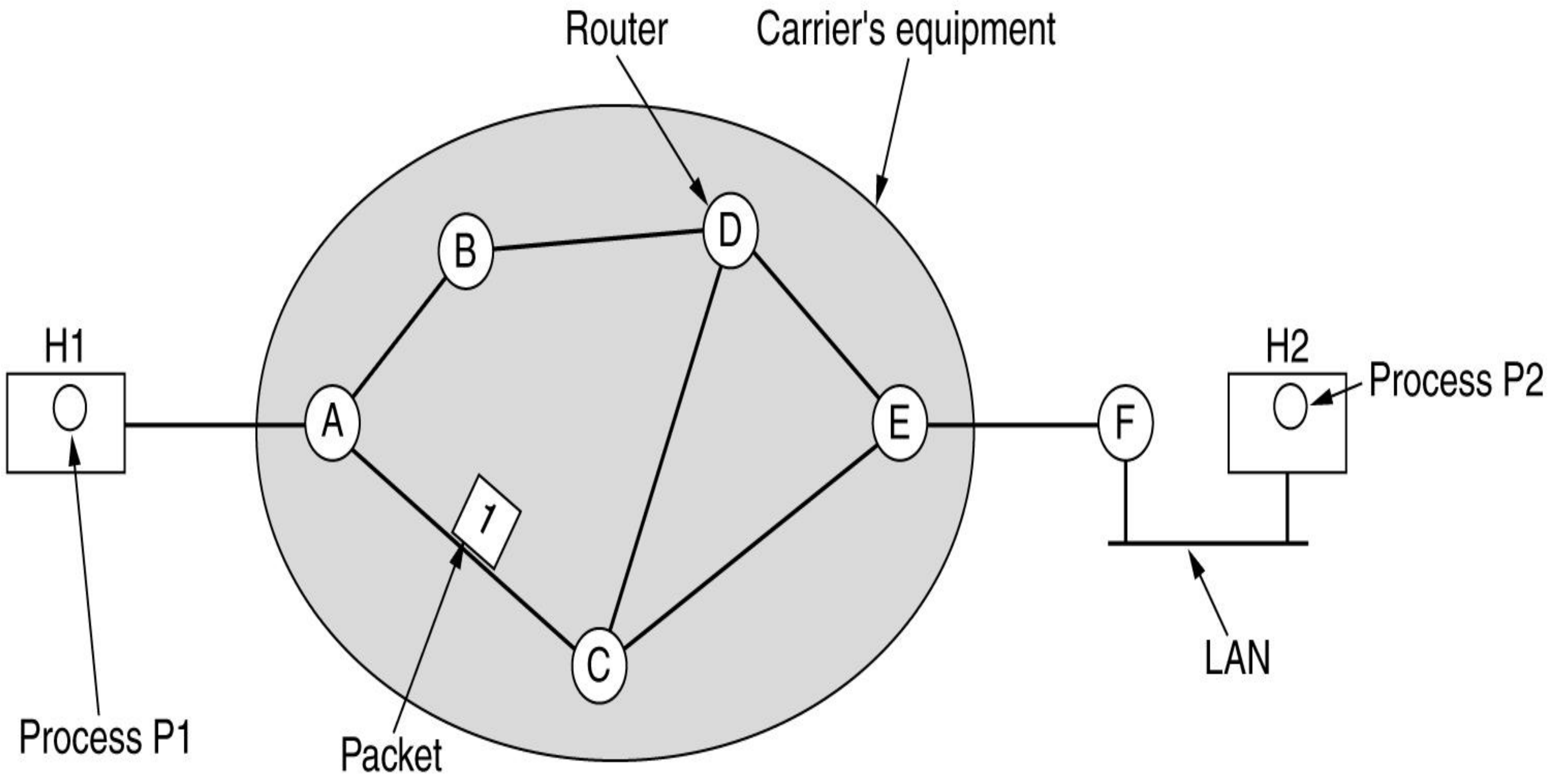
Network Layer Design Issues

- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

Store-and-Forward Packet Switching



The environment of the network layer protocols.

Services Provided to the Transport Layer

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

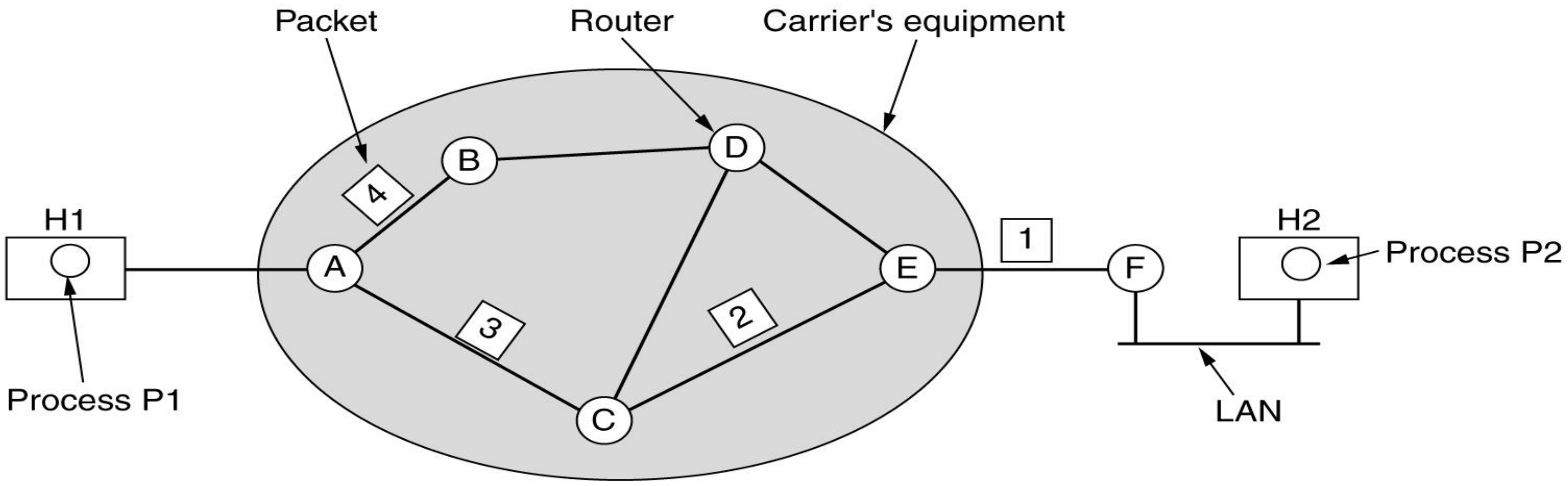
Services Provided to the Transport Layer

- One camp (represented by the Internet community) argues that network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET

  eg: Internet
- packets are frequently called **datagrams** (in analogy with telegrams)
- Other camp (represented by the telephone companies) argues that the subnet should provide a reliable, connection-oriented service

  eg: ATM (Asynchronous Transfer Mode)
- connection is called a **VC (virtual circuit),** in analogy with the physical circuits set up by the telephone system

Implementation of Connectionless Service



A's table

initially

Dest.	Line
A	–
B	B
C	C
D	B
E	C
F	C

later

A	–
B	B
C	C
D	B
E	B
F	B

C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Routing within a datagram subnet.

Implementation of Connectionless Service

- Let us assume that the message is four times longer than the maximum packet size
- so the network layer has to break it into four packets, 1, 2, 3, and 4 and
- sends each of them in turn to router A using PPP.
- At this point the carrier takes over.
- Every router has an internal table telling it where to send packets for each possible destination.
- Each table entry is a pair consisting of a destination and the outgoing line to use for that destination.
- Only directly-connected lines can be used.

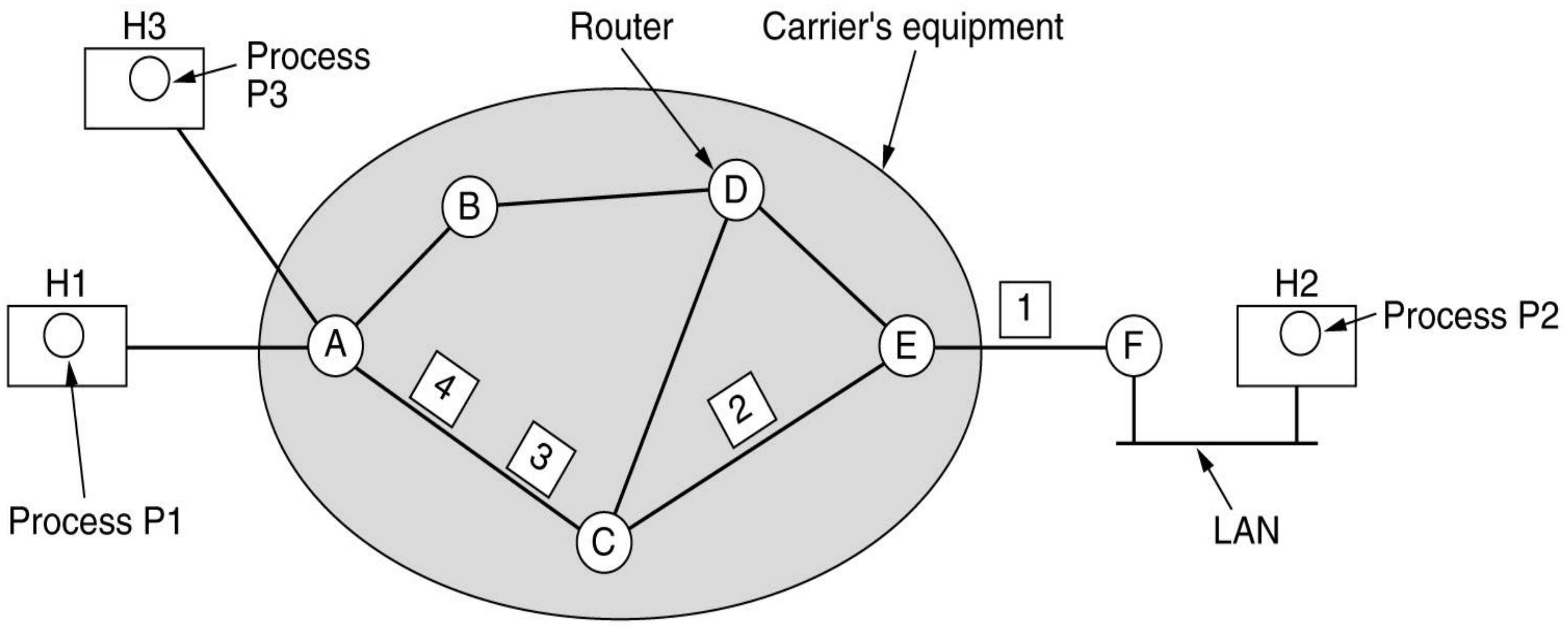
Implementation of Connectionless Service

- For example, in Fig., A has only two outgoing lines—to B and C
- so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router.
- A's initial routing table is shown in the figure under the label ‘initially’.
- As they arrived at A, packets 1, 2, and 3 were stored briefly (to verify their checksums).
- Then each was forwarded to C according to A's table.
- Packet 1 was then forwarded to E and then to F.
- When it got to F, it was encapsulated in a data link layer frame and sent to H2 over the LAN.

Implementation of Connectionless Service

- Packets 2 and 3 follow the same route
- something different happened to packet 4.
- When it got to A it was sent to router B, even though it is also destined for F.
- A learned about a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label ‘later’.
- So, A decided to send packet 4 via a different route than that of the first three.
- The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm**

Implementation of Connection-Oriented Service



A's table

In		Out	
H1	1	C	1
H3	1	C	2

C's table

A	1	E	1
A	2	E	2

E's table

C	1	F	1
C	2	F	2

Routing within a virtual-circuit subnet.

Implementation of Connection-Oriented Service

- For connection-oriented service, we need a virtual-circuit subnet.
- The idea behind virtual circuits is to avoid having to choose a new route for every packet sent.
- when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.
- That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works.
- When the connection is released, the virtual circuit is also terminated.

Implementation of Connection-Oriented Service

- With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.
- Eg: In fig, host H1 has established connection 1 with host H2.
- It is remembered as the first entry in each of the routing tables.
- The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1.
- Similarly, the first entry at C routes the packet to E, also with connection identifier 1.

Implementation of Connection-Oriented Service

- Now let us consider what happens if H3 also wants to establish a connection to H2.
- It chooses connection identifier 1 (because it is initiating the connection and this is its only connection)
- tells the subnet to establish the virtual circuit.
- This leads to the second row in the tables.
- Note that we have a conflict
- because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this.

Implementation of Connection-Oriented Service

- For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection.
- Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets.
- In some contexts, this is called label switching

Comparison of Virtual-Circuit and Datagram Subnets

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Routing Algorithms

- Routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.
- If the subnet uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.
- If the subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up.
  - Thereafter, data packets just follow the previously-established route.
  - This is sometimes called session routing because a route remains in force for an entire user session (e.g., a login session at a terminal or a file transfer).

Routing Algorithms

- Distinction between routing & forwarding:
  - Routing is the process for making the decision which routes to use
    - responsible for filling in and updating the routing tables
  - Forwarding is what happens when a packet arrives
    - each packet looks up the routing tables for the outgoing line to use
- Certain properties are desirable in a routing algorithm
  - correctness
  - simplicity
  - robustness
  - stability
  - fairness and
  - optimality

Routing Algorithms

- Robustness
  - When network comes on the air, it may be expected to run continuously for years without system wide failures
  - Expected failures:
    - Hosts, routers, and lines will fail repeatedly
    - hardware and software failures
    - topology will change many times
- Stability
  - A stable algorithm reaches equilibrium and stays there

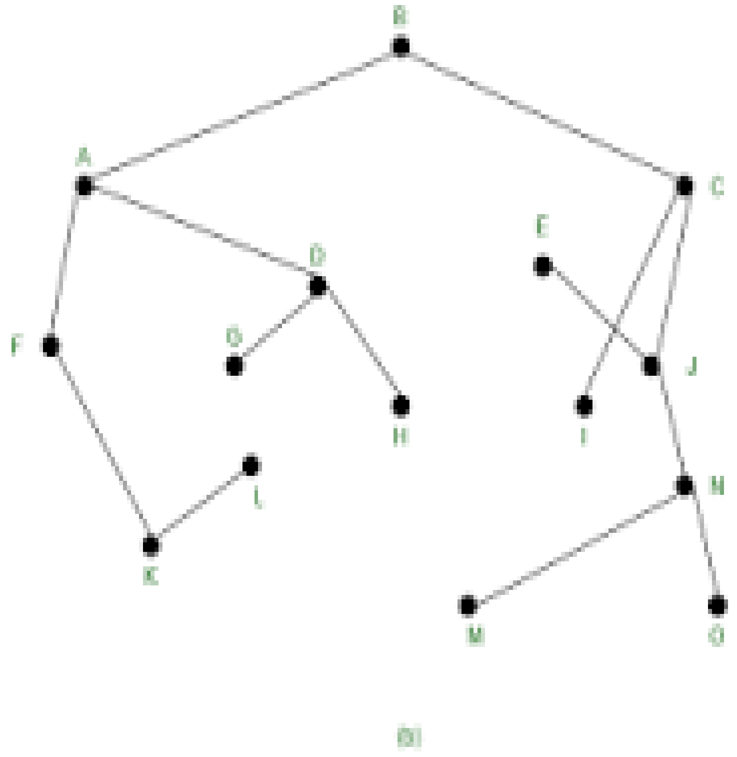
Routing Algorithms

- networks attempt to minimize the number of hops a packet must make
- because reducing the number of hops tends to
  - reduces the delay
  - reduce the amount of bandwidth consumed
  - which improves the throughput

Optimality Principle

Statement of the optimality principle :

It states that **if the router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route.**



Routing Algorithms

- Routing algorithms can be grouped into two major classes:
  - Nonadaptive
  - Adaptive
- Nonadaptive algorithms
  - do not base their routing decisions on measurements or estimates of the current traffic and topology.
  - Instead, the choice of the route to use to move the packet is computed in advance, off-line, and downloaded to the routers when the network is booted.
  - This procedure is sometimes called **static routing**

Routing Algorithms

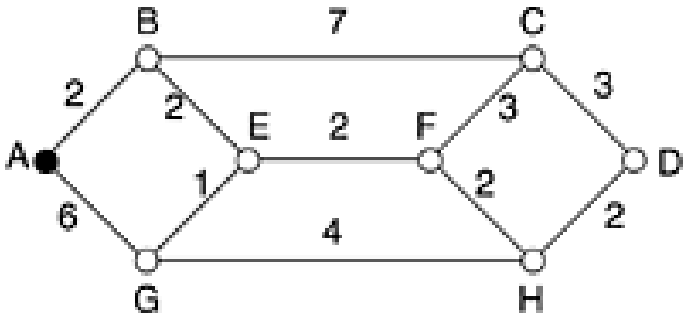
- Adaptive algorithms
  - change their routing decisions to reflect changes in the topology, and the traffic.
  - differ in
    - where they get their information
      - (e.g., locally from adjacent routers, or from all routers)
    - when they change the routes
      - (e.g., every ΔT sec, when the load changes or when the topology changes)
    - what metric is used for optimization
      - (e.g., distance, number of hops, or estimated transit time)

Routing Algorithms

- Shortest Path Routing
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing
- Routing for Mobile Hosts

Routing Algorithms - Shortest Path Routing

- **Shortest Path Routing**
  - widely used in many forms because it is simple and easy to understand.
  - Idea is to build a graph of the subnet with
    - each node of the graph representing a router and
    - each arc of the graph representing a communication line (often called a link).
  - To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.
  - One way of measuring path length is the number of hops.
    - Using this metric, the paths ABC and ABE in Fig. are equally long.



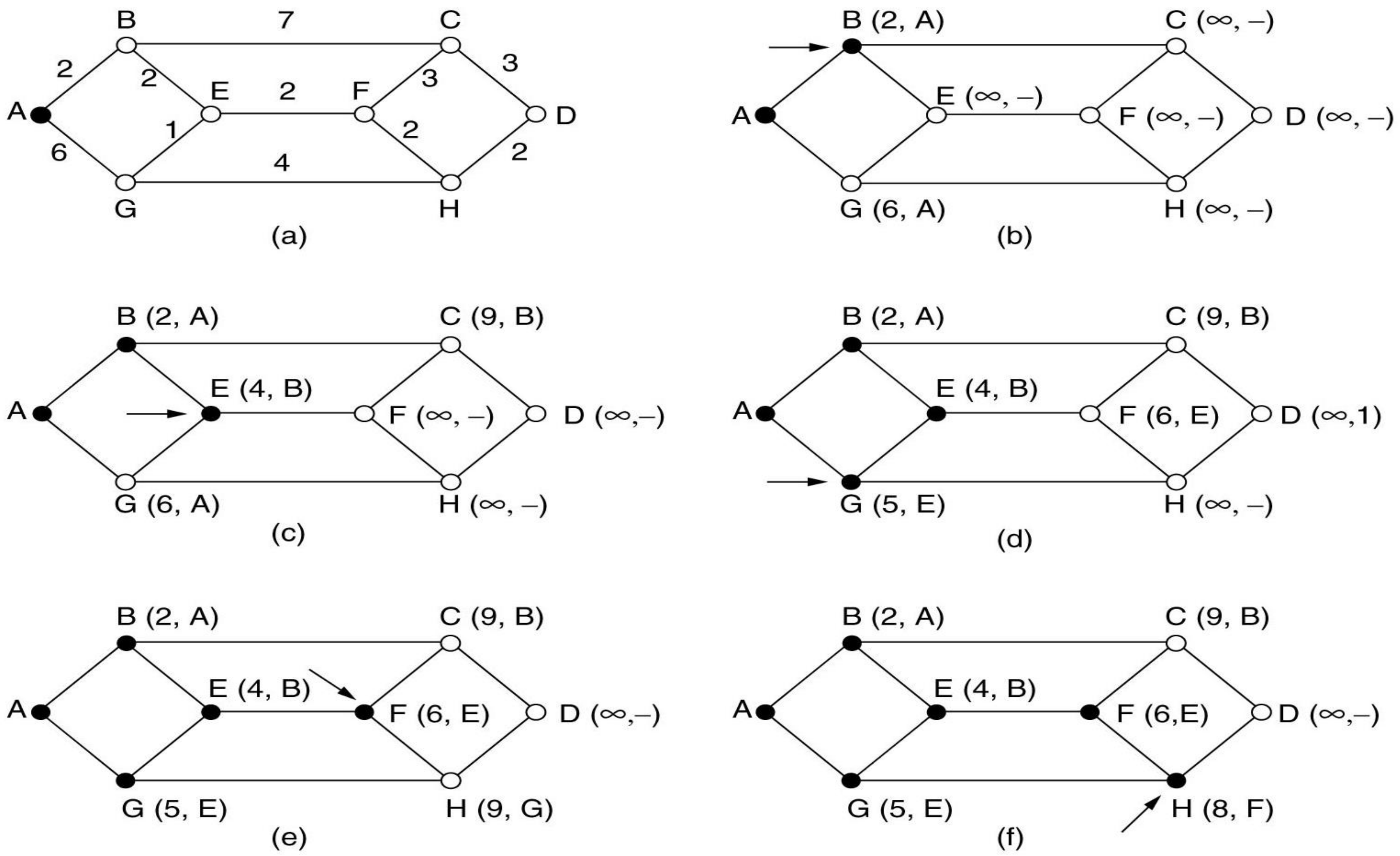
Shortest Path Routing

- Another metric is the geographic distance in kilometers,
  - in which case ABC is clearly much longer than ABE (assuming the figure is drawn to scale).
- Other metrics
  - each arc could be labeled with the mean queuing and transmission delay for some standard test packet as determined by hourly test runs.
  - With this graph labeling, the shortest path is the fastest path.
- Generally, the labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors.
- By changing the weighting function, the algorithm would then compute the "shortest" path measured according to any one of a number of criteria or to a combination of criteria

Shortest Path Routing

- **Shortest Path Routing (**Dijkstra's algorithm**)**
  - Several algorithms for computing the shortest path between two nodes of a graph are available. This one is due to Dijkstra (1959).
  - Each node is labeled (in parentheses) with its distance from the source node along the best known path.
  - Initially, no paths are known, so all nodes are labeled with infinity.
  - As the algorithm proceeds and paths are found, the labels may change, reflecting better paths.
  - A label may be either tentative or permanent.
  - Initially, all labels are tentative.
  - When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

Shortest Path Routing



The first 5 steps used in computing the shortest path from A to D. The arrows indicate the working node.

Shortest Path Routing

- **Shortest Path Routing (**Dijkstra’s algorithm**)**
  - Figure shows a weighted undirected graph where the weight represents distance
  - Let us start by marking node *A* as permanent, indicated by a filled-in circle
  - we examine each of the nodes adjacent to A (the working node), relabeling each one with the distance to A
  - we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent.
  - This one becomes the new working node
  - This Process is continued for all the nodes available in the graph.

Shortest Path Routing

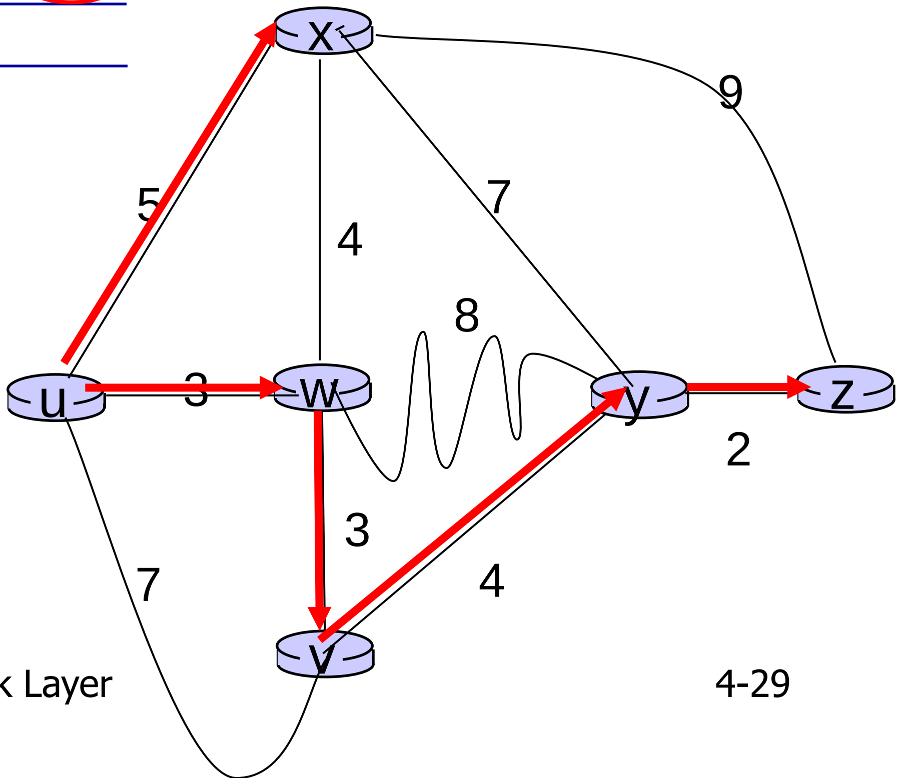
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

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

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

$$e.g.,\ \ D(v) = \min(D(v), D(w) + c(w,v))$$
$$= \min\{7, 3+3\} = 6$$



notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Routing Algorithms-Flooding

- **Flooding:**
  - every incoming packet is sent out on every outgoing line except the one it arrived on.
  - Flooding obviously generates vast (infinite) numbers of duplicate packets
  - some measures are taken to damp the process.
  - One such measure is
    - to have a hop counter contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero.
    - Ideally, the hop counter should be initialized to the length of the path from source to destination.
    - If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.

Routing Algorithms - Flooding

- An alternative technique is
  - to keep track of which packets have been flooded, to avoid sending them out a second time.
  - This is achieved by having the source router put a sequence number in each packet it receives from its hosts.
  - Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen.
  - If an incoming packet is on the list, it is not flooded.

Flooding

- **Flooding:**
  - To prevent the list from growing without bound, each list should be augmented by a counter, k, meaning that all sequence numbers through k have been seen.
  - When a packet comes in, it is easy to check if the packet is a duplicate;
  - if so, it is discarded.
  - Furthermore, the full list below k is not needed, since k effectively summarizes it.
- **Variant: Selective flooding:**
  - routers do not send every incoming packet out on every line,
  - It will send only on those lines that are going approximately in the right direction.

Flooding

- **Flooding** is not practical in most applications, but it does have some uses.
- For example,
  - In military applications, where large numbers of routers may be blown to bits at any instant, the tremendous robustness of flooding is highly desirable.
  - In distributed database applications, it is sometimes necessary to update all the databases concurrently, in which case flooding can be useful.
  - In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range.
  - Used as a metric against which other routing algorithms can be compared.

Flooding

- Advantages of Flooding:
  - always chooses the shortest path because it chooses every possible path in parallel.
  - Consequently, no other algorithm can produce a shorter delay (if we ignore the overhead generated by the flooding process itself).