Chapter 6
Packet-Switching Networks

Services and Operation
Topology
Datagrams and Virtual Circuits
Routing

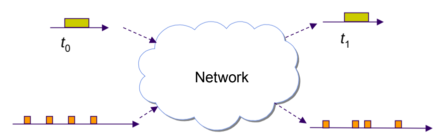
Services & Internal Operation

Network Layer

- Transfers *packets* across multiple links and/or multiple networks
- Requires the coordinated actions of multiple, geographically distributed network elements (switches & routers)
- Biggest Challenges
  - Addressing: where should information be directed to?
  - Routing: what path should be used to get information there?
  - Very large scales (billions of terminals).

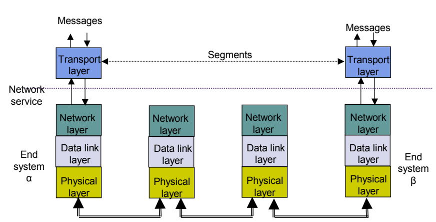
Packet vs. Circuit Switching

- Circuit-switching: end-to-end dedicated circuits between clients (save for later discussion)
- Packet-switching: transfer of information as payload in data packets (this and the next few lectures)



- Packets undergo random delays & possible loss
- Different applications impose different requirements

Network Service



- Network layer can offer a variety of services to transport layer
  - Connection-oriented service or connectionless service
  - Best-effort or delay/loss guarantees

Network Service vs. Operation

- Network Service
  - Connectionless: Datagram transfer
  - Connection-Oriented: Reliable and possibly constant bit rate transfer
- Internal Network Operation
  - Connectionless. E.g., IP
  - Connection-Oriented. E.g., telephone connection, ATM (later).
- Various combinations are possible
  - Connection-oriented service over Connectionless operation
  - Connectionless service over Connection-Oriented operation
  - Context & requirements determine what makes sense

Complexity at the Edge or in the Core?

- The End-to-End Argument for System Design
  An end-to-end function is best implemented at a higher level because a higher level is closer to the application and better positioned to ensure correct operation
- Example: stream transfer service
  - Establishing an explicit connection for each stream across network requires all network elements to be aware of connection;
  - In connectionless network operation, network elements do not deal with each explicit connection and hence are much simpler

Network Layer Functions

Essential
- **Routing**: mechanisms for determining the set of best paths. It requires the collaboration of network elements
- **Forwarding**: transfer of packets
- **Priority & Scheduling**: determining order of packet transmission

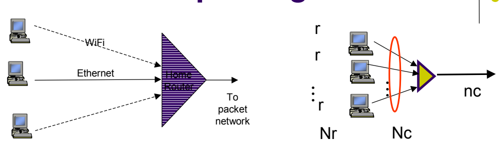
Optional:
- Congestion control
- Segmentation & reassembly
- Security

Packet Network Topology

Packet Network

- Internet structure highly decentralized
  - Paths traversed by packets can go through many networks controlled by different organizations
  - No single entity responsible for end-to-end service
- Individual packet streams can be highly bursty
  - Statistical multiplexing is used to concentrate streams
- User demand can undergo dramatic change
  - Peer-to-peer applications stimulated huge growth in traffic volumes

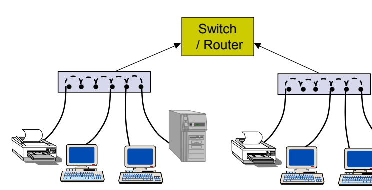
Access Multiplexing



- Packet traffic from users multiplexed at access to network into aggregated streams, e.g., DSL, Cable modem, Home LAN
- Access Multiplexer
  - N subscribers connected @ c bps to mux
  - Each subscriber active r/c of time
  - Mux has C=nc bps to network
  - Oversubscription rate: N/n
  - Find n so that at most 1% overflow probability

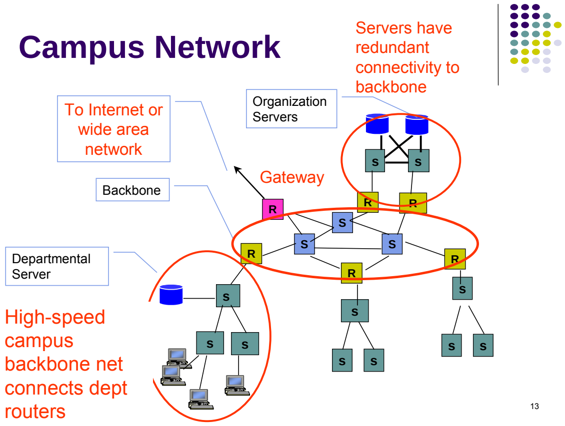
Feasible oversubscription rate N/n increases with size

LAN Concentration



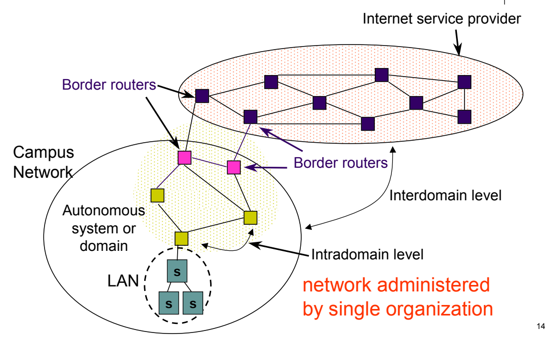
- LAN Hubs and switches in the access network also aggregate packet streams that flows into switches and routers

Campus Network



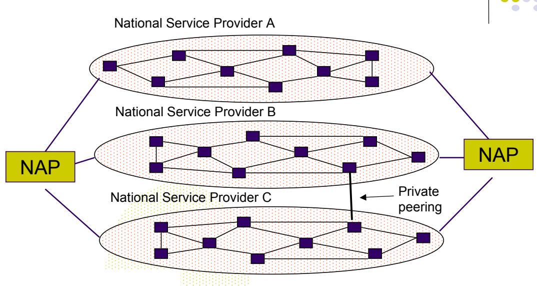
13

Connecting to Internet Service Provider (ISP)



14

Internet Backbone



- Network Access Points: set up during original commercialization of Internet to facilitate exchange of traffic
- Private Peering Points: two-party inter-ISP agreements to exchange traffic

15

Key Role of Routing

How to get packet from here to there?

- Decentralized routing
  - Interior gateway protocols (IGPs) determine routes within a domain
  - Exterior gateway protocols (EGPs) determine routes across domains
  - Routes must be consistent & produce stable flows
- Scalability
  - Hierarchical addressing essential to keeping size of routing tables manageable

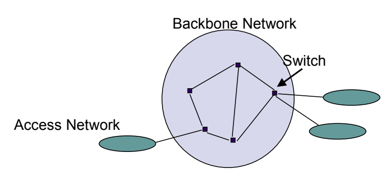
16

Datagrams & Virtual Circuits

17

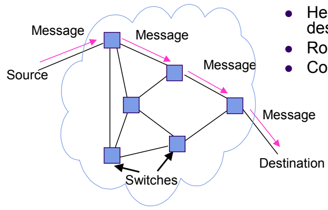
The Switching Function

- Dynamic interconnection of inputs to outputs
- Enables dynamic sharing of transmission resource
- Two fundamental approaches:
  - Connectionless
  - Connection-Oriented: Call setup control, Connection control



18

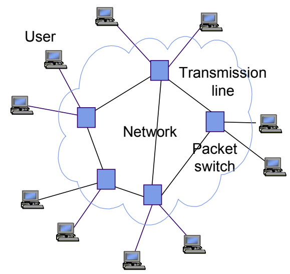
Example: Message Switching



- Message switching invented for telegraph
- Entire messages multiplexed onto shared lines, stored & forwarded
- Headers for source & destination addresses
- Routing at message switches
- Connectionless

19

Packet Switching Network

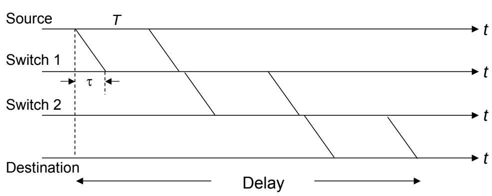


Packet switching network
- Transfers packets between users
- Transmission lines + packet switches (routers)
- Origin in message switching

Two modes of operation:
- Connectionless packet switching
- Virtual circuit switching

20

Switching Delay



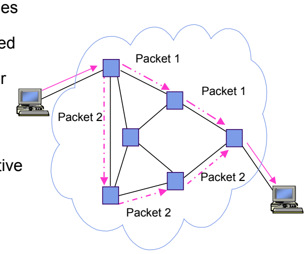
Minimum delay = L(τ + *T*), *L= # of hops*

Possible additional queueing delays at each link

21

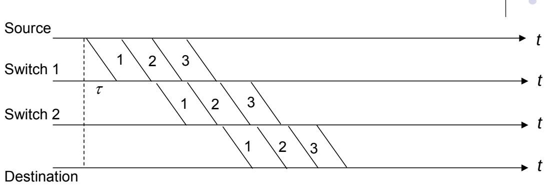
(Connectionless) Datagram Networks

- Messages broken into packets
- Source & destination addresses in packet header
- Connectionless, packets routed independently (datagram)
- Packet may arrive out of order
- Pipelining of packets across network can reduce delay, increase throughput
- Lower delay than message switching, suitable for interactive traffic



22

Packet Switching Delay:
k-Packet Message over L Hops



Let P=*T/k*

Delay = $L\tau + LP + (k\text{-}1)P$

$L\tau + (L\text{-}1)P$ first bit received
$L\tau + LP$ first bit released
$L\tau + LP + (k\text{-}1)P$ last bit released

23

Routing Tables in Datagram Networks

Destination address	Output port
0785	7
1345	12
1566	6
2458	12

- Route determined by table lookup
- Routing decision involves finding next hop in route to given destination
- Routing table has an entry for each destination specifying output port that leads to next hop
- Size of table becomes impractical for very large number of destinations

24

Example: Internet Routing

- Internet protocol uses datagram packet switching *across networks*
  - Networks are treated as data links
- Hosts have two-port IP address:

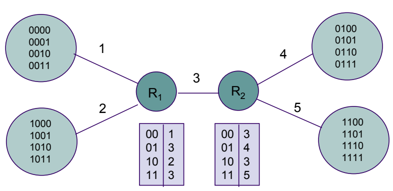
0	netid	hostid

- Routers do table lookup on network address
  - This reduces size of routing table
- In addition, network addresses are assigned so that they can also be aggregated

Flat vs Hierarchical Routing

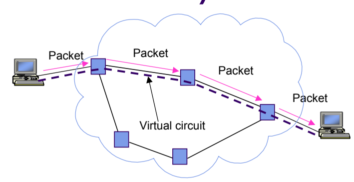
- Flat Routing
  - All routers are peers
  - Does not scale
- Hierarchical Routing
  - Partitioning: Domains, autonomous systems, areas...
  - Some routers part of routing backbone
  - Some routers only communicate within an area
  - Efficient because it matches typical traffic flow patterns
  - Scales

Hierarchical Addressing and Routing



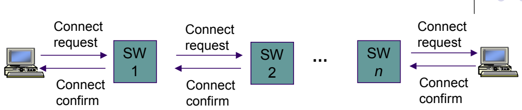
- Prefix indicates network where host is attached
- Routing tables require 4 entries each, much less than non-hierarchical routing

(Connection-Oriented) Virtual Circuits



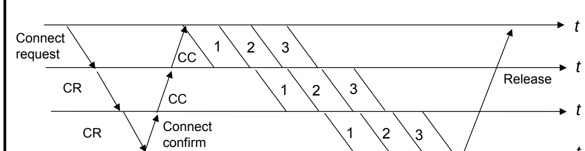
- Call set-up phase sets ups pointers in fixed path along network
- All packets for a connection follow the same path
- Abbreviated header identifies connection on each link
- Packets queue for transmission
- Variable bit rates possible, negotiated during call set-up
- Delays variable, cannot be less than circuit switching
- Example: ATM (Asynchronous Transfer Mode)

Connection Setup



- Signaling messages propagate as route is selected
- Signaling messages identify connection and setup tables in switches
- Typically a connection is identified by a local tag, Virtual Circuit Identifier (VCI)
- Each switch only needs to know how to relate an incoming tag in one input to an outgoing tag in the corresponding output
- Once tables are setup, packets can flow along path

Connection Setup Delay



- Connection setup delay incurred before packet transmission
- Delay acceptable for sustained transfer of large # of packets
- This delay overhead may be unacceptably high if only a few packets are being transferred

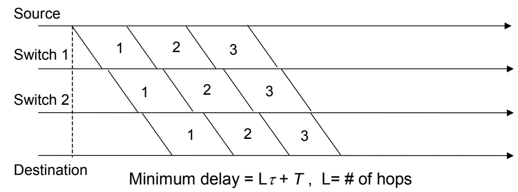
Virtual Circuit Forwarding Tables

Input VCI	Output port	Output VCI
12	13	44
15	15	23
27	13	16
58	7	34

- Each input port of packet switch has a forwarding table
- Lookup entry for VCI of incoming packet
- Determine output port (next hop) and insert VCI for next link
- Very high speeds are possible
- Table can also include priority or other information about how packet should be treated

31

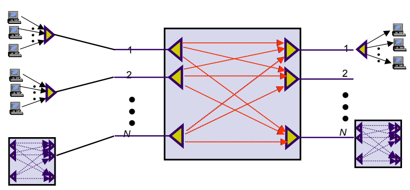
Cut-Through switching



- Some networks perform error checking on header only, so packet can be forwarded as soon as header is received & processed
- Delays reduced further with cut-through switching

32

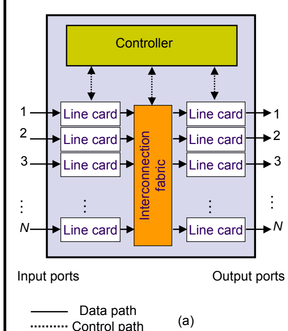
Packet Switch: Where Traffic Meet



- Inputs contain multiplexed flows from access MUXs & other packet switches
- Flows demultiplexed at input, routed and/or forwarded to output ports
- Packets buffered, prioritized, and multiplexed on output lines

33

Generic Packet Switch

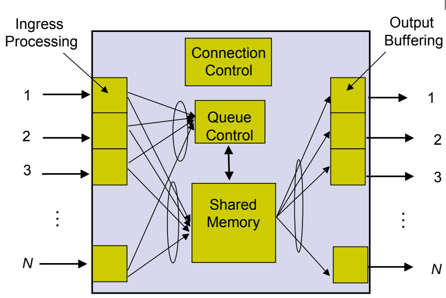


"Unfolded" View of Switch

- Ingress Line Cards
  - Header processing
  - Demultiplexing
  - Routing in large switches
- Controller
  - Routing in small switches
  - Signalling & resource allocation
- Interconnection Fabric
  - Transfer packets between line cards
- Egress Line Cards
  - Scheduling & priority
  - Multiplexing

34

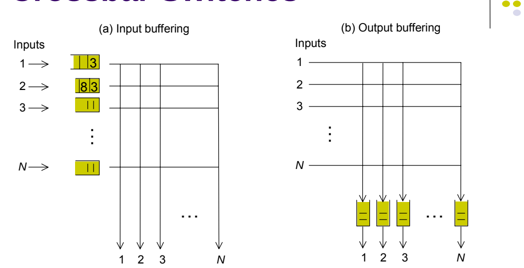
Shared Memory Packet Switch



Small switches can be built by reading/writing into shared memory

35

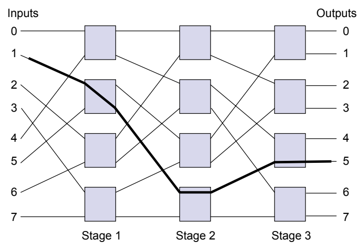
Crossbar Switches



- Large switches built from crossbar & multistage space switches
- Requires centralized controller/scheduler (who sends to whom when)
- Can buffer at input, output, or both (performance vs complexity)

36

Self-Routing Switches



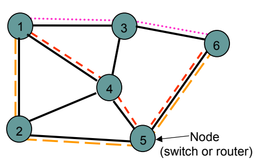
- Self-routing switches do not require controller
- Output port number determines route
- 101 → (1) lower port, (2) upper port, (3) lower port

37

Routing

38

Routing in Packet Networks



- Three possible (loopfree) routes from 1 to 6:
  - 1-3-6, 1-4-5-6, 1-2-5-6
- Which is "best"?
  - Min delay? Min hop? Max bandwidth? Min cost? Max reliability?

39

Routing Tables & Routing Algorithm

- Need information on state of links
  - Link up/down; congested; delay or other metrics
- Need to distribute link state information using a routing protocol
- Need to compute routes
- Responsiveness to changes
  - Topology or bandwidth changes, congestion
  - Rapid convergence of routers to consistent set of routes
  - Freedom from persistent loops
- Optimality, Robustness, Simplicity

40

Centralized vs Distributed Routing

- Centralized Routing
  - All routes determined by a central node
  - All state information sent to central node
  - Problems adapting to frequent topology changes
  - Does not scale
- Distributed Routing
  - Routes determined by routers using distributed algorithm
  - State information exchanged by routers
  - Adapts to topology and other changes
  - Better scalability

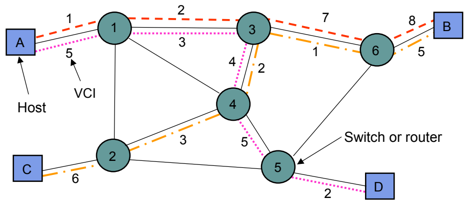
41

Static vs Dynamic Routing

- Static Routing
  - Set up manually, do not change; requires administration
  - Works when traffic predictable & network is simple
  - Used to override some routes set by dynamic algorithm
  - Used to provide default router
- Dynamic Routing
  - Adapt to changes in network conditions
  - Automated
  - Calculates routes based on received updated network state information

42

Routing in Virtual-Circuit Networks



- Route determined during connection setup
- Tables in switches implement forwarding that realizes selected route

Routing Tables in Virtual Circuit Networks

Node 1

Incoming Node	Incoming VCI	Outgoing Node	Outgoing VCI
A	1	3	2
A	5	3	3
3	2	A	1
3	3	A	5

Node 3

Incoming Node	Incoming VCI	Outgoing Node	Outgoing VCI
1	2	6	7
1	3	4	4
4	2	6	1
6	7	1	2
6	1	4	2
4	4	1	3

Node 6

Incoming Node	Incoming VCI	Outgoing Node	Outgoing VCI
3	7	B	8
3	1	B	5
B	5	3	1
B	8	3	7

Node 4

Incoming Node	Incoming VCI	Outgoing Node	Outgoing VCI
2	3	3	2
3	4	5	5
3	2	2	3
5	5	3	4

Node 2

Incoming Node	Incoming VCI	Outgoing Node	Outgoing VCI
C	6	4	3
4	3	C	6

Node 5

Incoming Node	Incoming VCI	Outgoing Node	Outgoing VCI
4	5	D	2
D	2	4	5

- Example: VCI from A to D
  - From A & VCI 5 → 3 & VCI 3 → 4 & VCI 4
  - → 5 & VCI 5 → D & VCI 2

Routing Tables in Datagram Networks

Node 1

Destination	Next node
2	2
3	3
4	4
5	2
6	3

Node 3

Destination	Next node
1	1
2	4
4	4
5	6
6	6

Node 6

Destination	Next node
1	3
2	5
3	3
4	3
5	5

Node 4

Destination	Next node
1	1
2	2
3	3
5	5
6	3

Node 2

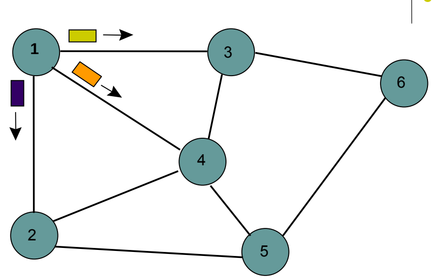
Destination	Next node
1	1
3	1
4	4
5	5
6	5

Node 5

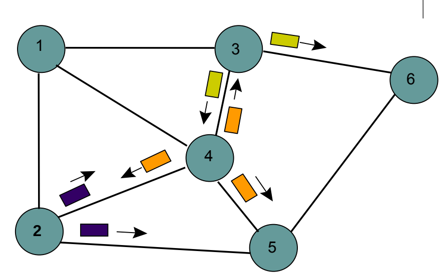
Destination	Next node
1	4
2	2
3	4
4	4
6	6

Specialized Routing

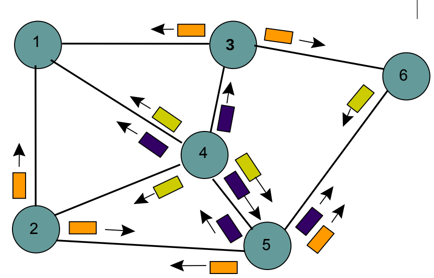
- Flooding: broadcast to all nodes
  - No routing tables
  - Useful in propagating info to all nodes, e.g., link state
  - Send packet on all ports except one where it arrived
  - Exponential growth in packet transmissions
  - Limited flooding using Time-to-Live, ID or sequence number
- Deflection Routing
  - Fixed, preset routing procedure, no route synthesis
  - Network nodes forward packets to preferred port, if busy, deflect packet to another port
  - Works well with regular topologies
    - Manhattan street network (one-way streets)
  - Bufferless operation is possible
    - Good for optical networks because all-optical buffering not viable



Flooding is initiated from Node 1: Hop 1 transmissions



Flooding is initiated from Node 1: Hop 2 transmissions



Flooding is initiated from Node 1: Hop 3 transmissions

49

Shortest Paths & Routing

- Typically it is possible to attach a cost or distance to a link connecting two nodes
- Routing can then be posed as a shortest path problem
- Path Length = sum of costs or distances
- Possible metrics
  - Hop count: rough measure of resources used
  - Reliability: link availability; BER
  - Delay: sum of delays along path; complex & dynamic
  - Bandwidth: “available capacity” in a path
  - Load: Link & router utilization along path
  - Cost: $$$

50

Shortest Path Approaches

Distance Vector Protocols

- Neighbors exchange list of distances to destinations
- Best next-hop determined for each destination
- Ford-Fulkerson (distributed) shortest path algorithm

Link State Protocols

- Link state information flooded to all routers
- Routers have complete topology information
- Shortest path (& hence next hop) calculated
- Dijkstra (centralized) shortest path algorithm

51

Distance Vector

Local Signpost

- Direction
- Distance

Routing Table

For each destination list:

- Next Node
- Distance

dest	next	dist

Table Synthesis

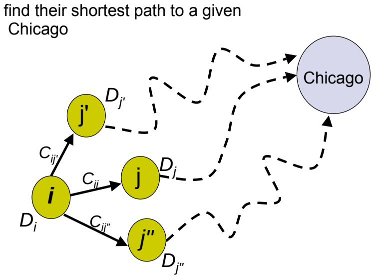
- Neighbors exchange table entries
- Determine current best next hop
- Inform neighbors
  - Periodically
  - After changes

52

Shortest Path

Focus on how nodes find their shortest path to a given destination node, i.e. Chicago

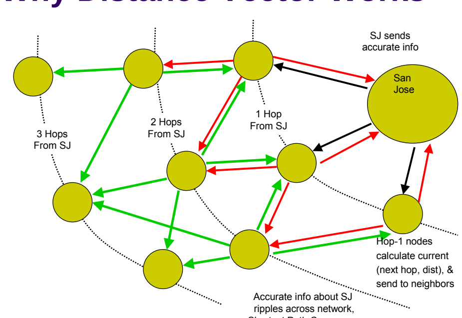
i only has local info from neighbors



If D_i is the shortest distance to Chicago from *i* and if *j* is a neighbor on the shortest path, then $D_i = C_{ij} + D_j$

53

Why Distance Vector Works



54

Bellman-Ford Algorithm

- *Consider computations for one destination d*
- *Initialization*
  - Each node table has 1 row for destination *d*
  - Distance of node *d* to itself is zero: $D_d=0$
  - Distance of other node *j* to *d* is infinite: $D_j=\infty$, for $j\neq d$
  - Next hop node n_j = -1 to indicate not yet defined for $j \neq d$
- *Send Step*
  - Send new distance vector to immediate neighbors across local link
- *Receive Step*
  - At node *j*, find the next hop that gives the minimum distance to *d*,
    - $Min_j \{ C_{ij} + D_j \}$
    - Replace old $(n_j, D_j(d))$ by new $(n_j^*, D_j^*(d))$ if new next node or distance
  - Go to send step

55

Bellman-Ford Algorithm

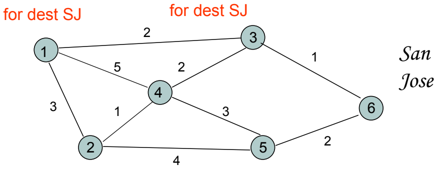
- *Now consider parallel computations for all destinations d*
- *Initialization*
  - Each node has 1 row for each destination *d*
  - Distance of node *d* to itself is zero: $D_d(d)=0$
  - Distance of other node *j* to *d* is infinite: $D_j(d)= \infty$, for $j \neq d$
  - Next node n_j = -1 since not yet defined
- *Send Step*
  - Send new distance vector to immediate neighbors across local link
- *Receive Step*
  - For each destination *d*, find the next hop that gives the minimum distance to *d*,
    - $Min_j \{ C_{ij}+ D_j(d) \}$
    - Replace old $(n_j, D_i(d))$ by new $(n_j^*, D_j^*(d))$ if new next node or distance found
  - Go to send step

56

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1					
2					
3					

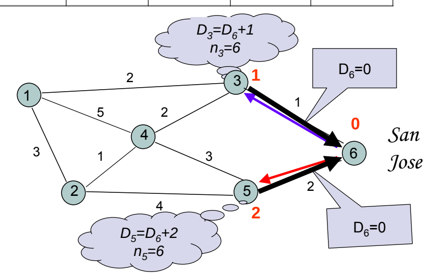
Table entry @ node 1 for dest SJ

Table entry @ node 3 for dest SJ



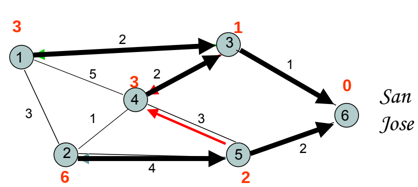
57

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1	(-1, ∞)	(-1, ∞)	(6,1)	(-1, ∞)	(6,2)
2					
3					



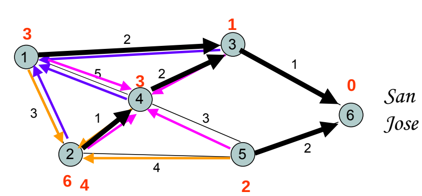
58

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6,2)
2	(3,3)	(5,6)	(6, 1)	(3,3)	(6,2)
3					



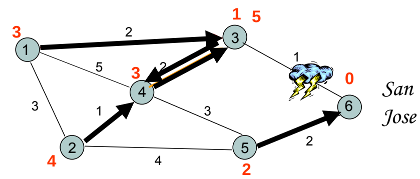
59

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6,2)
2	(3,3)	(5,6)	(6, 1)	(3,3)	(6,2)
3	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)



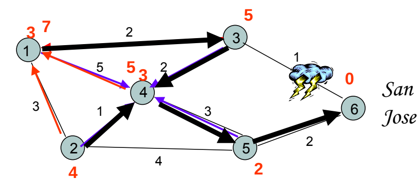
60

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2					
3					



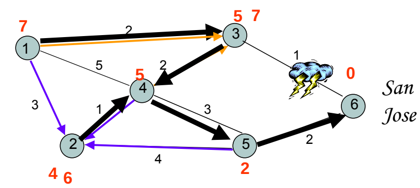
Network disconnected; Loop created between nodes 3 and 4

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2	(3,7)	(4,4)	(4, 5)	(5,5)	(6,2)
3					



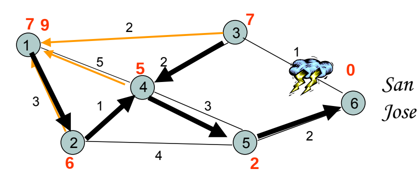
Node 4 could have chosen 2 as next node because of tie

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2	(3,7)	(4,4)	(4, 5)	(5,5)	(6,2)
3	(3,7)	(4,6)	(4, 7)	(5,5)	(6,2)



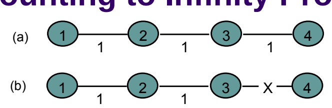
Node 2 could have chosen 5 as next node because of tie

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
1	(3,3)	(4,4)	(4, 5)	(3,3)	(6,2)
2	(3,7)	(4,4)	(4, 5)	(2,5)	(6,2)
3	(3,7)	(4,6)	(4, 7)	(5,5)	(6,2)
4	(2,9)	(4,6)	(4, 7)	(5,5)	(6,2)



Node 1 could have chose 3 as next node because of tie

Counting to Infinity Problem



Nodes believe best path is through each other

(Destination is node 4)

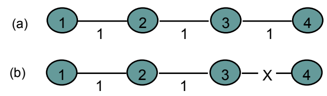
Update	Node 1	Node 2	Node 3
Before break	(2,3)	(3,2)	(4, 1)
After break	(2,3)	(3,2)	(2,3)
1	(2,3)	(3,4)	(2,3)
2	(2,5)	(3,4)	(2,5)
3	(2,5)	(3,6)	(2,5)
4	(2,7)	(3,6)	(2,7)
5	(2,7)	(3,8)	(2,7)
…	…	…	…

Problem: Bad News Travels Slowly

Remedies

- Split Horizon (Belief Propagation)
  - Do not report route to a destination to the neighbor from which route was learned
- Poisoned Reverse
  - Report route to a destination to the neighbor from which route was learned, but with infinite distance
  - Breaks erroneous direct loops immediately
  - Does not work on some indirect loops

Split Horizon with Poison Reverse



Nodes believe best path is through each other

Update	Node 1	Node 2	Node 3	
Before break	(2, 3)	(3, 2)	(4, 1)	
After break	(2, 3)	(3, 2)	(-1, ∞)	Node 2 advertizes its route to 4 to node 3 as having distance infinity; node 3 finds there is no route to 4
1	(2, 3)	(-1, ∞)	(-1, ∞)	Node 1 advertizes its route to 4 to node 2 as having distance infinity; node 2 finds there is no route to 4
2	(-1, ∞)	(-1, ∞)	(-1, ∞)	Node 1 finds there is no route to 4

67

Link-State Algorithm

- Basic idea: two step procedure
  - Each source node gets a map of all nodes and link metrics (link state) of the entire network
  - Find the shortest path on the map from the source node to all destination nodes
- Broadcast of link-state information
  - Every node *i* in the network broadcasts to every other node in the network:
    - ID's of its neighbors: $\mathcal{N}_i$=set of neighbors of i
    - Distances to its neighbors: $\{C_{ij} \mid j \in N_i\}$
  - Flooding is a popular method of broadcasting packets

68

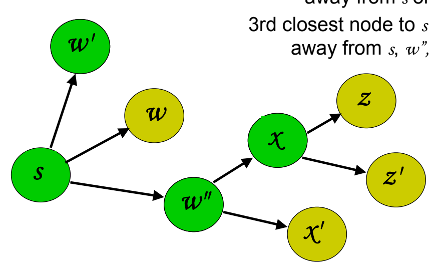
Dijkstra's Algorithm

Find shortest paths from source s to all other destinations

Closest node to s is 1 hop away

2nd closest node to s is 1 hop away from s or w''

3rd closest node to s is 1 hop away from s, w'', or x



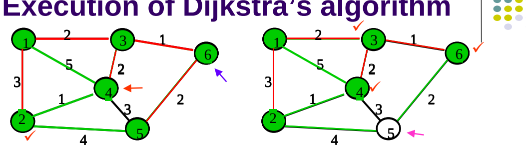
69

Dijkstra's algorithm

- *N*: set of nodes for which shortest path already found
- Initialization: (*Start with source node s*)
  - $N = \{s\}$, $D_s = 0$, "*s* is distance zero from itself"
  - $D_j = C_{sj}$ for all $j \neq s$, distances of directly-connected neighbors
- Step A: (*Find next closest node i*)
  - Find $i \notin N$ such that
  - $D_i = \min Dj$ for $j \notin N$
  - Add *i* to *N*
  - If *N* contains all the nodes, stop
- Step B: (*update minimum costs)*
  - For each node $j \notin N$
  - $D_j = \min (D_j, D_i + C_{ij})$ ← *Minimum distance from s to j through node **i** in N*
  - Go to Step A

70

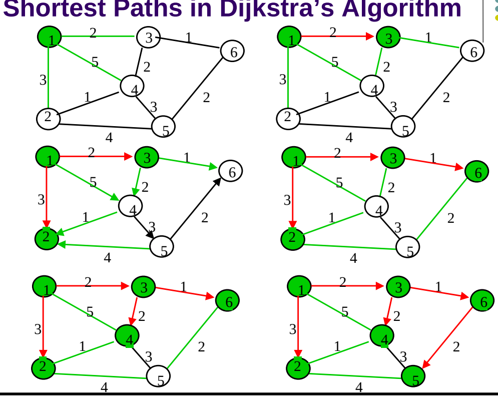
Execution of Dijkstra's algorithm



Iteration	N	D_2	D_3	D_4	D_5	D_6
Initial	{1}	3	2 ✓	5	∝	∝
1	{1,3}	3 ✓	2	4	∝	3
2	{1,2,3}	3	2	4	7	3 ✓
3	{1,2,3,6}	3	2	4 ✓	5	3
4	{1,2,3,4,6}	3	2	4	5 ✓	3
5	{1,2,3,4,5,6}	3	2	4	5	3

71

Shortest Paths in Dijkstra's Algorithm



72

Reaction to Failure

- If a link fails,
  - Router sets link distance to infinity & floods the network with an update packet
  - All routers immediately update their link database & recalculate their shortest paths
  - Recovery very quick
- But watch out for old update messages
  - Add time stamp or sequence # to each update message
  - Check whether each received update message is new
  - If new, add it to database and broadcast
  - If older, send update message on arriving link

Why is Link State Better?

- Fast, loopless convergence
- Support for precise metrics, and multiple metrics if necessary (throughput, delay, cost, reliability)
- Support for multiple paths to a destination
  - algorithm can be modified to find best two paths

The End

Complexity at the Edge or in the Core?