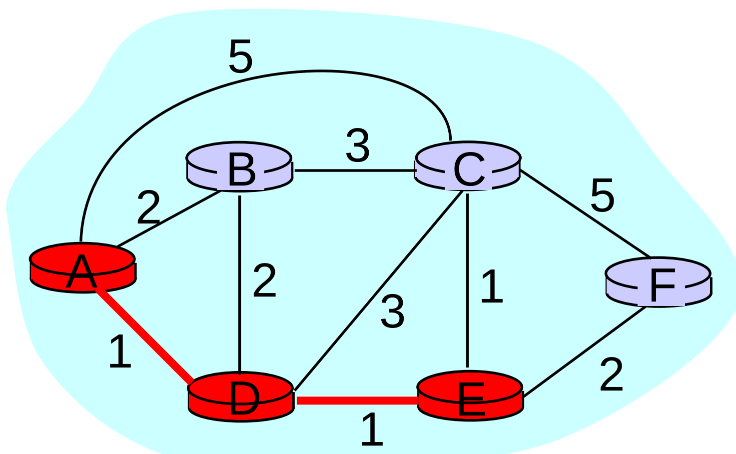
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						




```
…
Loop
    find w not in S s.t. D(w) is a minimum;
    add w to S;
   update D(v) for all v adjacent
    to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
    until all nodes in S;
```

Example: Dijkstra's Algorithm

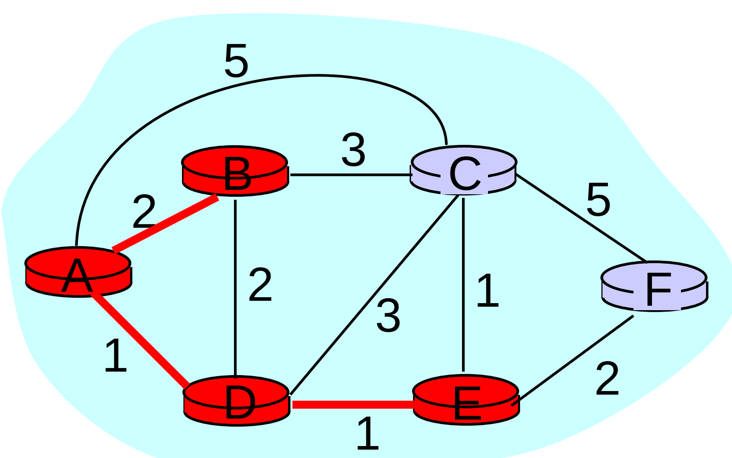
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						




```
...
Loop
    find w not in S s.t. D(w) is a minimum;
    add w to S;
   update D(v) for all v adjacent
    to w and not in S:
       D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

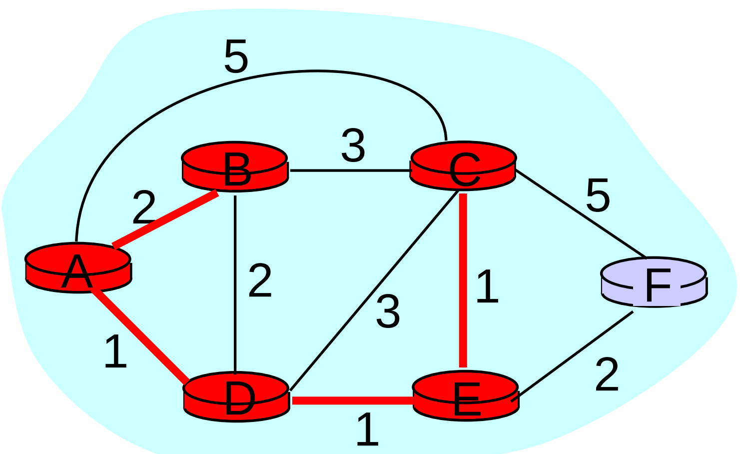
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						



…
Loop
find w not in S s.t. D(w) is a minimum;
add w to S;
update D(v) for all v adjacent
to w and not in S:
D(v) = min(D(v), D(w) + c(w,v));
until all nodes in S;

Example: Dijkstra's Algorithm

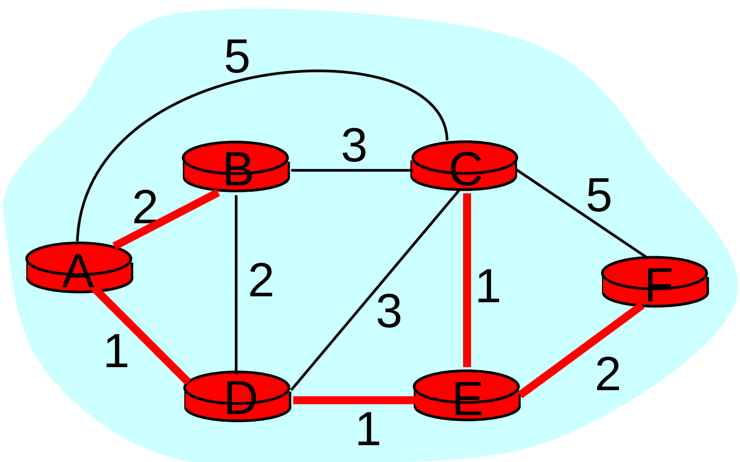
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
→ 4	ADEBC					
5						




```
…
Loop
    find w not in S s.t. D(w) is a minimum;
    add w to S;
   update D(v) for all v adjacent
    to w and not in S:
       D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Example: Dijkstra's Algorithm

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					



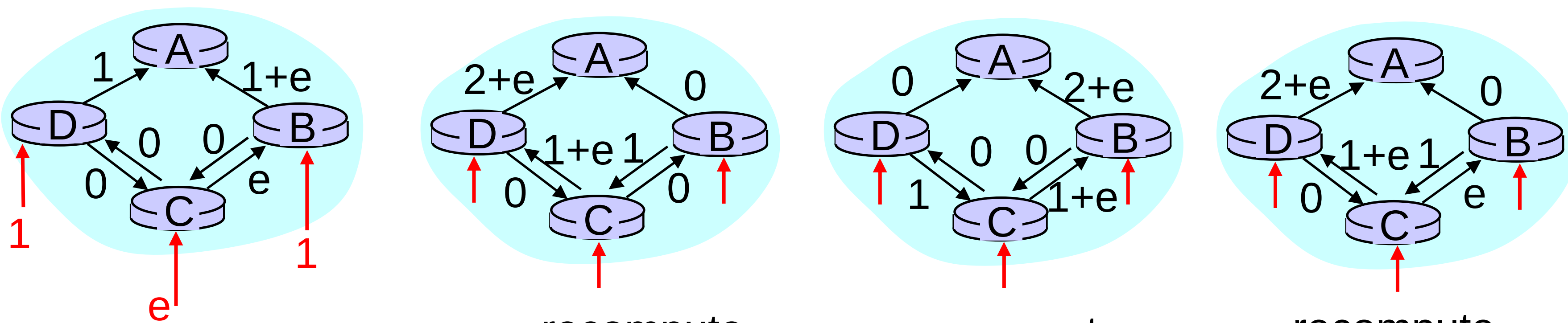

```
…
Loop
    find w not in S s.t. D(w) is a minimum;
    add w to S;
   update D(v) for all v adjacent
    to w and not in S:
       D(v) = min( D(v), D(w) + c(w,v) );
    until all nodes in S;
```

Complexity

- **Assume a network consisting of n nodes**
  - Each iteration: need to check all nodes, w, not in S
  - n*(n+1)/2 comparisons: $O(n^2)$
  - More efficient implementations possible: O(n*log(n))

Oscillations

- **Assume link cost = amount of carried traffic**



initially … recompute routing … recompute … recompute

- How can you avoid oscillations?

Hui Zhang

OSPF Routing Protocol

- **Open**
  - Open standard created by IETF
- **Shortest-Path First**
  - Another name for Dijkstra’s algorithm
- **Most Prevalent Intradomain Routing Protocol**

Hui Zhang

OSPF Reliable Flooding

- **Transmit Link State Advertisements**
  - Originating Router
    - – Typically, minimum IP address for router
  - Link ID
    - – ID of router at other end of link
  - Metric
    - – Cost of link
  - Link-State Age
    - – Incremented each second
    - – Packet expires when reaches 3600
  - Sequence Number
    - – Incremented each time sending new link information

OSPF Flooding Operation

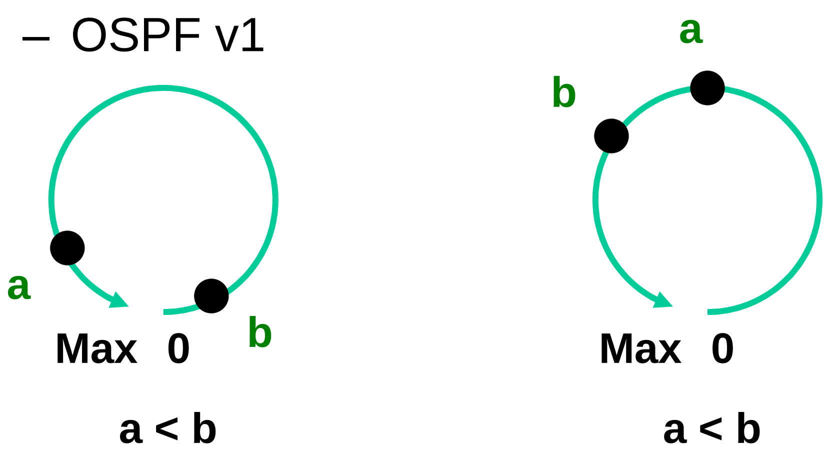
- **Node X Receives LSA from Node Y**
  - With Sequence Number q
  - Looks for entry with same origin/link ID
- **Cases**
  - No entry present
    - Add entry, propagate to all neighbors other than Y
  - Entry present with sequence number p < q
    - Update entry, propagate to all neighbors other than Y
  - Entry present with sequence number p > q
    - Send entry back to Y
    - To tell Y that it has out-of-date information
  - Entry present with sequence number p = q
    - Ignore it

Flooding Issues

- **When Should it be Performed**
  - Periodically
  - When status of link changes
    - Detected by connected node
- **What Happens when Router Goes Down & Back Up**
  - Sequence number reset to 0
    - Other routers may have entries with higher sequence numbers
  - Router will send out LSAs with number 0
  - Will get back LSAs with last valid sequence number p
  - Router sets sequence number to p+1 & resends

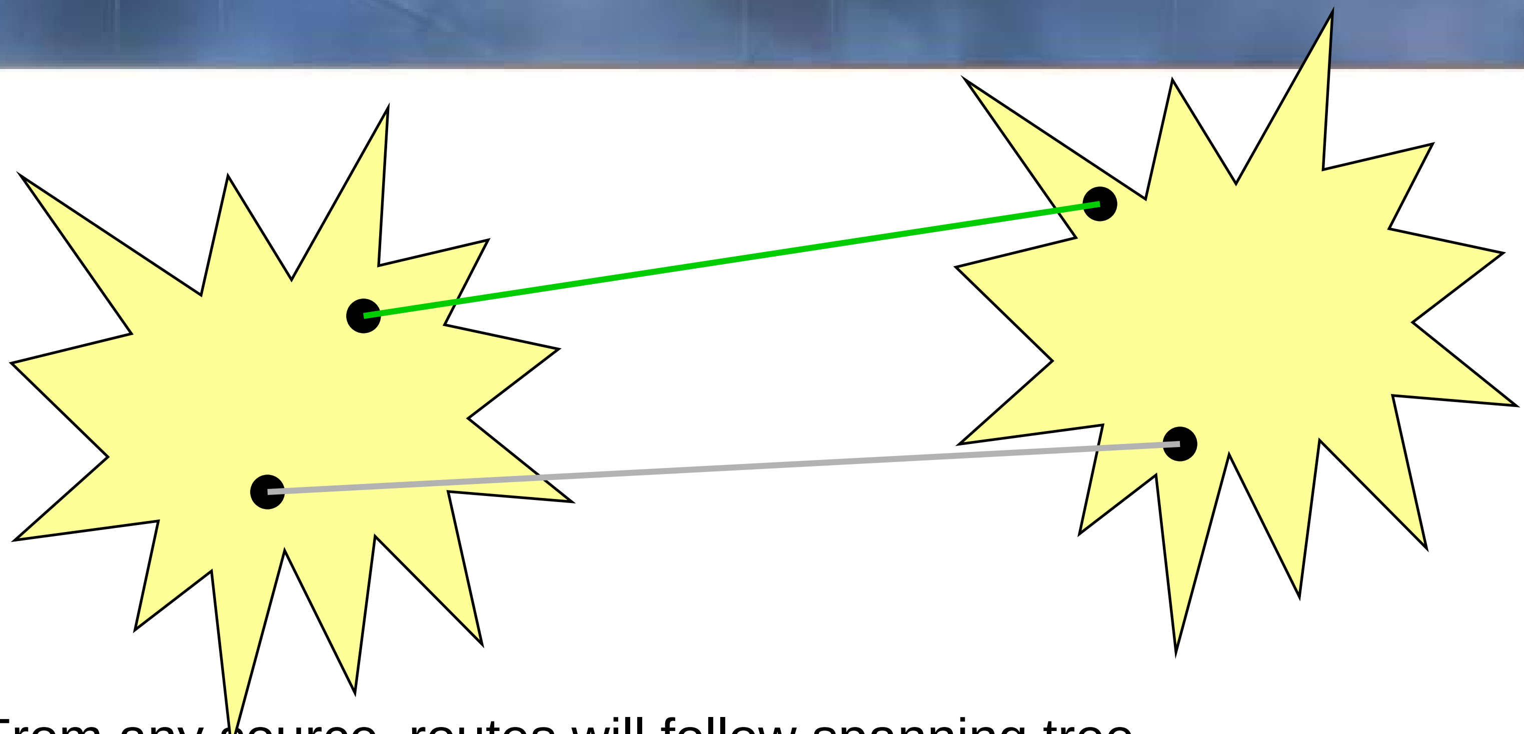
Flooding Issues (Cont.)

- **What if Sequence Number Wraps Around**
  - Use circular comparison
    - OSPF v1



  - Force sequence number back to 0
    - OSPF v2
    - With 32-bit counter, doesn’t happen very often

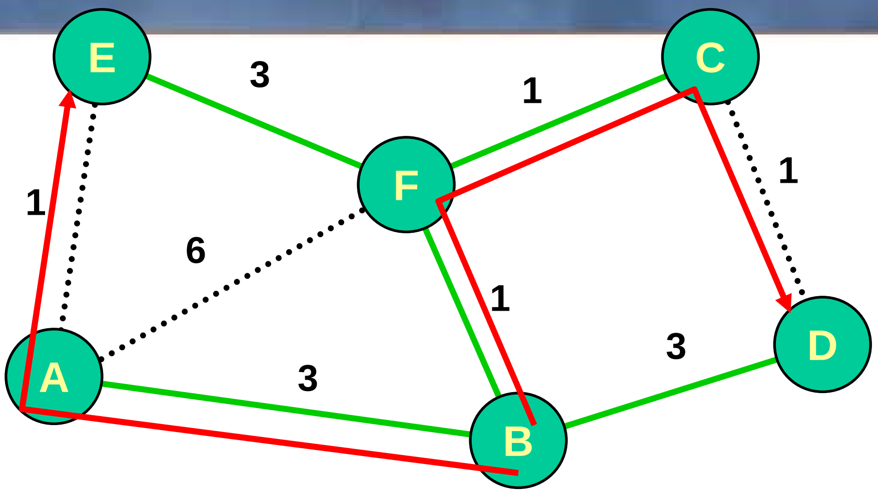
Load Balancing Motivation



- From any source, routes will follow spanning tree
- Single link may be chosen for many different sources
- Doesn’t spread traffic over all available links

OSPF Load Balancing

Table for B		
Dst	Cst	Hop
A	3	A
B	0	B
C	2	F
D	3	D,F
E	4	A,F
F	1	F



- **Modification to Dijkstra's algorithm**
  - Keep track of all links giving optimum cost d(v)
  - Only get multiple routes when exactly same cost
- **Routing**
  - Alternate link used
  - Tends to cause packets to arrive out of order

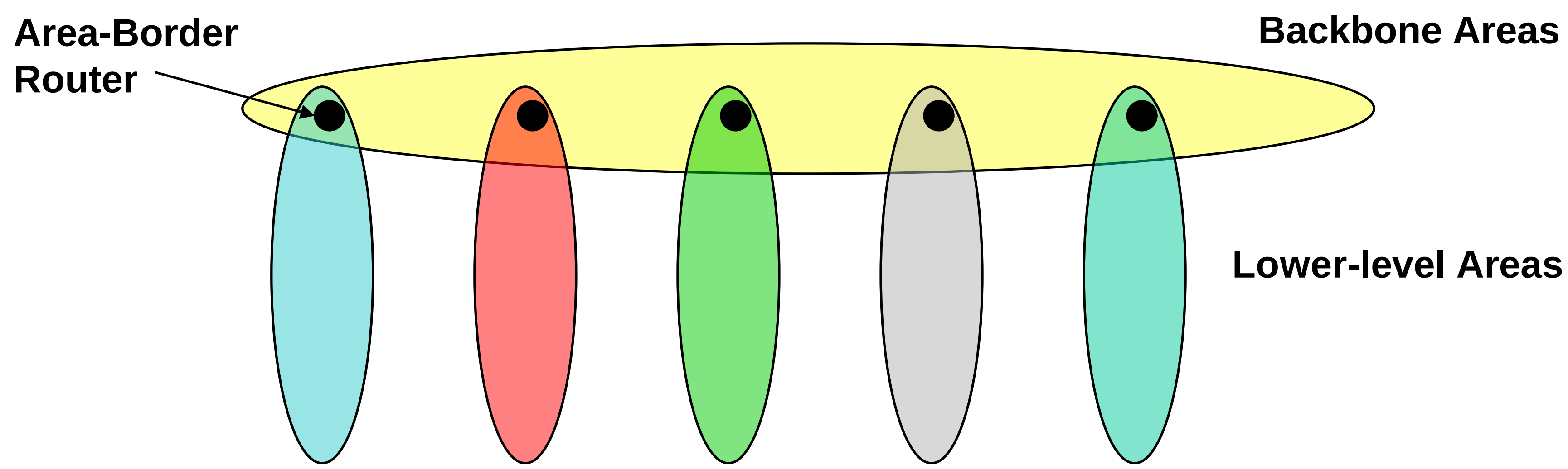
Type of Service (TOS) Metrics

- **Link Characteristic Vary in Multiple Dimensions**
  - Latency
  - Throughput
  - Cost
  - Reliability
- **Example**
  - Satellite link
    - High throughput, long latency
  - Fiber optic link
    - Lower throughput, low latency
- **Routing Requirements Vary**
  - Typing at terminal: minimize latency for short packet
  - Sending video data: maximize throughput

Proposed OSPF Support for TOS

- **Support up to Five Different Routing Metrics**
  - Normal service
    - Don't do anything extreme
  - Minimize cost
    - For networks that charge for traffic
  - Maximize reliability
  - Maximize throughput
  - Minimize delay
- **Link Can Have Different LSA for each TOS**
  - Expressed in units where lower value is better
  - Path cost either sum or maximum of link costs

OSPF Routing Hierarchy



- **Partition Network into "Areas"**
  - Router maintains link states of nodes within its area
  - Nodes in lower-level area use area-border router as default router
  - Backbone nodes can "summarize" routes within area