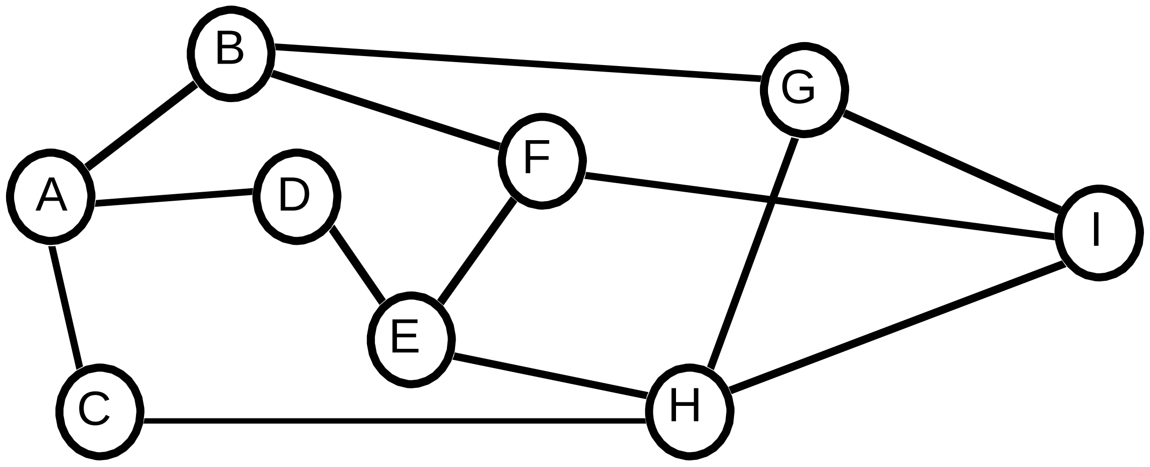
COMPSCI 314 S1 C

IP Routing

IP routing [Section 6.5]

- IP datagrams outside the local network (as determined by the host’s netmask) are sent to the local gateway router
- The gateway is another essential part of a host’s IP configuration and has an address within the host’s netmask
- The gateway routes the datagram by network. Note that IP contains no rule about how that route is obtained, or what ‘cost’ means, or if indeed it exists at all
- Same applies to all gateways/routers between there and the destination
- IP routing is ‘best effort,’ i.e., datagrams may get to the destination or they may not
- No notification if delivery fails – any recovery from lost data is left to the end points of the communication
- ICMP (IP Control Message Protocol) can return information, e.g., ‘no route to host’

Wide Area Networks — Routing Tables



A Wide Area Network (WAN) is usually an arbitrary mesh of –

- The *stations*, or *nodes*, usually have several *connections* or *ports*, with each port connecting to exactly one other station (emphasising hardware) or node (emphasising graph theory aspects)
- The stations are connected by point-to-point *links*

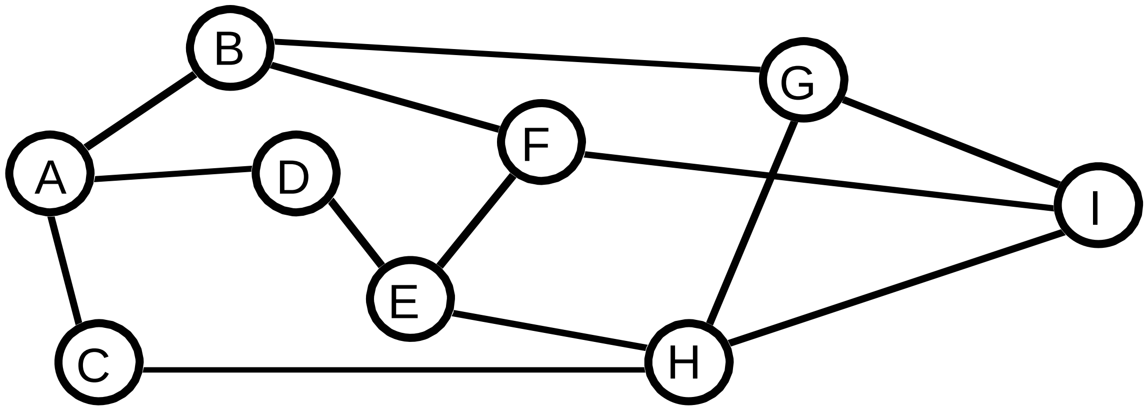
Wide Area Networks — Routing Tables

- A WAN node may be a complex of LANs (a university campus, etc.)
- A message arriving on one *input* port must usually be *forwarded*, *directed* or *routed* to some other *output* port on the same switch or router

Two types of routing —

1. Datagram routing has each packet independent and with the full destination address
   Question —
   "Which is the best output port to use for this packet?"
2. Virtual Circuit routing [1.2.4] assigns a temporary identifier (a '*virtual circuit*') to each end-to-end circuit over each point to point link
   Question —
   "If I receive VC = x over port p, what port and VC number should be assigned to forward it?"

Hop Counts



1. With datagram routing each station must know the cost of sending to each other station. The simplest cost is just the number of hops …

	A	B	C	D	E	F	G	H	I
A		1	1	1	2	2	2	2	3
B	1		2	2	2	1	1	2	2
C	1	2		2	3	3	2	1	2
D	1	2	2		1	2	3	2	3
E	2	2	3	1		1	2	1	2
F	2	1	3	2	1		2	2	1
G	2	1	2	3	2	2		1	1
H	2	2	1	2	1	2	1		1
I	3	2	2	3	2	1	1	1	

6.5.4 Dijkstra’s Algorithm

Dijkstra’s algorithm finds the shortest path from a *source* node to all other nodes. It partitions the nodes into two sets, those already *assigned* and those still *unassigned*. It is also called the *shortest path first* algorithm and is used for local IP routing.

It starts by determining the costs (or lengths) of all links between the nodes. Then, starting with only the source node in the assigned set, and all links inactive …

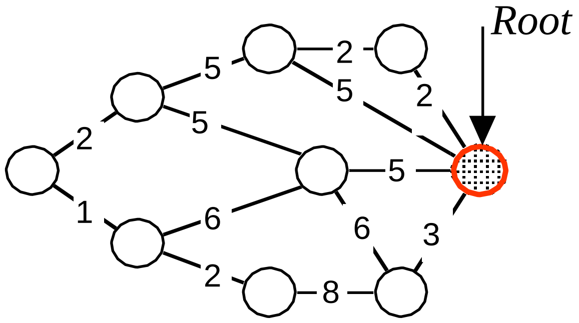
Dijkstra’s Algorithm (2)

1. Find the costs to the root node of all links which connect an unassigned node into the network.
2. Take the shortest or cheapest link, activate the link and add its node to the assigned set. (More than one link and node may be added if the link costs are equal.)
3. Repeat 1 and 2 until all the nodes have been added.

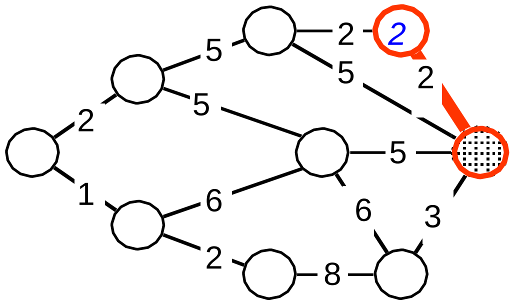
The diagram (next slide) shows the network with its link costs and shows how the paths develop as the algorithm proceeds.

An implementation of the algorithm will need tables of the link costs between nodes (some costs infinite) and continual searching of the costs between assigned and unassigned nodes; the graphical presentation is more obvious.

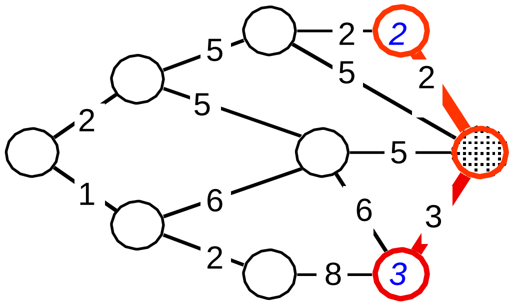
Dijkstra example: progressive evaluation



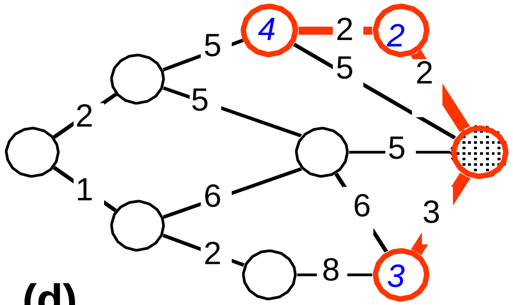
(a)



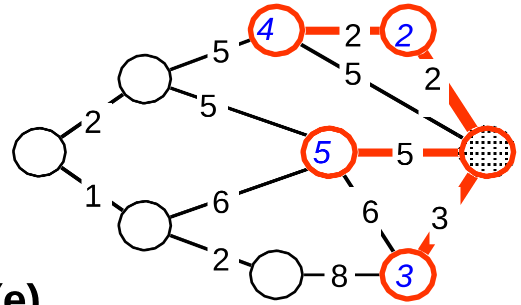
(b)



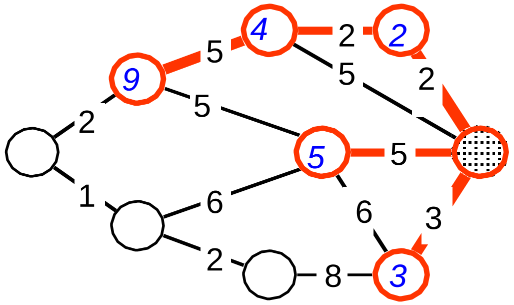
(c)



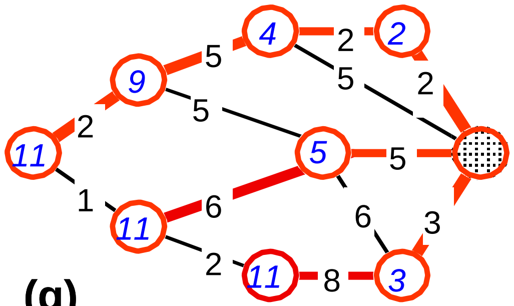
(d)



(e)



(f)

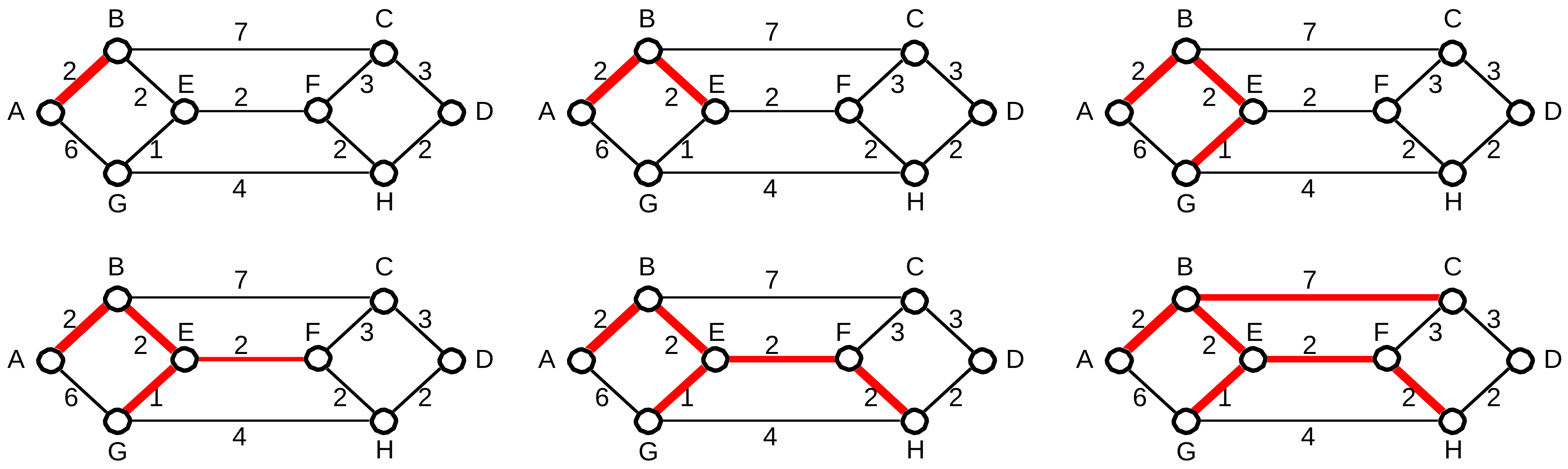


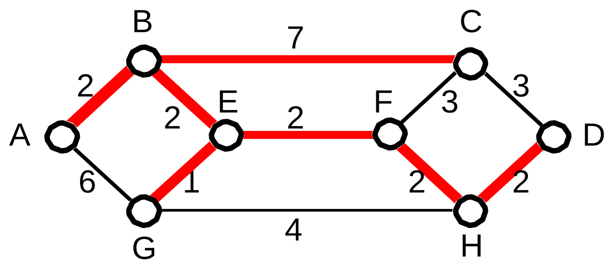
(g)

Step (g) brings in three nodes, all with a root cost of 11

Dijkstra – second example

From 2002 exam; find route from A to D, with cost





Comment: G and C are connected by ‘off route’ connections, or ‘stubs.’

Note that costs *always* increase as nodes are connected

6.5.3 Bellman-Ford Algorithm

The Bellman-Ford algorithm, also called the *distance-vector algorithm*, is a distributed algorithm which finds the shortest path to a destination from all other nodes. All costs are initially set to infinity.

- Each node keeps what it knows to be the shortest path from itself to the destination and informs its neighbours of this cost by sending the distance vector of {*dest*, *cost*} pairs

- A node i receives a distance vector from node j over a link with cost $d(i,j)$ and includes this in a table of all known costs to all destinations. If $L(i, k)$ is the cost from node i to the destination k, the cost from the current node i to the destination is the minimum over j of

$$L(i, k) = \min[L(i, k),\ d(i, j)+L(j, k)]$$

- If the new estimated cost is less than the current one, it is placed in the local table for transmission to neighbouring nodes

Bellman-Ford in action
(aka “Tux on the big OE”)

Hong Kong onward destinations:

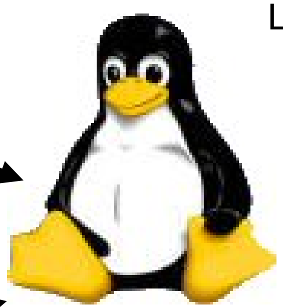
Destination	*Fare*	*via*
London	**$900**	**Singapore**
Singapore	$200	-
Beijing	$700	-
Tokyo	$400	-
…	…	…

Los Angeles onward destinations:

Destination	*Fare*	*via*
London	$400	-
Frankfurt	$500	-
Delhi	$1200	Frankfurt
Tokyo	$1000	-
…	…	…

Hong Kong: $600

Los Angeles: $900



Tux in Auckland

Sydney: $150

Buenos Aires: $700

Sydney onward destinations:

Destination	*Fare*	*via*
London	$1300	Singapore
Singapore	$700	-
Hong Kong	$700	-
Madrid	$1200	Dubai
…	…	…

Melbourne: $180

Melbourne onward destinations:

Destination	*Fare*	*via*
London	$1350	Singapore
Frankfurt	$1300	Singapore
Delhi	$1000	Perth
Hong Kong	**$400**	-
…	…	…

Tux’s routing table

Destination	*Fare*	*via*
London	$1300	Los Angeles
Singapore	$800	Hong Kong
Beijing	$1300	Hong Kong
Madrid	$1200	Buenos Aires
..	…	…

Buenos Aires onward destinations:

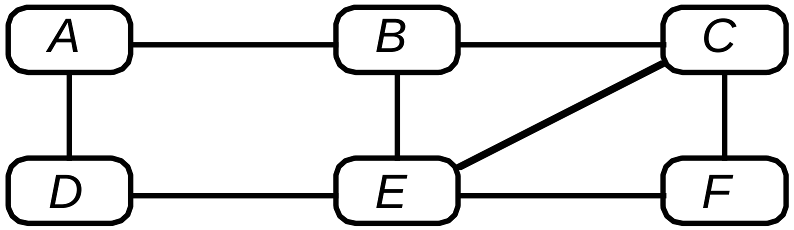
Destination	*Fare*	*via*
Helsinki	$1500	Frankfurt
Beijing	$1000	London
Madrid	$500	-
London	$800	-
…	…	…

N.B.: Note that the Hong Kong and Melbourne tables may not be in synch!

Bellman-Ford worked example

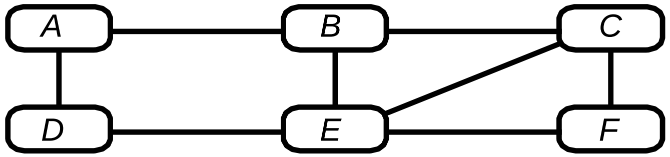
- In this network all costs are 1, making it a hop-count metric
- Consequently there is little change of route costs as better routes are found; the closest routes are the best ones
- If we had a few expensive links, we would find that some initial routes would be replaced by cheaper ones with more hops

All path costs are 1 (i.e. use hop-count metric)



Bellman-Ford worked example (2)

All path costs are 1 (i.e. use hop-count metric)



Iteration 1 Destination

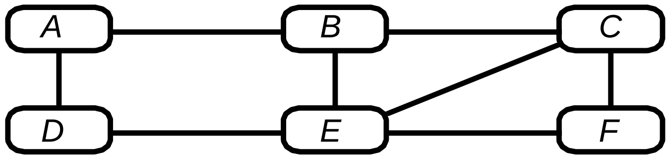
Source		A	B	C	D	E	F
	A	—	[B,1]	?	[D,1]	?	?
	B	[A,1]	—	[C,1]	?	[E,1]	?
	C	?	[B,1]	—	?	[E,1]	[F,1]
Source	D	[A,1]	?	?	—	[E,1]	?
	E	?	[B,1]	[C,1]	[D,1]	—	[F,1]
	F	?	?	[C,1]	?	[E,1]	—

Iteration 2 Destination

Source		A	B	C	D	E	F
	A	—	[B,1]	[B,2]	[D,1]	[B,2]	?
	B	[A,1]	—	[C,1]	[A,2]	[E,1]	[C,2]
	C	[B,2]	[B,1]	—	[E,2]	[E,1]	[F,1]
Source	D	[A,1]	[A,2]	[E,2]	—	[E,1]	[E,2]
	E	[B,2]	[B,1]	[C,1]	[D,1]	—	[F,1]
	F	?	[C,2]	[C,1]	[E,2]	[E,1]	—

Bellman-Ford worked example (3)

All path costs are 1 (i.e. use hop-count metric)



Iteration 3 Destination

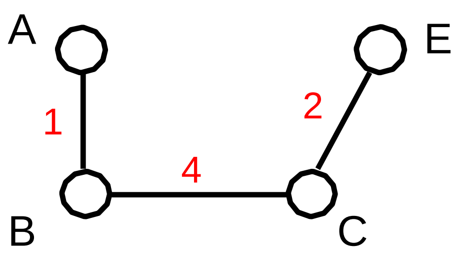
Source	A	B	C	D	E	F
A	—	[B,1]	[B,2]	[D,1]	[B,2]	[B,3]
B	[A,1]	—	[C,1]	[A,2]	[E,1]	[C,2]
C	[B,2]	[B,1]	—	[E,2]	[E,1]	[F,1]
D	[A,1]	[A,2]	[E,2]	—	[E,1]	[E,2]
E	[B,2]	[B,1]	[C,1]	[D,1]	—	[F,1]
F	[C,3]	[C,2]	[C,1]	[E,2]	[E,1]	—

Iteration 4 Destination
no change

A	B	C	D	E	F	
A	—	[B,1]	[B,2]	[D,1]	[B,2]	[B,3]
B	[A,1]	—	[C,1]	[A,2]	[E,1]	[C,2]
C	[B,2]	[B,1]	—	[E,2]	[E,1]	[F,1]
D	[A,1]	[A,2]	[E,2]	—	[E,1]	[E,2]
E	[B,2]	[B,1]	[C,1]	[D,1]	—	[F,1]
F	[C,3]	[C,2]	[C,1]	[E,2]	[E,1]	—

Source

Count-to-Infinity problem



Costs – C→E = 2
B→E = 6
A→E = 7

Consider the partial net above —

Then assume that link C → E fails (cost = ∞)

- C notes failure and sets C → E = ∞
- C passes this cost to B, which sets B → C → E = ∞
- but B hears from A of a route to E with cost = 7 and updates its best route to E to be 7+1 = 8
- now A hears of a route to E, via B, cost = 8 and sets its cost A→E = 9
- B hears from A and updates its cost B→E = 10
- loop continues as A→E and B→E both count upwards indefinitely
- Solve by setting an upper limit to link cost, at which cost = ∞

Problems of Bellman-Ford routing

- Count to infinity
- In a distributed algorithm, costs are updated as the routes are evaluated
- Routes may change during the calculation
- A low-cost route may suddenly become a preferred path and be overwhelmed as all traffic is directed to it; suddenly it is a very high cost route
- Traffic may oscillate between routes
- The routing process may become completely unstable
- In any case “bad news travels slowly” because congestion information travels out by only one hop per routing iteration. (If we use ‘choke’ packets, etc., the congestion information could travel much faster)

Link State Routing Algorithm

- When a node is initialised, it determines the link costs for all of its *interfaces* (connected links)
- Each node sends all of its costs to its neighbours, including all other costs which it knows
- Whenever any link costs change, or links become (unusable), the nodes connected to it advertise its new costs. We say that nodes use *reliable flooding* to propagate their link state
- Eventually each node knows all costs in the network and can execute a local algorithm, such as Dijkstra's, to construct its own routing table

Hierarchical Routing

- Simple routing algorithms get overwhelmed by large networks
- Split networks into groups of nodes called *domains*
- Routing within domains is done per-node, using a standard protocol
- Each domain has one or more *border routers*, to connect to routers in other domains
- The border routers are effectively a network of routers
- There may be several levels of the hierarchy

Tunneling

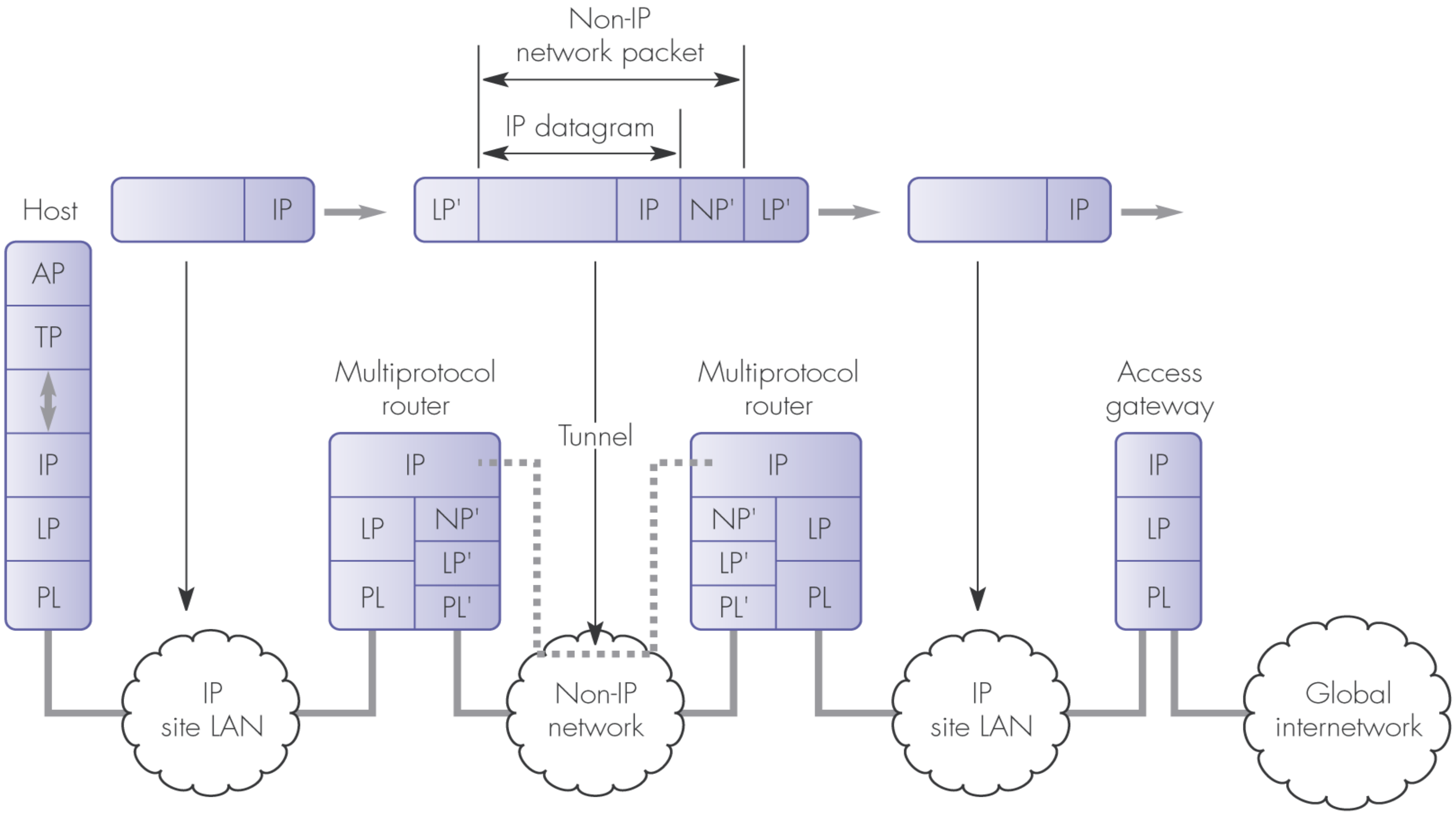


Figure 6.16 Tunneling example

IP Routing Protocols

- Static Routing
- Distance Vector (Bellman Ford):
  - RIP and RIP2
- Link State (Dijkstra):
  - OSPF, IS-IS
- Hierarchical:
  - BGP

Routing Information Protocol: RIP

- Popular IGP (Interior Gateway Protocol) for IP networks. Based on Bellman-Ford algorithm
- Bundled with BSD Unix as a program called **routed** – pronounced ‘route-d’ (‘d’ as in ‘daemon’)
- Uses routing information broadcasts rather than peer-to-peer updates (i.e., tries to take advantage of shared media such as Ethernet)
- *Broadcasts take place at 30-second intervals*
- Not all nodes broadcast routes, only active nodes do – usually the gateways
- Uses *hop count* metric
- No routing loop detection, not suitable for large networks!

More on RIP: RIP2

- Successor protocol to RIP
- RIP can only handle Class-Based network addresses.
- RIP only sends network addresses (with trailing zeroes) to other nodes. It uses the first few bits to determine each network's class
- RIP2 is CIDR-based, it sends network prefixes
- Simplest routing protocol to use for small- to medium-sized networks

Link State Routing protocols

- Nodes (i.e. routers) send out announcements whenever they, or the links connected to them, change state. Announcements are sent using *reliable flooding*
- When a router receives a state change announcement, it updates its network topology graph, then runs a shortest-path-first algorithm to compute its new routing table
- The announcements are usually called *Link State Packets (LSPs)*

Reliable Flooding Challenges

- Need to remove old data when link fails … how?
  - LSPs carry sequence numbers to distinguish new from old
  - Only accept (and forward) the ‘newest’ LSP seen from a node
  - Send a new LSP with cost infinity to signal a link is down
- What happens when a node (router) fails and restarts?
  - What sequence number should it use? Don’t want data ignored
  - Aging
    - Put a TTL in the LSP, periodically decremented by each router
    - When TTL=0, purge the LSP and flood that LSP to tell everyone else to do the same
  - Or : when receiving an ‘old’ LSP from a node, tell that node what the current sequence number is, rather than just dropping the LSP

Open SPF (OSPF): 6.6.4

- Open standard proposed by IETF
- Implements SPF/link state algorithm with Dijkstra at core
- Link State Packet (LSP) flooding communicates link states to all nodes
- Supports host-specific routes and network-specific routes
- Supports type-of-service routing and load balancing
- Addresses security issues between gateways (authentication)
- Allows for network partitioning etc.

IS-IS

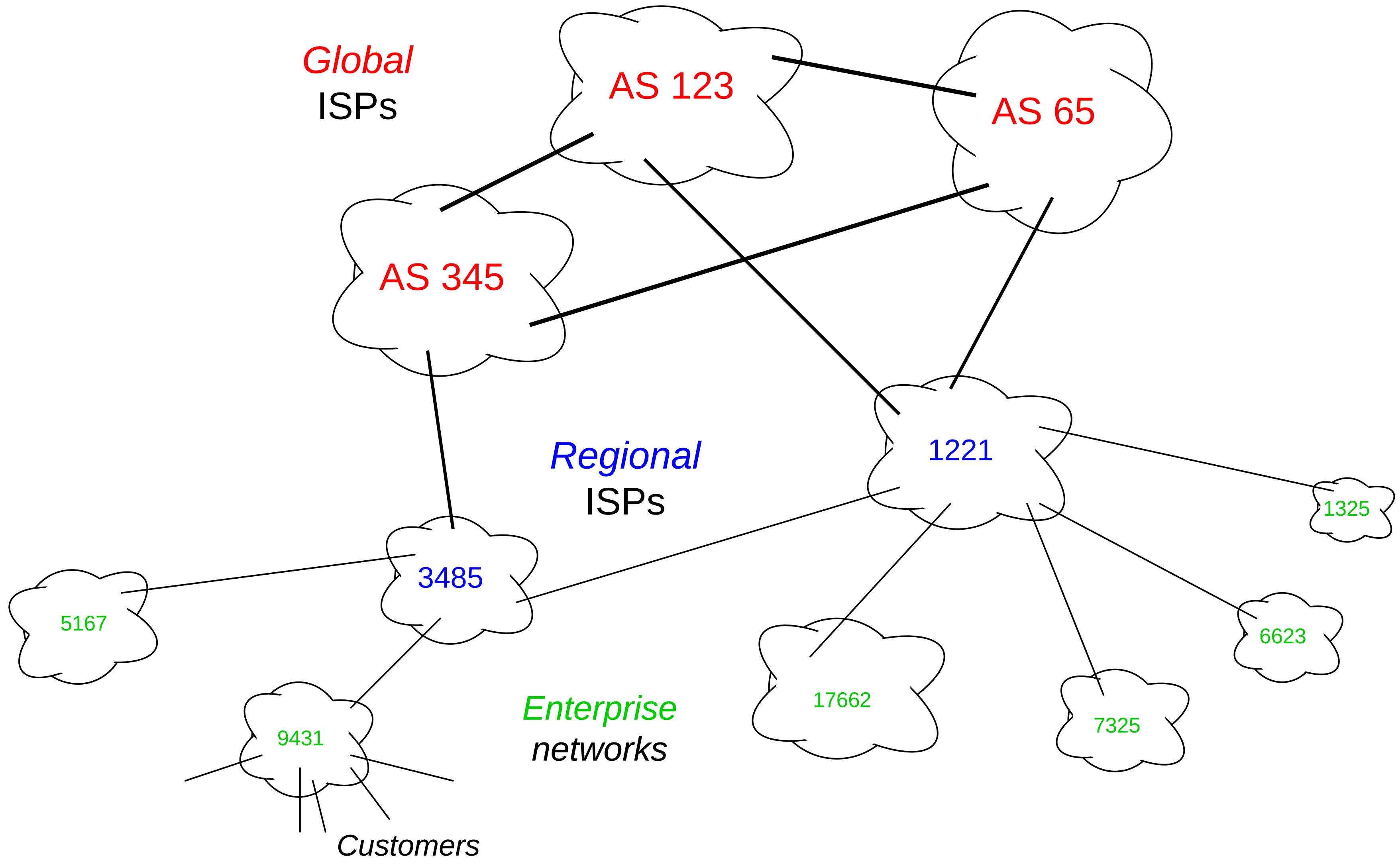
- Open standard proposed by OSI
- Very similar to OSPF, but with fewer ‘added features’
- Widely used as an IGP in large provider networks, e.g. Sprint and AT&T
- Providers often adjust link weights as a way to tune their network, e.g. to move traffic away from overloaded links

Hierarchical Routing, BGP: 6.6.5

- We can view the global Internet as a graph linking provider networks together, forming a hierarchy – ‘tier 1,’ tier 2,’ .. down to ‘customer’ networks
- Each network in the graph is described as an Autonomous System (AS), and referred to by an AS number (ASN)
- Autonomous Systems use the Border Gateway Protocol (BGP) as their Exterior Gateway protocol, so as to provide global routing in the Internet

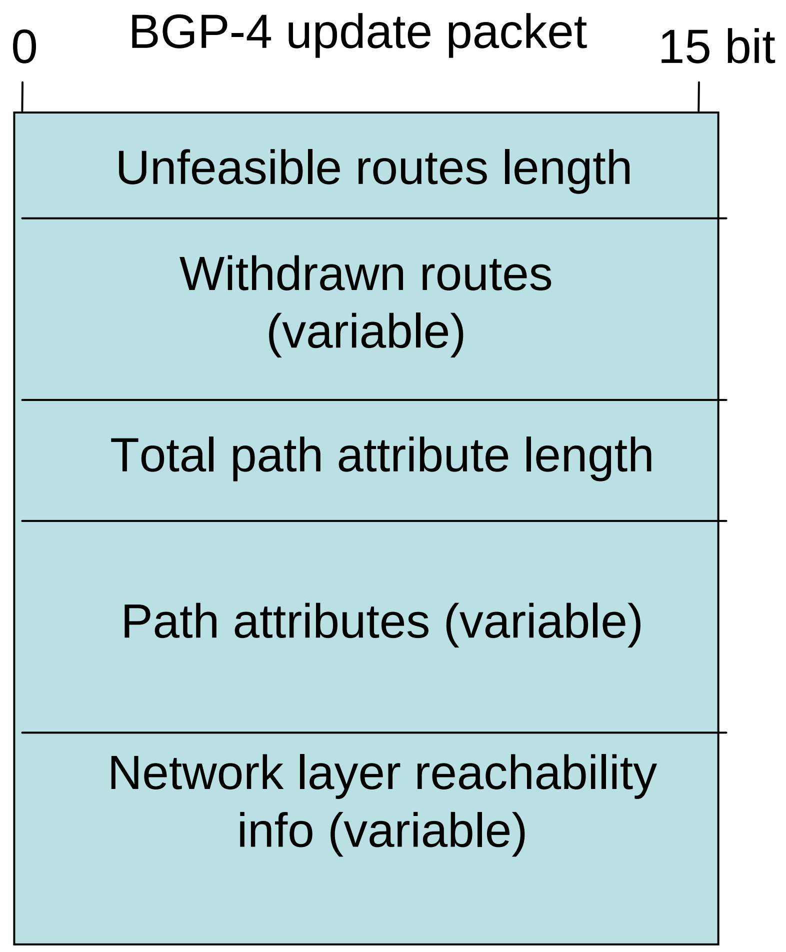
- BGP uses *paths* as its primary ‘cost’ measure
- A BGP *path* is a list of the ASNs indicating the path a packet may traverse to reach a specified *network prefix*

Internet Topology – Overview



Border Gateway Protocol (BGP)

- Uses a Bellman-Ford (distance vector) type algorithm to route between ASs
- 'Cost' becomes a fuzzy concept – other factors ('policy') can also influence routing decisions
- Swaps 'route information' rather than cost
- 'Getting there' is main concern
- Routing loop suppression
- BGP-4 is most common version


0
BGP-4 update packet
15 bit
Unfeasible routes length
Withdrawn routes (variable)
Total path attribute length
Path attributes (variable)
Network layer reachability info (variable)

Routing summary

- Scalability is an important aspect
- Reality demands a distributed approach
- Network hierarchies and routing by network reduce complexity – route between gateways at higher levels
- Bellman Ford and SPF make good IGPs
- At top level, ‘cost’ becomes a fuzzy issue
- Can manage routing in very large networks if we compromise ‘cost’ for ‘getting there’

Circuit Types

- A *permanent circuit* is a hardwired connection between two end points. It is permanently assigned to that service, usually uses dedicated cables, and is unavailable to anybody else. Example is a private intercom.
- A *switched circuit* is *established* on demand by a *connection request*. It then resembles a permanent circuit and is dedicated to the user until a *disconnection request* asks for the connection to be *broken*. Example is a traditional telephone. The component connections are dedicated for the life of the connection, but then released for other users when the connection is broken.
- A *virtual circuit* has no links dedicated to any user. Users establish a virtual connection which looks like a switched circuit to the user. They send packets or messages which are directed through the network by *routing tables*, set up by the *connection request*.

Virtual Circuits

A *virtual circuit* appears to the user as equivalent to a dedicated point-point service but is maintained by computers. It will usually transport data at a guaranteed rate (bit/s) and with guaranteed reliability and error rate.

- Internally information is carried by many small packets, each with a short *virtual circuit number* which identifies its virtual circuit over that physical link
- The virtual circuit number changes as the packet is switched by a switch or router from one incoming link to an outgoing link, according to information held in *routing tables*
- It is the combination of entries in the routing tables of successive switches which defines the end-to-end virtual circuits

DO NOT confuse Virtual Circuits with Virtual LANs!