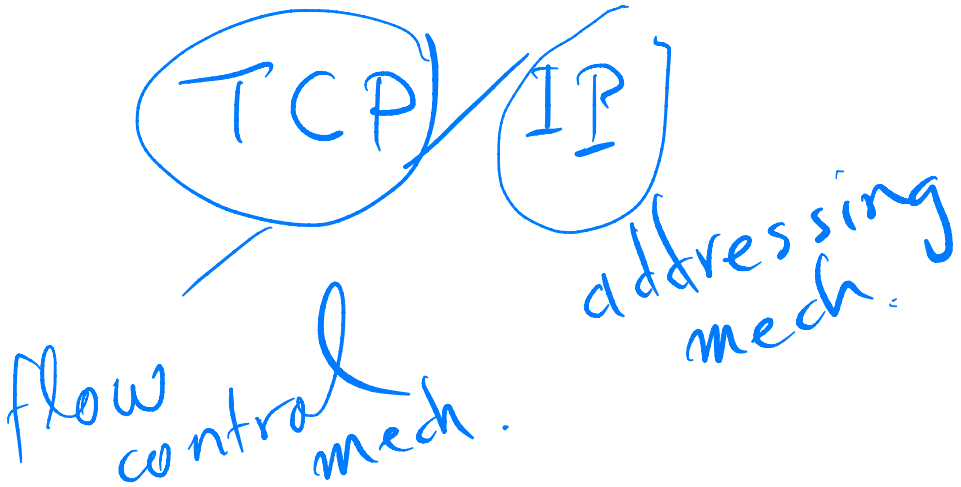
ROUTING



November 22, 2020

Outline

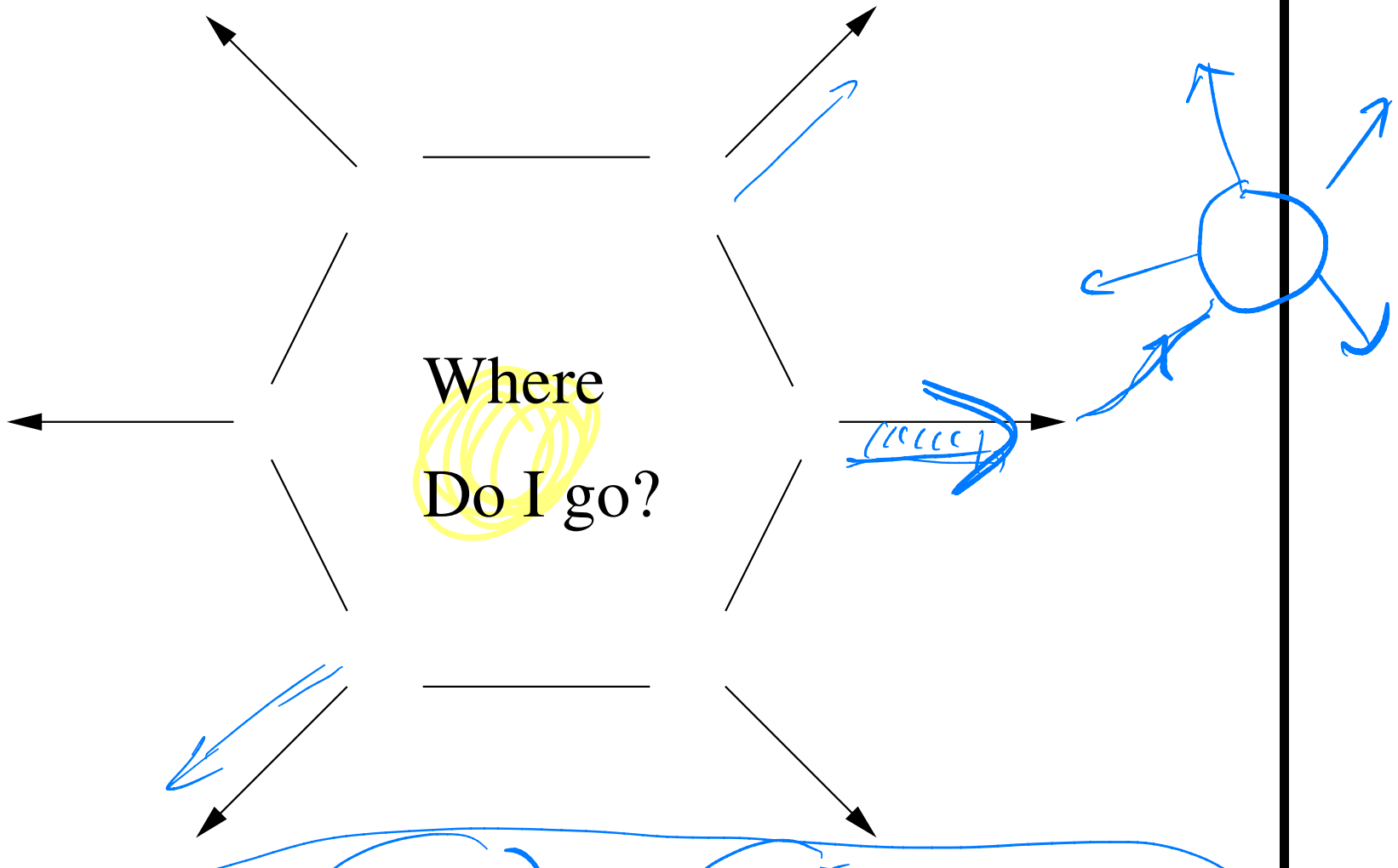
- Introductory concepts
- Distance Vector Routing (RIP)
- Link State Routing (OSPF)
  - Flooding
  - BFS and Dijkstra (Appendix)
- Micellaneous
  - Distance Vector vs Link State Routing
  - Routing in Mobile IP
  - Inter Domain Routing
- Spanning Trees (see Appendix)
  - Prim (Outline)
  - Kruskal (Outline)

used in the full range of networks. In any network which requires multi-hop transfers of packets.

Used in Backbone networks.

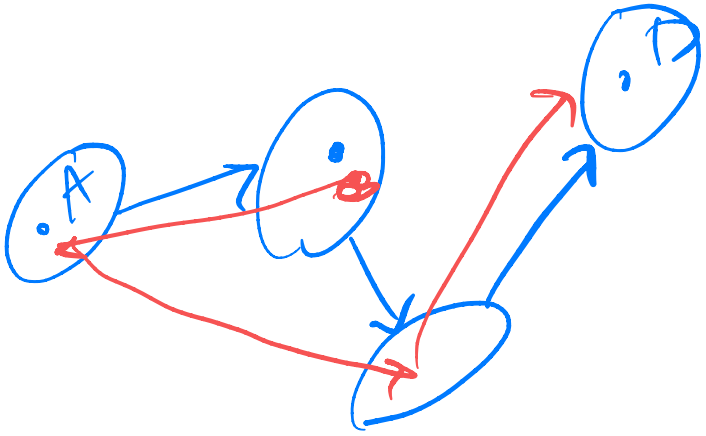
Introduction

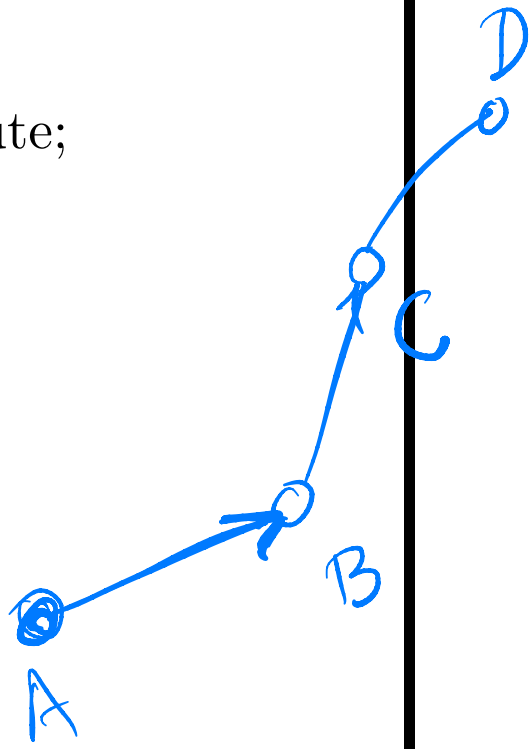
The Routing Concept



Routing vs Route Discovery

- **Routing** is a procedure (or algorithm)
  - being used to deliver packets between nonadjacent nodes in a point-to-point network and/or between subnetworks of a network.
- **Route Discovery** differs from Routing:
  - it is an algorithm being used to discover a route;
  - as such it must precede Routing.





Routing Table

next_hop

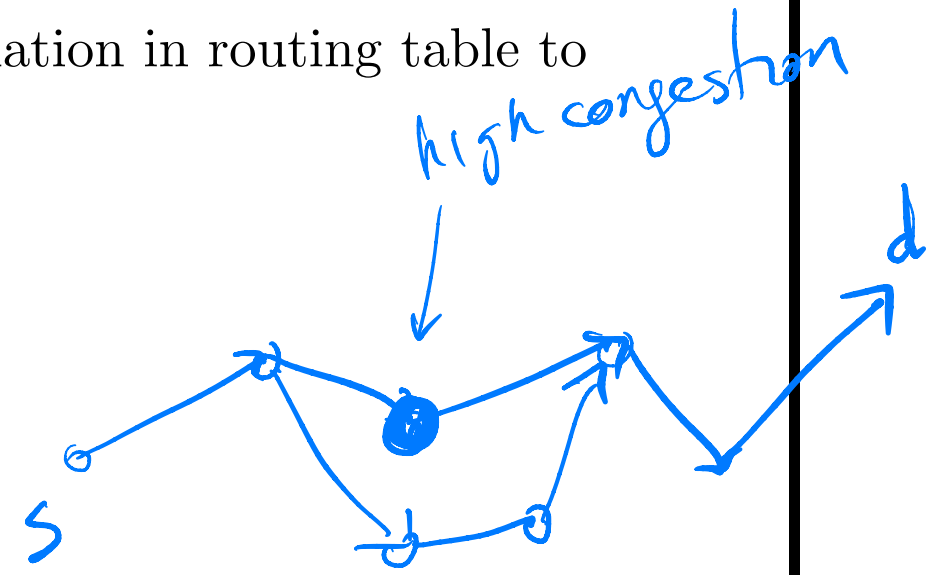
- A fundamental ingredient of routing is the routing table.
  - Standard routing table contains an entry for each possible destination with the out-going link to use for destination.
- Message delivery proceeds node-to-node in the obvious manner one link at a time, looking up next link in the table
- Variations on the above standard scheme exist:
  - Source routing: entries in the table contain the complete path from source to destination
  - Virtual circuit routing: routing tables are used to maintain virtual circuits between communicating nodes

VPN

source routing: $P: v_0, v_1, v_2, \ldots, v_n$

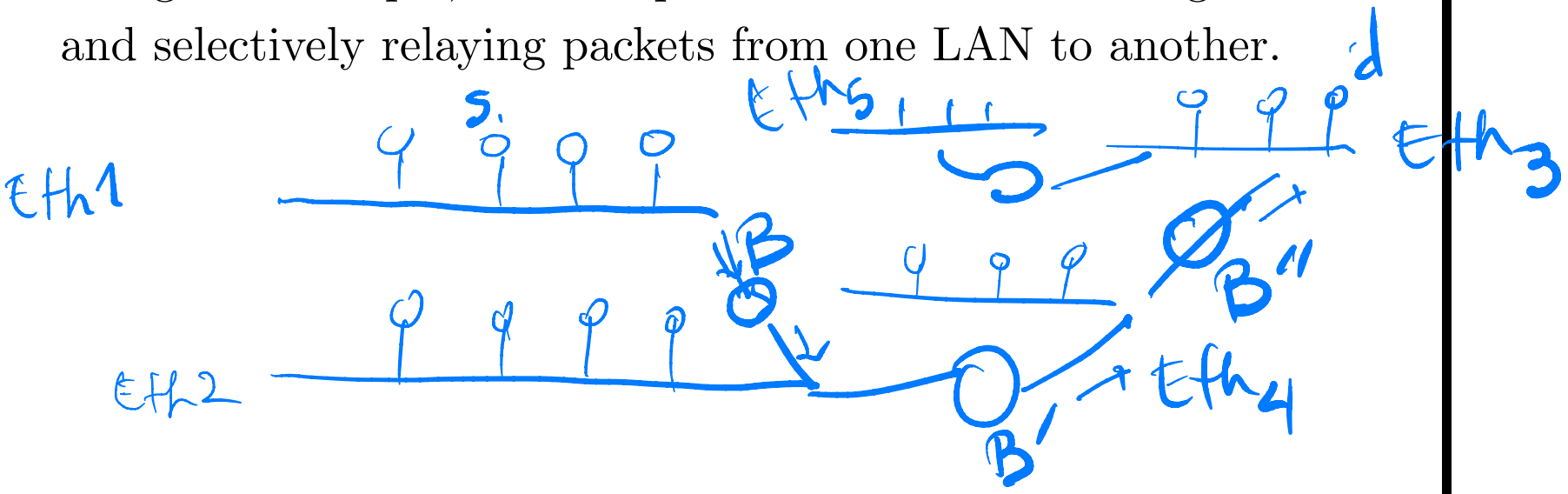
Routing

- Two main components of routing problem are
  - **Route selection:**
    determination of paths in the network from each source to each destination, i.e., construction and maintenance of *routing tables*
  - **Message delivery:**
    protocol for converting information in routing table to active packet forwarding



Need for Relaying and Optimization

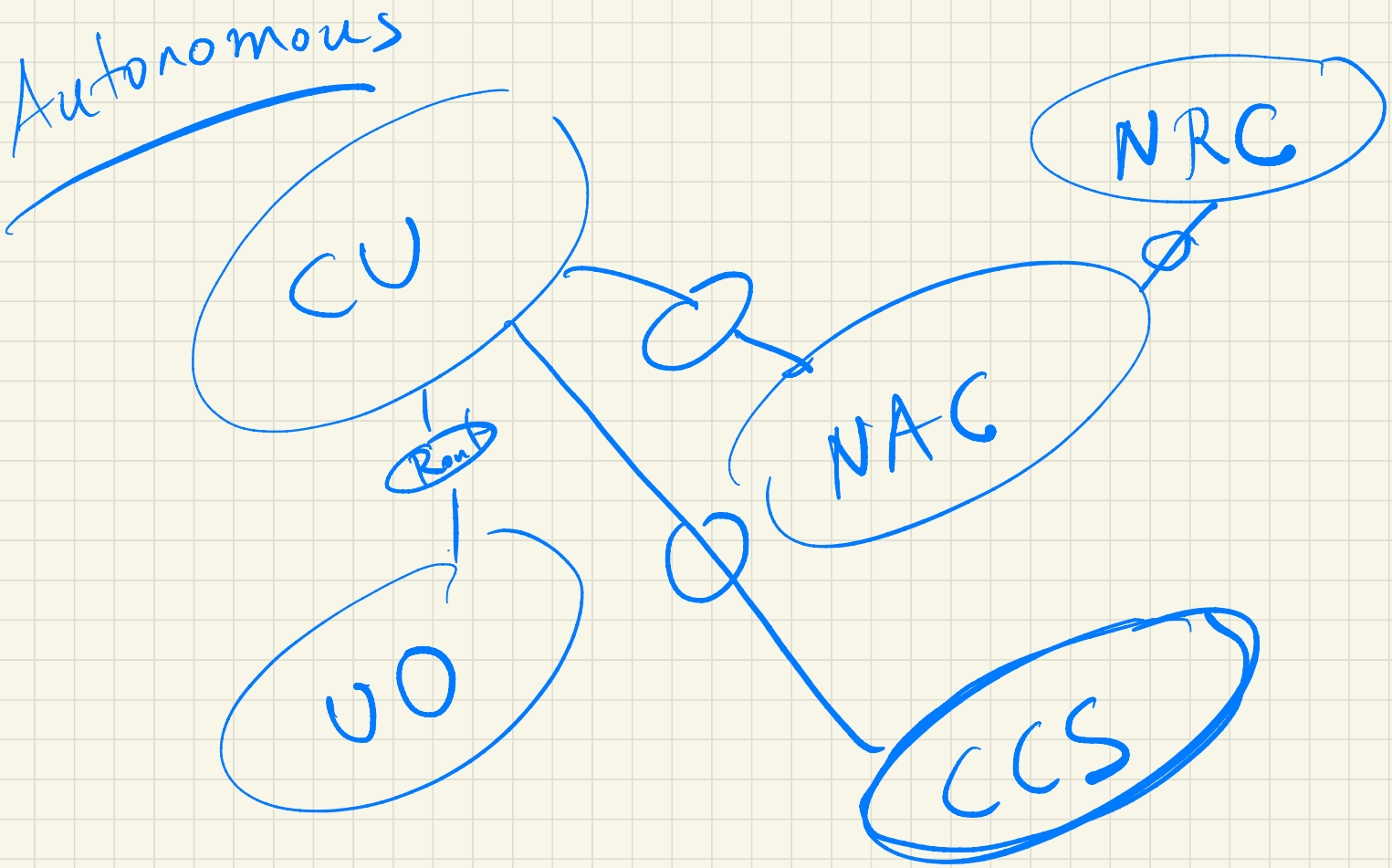
- Standard LANs like ethernet, token-ring, etc., have fixed topology (e.g., bus, star, ring, or arbitrary graph) and are limited to some maximum number of hosts
- LANs with the same MAC sublayer can be easily interconnected by using *bridges* to form a bigger LAN also called bridged LAN
- Bridges are simple, fast inexpensive routers connecting LANs and selectively relaying packets from one LAN to another.



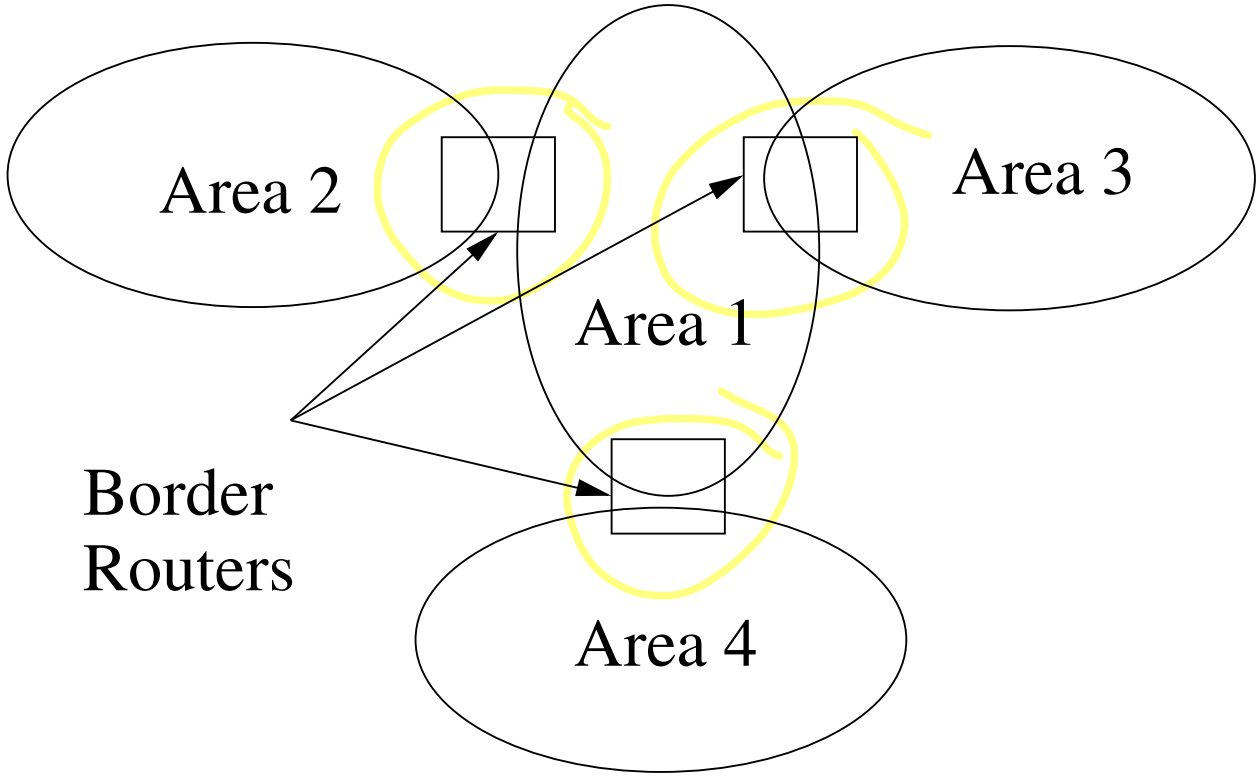
Network Units: Autonomous Systems (AS)[a]

- Subnetworks are aggregated to form larger networks.
- Routing becomes more complex and there is a need to optimize routing in aggregation consisting of smaller "network units".
- An autonomous system (AS)
  - consists of a number of subnets exchanging packets via routers that are using the same routing protocol, and
  - its routers are managed by a single or cooperating organizations
- Routing protocol used by an AS called *interior* routing protocol (IRP)
- Since all routers are managed by one organization the protocol can be optimized to best serve the users of the AS

[a] An AS may be a network used by a large company or organization.


Autonomous
CU
NRC
NAC
Rm
UO
CCS

Example



Internets

- In general, an internet will connect a number of different autonomous systems (ASs) run by many different organizations and running many different interior routing protocols
- Routers used to connect different ASs often called *gateways*
- Protocols used by gateways are called *exterior* routing protocols (ERP).
- Routing protocols operating at the "network frontier" are called *border gateway* protocols (BGP),

Two Routing Algorithms

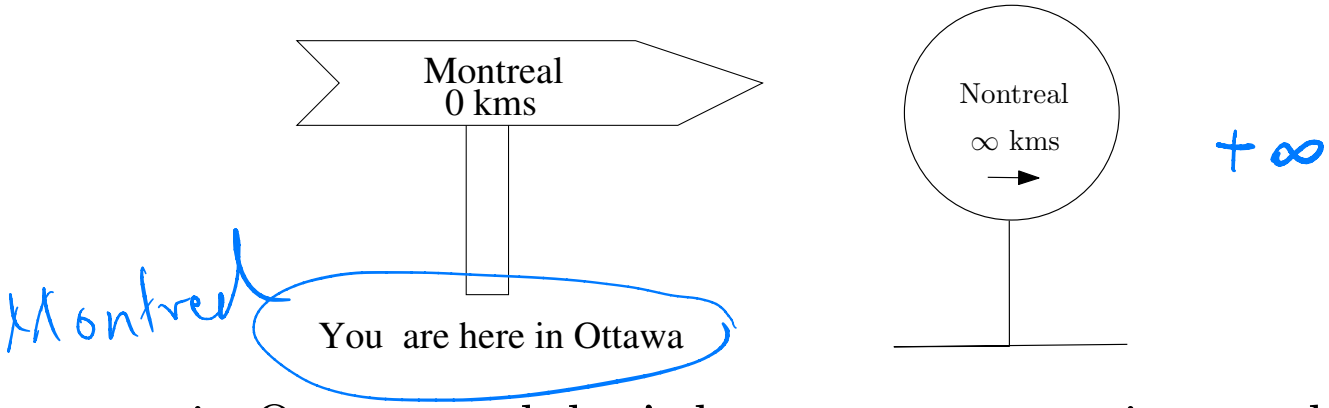
- All popular network layer protocols one way or the other are based on two types of distributed routing algorithms:
  - **Distance Vector**
    Also known as RIP (Routing Information Protocol) it is based on "Bellman-Ford" algorithm and is sometimes referred to as "old arpanet routing",
  - **Link State**
    Also known as LS Routing, it is based on Dijkstra's "shortest paths" algorithm.
- These algorithms are applicable in the entire range of networks: from wireless to wireline and optical.
- We will describe both of these algorithms, though in practice they are way more sophisticated.

Distance Vector (RIP)

November 22, 2020

Distance Vector Routing

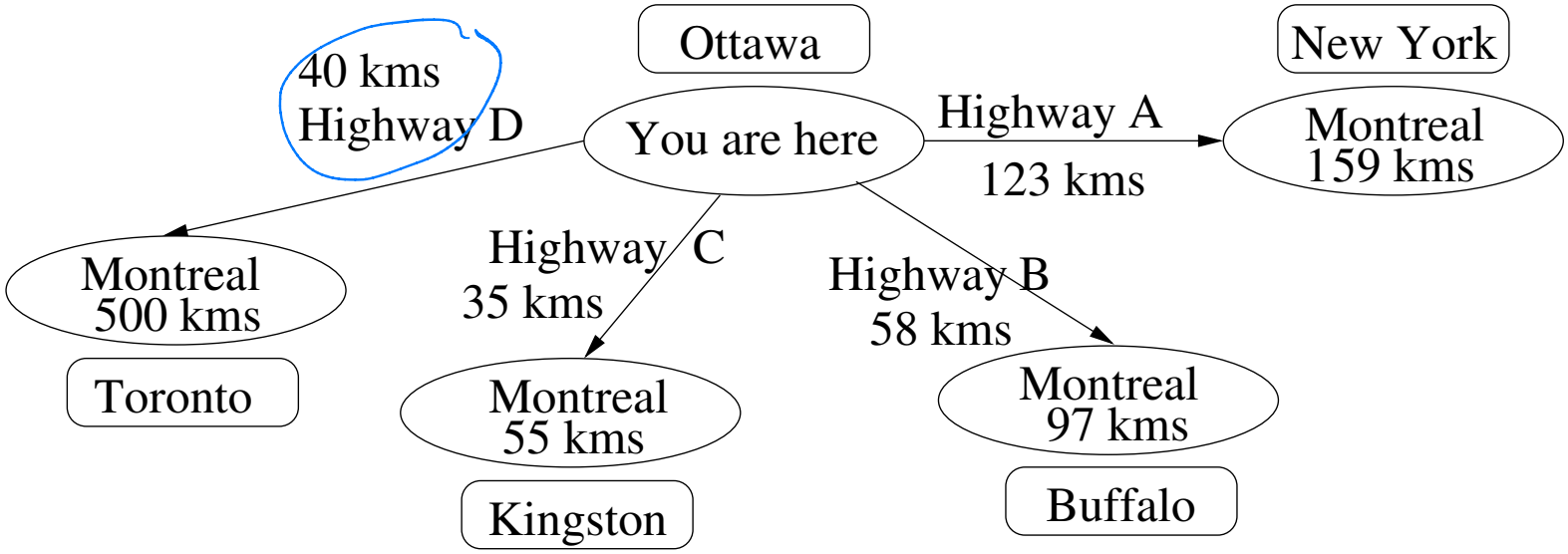
- You have been told to post distance signs (one for each city). You may not even know all the city names!
- If you are in Montreal, initially, you post sign depicted left.



  If you are in Ottawa and don't know you post a sign marked ∞.
- At every intersection there is someone doing the same thing!
- Measure distance to nearest intersection for each of the roads in your intersection and update your sign.
- Keep track of signs posted at each other intersection.

Distance Vector Routing

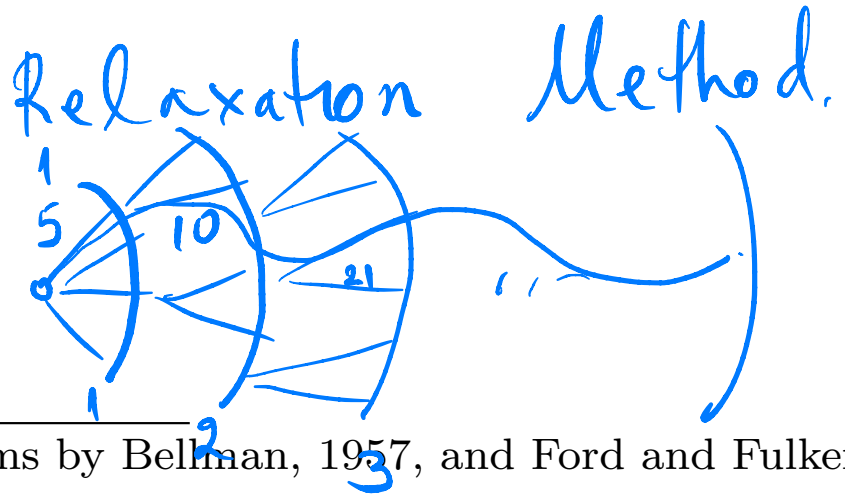
- Now calculate distance to each city by determining which direction gives the smallest distance.



- For example, to go to Montreal if you follow Highway D, it will require 40 kms to next intersection, and there you find an intersection with a new sign posted that says 500 kms to Montreal..

Distance Vector Routing: Idea[a]

- Initializes distance to the source to 0 and all other nodes to ∞.
- Now update distances.
  - For all edges, if the distance to the destination can be shortened by taking the edge, the distance is updated to the new lower value.
- At ith iteration when edges are scanned, the algorithm finds all shortest paths of at most length i (in at most edges).



[a]Based on algorithms by Bellman, 1957, and Ford and Fulkerson, 1962

Distance Vector Routing: Details (1/2)

- Uses the basic idea of shortest path routing, but handles topology changes. Forms Routing table as an array of triples

$$(destination, distance, nexthop).$$

- To send a packet to a given destination, it is forwarded to the process in the corresponding nexthop field of the tuple.
- In a graph with n nodes, the distance vector D for each node i contains n elements $D[i,0]$ through $D[i,n-1]$, where $D[i,j]$ defines the distance of node i from node j. Initially,

$$D[i,j] = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } j \text{ is a neighbor of } i \\ \infty & \text{otherwise} \end{cases}$$

- Can also use weighted link distances $w[i,j]$.

$$\left(D[i,0], D[i,1], \ldots, D[i,n-1]\right)$$

i

vertex you're now

$D[i,j]$

$=$

$+\infty$

$D[i,i] = 0$

Distance Vector Routing: Details (2/2)

- Each node j periodically broadcasts its distance vector to its immediate neighbors.
- Every neighbor i of j, after receiving the broadcasts from its neighbors, updates its distance vector as follows:

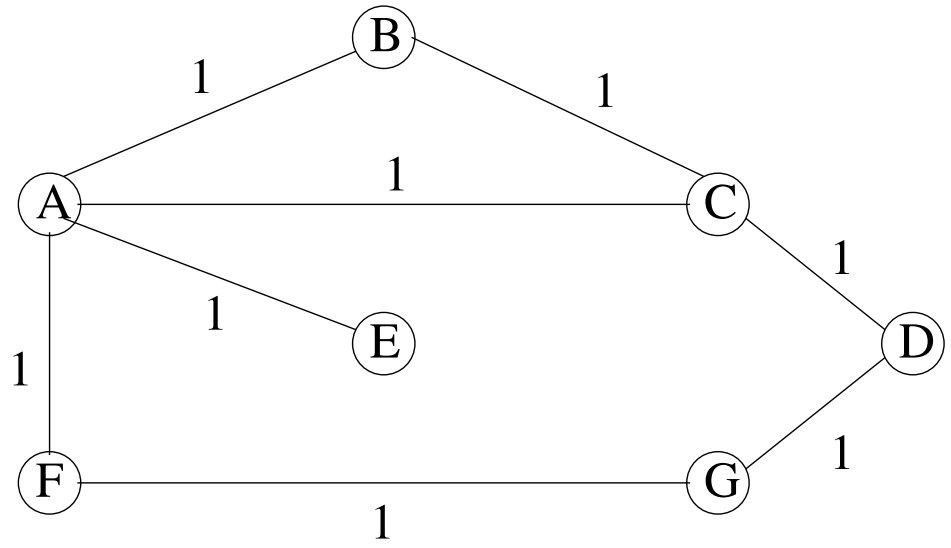
$$D[i,k] = \min_j\{w[i,j] + D[j,k]\}, \forall k \neq i$$

Relaxation

- When a node j or a link crashes, some neighbor k of it detects the failure, and sets the corresponding distance $D[j,k]$ to ∞.
- When new node joins, or an existing node is repaired, the neighbor detecting it sets corresponding distance to 1.
- Following this, the distance vectors are corrected, and routing table is eventually recomputed.

Example of Distance Vector Routing

- Each node maintains a one dimensional array (vector) of distances to all other nodes.



- For simplicity, in this example all link weights are equal to 1. Similar argument even when the weights are arbitrary positive numbers.
- Next you form a distance matrix between nodes of the network.

Distance Vector Routing: Vectors

- Each node maintains a one dimensional array (vector) of distances to all other nodes.

Node	Vector (of distances)
A	$(d_A^A, d_A^B, d_A^C, d_A^D, d_A^E, d_A^F, d_A^G)$
B	$(d_B^A, d_B^B, d_B^C, d_B^D, d_B^E, d_B^F, d_B^G)$
C	$(d_C^A, d_C^B, d_C^C, d_C^D, d_C^E, d_C^F, d_C^G)$
D	$(d_D^A, d_D^B, d_D^C, d_D^D, d_D^E, d_D^F, d_D^G)$
E	$(d_E^A, d_E^B, d_E^C, d_E^D, d_E^E, d_E^F, d_E^G)$
F	$(d_F^A, d_F^B, d_F^C, d_F^D, d_F^E, d_F^F, d_F^G)$
G	$(d_G^A, d_G^B, d_G^C, d_G^D, d_G^E, d_G^F, d_G^G)$

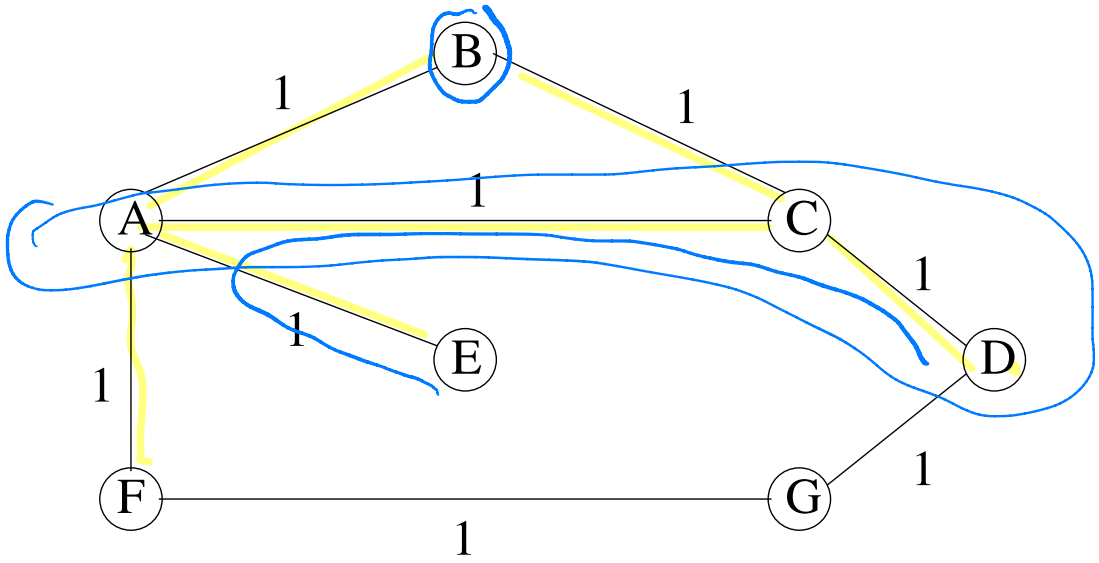
- E.g,. where d_u^v is u's view of its current distance from v.

Distance Vector Routing Algorithm

- Algorithm
  1. Nodes initialize the routing tables.
  2. Every node sends its vector to all its directlly connected neighbors.
  3. Every node after receiving this information updates its distance vector.
- Nodes maintain locally their own information: No node has all the information.
- **Convergence:** is the process of getting consistent routing information.
- Periodic updates are required in order to deal with failures.
- It is simple to implement this routing algorithm in software

Initial Routing Cost Tables

Desination (from A)	B	C	D	E	F	G
Distance (Cost)	1	1	∞	1	1	∞
NextHop	B	C	-	E	F	-



dist(E, D) = 3 hops

Initial Routing Cost Tables

- All nodes do the same.

Source/Destination	A	B	C	D	E	F	G
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

- Remaining nodes "try to" communicate their vector to A.

Computing Routing Cost Tables

- Routing algorithm is in rounds of
  - Send → Receive → Update
- In each round each node
  1. transmits its vector to all its neighbors
  2. receives vectors from all its neighbors
  3. updates its vector on the basis of what it receives

Computation at A

Look at the first round at A:

	A	B	C	D	E	F	G
$A's$ current vector	0	1	1	∞	1	1	∞
A sees $B's$ vector	1	0	1	∞	∞	∞	∞
A sees $C's$ vector	1	1	0	1	∞	∞	∞
A sees $D's$ vector	∞	∞	1	0	∞	∞	1
A sees $E's$ vector	1	∞	∞	∞	0	∞	∞
A sees $F's$ vector	1	∞	∞	∞	∞	0	1
A sees $G's$ vector	∞	∞	∞	1	∞	1	0
A computes new vector	0	1	1	2	1	1	2

Final Routing Cost Tables

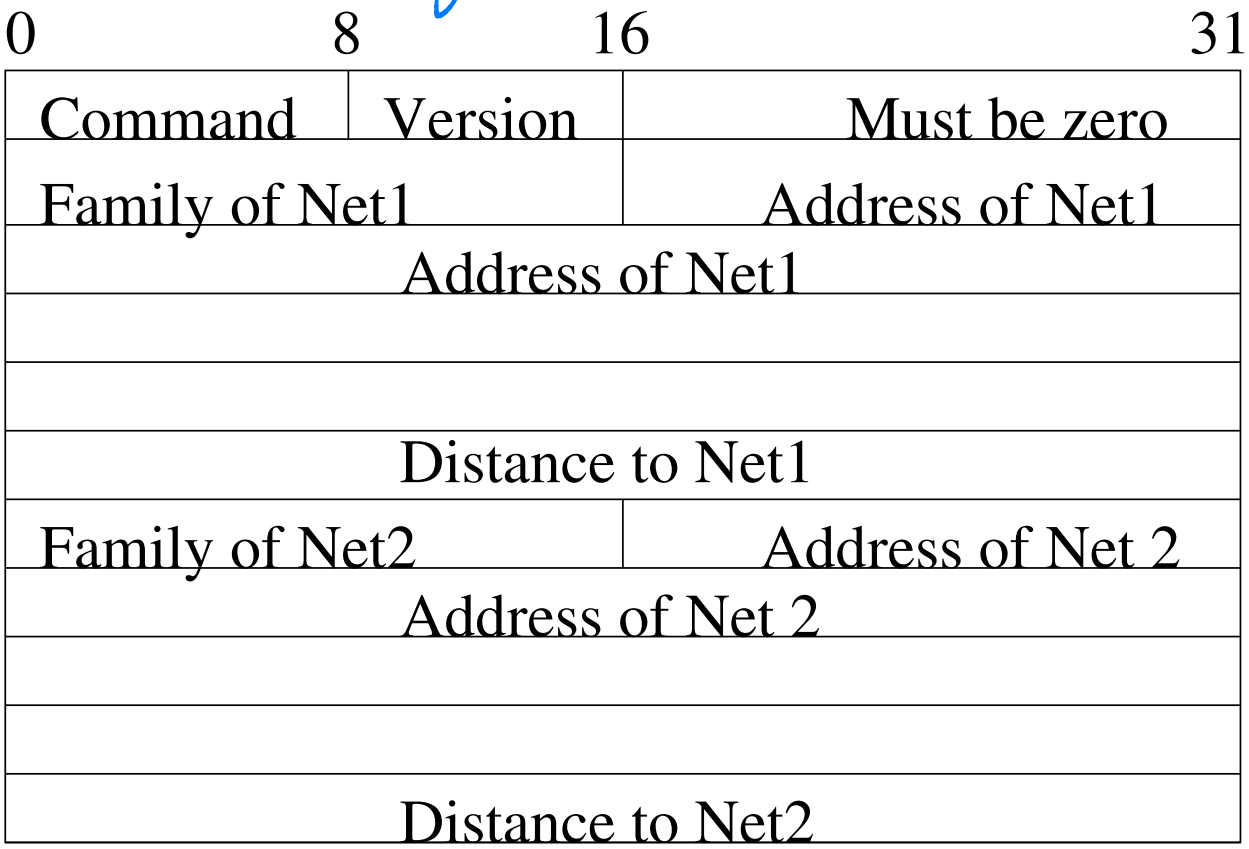
Desination (from A)	B	C	D	E	F	G
Cost	1	1	2	1	1	2
NextHop	B	C	C	E	F	F

Source/Destination	A	B	C	D	E	F	G
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

hops, delay, bandwidth

Routing Information Protocol (RIP)

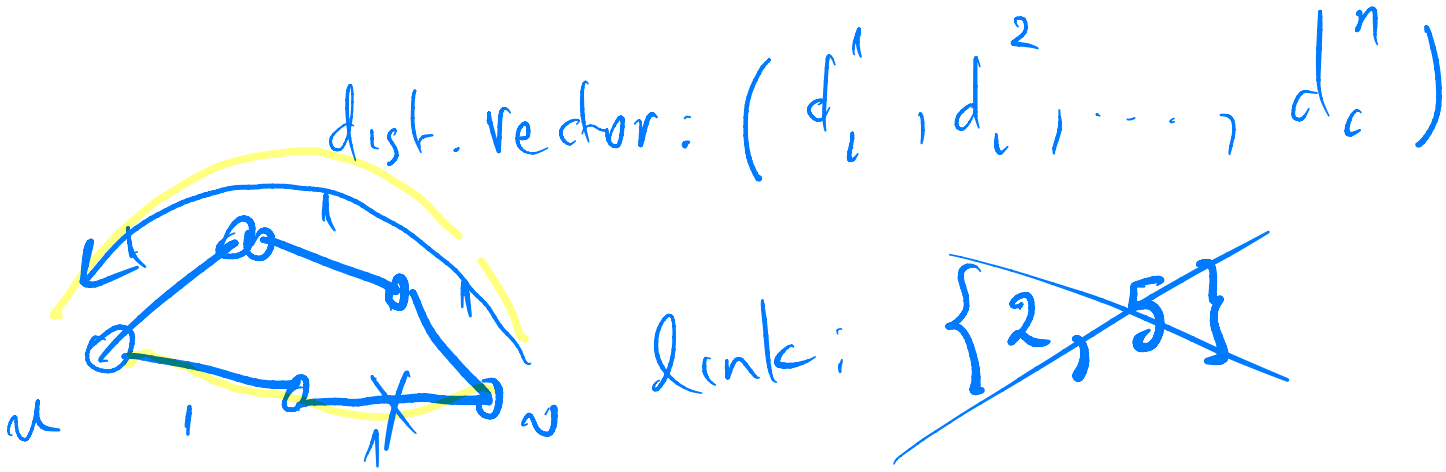
- RIP is based on distance vector routing, was distributed with UNIX BSD. Berkeley Distribution



- RIP runs advertisements every 30 sec, and is limited to 16 hops.

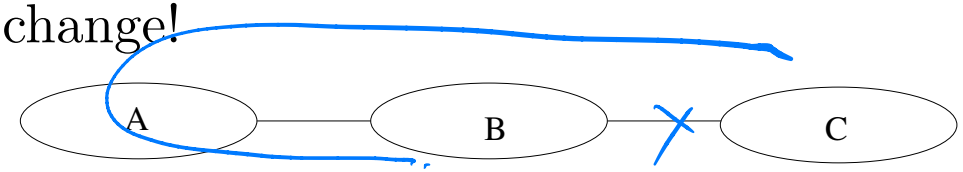
Fault Tolerant

- RIP is not fault tolerant.
- Can fail to give the correct answer even after a single topological change (e.g., link or vertex disruption).
- Sometimes this is due to the time it takes to converge to the correct answer.



Counting to Infinity!

- Distance vector can take a long time to converge after a single topological change!



- Metric is # of hops: C calculates distance to C as 0, B calculates distance to C as 1, A calculates distance to C as 2.
- Now assume C dies or that the link between B and C is broken!
  - B does not conclude immediately that C is unreachable.
  - Instead, it reports its distance to C as 3: because it knows its distance to A is 1, and that A's distance to C is 2!
- In the next iteration, B reports its distance to C as 4: because it knows its distance to A is 1, and that A's distance to C is 3!
- And so on...counting to infinity! In practice *infinity* is set to a fixed value, like 20!

Some Solutions

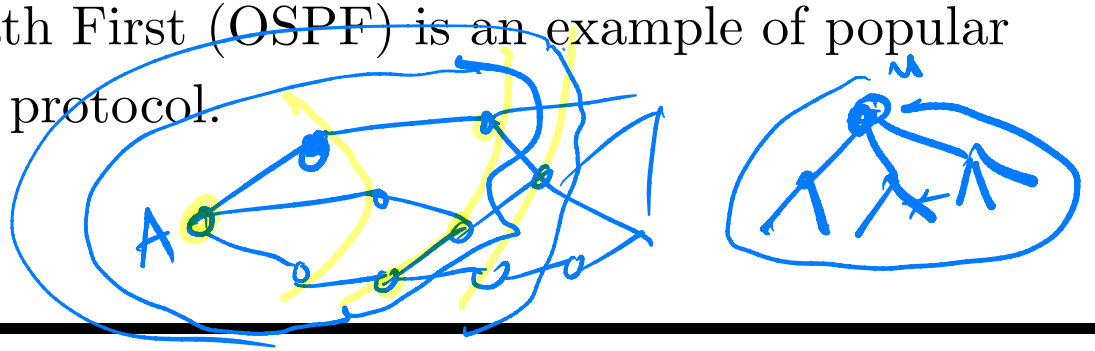
- **Hold Down:** If the path you are using goes down, you wait for some time before switching to another path.
- **Report Entire Path:** You do not report only cost of distances but also path to destination.
- **Multiple Metrics:** Compute routes based on more metrics, e.g. # of hops, link bandwidth, host speed, etc. Used by DECnet, IPX RIP.
- **Triggered Updates:** Periodically updates tables not when vector changes.
- **Split Horizon:** Look at previous example. If A forwards traffic for destination C through B then B reports to A that its distance is ∞.

Link State

Link-State Protocols

- First introduced for ARPANET to overcome slow response problem of RIP algorithms for use by interior routing protocols
- Each node maintains information on the state of all links of the network (i.e., global information).
  - This is formed from the Link State Advertisements (LSAs).
- When state of out-going link changes, node broadcasts this information to all nodes in the network using **flooding**.
- Each node computes locally its routing table (usually using single source version of Dijkstra's algorithm).
- Open Shortest Path First (OSPF) is an example of popular interior link-state protocol.

Satan



Calculating Maps and Shortest Paths

- The *first stage* in the link-state algorithm is to give a map of the network to every node.
  - Done with several simple subsidiary steps. 1) Determine the neighbours of each node, 2) Distribute the information for the map, and 3) Create the map.
- In the *second stage*, each node independently runs an algorithm over the map to determine the shortest path from themselves to every other node in the network; generally some variant of Dijkstra's algorithm is used.
- A node maintains two data structures: a tree containing nodes which are "done", and a list of candidates.
- The algorithm starts with both structures empty; it then adds to the first one the node itself.

Calculating Shortest Paths: Repeat:

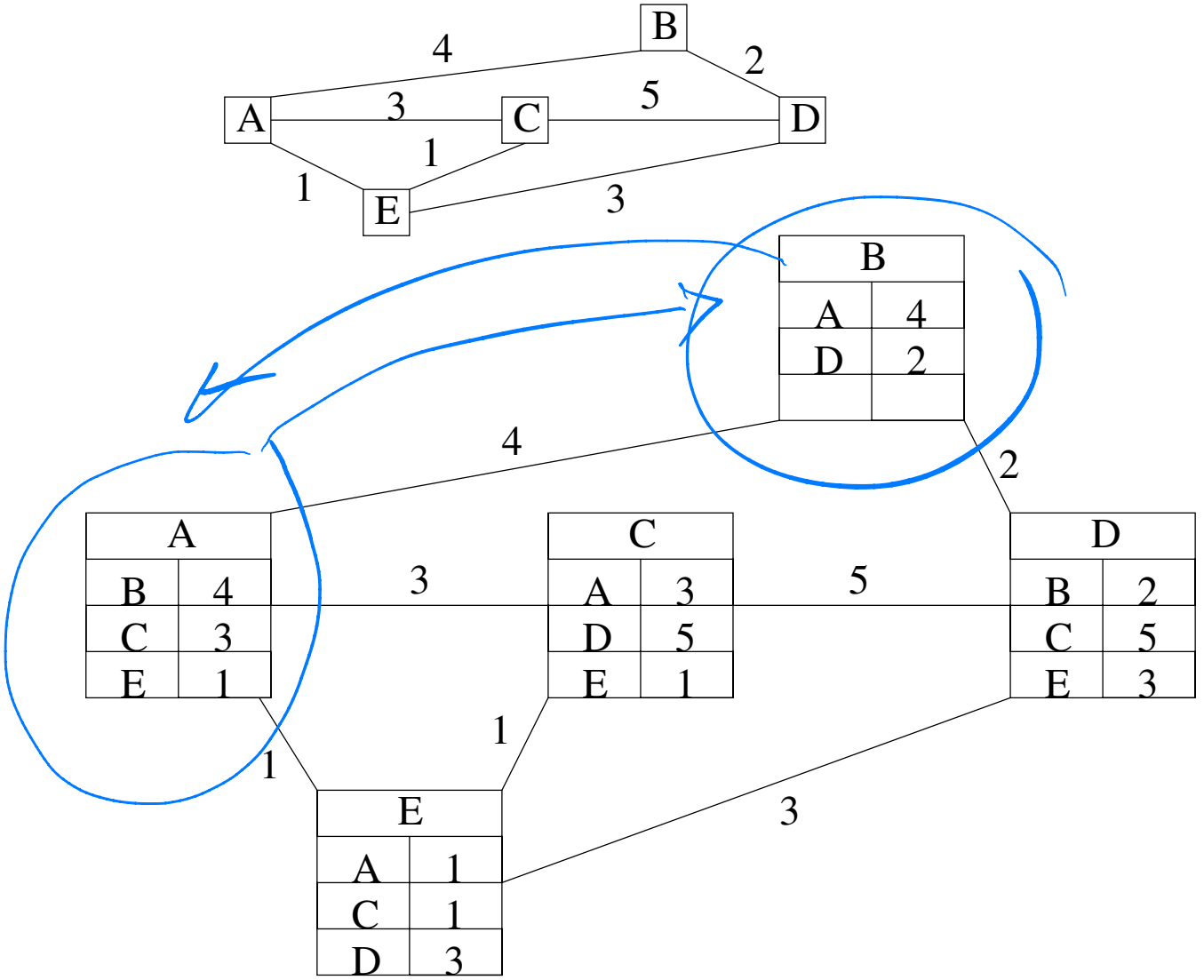
- Adds to the second (candidate) list all nodes which are connected to the node just added to the tree (excepting of course any nodes which are already in either the tree or the candidate list).
- Of the nodes in the candidate list, moves to the tree (attaching it to the appropriate neighbour node already there) the one which is the closest to any of the nodes already in the tree.
- Repeat as long as there are any nodes left in the candidate list.
- This procedure ends with the tree containing all the nodes in the network, with the node on which the algorithm is running as the root of the tree. The shortest path from that node to any other node is indicated by the list of nodes one traverses to get from the root of the tree, to the desired node in the tree.

Link State Protocol

1. Each router is responsible for information on its neighbors.
2. Each router constructs a Link State Packet (LSP) containing
   (a) ID of node that created packet
   (b) list of all neighbors together with cost
   (c) sequence number
   (d) a TTL (time to live) for this packet
3. LSP is transmitted to all other routers.
4. Each router (now knows the complete map of the network) and computes routes to each destination.

Route Updating/Calculation is done according to Dijkstra's algorithm.

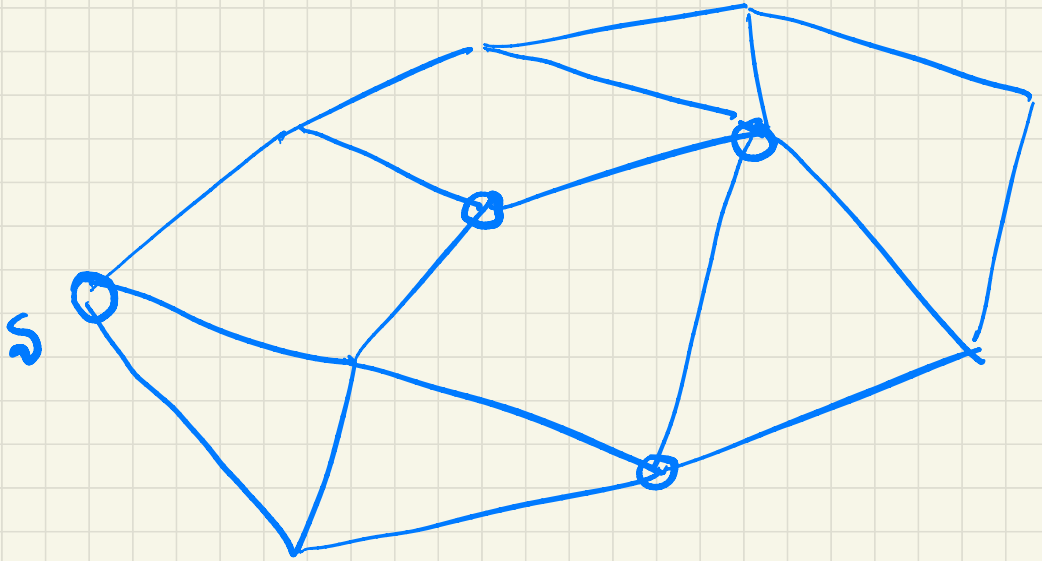
Link-State Protocols



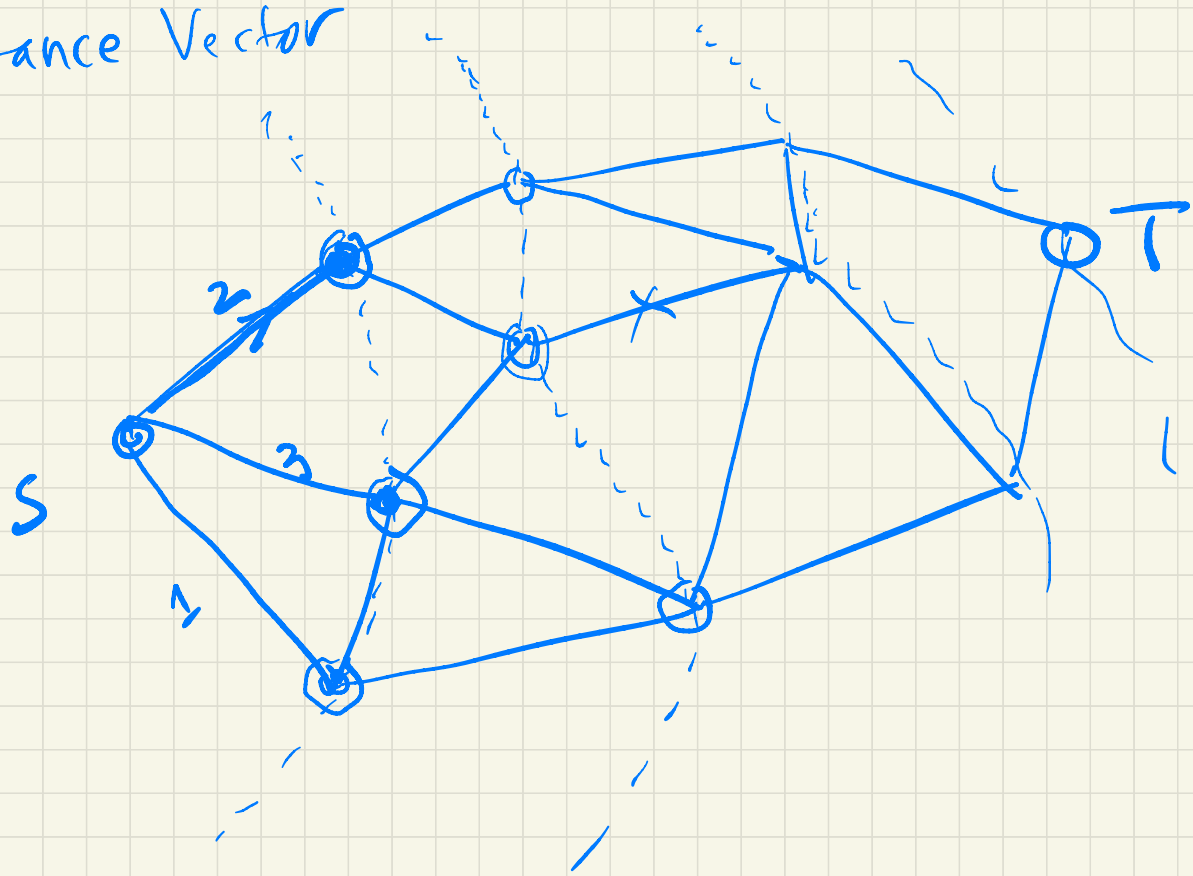
Two Routing Protocols

1) Distance Vector (Floyd's alg)

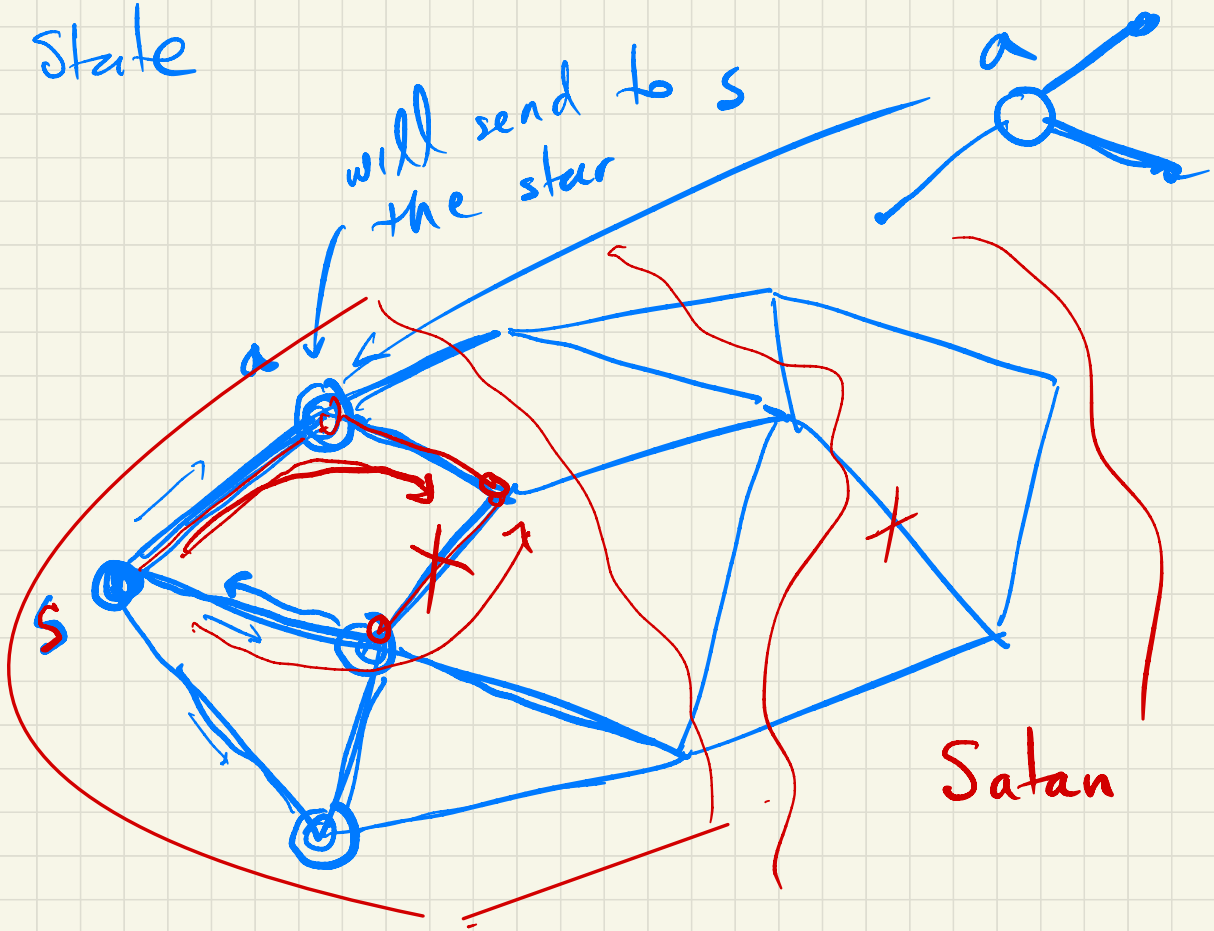
2) Link State (Dijkstra's alg).



Distance Vector



Link State



Issues in Link State Protocol

- A router generates an LSP when either it has new neighbors or the cost of a link to a neighbor has changed or link has gone down.

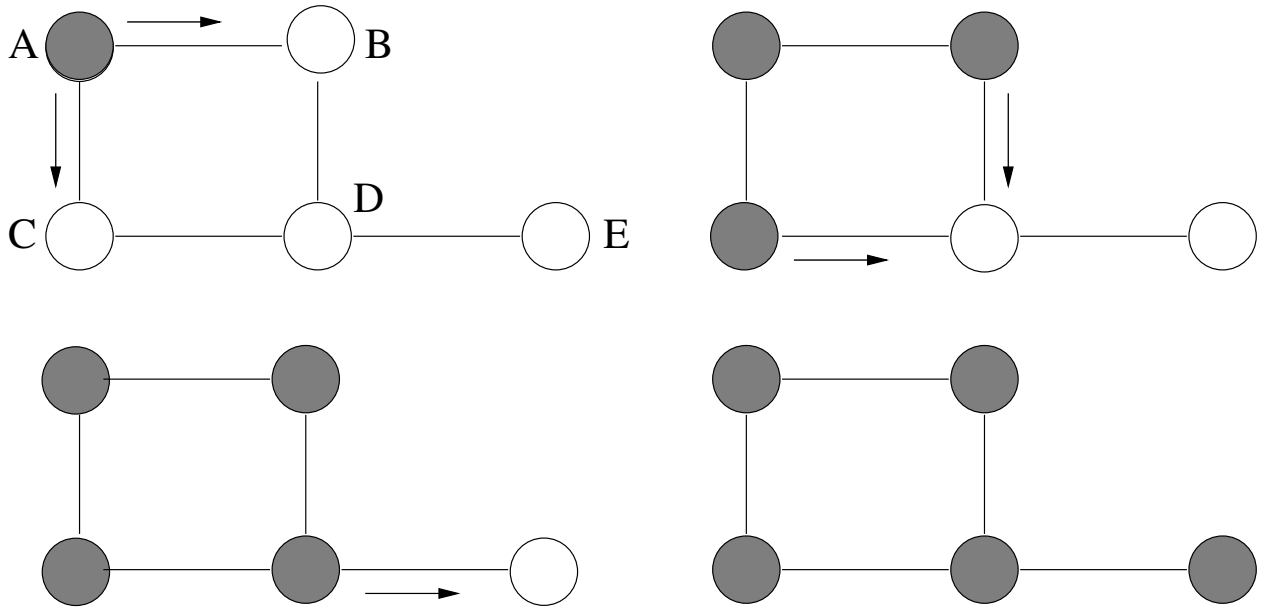
  Dijkstra's Algorithm

  1. Constructing the LSP
  2. Disseminating the LSP
  3. Timestamping LSPs (helps diecover most recent LSP).
  4. Sequence Number/Age Schemes (each router keeps track of sequence number it used last time).
  5. The Arpanet LSP contains: source, sequence number, age, list of neighbors.

- LSP generation uses *Flooding*.

Flooding

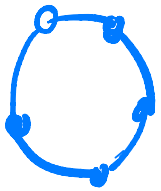
- Flooding sends a packet to all edges but the one you received the packet from.



- When change in state of a link occurs node sends a Link State Packet (LSP) to all of its neighbors
- When a node receives an LSP it forwards it to each of its neighbors except the one it received it from

New Information and Problems

- New Information
  - Newest information must be flooded as quickly as possible while old information be removed.
  - To minimize traffic one uses timers (in the order of hours) for LSP generation. LSPs carry sequence numbers up to 64 bits long.
  - LSPs also carry a time to live (TTL) information.
- **Some Problems With Flooding** (Is not good for Backbone)
  - As stated generates an infinite number of messages except in acyclic network
  - Slow dissemination and/or disconnections can lead to inconsistent view of the network
  - Spurious LSPs (possibly malicious) can introduce loops



Some Solutions

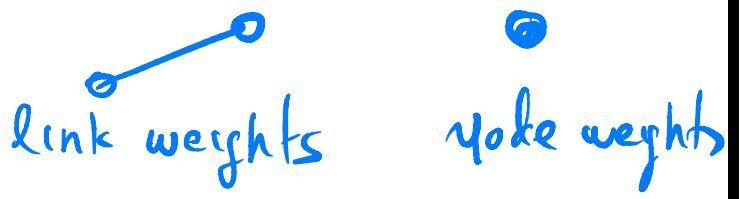
- Sequence numbers: LSP are given numbers and a database is maintained of highest number LSP received. Lower numbered LSPs are ignored.
- Wrapped sequence numbers:[a] If sequence has n possible values then a is older than b if (a) $a < b$ & $|b - a| < n/2$ or (b) $a > b$ & $|b - a| > n/2$ modulo arithmetic
- Aging: Sequence numbers are insufficient to deal with nodes that come up and start at 0. LSPs also have an age field which is incremented to max age at which point it gets discarded
- Authentication: LSPs are authenticated to ensure they were not created by malicious agent

[a] Modulo arithmetic.

Open Shortest Path First (OSPF)

Created under auspices of IETF, OSPF is an Open Standard for link-state interior protocol with:

- Sequence numbers to control flooding
- Age fields to deal with failures
- Authentication for security. (E.g., misconfigured host may advertise it can reach every node at cost 0, thus causing wrong LSP updating).
- Includes multiple link metrics



- Two level hierarchy of "areas" and "backbone" to allow for route aggregation
- Allows multiple routes to same destination thus distributing traffic evenly.

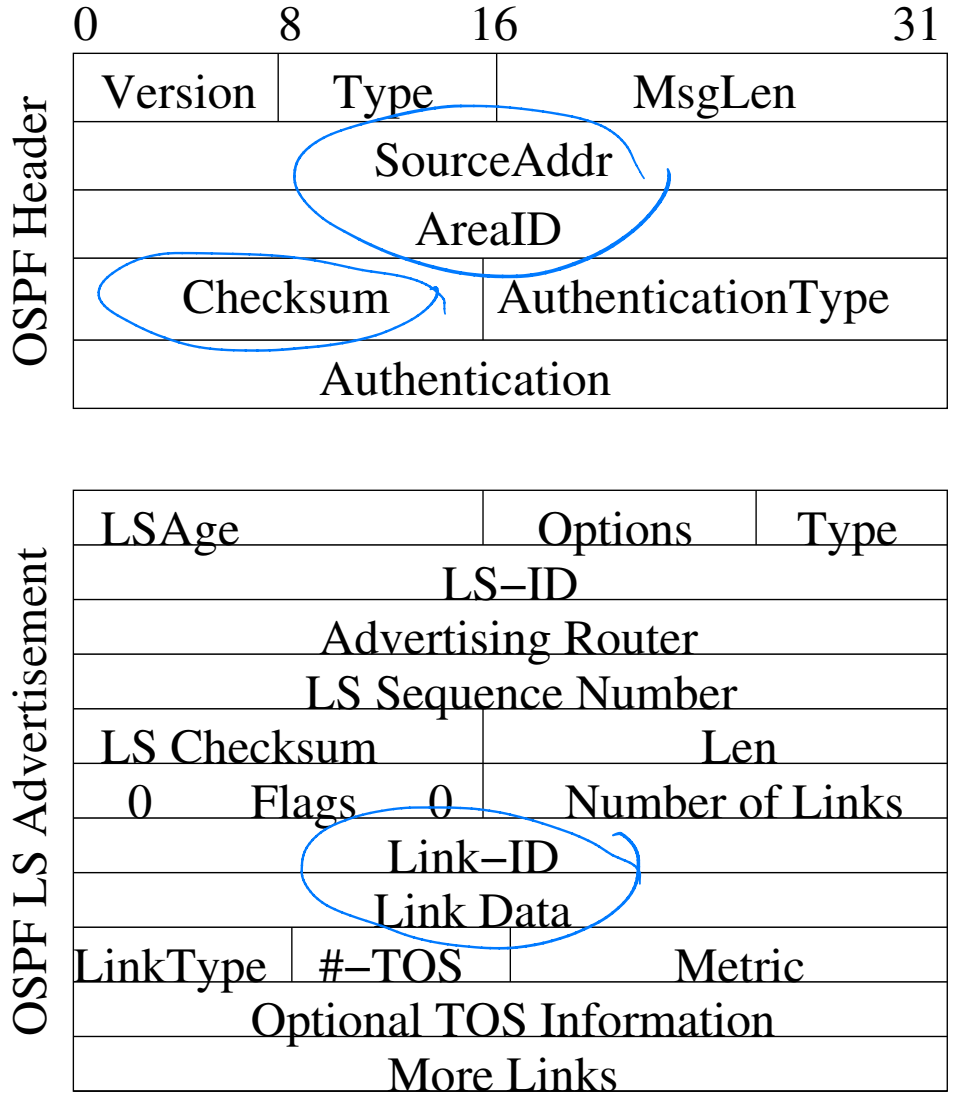
metric:

metric

(distance, delay, bandwidth)

average: $\frac{dist + del. + band}{3}$

OSPF Frames



Service Management

Service Agreement

Distance-Vector vs Link-State

- Most researchers favor link-state because:
  - Faster convergence after single change
  - Supports multiple paths for routing

Nodes need to transmit their views of the topology

- But some prefer distance-vector because:
  - Much simpler structure
  - Less storage space required
- There are techniques available for improving space: Interval Routing, Compact Routing, etc.

BFS

BFS Algorithm[a]

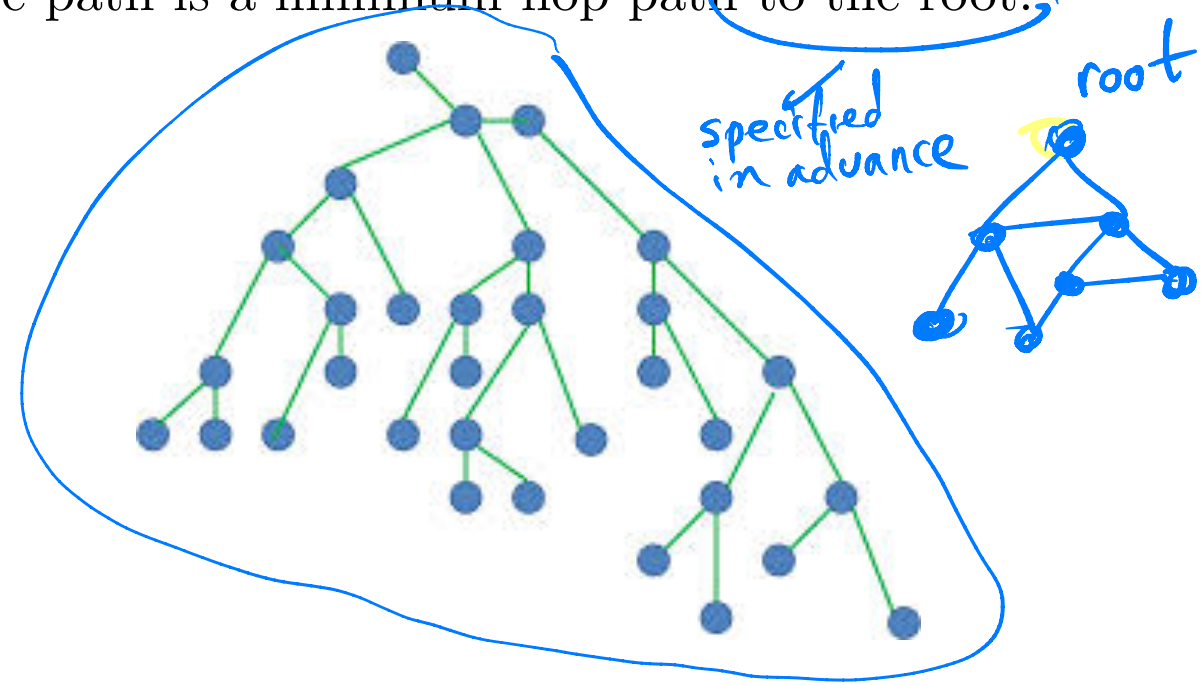
- LSP is based on Dijkstra's Tree Distributed algorithm
- Dijkstra's Tree Distributed algorithm is based on Dijkstra's algorithm (presented on the appendix).
- Dijkstra's algorithm is based on BFS algorithm.

Breadth First Search

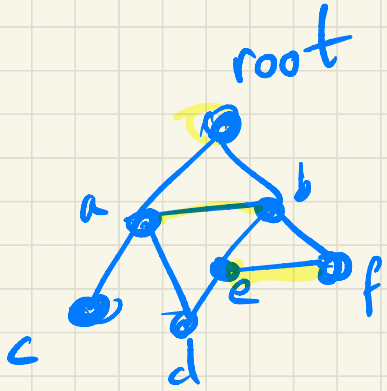
[a]Invented in 1945 by Konrad Zuse

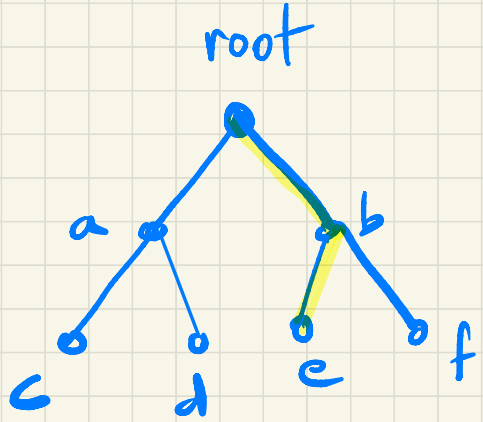
BFS Spanning Trees

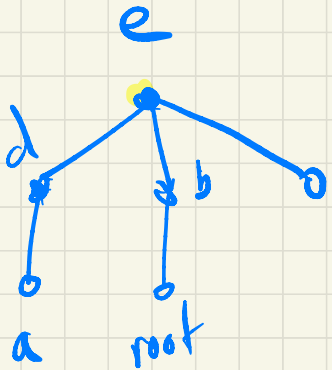
- Traversal of a graph is performed by visiting all of its vertices in some predefined order.
- **Breadth-First-Search Tree.** A breadth-first-search tree T of a graph G is a spanning tree of G such that for every node of G, the tree path is a minimum-hop path to the root.



- Of course the root must be specified!







Shortest path from e to the root is

~~Sortest path from e to f is not~~

BFS Algorithm[a]

- **BFS Algorithm:** Input a graph $G = (V, E)$
  Proceed by layers,
  1. mark the root r;
  2. mark all neighbor vertices that are one hop away from r;
  3. mark vertices that are one hop away from these neighbors, which are two hops away from r; have not been included before
  4. and so on.
- It uses a FIFO queue
- It checks whether a vertex has been discovered before enqueueing the vertex rather than delaying this check until the vertex is dequeued from the queue

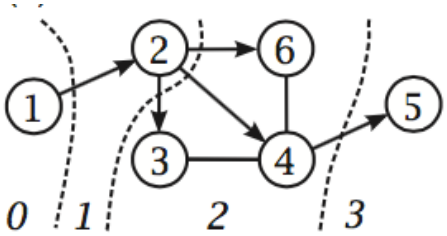
[a] Invented in 1945 by Konrad Zuse

BFS (Distance Computation (1/2))

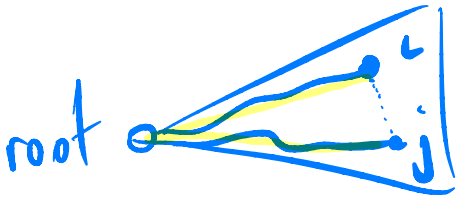
- It starts by placing the source node s at distance $d(s) = 0$; the distance of all other nodes starts as $d(i) = \infty$. undiscovered
- At the kth step (starting at $k = 0$), all nodes i at distance $d(i) = k$ are examined, and any neighbors j with $d(j) = \infty$ undiscovered have their distance $d(j)$ set to $k + 1$.
- The process halts when step k finds no such neighbors; $d(j)$ is then the length of the shortest path from s to j, or $d(j) = \infty$ if there is no such path.

BFS (Distance Computation (2/2))

- BFS is the simplest way to search a graph.
- It is suited only for unweighted graphs: ignores edge weights.
- **Example:**



- **Example:**
  In a social network, your friends are at level one and your friends of friends are at level two in a BFS starting at your node.



What is BFS Tree Used for?

- Finding all nodes within one connected component
  - BFS by itself is not enough: some message passing is needed!
- Finding the shortest path between two nodes u and v (with path length measured by number of edges)
  - u and v could be the nodes initiating a BFS tree.
- Doing efficient broadcast
  - from any any node.

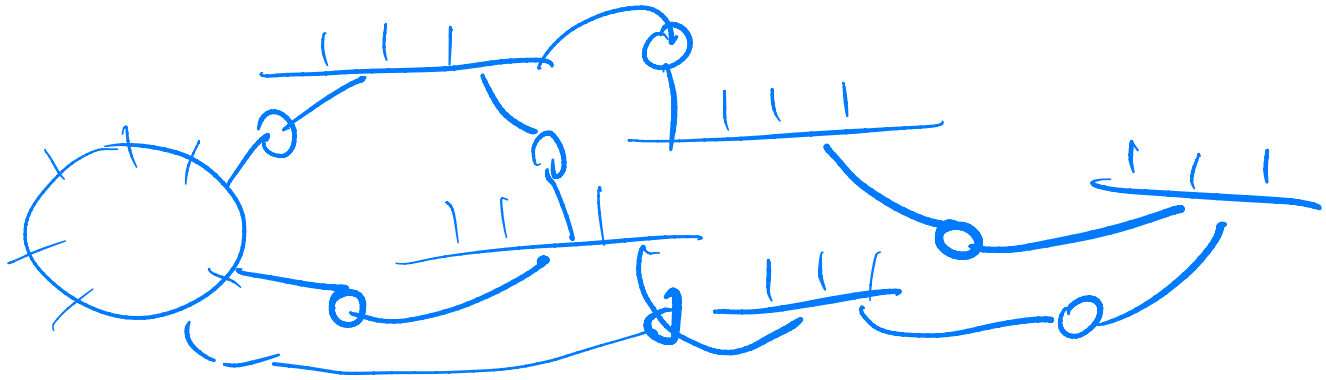
Spanning Trees

November 22, 2020

Bridged LANs

Bridges connect LANs at the MAC sublayer

- The resulting network is nonhierarchical
- Two standard methods of performing routing on bridged LANs are:
  - Spanning tree routing
  - Source routing
- Both schemes assume unique IDs for nodes and allow nodes to be turned off/on and to move from one location to another

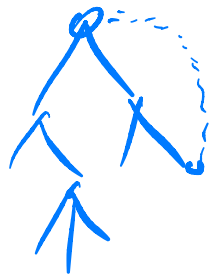


Spanning Tree Routing

- Bridge has *ports* corresponding to each of the LANs it is connected to
- For each port it maintains a Forwarding DataBase (FDB) which lists nodes with which it communicates via that port
- Bridges listen on each port and forward packets from one LAN to another using the FDBs
- Each node appears in exactly one FDB resulting in a *spanning tree* of the network with unique paths between each pair of nodes

What is a spanning tree?

There is a connection to each original node of the network but there are no cycles



Gabriel Test

Dynamic ST routing

- Due to nodes going down, coming up and/or changing location the FDBs of bridges are not always complete (i.e., the spanning tree may change)
- FDBs are updated using *bridge learning*:
  - FDB initialized to empty
  - Add source IDs to FDB
  - Delete inactive nodes
  - Initiate search and respond for unknown IDs

Failures

- Methods are required to deal with failures (and recoveries) of bridges or LANs
- Centralized schemes use MST algorithm to compute new spanning tree and update FDBs
- Distributed schemes use spanning tree to a fixed leader
- If leader goes down then distributed leader election algorithm is invoked

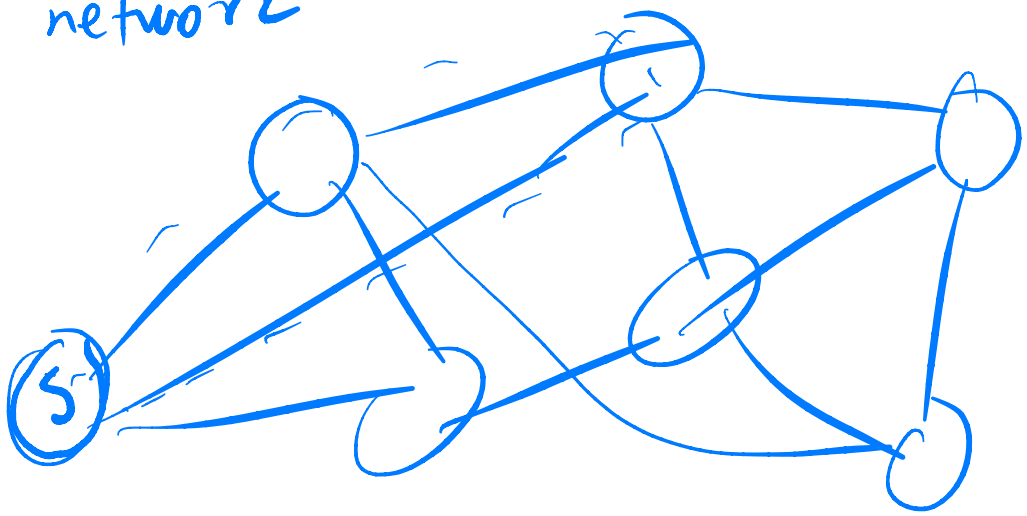
Kruskal's
Jarnik's
Gallagher's

MST: a spanning tree s.t. the sum of weights of all its edges is minimized.

MSTs

- Minimum spanning tree algorithms are presented in the appendix

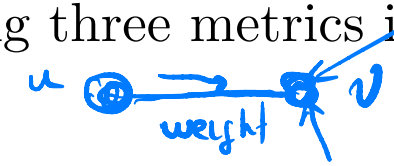
Very interesting topic
Where is it being used?
Backbone network



Miscellaneous

Measuring Performance of Routing

- Weights are placed on links according to either of several rules. ARPANET considered the following three metrics in historical order:

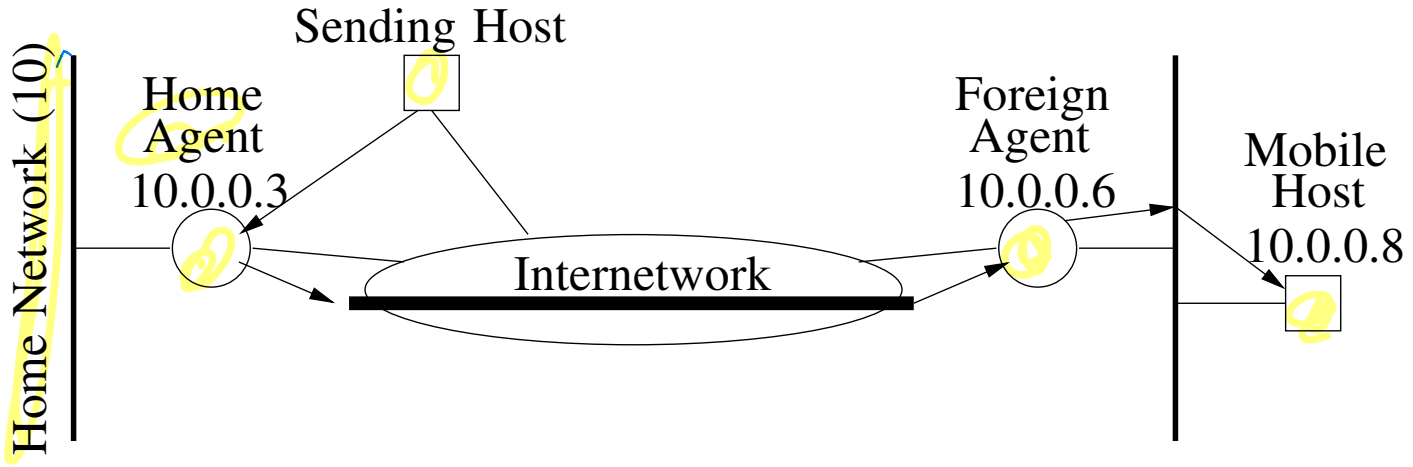


  1. **Queue Length Metric:** # of packets queued and waiting for transmission on each link.
  2. **Delay Metric:** Link bandwidth and latency (measured by timestamping packets).
     Delay = (DepartTime - ArrivalTime) + TransmissionTime + Latency.
  3. **Utilization Metric:** Link Utilization averaged over last reported utilization together with limits on how much the measurement could change over time from previous value.

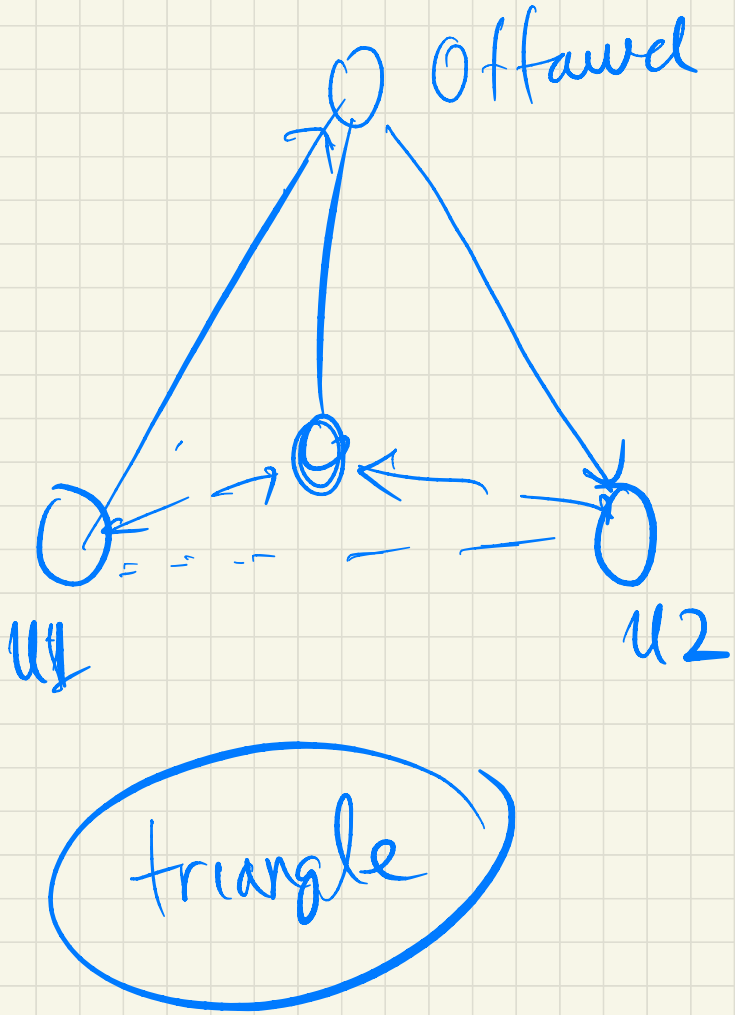
- SNMP (Simple Network Management Protocol) is a management tool for monitoring.

Routing for Mobile IP

- Mobile applications require smooth and transparent transition of usability from host-to-host and across platforms in the course of mobility.



- Routes provided may be suboptimal: Sending and Mobile node may be on the same network, but the home network of the Mobile nodes is far. This is known as the **triangle problem**. Solution is to let the sending node know the care-of address of the Mobile node.

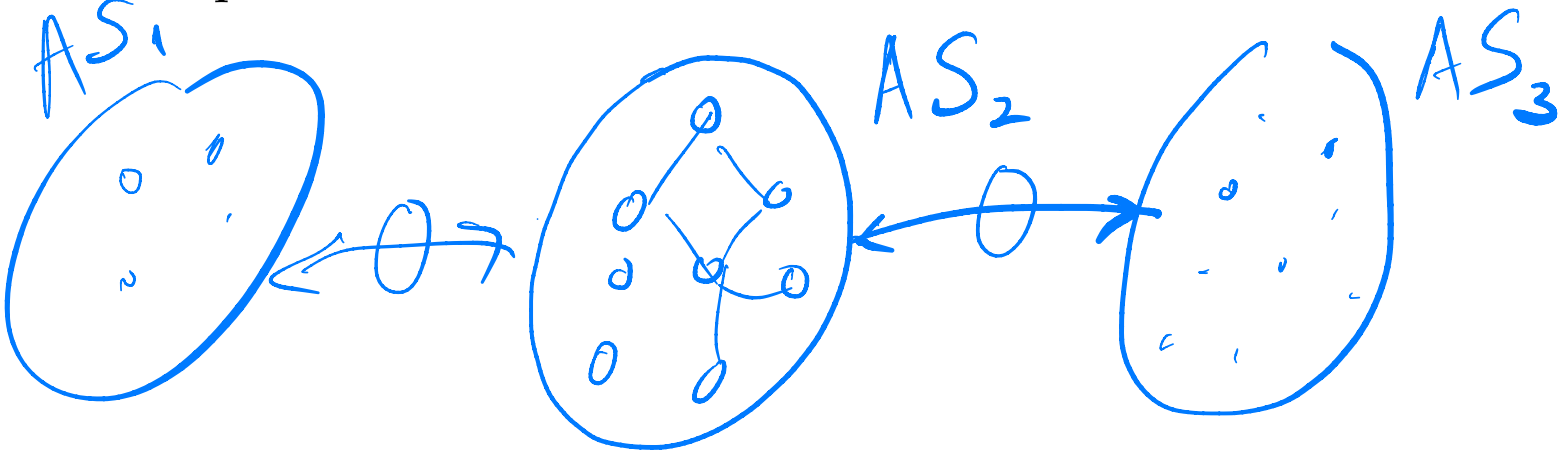

Offavel
u1
u2
triangle

Routing for Mobile IP

1. FA and HA periodically announce their presence to the networks they are attached. This way the MH learns the address of the HA.
2. When the MH attaches to a foreign network it hears an advertisement from a FA and registers with it.
3. The FA contacts the HA providing a care-of address. This is the IP address of the FA.
4. A Sending host that wants to send a packet to a MH will send it with a destination address equal to home address of that node.
5. The HA tunnels the packet and sends to the FA.
6. The FA unwraps the packet and sends it to the Mobile node.

Autonomous Systems

- An autonomous system consists of a number of subnets exchanging packets via routers that are using the same routing protocol
- Routers are managed by a single or cooperating organizations
- Routing protocol used by an AS called *interior* routing protocol
- Since all routers are managed by one organization the protocol can be optimized to best serve the users of the AS



Internets

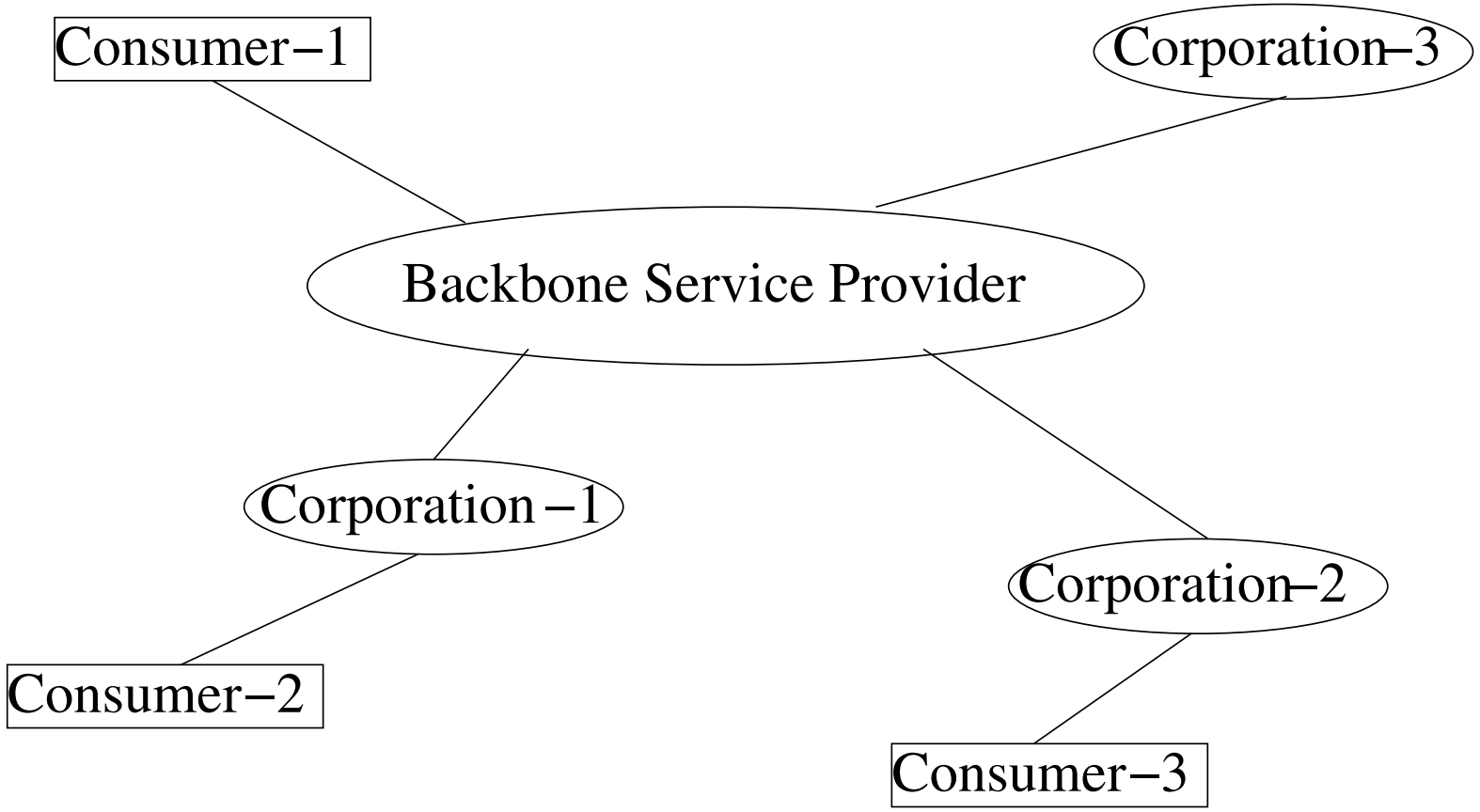
- An internet will connect a number of different autonomous systems run by many different organizations and running many different interior routing protocols
- Routers used to connect different ASs are often called *gateways*
- Protocols used by gateways are called *exterior* routing protocols

Bridged LANs

- Bridges connect LANs at the MAC sublayer
- The resulting network is non hierarchical
- Two standard methods of performing routing on bridged LANs are:
  - Spanning tree routing
  - Source routing
- Both schemes assume unique IDs for nodes and allow nodes to be turned off/on and to move from one location to another

remove

Inter Domain Routing in a Network of ASs



Classless Routing (CIDR)

- The IP-address A, B, C class system has inefficiencies. E.g., a two-host network with two hosts will have a class C address leading to a utilization of $\frac{2}{255}$.
- Efficiency improves if instead of allocating a class B address (amounting to 64K addresses) we provide a sufficient amount of class C addresses (i.e., blocks of 256 addresses). For Autonomous systems with more than 256 hosts this gives a utilization $\geq 50\%$.
- This raises the following problem: if an autonomous system is allocated n blocks of class C addresses then every backbone router needs n entries for this AS. If we had assigned a single class C address then we would need only one entry.
- How do we optimize address utilization and routing table size?

Classless Routing (CIDR)

- CIDR addresses this problem by aggregating routes.
- Rather than assigning class C addresses at random they are assigned as a block.
- E.g., 20 class C addresses in an interval of numbers from 195.34.17 195.34.36 can be assigned to an AS.
- The BGP (version 4) protocol is defined to understand this.
- All that is needed, is for the router to remember the interval.
- This can be done by remembering one of the two addresses and the range (in this case 20).

Inter Domain Routing

- Routing usually partitioned into inter-domain and intra-domaim.
- Inter-domain routing has its origins on ARPANET and its design was influenced by the internet.
  1. EGP (Exterior Gateway Protocol: early version of BGP)
  2. BGP (Border Gateway Protocol)
  3. IDRP (Inter Domain Routing Protocol)
- Protocols operate in conjunction with IP.
- Information exchanged between hosts via a connectionless protocol.

These are quite complex protocols

BGP Considerations

Interdomain routing is difficult and affected by

1. Scale factors,
2. Different routing protocols on different Autonomous Systems,
3. Security.

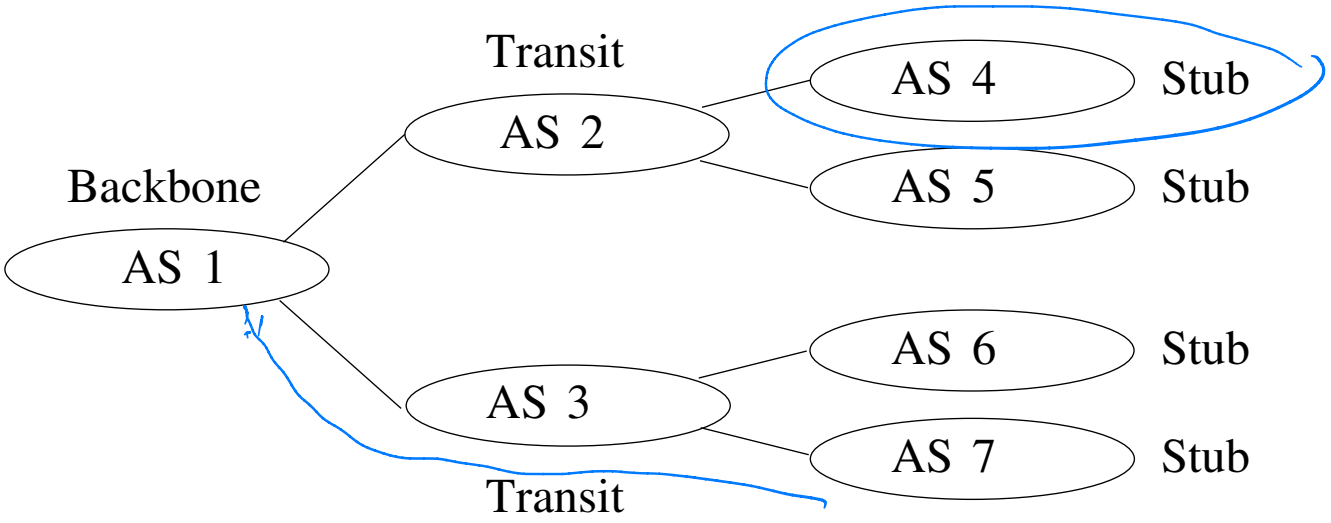
There are two types of traffic **local** and **transit**.

Interdomain routing assumes internet is a collection of Autonomous Systems. There are three types of ASs:

1. **stub AS:** has only a single connection to one other AS.
2. **multihomed AS:** has connections to more than one other AS but refuses to carry transit traffic.
3. **transit AS:** designed to carry transit traffic.

BGP

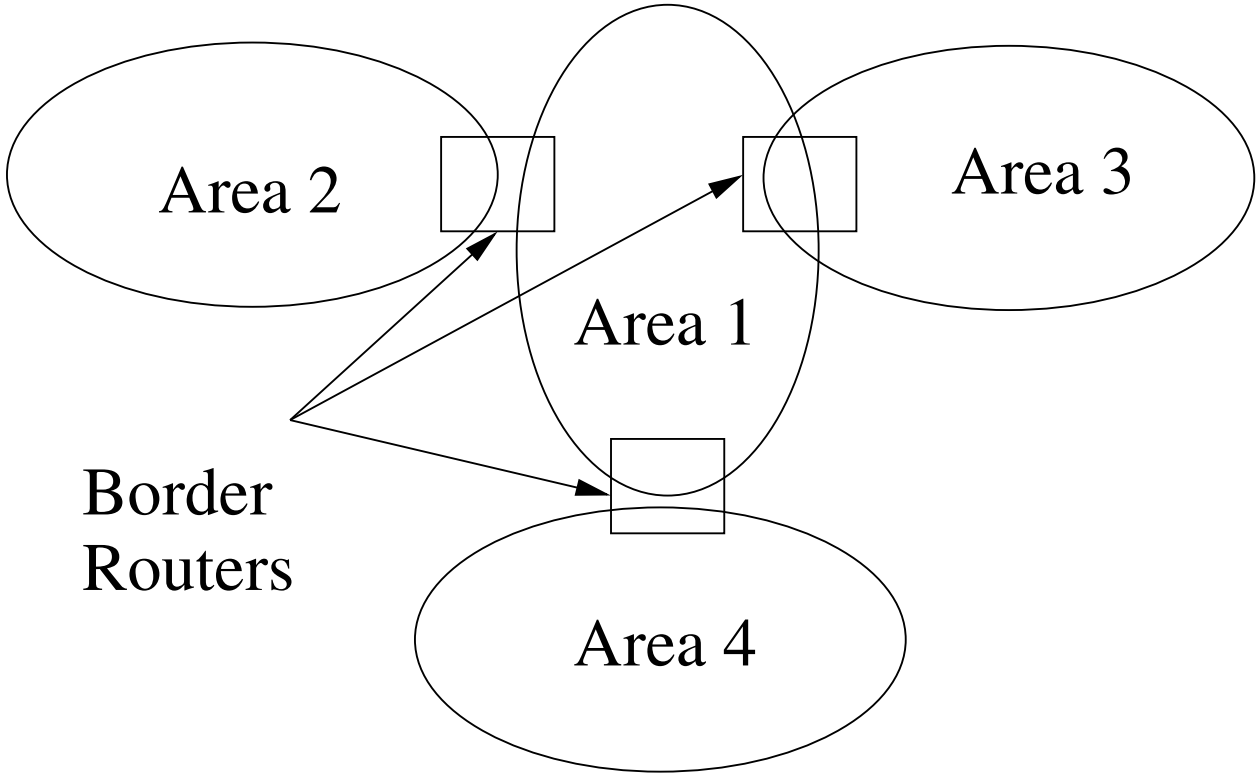
Unlike Distance-Vector or Link-State Protocols, BGP advertises complete paths as an enumerated list of ASs..



A2 can advertise reachability of A4 and A5; A3 can advertise reachability of A6 and A7. The backbone network on receiving this information, can advertise a path on how a node can be reached.

Routing Areas

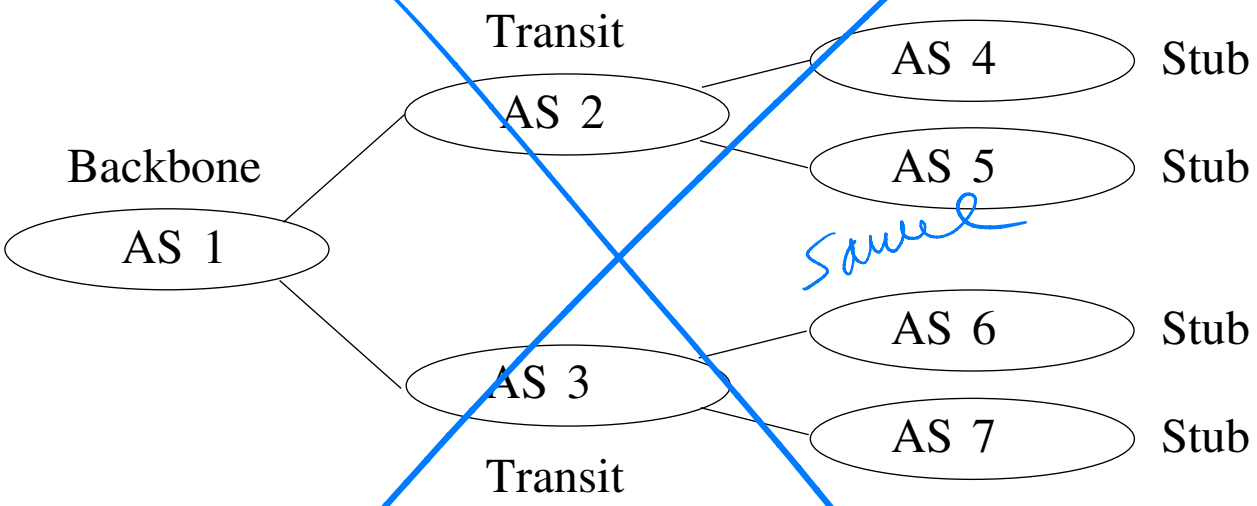
Administrators can divide a network into routing areas.



Here we try to create hierarchies at the cost of hindering optimal routing. But this is an essential tradeoff!

BGP

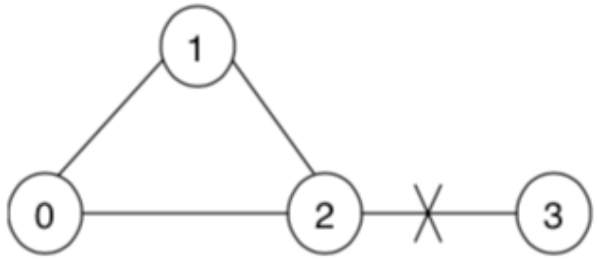
Unlike Distance-Vector or Link-State Protocols, BGP advertises complete paths as an enumerated list of ASs..



A2 can advertise reachability of A4 and A5; A3 can advertise reachability of A6 and A7. The backbone network on receiving this information, can advertise a path on how a node can be reached.

Exercises[a]

1. Consider RIP in the network below:



   (a) Compute, distances $d(1,3), d(2,3)$ without errors on edges.

   (b) What is the new value of $d(2,3)$ when the link $(2,3)$ fails?

   (c) Node 1 does not know yet of the failure of edge $(2,3)$. What information does it pass to nodes $0, 2$?

   (d) To what value does node 2 update $d(2,3)$?

   (e) What is node 1's subsequent update?

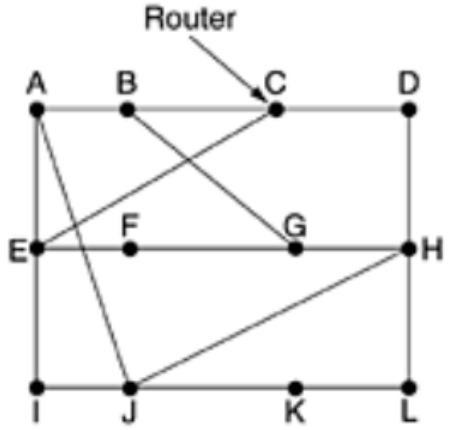
2. Consider a path consisting of n links connecting node s to node t. Let $p_i j$ be the probability that the i-th link fails. Assuming

[a] Do not submit.

that links fail independently at random what is the probability that the path succeeds to transmit a message?

3. ($\star$) Consider a network and let p_e be the probability that the link e fails. Assume that link failures are independent. Use Dijkstra;s algorithm and your solution of Exercise 2 to design an algorithm so that for any pair of nodes s, t it finds a path that maximizes the probability that a message from s to t will be transmitted successfully. **Hint:** Consider your answer in Exercise 2 and optimize the quantity arising when you take its logarithm.

4. Consider the network depicted below in which delay is being used as a metric and the router knows the delay to each of its neighbors. Once every T $msec$ each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor. Imagine that one of these tables has just come in from neighbor X, with X_i



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

being X's estimate of how long it takes to get to router i. If the router knows that the delay to X is m $msec$, it also knows that it can reach router i via X in $X_i + m$ $msec$. By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding line in its new routing table.

The updating process is illustrated in the table. The first four columns of part show the delay vectors received from the neighbors of router J. A claims to have a 12 $msec$ delay to B, a 25 $msec$ delay to C, a 40 $msec$ delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, K as $8, 10, 12, 6$ $msec$, respectively.

Appendix

Route Calculation in LSP: Dijkstra's Algorithm

- We describe Dijkstra's Algorithm as it generates shortest paths.
- A modification can be made which provides the LSPs (these are trees forming the view from a node)
  - N set of nodes in network.
  - $l(i,j)$ (non-negative) cost associated with edge $\{i,j\}$.
  - Let s be the node executing the algorithm in order to find shortest paths to all other nodes in the network.
  - M is the set of nodes incorporated so far by algorithm.
  - $C(n)$ is the cost of the path from s to node n.

Dijkstra's Algorithm

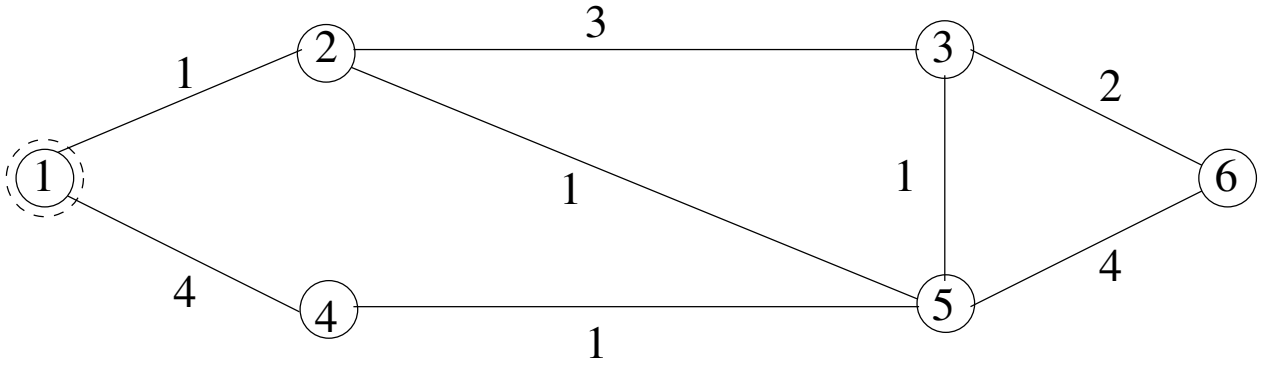
1. $M = \{s\}$
2. **for each** $n \in N \setminus \{s\}$
3. $\quad C(n) = l(s, n)$
4. **while** $N \neq M$ **do**
5. $\quad M = M \cup \{w\}$ such that $C(w)$ is min for all $w \in N \setminus M$
6. $\quad$ **for each** $n \in N \setminus M$
7. $C(n) = \min\{C(n), C(w) + l(w, n)\}$

Route Calculation in LSP: Dijkstra's Algorithm

- Also finds shortest paths from all nodes to some fixed destination (or source)
- Requires that all edge weights are nonnegative (not a restriction for most network applications)
- Shortest paths found in order of increasing path length.
- Crucial idea:
  - During the kth step the kth closest node to the destination is found by considering the distance of nodes not among the $k-1$ closest to any node among the $k-1$ closest

Example: Dijkstra's Algorithm

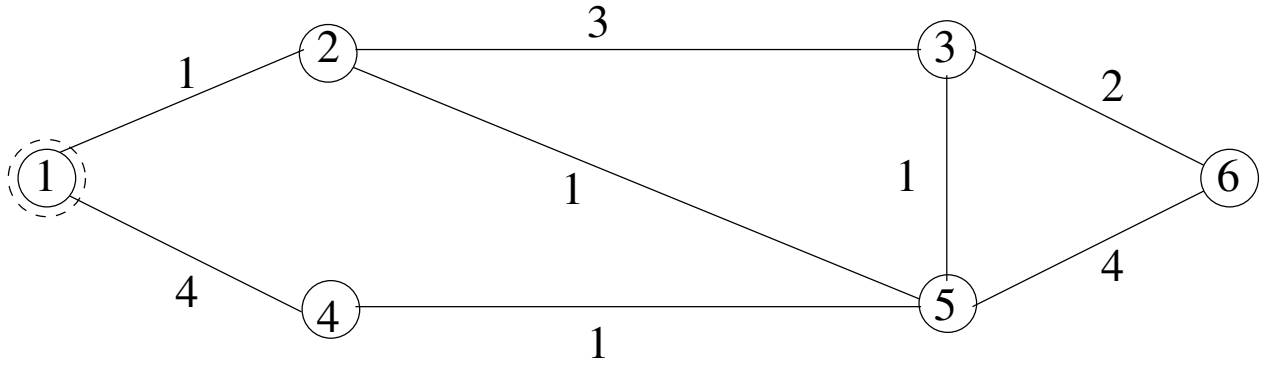
Start node is 1



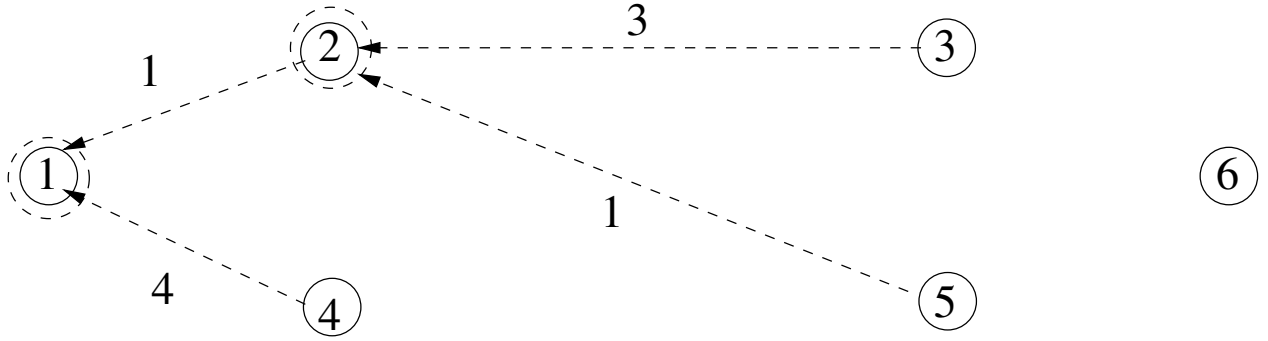
Iteration 1: Compute all costs to 1. Update min cost routes to 1.

Example: Dijkstra's Algorithm

Start node is 1

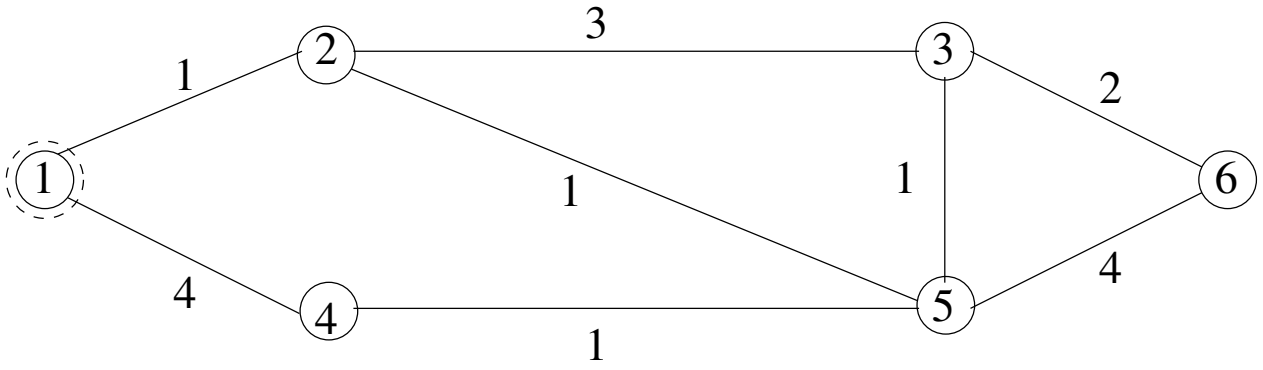


Iteration 2: Add min cost node to M (node 2). Update min cost routes to 1.

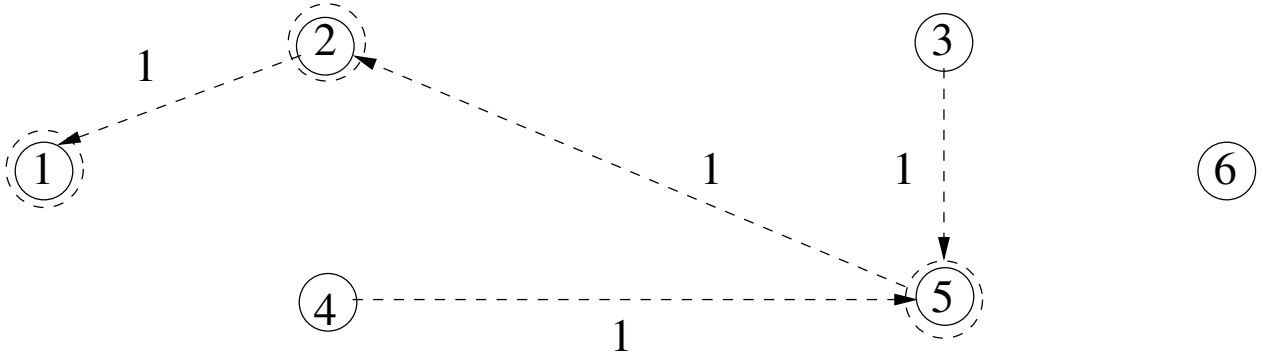


Example: Dijkstra's Algorithm

Every node executes Dijkstra's algorithm

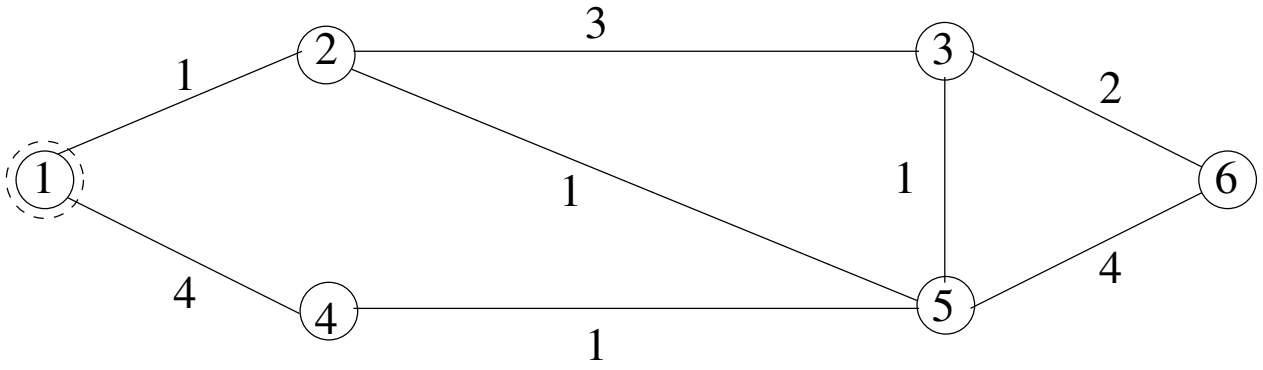


Iteration 3: Add min cost node to M (node 5). Update min cost routes to 1.

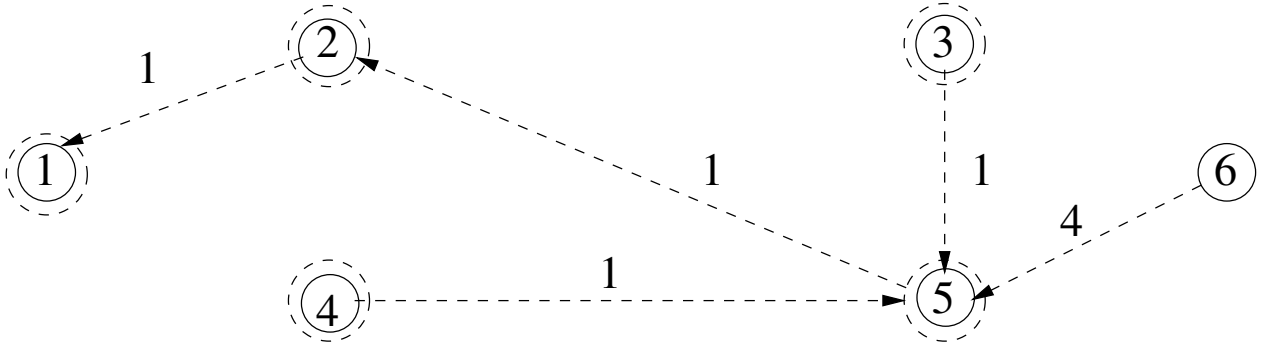


Example: Dijkstra's Algorithm

Every node executes Dijkstra's algorithm

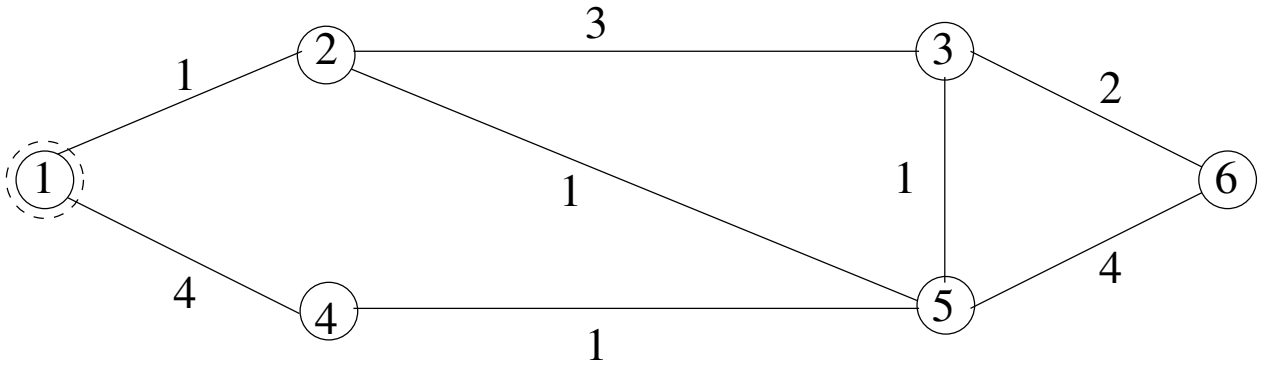


Iterations 4: Add min cost node to M (node 3). Update min cost routes to 1.

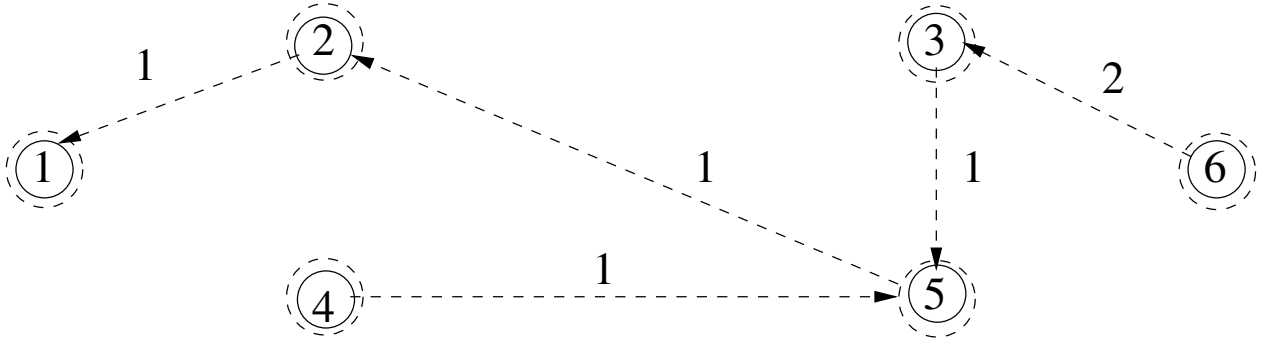


Example: Dijkstra's Algorithm

Every node executes Dijkstra's algorithm



Iteration 5: Add min cost node to M (node 6). Update min cost routes to 1.



Analysis

- Algorithm can be implemented so as to run in time $O(n^2)$, where n is the number of nodes.
- The algorithm computes weights of paths not the paths
- Can be easily modified to compute the paths:
- The last edge found in update step 3 is the first edge in a shortest path to destination and can be used to compute a shortest path tree and to compute routing tables

Spanning Trees

November 22, 2020

Spanning Trees

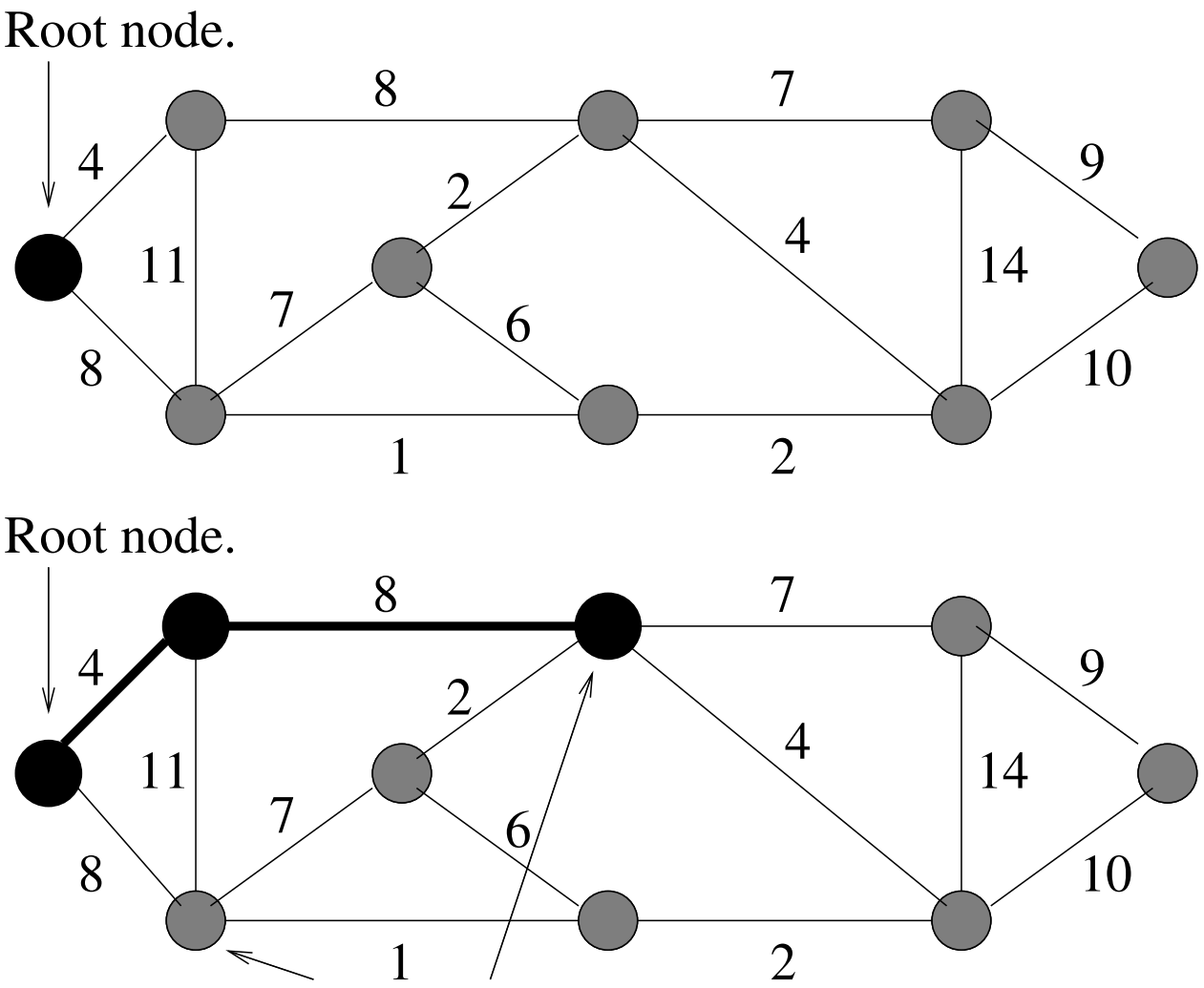
- A spanning tree of a network is a subnetwork that is a tree (i.e., contains no cycles) and includes all of the nodes of the network
- If the edges of the network are weighted (e.g., representing average delay expected on a given LAN) a minimum weight spanning tree is one with minimum sum of edge weights
- Two standard algorithms for computing MST are:
  - Prim's algorithm
  - Kruskal's algorithm

Spanning Tree Terminology

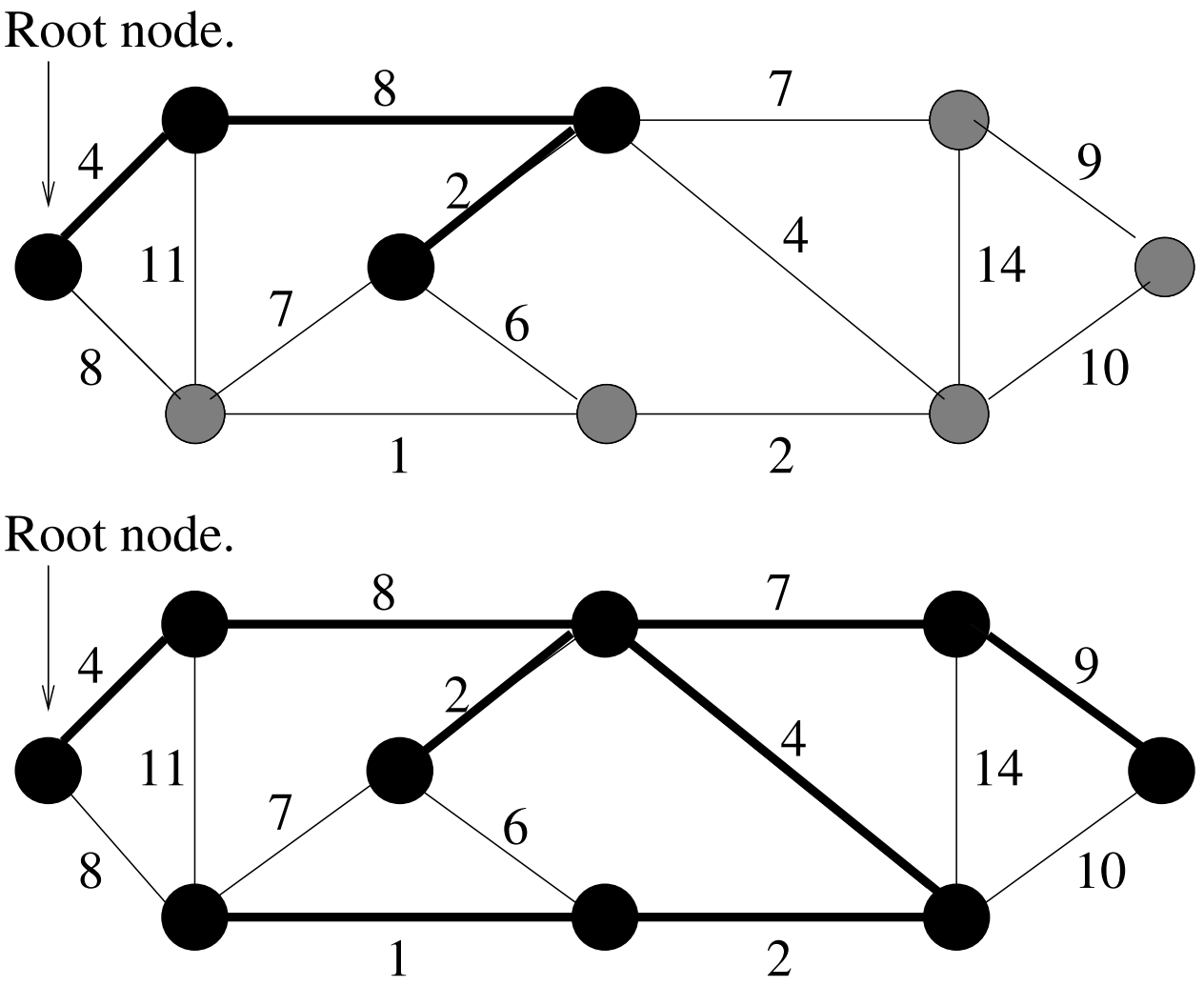
- $G = (V, E)$ is an undirected graph
- V is the set of nodes (vertices)
- E is the set of edges
- $w_{i,j}$ is the weight of the edge (i, j)
- A *spanning tree* is an acyclic subgraph containing all nodes
- Weight of a tree is the sum of its edge weights
- Minimum weight spanning tree (MST) is ST of minimum weight

Prim's Algorithm (Jarnik, 1930)

- P is set of nodes in tree and D_i is min weight edge from node i to a node in P
- Initially $P = \{1\}$, and $D_i = w_{i,1}$ if $(i, 1)$ exists, ∞ otherwise
  1. Find $i \notin P$ such that D_i is minimum
  2. $P = P \cup \{i\}$
  3. For $j \notin P$, $D_j = \min\{D_j, w_{j,i}\}$
  4. Go back to 1
- Can be implemented in $O(|E| + |V| \log |V|)$ time



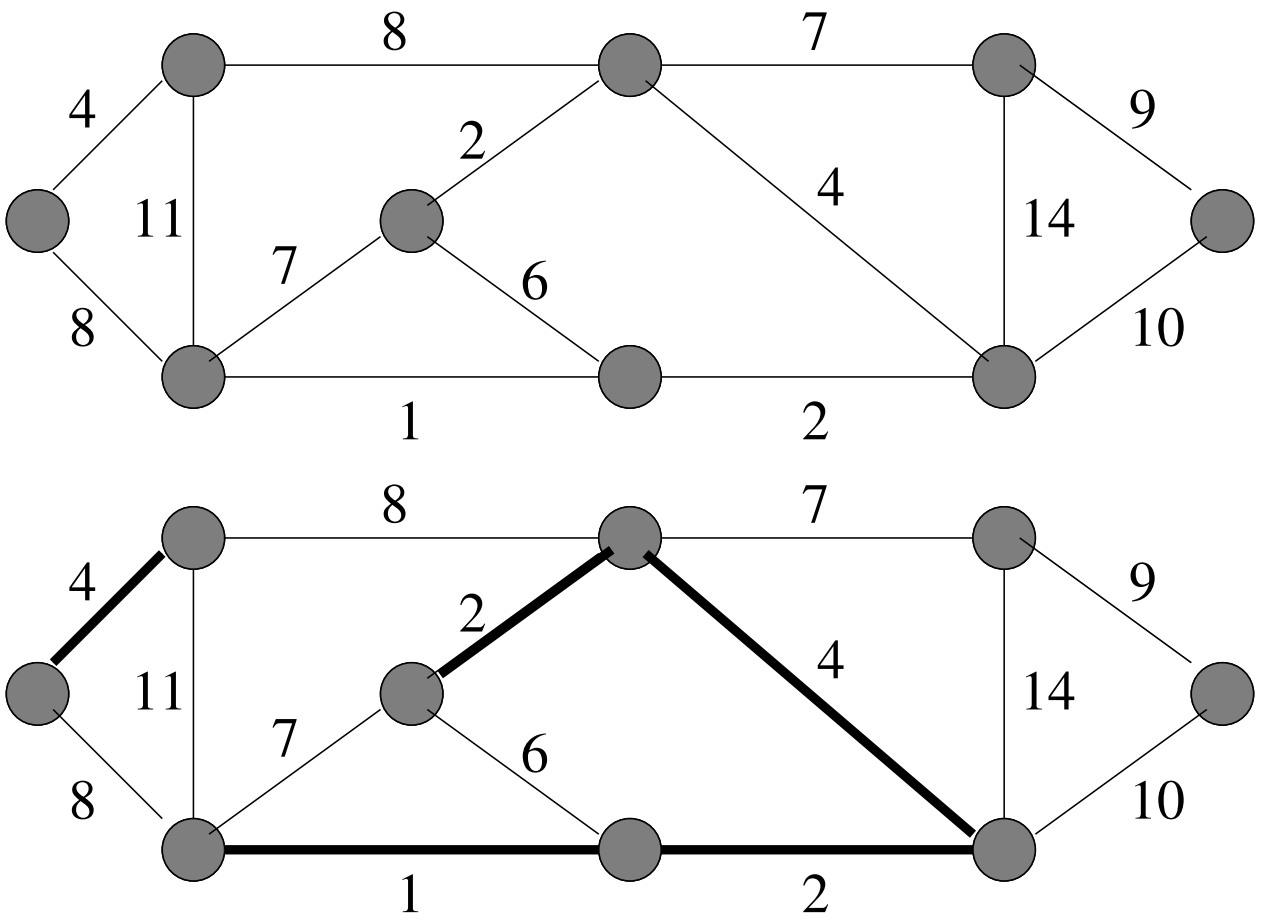
We have a choice: can add either of these two nodes.



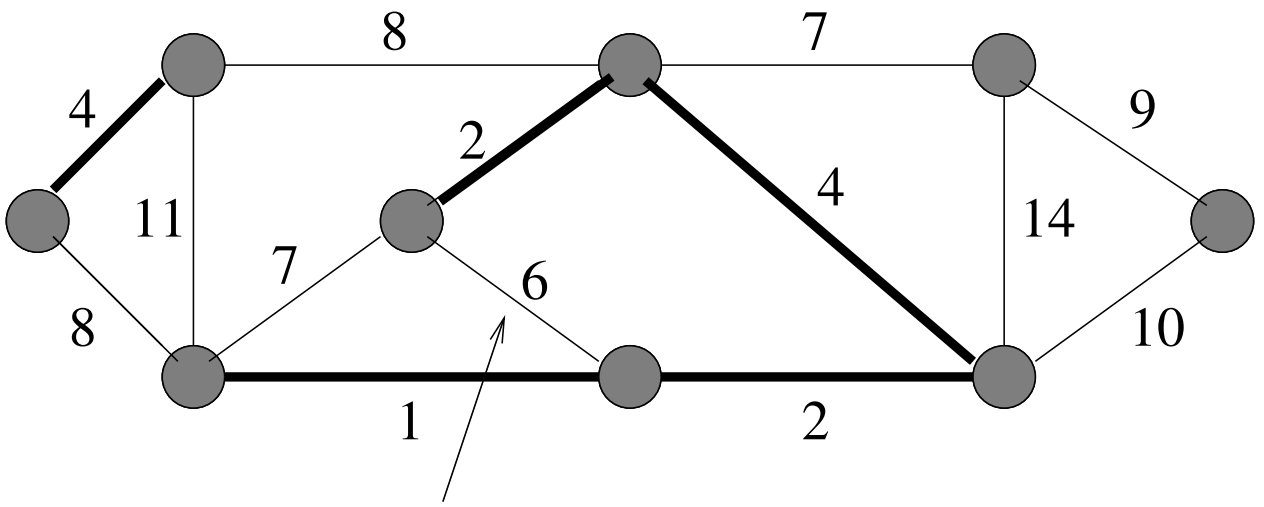
Then we add nodes adjacent to links
4. 2. 1. 7. 9 in this order

Kruskal's Algorithm (Boruvka, 1926)

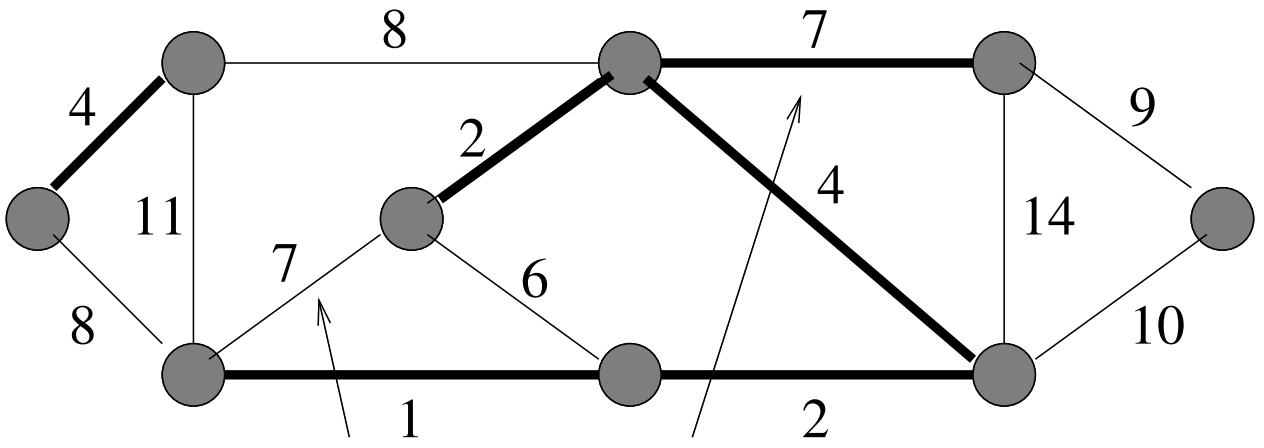
- Algorithm
  1. Sort the edges of G in increasing order
  2. Consider edges in order and add edge to tree if the result does not form a cycle
- Time complexity is $O(|E| \log |E|)$
- Can be implemented in a distributed manner and used to elect a leader



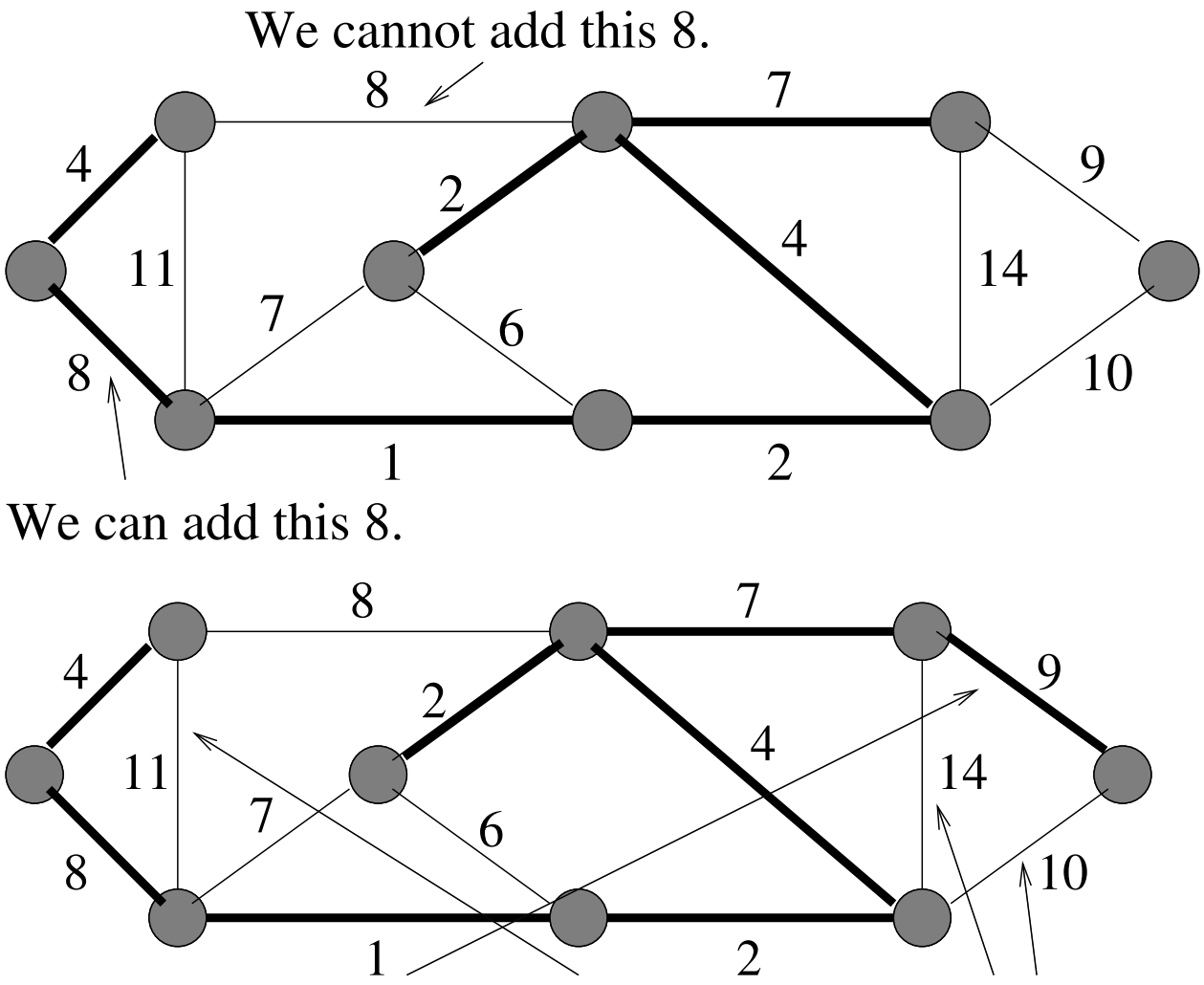
Links Added: 1. 2. 2. 4. 4



We cannot add this link: creates a cycle



We can add the 7 that does not create a cycle


We cannot add this 8.
8
7
4
9
2
4
11
14
7
6
8
10
1
2
We can add this 8.
8
7
4
9
2
4
11
14
7
6
8
10
1
2
We add the 9.
None of 10. 11. 14 can be added