Lecture 5B

Routing Algorithms, Routing Details and BGP

Routing Algorithms - Classification

- Centralized versus Distributed
  - Central entity computes paths for routers
  - Routers cooperate by means of message exchanges to perform their own routing calculations
- Static versus Adaptive (dynamic)
  - Pre-computed entries are loaded into a routing table (fixed for some time)
  - State of the network is continuously changing hence routing tables are designed to adapt and change
- Routing Tables
  - At each node a table describes the next node to send the packet to:

Destination node:	1	2	3	4	5
Next node:	2	2	0	2	5

More elaborate tables

- Can give preference order to next nodes; ie first choice, second choice
- Can be based on virtual circuit numbers

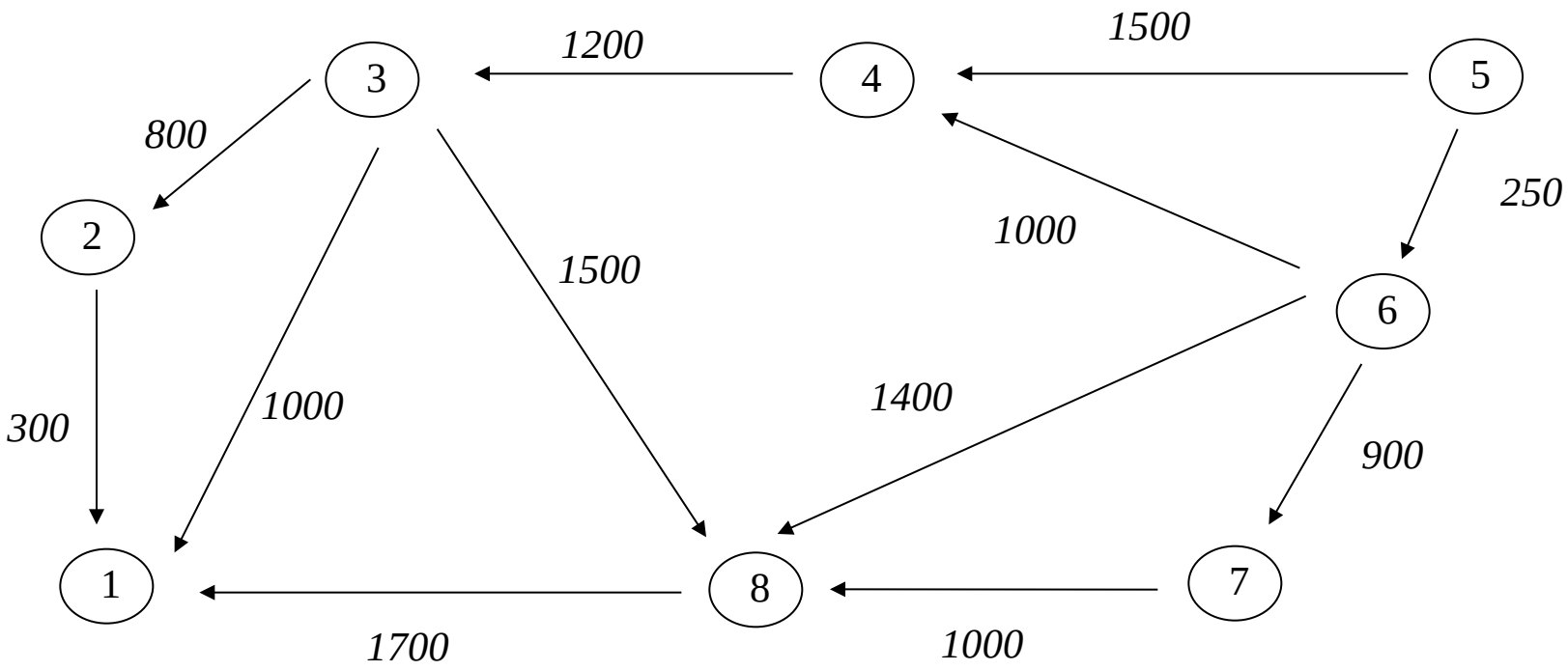
Incoming		*Outgoing*	
Node	*VCI*	*Node*	*VCI*
1	*2*	*6*	*7*
1	*3*	*4*	*4*
4	*2*	*6*	*1*
6	*7*	*1*	*2*
6	*1*	*4*	*2*
4	*4*	*1*	*3*

Dijkstra's Shortest Path Algorithm

- An example of a "greedy" algorithm
- Basic algorithm is single source shortest path – find the shortest path from a given start vertex.
- Graph G = (V, E) where V is a set of nodes, and E is a set of edges. Each edge has a weight $c_{i,j}$ which is the cost (on the edge i,j) to go from node i to node j.
- Main idea – identify closest nodes from the source in order of increasing path costs
  - First find closest node
  - Next find second closest node; this must be neighbor of source or closest node above
  - Etc.
- Requires link costs to be positive
- Algorithm works by labeling a set S of permanently labeled nodes. At each iteration, next closest node is added to the set.
  - Let s be the start node and let D_j be current min cost from s to j for all j in V.
  - S = {s}, $D_s = 0$, $D_j = c_{s.j}$ for all neighbors (of s) j not equal to s.
  - Find the next closest node, that is a new node such that it has min cost and is not permanent. Label this node permanent and add to set S. If S is full set V, stop.
  - Else, update minimum costs; need only look at neighbors of new node. Go back to previous step.

Shortest Path Routing

Dijkstra's Algorithm – an example



Find the shortest path from 5 to the other nodes.

$c_{i,j}$ *is cost to go from vertex i to vertex j*

Example

S	add V	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
{5}	6	∞	∞	∞	1500		250P	∞	∞
{5,6}	7	∞	∞	∞	1250			1150P	1650
{5,6,7}	4	∞	∞	∞	1250P				1650
(5,6,7,4}	8	∞	∞	2450					1650P
{5,6,7,4,8}	3	3350	∞	2450P					
{5,6,7,4,8,3}	2	3350	3250P						
5,6,7,4,8,3,2,		3350P							

IDEA:

Calculate new distances using the current set S. Note that the only changes are through the addition of new element to S, that is, the (new) vertex chosen to add to S. Find smallest distance in complement of S

Compute changes by: $Min\ [\ D_j\ ,\ D_{new} + c_{new,\,j}\ ]$ *for j neighbor of new vertex*

Keeping track of paths & complexity of algorithm

- As each node is labeled (or changed) keep track of predecessor node. Backtracking from a destination node defines the path.
- Algorithm as described is $O(|V|^2)$ and was proposed by Dijkstra.
  - Graph is stored as an adjacency list
  - Linear array implementation of a min priority queue. Operations are:
    - Insert: insert a node (vertex) into the queue (the linear array shown), O(1).
    - Min: determine the minimum valued vertex, O(V).
    - Extract-min: remove min value (Make a node permanent), O(V).
    - Decrease key: decrease the key value of a given node, O(1).
    - Complexity is $O(|V|^2 + |E|)$ or $O(|V|^2)$
- Binary heap implementation is $O(|E| \log|V|)$
- Fibonacci heap implementation is $O(|E| + |V| \log|V|)$
- Algorithm requires knowledge of link states or costs on the links (edges) to be distributed to all nodes that use the algorithm to compute distances. Thus it is called a *link state algorithm.*

Bellman Ford Moore Shortest Path Algorithm

Undirected graph with edge weights. Find the shortest distance (and path) to (or from) source s for all nodes in the graph.

Basic Idea: Let D_i be the distance to node i from node s.

1. Initialize: $D_s = 0$, and for v ≠ source $D_v = \infty$
2. As long as there is an edge (i, j) such that

$$D_j > D_i + c(i,j)$$

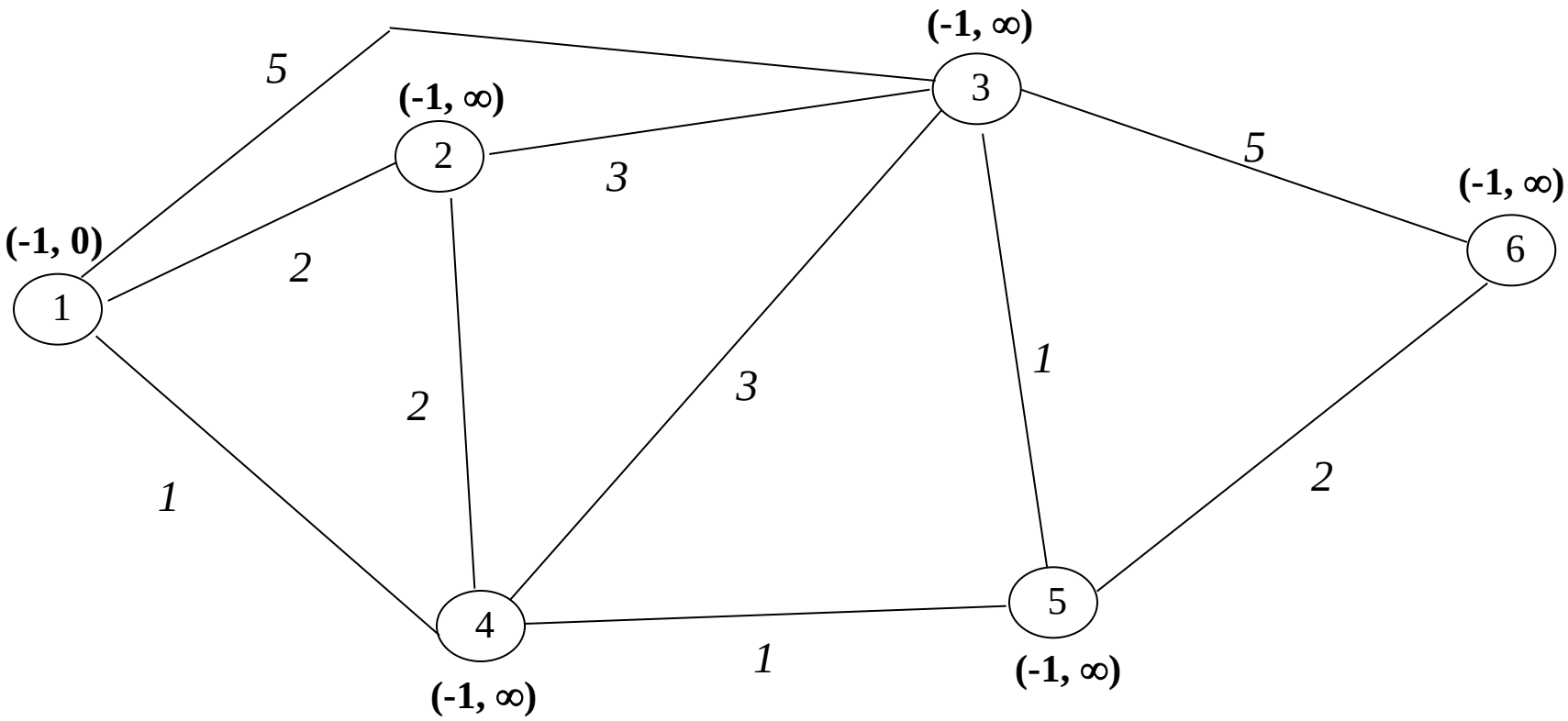
replace current D_j by $D_i + c(i,j)$

How do we search nodes to ensure that condition 2 above is met and we are able to terminate?

One possibility is breadth first search.

Label nodes by (p, d) where p is the current predecessor node and d is the current estimated distance.

A graph and associated edge costs, source node is 1 and initial labels (p,d) = (predecessor, distance)



One version of Bellman Ford Moore

- Keep a queue of nodes to examine, use breadth first search
- Start with queue = {1}
- Initial label for node 1 is (-1, 0)
- Initial label for other nodes is (-1, ∞)

- Look at neighbors of node x, , ie, nodes 2,3,4 if x = 1
- Determine if a label of the neighbor node under consideration say j, is changed by the formula

  Compare D_j and $D_x + c(x,j)$

- If node j distance D_j is changed (ie reduced) add j to the queue.
- Algorithm: While queue is non-empty, delete head node x and look at links x,j and calculate new distances to each neighbor j. If change, add j to queue (unless already in queue).

Q = <1>;

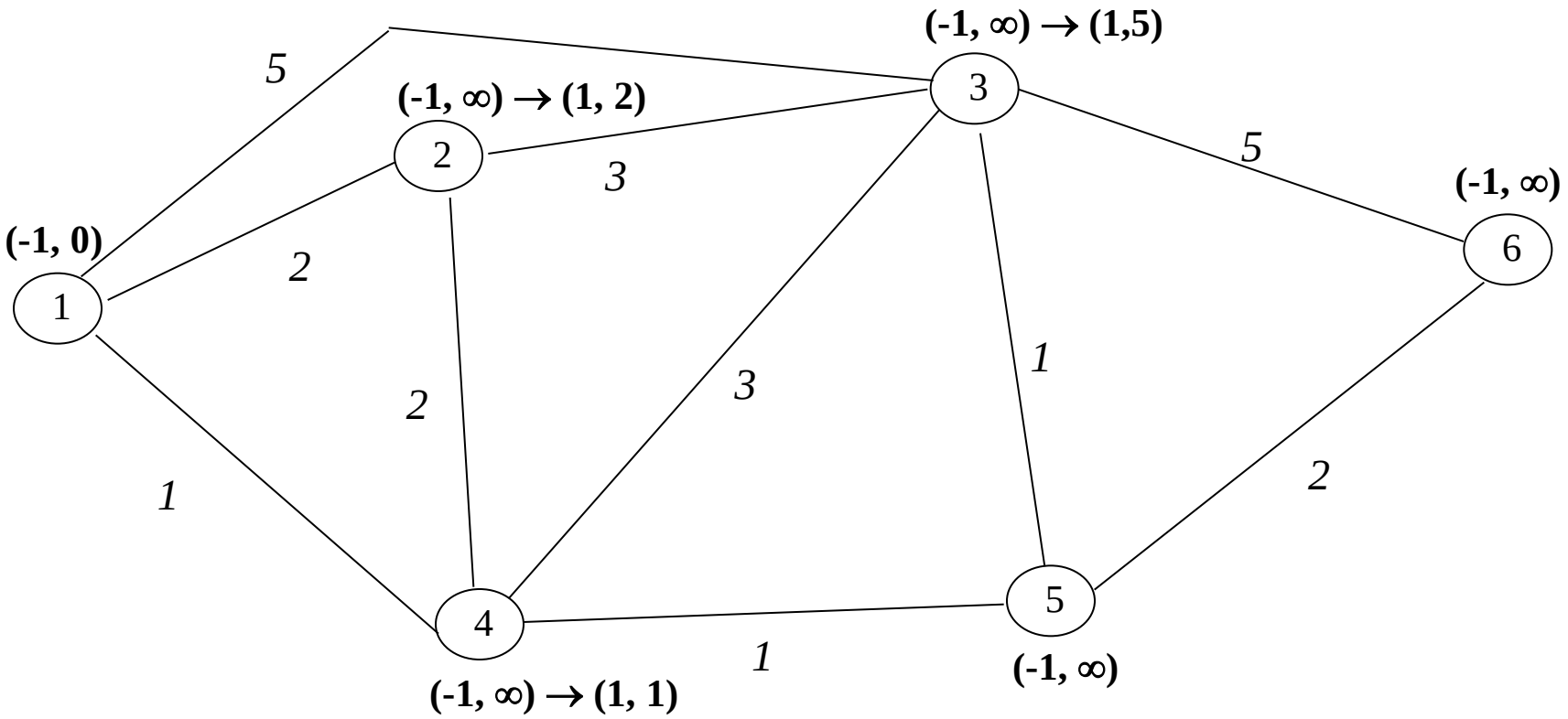
Delete 1 from head of queue, look at neighbors of 1

Compute D_2, distance of 2 becomes 2, label of node 2 becomes (p,d) = (1,2)

Compute label of 3 becomes (1,5)

Compute label of 4 becomes (1, 1)

Add nodes 2,3,4, to queue which becomes <2,3,4> where left most node 2 is at the head.



Q = <2,3,4>;

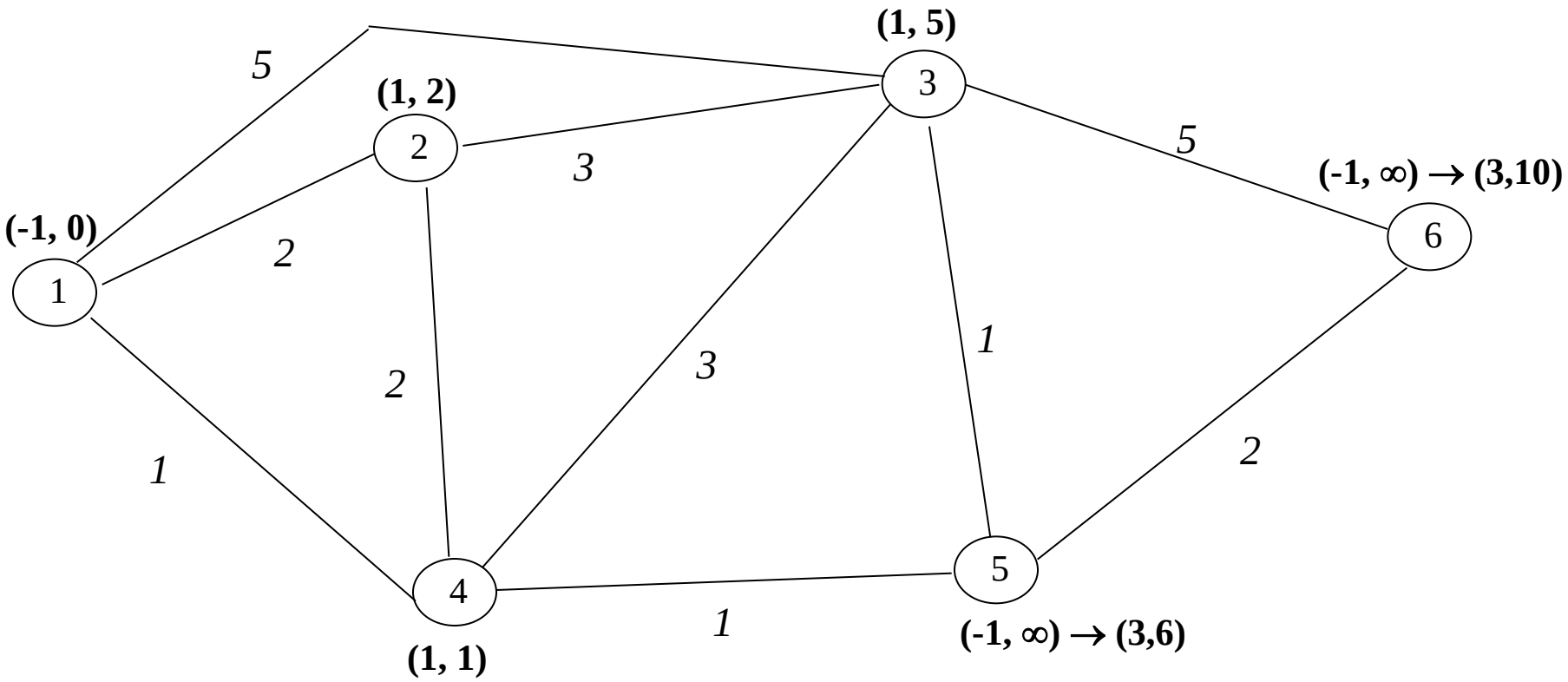
Delete 2 from head of queue, look at neighbors of 2

No change for any neighbor of 2.

Delete 3 from head of queue, look at neighbors of 3

Label of node 5 becomes (3, 6) and label of node 6 becomes (3,10)

Add nodes 5, 6 to queue which becomes <4, 5, 6>



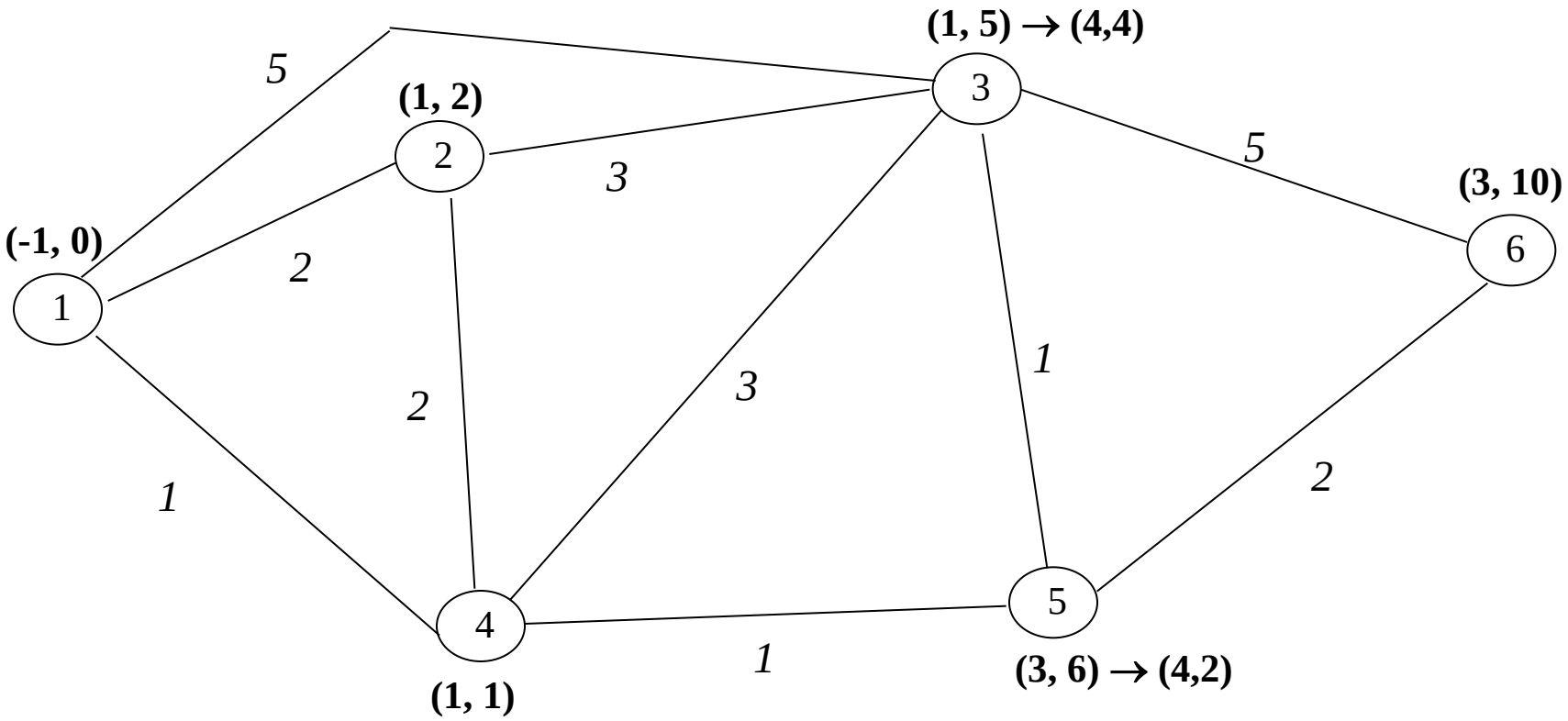
Q = <4,5,6>;

Delete 4 from head of queue, look at neighbors of 4, label of neighbor 2 does not change.

Label of node 3 becomes (4,4) Add node 3 to queue.

Label of node 5 becomes (4,2), 5 is already in queue

New queue is <5, 6, 3>



Q = <5,6,3>;

Delete 5 from head of queue, look at neighbors of 5.

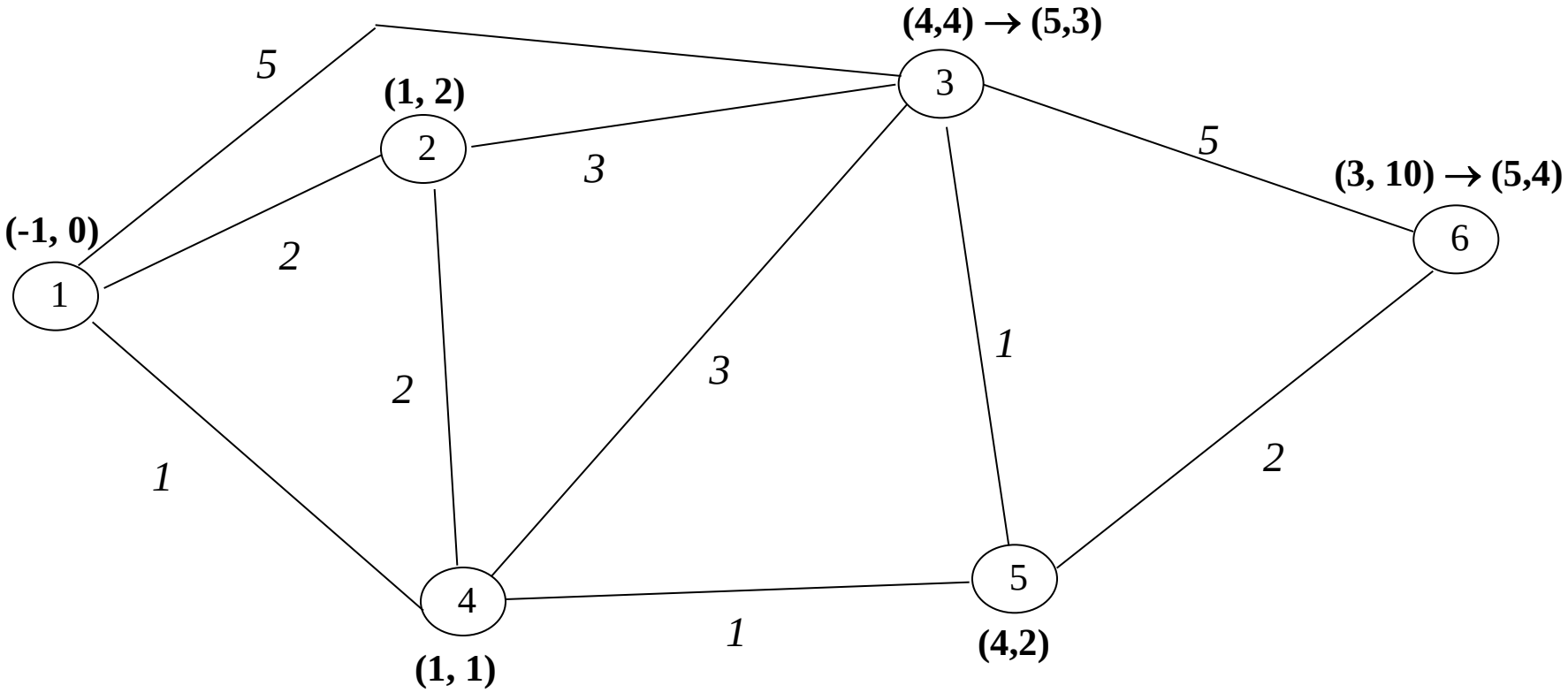
Label of node 3 becomes (5,3) Node 3 is already in queue.

Label of node 6 becomes (5,4), 6 is already in queue

New queue is <6, 3>, no change for neighbors of node 6.

New queue is <3>, no change for neighbors of node 3.

New queue is <>, algorithm terminates.



Evaluating Bellman Ford Moore

- Basic algorithm is $O(|V|^3)$. A more precise analysis shows that it is actually O(m |E|) where m is the diameter of the network and E is the number of links in the network.

- D'Esopo Pape Version – If k was never in the queue, add k to tail of the queue. If k was previously in queue, but is currently not in the queue, add k to the head of the queue.

- For large, sparse networks, this is one of the better algorithms.

Distributed Bellman Ford Moore

- Nodes exchange information about their current notion of shortest paths to all nodes with their neighbor nodes.
- The information exchanged is used to update the vector of shortest paths maintained by the nodes.
- This is done in parallel and in a distributed fashion.
- Cost vectors or distance vectors are exchanged, hence the term *distance vector algorithm.* These vectors are also called tables which can be confusing.
- Each node i has a vector of current distances to each node j (the distance vector) in the network called $DV_i(j)$ and certainly knows the cost to its own neighbors.

  Initialization for each node i,

  $D_i(i) = 0$ and $D_i(j) = c(i, j)$ for the neighbors j of i. All other values are initially set to ∞.

  Iteration:

  (1) Each node broadcasts its DV() to its neighbors.

  (2) Each node i upon receiving a $DV_k()$ from neighbor k, *updates* its DV to DV' as follows:

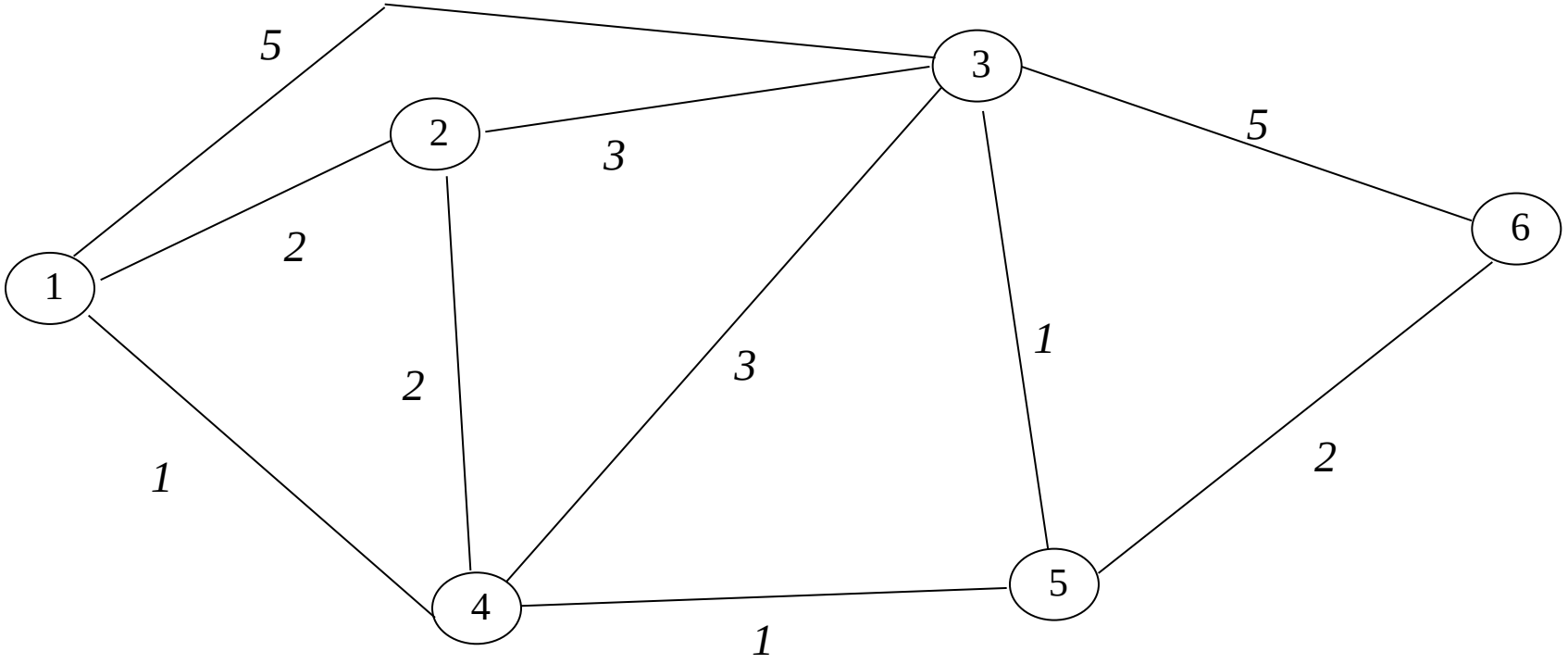
  $DV'_i(j) = \min [DV_i(j), DV_i(k) + DV_k(j)]$ for all values of j.

- Note that the distance vector maintained by a node actually also needs to have not only the *distance* but also the *forwarding neighbor* for that j. This is indicated in the examples when relevant.

Example of distributed Bellman Ford Moore

Initial vectors for each node

	1	2	3	4	5	6
DV_1	0	2	5	1	∞	∞
DV_2	2	0	3	2	∞	∞
DV_3	5	3	0	3	1	5
DV_4	1	2	3	0	1	∞
DV_5	∞	∞	1	1	0	2
DV_6	∞	∞	5	∞	2	0



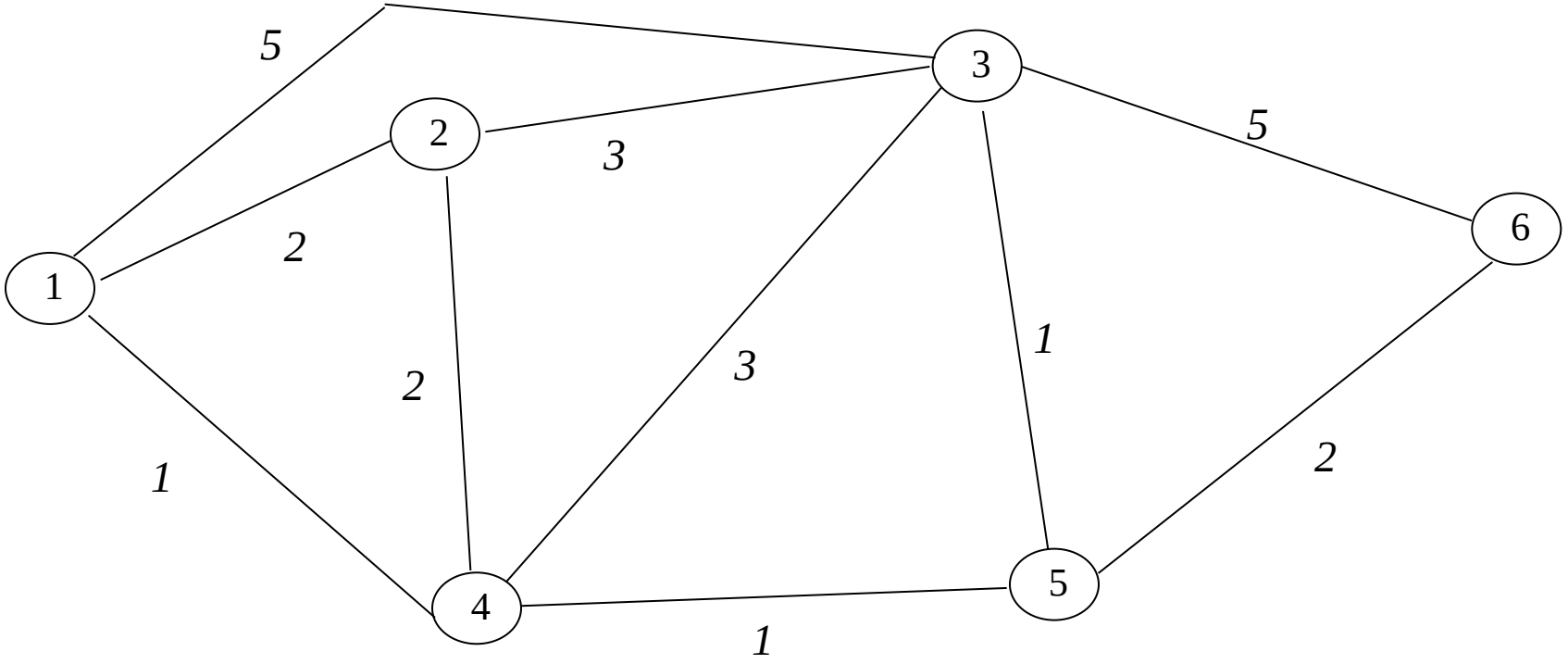
Example of distributed Bellman Ford Moore: update for Node1

Has	1	2	3	4	5	6
DV_1	0	2	5	1	∞	∞

Receives	1	2	3	4	5	6
DV_2	2	0	3	2	∞	∞
DV_3	5	3	0	3	1	5
DV_4	1	2	3	0	1	∞

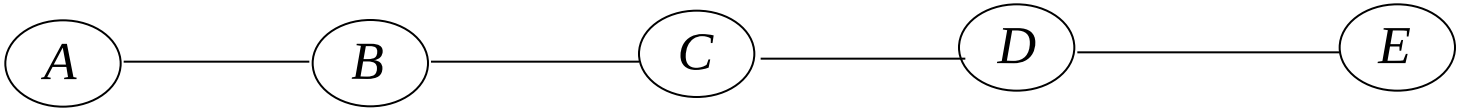
Updates from	1	2	3	4	5	6
DV_2	0	2	5	1	∞	∞
DV_3	0	2	5	1	6 via 3	10 via 3
DV_4	0	2	4 via 4	1	2 via 4	10 via 3

In above, Node 1 is updating its DV iteratively as DVs are received from neighbors 2, 3, and 4 with the last line thus being its DV' after the 1st round of exchange of distance vectors. Note that where necessary, the forwarding node is also indicated.



Simple examples

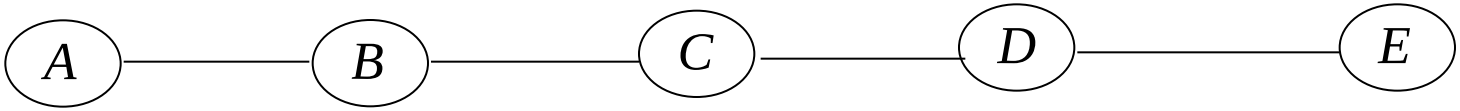
Good News Travels Fast: Determining my distance from node A



0	∞	∞	∞	∞
0	1	∞	∞	∞
0	1	2	∞	∞
0	1	2	3	∞
0	1	2	3	4

Simple examples

Bad News is a Problem: Determining my distance from node A if Node A goes down: ***Counting to Infinity Problem***



0	*1*	*2*	*3*	*4*
X	*3*	*2*	*3*	*4*
	3	*4*	*3*	*4*
	5	*4*	*5*	*4*
	5	*6*	*5*	*6*
	7	*6*	*7*	*6*

ETC

Trying to Fix the Problem

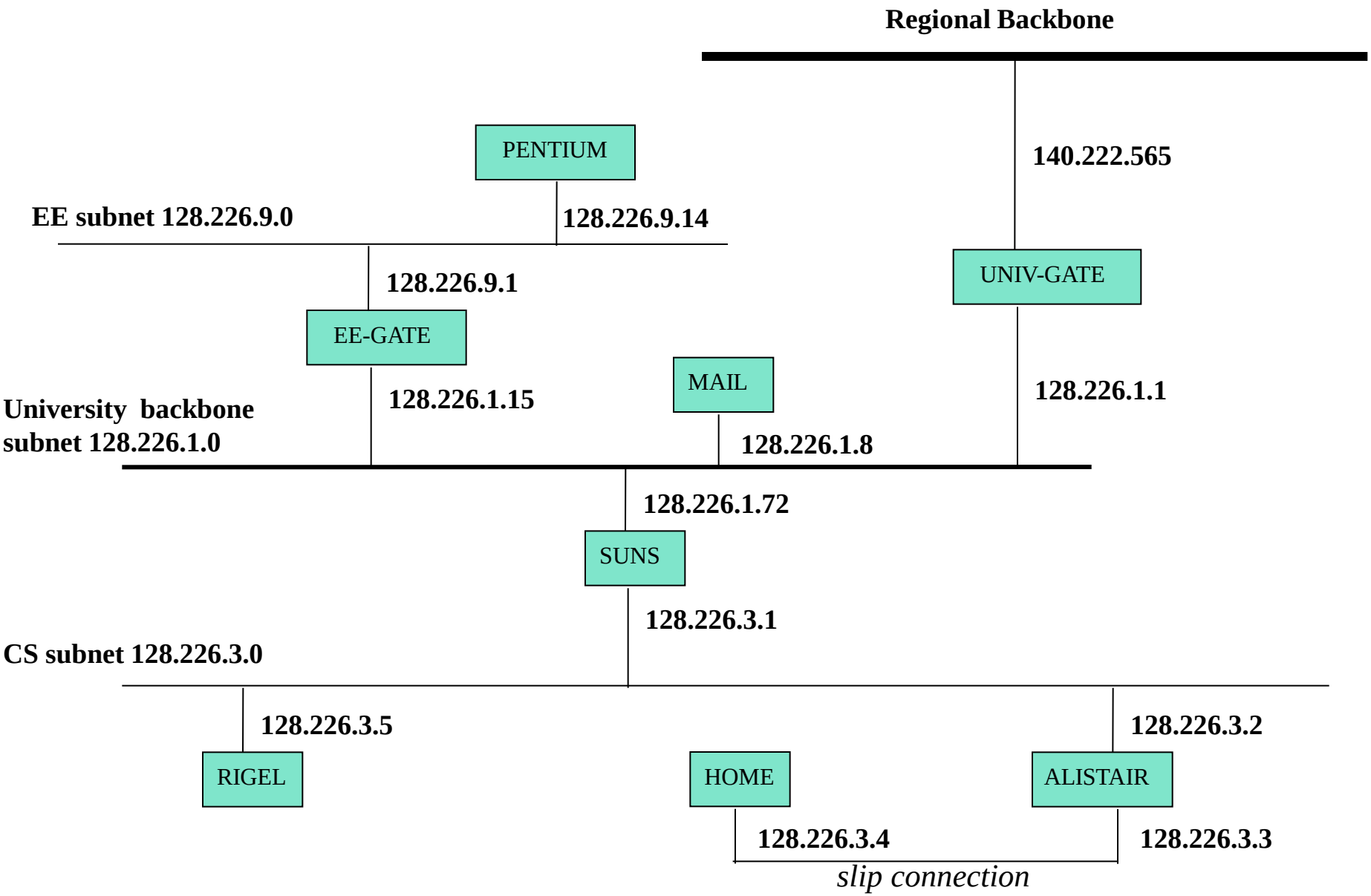
- Split Horizon: Minimum cost to a given node is not sent to neighbor if neighbor is the next node on the shortest path to the given node
  - Node X thinks best path to Z is through Y
  - Then X should not send min cost vector for Z to Y

- Split Horizon with Poison Reverse: minimum cost is sent to all nodes but distance to Z is set to infinity if neighbor is on the path to Z

Other Routing Approaches

- Recall that basic approach is Next Hop Routing
  - given knowledge of distances and optimal paths, packet is sent to the next hop for the particular destination
- Source Routing
  - Source gives complete route information in the header. This is stripped off at each node in the path.
  - Packet at source A to D: B – G – D; Packet at B: G – D; Packet at G: D; Packet at D: done.
- Random Routing, also called *deflection* routing
  - Randomly choose outgoing link
  - Assign probabilities to outgoing link based on traffic, etc.
  - Also call “hot potato” routing
- Flooding
  - Send copy of packet on each outgoing link except link it arrived on
  - Modifications to limit copies of packets seen by nodes through keeping track of packet lifetime, or recognizing packet you have seen before

Part of a University Routing Domain



Examples

1. Rigel to Pentium
   destination IP: 128.226.9.14
   source IP: 128.226.3.5

2. Outside to Home
   destination IP: 128.226.3.4
   source IP: 198.5.7.6

3. Rigel to Mail
   destination IP 128.226.1.8
   source IP: 128.226.3.5

4. Rigel to Outside
   destination IP: 198.5.7.6
   source IP: 128.226.3.5

How does a node make a forwarding decision?

- Based on:
  - IP packet header (mainly destination address)
  - Information in a routing table

- From the node's perspective there are two possibilities:
  - I am the ultimate destination of the packet. Compare the packet destination address with the IP addresses of the node. Packets destined for the node are consumed. That is, they are sent to the higher layer protocol indicated in the protocol field of the IP header.

  - I need to forward the packet to some other node. Make a forwarding decision. This decision is made for packets not destined for the node arriving on an interface, or packets generated by the node itself by an application on the node.

  - ***Note that every node, whether a host or router, operates the same way***

SUNS Routing Table

Destination / prefix length	Gateway or Next Hop	Flags	Physical Interface
127.0.0.1 / 32	127.0.0.1	H	lo0
128.226.3.4 /32	128.226.3.2	HG	le0
128.226.3.0 / 24	128.226.3.1	direct	le0
128.226.1.0 / 24	128.226.1.72	direct	le1
128.226.9.0 /24	128.226.1.15	G	le1
default	128.226.1.1	G	le1

SUNS Routing Table (another possibility)

Network Destination	Netmask	Gateway	Interface	Metric
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
128.226.3.4	255.255.255.255	128.226.3.2	128.226.3.1	10
128.226.3.0	255.225.255.0	128.226.3.1	128.226.3.1	10
128.226.1.0	255.225.255.0	128.226.1.72	128.226.1.72	10
128.226.9.0	255.225.255.0	128.226.1.15	128.226.1.72	10
0.0.0.0	0.0.0.0	128.226.1.1	128.226.1.72	10

Details of Routing Entries

- Host specific route
  - Route is a routing entry of prefix length 32 bits and provides a match for exactly one specific address.
  - This is indicated by the H flag being set
- Network-prefix route (prefix length less than 32 bits)
  - Match for IP addresses is based on the network portion of the address indicated by the prefix length
- Indirect or Direct Route
  - If G bit is set then this is an indirect route. That is, the packet has to be first sent to the next hop router. If G bit is not set then it is a direct route. The next hop router is simply the information of the IP address of the physical interface
- Default Route
  - If no other entry matches, use this.
  - Also indicated by 0.0.0.0 / 0

Routing mechanism (forwarding decision)

- Find the first match in the routing table for the IP destination address of the packet under consideration
  - A match occurs if the left most number of bits defined by the prefix-length of the destination address in the table matches the same number of bits in the IP packet. Note: if there are two or more matches then the longest prefix length match is used. First try to match a host specific route. Else try to match a network prefix route.
- If a match occurs:
  - Send the packet to the next hop router indicated by the Gateway field via the interface indicated if the G bit is set. Use the MAC address corresponding to the IP address of the router in the Gateway field.
  - Note that if the flags indicate “direct” this simply means that the corresponding “next hop” entry is just the IP address of the directly connected interface and that the packet should be sent over the interface indicated by determining the correct MAC address corresponding to the packet IP destination address in the ARP cache.

Autonomous Systems

- Autonomous System (AS)
  - Set of routers or networks administered by a single organization
  - AS may run more than 1 internal routing protocol but typically runs only one
  - Examples of AS are for example a university network, an ISP network, or a corporate network.
- Categories of AS
  - Stub AS: has a single connection to the outside world. Also called single homed AS
  - Multi-homed AS: has multiple connections to the outside world but carries no transit traffic. Only carries traffic locally generated or locally destined (local traffic).
  - Transit AS: has multiple connections and carries local and transit traffic.
- AS Identification
  - Globally unique 16 bit number.
  - Used for exterior routing
  - Stub AS does not need an identification number

Internet Routing Protocols

- Interior Gateway Protocols
  - Used in in AS
  - Two main IGPs are used
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
- Exterior Gateway Protocols
  - Used by special routers that communicate among different AS
  - Use AS identification numbers to construct correct paths to the destination AS
  - One main EGP is used
  - BGP-4: Border Gateway Protocol Version 4

Routing Information Protocol (RIP)

- RIP is based on the distributed Bellman-Ford-Moore algorithm.
  - The BSD program code is called *routed* for routing daemon
  - RIP runs above UDP on well-known port 520
  - Metric for shortest path computation is typically number of hops with infinity represented by 16.
  - Uses split horizon with poisoned reverse
- RIP message exchanges
  - RIP messages are exchanged with node neighbors every 30 seconds and an update is expected in 180 seconds
  - If message not received in expected time, node assumes that the neighbor link has failed and sets the distance to infinity
  - RIP messages basically exchange a set of IP addresses and their hop distances (called RIP entries). The IP address is either a network address or a host address
- RIP protocol
  - Basically consists of requests and responses
- RIP-2 is somewhat more sophisticated and allows more information to be exchanged

OSPF

- OSPF is based on the single source shortest path computation based on Dijkstra's algorithm
  - OSPF runs directly over IP as protocol type 89
  - Multiple routes to a given destination can be calculated
  - Subnet mask included in routing messages
  - Link costs can be more general
  - Two-layer hierarchy by subdividing AS in areas with a 32 bit area id
- OSPF operation
  - Hello messages: used to discover neighbors and elect designated routers on multi-access networks. These are multicast / broadcast on the network via each interface about every 10 seconds.
  - Link state advertisements (LSAs) are sent by routers to all others in the appropriate router connectivity set. These advertisements give the status and cost of the neighbor links of each router. Thus, when routers get these from all the routers, the "network graph" can be built to run the shortest path algorithm

Border Gateway Protocol

- Operates over TCP on port 179
- BGP supports policy-based routing through configuration files. Policies can be based on economic and security considerations
- BGP routers exchange AS routing sequence numbers to networks
  - Example: to get to network 128.226.3.0/24 and 128.226.9.0/24, the AS sequence is A5, A3, A18, and A4.
  - Thus, BGP specifies a destination to a network as an AS path. (also specifies the next hop router that it uses)
- Messages exchanged
  - **Open**: used to establish the connection between BGP peers once the TCP connection is established and exchange basic id information
  - **Update**: used to construct a graph of AS connectivity. Messages contain information about unfeasible routes, path attributes, and Network Layer Reachability Information (NLRI). This is done somewhat analogously to exchanging distance vector information except that the full path is sent, and not just the distance information. This makes the algorithm converge and work better.
  - **Keepalive**: used to insure that the BGP peers of a BGP router are still up
  - **Notification**: Used to close BGP connection due to some error.

Multicast and Anycast

- Multicast
  - Multicast was defined as part of the IP address space
  - Allows a packet to be sent to a specific group of receivers, those who want to join the multicast group
  - Difficulties: How do you set up the multicast group? How do you route to the multicast group. Can the group change dynamically?
  - Multicasting is usually implemented as an *overlay* network at the application level
  - Routers normally do not support multicasting
- Anycast
  - A packet is delivered to the nearest member of a group
  - It is used when a distributed set of servers provide the same service such as some type of information delivery
  - It can and is implemented using the standard routing algorithms
  - Multiple servers are given the same IP address and existing routing schemes will route to the nearest server to the requesting client