《计算机网络原理》

课程编号：40240513

讲课教师：吴建平 徐明伟 尹霞

本科生必修课

计算机科学与技术系

主要教学内容和学时分配

第一章	引言	3
第二章	计算机网络的体系结构	6
第三章	数据通信的基本原理	3
第四章	物理层	3
第五章	数据链路层	9
第六章	局域网和介质访问控制	6
第七章	网络层	6
第八章	传送层	3
第九章	应用层	6
第十章	计算机网络新技术/复习	3
共计		48

第七章 网络层

第一部分

主要内容（1）

7.1 网络层概述

7.2 路由算法

- 7.2.1 最优化原则
- 7.2.2 最短路径路由算法
- 7.2.3 洪泛算法
- 7.2.4 基于流量的路由算法
- 7.2.5 距离向量路由算法
- 7.2.6 链路状态路由算法
- 7.2.7 分层路由
- 7.2.8 移动主机的路由

主要内容（2）

7.3 拥塞控制算法

- 7.3.1 拥塞控制的基本原理
- 7.3.2 拥塞控制算法

7.4 网络互连

- 7.4.1 级联虚电路
- 7.4.2 无连接网络互连
- 7.4.3 隧道技术
- 7.4.4 互联网路由
- 7.4.5 分段
- 7.4.6 防火墙

主要内容（3）

7.5　互联网网络层协议

7.5.1　IPv4和IPv6协议

7.5.2　Internet控制协议

7.5.3　内部网关路由协议：OSPF

7.5.4　外部网关路由协议：BGP

7.6　路由器体系结构和关键技术

7.1 网络层概述（1）

- ISO 定义
  - 网络层为一个网络连接的两个传送实体间交换网络服务数据单元提供功能和规程的方法，它使传送实体独立于路由选择和交换的方式。
- 网络层是处理端到端传输的最低层。
- 网络层要解决的关键问题是通信子网的拓扑结构和路由选择。

7.1 网络层概述（2）

- 网络层设计的有关问题
  - 为传输层提供服务
    - 面向连接服务

      传统电信的观点：通信子网应该提供可靠的、面向连接的服务。
    - 无连接服务

      Internet的观点：通信子网无论怎么设计都是不可靠的，因此网络层只需提供无连接服务。
    - IP/ATM

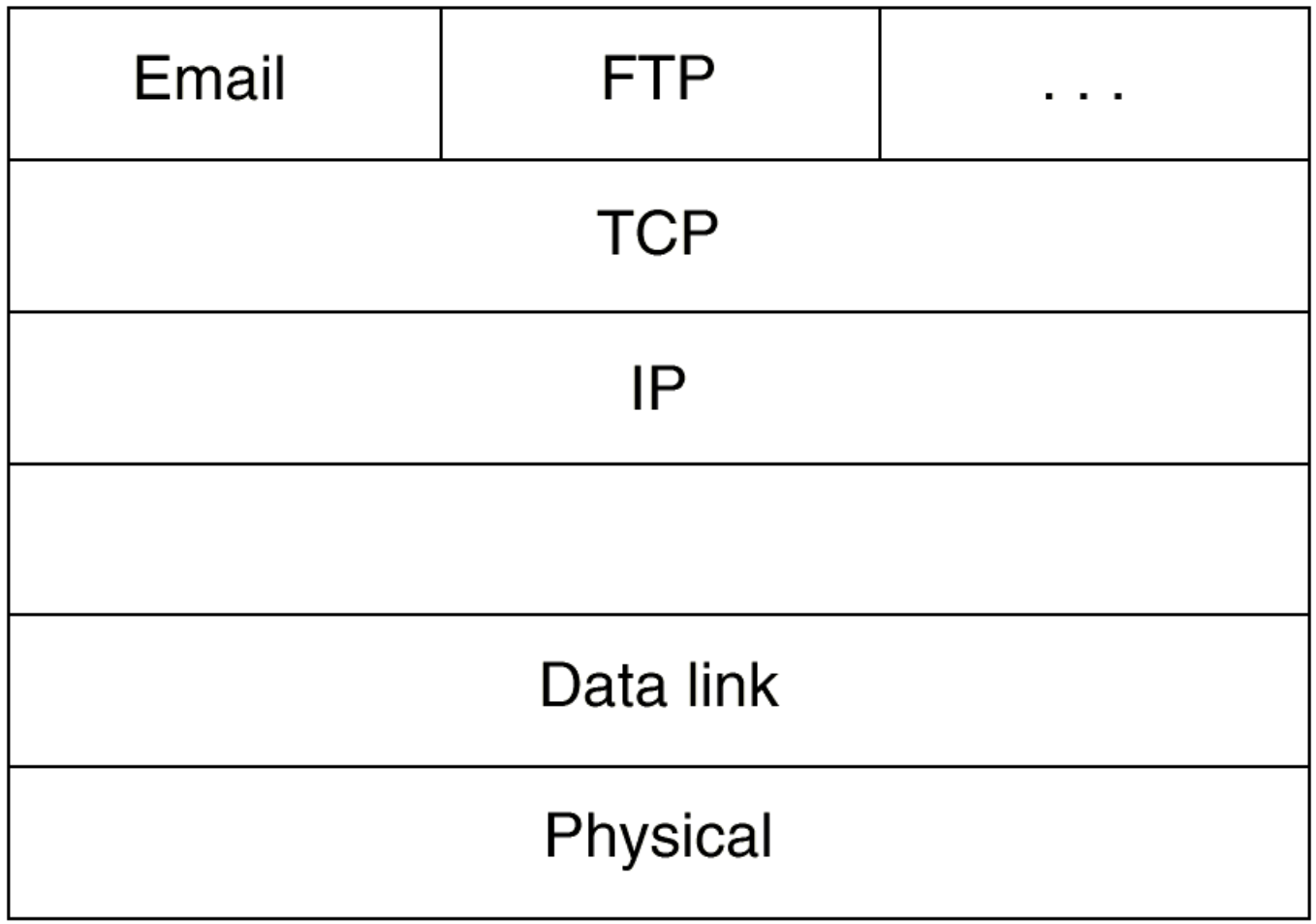
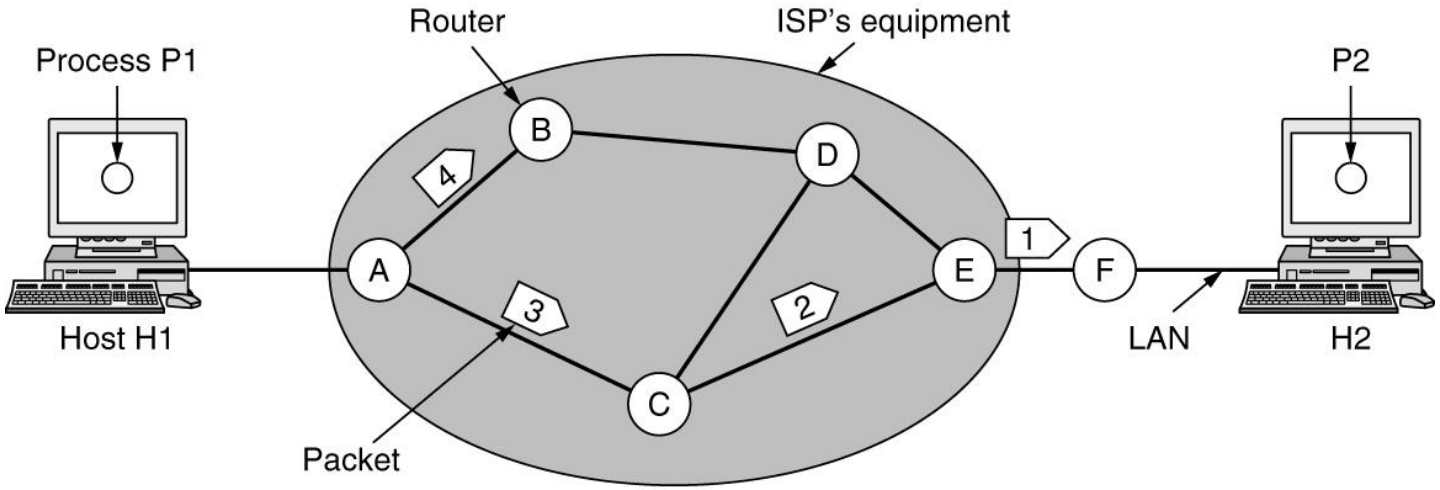


Fig. 5-1. Running TCP/IP over subnet.

7.1 网络层概述（3）

- 网络层的内部组织
  - 虚电路（virtual circuit）
  - 数据报（datagram）
- 虚电路子网与数据报子网的比较
  - 路由器内存空间与带宽的权衡
    - 虚电路方式，路由器需要维护虚电路的状态信息；
    - 数据报方式，每个数据报都携带完整的目的/源地址，浪费带宽

无连接分组交换通信子网



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

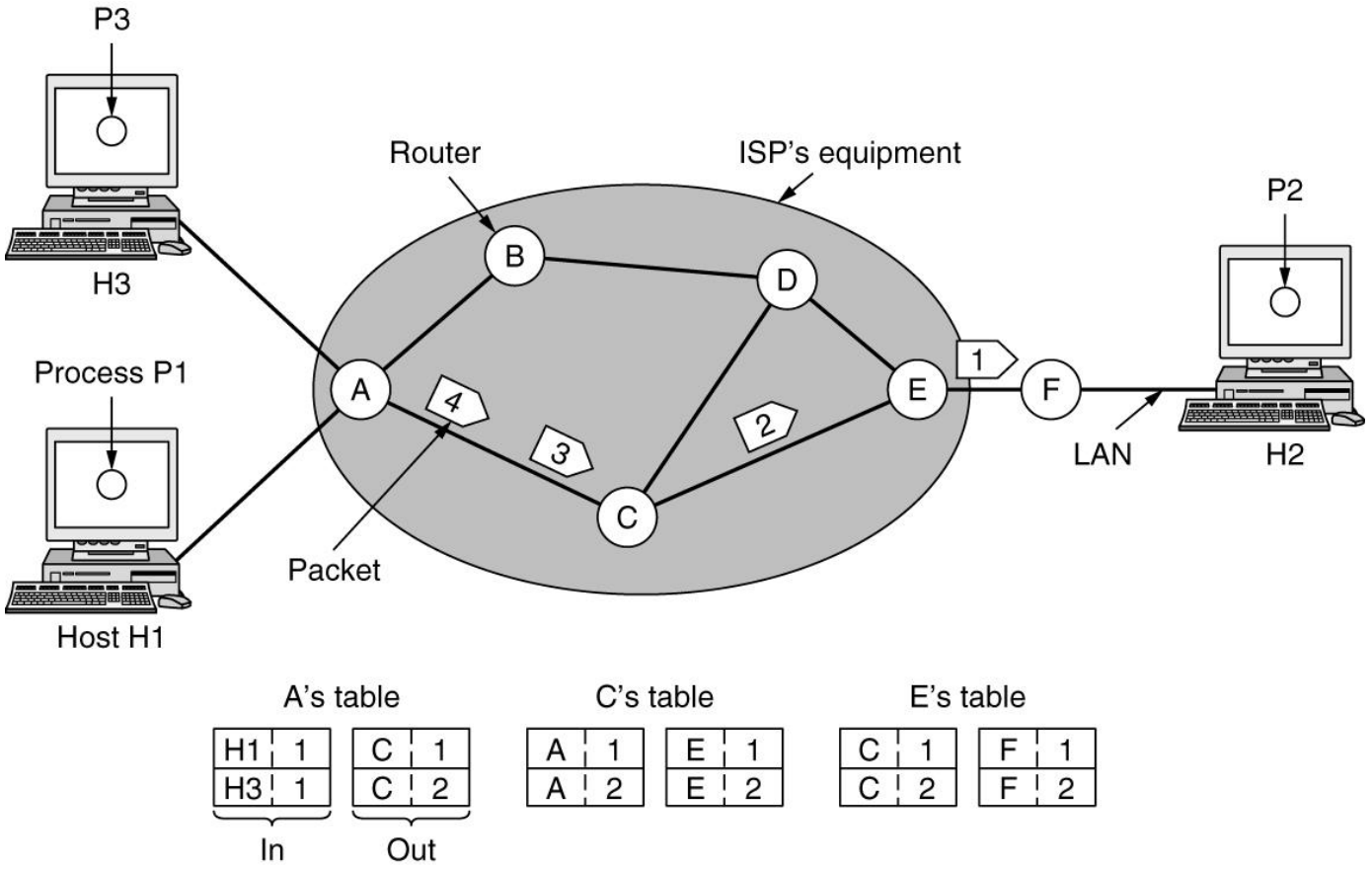
A	A
B	A
C	–
D	E
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

数据报网络内部路由

有连接分组交换通信子网



虚电路网络内部路由

7.1 网络层概述（4）

- 连接建立时间与地址查找时间的权衡
  - 虚电路需要在建立连接时花费时间
  - 数据报则在每次路由时过程复杂
- 服务质量与可靠性的权衡
  - 虚电路方式很容易保证服务质量QoS(Quality of Service)，适用于实时操作，但比较脆弱。
  - 数据报不太容易保证服务质量，但是对于通信线路的故障，适应性很强。

7.1 网络层概述（5）

- 网络层为传送层提供的服务
  - 面向连接服务：将复杂的功能放在网络层(通信子网)。
  - 无连接服务：将复杂的功能放在传输层。
  - 通信子网提供的服务（面向连接或无连接）与通信子网结构（虚电路或数据报）没有必然联系。
  - 服务与子网结构的不同组合的例子

Upper layer	Type of subnet	
	Datagram	Virtual circuit
Connectionless	UDP over IP	UDP over IP over ATM
Connection-oriented	TCP over IP	ATM AAL1 over ATM

Fig. 5-3. Examples of different combinations of service and subne structure.

Issue	Datagram subnet	VC 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	Subnet does not hold state information	Each VC requires subnet table space
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow this route
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Congestion control	Difficult	Easy if enough buffers can be allocated in advance for each VC

Fig. 5-2. Comparison of datagram and virtual circuit subnets.

小结（1）

- 网络层的地位
  - 位于数据链路层和传输层之间，使用数据链路层提供的服务，为传输层提供服务；
  - 通信子网的最高层；
  - 处理端到端传输的最低层。
- 网络层的作用
  - 屏蔽各种不同类型网络之间的差异，实现互连
  - 了解通信子网的拓扑结构，选择路由，实现报文的网络传输

小结（2）

- 网络层的两种实现方式： 数据报和虚电路
  - 都属于分组交换，采用存储转发机制。
  - 数据报(datagram)：每个分组被单独路由，分组带有全网唯一的地址
  - 虚电路(virtual circuit)：先在源端和目的端之间建立一条虚电路，所有分组沿虚电路按次序存储转发，最后拆除虚电路。在虚电路中，每个分组无须进行路径选择。
- 网络层提供的服务
  - 面向连接的服务和无连接的服务。

7.2 路由算法

- 路由算法是网络层软件的一部分
  - 通信子网采用数据报分组交换方式，每个包都要做路由选择；
  - 通信子网采用虚电路分组交换方式，只需在建立连接时做一次路由选择。
- 路由算法应具有的特性
  - 正确性（correctness）,简单性（simplicity）
  - 健壮性（robustness）,稳定性（stability）
  - 公平性（fairness）,最优性（optimality）
- 路由算法分类
  - 非自适应算法，静态路由算法
  - 自适应算法，动态路由算法

7.2.1 最优化原则

- 最优化原则（optimality principle）
  - 如果路由器 J 在路由器 I 到 K 的最优路由上，那么从J 到 K 的最优路由会落在同一路由上。
- 汇集树（sink tree）
  - 从所有的源结点到一个给定的目的结点的最优路由的集合形成了一个以目的结点为根的树，称为汇集树；
  - 路由算法的目的是找出并使用汇集树。

路由节点B的汇集树

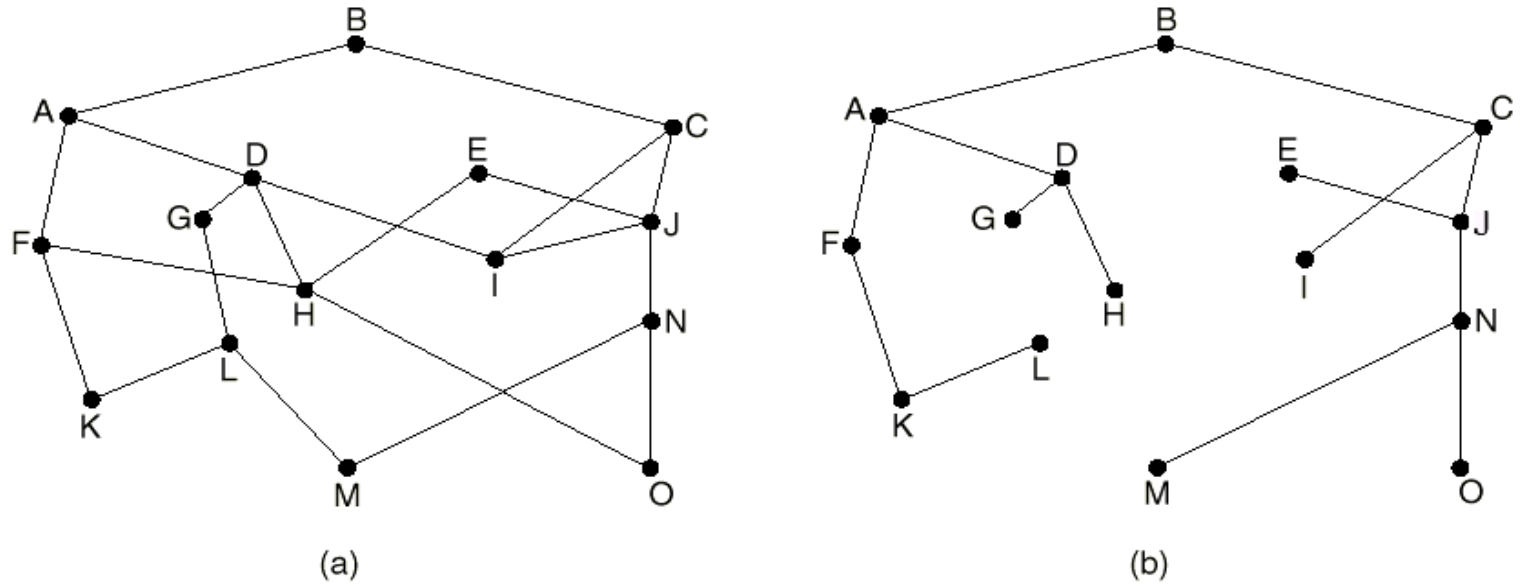


Fig. 5-5. (a) A subnet. (b) A sink tree for router *B*.

7.2.2 最短路径路由算法 (Shortest Path Routing)

- 属于静态路由算法
- 基本思想
  - 构建子网的拓扑图，图中的每个结点代表一个路由器，每条弧代表一条通信线路。为了选择两个路由器间的路由，算法在图中找出最短路径。
- 测量路径长度的方法
  - 结点数量
  - 地理距离
  - 传输延迟
  - 距离、信道带宽等参数的加权函数

Dijkstra（迪杰斯特拉）算法

- 每个结点用从源结点沿已知最佳路径到本结点的距离来标注，标注分为临时性标注和永久性标注
- 初始时，所有结点都为临时性标注，标注为无穷大；
- 将源结点标注为0，且为永久性标注，并令其为工作结点；
- 检查与工作结点相邻的临时性结点，若该结点到工作结点的距离与工作结点的标注之和小于该结点的标注，则用新计算得到的和重新标注该结点
- 在整个图中查找具有最小值的临时性标注结点，将其变为永久性结点，并成为下一轮检查的工作结点；
- 重复第四、五步，直到目的结点成为工作结点。
- 算法实现，程序与算法的区别是：从目的结点开始。

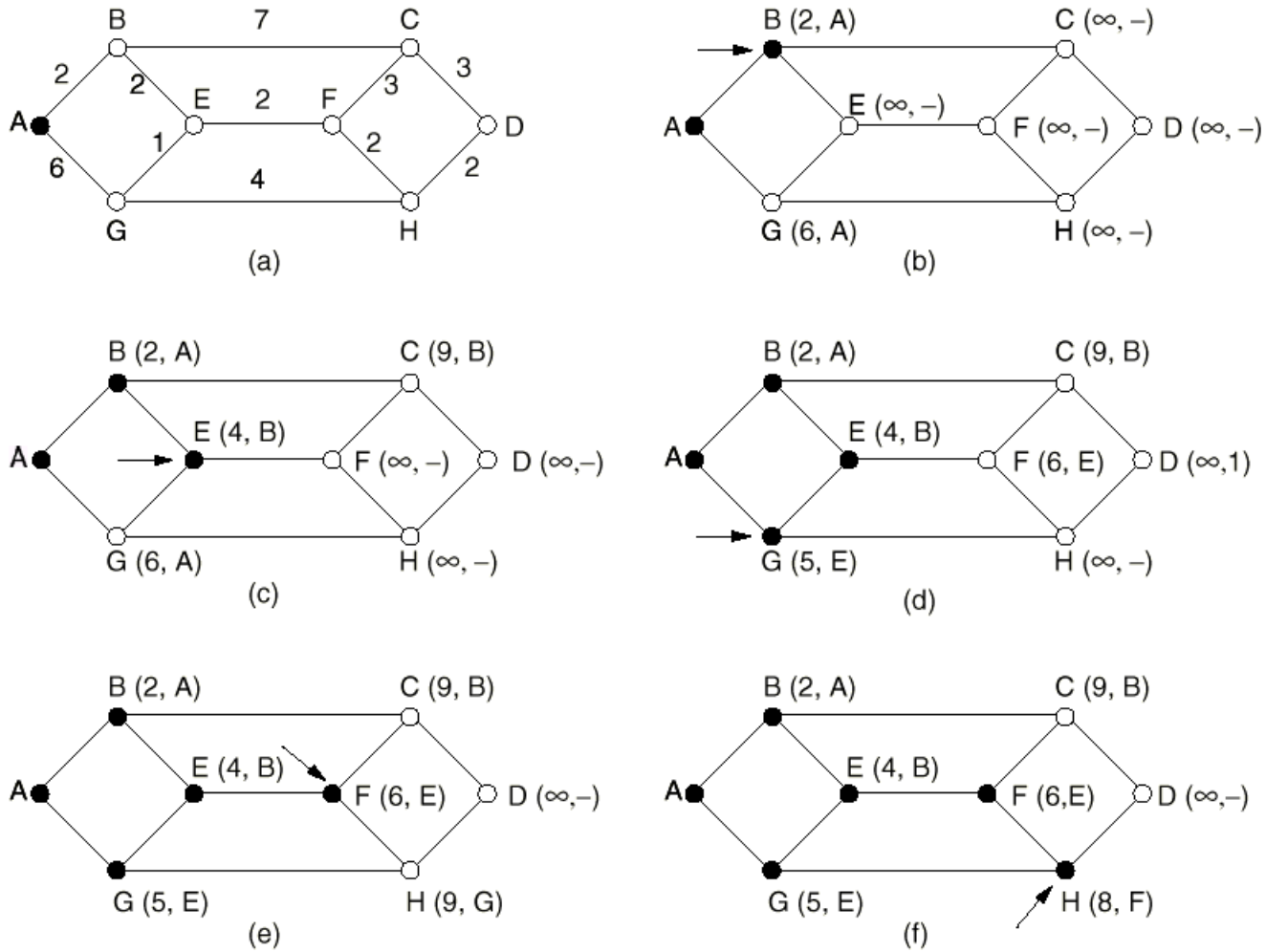


Fig. 5-6. The first five steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

```
#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];
              }
          }

      /* 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);
```

Fig. 5-7. Dijkstra's algorithm to compute the shortest path through a graph.

7.2.3 洪泛算法（Flooding）（1）

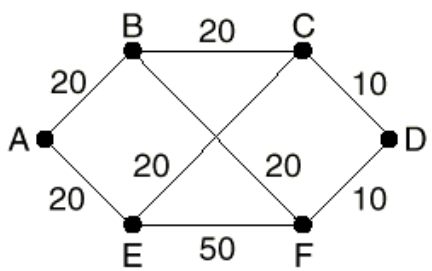
- 属于静态路由算法
- 基本思想
  - 把收到的每一个包，向除了该包到来的线路外的所有输出线路发送。
- 主要问题
  - 洪泛要产生大量重复包。
- 解决措施
  - 每个包头包含站点计数器，每经过一站计数器减1，为0时则丢弃该包；
  - 记录包经过的路径

7.2.3 洪泛算法（Flooding）（2）

- 选择性洪泛算法（selective flooding）
  - 洪泛法的一种改进。将进来的每个包仅发送到与正确方向接近的线路上。
- 应用情况
  - 对路由器和线路的资源过于浪费，实际很少直接采用；
  - 具有极好的健壮性，可用于军事应用；
  - 作为衡量标准评价其它路由算法。

7.2.4 基于流量的路由算法 (Flow-Based Routing)

- 属于静态路由算法
- 基本思想
  - 既考虑拓扑结构，又兼顾网络负荷；
  - 前提：每对结点间平均数据流是相对稳定和可预测的；
  - 根据网络带宽和平均流量，可得出平均包延迟，因此路由选择问题归结为找产生网络最小延迟的路由选择算法。
  - 提前离线（off-line）计算。
- 需要预知的信息
  - 网络拓扑结构；
  - 通信量矩阵F_{ij}；
  - 线路带宽矩阵C_{ij}；
  - 路由算法（可能是临时的）。



(a)

Source \ Destination	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	3 BFD	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

(b)

Fig. 5-8. (a) A subnet with line capacities shown in kbps. (b) The traffic in packets/sec and the routing matrix.

i	Line	λ_i (pkts/sec)	C_i (kbps)	μC_i (pkts/sec)	T_i (msec)	Weight
1	AB	14	20	25	91	0.171
2	BC	12	20	25	77	0.146
3	CD	6	10	12.5	154	0.073
4	AE	11	20	25	71	0.134
5	EF	13	50	62.5	20	0.159
6	FD	8	10	12.5	222	0.098
7	BF	10	20	25	67	0.122
8	EC	8	20	25	59	0.098

Fig. 5-9. Analysis of the subnet of Fig. 5-8 using a mean packet size of 800 bits. The reverse traffic (*BA*, *CB*, etc.) is the same as the forward traffic.

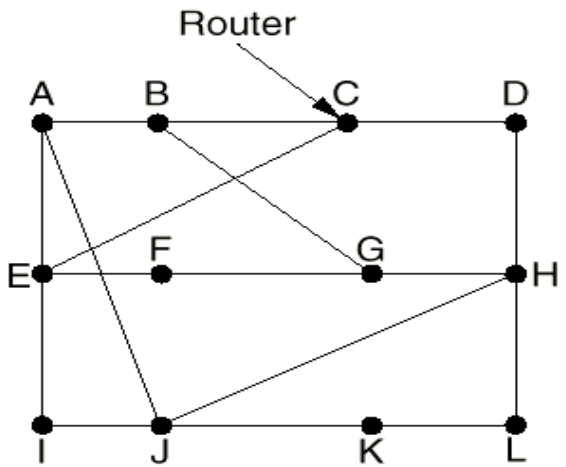
- $1/\mu$ = 800 bits
- 根据排队论，平均延迟 $T = 1/(\mu C - \lambda)$

7.2.5 距离向量路由算法（1）
(Distance Vector Routing)

- 属于动态路由算法
  - 也称Bellman-Ford路由算法和Ford-Fulkerson算法
- 最初用于ARPANET，被RIP协议采用。
- 基本思想
  - 每个路由器维护一张表，表中给出了到每个目的地的已知最佳距离和线路，并通过与**相邻路由器**交换距离信息来更新表；
  - 以子网中其它路由器为表的索引，表项包括两部分：到达目的结点的最佳输出线路，和到达目的结点所需时间或距离；

7.2.5 距离向量路由算法（2）

- 每隔一段时间，路由器向所有邻居结点发送它到每个目的结点的距离表，同时它也接收每个邻居结点发来的距离表；
- 邻居结点X发来的表中，X到路由器i的距离为X_i，本路由器到X的距离为m，则路由器经过X到i的距离为$X_i + m$。根据不同邻居发来的信息，计算$X_i + m$，并取最小值，更新本路由器的路由表；
- 注意：本路由器中的老路由表在计算中不被使用



(a)

New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K

JA delay is 8 | JI delay is 10 | JH delay is 12 | JK delay is 6

Vectors received from J's four neighbors

New routing table for J

(b)

Fig. 5-10. (a) A subnet. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

7.2.5 距离向量路由算法（3）

- 无限计算问题
  - 算法的缺陷：对好消息反应迅速,对坏消息反应迟钝;

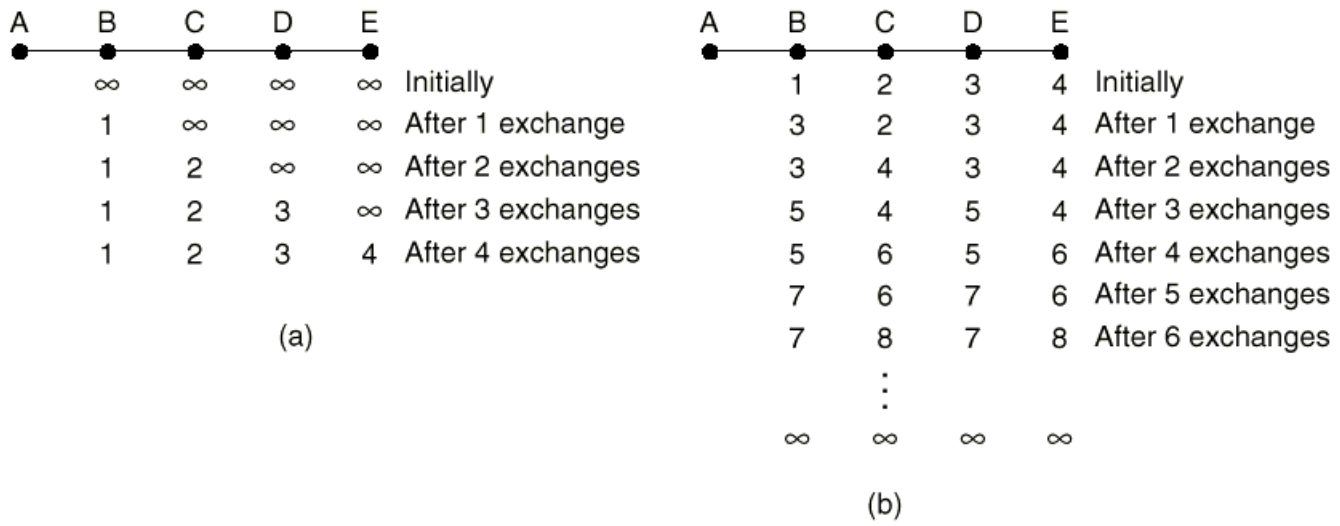


Fig. 5-11. The count-to-infinity problem.

7.2.6 链路状态路由算法（1）

- 距离向量路由算法的主要问题
  - 选择路由时，没有考虑线路带宽；
  - 路由收敛速度慢。
- 链路状态路由算法
  - 发现邻居结点，并学习它们的网络地址；
    - 路由器启动后，通过发送HELLO包发现邻居结点；
    - 两个或多个路由器连在一个LAN时,引入人工结点；

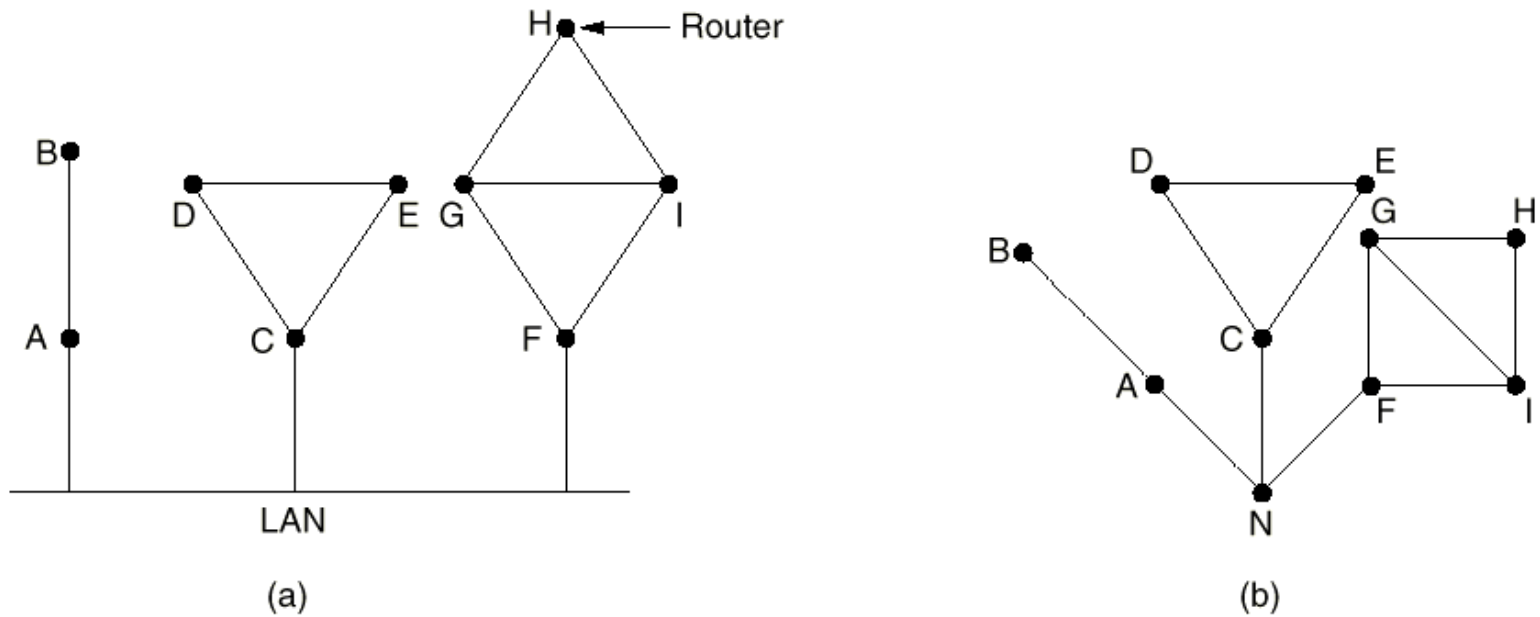


Fig. 5-13. (a) Nine routers and a LAN. (b) A graph model of (a).

7.2.6 链路状态路由算法（2）

- 测量到每个邻居结点的延迟或开销；
  - 一种直接的方法是：发送一个要对方立即响应的ECHO包，来回时间除以2即为延迟。
- 将所有学习到的内容封装成一个包；
  - 包以发送方的标识符开头，后面是序号、年龄和一个邻居结点列表；
  - 列表中对应每个邻居结点，都有发送方到它们的延迟或开销；
  - 链路状态包定期创建或发生重大事件时创建。

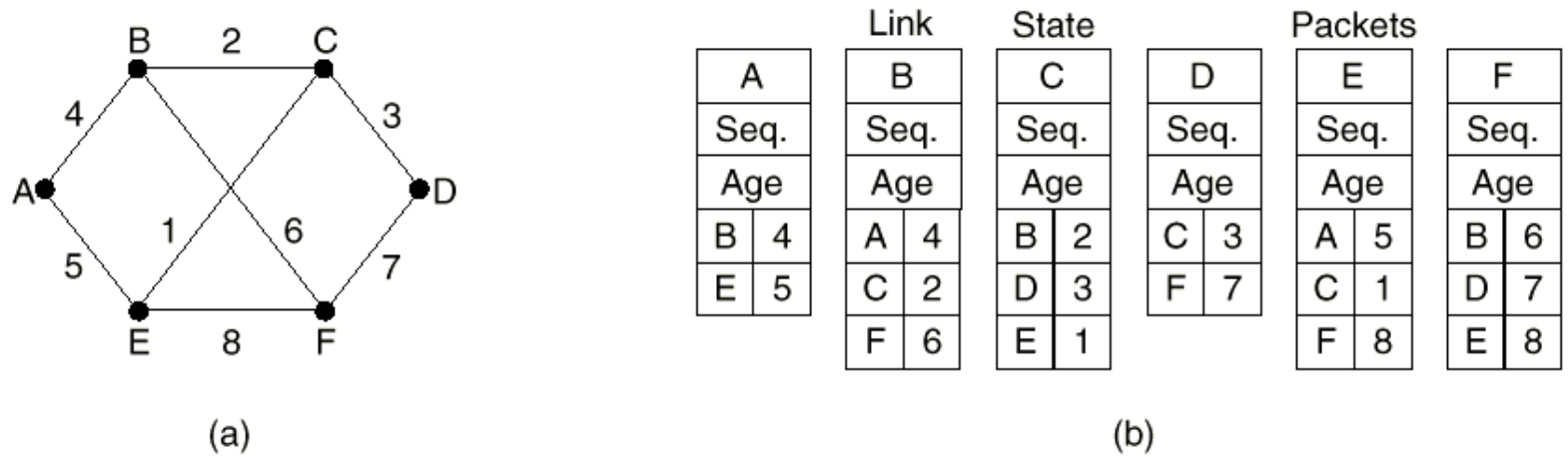


Fig. 5-15. (a) A subnet. (b) The link state packets for this subnet.

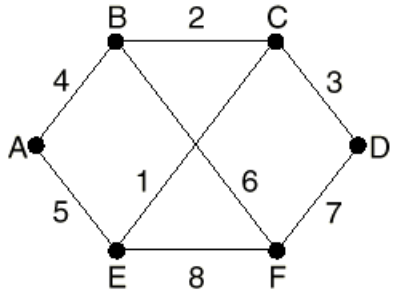
7.2.6 链路状态路由算法（3）

- 将这个包发送给所有其它路由器；
  - 基本思想：洪泛链路状态包，为控制洪泛，每个包包含一个序号，每次发送新包时加1。路由器记录信息对（源路由器，序号），当一个链路状态包到达时，若是新的，则分发；若是重复的，则丢弃；若序号比路由器记录中的最大序号小，则认为过时而丢弃；
  - 改进
    - 序号循环使用会混淆，解决办法：使用32位序号；
    - 路由器崩溃后，序号重置；
    - 序号出错；

7.2.6 链路状态路由算法（4）

  - 第二、三问题的解决办法：增加年龄（age）域，每秒钟年龄减1，为零则丢弃。
  - 链路状态包到达后，延迟一段时间，并与其它已到达的来自同一路由器的链路状态包比较序号，丢弃重复包，保留新包；
  - 链路状态包需要应答；
- 计算到每个其它路由器的最短路径。
  - 根据Dijkstra算法计算最短路径；

- 实用协议
  - OSPF
  - IS-IS



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	

Fig. 5-16. The packet buffer for router *B* in Fig.5-15.

- 从E发来的链路状态包有两个，一个经过EAB，另一个经过EFB；
- 从D发链路状态包有两个，一个经过DCB，另一个经过DFB；

7.2.6 链路状态路由算法（5）

- 链路状态算法（LS）和距离向量算法（DV）的比较
  - 路由信息的复杂性
    - LS
      - 路由信息向全网发送
      - N节点，E个连接的情况下，每个节点发送O(nE)的报文
    - DV
      - 仅在邻居节点之间交换

7.2.6 链路状态路由算法（6）

- 收敛（Convergence）速度
  - LS
    - 使用最短路径优先算法，算法复杂度为O(n**2)
      - n个结点（不包括源结点），需要n*(n+1)/2 次比较
      - 使用更有效的实现方法，算法复杂度可以达到O(nlogn)
    - 可能存在路由振荡（oscillations）
  - DV
    - 收敛时间不定
      - 可能会出现路由循环
      - count-to-infinity问题

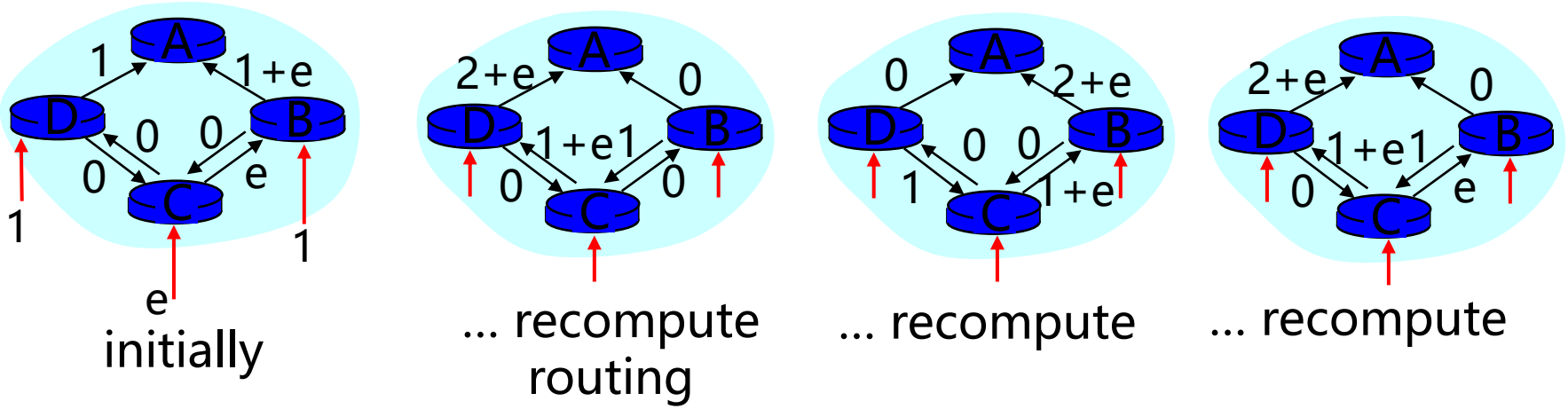
7.2 链路状态路由算法（7）

- 健壮性: 如果路由器不能正常工作会发生什么?
  - LS
    - 结点会广播错误的链路开销
    - 每个结点只计算自己的路由表
  - DV
    - 结点会广播错误的路径开销
    - 每个结点的路由表被别的结点使用，错误会传播到全网

7.2.6 链路状态路由算法（8）

Oscillations （震荡）possible:

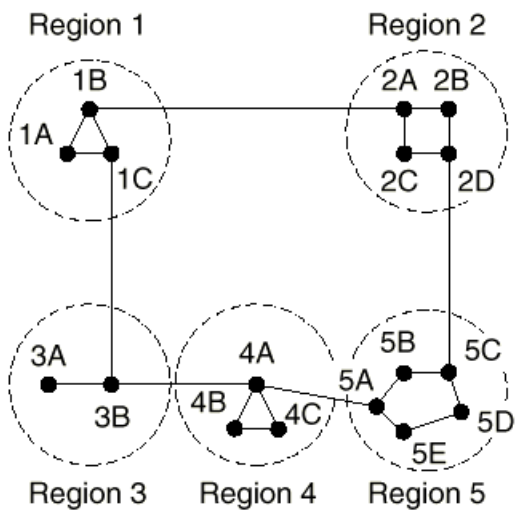
- e.g., link cost = amount of carried traffic



7.2.7 分层路由（1）

- 网络规模增长带来的问题
  - 路由器中的路由表增大；
  - 路由器为选择路由而占用的内存、CPU时间和网络带宽增大。
- 分层路由
  - 分而治之的思想；
  - 根据需要，将路由器分成区域（regions）、聚类（clusters）、区（zones）和组（groups）…
  - Fig. 5-17，路由表由17项减为7项。
- 分层路由带来的问题
  - 路由表中的路由不一定是最优路由。

7.2.7 分层路由（2）



(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)

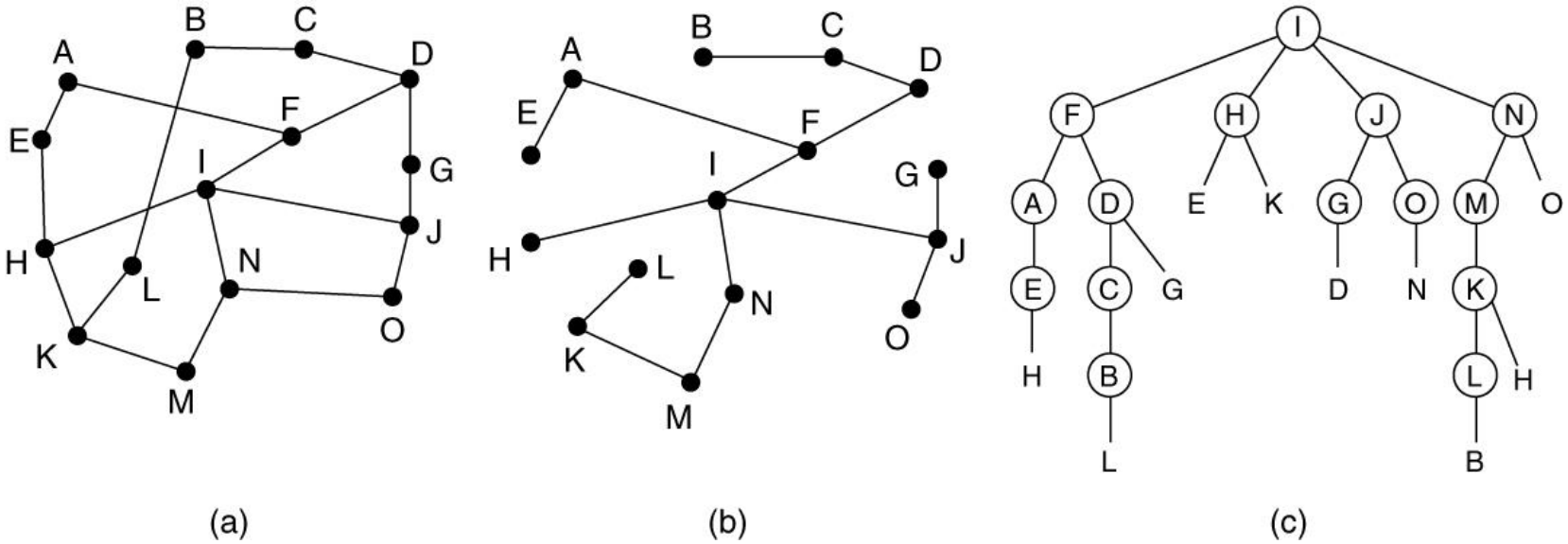
Hierarchical table for 1A

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

(c)

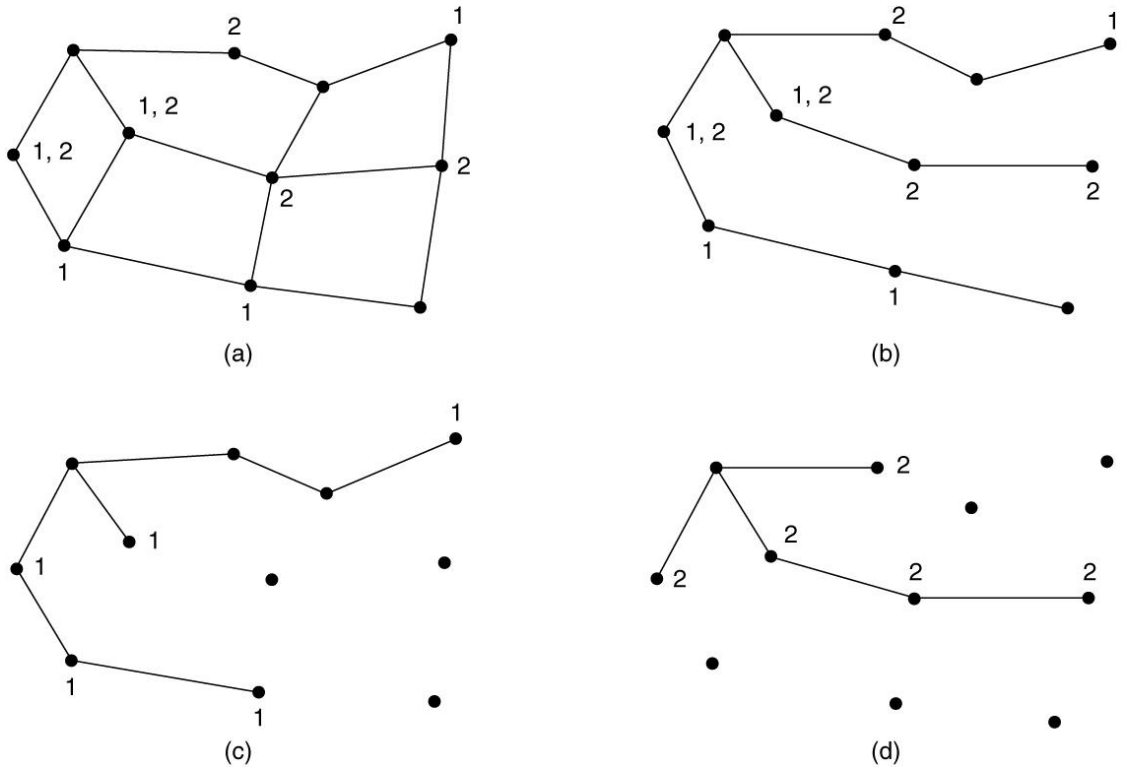
Fig. 5-17. Hierarchical routing.

7.2.8 广播路由



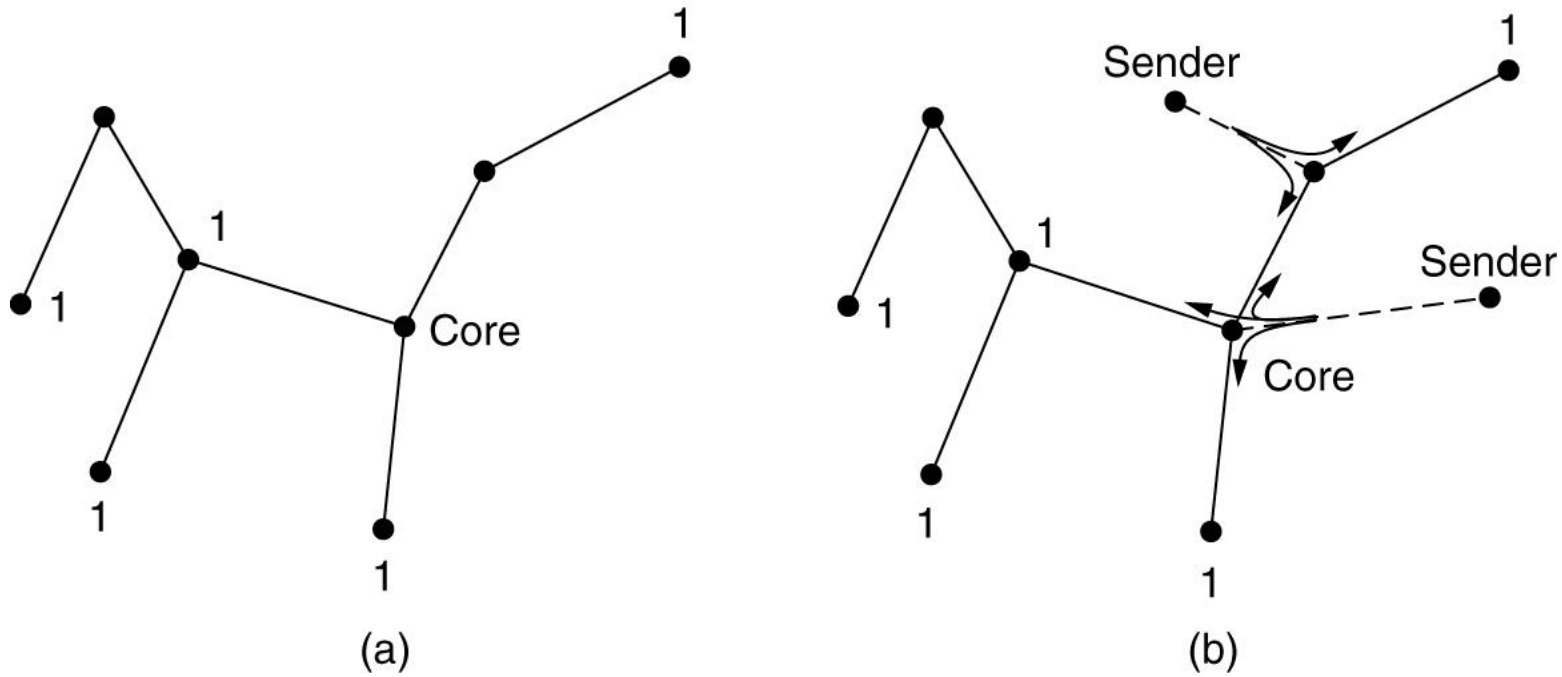
Reverse path forwarding. (a) A network. (b) Sink tree for router *I*. (c) The tree built by reverse path forwarding from *I*.

7.2.9 多播路由（1）



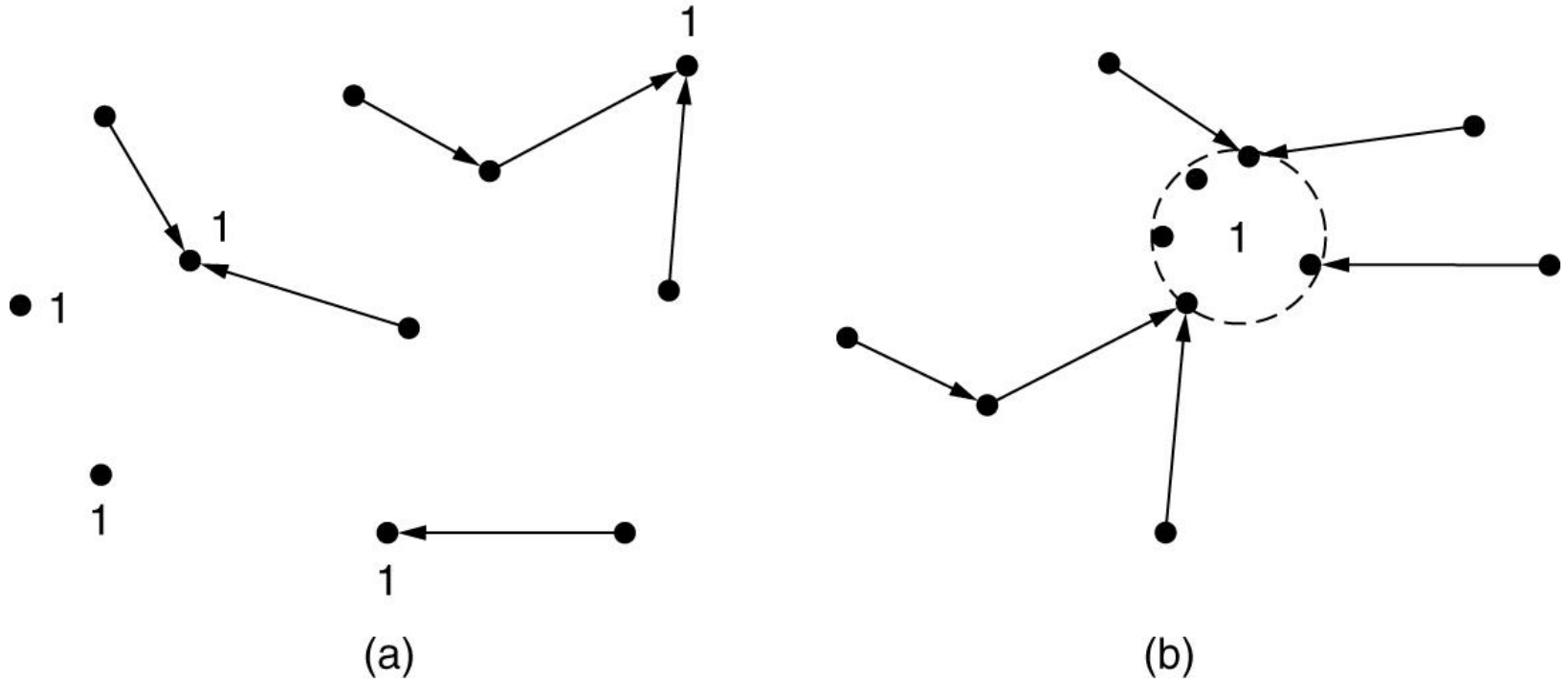
(a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

7.2.9 多播路由（2）



(a) Core-based tree for group 1. (b) Sending to group 1.

7.2.10 任播路由



(a) Anycast routes to group 1. (b) Topology seen by the routing protocol.

小结（1）

- 最优化原则
  - 路由算法的目的是找出并使用汇集树。
- 静态路由算法
  - 最短路径路由算法
  - 洪泛算法
  - 基于流量的路由算法

小结（2）

- 动态路由算法
  - 距离向量路由算法
    - 将自己（路由结点）对全网拓扑结构的认识告诉给邻居
    - 无穷计算问题
  - 链路状态路由算法
    - 将自己（路由结点）对邻居的认识洪泛给全网
- 分层路由
- 广播路由、多播路由、任播路由

7.3 流量管理和拥塞控制（1）

- 拥塞（congestion）
  - 网络上有太多的包时，性能会下降，这种情况称为**拥塞**。
- 拥塞产生的原因
  - 多个输入对应一个输出；
  - 慢速处理器；
  - 低带宽线路。
- 解决办法
  - 针对某个因素的解决方案，只能对提高网络性能起到一点点好处，甚至可能仅仅是转移了影响性能的瓶颈；
  - 需要全面考虑各个因素。

7.3 流量管理和拥塞控制（2）

- 拥塞控制与流量控制的差别
  - 拥塞控制（congestion control）需要确保通信子网能够承载用户提交的通信量，是一个全局性问题，涉及主机、路由器等很多因素；
  - 流量控制（flow control）与点到点的通信量有关，主要解决快速发送方与慢速接收方的问题，是局部问题，一般都是基于反馈进行控制的。

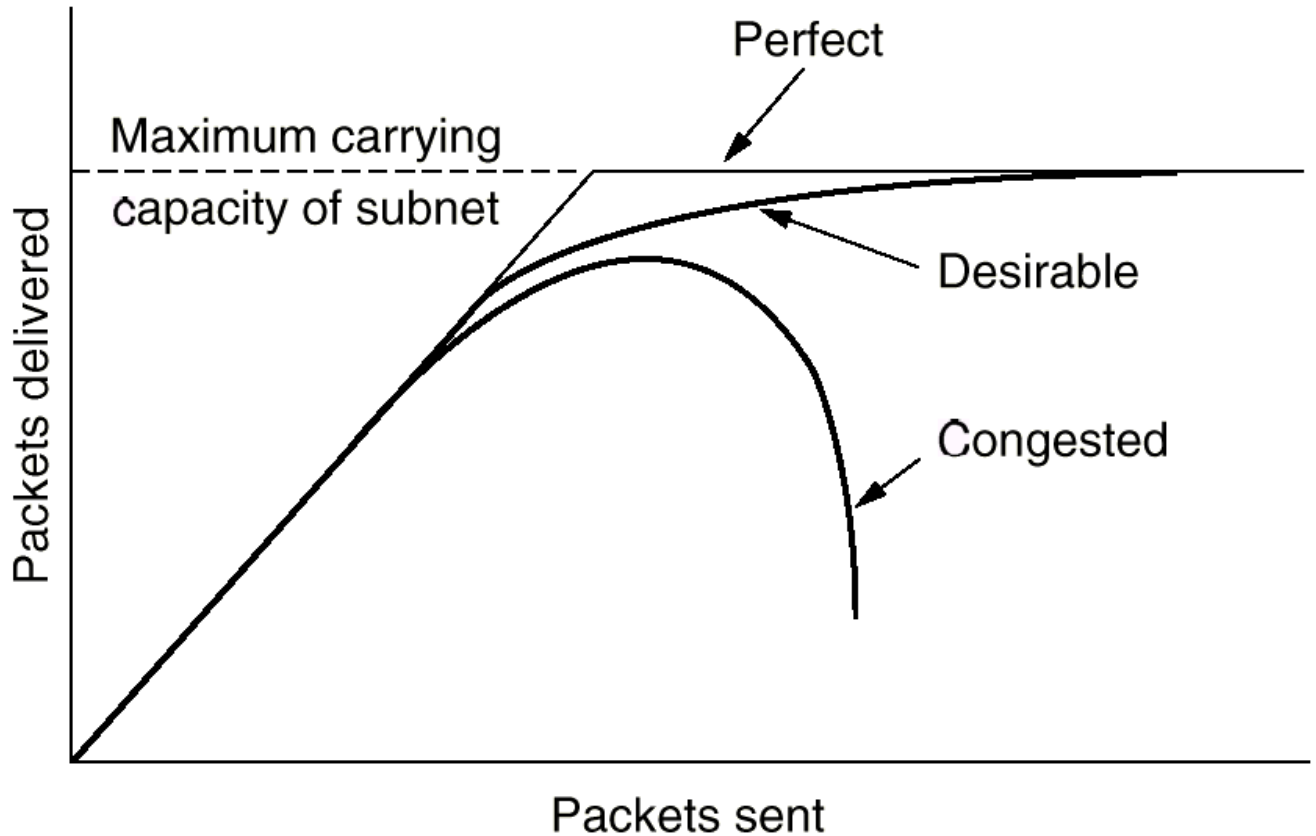


Fig. 5-22. When too much traffic is offered, congestion sets in and performance degrades sharply.

7.3 流量管理和拥塞控制（3）

7.3.1拥塞控制的基本原理

- 根据控制论，拥塞控制方法分为两类
  - 开环控制
    - 通过好的设计来解决问题，避免拥塞发生；
    - 拥塞控制时，不考虑网络当前状态。
  - 闭环控制
    - 基于反馈机制；
    - 工作过程
      - 监控系统，发现何时何地发生拥塞；
      - 把发生拥塞的消息传给能采取动作的站点
      - 调整系统操作，解决问题。

7.3 流量管理和拥塞控制（4）

- 衡量网络是否拥塞的参数
  - 缺乏缓冲区造成的丢包率；
  - 平均队列长度；
  - 超时重传的包的数目；
  - 平均包延迟；
  - 包延迟抖动（Jitter）。
- 反馈方法
  - 向负载发生源发送一个告警包；
  - 包结构中保留一个位或域用来表示发生拥塞，一旦发生拥塞，路由器将所有的输出包置位，向邻居告警；
  - 主机或路由器主动地、周期性地发送探报（probe），查询是否发生拥塞。

7.3 流量管理和拥塞控制（5）

7.3.2 拥塞控制算法

- 拥塞预防策略
  - 开环控制
  - 影响拥塞的网络设计策略

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Fig. 5-23. Policies that affect congestion.

7.3 流量管理和拥塞控制（6）

- 流量整形（Traffic Shaping）
  - 开环控制
  - 基本思想
    - 造成拥塞的主要原因是网络流量通常是突发性的；
    - 强迫包以一种可预测的速率发送；
    - 在ATM网中广泛使用。

7.3 流量管理和拥塞控制（7）

- 漏桶算法（The Leaky Bucket Algorithm）
  - 将用户发出的不平滑的数据包流转变成网络中平滑的数据包流；
  - 可用于固定包长的协议，如ATM；也可用于可变包长的协议，如IP，使用字节计数；
  - 无论负载突发性如何，漏桶算法强迫输出按平均速率进行，不灵活。

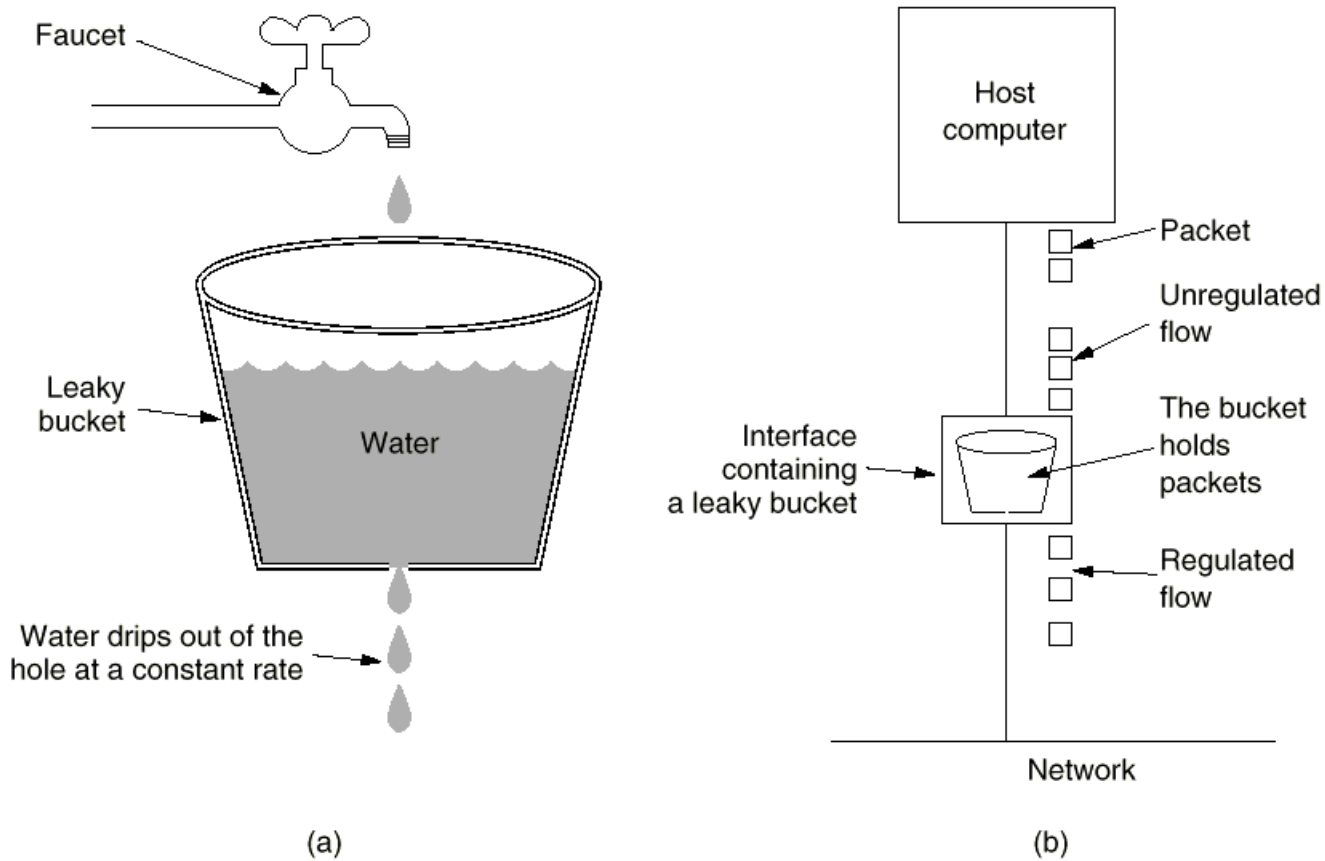


Fig. 5-24. (a) A leaky bucket with water. (b) A leaky bucket with packets.

7.3 流量管理和拥塞控制（8）

– 令牌桶算法（The Token Bucket Algorithm）

- 漏桶算法不够灵活，因此加入令牌机制；
- 基本思想：漏桶存放令牌，每ΔT秒产生一个令牌，令牌累积到超过漏桶上界时就不再增加。包传输之前必须获得一个令牌，传输之后删除该令牌；

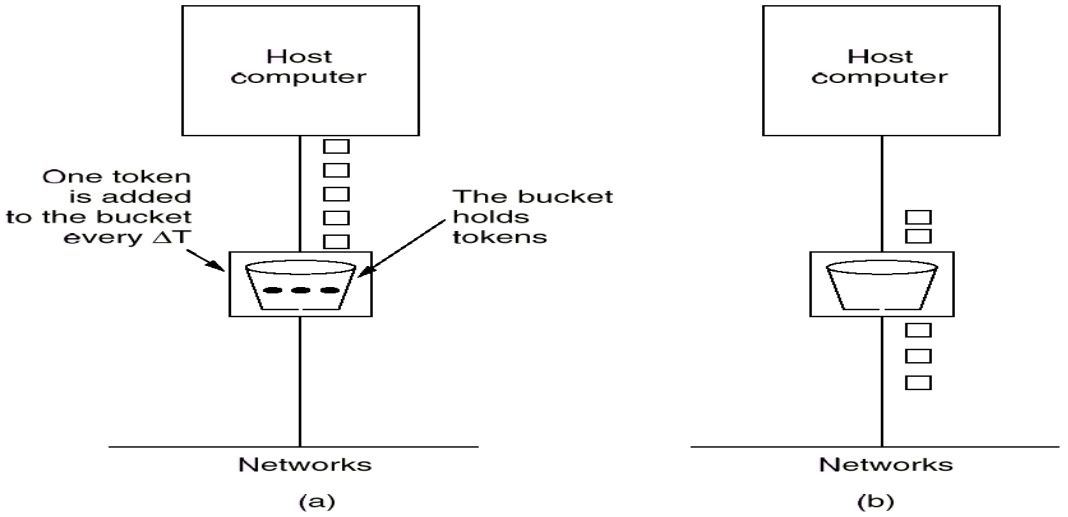


Fig. 5-26. The token bucket algorithm. (a) Before. (b) After.

7.3 流量管理和拥塞控制（9）

- 漏桶算法与令牌桶算法的区别
  - 流量整形策略不同：漏桶算法不允许空闲主机积累发送权，以便以后发送大的突发数据；令牌桶算法允许，最大为桶的大小。
  - 漏桶中存放的是数据包，桶满了丢弃数据包；令牌桶中存放的是令牌，桶满了丢弃令牌，不丢弃数据包。

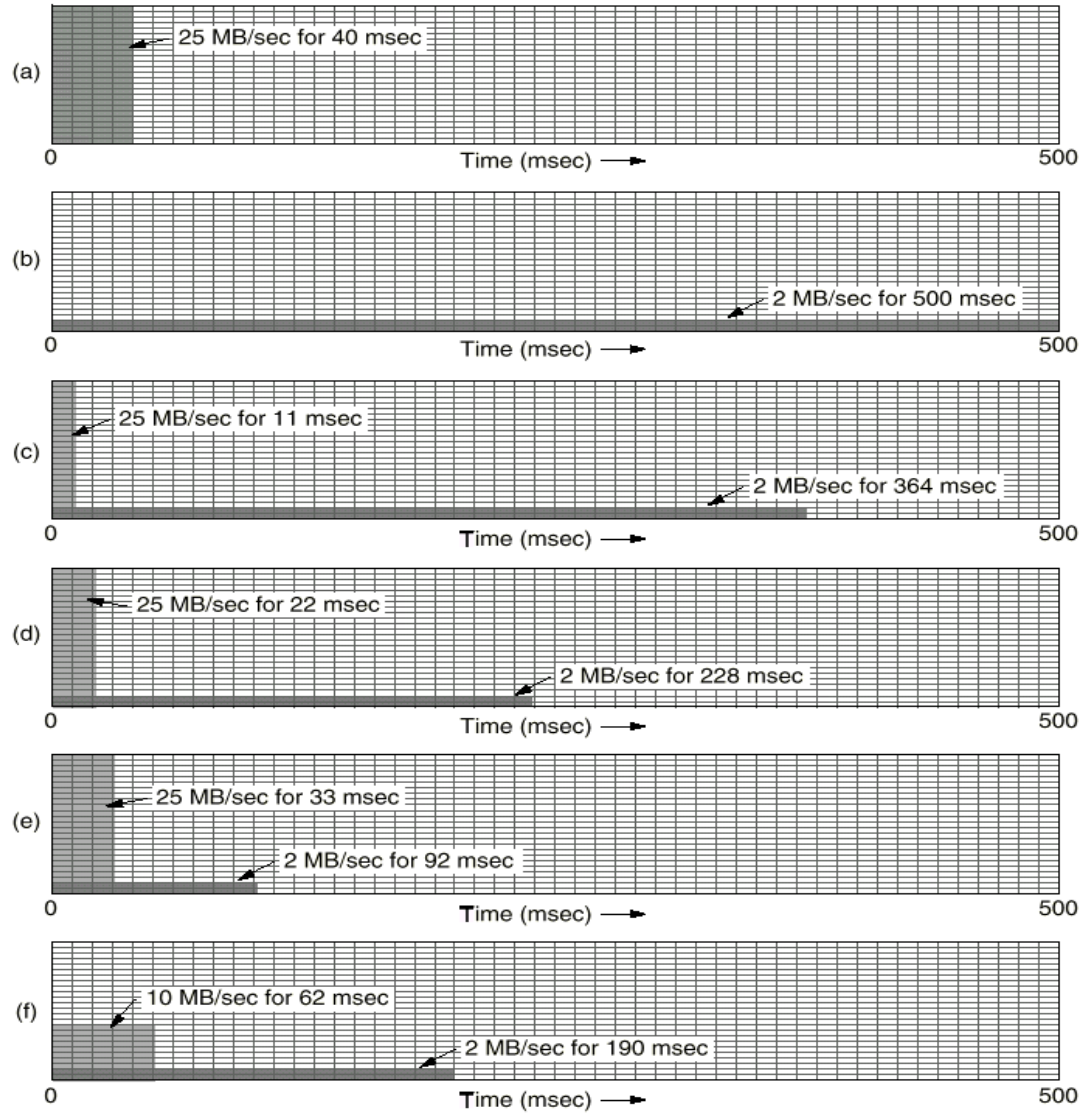


Fig. 5-25. (a) Input to a leaky bucket. (b) Output from a leaky bucket. (c) - (e) Output from a token bucket with capacities of 250KB, 500KB, and 750KB. (f) Output from a 500KB token bucket feeding a 10 MB/sec leaky bucket.

7.3 流量管理和拥塞控制（10）

- 流说明（Flow Specification）
  - 一个数据流的发送方、接收方和通信子网三方认可的、描述发送数据流的模式和希望得到的服务质量的数据结构，称为**流说明**。
  - 对发送方的流说明，子网和接收方可以做出三种答复：同意、拒绝、其它建议。

Characteristics of the Input	Service Desired
Maximum packet size (bytes)	Loss sensitivity (bytes)
Token bucket rate (bytes/sec)	Loss interval (μsec)
Token bucket size (bytes)	Burst loss sensitivity (packets)
Maximum transmission rate (bytes/sec)	Minimum delay noticed (μsec)
	Maximum delay variation (μsec)
	Quality of guarantee

Fig. 5-27. An example flow specification.

7.3 流量管理和拥塞控制（11）

- 虚电路子网中的拥塞控制
  - 许可控制（admission control），基本思想：一旦发生拥塞，在问题解决之前，不允许建立新的虚电路；
  - 另一种方法是发生拥塞后可以建立新的虚电路，但要绕开发生拥塞的地区；
  - 资源预留：建立虚电路时，主机与子网达成协议，子网根据协议在虚电路上为此连接预留资源。

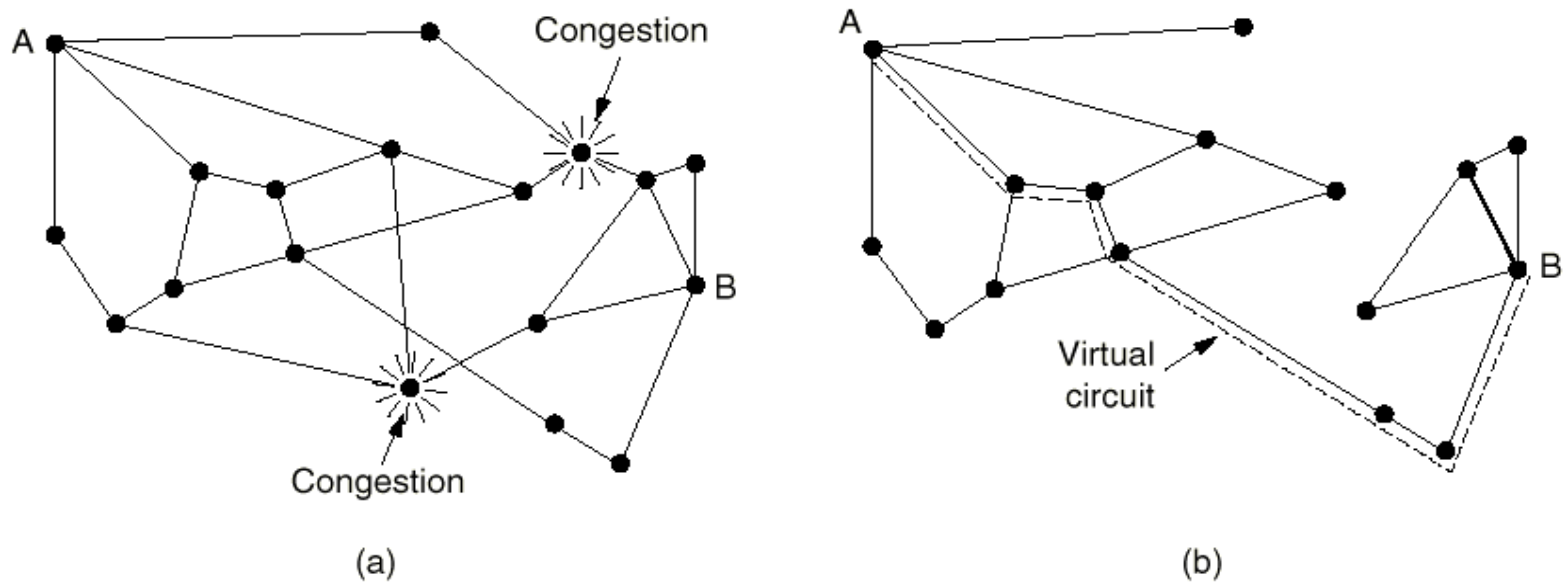


Fig. 5-28. (a) A congested subnet. (b) A redrawn subnet that eliminates the congestion and a virtual circuit from *A* to *B*.

7.3 流量管理和拥塞控制（12）

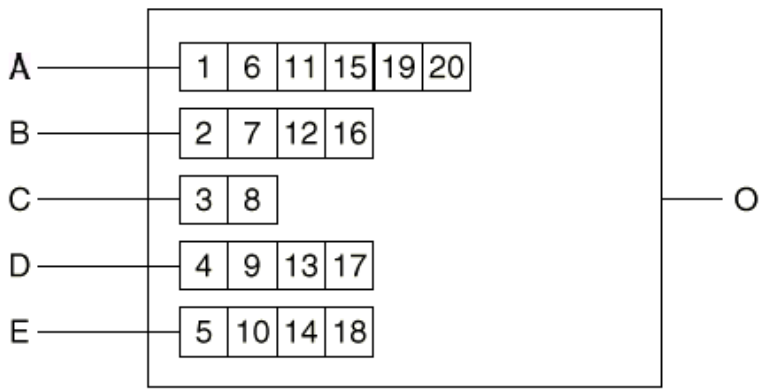
- 抑制包（Choke Packets）
  - 基本思想
    - 路由器监控输出线路及其它资源的利用情况，超过某个阈值，则此资源进入警戒状态；
    - 每个新包到来，检查它的输出线路是否处于警戒状态；
    - 若是，则向源主机发送抑制包，包中指出发生拥塞的目的地址。同时将原包打上标记（为了以后不再产生抑制包），正常转发；

7.3 流量管理和拥塞控制（13）

- 源主机收到抑制包后，按一定比例减少发向特定目的地的流量，并在固定时间间隔内忽略指示同一目的地的抑制包。然后开始监听，若此线路仍然拥塞，则主机在固定时间内减轻负载、忽略抑制包；若在监听周期内没有收到抑制包，则增加负载；
- 通常采用的流量增减策略是：减少时，按一定比例减少，保证快速解除拥塞；增加时，以常量增加，防止很快导致拥塞。

7.3 流量管理和拥塞控制（14）

- 加权公平队列（Weighted Fair Queueing）
  - 公平队列（Fair Queueing）算法
  - 路由器的每个输出线路有多个队列；
    - 路由器循环扫描各个队列，发送队头的包；
    - 所有队列具有相同优先级；
    - 一些ATM交换机、路由器使用这种算法；
    - 一种改进：对于变长包，由逐包轮讯改为逐字节轮讯
  - 加权公平队列算法
    - 给不同主机以不同的优先级；
    - 优先级高的主机在一个轮讯周期内获得更多的时间片。



(a)

Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

Fig. 5-29. (a) A router with five packets queued for line *O*. (b) Finishing times for the five packets.

7.3 流量管理和拥塞控制（15）

- 逐跳抑制包（Hop-by-Hop Choke Packets）
  - 在高速、长距离的网络中，由于源主机响应太慢，抑制包算法对拥塞控制的效果并不好，可采用逐跳抑制包算法。
  - 基本思想
    - 抑制包对它经过的每个路由器都起作用；
    - 能够迅速缓解发生拥塞处的拥塞；
    - 上游路由器要求有更多的缓冲区；

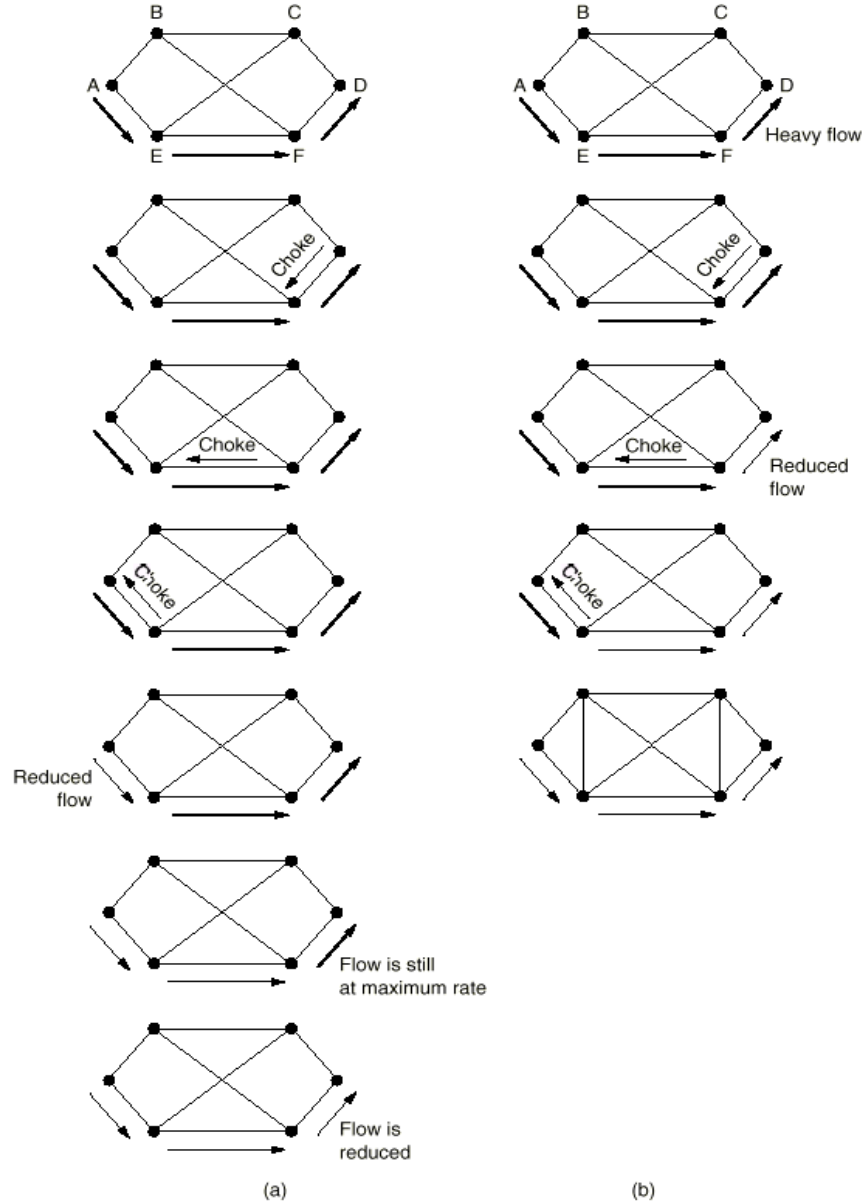


Fig. 5-30. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.

7.3 流量管理和拥塞控制（16）

- 负载丢弃（Load Shedding）
  - 上述算法都不能消除拥塞时，路由器只得将包丢弃；
  - 针对不同服务，可采取不同丢弃策略
    - 文件传输，优先丢弃新包，wine策略；
    - 多媒体服务，优先丢弃旧包，milk策略；
  - 早期丢弃包，会减少拥塞发生的概率，提高网络性能。

7.4 网络互联（1）

- 互联网（internet）：两个或多个网络构成互联网。

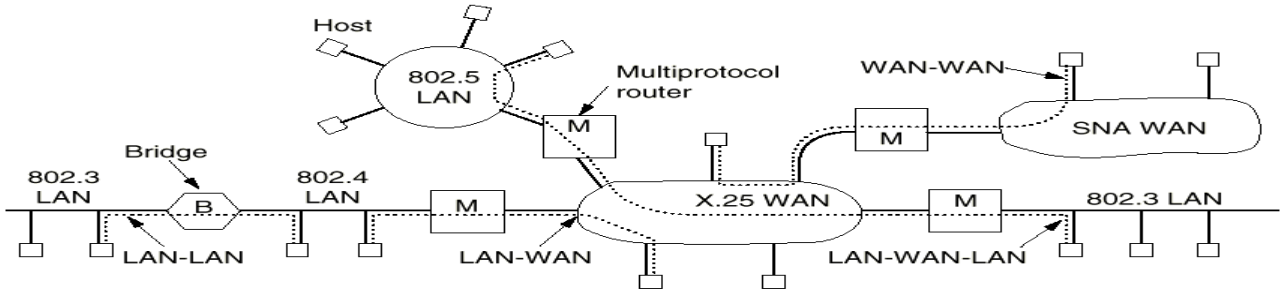


Fig. 5-33. Network interconnection.

- 多种不同网络（协议）存在的原因
  - 历史原因：不同公司的网络产品大量使用；
  - 价格原因：网络产品价格低，更多的人有权决定使用何种网络；
  - 技术原因：不同网络采用不同技术、不同硬件、不同协议。

7.4 网络互联（2）

- 网络互连设备
  - 中继器（repeater）
    - 物理层设备，在电缆段之间拷贝比特；
    - 对弱信号进行放大或再生，以便延长传输距离
  - 网桥（bridge）
    - 数据链路层设备，在局域网之间存储转发帧；
    - 网桥可以改变帧格式。
  - 多协议路由器（multiprotocol router）
    - 网络层设备，在网络之间存储转发包；
    - 必要时，做网络层协议转换。

7.4 网络互联（3）

- 传输网关（transport gateway）
  - 传输层设备，在传输层转发字节流。
- 应用网关（application gateway）
  - 应用层设备，在应用层实现互连；
  - half-gateway(为了满足不同国家、组织的管理需要)

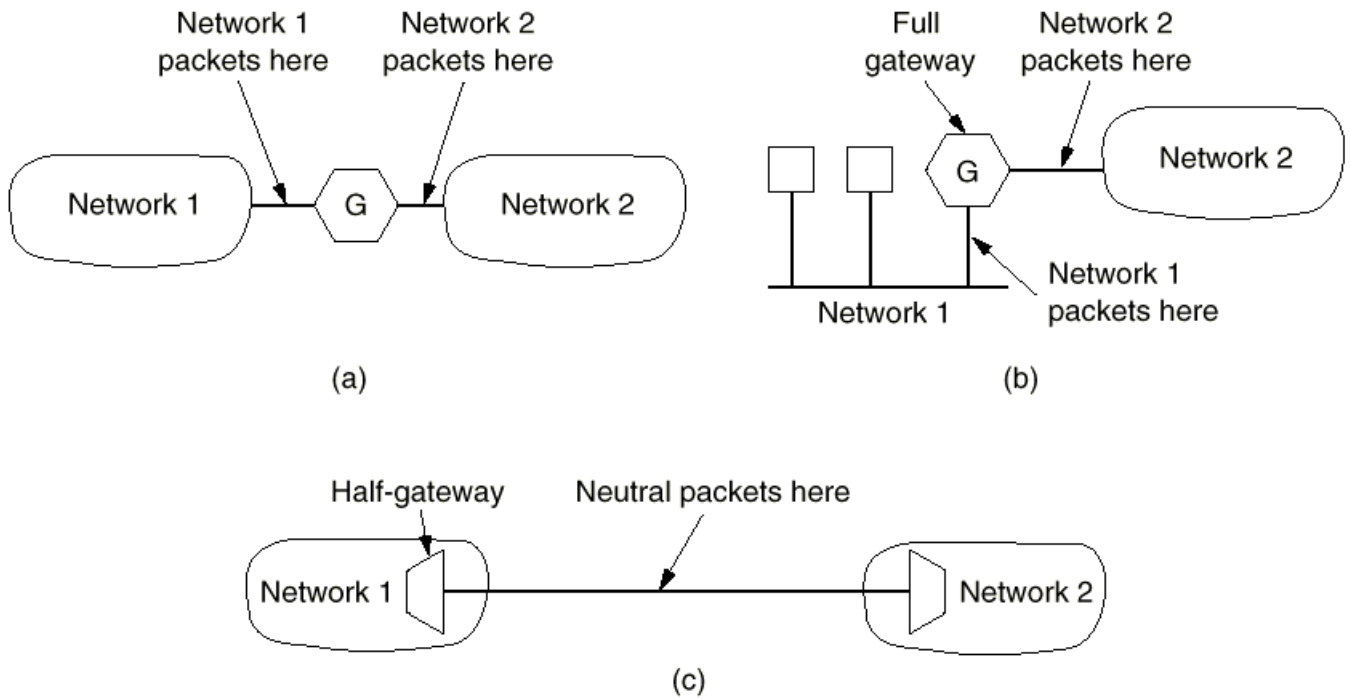


Fig. 5-34. (a) A full gateway between two WANs. (b) A full gateway between a LAN and a WAN. (c) Two half-gateways.

7.4 网络互联（4）

- 网络之间的区别

Item	Some Possibilities
Service offered	Connection-oriented versus connectionless
Protocols	IP, IPX, CLNP, AppleTalk, DECnet, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	May be present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

Fig. 5-35. Some of the many ways networks can differ.

7.4 网络互联（5）

7.4.1 级联虚电路（Concatenated Virtual Circuits）

- 工作过程
  - 级联虚电路工作过程与虚电路子网工作过程相似
  - 建立连接
    - 当目的主机不在子网内时，则在子网内找一个离目的网络最近的路由器，与之建立一条虚电路；
    - 该路由器与外部网关建立虚电路；
    - 该网关与下一个子网中的一个路由器建立虚电路；
    - 重复上述操作，直到到达目的主机。

7.4 网络互联（6）

- 传输数据
  - 相同连接的包沿同一虚电路按序号传输；
  - 网关根据需要转换包格式和虚电路号。
- 拆除连接

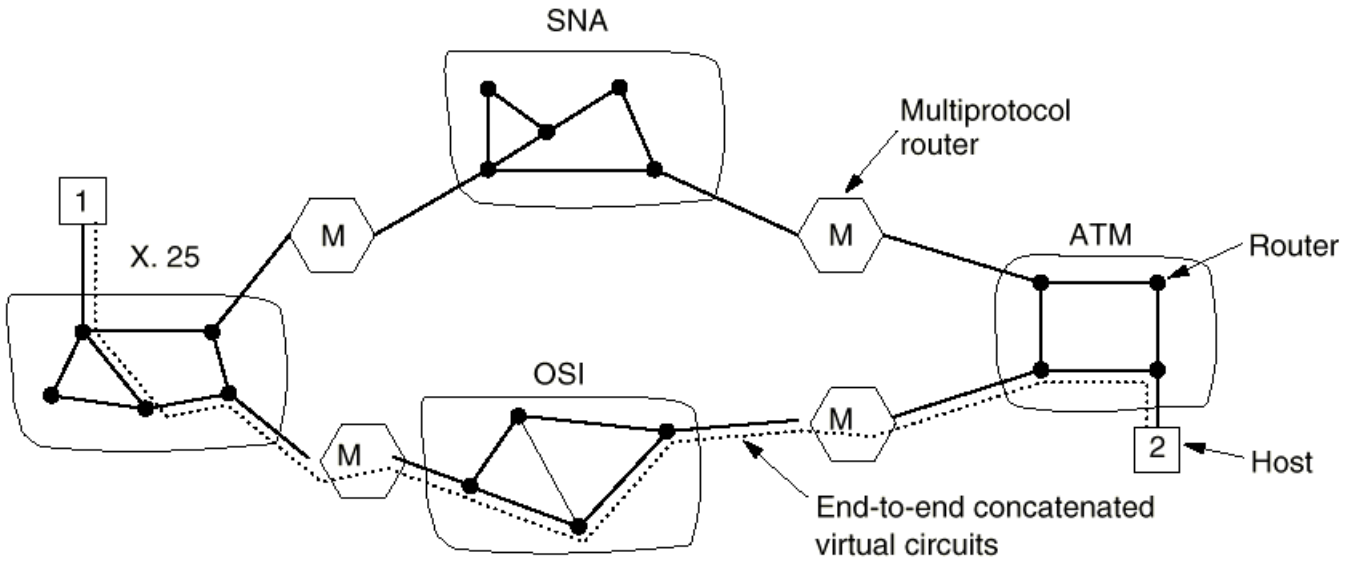


Fig. 5-36. Internetworking using concatenated virtual circuits.

7.4 网络互联（7）

7.4.2 无连接网络互连 (Connectionless Internetworking)

- 工作过程
  - 无连接网络互连的工作过程与数据报子网的工作过程相似；
  - 每个包单独路由，提高网络利用率，但不能保证包按顺序到达；
  - 根据需要，连接不同子网的多协议路由器做协议转换，包括包格式转换和地址转换等。

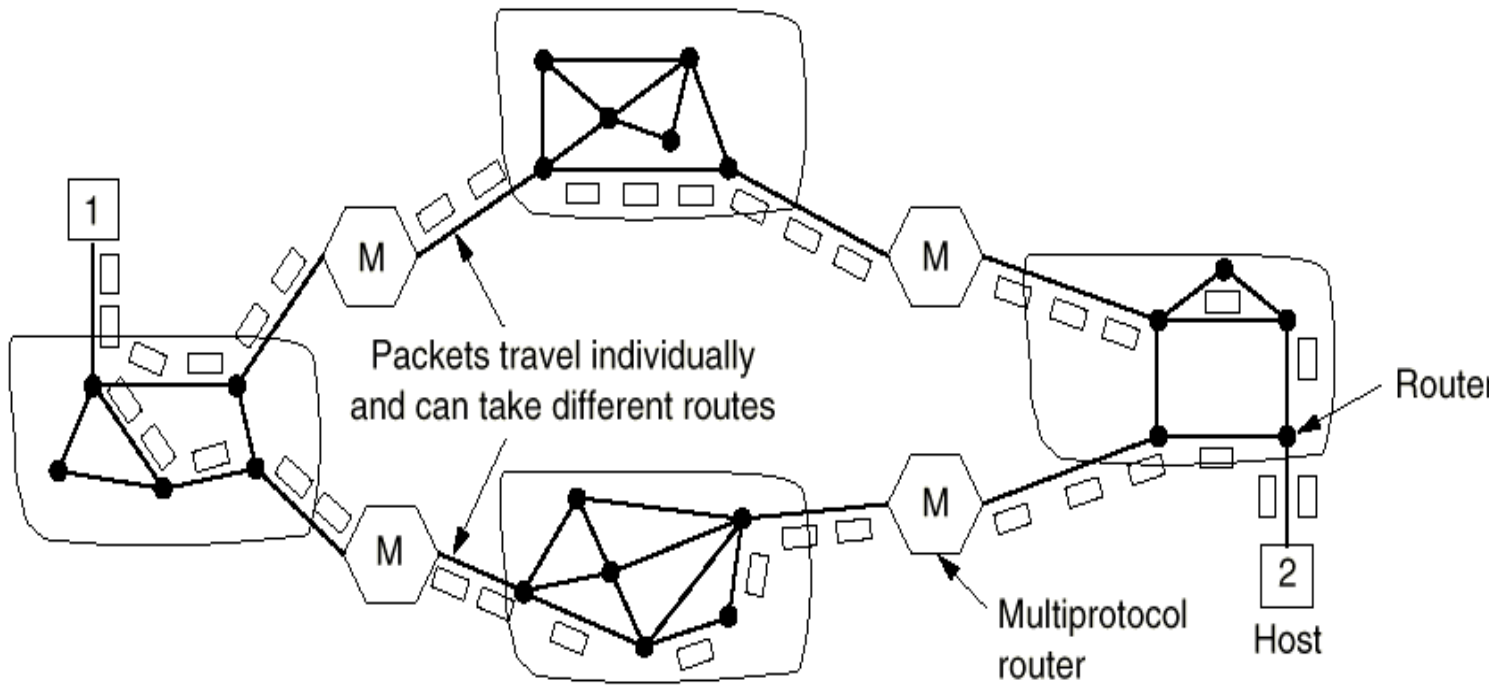


Fig. 5-37. A connectionless internet.

7.4 网络互联（8）

- 级联虚电路与无连接网络互连的比较
  - 级联虚电路的优点
    - 路由器预留缓冲区等资源，保证服务质量；
    - 包按序号传输；
    - 短包头。
  - 级联虚电路的缺点
    - 路由器需要大量内存，存储虚电路信息；
    - 一旦发生拥塞，没有其它路由；
    - 健壮性差；
    - 如果网络中有一个不可靠的数据报子网，级连虚电路很难实现。

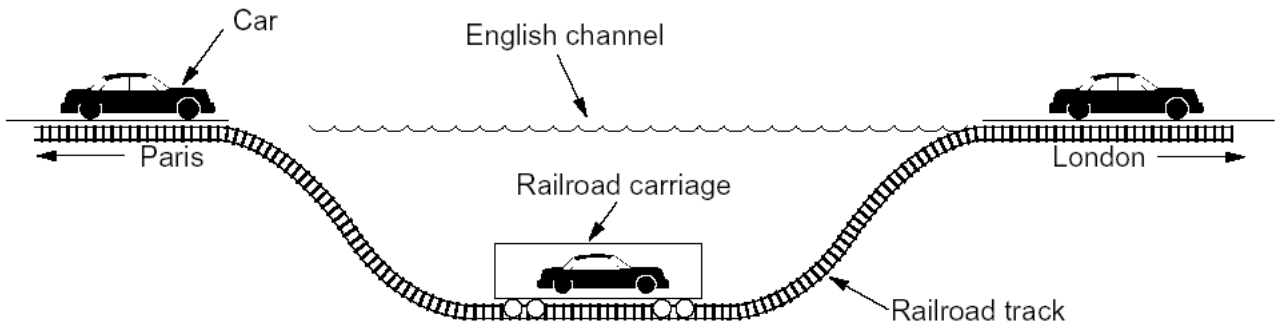
7.4 网络互联（9）

- 无连接网络互连的优点
  - 能够容忍拥塞，并能适应拥塞；
  - 健壮性好；
  - 可用于多种网络互连。
- 无连接网络互连的缺点
  - 长包头；
  - 包不能保证按序号到达；
  - 不能保证服务质量。

7.4 网络互联（10）

7.4.3 隧道技术（Tunneling）

- 隧道技术
  - 源和目的主机所在网络类型相同，连接它们的是一个不同类型的网络，这种情况下可以采用隧道技术。



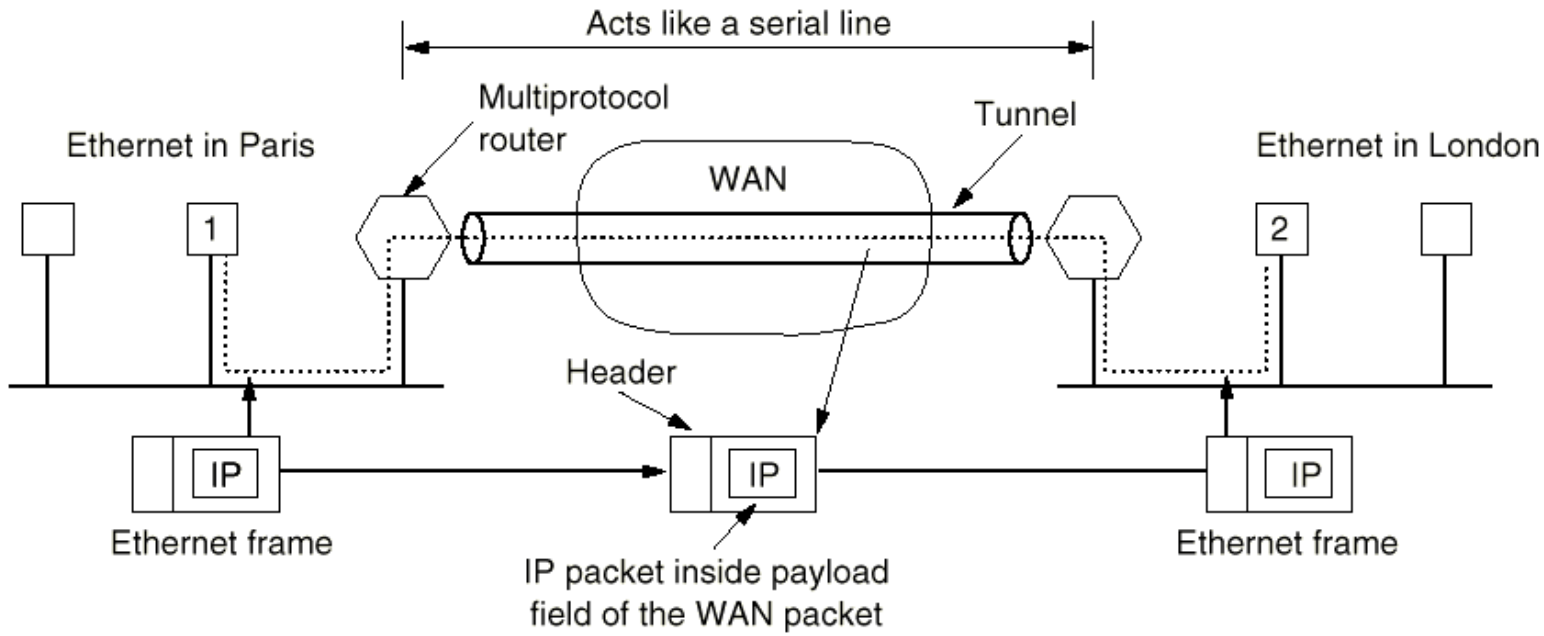


Fig. 5-38. Tunneling a packet from Paris to London.

7.4 网络互联（11）

- 工作过程（以Fig. 5-38为例）
  - 主机1发送一个包，目的IP地址 = 主机2-IP，将包封装到局域网帧中，帧目的地址 = 路由器1-MAC;
  - 局域网传输;
  - 路由器1剥掉局域网帧头、帧尾，将得到的IP包封装到广域网***网络层包***中，包目的地址 = 路由器2地址;
  - 广域网传输;
  - 路由器2剥掉广域网包头，将得到的IP包封装到局域网帧中，包目的IP地址 = 主机2-IP，帧目的地址 = 主机2-MAC地址;
  - 局域网传输;
  - 主机2接收。

7.4 网络互联（12）

7.4.4 互联网路由（Internetwork Routing）

- 工作过程
  - 互联网络的路由与单独子网的路由过程相似，只是复杂性增加；

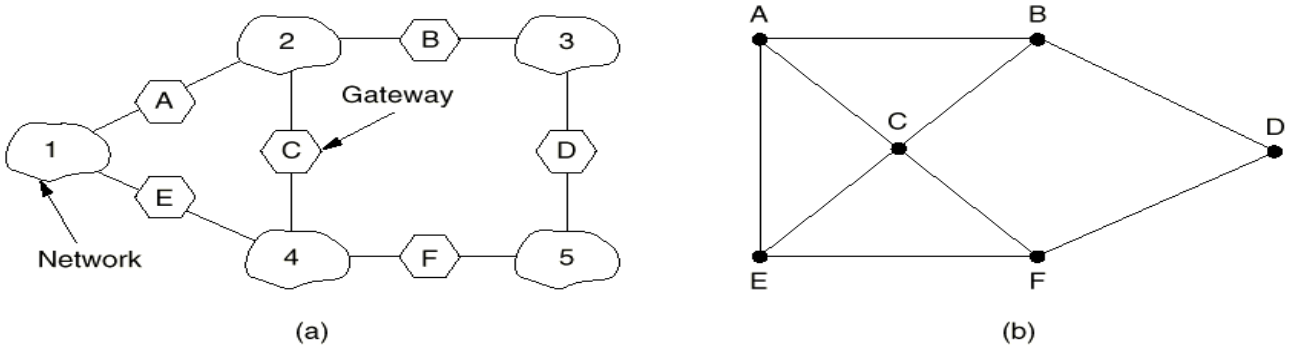


Fig. 5-40. (a) An internetwork. (b) A graph of the internetwork.

7.4 网络互联（13）

- 两级路由算法
  - 内部网关协议(IGP: Interior Gateway Protocol)
    RIP，OSPF
  - 外部网关协议(EGP: Exterior Gateway Protocol)
    BGP
  - 自治系统AS（Autonomous System）

7.4 网络互联（14）

7.4.5 分段（ Fragmentation ）

- 每种网络都对最大包长有限制，有以下原因
  - 硬件，例如 TDM 的时槽限制；
  - 操作系统；
  - 协议，例如包长度域的比特个数；
  - 与标准的兼容性；
  - 希望减少传输出错的概率；
  - 希望避免一个包占用信道时间过长。
- 大包经过小包网络时，网关要将大包分成若干段（fragment），每段作为独立的包传输。

7.4 网络互联（15）

- 段重组策略
  - 分段重组过程对其它网络透明
    - 网关将大包分段后，每段都要经过同一出口网关，并在那里重组；
    - 带来的问题
      - 出口网关需要知道何时所有分组都到齐；
      - 所有分组必须从同一出口网关离开；
      - 大包经过一系列小包网络时，需要反复地分段重组，开销大。

7.4 网络互联（16）

- 分段重组过程对其它网络不透明
  - 中间网关不做重组，而由目的主机做；
  - 带来的问题
    - 对主机要求高，能够重组；
    - 每个段都要有一个包头，网络开销增大；

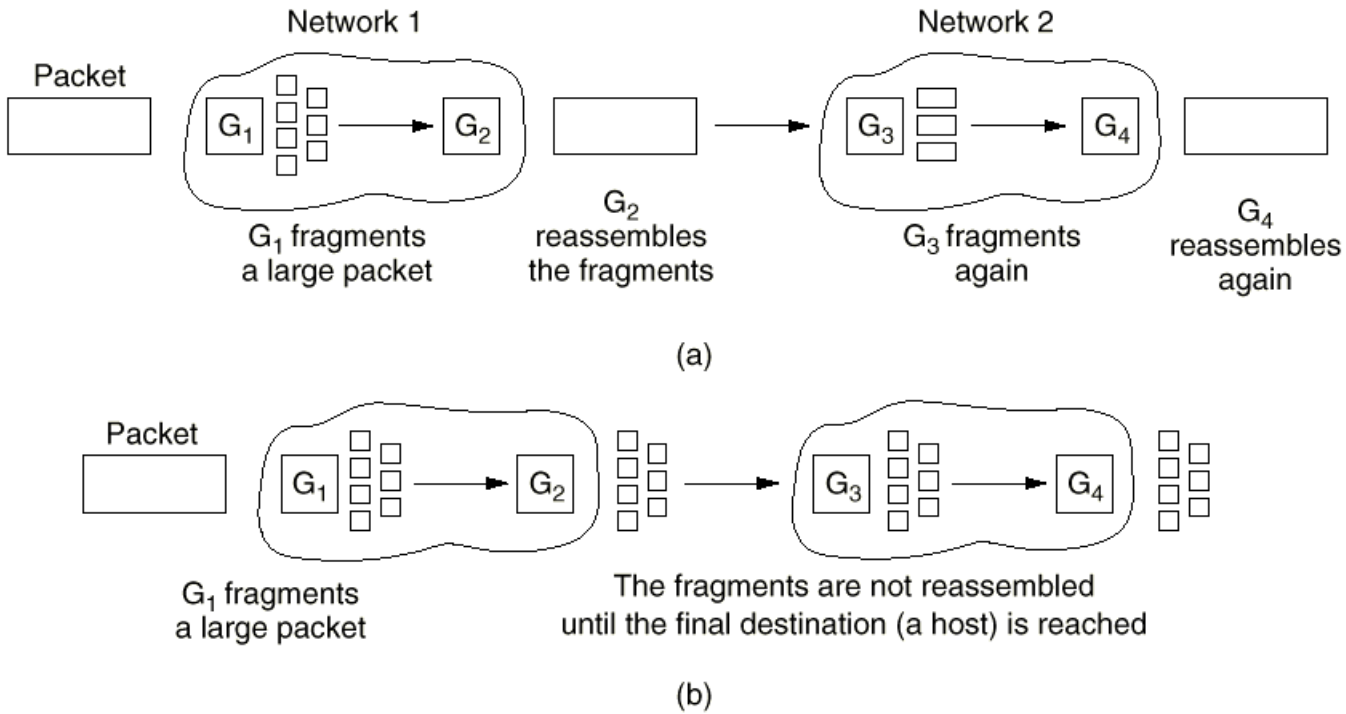


Fig. 5-41. (a) Transparent fragmentation. (b) Nontransparent fragmentation.

7.4 网络互联（17）

- 标记段
  - 树型标记法
    - 例，包0分成三段，分别标记为0.0, 0.1, 0.2，段0.0构成的包被分成三段，分别标记为0.0.0, 0.0.1, 0.0.2；
    - 存在的问题
      - 段标记域要足够长
      - 分段长度前后要一致

7.4 网络互联（18）

- 偏移量法
  - 定义一个基本段长度,使得基本段能够通过所有网络;
  - 包分段时，除最后一个段小于等于基本段长度外，所有段长度都等于基本段长度;
  - 一个包可以包括几个段，包头中包括；原始包序号，包中第一个基本段的偏移量，最后段指示位。

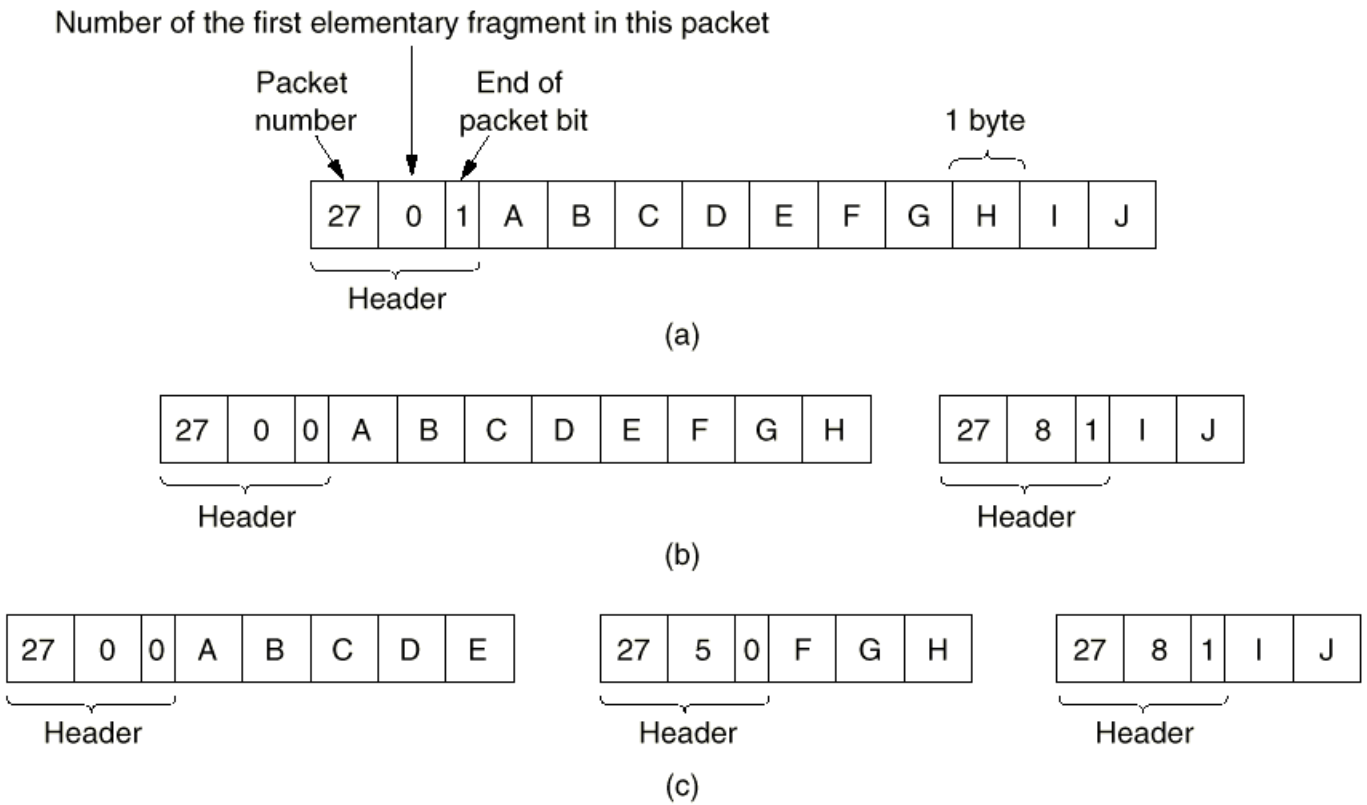


Fig. 5-42. Fragmentation when the elementary data size is 1 byte. (a) Original packet, containing 10 data bytes. (b) Fragments after passing through a network with maximum packet size of 8 bytes. (c) Fragments after passing through a size 5 gateway.

7.4 网络互联（19）

7.4.6 防火墙（Firewalls）

- 什么情况下使用防火墙?
  - 为防止网络中的信息泄露出去或不好的信息渗透进来，在网络边缘设置防火墙；
- 防火墙的一种常用配置
  - 两个路由器，根据某种规则表，进行包过滤；
  - 一个应用网关，审查应用层信息。

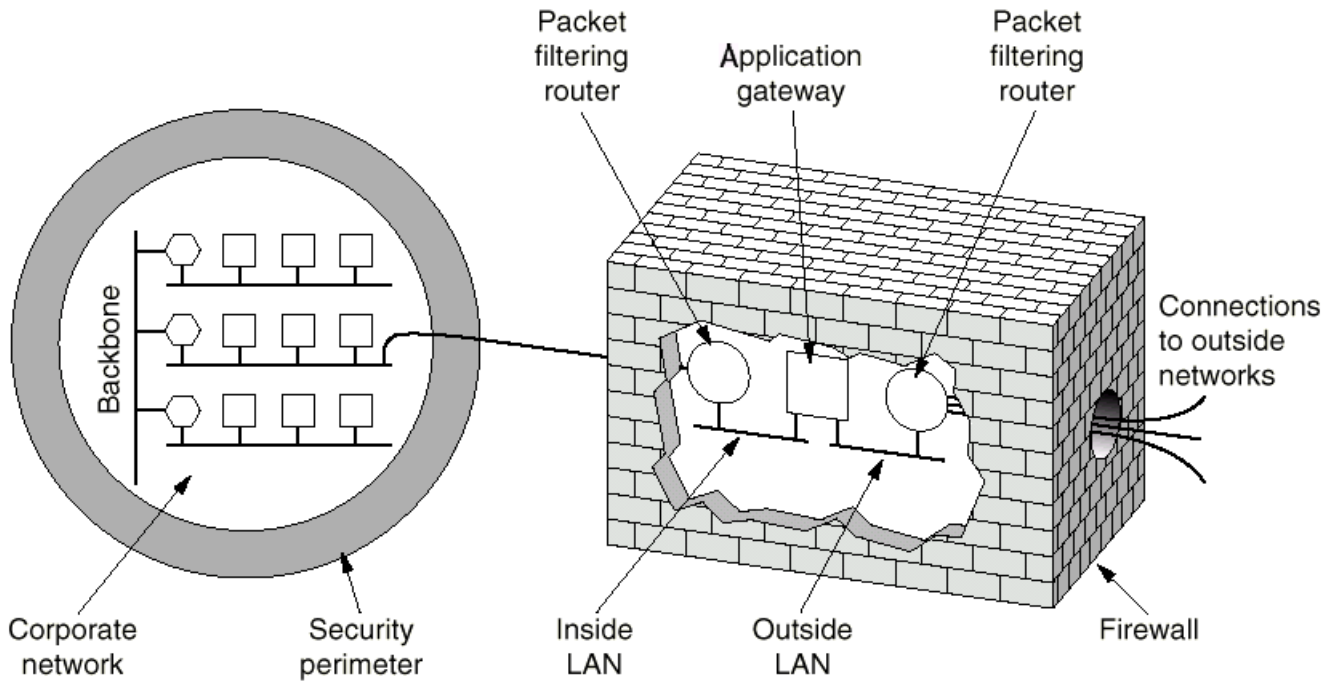


Fig. 5-43. A firewall consisting of two packet filters and an application gateway.

第七章 网络层

第一部分结束