CHAPTER 5
THE NETWORK LAYER

I. Network Layer Design Issues (网络层设计概述)

II. Routing Algorithm (路由算法)

III. Internetworking (网络互连)

IV. The Network Layer in the Internet

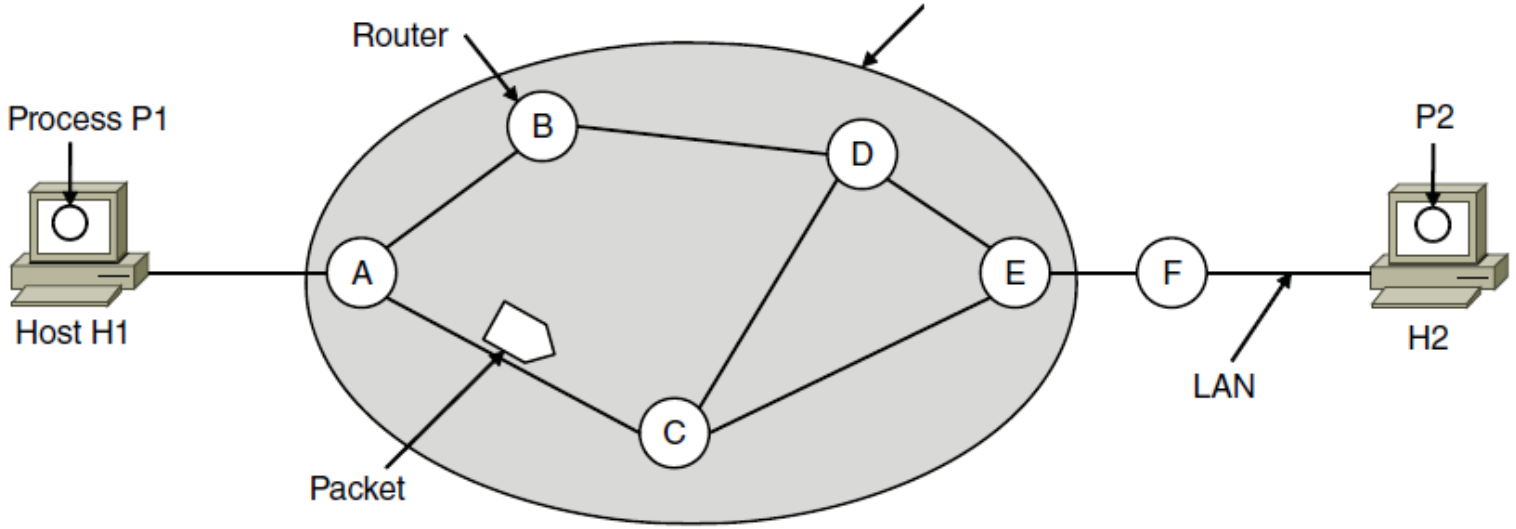
V. *Congestion Control Algorithms (拥塞控制算法)

VI. *QoS Control Algorithms (服务质量控制算法)

NETWORK LAYER DESIGN ISSUES

1. Store-and-Forward Packet Switching
   (存储转发分组交换)
2. Services Provided to the Transport Layer
   (为传输层提供的服务)
3. Implementation of Connectionless Service
   (无连接服务的实现)
4. Implementation of Connection-Oriented Service
   (面向连接服务的实现)
5. Comparison of Virtual-Circuit and Datagram Subnets
   (虚电路或数据报子网的比较)

Network Layer Design Issues: Store-and-forward packet switching



A host with a packet to send transmits it to the nearest router.

The packet is received, verified, and stored.

Then it is forwarded to the next router.

This step can be repeated many times.

Finally the packet reaches the destination host.

Network Layer Design Issues: Services provided to the transport layer

- The network layer services should been designed with the following goals in mind.
  - The services should be **independent** of the router technology.
  - The transport layer should be shielded from the number, type, and topology of the routers present.
  - The network addresses made available to the transport layer should use a **uniform** numbering plan, even cross LANs and WANs.
- Two types of network layer services:
  - Connection-oriented service vs connection-less service

Network Layer Design Issues: Services provided to the transport layer

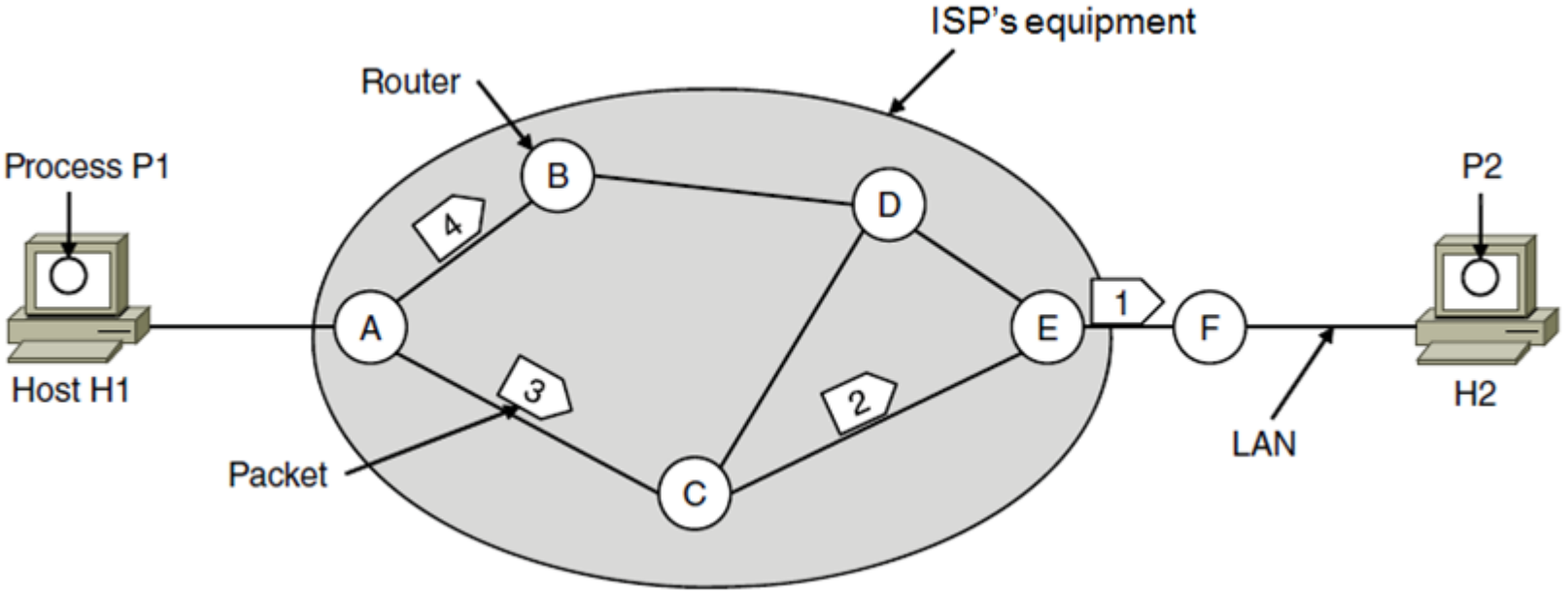
- Two-type of services:
  - **Connection-less service (无连接服务)**:
    - 40+ years of experience with the computer network, unreliable internet, hosts doing error control and flow control.
  - **Connection-oriented service (面向连接服务)**:
    - 100+ years of experience with the worldwide telephone system, quality of service.

→ **Connection-less + connection-oriented services**.

Network Layer Design Issues: Services provided to the transport layer

- Implementation of connectionless service
  - No advance setup is needed.
  - Packets are injected into the subnet individually and routed independently of each other.
  - The packets are frequently called datagrams (in analogy with telegrams) and the subnet is called a datagram subnet.
- Implementation of connection-oriented service
  - A path from the source router to the destination router must be established before any data packets can be sent.
  - This connection is called a VC (virtual circuit), similar to physical circuits set up by the telephone system,
  - The subnet is called a virtual-circuit subnet.

Network Layer Design Issues: Implementation of Connectionless Service



A's table (initially)

Dest.	Line
A	–
B	B
C	C
D	B
E	C
F	C

A's table (later)

A	–
B	B
C	C
D	B
E	B
F	B

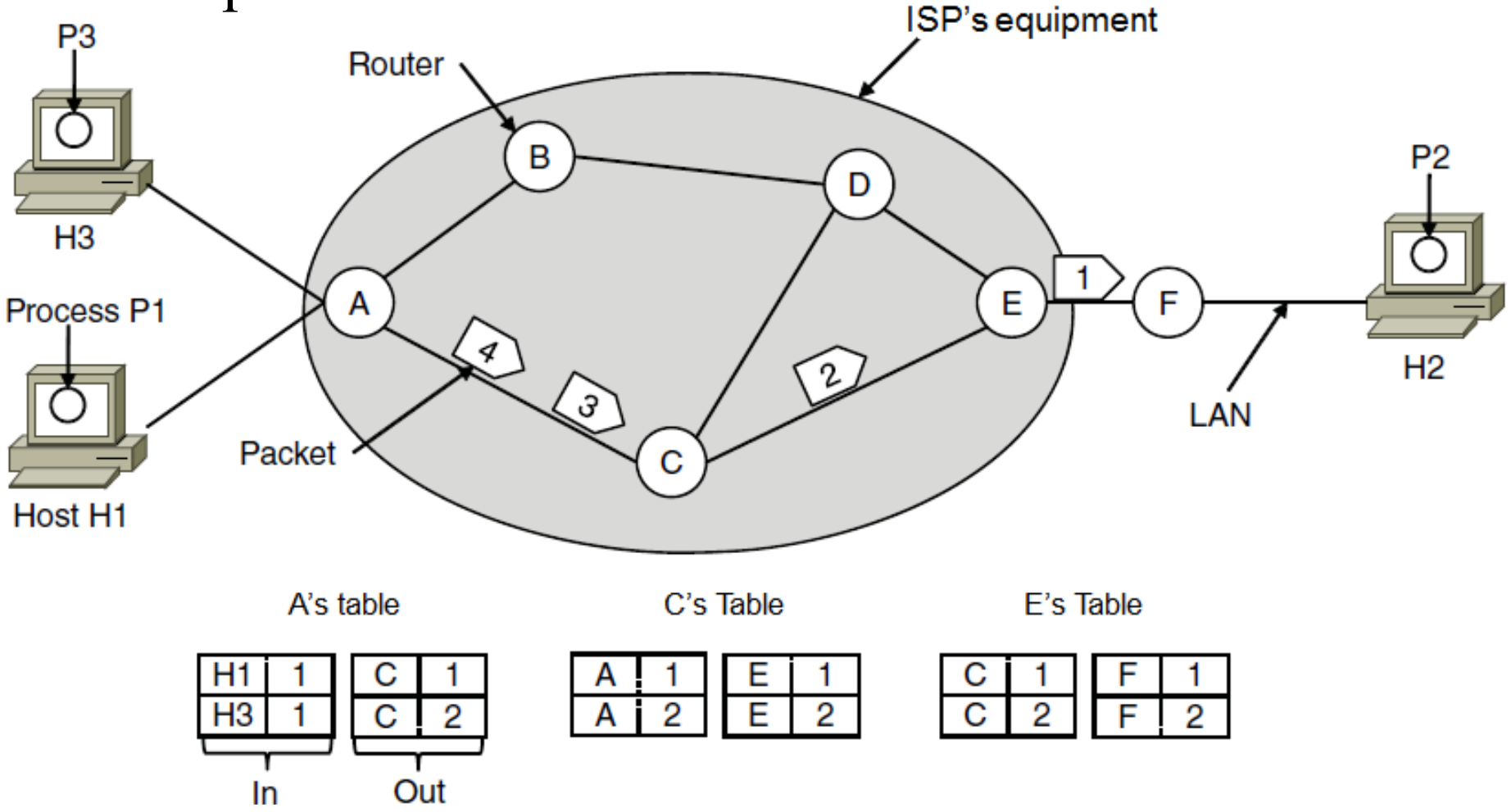
C's table

A	A
B	A
C	–
D	E
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Network Layer Design Issues:
Implementation of Connection-Oriented Service



H1-H2 connection exists (1st line) and now H3-H2 establish connection:
2nd line@A: pkt with ID=1 comes from H3, A outputs to C with ID=2 (A can distinguish). C…, E…

Network Layer Design Issues:
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 (路由算法)

1. The Optimality Principle (最优化原则)
2. Shortest Path Routing (最短路径路由)
3. Flooding (泛洪路由)
4. **Distance Vector (DV) Routing** (距离向量路由)
5. **Link State (LS) Routing** (链路状态路由)
6. Hierarchical Routing (分层路由)
7. Broadcast Routing (广播路由)
8. Multicast Routing (组播路由)
9. Anycast Routing (选播路由)
10. Routing for Mobile Hosts (移动主机的路由)
11. Routing in Ad Hoc Networks (自组织网络的路由)

Routing Patterns and Techniques

- **Unicast**: point-to-point, one-to-one
  - DV routing
  - LS routing
- **Broadcast**: one-to-all
  - Flooding
  - Reverse path forwarding
- **Multicast**: one to many
- **Anycast**: one to any
- **Collection**: all-to-one
  - Used in sensor networks

Routing Algorithms: Introduction

- The main function of the network layer is to route packets from the source machine to the destination machine.
- The **routing algorithm** is used to decide which output line an incoming packet should be transmitted on
  - **For datagram networks**, this decision must be made anew for every arriving data packet since the best route may have changed since last time.
  - **For virtual-circuit networks**, this decision is made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously established route.

Routing Algorithms: Introduction

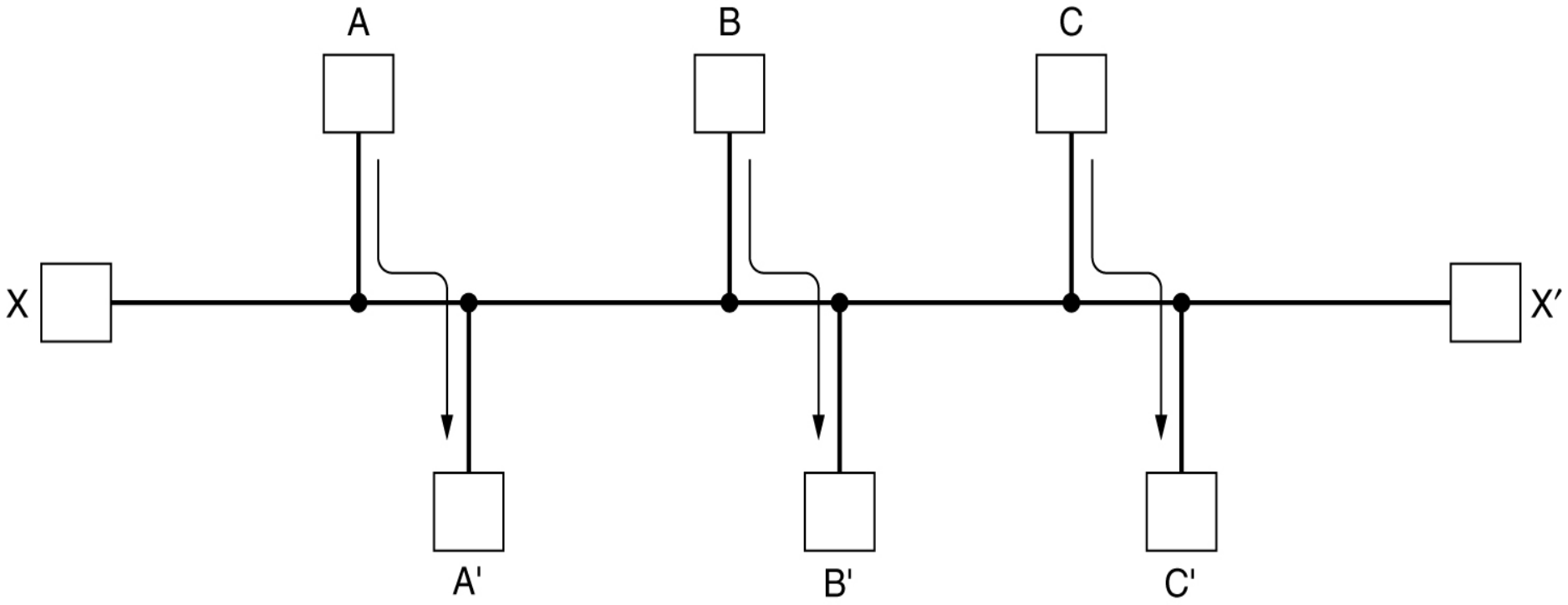
- A router performs **two tasks**:
  - **Forwarding:** To forward the incoming packet according to the routing table
  - **Routing:** To fill in and update the routing table
- Desirable properties in a routing algorithm:
  - **Correctness, simplicity**
  - **Robustness:** The routing algorithm should cope with changes in the topology and traffic without requiring all processes in all hosts to be aborted and the network to be rebooted every time some router crashes.
  - **Stability:** A stable algorithm reaches equilibrium and stays there.
  - **Fairness, efficiency:** Conflict between fairness and efficiency (see next).

Routing Algorithms: Introduction

Assume: the link capacity is unit 1;
there are 4 flows with enough traffic

Efficiency (max total flow): (X,X')=0, others=1 → total=3

Fairness (max-min fairness): All=1/2 → total=2

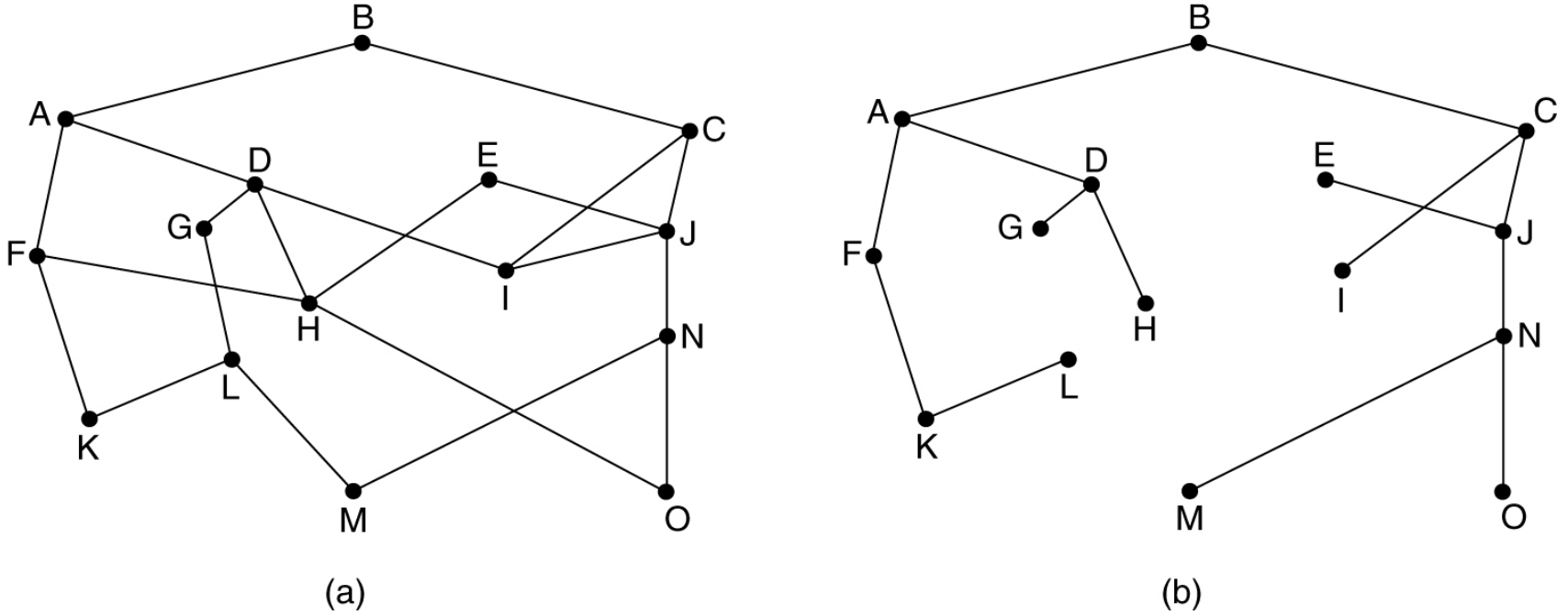


Routing Algorithms: Introduction

- Classes of routing algorithms
  - ***Nonadaptive* algorithms (非自适应算法)** 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 get from ***i*** to ***j*** (for all ***i*** and ***j***) is computed in advance, offline, and downloaded to the routers when the network is booted.
  - ***Adaptive* algorithms (自适应算法)**, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. They differ in
    - where they get their information,
    - when they change the routes, and
    - what metric is used for optimization.

Routing Algorithms: The optimality principle

- **Optimality principle (最优化原则):** If router J is on the optimal path from router I to router K (I→J→K), then the optimal path from J to K also falls along the same route.
- Sink tree (汇集树): The set of optimal routes from all sources to a given destination form a tree rooted at the destination. (a) A subnet. (b) A sink tree for router B.



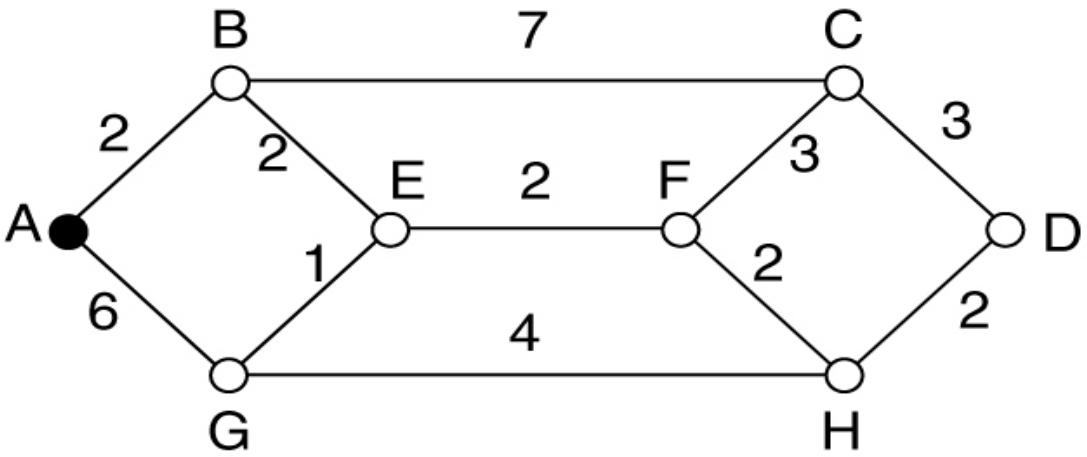
Routing Algorithms: Shortest path routing

- To build a graph of the subnet,
  - with each graph node representing a router and
  - each graph edge representing a communication line.
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them.
- How to measure path length:
  - Hops (跳数), physical distance (物理距离), bandwidth (带宽), traffic (流量), cost (费用), measured delay (测量延迟), mean queue length (平均队列长度) and
  - Other factors or combinations of these factors.

Routing Algorithms: Shortest path routing

- Use Dijkstra's algorithm to compute the shortest path.
  - Each node is labeled with its distance from the source node along the **best known** path.
  - Initially, no paths are known, so all nodes are **temporarily** labeled with **infinity**.
  - As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either **temporary** or **permanent**.
  - 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.

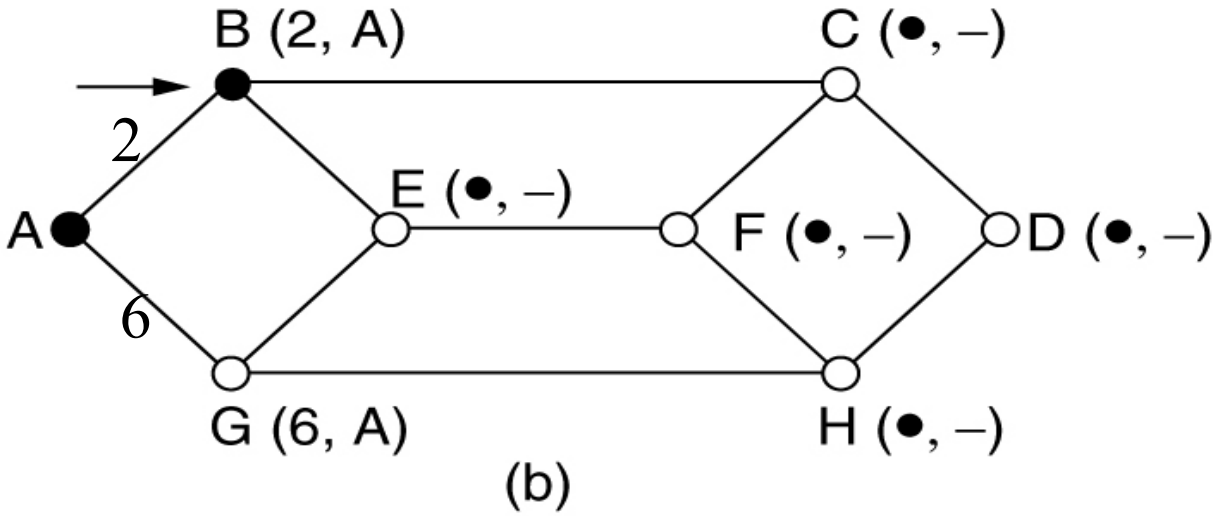
Routing Algorithms: Shortest path routing



Goal: find shortest path from A to D

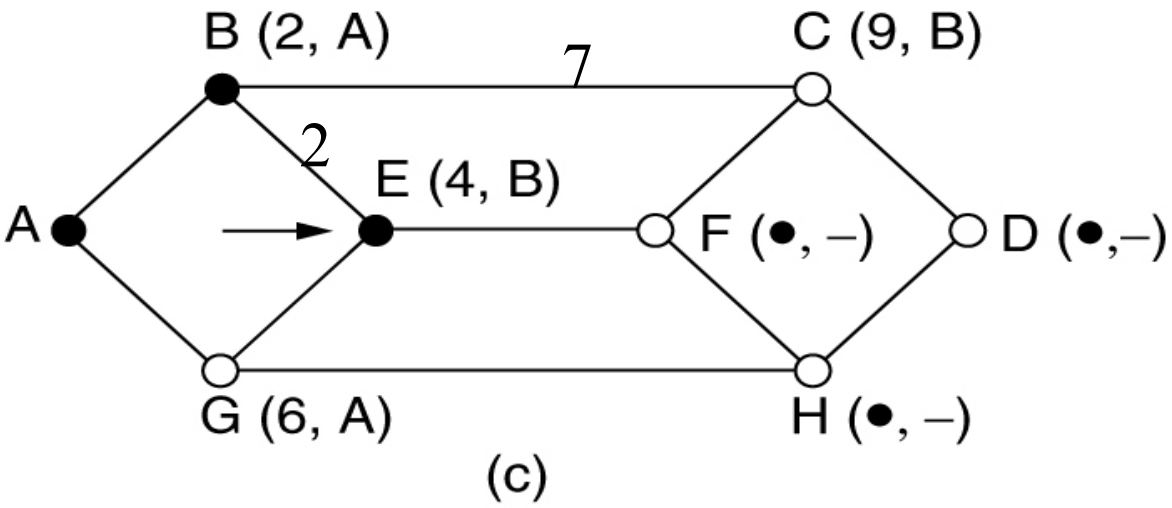
- Mark A as permanent ●

Routing Algorithms: Shortest path routing



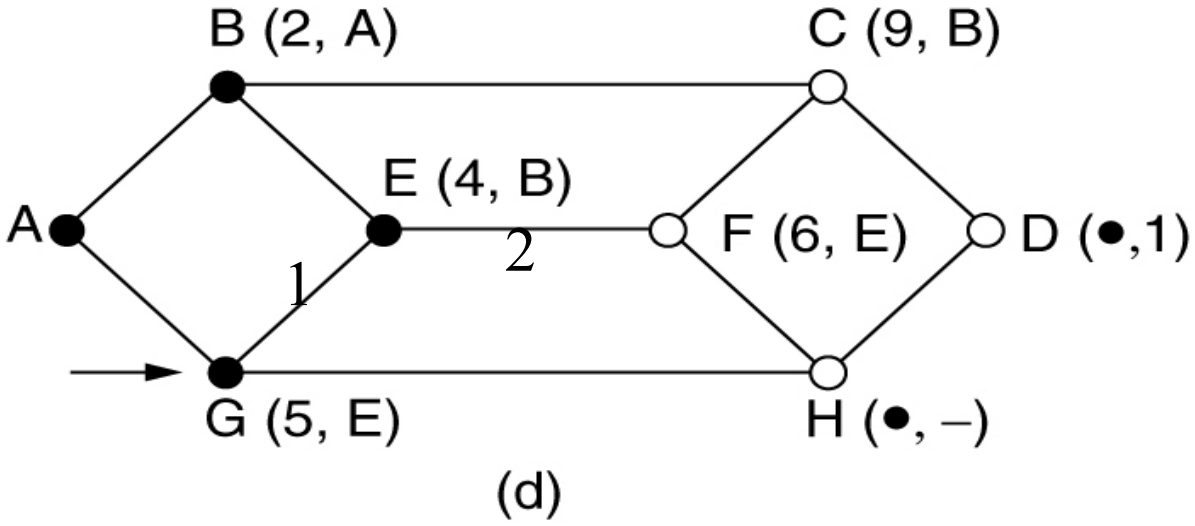
- Examine each node adjacent to A (working node), label it with (distance to A, node from which the probe was made)
  - B: (2,A)
  - G: (6,A)
- Examine all tentatively labeled nodes in graph and make the smallest one (i.e. **B**) as permanent

Routing Algorithms: Shortest path routing



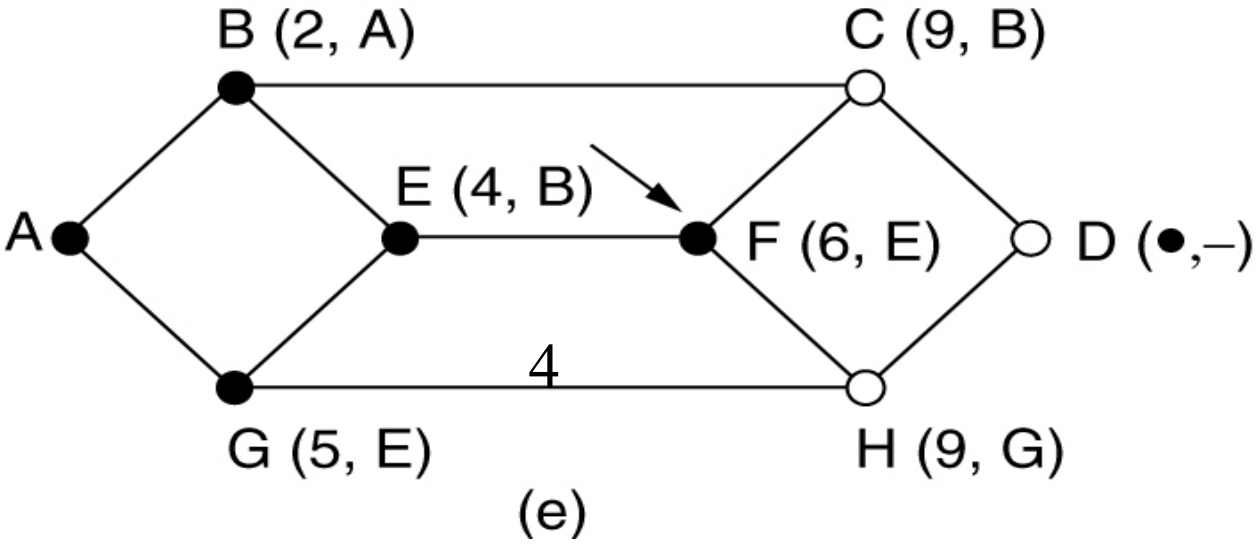
- Examine each node adjacent to **B** (working node), relabel it if a shorter path through B is found:
  - C: infinity → (9,B)
  - E: infinity → (4,B)
- Examine all tentatively labeled nodes in graph and make the smallest one (i.e. **E**) as permanent

Routing Algorithms: Shortest path routing



- Examine each node adjacent to **E** (working node), relabel it if a shorter path through E is found:
  - F: infinity → (6,E)
  - G: (6,A) → (5,E)
- Examine all tentatively labeled nodes in graph and make the smallest one (i.e. **G**) as permanent

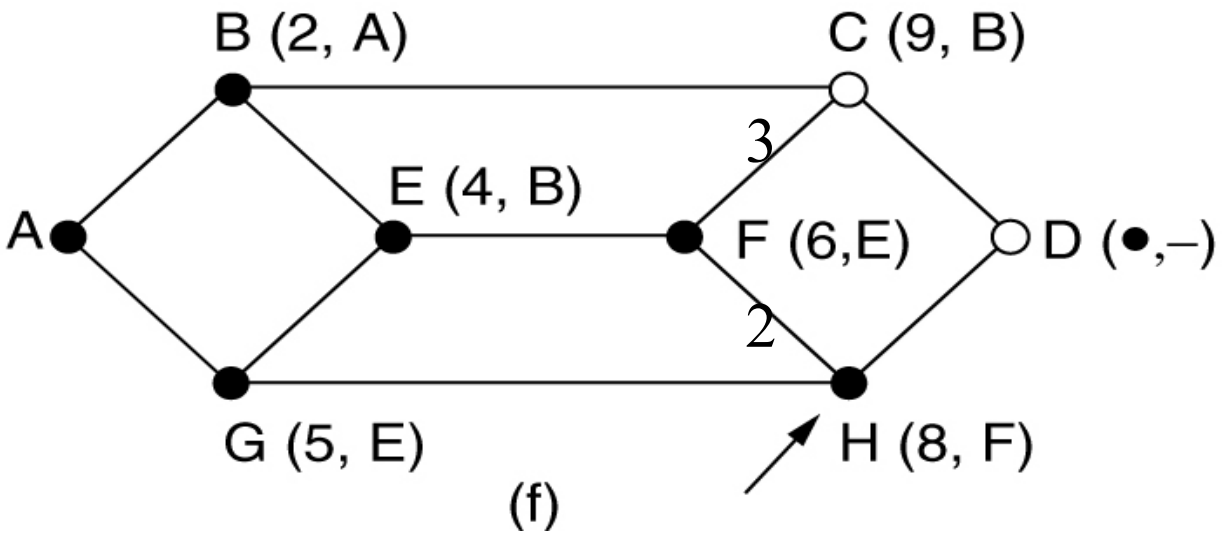
Routing Algorithms: Shortest path routing



- Examine each node adjacent to **G** (working node), relabel it if a shorter path through G is found:
  - H: infinity → (9,G)

- Examine all tentatively labeled nodes in graph and make the smallest one (i.e. **F**) as permanent

Routing Algorithms: Shortest path routing



- Examine each node adjacent to **F** (working node), relabel it if a shorter path through F is found:
  - H: (9,G) → (8,F)
  - C: does not change
- Examine all tentatively labeled nodes in graph and make the smallest one (i.e. **H**) as permanent

Routing Algorithms: Shortest path routing

```
#define MAX_NODES 1024                          /* maximum number of nodes */
#define INFINITY 1000000000                     /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];              /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                                /* the path being worked on */
      int predecessor;                          /* previous node */
      int length;                               /* length from source to this node */
      enum {permanent, tentative} label;        /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) {     /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                        /* k is the initial working node */
  do {                                          /* Is there a better path from k? */
      for (i = 0; i < n; i++)                   /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }
```

Routing Algorithms: Shortest path routing

```
        /* Find the tentatively labeled node with the smallest label. */
        k = 0; min = INFINITY;
        for (i = 0; i < n; i++)
                if (state[i].label == tentative && state[i].length < min) {
                        min = state[i].length;
                        k = i;
                }
        state[k].label = permanent;
    } while (k != s);

    /* Copy the path into the output array. */
    i = 0;  k = s;
    do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Routing Algorithms: Flooding

- **Flooding (泛洪):** Every incoming packet is sent out on every outgoing line except the one it arrived on.
- How to damp the flooding process:
  - One is to have a hop counter contained in the header of each packet, which is decremented at each hop.
  - The other is to keep track of which packets have been flooded, to avoid sending them out a second time.
- A variation of flooding that is slightly more practical is **selective flooding**, i.e. send pkt on those lines in right direction.
- Flooding is not practical in most applications, but it does have some uses such as military applications.

Routing Algorithms: Distance vector routing

- **Distance vector routing (距离矢量路由, 也叫 Bellman-Ford routing)**:
  - Each router maintains a vector or table giving
    - the best known distance to each destination and
    - which line to use to get there.
  - These tables are updated by exchanging information with the neighbors. To update the table, a node:
    - Measure its distance to its neighbors
    - Receive the vectors from its neighbors
    - Compute its own new vector.

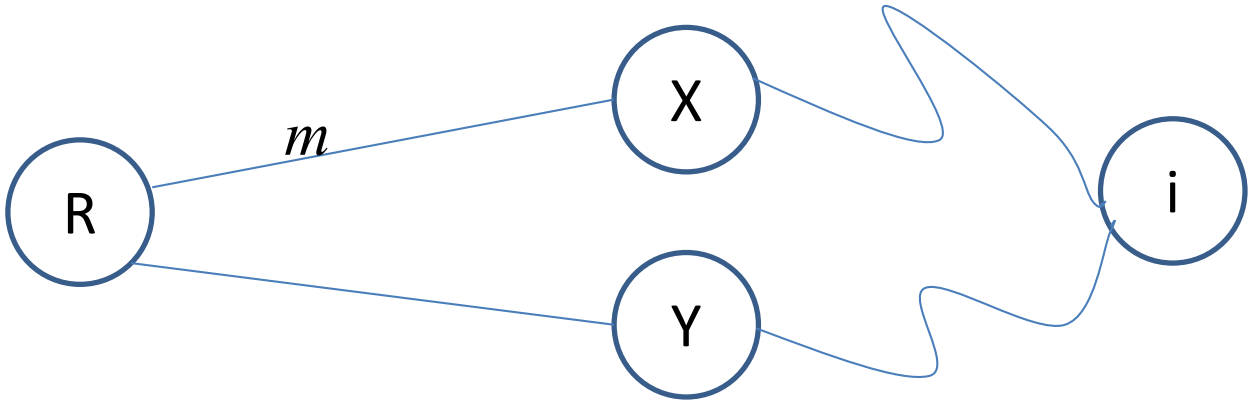
Routing Algorithms: Distance vector routing

- Assume
  - **Delay** is used as a metric
  - A router knows the delay to each of its neighbors.
  - Once every *T* msec, each router sends to each neighbor a DV: a list of its estimated delays to each destination.
  - It also receives a DV from each neighbor.

- For an *n*-node network, how many elements in the DV?

Routing Algorithms: Distance vector routing

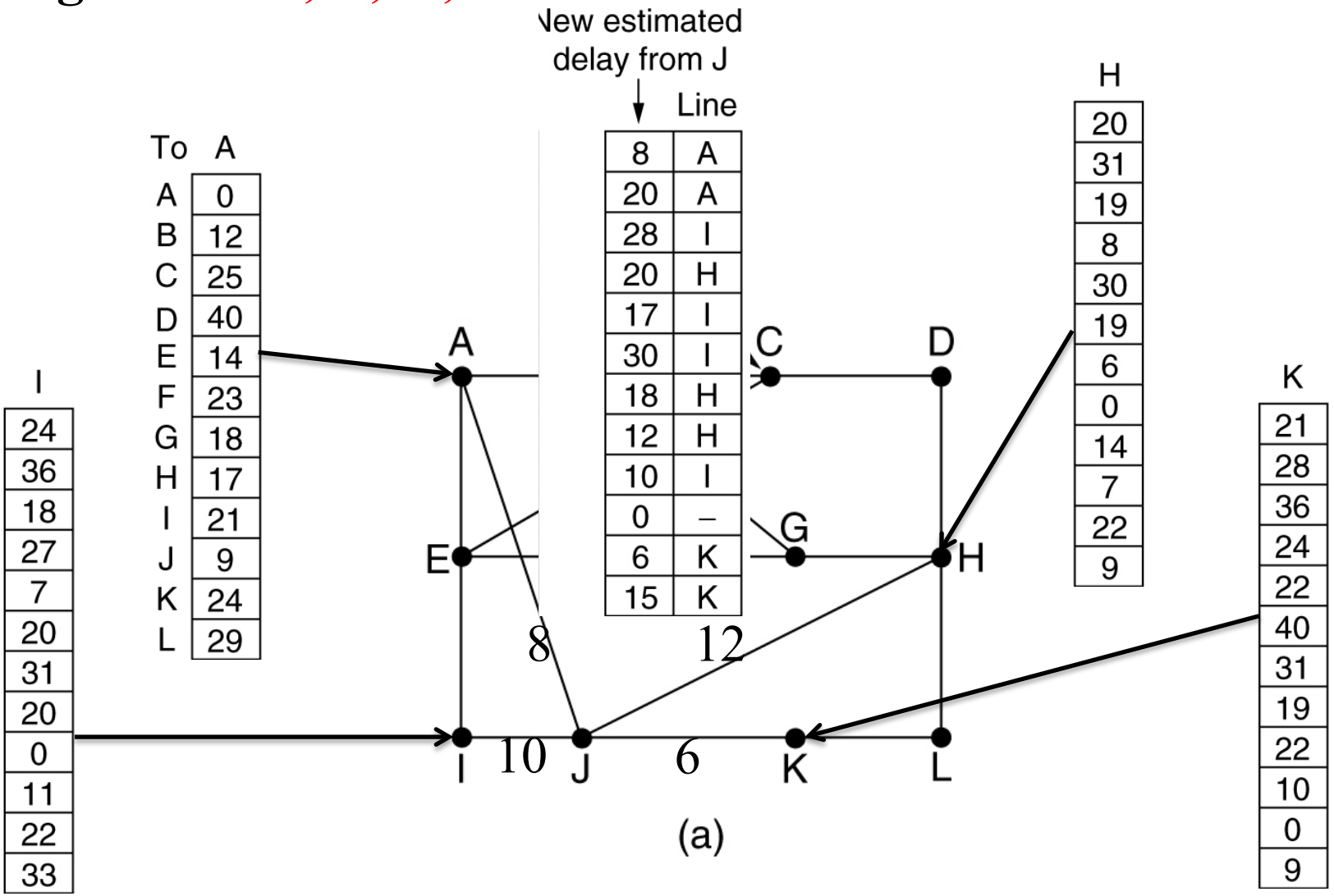
- Imagine for a router **R**:
  - one of these tables has just come in from neighbor **X**, with $\mathbf{X}_i$ being estimate of how long it takes to get to router ***i***.
  - If the router knows that the delay to **X** is ***m*** msec,
  - Then it can reach router ***i*** via **X** in $\mathbf{X}_i + \boldsymbol{m}$ 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 link in its new routing table.
- Note that the old routing table is not used in the calculation.

Routing Algorithms: an example for DV

How router J updates its DV after it receives DVs from its neighbors---I, A, H, K?



Routing Algorithms: Distance vector routing

A drawback of DV:

It reacts rapidly to good news, but leisurely to bad news.

(a)

Consider:

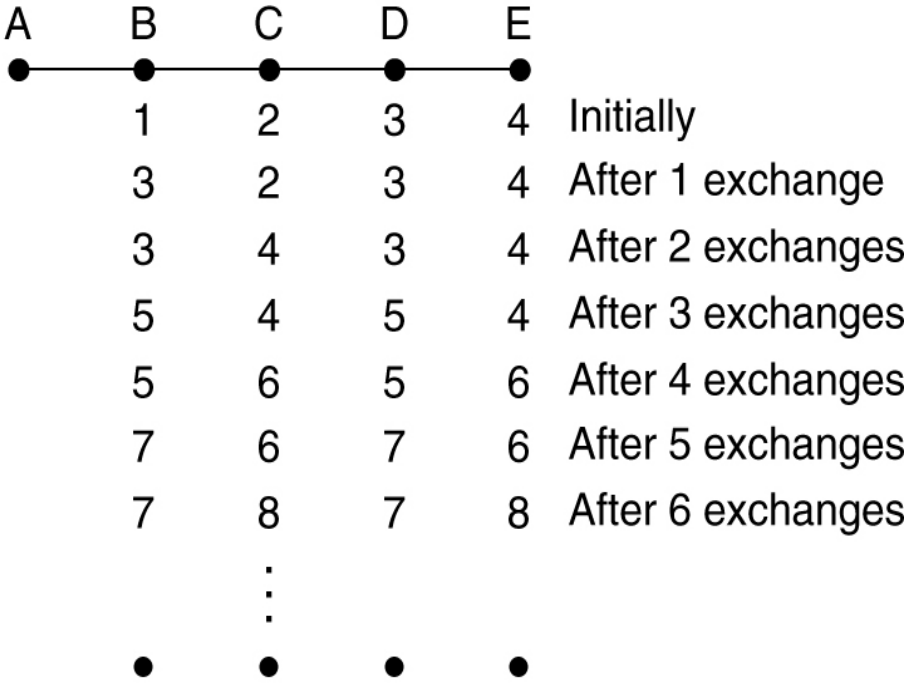
- Five-node linear network
- Routing metric = # of hops
- Time is slotted and at start DV exchanges at all routers simultaneously.
- Initially, A is down, others know this (first line)
- A comes up, …

In subset with longest path *N* hops, within *N* exchanges everyone know about revived lines and routers.

Routing Algorithms: Distance vector routing

A drawback of DV:

It reacts rapidly to good news, but leisurely to bad news.



(b)

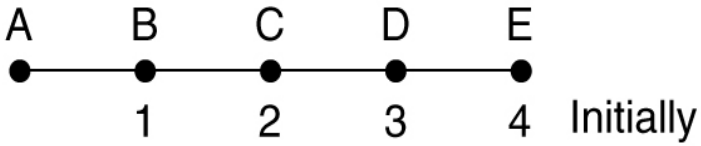
Consider:

- Initially, all routers and lines are up (first line)
- A goes down, …

It is known as the count-to-infinity problem.

Routing Algorithms: Distance vector routing

Attempts to solve **Count-to-infinity** problem



- **Poisoned reverse** (毒性逆转, RFC 1058)

  If C routes through B to get to A:

  - C tells B its distance to A is infinite

- will this completely solve count-to-infinity problem?

  - **No**
  - Can you give an example?

Routing Algorithms: Link state routing

- Problems with distance vector routing
  - The delay metric was queue length, thus it did not take line bandwidth into account when choosing routes. (but it can be replaced by a proper metric)
  - The algorithm often **took too long to converge**.
- → **Link state (LS) routing**: Each router must do:
  1. Discover its neighbors, learn their network address.
  2. Set the distance or cost metric to each of its neighbors.
  3. Construct a packet containing all it has just learned.
  4. Send this packet to and receive packets from all other routers.
  5. Compute the shortest path to every other router.

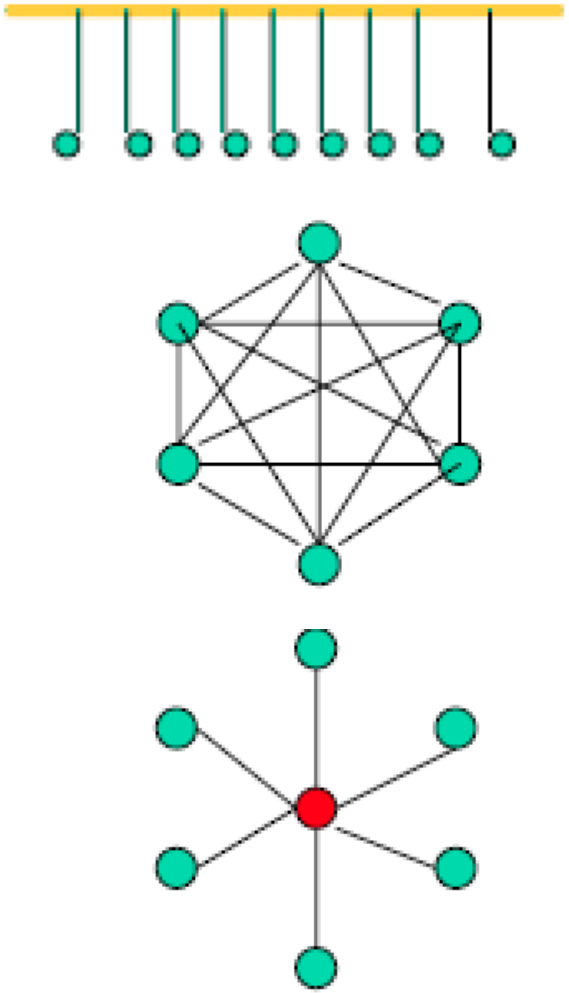
Routing Algorithms: Link state routing

- **Step 1: Learning about the neighbors**
  - One router sends a special HELLO packet on each point-to-point line.
  - The router on the other end is expected to send back a reply telling who it is.
  - These names must be globally unique.
  - **The info about the neighbors can be found out.**

Routing Algorithms: Link state routing

The situation is slightly **complicated** when using Broadcast LAN:

- Modeling LAN as many point-to-point links increases the size of topology and leads to wasteful messages → n^2 links



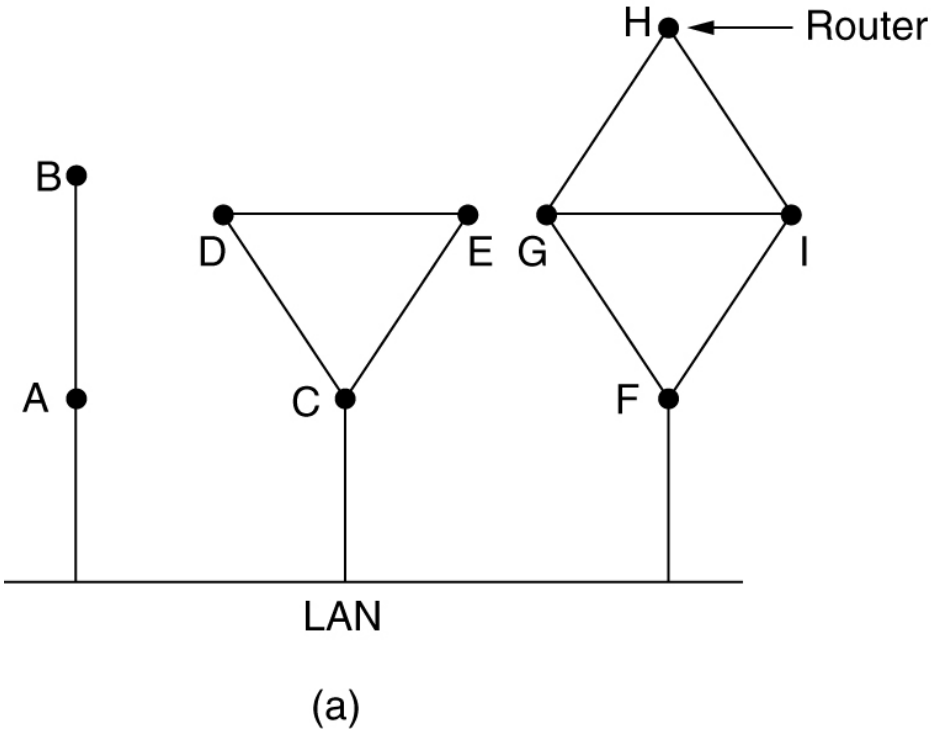
A better way

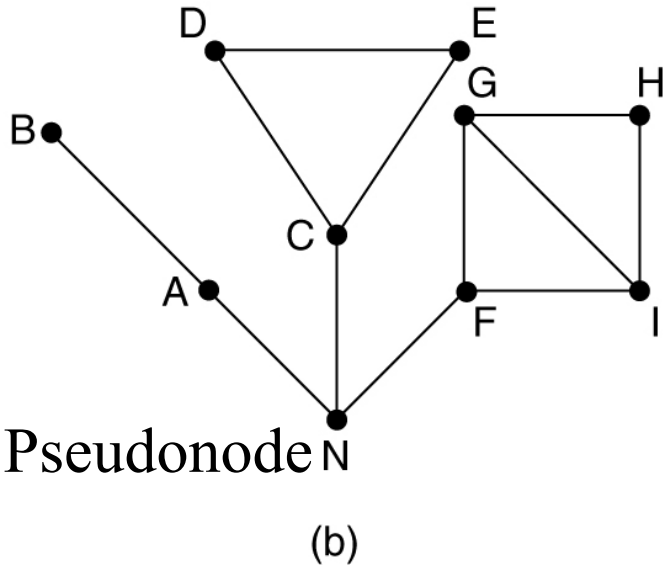
- **Pseudonode**: A new artificial node, e.g. N
- **Designated router**: play the role of N.
- → n links and n+1 nodes

Routing Algorithms: Link state routing

(a) Nine routers and a LAN.

(b) A graph model of (a)





Routing Algorithms: Link state routing

- **Step 2: Measuring line cost**
  - To determine the delay is to send over the line a special ECHO packet that the other side is required to send back immediately.
  - By measuring the round-trip time (RTT) and dividing it by 2, the sending router can get a reasonable estimate of the delay.
  - Average delay value can be better

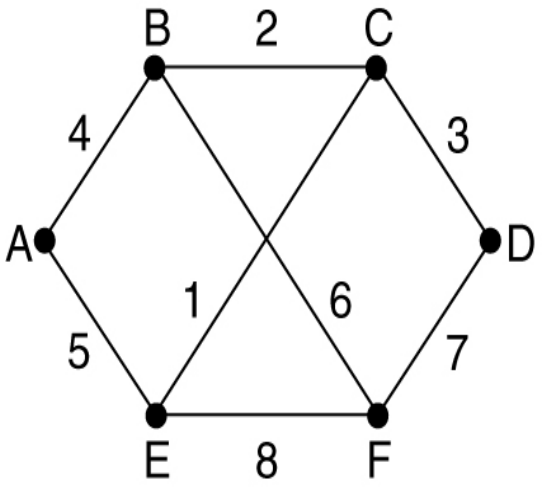
Routing Algorithms: Link state routing

- **Steps 3: Building link state packets**
  - Each router builds a packet containing all the data.
  - The packet starts with the **identity** of the sender, followed by a **sequence number** and **age** and a **list of neighbors (and costs to them)**.
  - To build them is *easy*, but when to build them is *difficult* to determine:
    - To build them periodically (at regular intervals)
    - To build them when some significant event occurs, such as a line or neighbor going down or coming back up again or changing its properties appreciably.

Routing Algorithms: Link state routing

(a) A subnet.

(b) The link state (LS) packets for this subnet.



(a)

Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

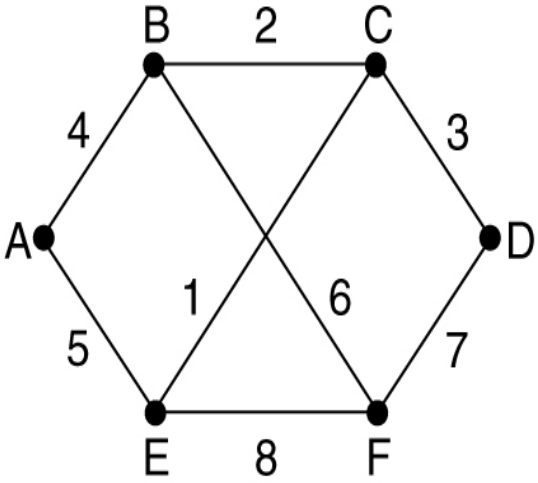
F	
Seq.	
Age	
B	6
D	7
E	8

(b)

Routing Algorithms: Link state routing

- **Steps 4: Distributing the link state packets**
  - **Fundamental idea**: to use **flooding** to distribute the link state packets
  - **Refinement**:
    - Sequence number increments for each new pkt sent.
    - Routers keeps track of all the (source router, sequence) pairs they see.
    - If pkt is new, forward to lines except the incoming one
    - Else (duplicate), discard.
    - If sequence number is lower than highest one seen so far, reject as obsolete.
    - To include age of each packet and decrement it once per second. If age hits 0, discard the information.

Routing Algorithms: LS routing



Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

The data structures for router **B**

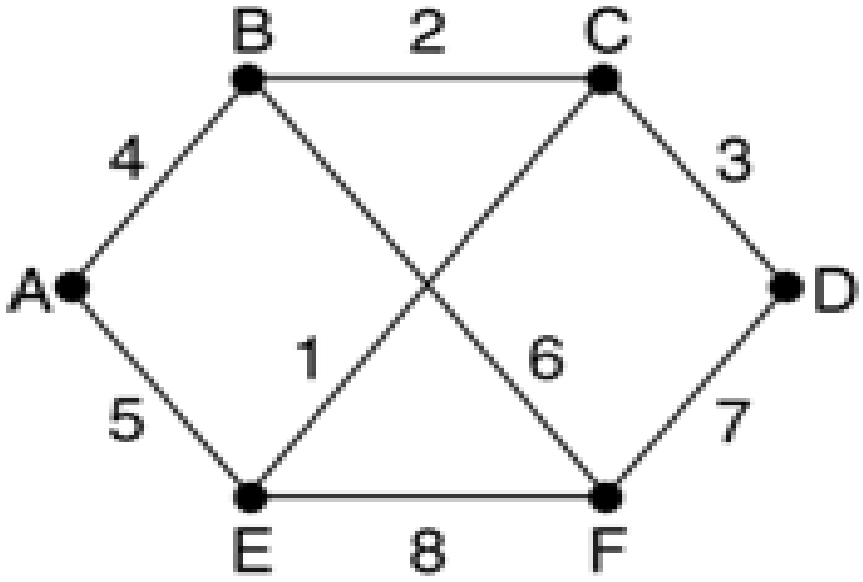
Routing Algorithms: Link state routing

- **Steps 5: Computing the new routes**
  - Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented.
  - **Dijkstra's algorithm** can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables.
- **Some example LS protocols**
  - **IS-IS** (Intermediate System – Intermediate System): designed for early network DECnet, later adopted by ISO for use with OSI protocols.
  - **OSPF** (Open Shortest Path First): designed by IETF several years after IS-IS.

Comparison of LS vs. DV

	Link-State	Distance Vector
Summary	"Tell the world about neighbors"	"Tell the neighbors about the world"
# Messages	More (Broadcast)	Less (Exchange)
Message Size	Smaller (Link Costs)	Larger (DVs)
Convergence Speed	Faster	Slower
Robustness	Better	Worse

Question



- Consider the network:

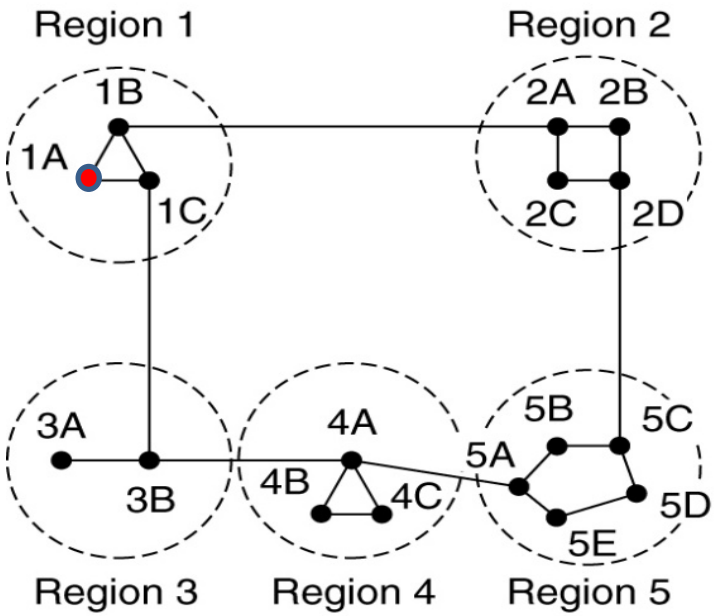
- (1) Link State Routing needs to broadcast link state packets. What is F's link state packet?
- (2) Distance Vector Routing needs to exchange a list of estimated delays (i.e., distance vectors) among neighbors. What is F's initial distance vector? What is F's distance vector after receiving B's initial distance vector?

Routing Algorithms: Hierarchical routing

- As networks grow in size, the routing tables grow proportional, and so do router memory and computing power.
- Two level routing: Every router knows
  - all the details about how to route packets to destinations within its own region
  - but knows nothing about the internal structure of other regions.
- Multiple-level routing:
  - Regions → clusters → zones → groups → …

Routing Algorithms: Hierarchical routing

Reduction of routing tables



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Routing Algorithms: Hierarchical routing

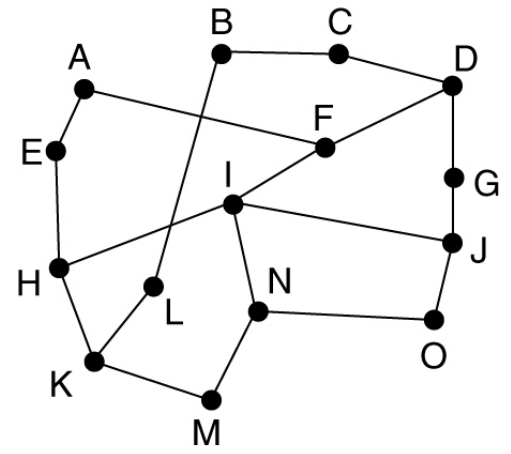
- How many levels should the hierarchy have?
  - Consider a subnet with 720 routers
    - No hierarchy: every router needs 720 routing table entries.
    - 30 routers/region x 24 regions: every router needs 30 for local entries + 23 for other regions = 53 table entries.
    - 10 routers/region x 9 regions/cluster x 8 clusters: every router needs 10 + 8 + 7 = 25 table entries.
  - ➔Kamount and Kleinrock (1979): The optimal number of levels for an N router subnet is **ln** N, requiring a total of e ln N entries per router.

Routing Algorithms: Broadcast routing

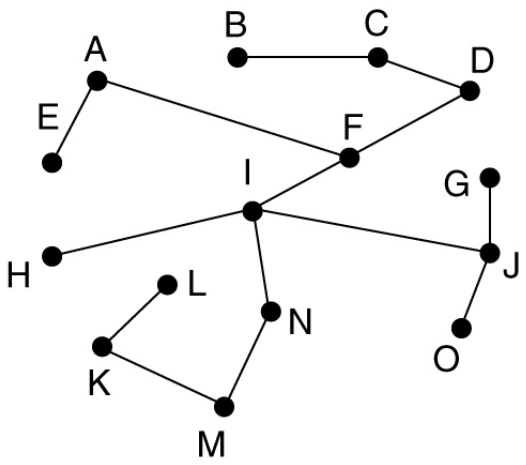
- **Broadcasting**: to send a packet to all destinations simultaneously.
  - The source simply sends a distinct packet to every destination.
  - **Multidestination routing**: each packet contains either a list of destinations or a bit map indicating the desired destinations
  - **Flooding** and its refinements.
  - **Sink-tree based broadcast**: To make explicit use of the sink tree (or other **spanning trees**) for the router initiating the broadcast. → **optimal** in terms of # transmitted pkts.
  - **Reverse path forwarding** when shortest path for regular pkts have been computed → not optimal

Routing Algorithms: Broadcast routing

- **Sink-tree based broadcast**
  - If each router knows which of its lines (both input and output) belongs to the sink tree, it can copy a pkt to all sink tree lines except the incoming one.
  - **Optimal** in terms of # of transmissions, i.e., # of transmissions is **minimum** (e.g. 14 for this example)
  - Does this optimal broadcasting possible in a network only with DV routing?



(a)



(b)

(a) A subnet. (b) a sink tree rooted at I.

Routing Algorithms: Broadcast routing

@router *R*:

event broadcast pkt *p* arrives:

Check *p*'s incoming line
==*lookForLinebyDest*(p.source)

If yes
  Transmit to **all lines**
  except p's incoming line

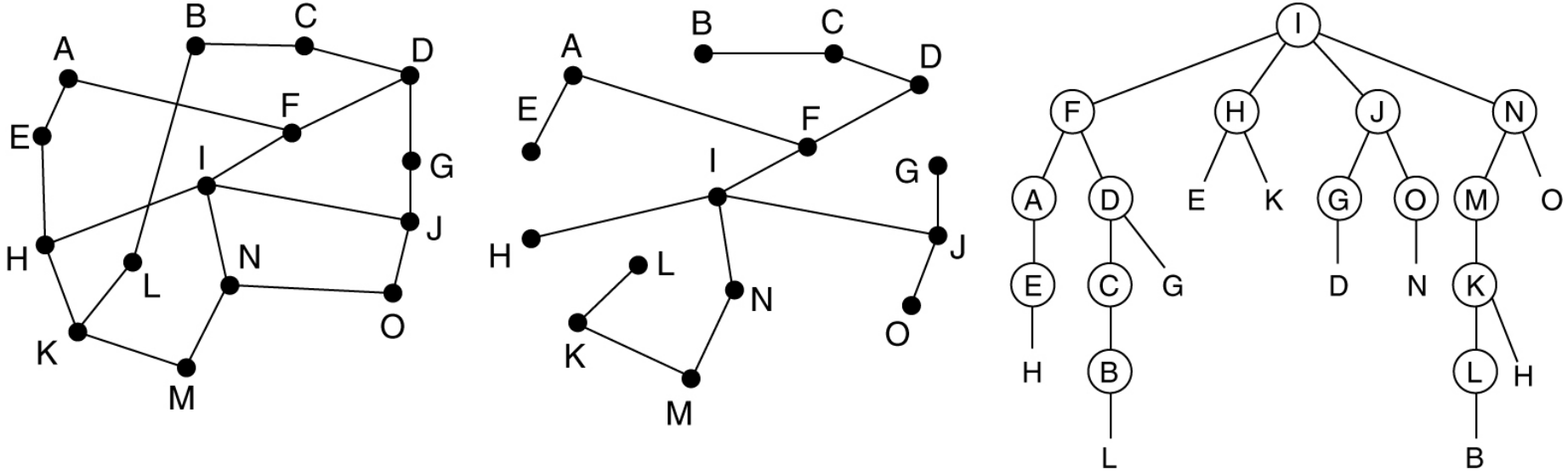
Else
  discard

All lines instead of all sink-tree lines, cause some duplicate transmissions

- **Reverse path forwarding:**
  - When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the packets.
  - If so, there is an excellent chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. Then the router forwards copies of it onto all lines except the one it arrived on.
  - If no, it is discarded as a duplicate.

Routing Algorithms: Broadcast routing

(3rd figure) The tree built by reverse path forwarding. Circle indicates that the router receives a broadcast pkt from the preferred line.



Some explanations (look at (a)):

I broadcasts 4 pkts to F,H,J,N

All on preferred paths, …

Look at N:

Transmits to M and O

Look at M: M continues since N-M is preferred (in sink tree)

Look at O: O stops since N-O is not on preferred path

Routing Algorithms: Multicast routing

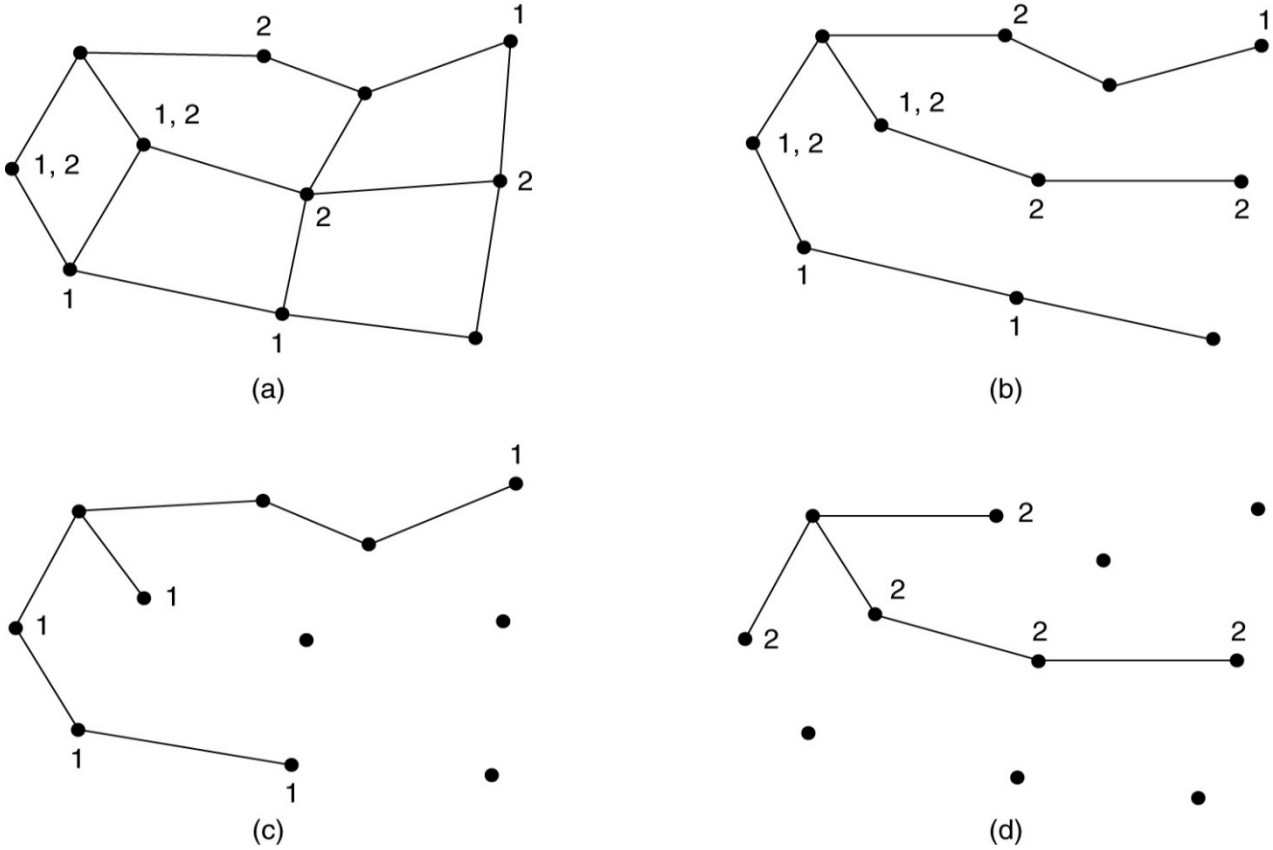
- **Multicasting (组播):** to send messages to well-defined groups that are numerically large in size but small compared to the network as whole.
  - **Group management**: some way is needed to create and destroy groups, and to allow processes to join and leave groups.
  - Basic idea builds on broadcast scheme: send along spanning tree.
  - To make it efficient, multicast routing is to prune the spanning tree, i.e., removing all the lines that do not lead to hosts that are members of the group.

Routing Algorithms: Multicast routing

(a) A network. (b) A spanning tree for the leftmost router.

(c) A multicast tree for group 1.

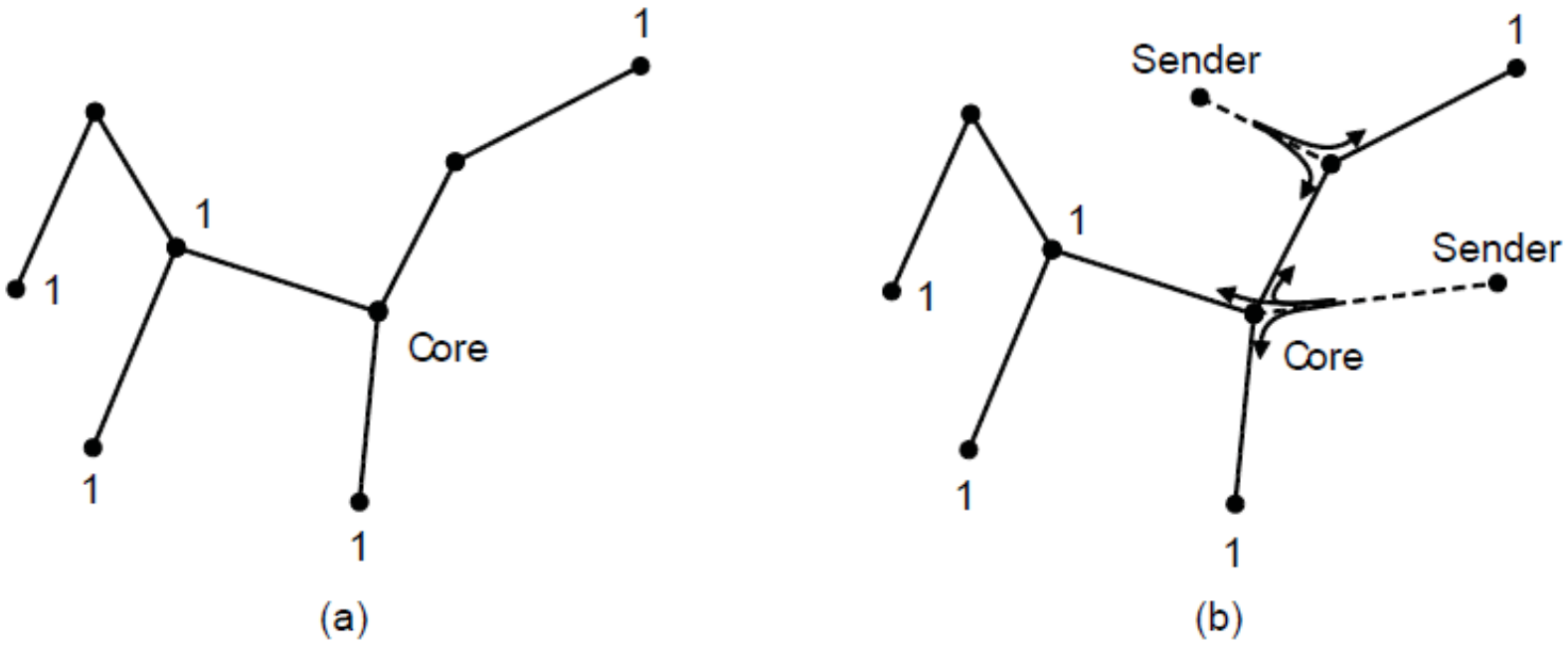
(d) A multicast tree for group 2.



Routing Algorithms: Multicast routing

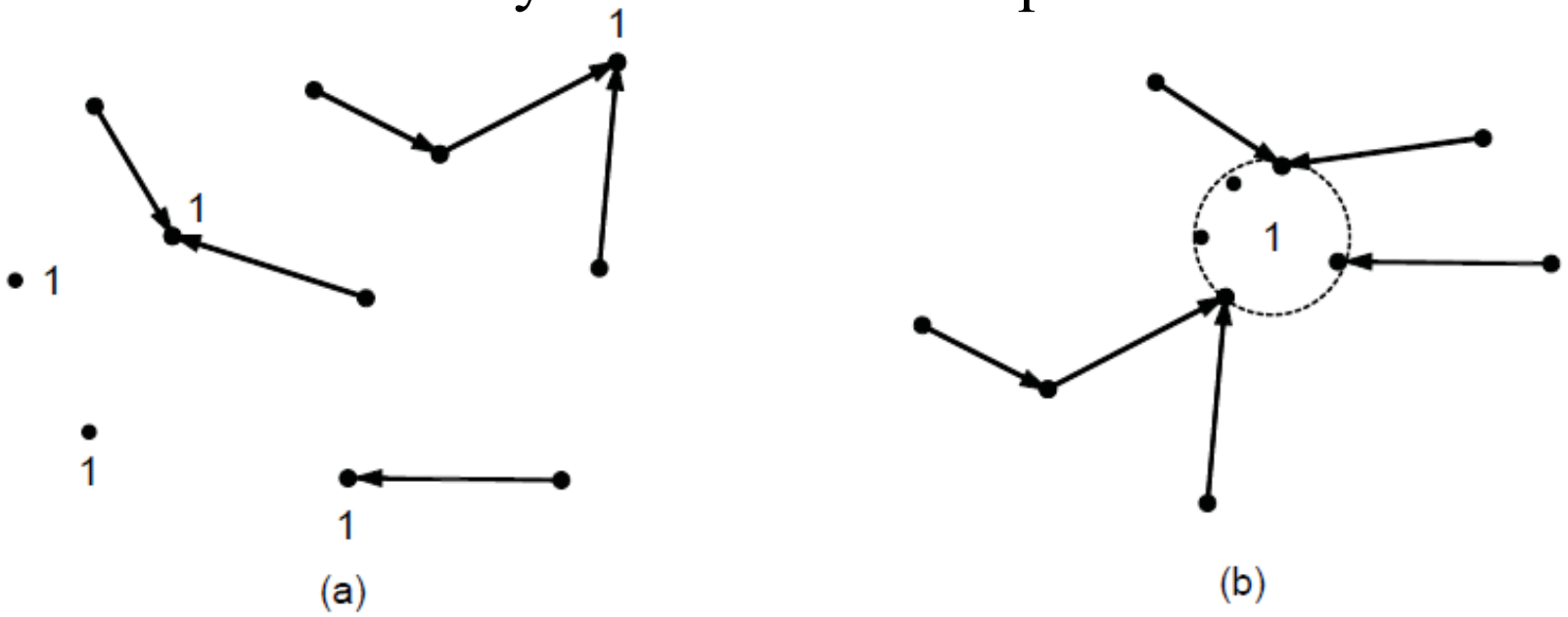
(a)Core-based tree for group 1.

(b)Sending to group 1.

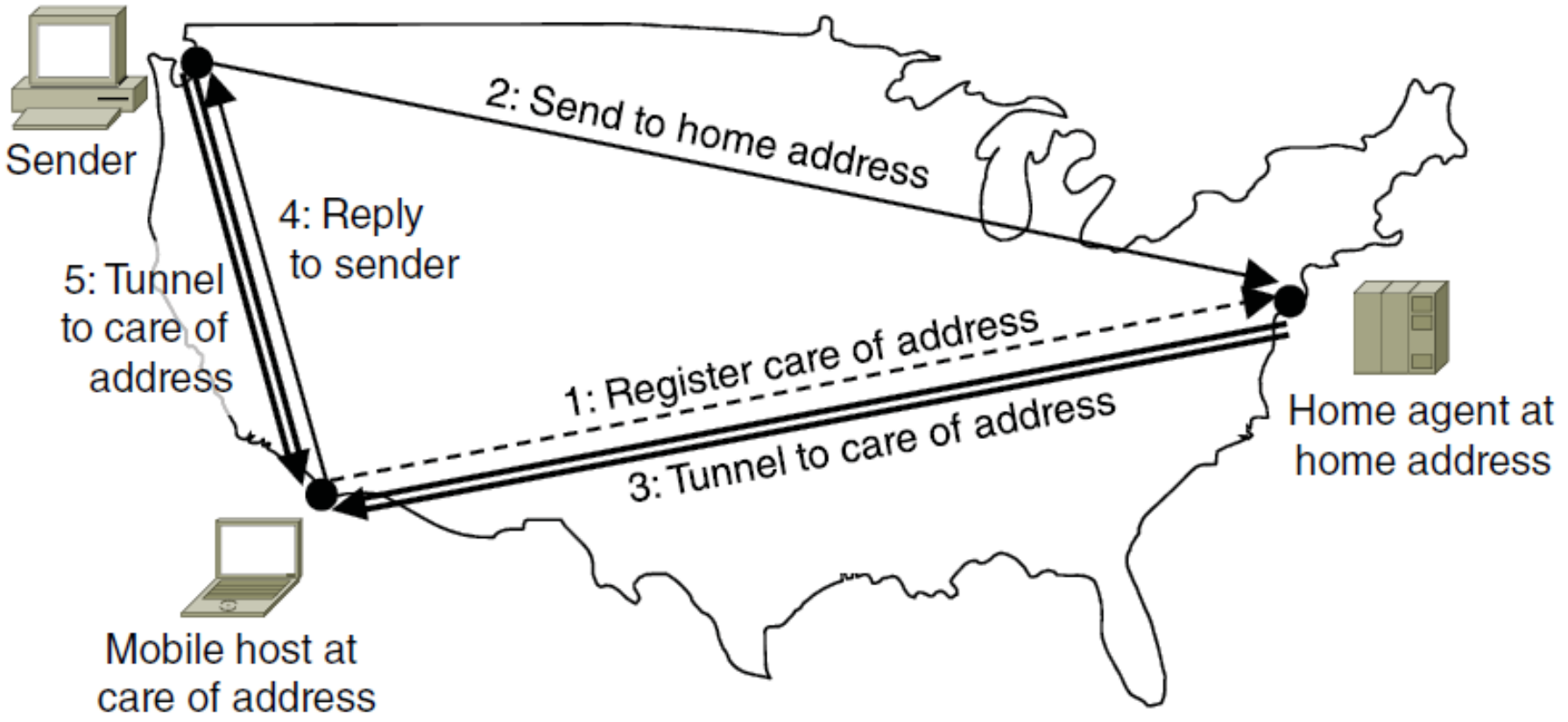


Routing Algorithms: Anycast routing

- **Anycast**
  a packet is delivered to the nearest member of a group. Schemes that find these paths are called anycast routing.
- Example usage: DNS
- Do not require new routing schemes, since DV and LS can produce anycast routes
  - E.g. For DV, all members are given the same address, DV can automatically find the shortest path!

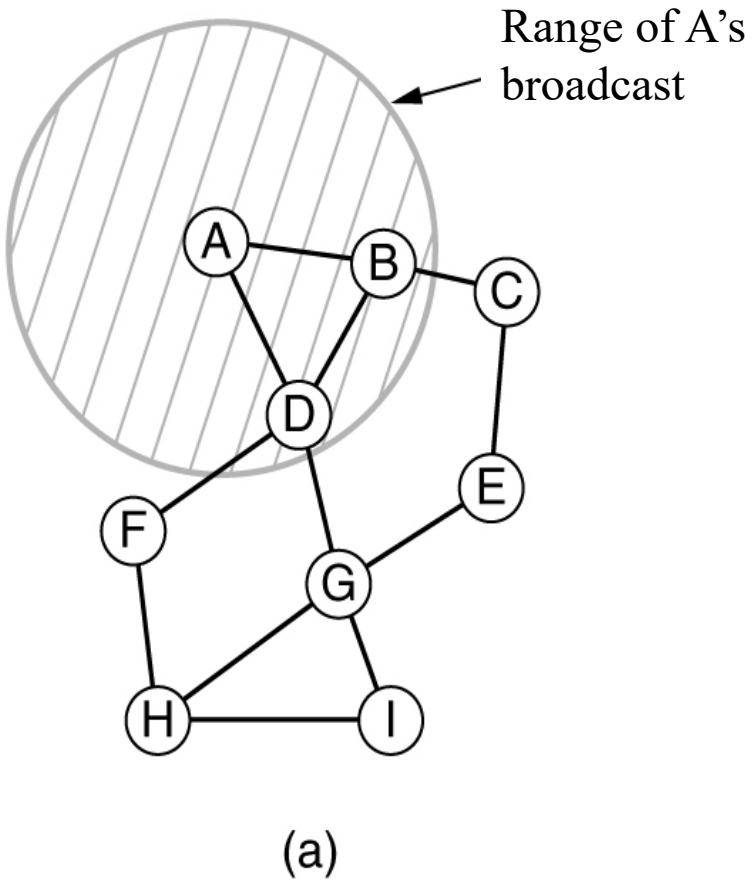


Routing Algorithms: Routing for the mobile hosts



Routing Algorithms:
Routing in Ad Hoc Networks

- Ad Hoc networks or MANET (Mobile Ad Hoc Networks): The network topology may be **changing all the time**
- **AODV**: Ad hoc On-demand Distance Vector. The most popular routing algorithm for Ad Hoc Networks.
- Consider the problem:
  A wants to send to I



Routing in Ad Hoc Networks: Route discovery

(a) A broadcasts, attempting to reach I.

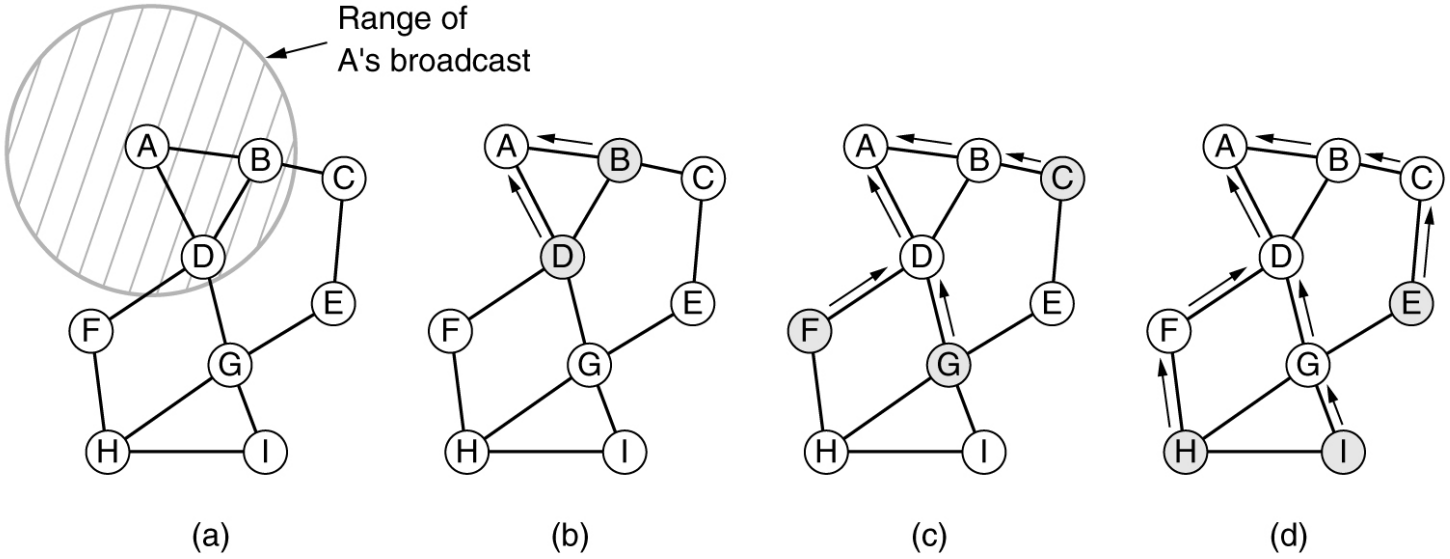
(b) B and D received, establish reverse paths to A.

(c) B and D broadcast. C, F, and G received, establish reverse paths to B and D.

(d) C,F,G broadcast. E, H, and I received (I reached!).

I sends back a REPLY along the reverse path. G and D establish to path to I when they process REPLY.

Shaded nodes are new recipients. Arrows show possible reverse routes.



Routing in Wireless Sensor Networks

- Wireless Sensor Networks
- **CTP**: Collection Tree Protocol is a popular routing protocol for WSN. **RPL** (IPv6 Routing Protocol for LLNs, RFC 6550) is a newer version that inherits many ideas from CTP.
- **Some comments about CTP**
  - Many-to-one: suppose A is the sink. Only needs to build paths to A. → # of elements in DV reduced to 1.
  - Link metric: ETX (Expected Transmission Count)
    ETX is **additive**
  - Adaptive beaconing: large beaconing interval if network is steady.
  - Problems? Sink Down, Count-to-infinity, …

INTERNETWORKING (网络互连)

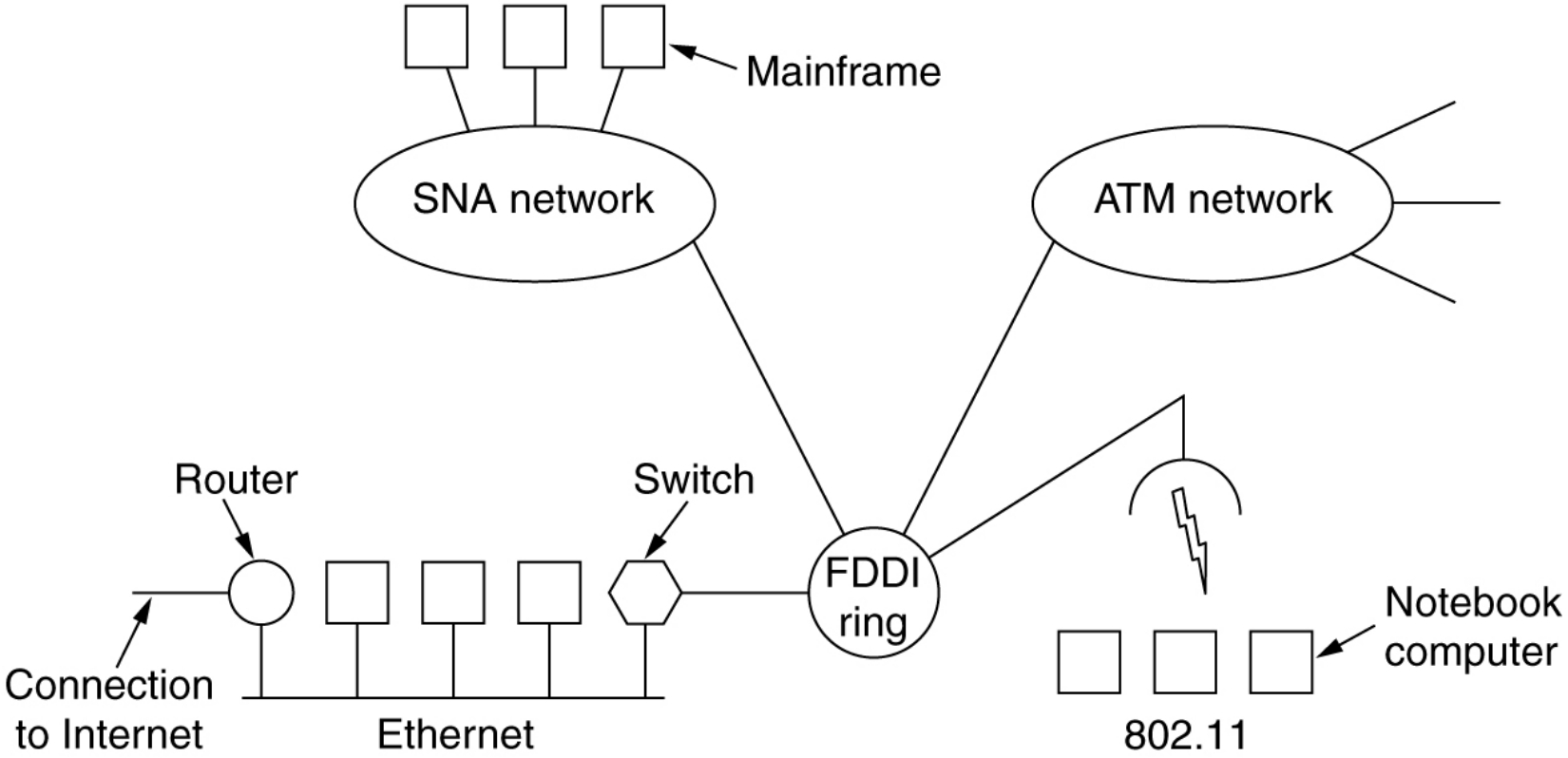
- How Networks Differ
- How Networks Can Be Connected
- Tunneling
- Internetwork Routing
- Packet Fragmentation

Internetworking: Introduction

- A variety of different networks (and thus protocols) will be around
  - The installed base of different networks is large and growing.
  - As computers and networks get cheaper, the place where decisions get made moves downward.
  - Different networks have radically different technology, so it should not be surprising that as new hardware developments occur, new software will be created to fit the new hardware.

Internetworking: Introduction

A collection of interconnected networks.



Internetworking: How Networks Differ

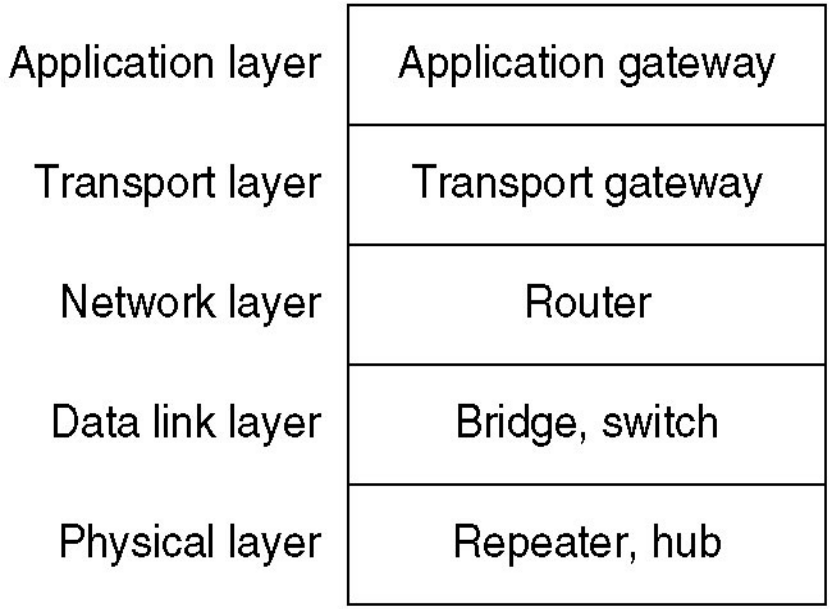
Some of the many ways networks can differ

Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

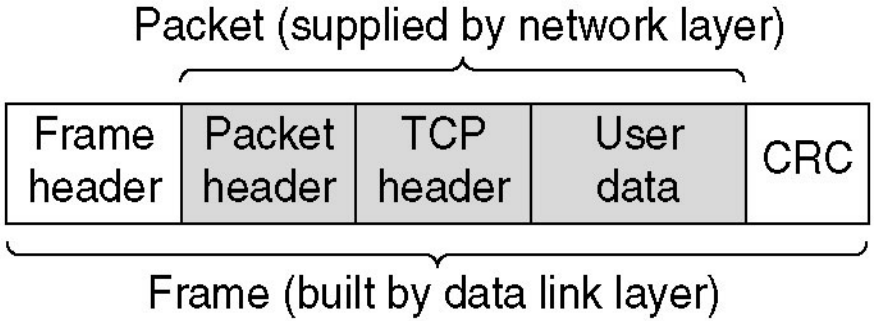
Internetworking: How Networks Can Be Connected

(a) Which device is in which layer.

(b) Frames, packets, and headers.



(a)

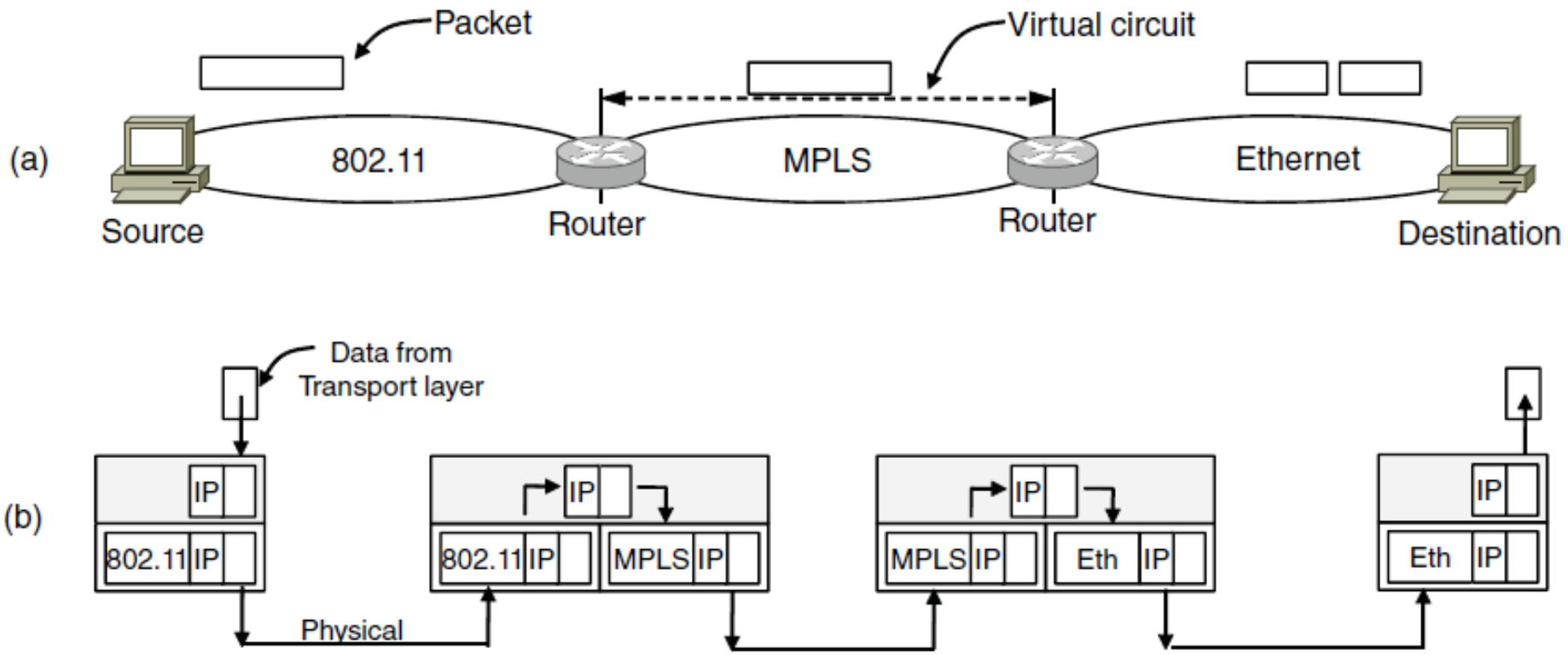


(b)

Internetworking:
How Networks Can Be Connected

(a) A packet crossing different networks.

(b) Network and link layer protocol processing.

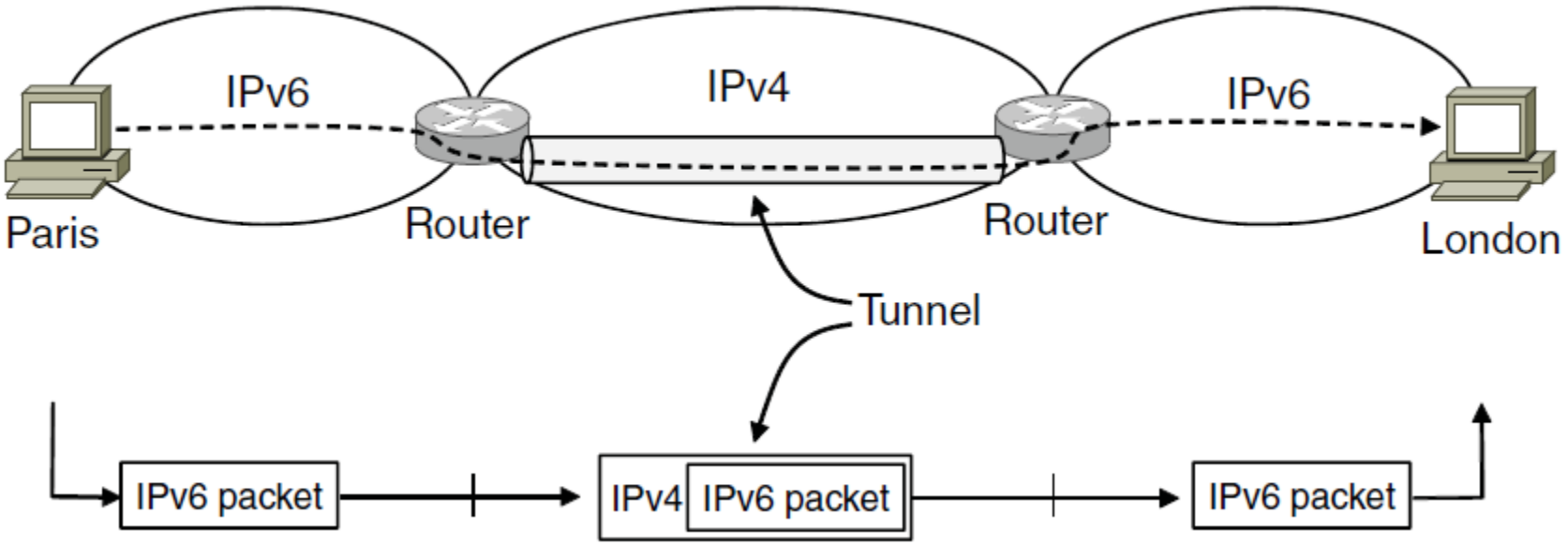


Internetworking: Internetwork Routing

- To route a packet:
  - A typical internet packet starts out on its LAN addressed to the local multiprotocol router.
  - After it gets there, the network layer code decides which multiprotocol router to forward the packet to, using its own routing tables.
    - Direct forwarding using native network protocol
    - Tunneling using the intervening network protocol
  - This process repeats until the packet reaches the desination network.

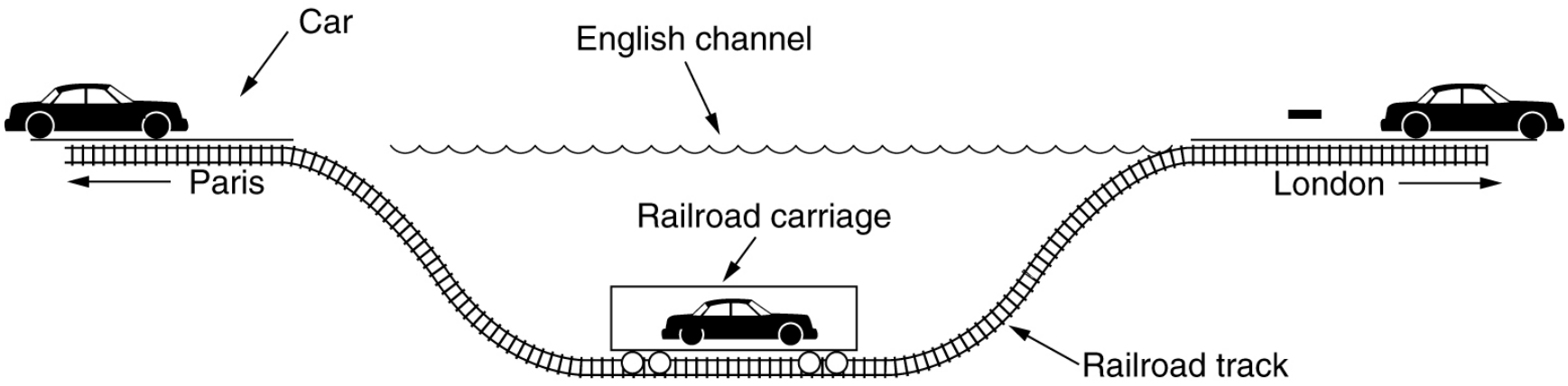
Internetworking: Tunneling (隧道)

Tunneling a packet from Paris to London.



Internetworking: Tunneling

Tunneling a car from France to England.



Internetworking: Internetwork Routing

- Two-level routing
  - Internet (inter-AS) routing: Exterior gateway protocol
  - Intranet (intra-AS) routing: Interior gateway protocol

- **AS** (Autonamous System)
  - each network is operated independently of all the others, it is often referred to as an AS (Autonomous System).
  - A good mental model for an AS is an ISP network.

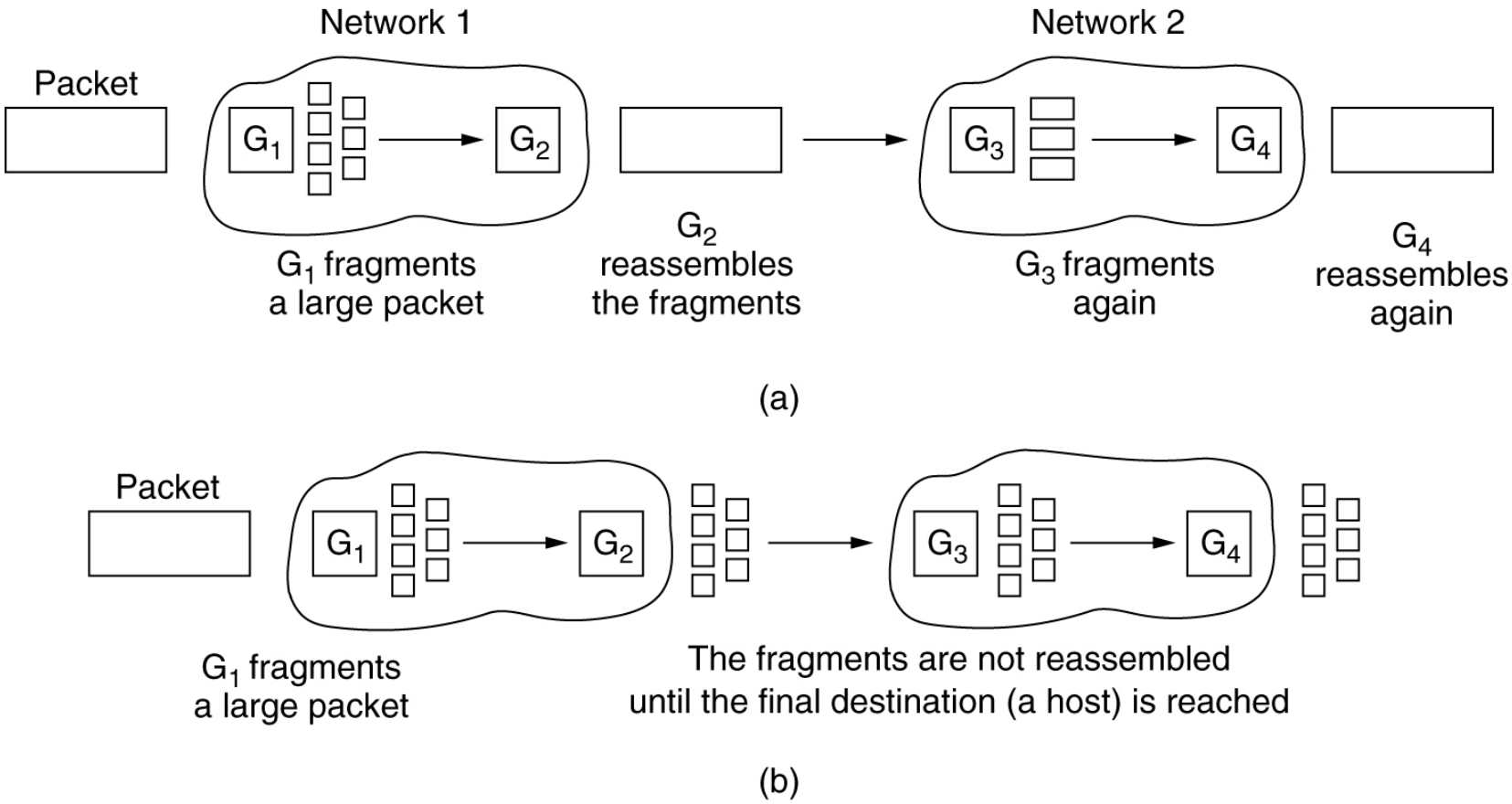
Internetworking: Fragmentation

- Each network imposes some maximum size on its packets. These limits have *various causes*, among them:
  - Hardware (e.g., the width of a TDM transmission slot).
  - Operating system (e.g., all buffers are 512 bytes).
  - Protocols (e.g., the number of bits in the packet length field).
  - Compliance with some (inter)national standard.
  - Desire to reduce error induced retransmissions to some level.
  - Desire to prevent one packet from occupying the channel too long.

Internetworking: Fragmentation

(a) Transparent fragmentation.

(b) Nontransparent fragmentation.



More on transparent

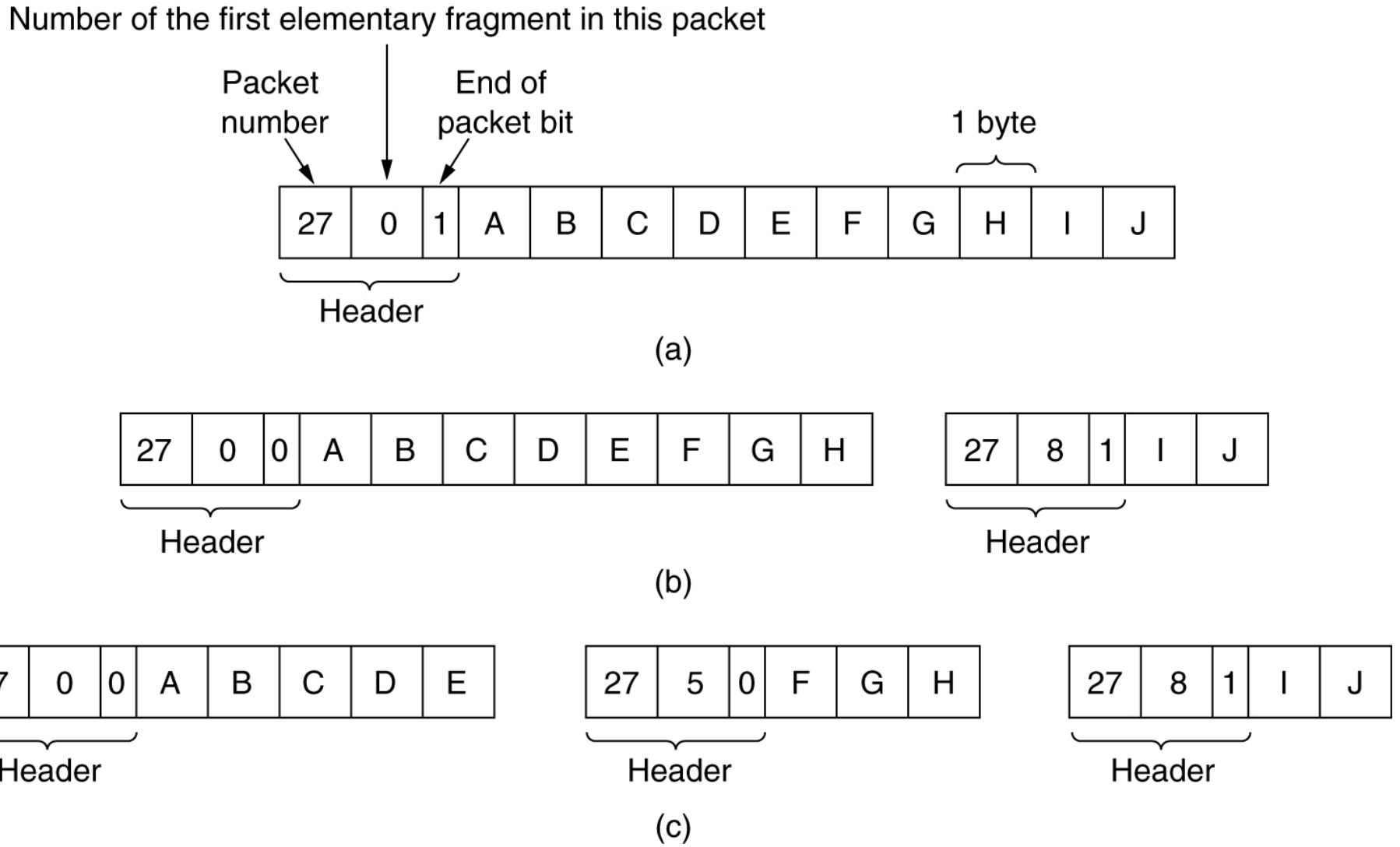
- ***Transparent*** is being used to mean hidden in the sense of things taking place automatically behind scenes (i.e. without user of the code or the program having to interact).
- ***Transparent*** is used where something is present, but you can't see it.

Internetworking: Fragmentation

- When a packet is fragmented, the fragments must be numbered in such a way that the original data stream can be reconstructed.
  - To define an elementary fragment size small enough that the elementary fragment can pass through every network.
  - When a packet is fragmented, all the pieces are multiple of the elementary fragment size.
  - The internet header provide
    - an original packet number
    - the number of the (first) elementary fragment contained in the packet
    - a bit indicating the last piece of the original packet.

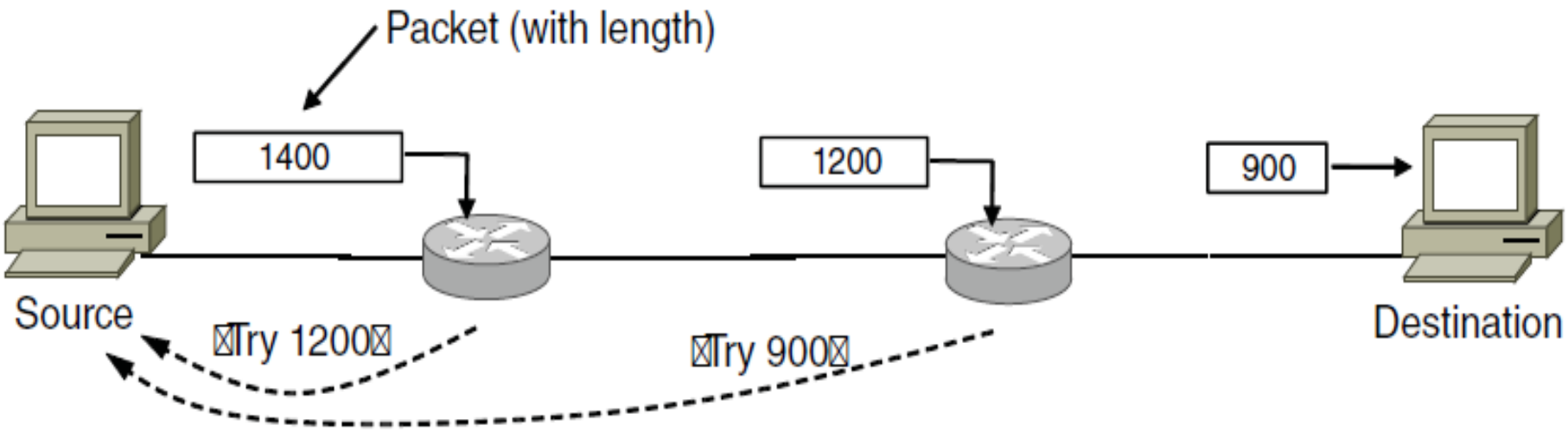
Internetworking: Fragmentation

Fragmentation when the elementary data size is 1 byte.



Internetworking: Fragmentation

The strategy modern Internet adopts: **Path MTU Discovery**



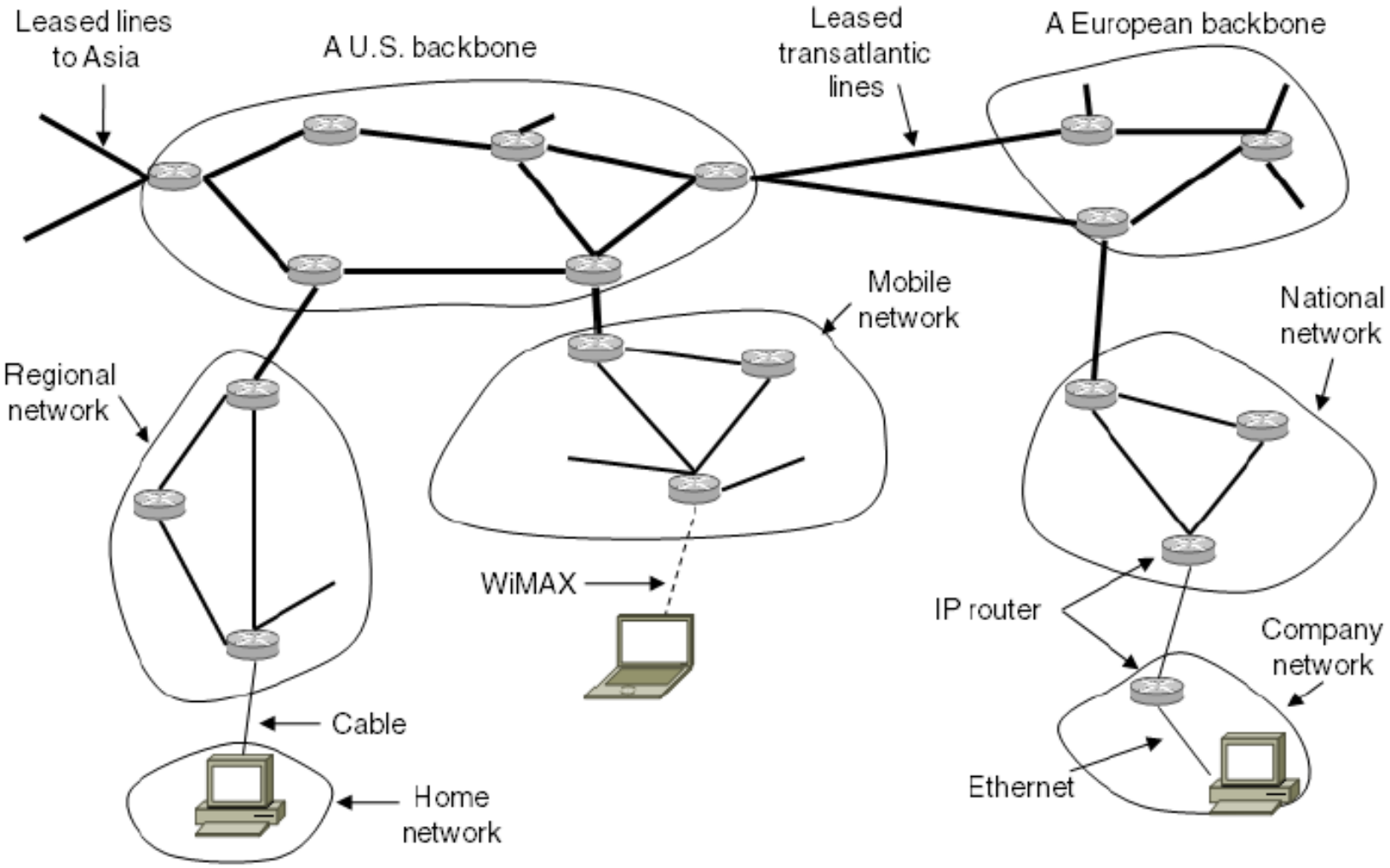
THE NETWORK LAYER IN THE INTERNET

- The IPv4 Protocol
- IP Addresses
- The IPv6 Protocol
- Internet Control Protocols
- OSPF – The Interior Gateway Routing Protocol
- BGP – The Exterior Gateway Routing Protocol
- Internet Multicasting
- Mobile IP

The Network Layer in the Internet:
Top 10 principles for the Internet

1. Make sure it works.
2. Keep it simple.
3. Make clear choices.
4. Exploit modularity.
5. Expect heterogeneity.
6. Avoid static options and parameters.
7. Look for a good design; it need not be perfect.
8. Be strict when sending and tolerant when receiving.
9. Think about scalability.
10. Consider performance and cost.

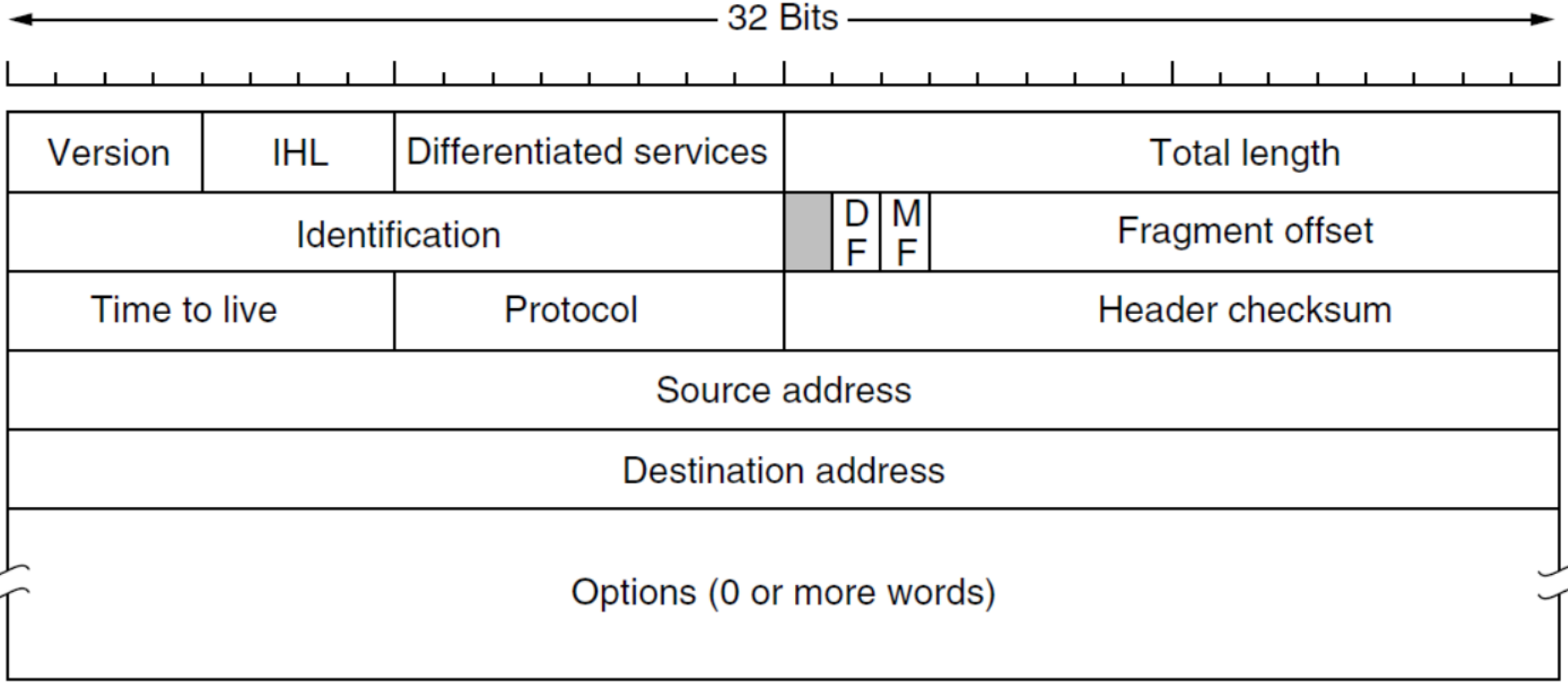
The Network Layer in the Internet:
The Internet: Collections of Subnetworks or ASes



The Network Layer in the Internet: The IPv4 Protocol

An IP datagram consists of a header part and a text part.

The IPv4 (Internet Protocol) header.



The Network Layer in the Internet:
The IPv4 Protocol: Header fields

- **Version**: to keep track of which version of the protocol the datagram belongs to.
- **IHL**: to tell how long the header is, in 32-bit words. 5 <= IHL <= 15.
- **Differentiated service:**
  - **Type of service** (past): 3 for priority, 3 for Delay, Throughput, and Reliability, and 2 unused.
  - **Differentiated services** (now): 6 for service class, 2 for congestion (e.g. ECN).
- **Total length**: the length of **header and data**. The maximum is 65,535 bytes.
- **Identification**: datagram ID. All fragments belong to same ID

The Network Layer in the Internet:
The IPv4 Protocol: Header fields

- **DF**: Don't Fragment
- **MF**: More Fragment. The flag is set for all fragments except the last
- **Fragment offset**: to tell where in the current datagram this fragment belongs
- **Time to Live**: a counter used to limit packet lifetimes.
- **Protocol**: which transport process to give this datagram to. (http://www.iana.org/assignments/protocol-numbers)
- **Header checksum**: to verify the header only
- **Source and destination address**: to indicate the network number and host number.
- **Options** (next slides)

The Network Layer in the Internet:
IP Header Options

Some of the IP options:

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

The Network Layer in the Internet:
IP Addresses

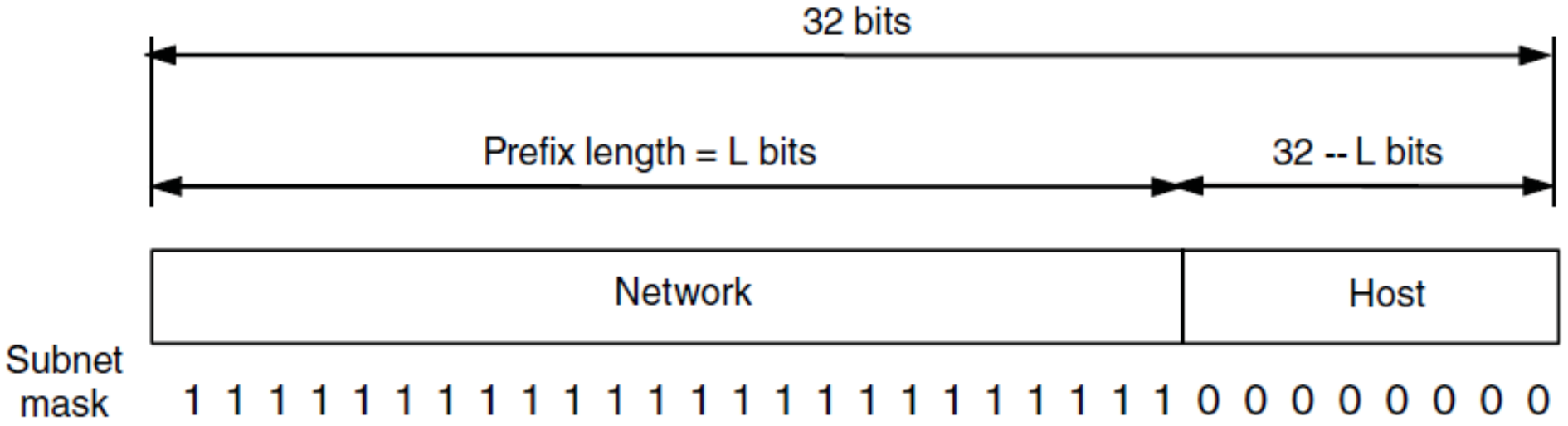
- Every **Internet interface** has an IP address, which encodes its network number and host number. The combination is unique: no two interfaces have the same IP address.

- All IP(v4) addresses are 32 bits long and are used in the source address and destination address fields of IP packets

- IP addressing
  - Prefixes
  - Subnets (division),
  - CIDR (mergement),
  - Classful and Special Addressing
  - NAT

The Network Layer in the Internet:
IP Addresses: Prefixes

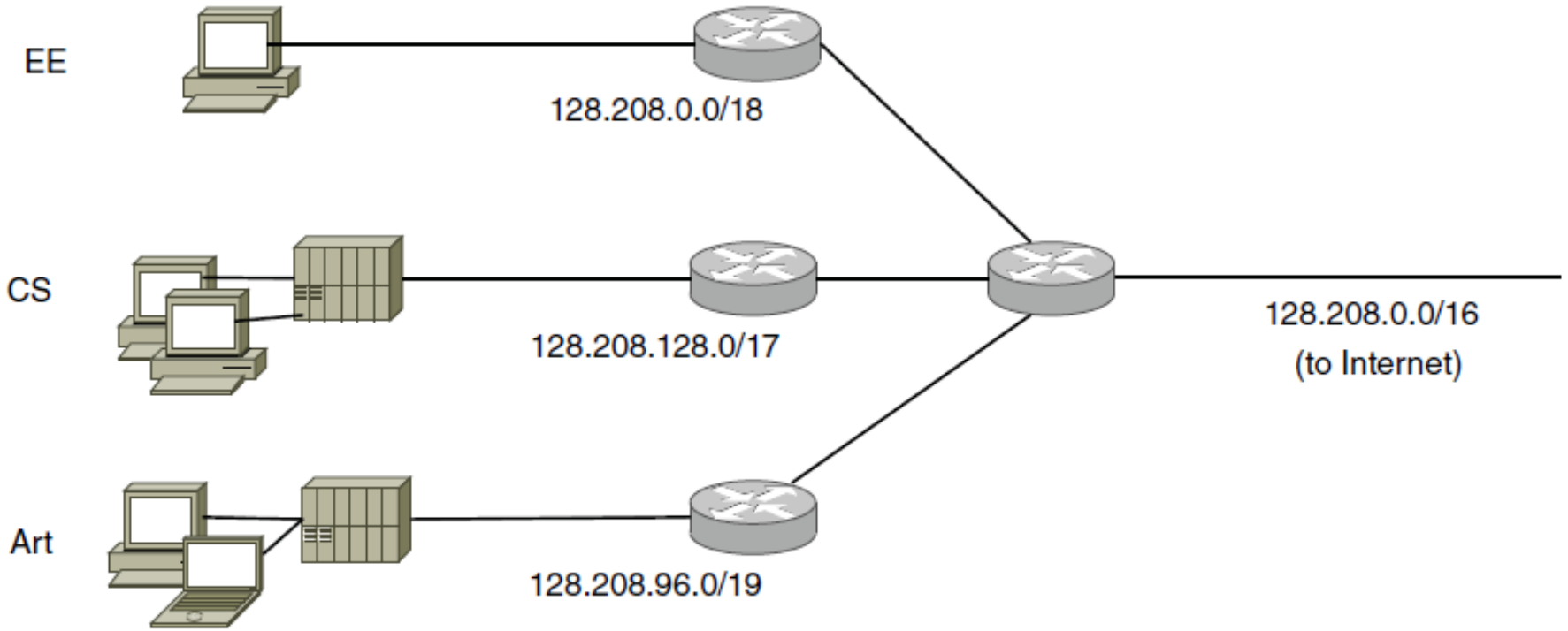
An IP prefix.



The Network Layer in the Internet:
IP Addresses: Prefixes

Splitting an IP prefix into separate networks with subnetting.

Computer Science:	10000000	11010000	1xxxxxxx	xxxxxxxx
Electrical Eng.:	10000000	11010000	00xxxxxx	xxxxxxxx
Art:	10000000	11010000	011xxxxx	xxxxxxxx



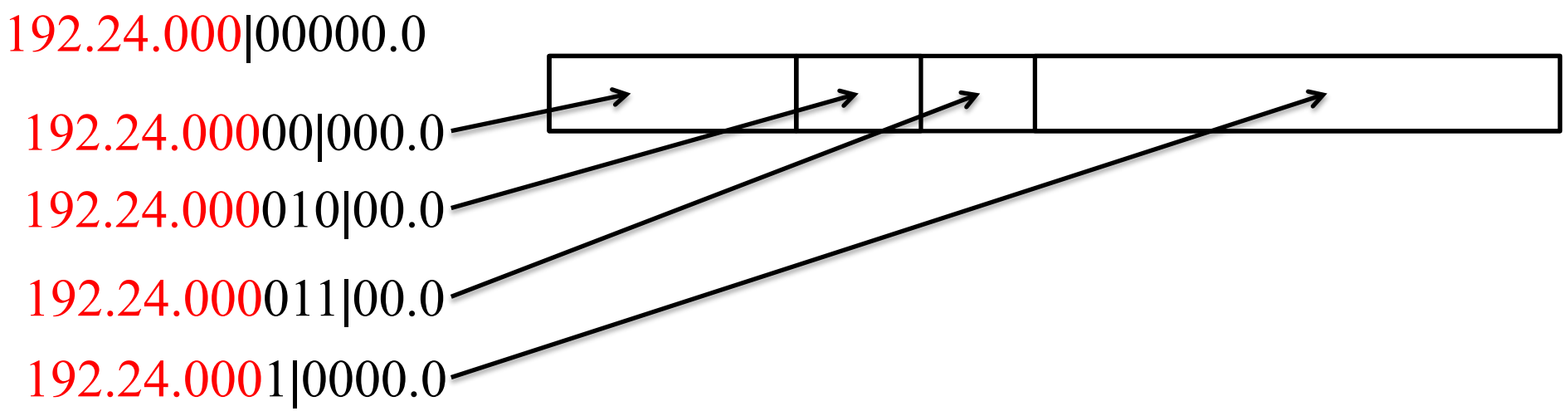
The Network Layer in the Internet:

IP Addresses: Classless Inter-Domain Routing (CIDR)

A set of IP address assignments

Divide the prefix 192.24.0.0/19 ($2^{(32-19)}$=8192 IPs) into:

University	First address	Last address	How many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

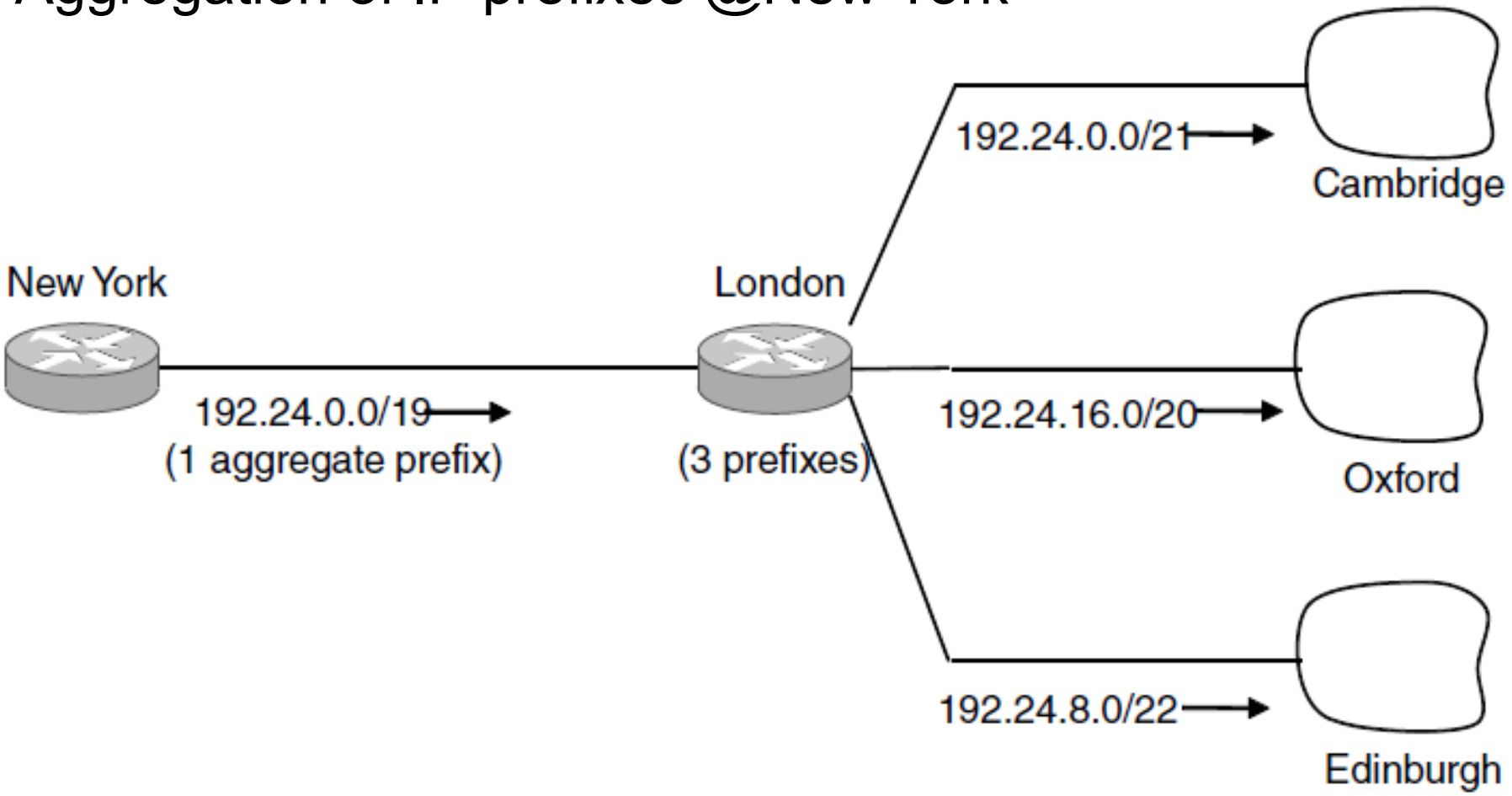


Question

- A large number of consecutive IP addresses are available starting at 202.101.0.0. Suppose that five organizations, A, B, C, D, and E, request 1024, 2000, 2000, 4096 and 512 addresses, respectively, and in that order. Please assign the IP address and the mask in the w.x.y.z/s notation (following the order of allocating the minimum address segment firstly).

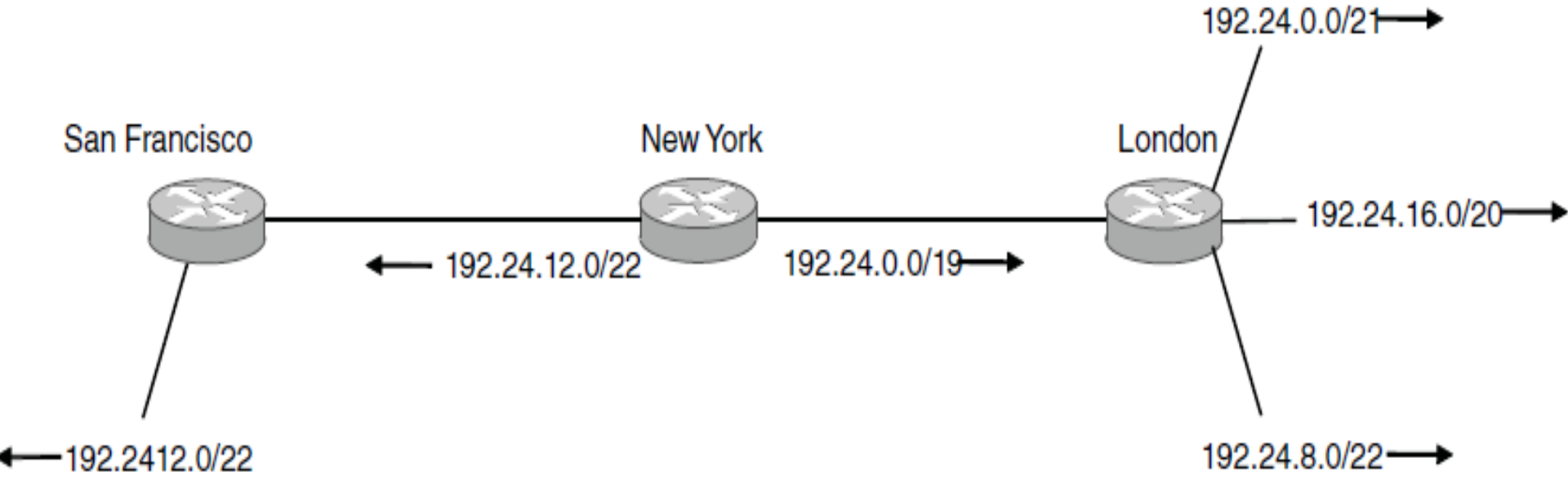
The Network Layer in the Internet:
IP Addresses: CIDR

Aggregation of IP prefixes @New York



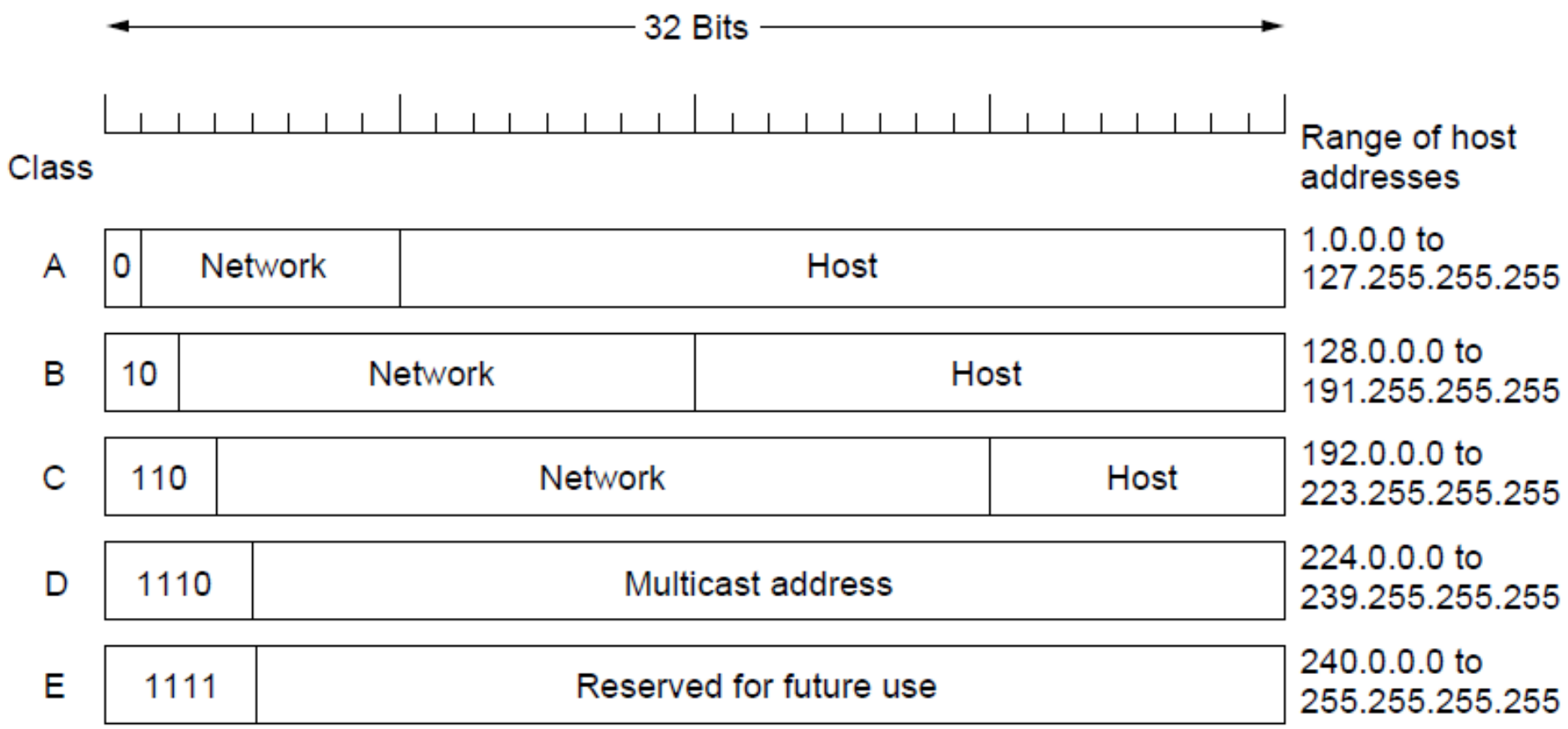
The Network Layer in the Internet:
IP Addresses: CIDR

Longest matching prefix routing @New York router.



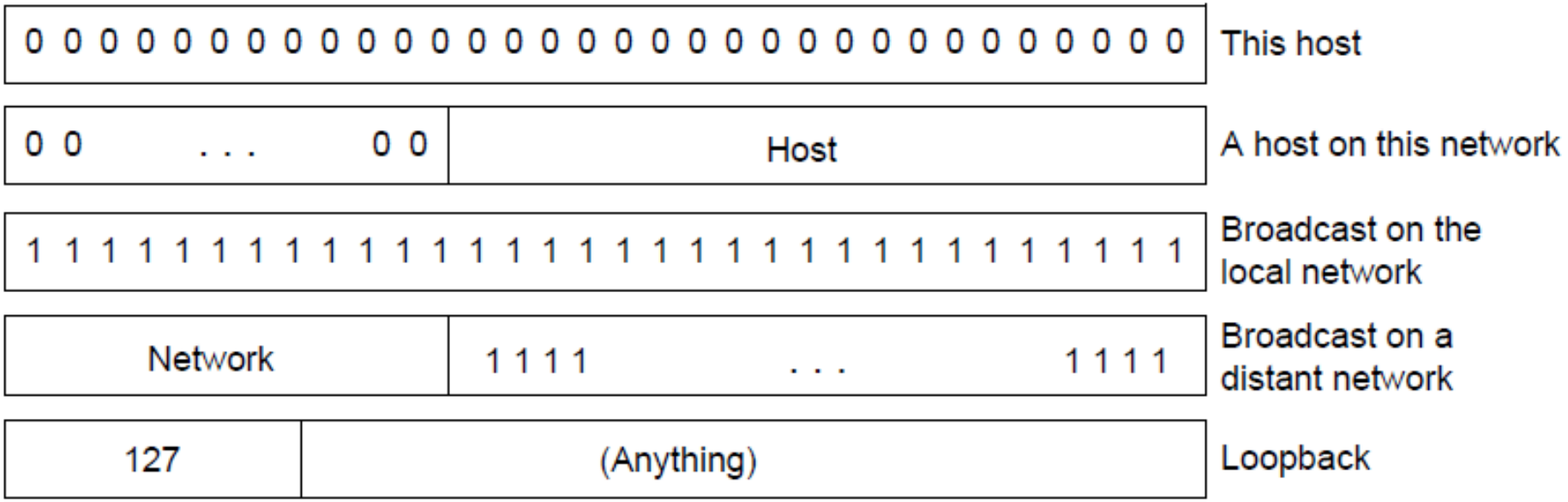
The Network Layer in the Internet:
IP Addresses: Classful and Special Addressing

Classful addressing (past)



The Network Layer in the Internet:
IP Addresses: Classful and Special Addressing

Special IP addresses



The Network Layer in the Internet:
IP Addresses: Network Address Translation (NAT)

- Addressing
  - Permanent addresses: too few
  - Temporary addresses: servers

- Direct addressing (1-level) → indirect addressing (two-level)

- **NAT** (Network Address Translation)
  - To assign each company a single IP address (or at most, a small number of them) for Internet traffic
  - Within the company, every computer gets a unique IP address

The Network Layer in the Internet:
NAT: **reserved private IPs**

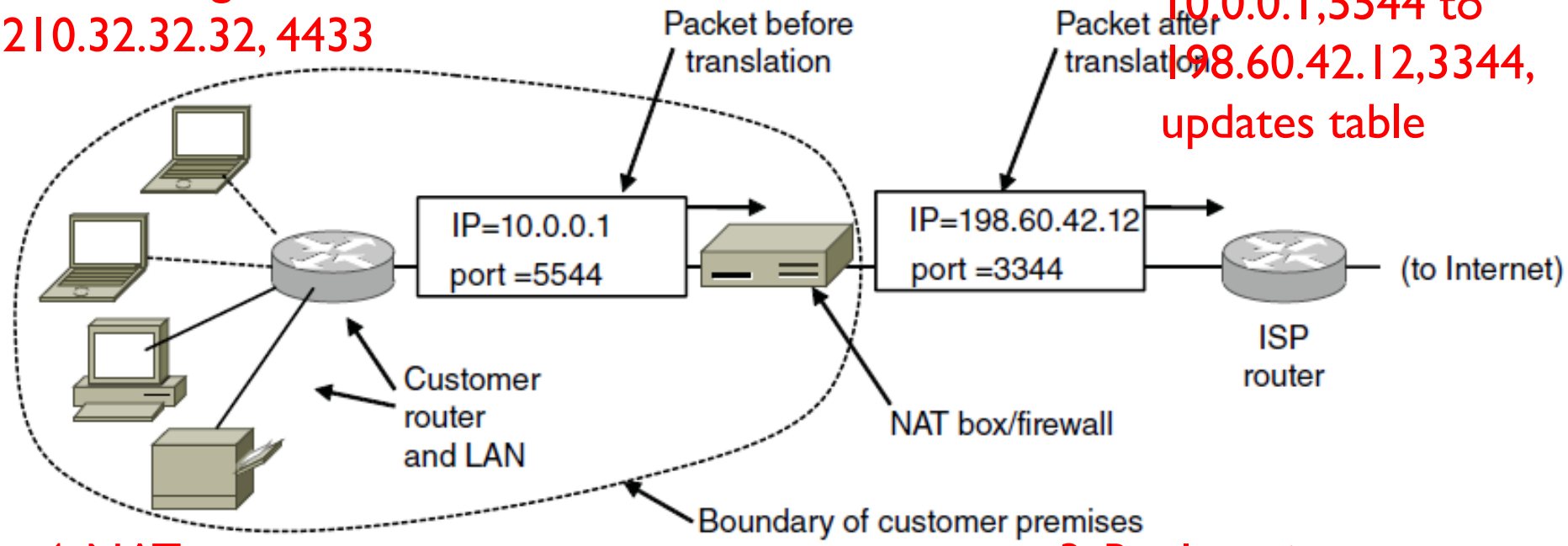
- 10.0.0.0/8
  - **10.0.0.0 ~ 10.255.255.255**, 16,777,216 hosts
- 172.16.0.0/12
  - **172.16.0.0 ~ 172.31.255.255**, 1,048,576 hosts
- 192.168.0.0/16
  - **192.168.0.0 ~ 192.168.255.255**, 65,536 hosts

NAT translation table

LAN side addr	WAN side addr
10.0.0.1,5544	198.60.42.2,3344
……	……

1: host 10.0.0.1 sends datagram to 210.32.32.32, 4433

2: NAT router changes datagram source addr from 10.0.0.1,5544 to 198.60.42.12,3344, updates table



4: NAT router changes datagram dest addr from 198.60.42.12,3344 to 10.0.0.1,5544

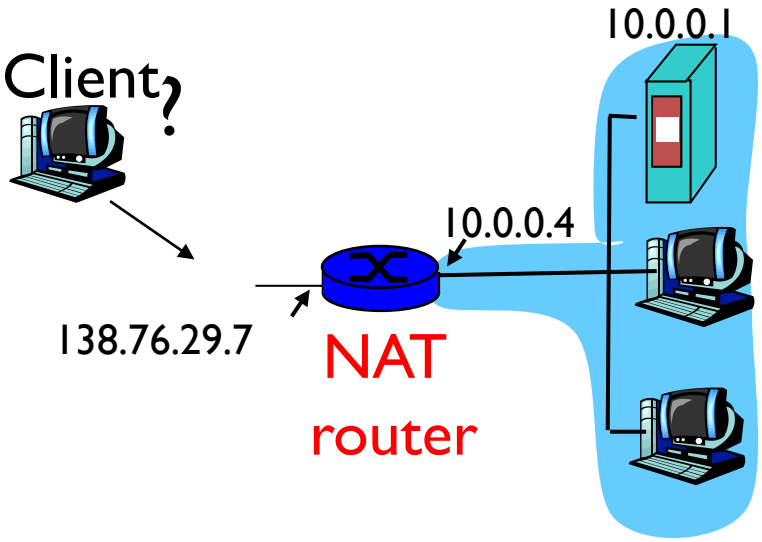
3: Reply arrives dest. address: 198.60.42.12,3344

Drawbacks of NAT

1. NAT violates the architectural model of IP, which states that every IP address uniquely identifies a single machine worldwide
2. NAT breaks the end-to-end connectivity model of the Internet, which says that any host can send a packet to any other host at any time. Since the mapping in the NAT box is set up by outgoing packets, incoming packets cannot be accepted until after outgoing ones (**NAT traversal problem**).
3. NAT changes the Internet from a connectionless network to a peculiar kind of connection-oriented network.
4. NAT violates the most fundamental rule of protocol layering
5. Processes on the Internet are not required to use TCP or UDP.
6. Some applications use multiple TCP/IP connections or UDP ports in prescribed ways
7. Since the TCP *Source port* field is 16 bits, at most 65,536 machines can be mapped onto an IP address.

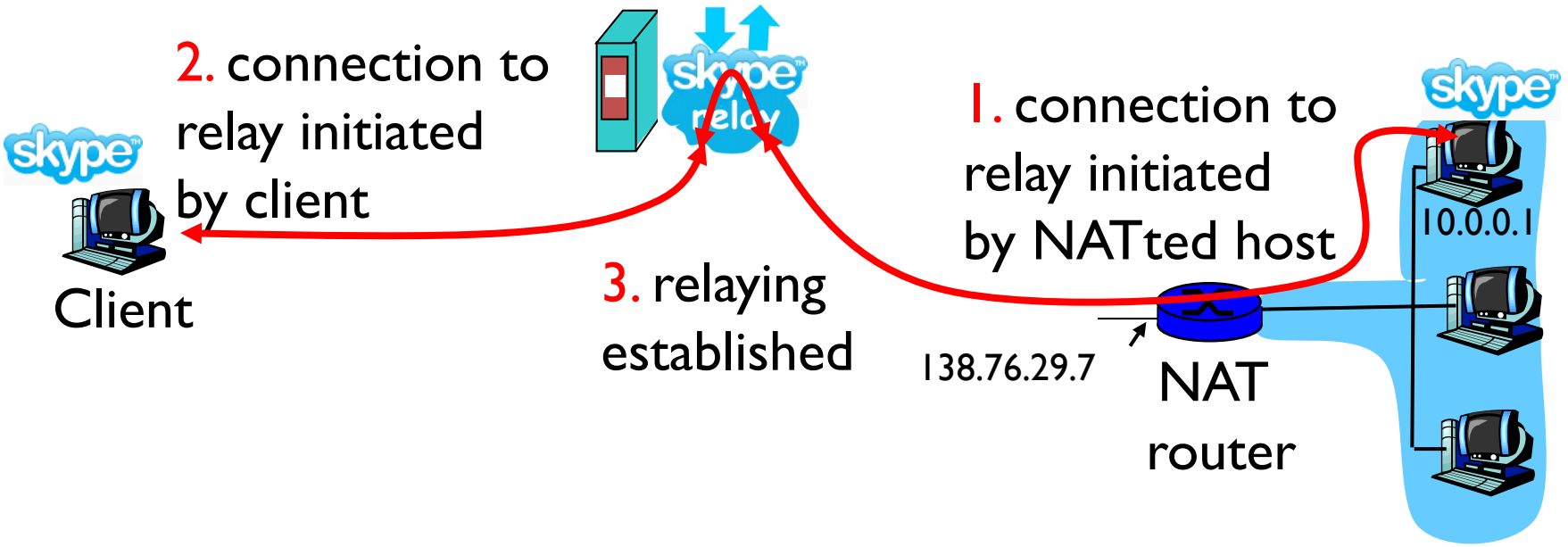
NAT traversal problem

- client wants to connect to server with address 10.0.0.1 (*server in LAN*)
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7



NAT traversal problem

- One possible solution: *relaying* (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between two connections



The Network Layer in the Internet: IPv6

Major goals for IPv6

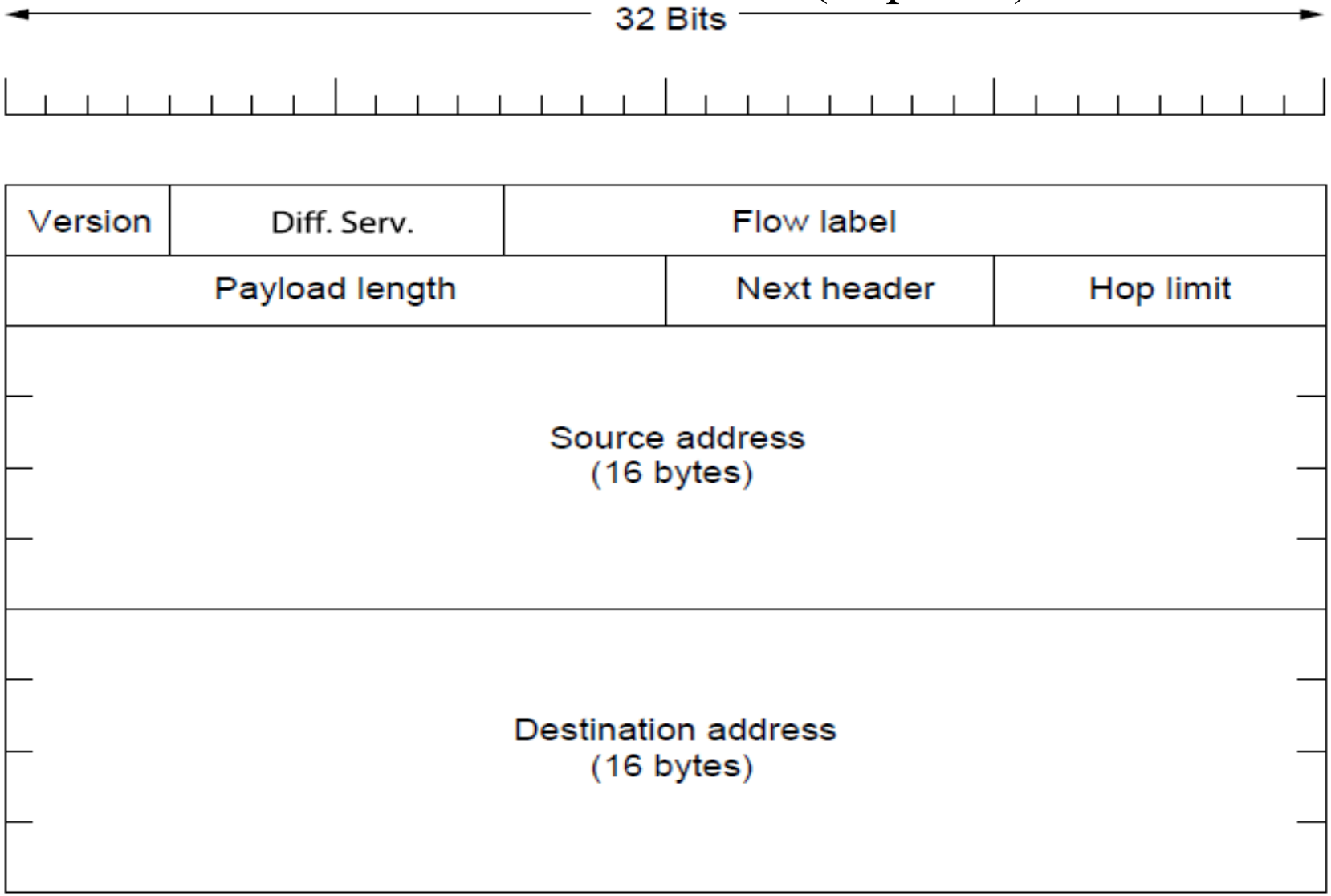
- Support billions of hosts.
- Reduce the size of the routing tables.
- Simplify the protocol, to allow routers to process packets faster.
- Provide better security (authentication and privacy).
- Pay more attention to type of service.
- Aid multicasting by allowing scopes to be specified.
- Make it possible for a host to roam without changing its address.
- Allow the protocol to evolve in the future.
- Permit the old and new protocols to coexist for years

The Network Layer in the Internet: IPv6

- In 1990, IETF started work on a new version of IP.
- IETF issued a call for proposals and discussion in RFC 1550 (IP: Next Generation (IPng) White Paper Solicitation).
- 21 in 1990 → 7 in 1992 → 3 in 1993 → 1
- IPv6 meets the goals fairly well.
  - IPv6 has longer addresses than IPv4. They are **16 bytes long**.
  - The simplification of the header.
  - Better support for options.
  - Security
  - More attention has been paid to type of service than in the past.

The Network Layer in the Internet: IPv6

The IPv6 fixed header (required).



Some fields in IPv6

- **Differentiated services**: used to distinguish the class of service
- **Flow Label**: identify datagrams in same "flow." (concept of "flow" not well defined).
- **Payload length**: tells how many bytes follow the 40-byte header
- **Next header**: tells which of the (currently) six extension headers, if any, follow this one
- **Checksum**: removed entirely to reduce processing time at each hop
- **Options**: allowed, but outside of header, indicated by "Next Header" field

IPv6 notations

- eight groups of four hexadecimal digits with colons between the groups
  - e.g. 8000:0000:0000:0000:0123:4567:89AB:CDEF
- Three optimizations
  - First, leading zeros within a group can be omitted
    - e.g. 0123 can be written as 123
  - Second, one or more groups of 16 zero bits can be replaced by a pair of colons
    - e.g. 8000::123:4567:89AB:CDEF
  - IPv4 addresses can be written as a pair of colons and an old dotted decimal number
    - e.g. ::192.31.20.46

The Network Layer in the Internet: IPv6

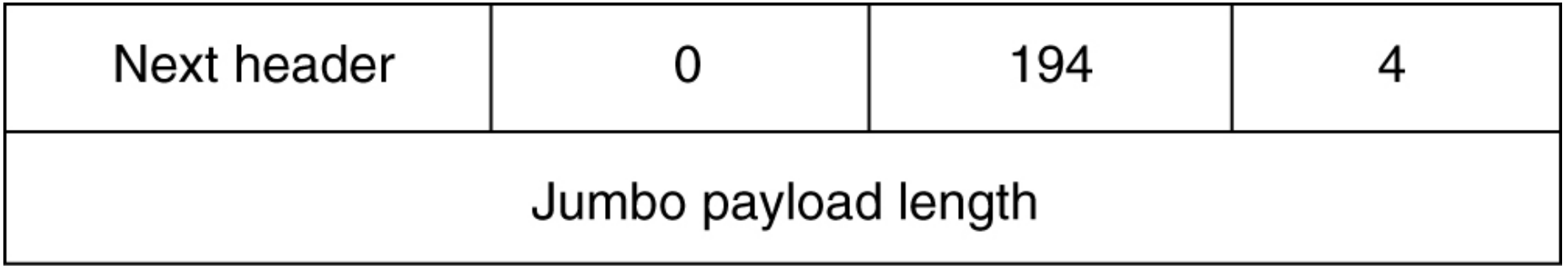
IPv6 extension headers.

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

The Network Layer in the Internet: IPv6

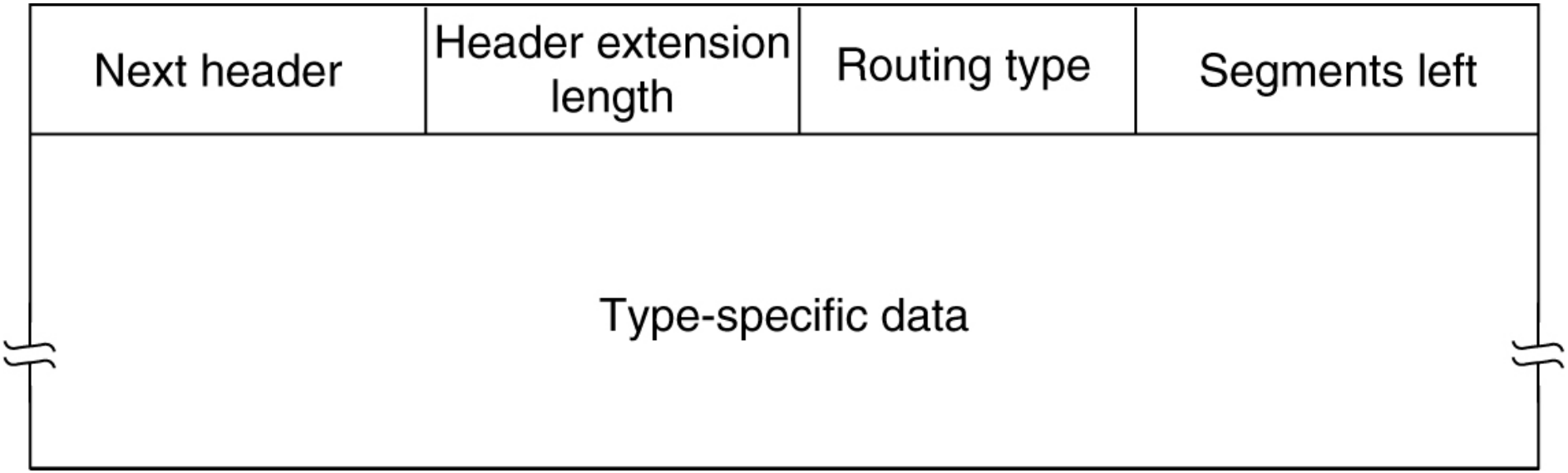
The hop-by-hop extension header for large datagrams (jumbograms)

- **Next header**: a byte telling what kind of header comes next.
- **Next byte (0)**: how long the hop-by-hop header is in bytes, excluding the first 8 bytes
- **Next two bytes (194,4)**: defines the datagram size (code 194) and that the size is a 4-byte number.
- **Jumbo payload length** give the size of the datagram. Sizes less than 65,536 bytes are not permitted and will result in the first router discarding the packet and sending back an ICMP error message.



The Network Layer in the Internet: IPv6

- **The routing header** lists one or more routers that must be visited on the way to the destination.



The Network Layer in the Internet: IPv6

Controversies

- Address length: 8 byte, 16 byte, 20 bytes → 16 bytes
- Hop limit: 8 bits or more → 8 bits
- Maximum packet size: 64 KB or larger → normal 64KB and permit jumbograms.
- Checksum: needed or not → not needed any more.
- Mobile support: yes or no →no but.
- Security: yes or no → no but.

- Huitema, C. 1998. “IPv6: The New Internet Protocol” Prentice-Hall.