



第七章

路由选择和网络层



主要内容

- 网络层概述
- 路由算法
 - **最优化原则**
 - **最短路径路由算法**
 - **洪泛算法**
 - **基于流量的路由算法**
 - **距离向量路由算法**
 - **链路状态路由算法**
 - **分层路由**
 - **移动主机的路由**



主要内容（续）

- 拥塞控制算法
 - 拥塞控制的基本原理
 - 拥塞控制算法
- 网络互连
 - 级联虚电路
 - 无连接网络互连
 - 隧道技术
 - 互联网路由
 - 分片
 - 防火墙



主要内容（续）

- 互联网网络层协议
 - IP协议
 - Internet控制协议ICMP
 - 内部网关路由协议OSPF
 - 外部网关路由协议BGP
- 路由器体系结构和关键技术



网络层概述

- 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（传统意义）
- 网络层的功能
 - 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



网络层提供的服务

- 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
 - 将复杂的功能放在网络层(通信子网)
 - 无连接服务
 - 互联网的观点：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
 - 将复杂的功能放在传输层



IP over ATM

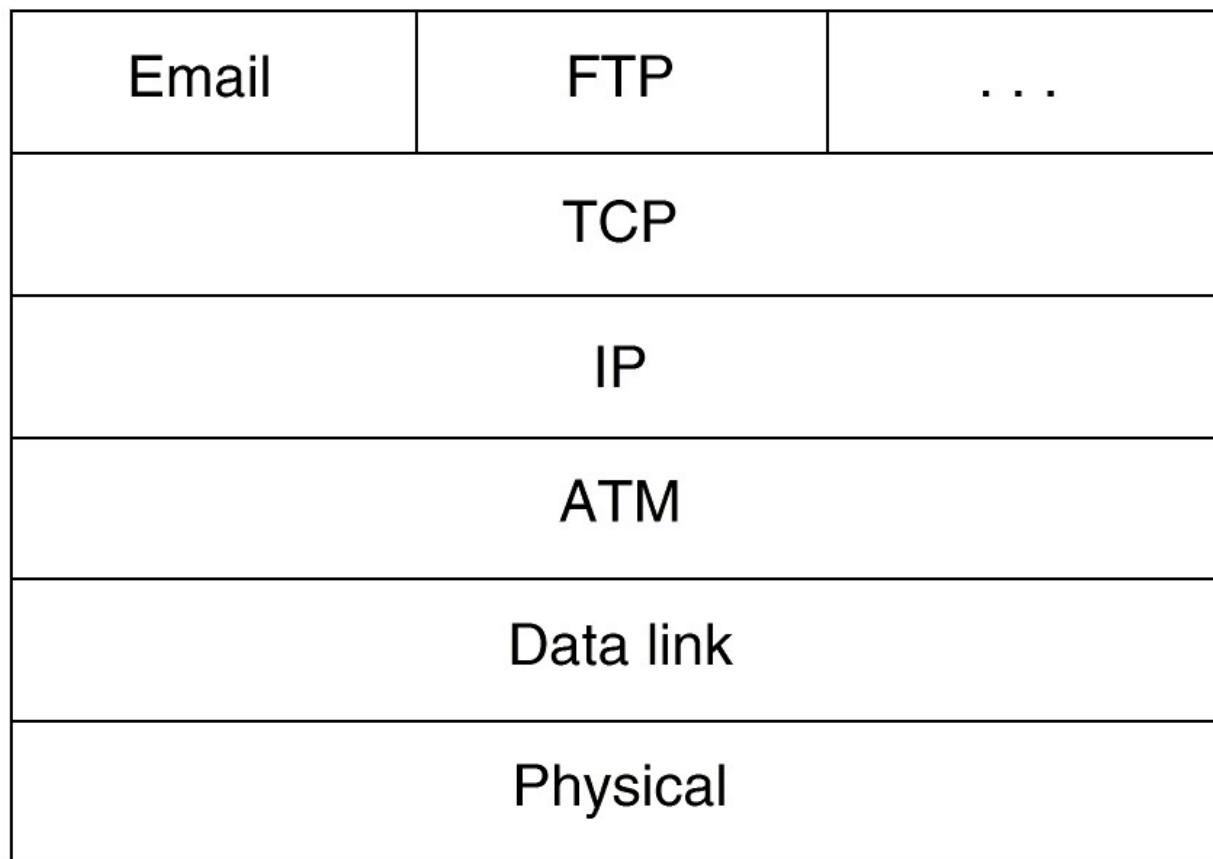


Fig. 5-1. Running TCP/IP over an ATM subnet.



数据报子网与虚电路子网

- 网络层的内部结构
 - 数据报 (datagram) 子网
 - 采用数据报分组交换
 - 每个分组被独立转发，分组带有全网唯一的地址
 - 虚电路 (virtual circuit) 子网
 - 采用虚电路分组交换
 - 先在源结点和目的结点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



数据报子网与虚电路子网（续）

- 虚电路子网与数据报子网的比较
 - 带宽与状态的权衡
 - 数据报子网中，每个数据报都携带完整的目的/源地址，开销大，浪费带宽
 - 虚电路子网中，路由器需要维护虚电路的状态信息；
 - 地址查找时间与连接建立时间的权衡
 - 数据报子网对每个分组的路由查找过程复杂
 - 虚电路子网需要在建立连接时花费较长的路由查找时间
 - 可靠性与服务质量的权衡
 - 数据报不太容易保证服务质量QoS(Quality of Service)，但是对于通信线路的故障，适应性很强
 - 虚电路方式很容易保证服务质量，适用于实时操作，但比较脆弱
 - 可扩展性
 - 数据报子网具有更好的可扩展性



数据报子网与虚电路子网（续）

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.



路由算法

- 路由算法是网络层协议的一部分
 - 通信子网采用数据报分组交换方式，每个分组都要做路由选择
 - 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)
 - 可扩展性 (scalability)



路由算法分类

- 路由算法分类
 - 静态路由算法 (非自适应算法)
 - 动态路由算法 (自适应算法)
- 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

- 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。

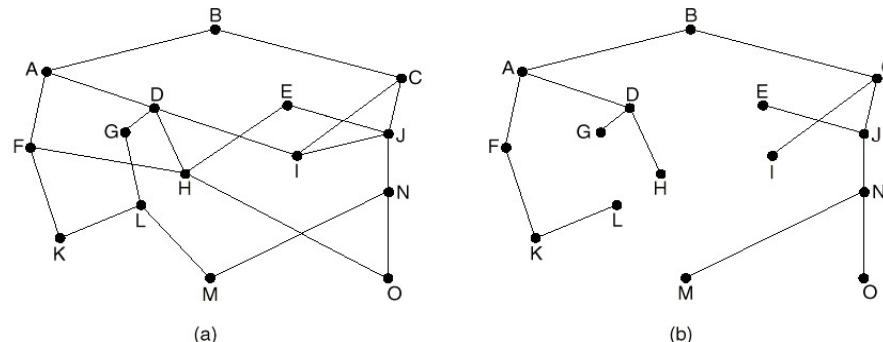


Fig. 5-5. (a) A subnet. (b) A sink tree for router B.



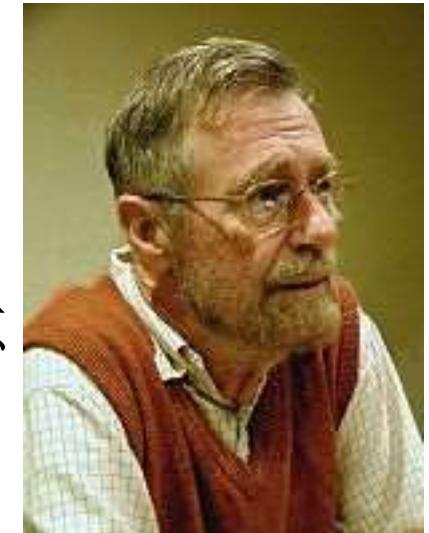
最短路径路由算法

- 最短路径路由算法 (**Shortest Path Routing**)
- 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 (hop)
 - 信道带宽
 - 地理距离
 - 传输延迟
 - 距离、信道带宽等参数的加权函数



Dijkstra 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



Edsger Wybe
Dijkstra,
(1930.5.11 –
2002.8.6), 荷
兰计算机科学家,
1972年获得图灵
奖

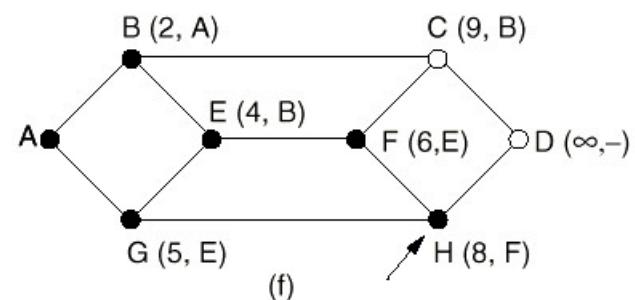
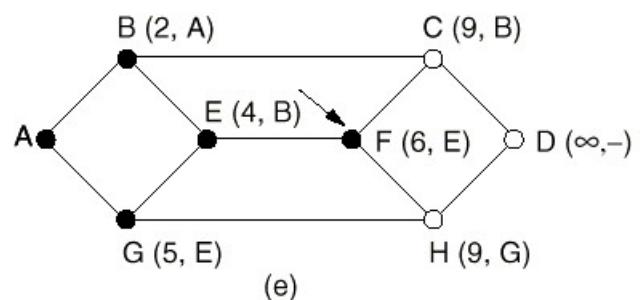
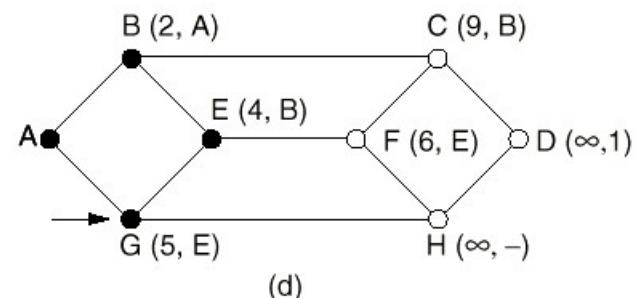
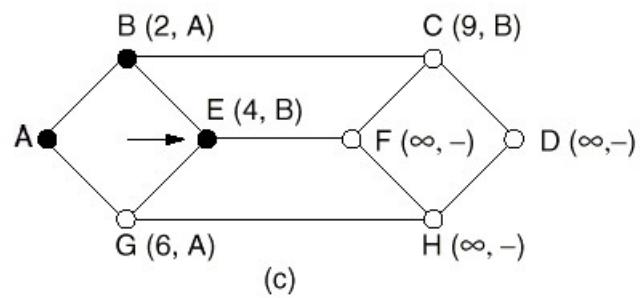
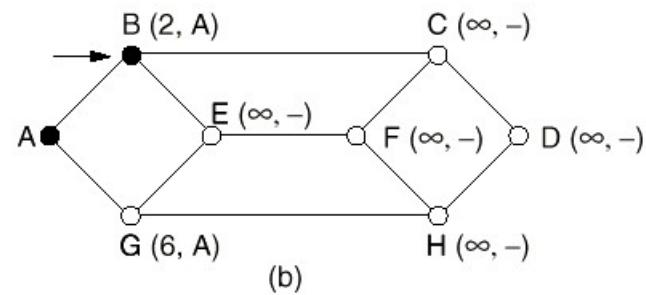
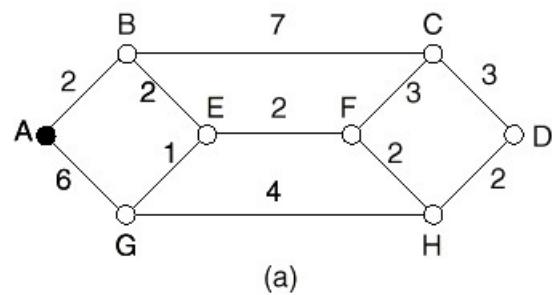


Fig. 5-6. The first five steps used in computing the shortest path from A to D . The arrows indicate the working node.



洪泛算法

- 洪泛算法（**Flooding**）属于静态路由算法
- 基本思想
 - 把收到的每一个分组，向除了该分组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分组，可能产生回路（loop）
- 解决措施
 - 每个分组头含站点计数器，每经过一站计数器减1，为0时则丢弃该分组
 - 记录分组经过的路径



洪泛算法（续）

- 选择性洪泛算法 (**selective flooding**)
 - 洪泛法的一种改进，将接收的每个分组仅发送到与正确方向接近的线路上
- 算法特点
 - 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



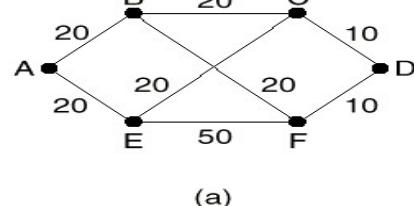
基于流量的路由算法

- 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，又兼顾网络负载
 - 前提：每对结点间平均数据流相对稳定和可预测
 - 根据网络带宽和平均流量，可得出平均分组延迟，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
 - 可用于流量工程 (traffic engineering)



基于流量的路由算法（续）

- 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）

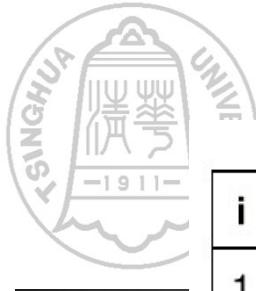


(a)

		Destination					
		A	B	C	D	E	F
Source	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 $T = 1 / (\mu C - \lambda)$



距离向量路由算法

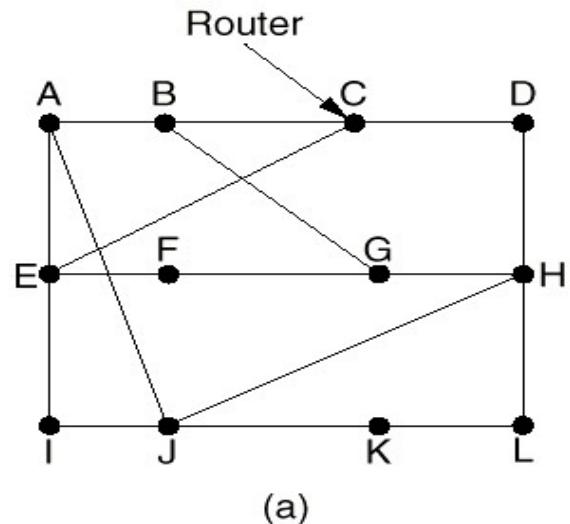
- 距离向量路由算法（**Distance Vector Routing**）
- 属于动态路由算法
- 也称**Bellman-Ford**路由算法或**Ford-Fulkerson**算法
- 最初用于**ARPANET**，被**RIP**协议采用



距离向量路由算法（续）

■ 算法步骤

- 每个路由器维护一张表，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，表项分组括两部分：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
- 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



New estimated delay from J

Line

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 .



距离向量路由算法（续）

- 无穷计算问题(**count-to-infinity problem**)
 - 对好消息反应迅速，对坏消息反应迟钝

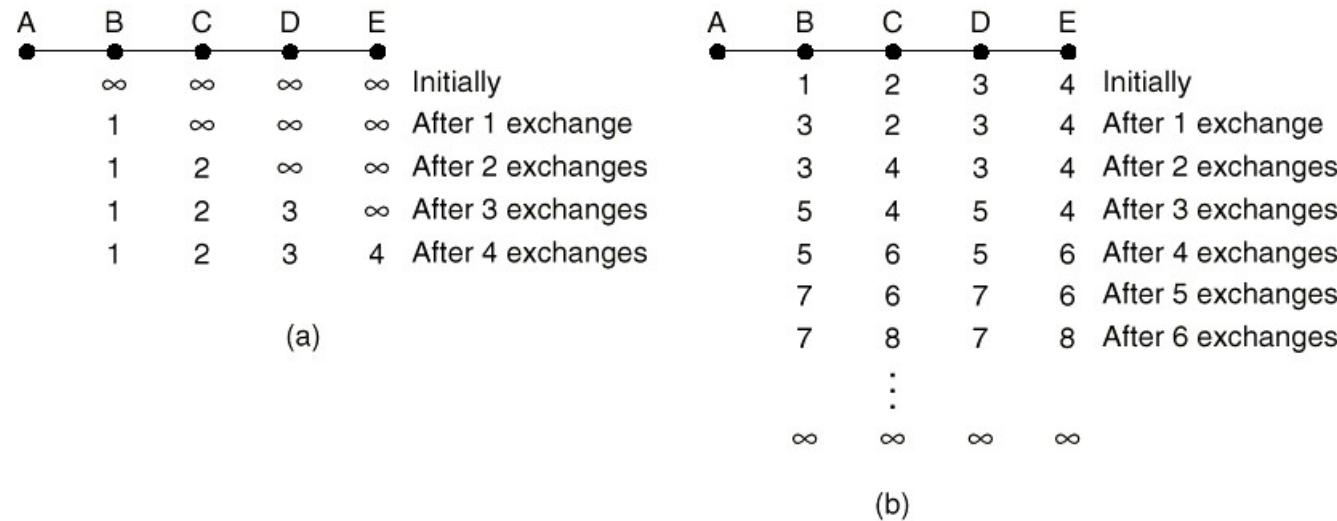
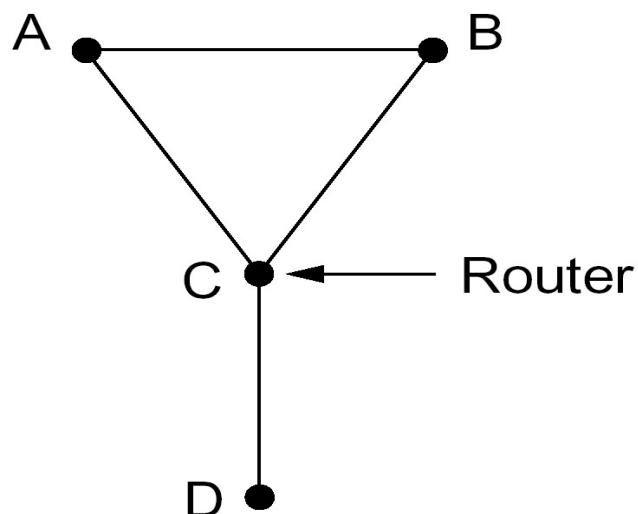


Fig. 5-11. The count-to-infinity problem.



距离向量路由算法（续）

- 水平分裂算法
 - 工作过程与距离向量算法相同，区别在于从邻居结点学到的到X的距离不向邻居结点报告，使得坏消息传播的也快
 - 虽然广泛使用，但有时候会失败





链路状态路由算法

- 链路状态路由算法 (**Link State Routing**)
- 距离向量路由算法的主要问题
 - 选择路由时，没有考虑链路带宽
 - 路由收敛速度慢
 - 存在无穷计算问题
 - 路由报文开销大（不是增量更新）
 - 不适合用于大规模网络（RIP协议最大支持15跳）



链路状态路由算法（续）

■ 算法步骤

- 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分组发现邻居结点
 - 两个或多个路由器连在一个LAN时，引入人工结点（代表路由器DR, Designated Router）
- 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分组，往返时间除以2即为延迟
 - 另一种方法是根据带宽来设定

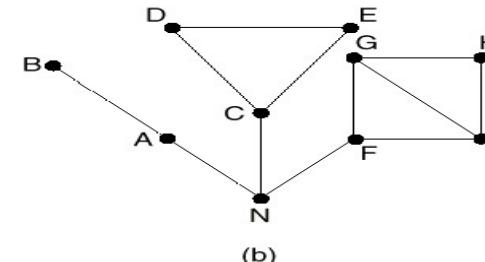
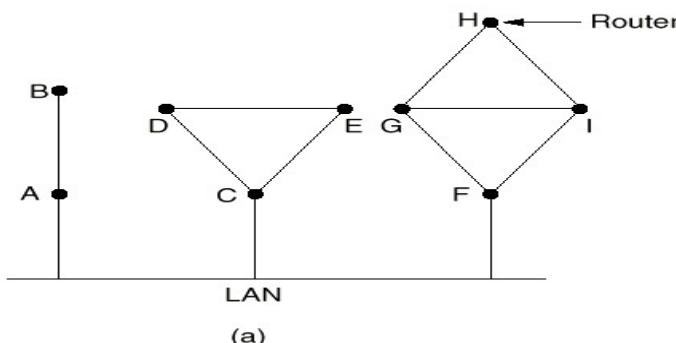
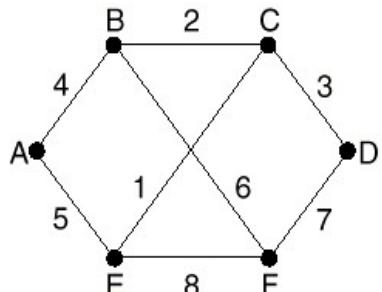


Fig. 5-13. (a) Nine routers and a LAN. (b) A graph model of (a).



链路状态路由算法（续）

- 将所有学习到的内容封装成一个分组
 - 分组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
 - 链路状态分组定期创建或发生重大事件时创建



(a)

Link	State	Packets
A	B	E
	Seq.	Seq.
	Age	Seq.
	B 4	A 4
	E 5	C 2
B	C	D
Seq.	Seq.	Seq.
Age	Age	Age
A 4	B 2	C 3
C 2	D 3	F 7
F 6	E 1	

(b)

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



链路状态路由算法（续）

- 将这个分组发送给所有其它路由器
 - 洪泛链路状态分组，为控制洪泛，每个分组分组含有一个序号，每次发送新分组时加1
 - 路由器记录信息对（源路由器，序号），当一个链路状态分组到达时，若是新分组，则处理；若是重复分组或过时分组，则丢弃
 - 问题
 - 序号循环使用会混淆
 - 路由器崩溃后，序号重置
 - 序号出错



链路状态路由算法（续）

■ 改进

- 第一个问题的解决办法：使用32位序号
- 第二、三个问题的解决办法：增加年龄（age）域，每秒钟年龄减1，为零则丢弃（OSPF协议规定，age初始值为0，递增，超过MaxAge（一般为3600秒）要丢弃）
- 链路状态分组到达后，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分组比较序号，丢弃重复分组，保留新分组
- 链路状态分组需要应答

■ 计算到所有其它路由器的最短路径

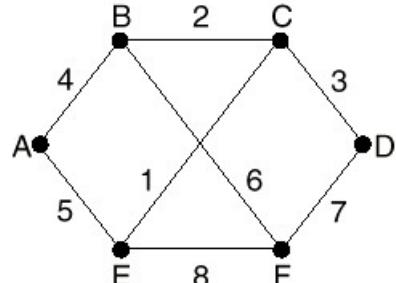
■ 根据Dijkstra算法计算最短路径

■ 实用协议

■ OSPF, IS-IS



例子：路由器B中的链路状态信息



Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
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**发来的链路状态分组有两个，一个经过**EAB**，另一个经过**EFB**
- 从**D**发链路状态分组有两个，一个经过**DCB**，另一个经过**DFB**



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 个节点, E 个链路的情况下, 发送 $O(NE)$ 个报文
 - DV
 - 仅在邻居节点之间交换
 - 注意
 - LS发送的是链路信息, DV发送的是到所有结点的向量信息
 - LS信息定期创建 (30分钟) 或发生重大事件时创建, DV定期创建 (30秒钟)
 - LS发布增量信息, DV发布全部信息



链路状态算法 (LS) 和距离向量 算法 (DV) 的比较 (续)

■ 收敛 (Convergence) 速度

■ LS

- 使用最短路径优先算法，算法复杂度为 $O(n \log n)$

- n 个结点（不分组括源结点），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 $O(n \log n)$

- 可能存在路由振荡 (oscillations)

■ DV

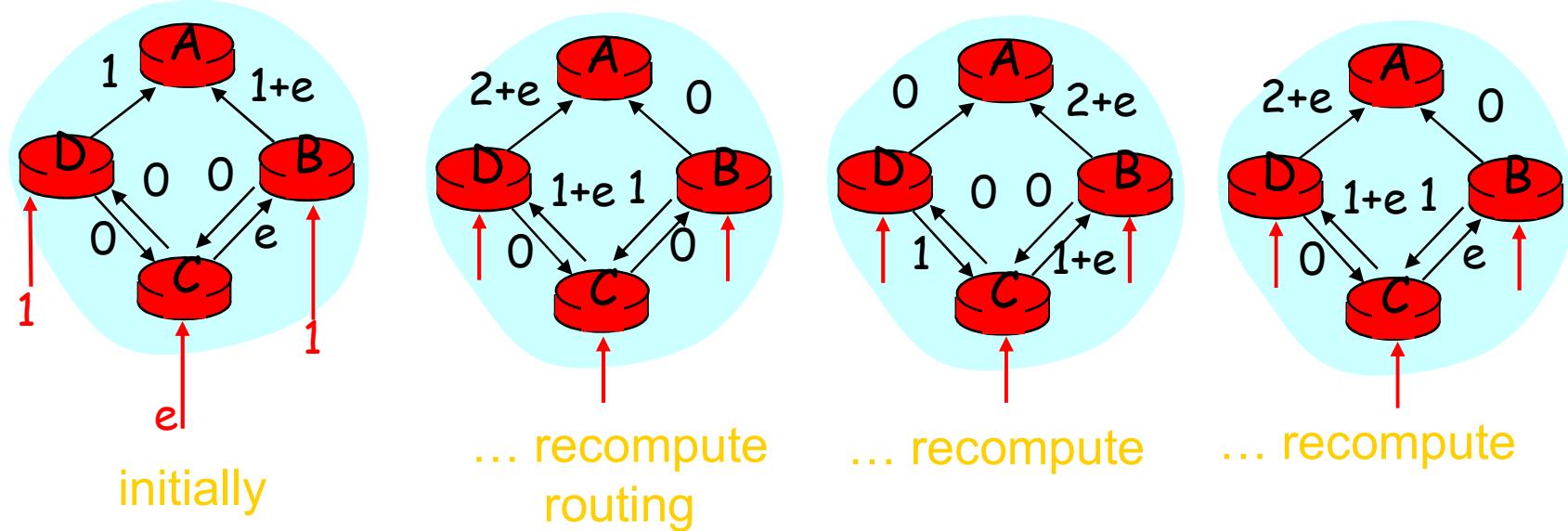
- 收敛时间不确定

- 可能会出现路由循环
 - count-to-infinity 问题



路由振荡 (oscillations)

- 例如, link cost = amount of carried traffic





链路状态算法 (LS) 和距离向量 算法 (DV) 的比较 (续)

- 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

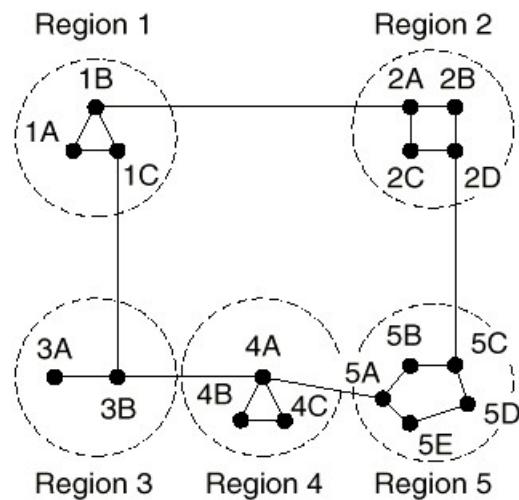


分层路由

- 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
 - 路由收敛慢
- 分层路由
 - 分而治之的思想
 - 根据需要，将网络分成若干域（domain）
 - 路由表规模大幅减少
 - Fig. 5-17, 路由表由17项减为7项。
- 分层路由带来的问题
 - 分层后计算得到的路由不一定是最优路由



分层路由（续）



(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)



移动主机的路由

- 需要解决的问题
 - 为了能够将分组转发给移动主机，网络必须首先要找到移动的主机
- 网络结构示意图

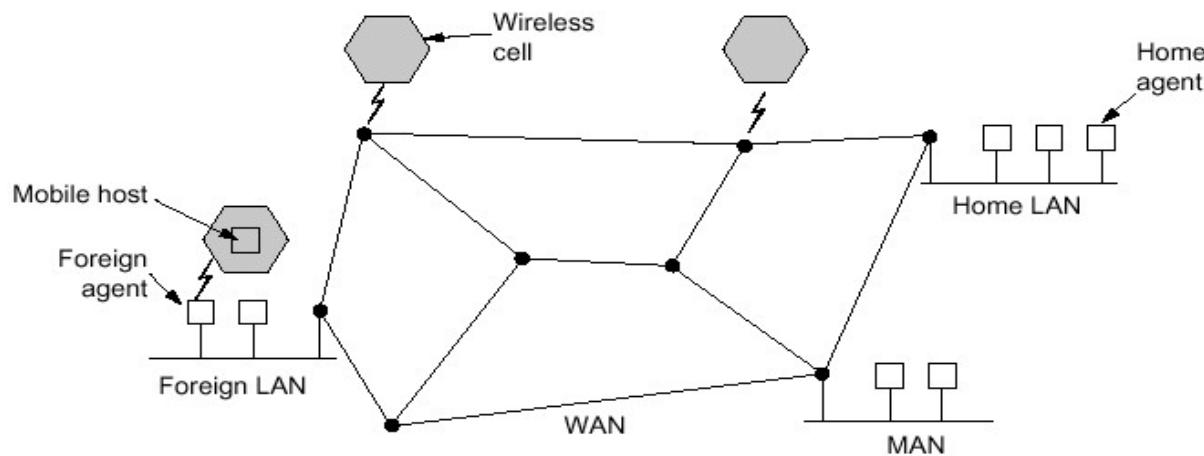


Fig. 5-18. A WAN to which LANs, MANs, and wireless cells are attached.



移动主机的路由（续）

■ 一些基本概念

- 移动用户 (mobile users) : 位置发生变化, 包括通过固定方式或移动方式与网络连接的两类用户
- 家乡位置 (home location) : 所有用户都有一个永久的家乡位置, 用一个地址来标识
- 家乡代理 (home agent) : 每个区域有一个家乡代理, 负责记录家乡在该区域, 但是目前正在访问其它区域的用户
- 外部代理 (foreign agent) : 每个区域 (一个LAN或一个wireless cell) 有一个或多个外部代理, 它们记录正在访问该区域的移动用户



移动主机的路由（续）

- 移动用户进入一个新区域时，必须首先向外部代理注册
 - 外部代理定期广播声明自己的存在和地址的分组，新到达的移动主机接收该信息；若移动用户未能收到该信息，则移动主机广播分组，询问外部代理的地址
 - 移动主机向外部代理注册，告知其家乡地址、目前的数据链路层地址和一些安全信息
 - 外部代理与移动主机的家乡代理联系，告知移动主机的当前位置、自己的网络地址和一些安全信息
 - 家乡代理检查安全信息，通过，则给外部代理确认
 - 外部代理收到确认后，在登记表中加入一项，并通知移动主机注册成功

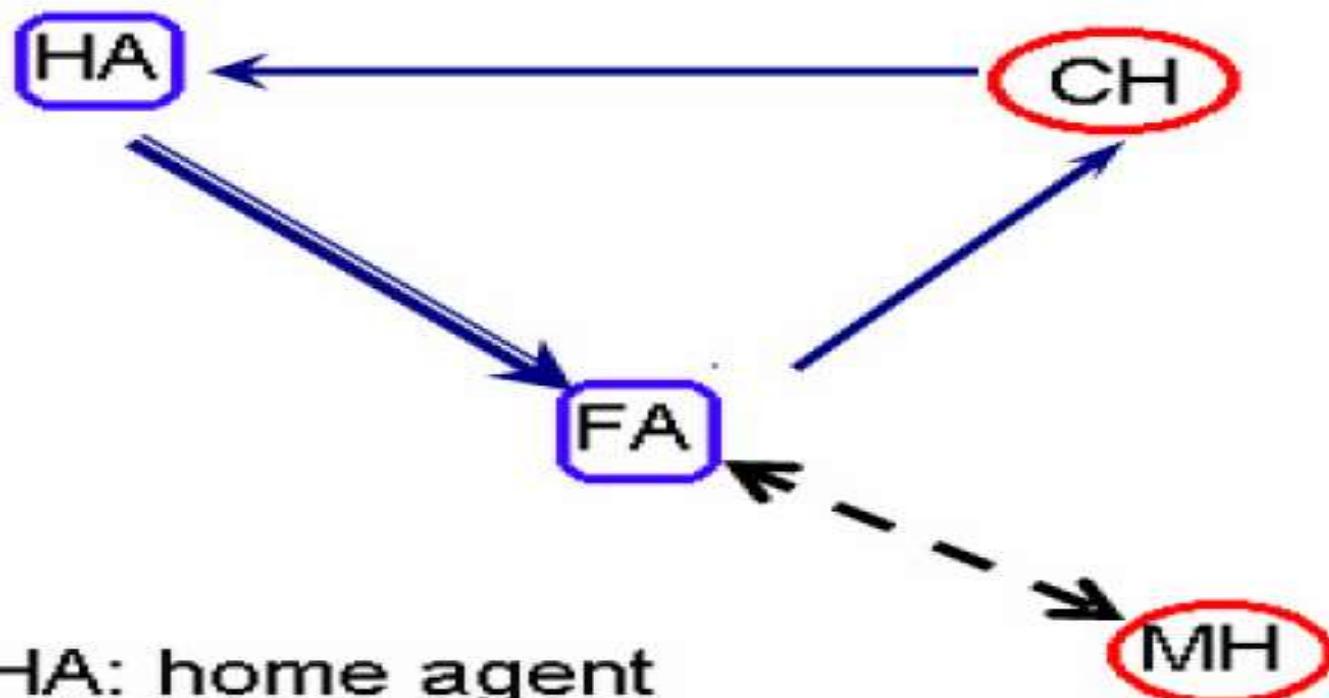


移动主机的路由（续）

- 移动用户的路由转发过程
 - 当一个分组发给移动用户时，首先被转发到用户的家乡局域网
 - 该分组到达用户的家乡局域网后，被家乡代理接收，家乡代理查询移动用户的新位置和与其对应的外部代理的地址
 - 家乡代理采用隧道技术，将收到的分组作为净负荷封装到一个新分组中，发给外部代理
 - 家乡代理告诉发送方，发给移动用户的后续分组作为净荷封装成分组直接发给外部代理（发送方要修改协议栈）
 - 外部代理收到分组后，将净负荷封装成数据链路帧发给移动用户



三角路由(Triangle Routing)



HA: home agent

FA: foreign agent

CH: correspondent host

MH: mobile host



路由算法小结

- 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



路由算法小结（续）

- 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，水平分裂算法
 - 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
- 移动主机的路由



拥塞控制算法

- 拥塞（congestion）
 - 网络上有太多的分组时，性能会下降，这种情况称为拥塞
- 拥塞产生的原因
 - 网络设备处理器性能低
 - 高速端口输入，低速端口输出
 - 多个输入对应一个输出

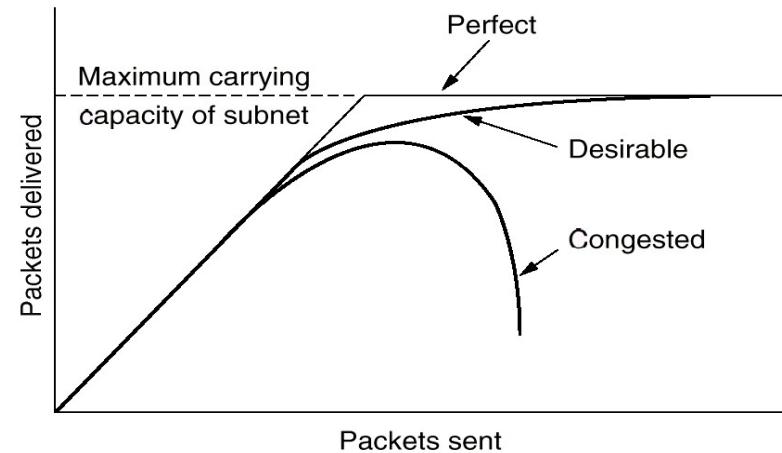


Fig. 5-22. When too much traffic is offered, congestion sets in and performance degrades sharply.



拥塞控制算法（续）

- 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，因此需要全面考虑各个因素
 - 目前，主要在网络层和传输层（TCP）进行控制
- 拥塞控制与流控制的差别
 - 拥塞控制（congestion control）需要确保通信子网能够承载用户提交的数据流，是一个全局性问题，涉及主机、路由器等很多因素
 - 流控制（flow control）与点到点的传输有关，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



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



拥塞控制的基本原理（续）

- 衡量网络是否拥塞的参数
 - 缺乏缓冲区造成的分组丢失率
 - 平均队列长度
 - 超时重传的分组的数目
 - 平均分组延迟
 - 分组延迟变化 (Jitter)
- 反馈方法
 - 向负载发生源发送一个告警分组
 - 分组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分组置位，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



拥塞预防策略

- 开环控制
- 影响拥塞的网络设计策略

Layer	Policies
Transport	<ul style="list-style-type: none">• Retransmission policy• Out-of-order caching policy• Acknowledgement policy• Flow control policy• Timeout determination
Network	<ul style="list-style-type: none">• Virtual circuits versus datagram inside the subnet• Packet queueing and service policy• Packet discard policy• Routing algorithm• Packet lifetime management
Data link	<ul style="list-style-type: none">• Retransmission policy• Out-of-order caching policy• Acknowledgement policy• Flow control policy

Fig. 5-23. Policies that affect congestion.



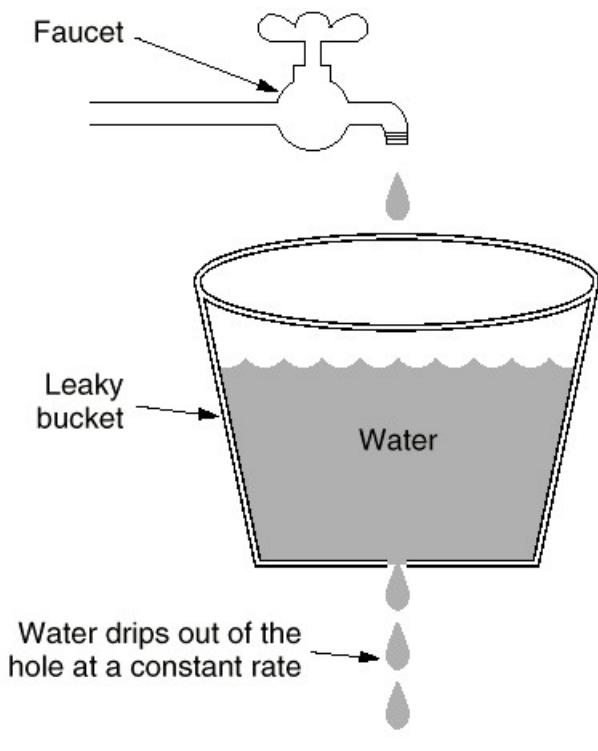
流量整形(Traffic Shaping)

- 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
 - 强迫分组以一种可预测的速率发送
- 典型算法
 - 漏桶算法
 - 令牌桶算法

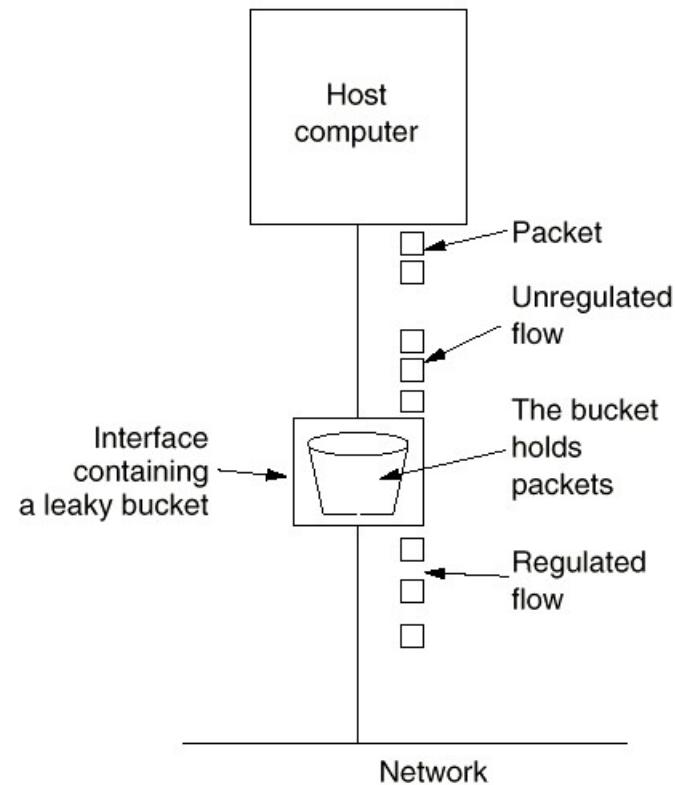


漏桶算法(Leaky Bucket)

- 将用户发出的不平滑的数据分组流转变成网络中平滑的数据分组流
- 可用于固定分组长的协议，如**ATM**；也可用于可变分组长的协议，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



(a)



(b)

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



令牌桶算法(Token Bucket)

- 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。分组传输之前必须获得一个令牌，传输之后删除该令牌

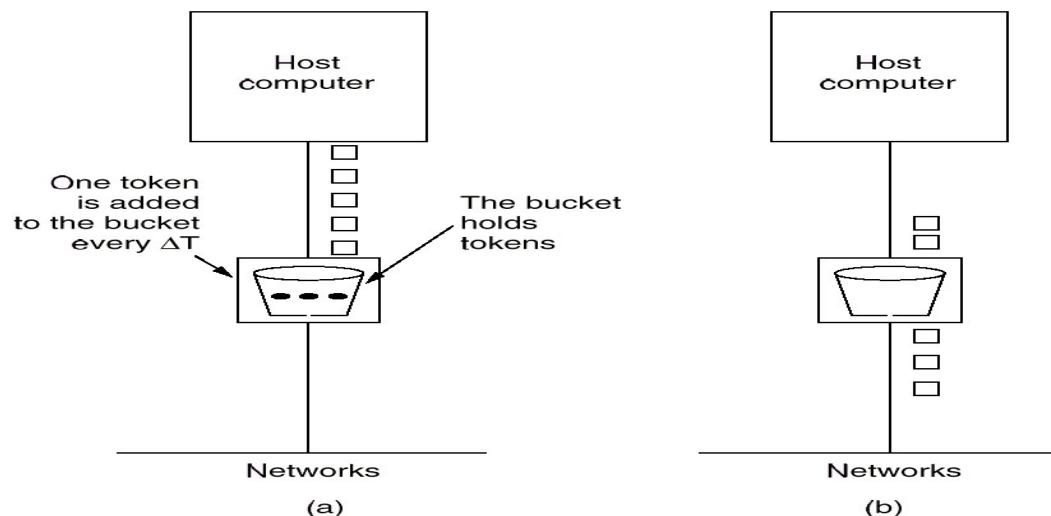


Fig. 5-26. The token bucket algorithm. (a) Before. (b) After.



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
 - 令牌桶算法允许空闲主机积累发送权，最大为桶的大小
- 漏桶维护策略不同
 - 漏桶算法中，漏桶存放的是数据分组，桶满了丢弃数据分组
 - 令牌桶算法中，漏桶存放的是令牌，桶满了丢弃令牌

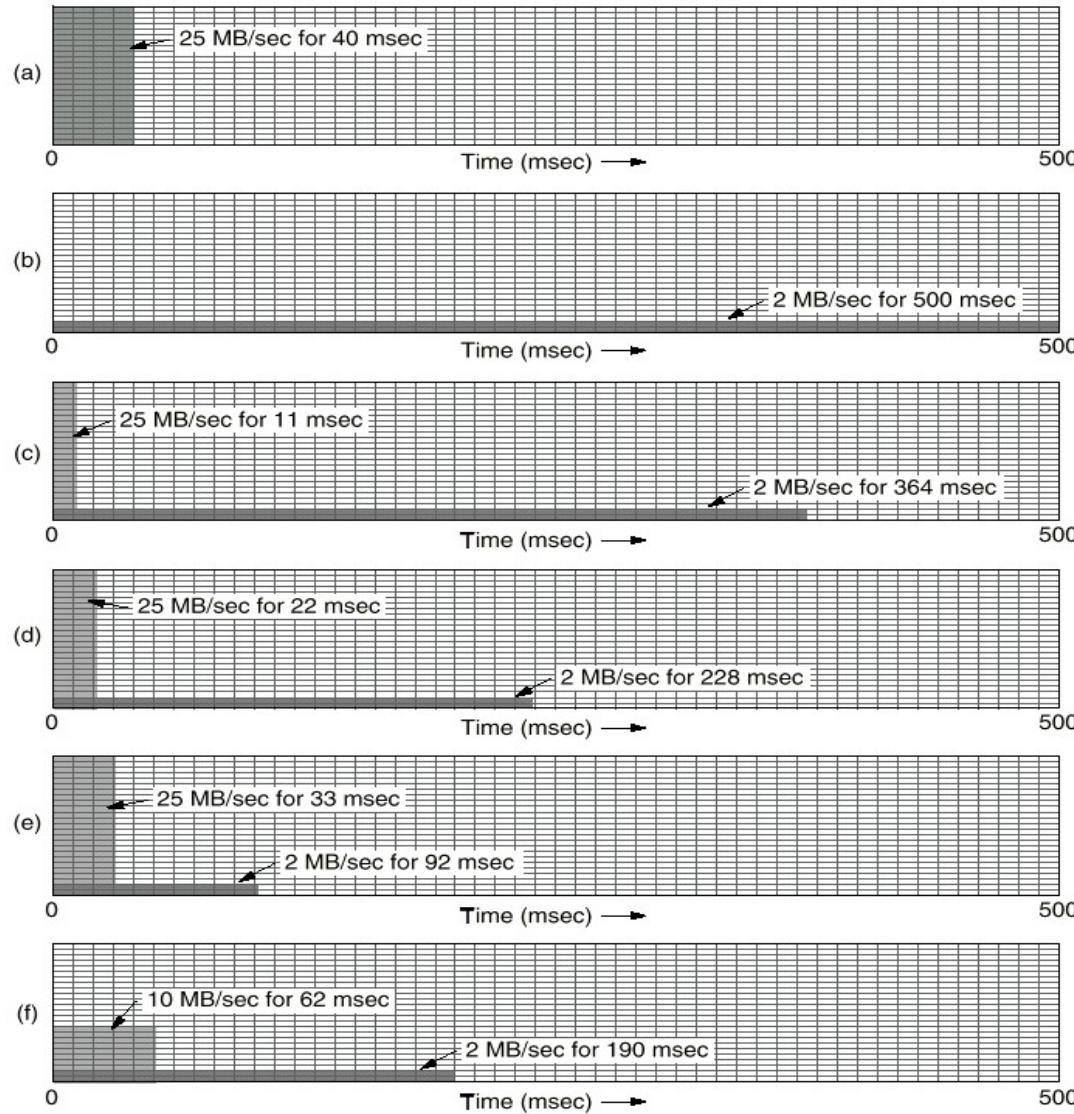


Fig. 5-25. (a) Input to a leaky bucket. (b) Output from a leaky bucket. (c) - (e) Output from a token bucket with capacities of 250KB, 500KB, and 750KB. (f) Output from a 500KB token bucket feeding a 10 MB/sec leaky bucket.



虚电路子网中的拥塞控制

■ 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 (μ sec)
Token bucket size (bytes)	Burst loss sensitivity (packets)
Maximum transmission rate (bytes/sec)	Minimum delay noticed (μ sec)
	Maximum delay variation (μ sec)
	Quality of guarantee

Fig. 5-27. An example flow specification.



虚电路子网中的拥塞控制

- 准入控制（admission control）
 - 根据流说明和网络资源分配情况，进行准入控制
 - 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

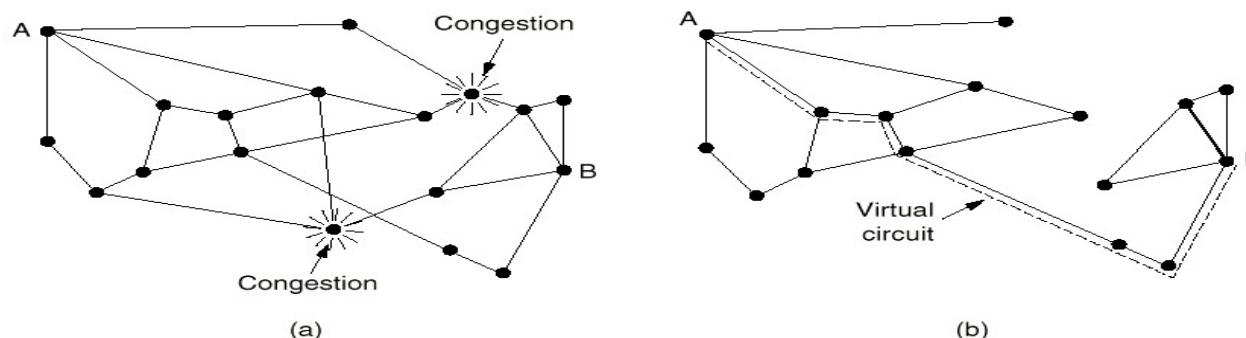


Fig. 5-28. (a) A congested subnet. (b) A redrawn subnet that eliminates the congestion and a virtual circuit from *A* to *B*.



抑制分组(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
- 每个新分组到来，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分组，分组中指出发生拥塞的目的地址。同时将原分组打上标记（为了以后不再产生抑制分组），正常转发
- 源主机收到抑制分组后，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分组。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、忽略抑制分组；若在监听周期内没有收到抑制分组，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(AIMD)



逐跳抑制分组 (Hop-by-Hop Choke Packets)

- 在高速、长距离的网络中，由于源主机响应太慢，抑制分组算法对拥塞控制的效果并不好，可采用逐跳抑制分组算法
- 基本思想
 - 抑制分组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

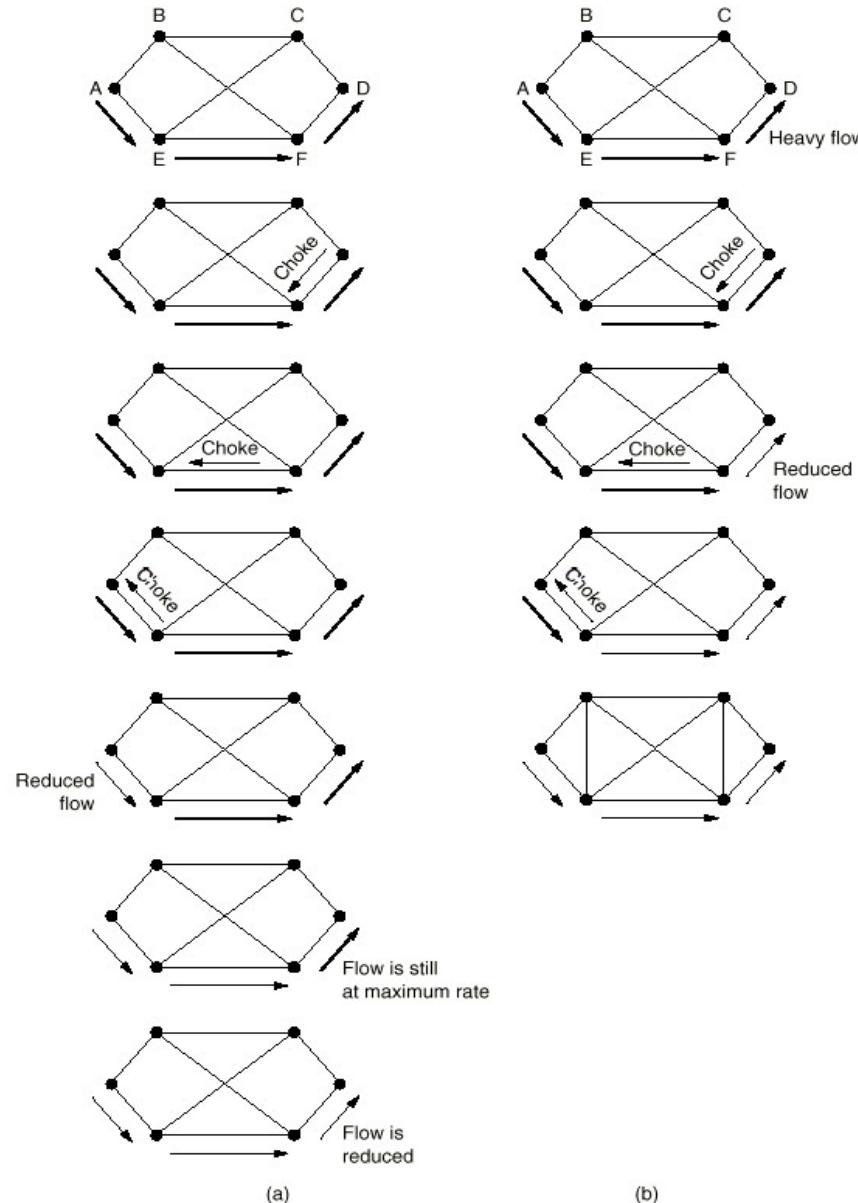


Fig. 5-30. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.



公平队列算法 (Fair Queueing)

- 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，发送队头的分组
- 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：对于变长分组，由逐分组轮讯改为逐字节轮讯

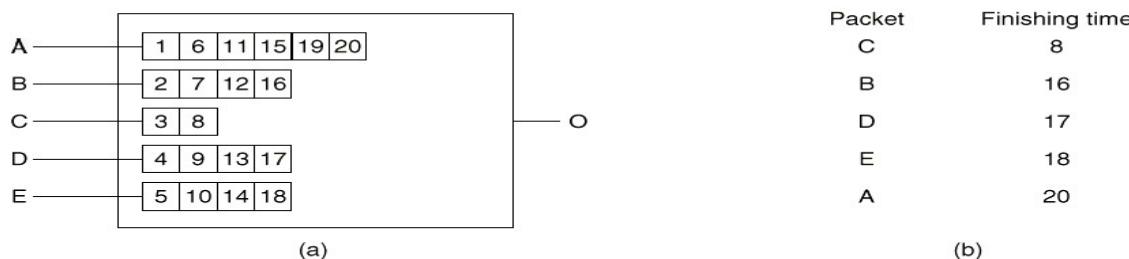


Fig. 5-29. (a) A router with five packets queued for line O . (b) Finishing times for the five packets.



加权公平队列 (Weighted Fair Queueing)

- 给不同队列以不同的优先级
- 优先级高的队列在一个轮询周期内获得更多的时间片



负载丢弃(Load Shedding)

- 上述算法都不能消除拥塞时，路由器只得将分组丢弃
- 针对不同服务，可采取不同丢弃策略
 - **文件传输，优先丢弃新分组，wine策略**
 - **多媒体服务，优先丢弃旧分组，milk策略**
- 早期丢弃分组，会减少拥塞发生的概率，提高网络性能



网络互联

- 互连网络（**internet**）：两个或多个网络构成互连网络

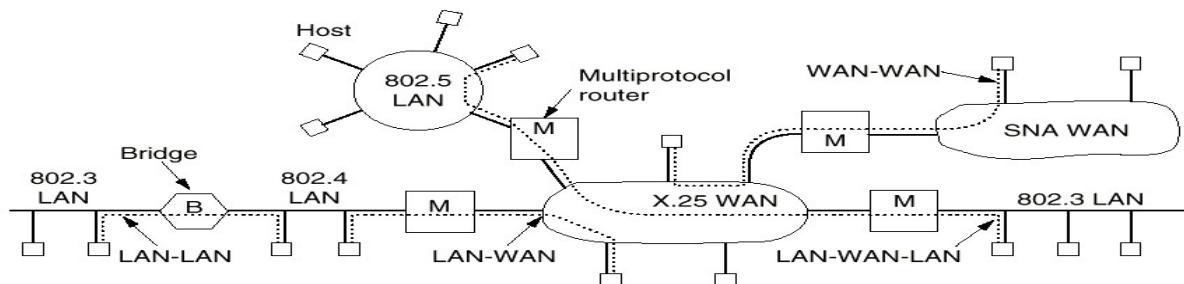


Fig. 5-33. Network interconnection.

- 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



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.



网络互连设备

■ 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分组
 - 必要时，做网络层协议转换



网络互连设备（续）

- **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



无连接网络互连

- 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
- 每个分组单独路由，提高网络利用率
- 根据需要，连接不同子网的多协议路由器做协议转换，分组括分组格式转换和地址转换等

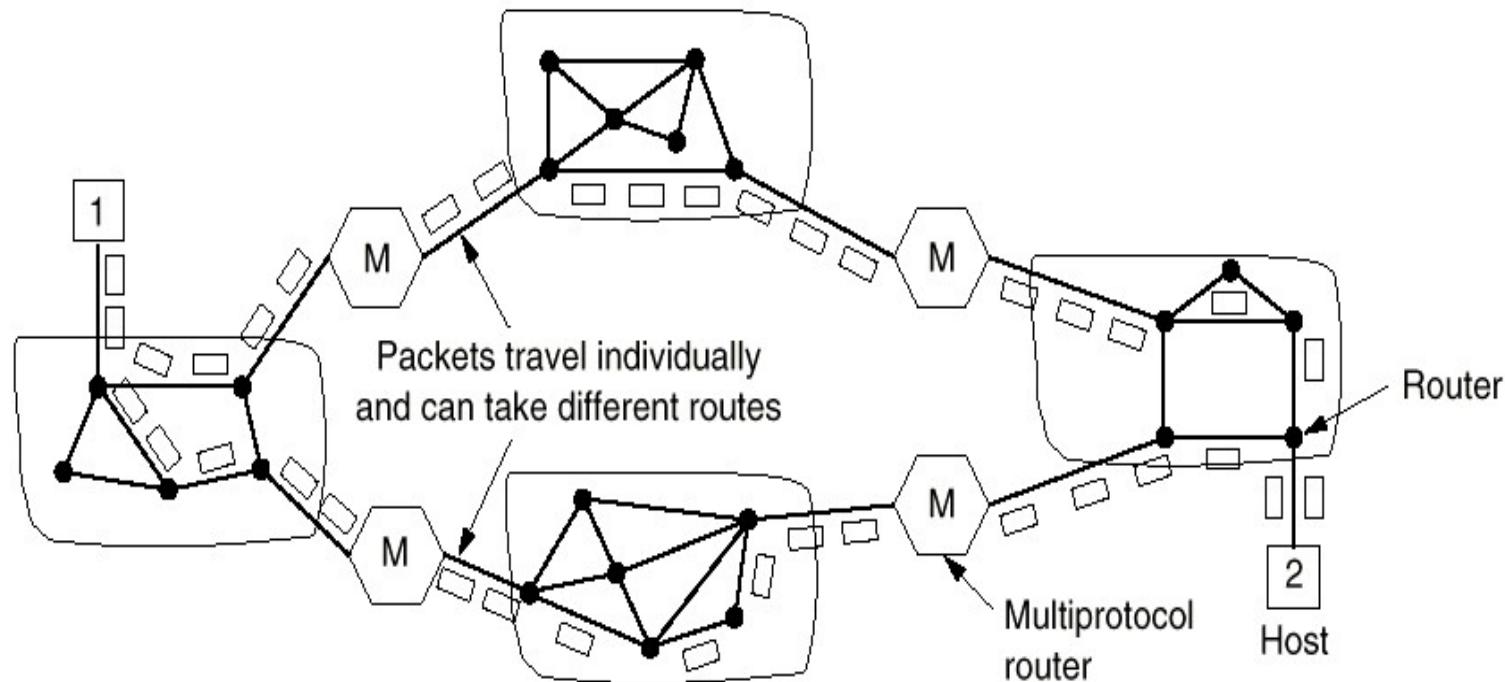
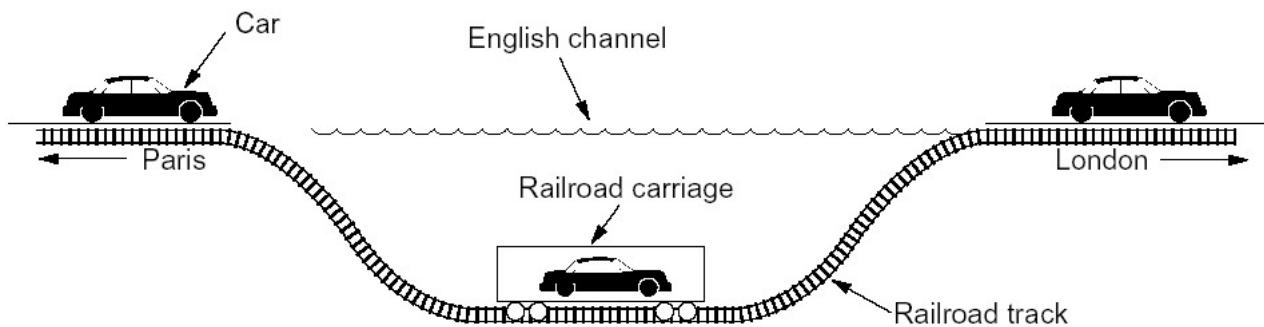


Fig. 5-37. A connectionless internet.



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





隧道技术（续）

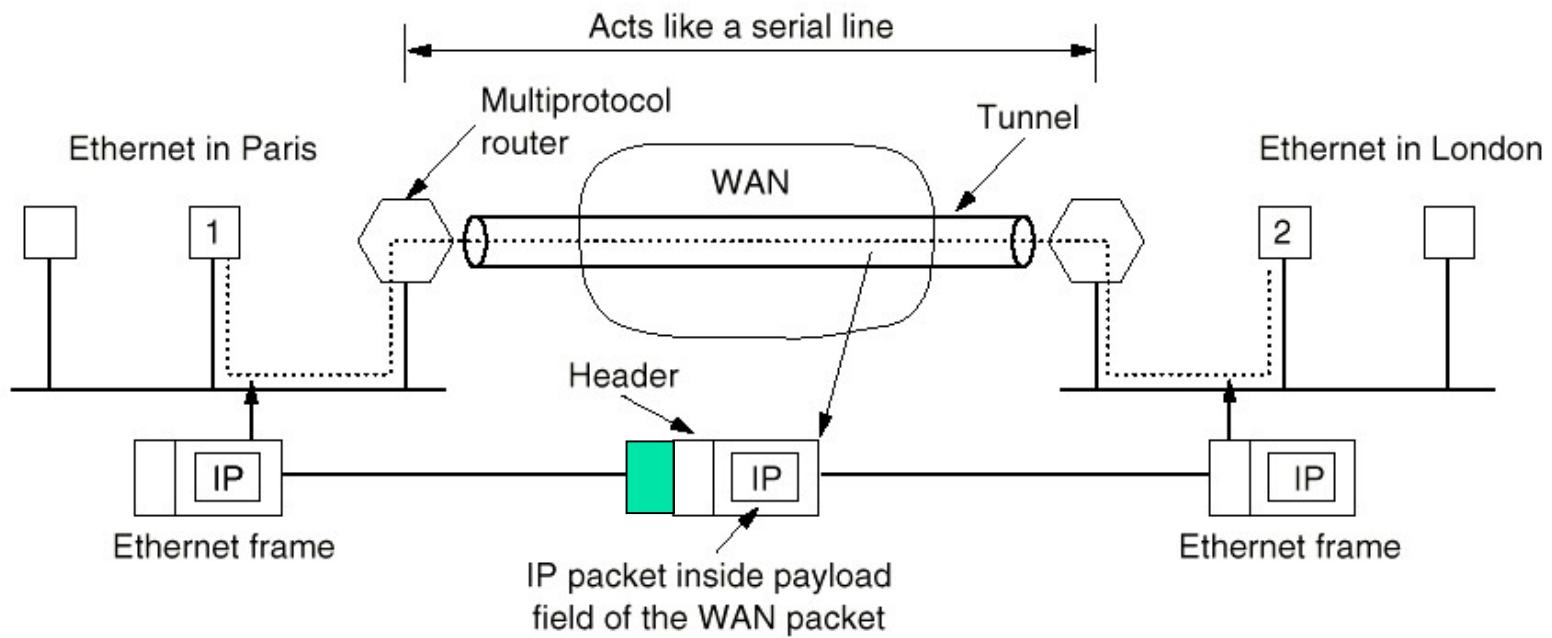


Fig. 5-38. Tunneling a packet from Paris to London.



隧道技术（续）

- 工作过程（以**Fig. 5-38**为例）
 - 主机1发送一个分组，目的IP地址 = 主机2-IP，将分组封装到局域网帧中，帧目的MAC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分组封装到广域网网络层分组中，分组目的IP地址 = 路由器2-IP
 - 广域网传输
 - 路由器2剥掉广域网分组头，将得到的IP分组封装到局域网帧中，帧目的MAC地址 = 主机2-MAC地址
 - 局域网传输
 - 主机2接收



互连网络路由

- 互连网络路由工作过程
 - 互连网络的路由与单独子网的路由过程相似，只是复杂性增加
- 两级路由算法
 - 自治系统AS
 - 内部网关协议IGP
 - RIP, OSPF
 - 外部网关协议EGP
 - BGP

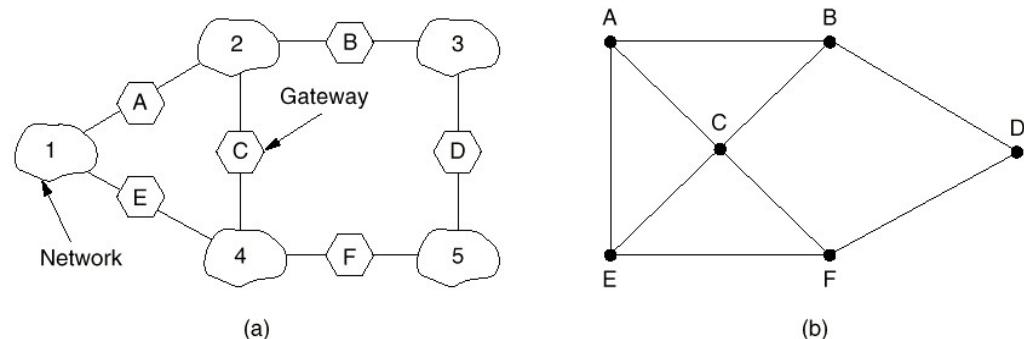


Fig. 5-40. (a) An internetwork. (b) A graph of the internetwork.



分片(Fragmentation)

- 每种网络都对最大分组长度有限制，有以下原因
 - 硬件，例如 TDM 的时槽限制
 - 操作系统
 - 协议，例如分组长度域的比特个数
 - 与标准的兼容性
 - 希望减少传输出错的概率
 - 希望避免一个分组占用信道时间过长
- 大分组经过小分组网络时，网关要将大分组分成若干片段（**fragment**），每个片段作为独立的分组传输



分片（续）

- 分片重组策略

- 分片重组过程对其它网络透明

- 网关将大分组分片后，每个片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片段都到齐
 - 所有片段必须从同一出口网关离开
 - 大分组经过一系列小分组网络时，需要反复分片重组，开销大

- 分片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
 - 每个片段都要有一个分组头，网络开销增大



分片 (续)

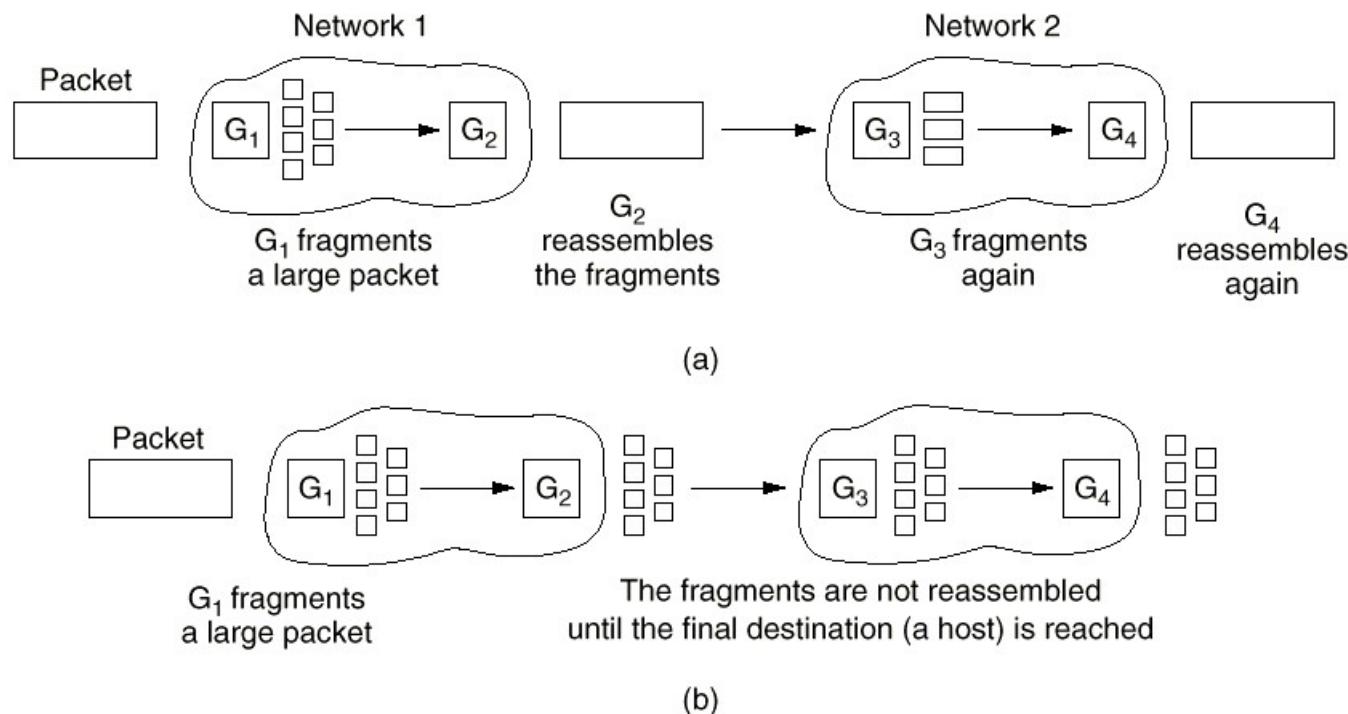


Fig. 5-41. (a) Transparent fragmentation. (b) Nontransparent fragmentation.



分片（续）

■ 标记片段

■ 树型标记法

- 分组0分成三段，分别标记为0.0, 0.1, 0.2，片段0.0构成的分组被分成三片，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
- 分片长度前后要一致

■ 偏移量法

- 定义一个基本片段长度，使得基本片段能够通过所有网络
- 分片时，除最后一个片段小于等于基本片段长度外，所有片段长度都等于基本片段长度
- 分片后的分组头中包括：原始分组序号，分组中第一个基本片段的偏移量，最后片段指示位

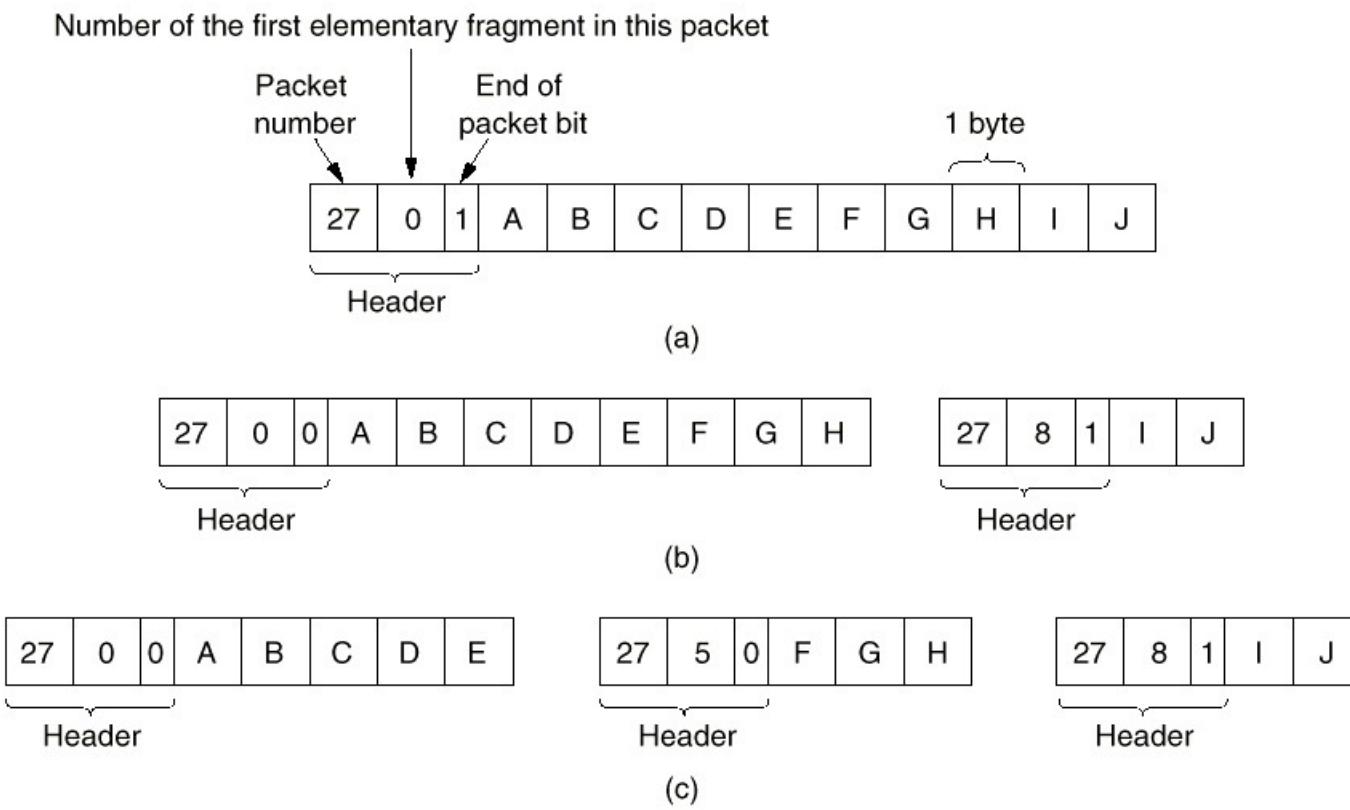


Fig. 5-42. Fragmentation when the elementary data size is 1 byte. (a) Original packet, containing 10 data bytes. (b) Fragments after passing through a network with maximum packet size of 8 bytes. (c) Fragments after passing through a size 5 gateway.



防火墙(Firewall)

- 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
- 防火墙的一种早期配置
 - 两个路由器，根据某种规则表，进行分组过滤
 - 一个应用网关，审查应用层信息

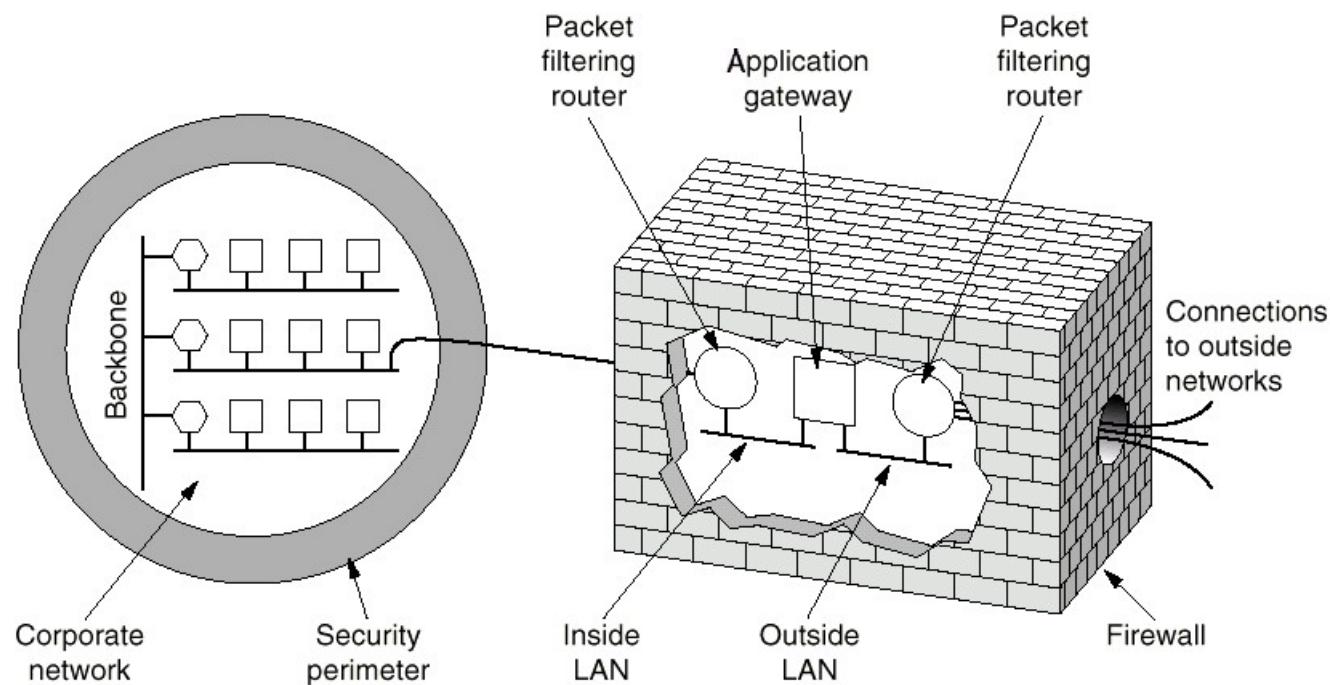


Fig. 5-43. A firewall consisting of two packet filters and an application gateway.



网络互连小结

- 互联网：两个或多个网络构成互联网
- 网络互连设备：中继器、网桥、路由器、传输网关、应用网关
- 隧道技术
- 分片和重组
- 防火墙



网络层协议

- 在网络层，互联网可以看成是自治系统的集合，是由网络组成的网络
- 网络之间互连的纽带是**IP (Internet Protocol)**协议

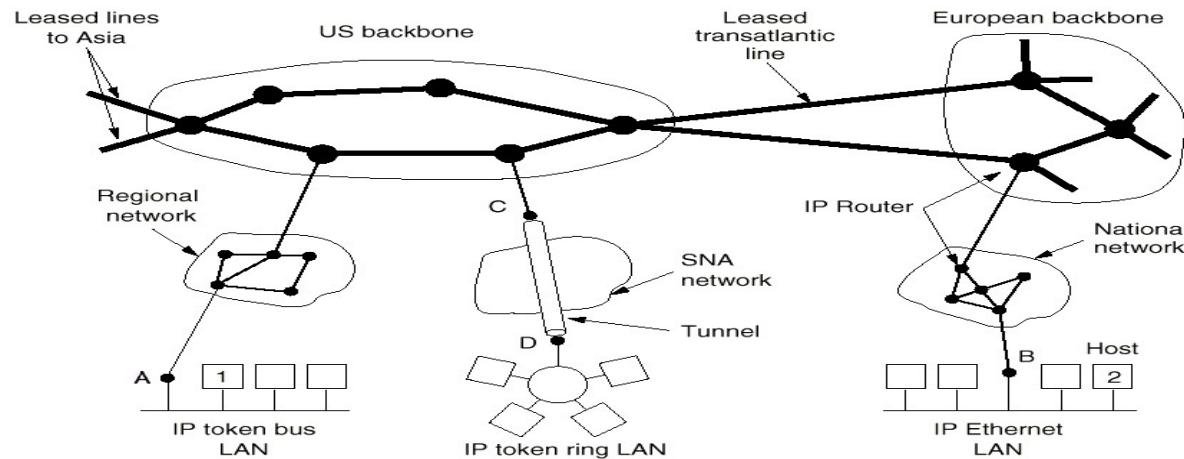


Fig. 5-44. The Internet is an interconnected collection of many networks.



IP协议

■ IP头格式

- IP头包括20个字节的固定部分和变长（最长40字节）的可选部分，从左到右传输

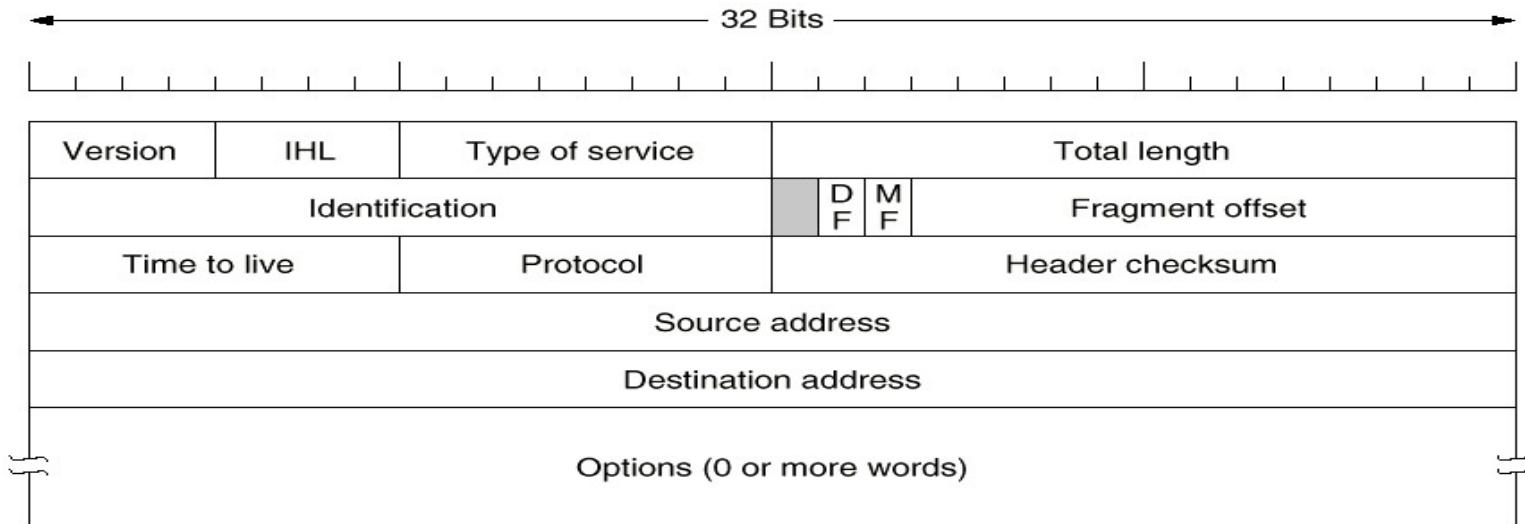


Fig. 5-45. The IP (Internet Protocol) header.



IP协议（续）

- 版本域 (version)
- IHL: IP分组头长度, 最小为5, 最大为15, 单位为32-bit word
- 服务类型域 (Type of Service)
 - 3个优先级位
 - 3个标志位: D (Delay) 、 T (Throughput) 、 R (Reliability)
 - 2个保留位
 - 目前, 很多路由器都忽略服务类型域
- 总长度域 (Total length)
- 标识域 (Identification)



IP协议（续）

- **DF: Don't Fragment**
 - 所有机器必须能够接收小于等于576字节的片段
- **MF: More Fragments**
 - 除最后一个片段外的所有片段都要置MF位
- **片段偏移量 (Fragment offset)**
 - 除最后一个片段外的所有片段的长度必须是8字节的倍数



IP协议（续）

	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

One large datagram becomes
several smaller datagrams

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	
	=1500	=x	=1	=1480	

	length	ID	fragflag	offset	
	=1040	=x	=0	=2960	



IP协议（续）

- 生存期 (Time to live)
 - 实际实现中，IP分组每经过一个路由器TTL减1，为0则丢弃，并给源主机发送一个告警分组
 - 最大值为255。源主机设定初始值，UNIX操作系统一般为255，Windows操作系统一般为128，Linux一般为64
- 协议域 (Protocol)：上层为哪种传输协议，TCP、UDP...
- 头校验和 (Header checksum)
 - 只对IP分组头做校验
 - 算法：每16位求反，循环相加（进位加在末尾），和再求反
- 源地址(Source address)和目的地址(Destination address)



IP协议（续）

■ 选项 (Options)

- 变长，长度为4字节的倍数，不够则填充，最长为40字节

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

Fig. 5-46. IP options.



IP地址

- 地址组成：网络号 + 主机号
- 有类地址划分(**classful addressing**)

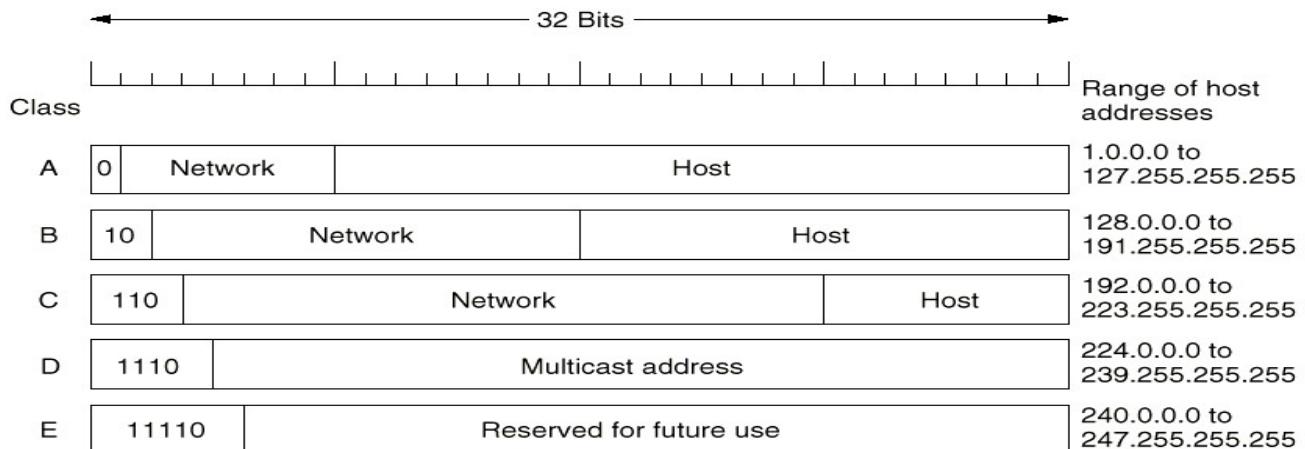


Fig. 5-47. IP address formats.

- 地址表示采用用点分隔的十进制表示法
 - 例如**166.111.68.3**



IP地址（续）

- 全0和全1有特殊含义
 - 全0：表示本网络或本主机
 - 全1：表示广播地址

Fig. 5-48. Special IP addresses.



子网

■ 子网(Subnet)

- 分而治之的思想：为了便于管理和使用，可以将网络分成若干供内部使用的部分，称为子网

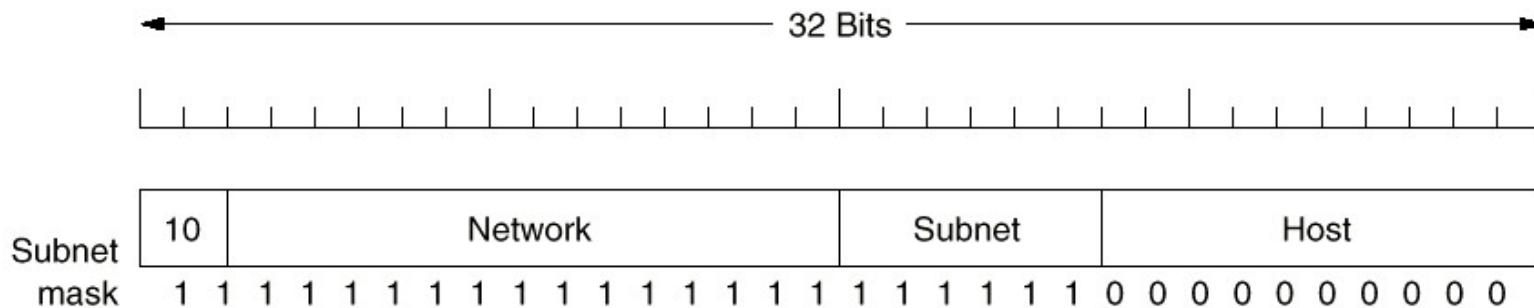


Fig. 5-49. One of the ways to subnet a class B network.



ICMP协议

- 互联网控制消息协议**ICMP**(Internet Control Message Protocol)

- 主要用来报告错误和测试
- 报文类型
- ICMP报文封装在IP分组中

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Fig. 5-50. The principal ICMP message types.



ARP协议

■ 地址解析协议ARP (Address Resolution Protocol)

- 解决网络层地址 (IP地址) 与数据链路层地址 (MAC地址) 的映射问题
- 工作过程
 - 建立一个ARP表，表中存放(IP地址, MAC地址)对
 - 若目的主机在同一子网内，用目的IP地址在ARP表中查找，否则用缺省网关的IP地址在ARP表中查找。
 - 若未找到，则发送广播分组，目的主机收到后给出应答，ARP表增加一项
 - 每个主机启动时，广播它的(IP地址, MAC地址)映射
 - ARP表中的表项有生存期，超时则删除



ARP协议（续）

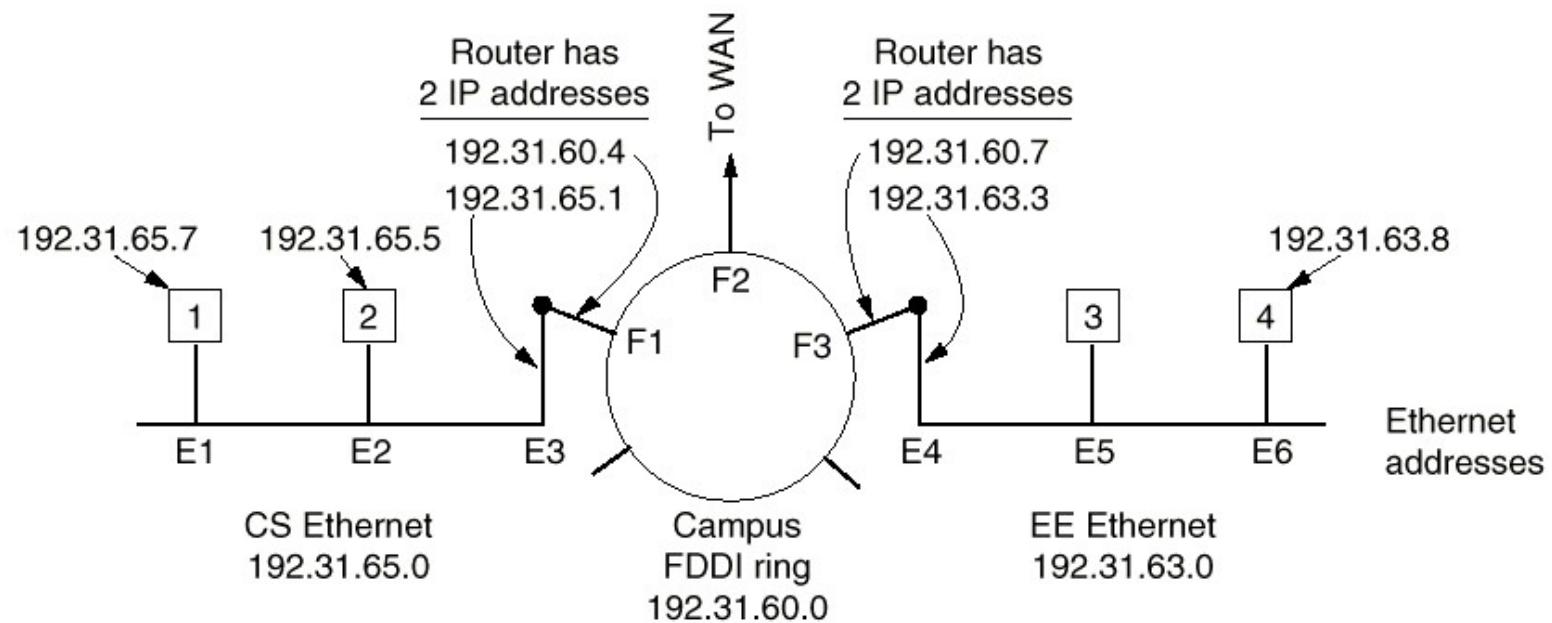
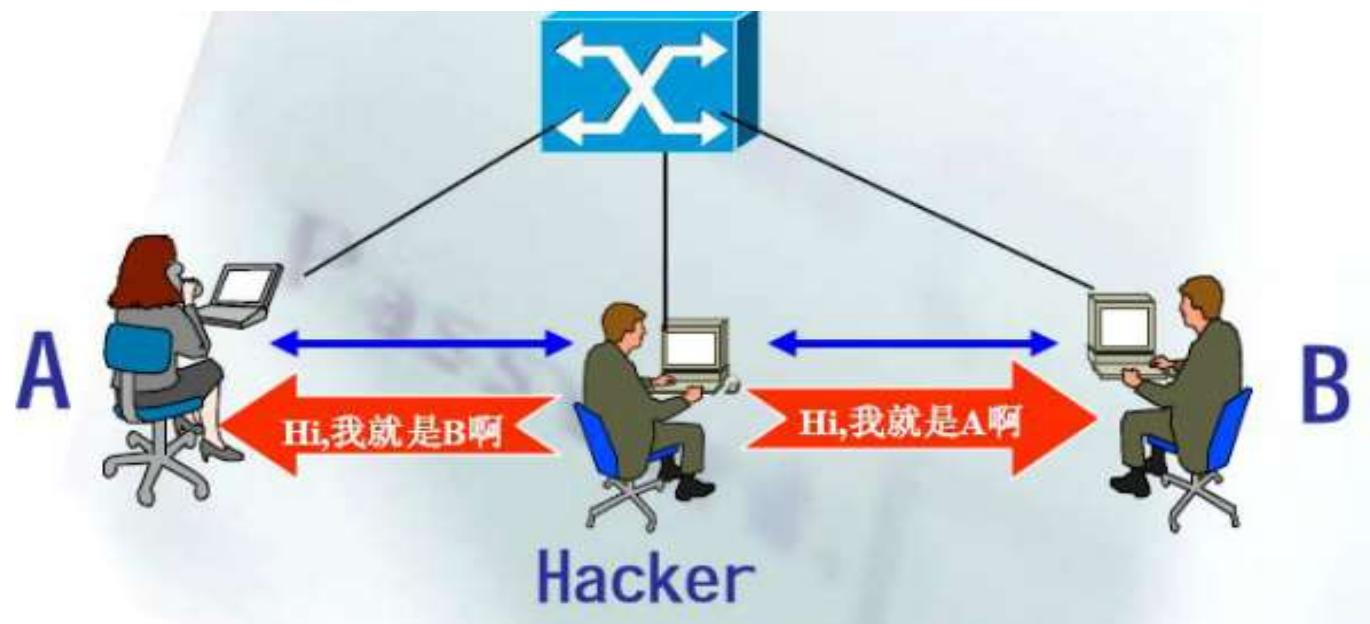


Fig. 5-51. Three interconnected class C networks: two Ethernet and an FDDI ring.



ARP 攻击

- **ARP**攻击：攻击者持续不断的发出伪造的**ARP**响应包，更改目标主机**ARP**缓存中的**IP-MAC**表项，造成网络中断或中间人攻击。
- **ARP**攻击存在于局域网





RARP 协议

- 反向地址解析协议**RARP**（**Reverse Address Resolution Protocol**）
- 解决数据链路层地址（**MAC**地址）与网络层地址（**IP**地址）的映射问题
- 主要用于无盘工作站启动
- 缺点：由于路由器不转发广播帧，**RARP**服务器必须与无盘工作站在同一子网内
- 一种替代协议**BOOTP**（使用**UDP**）

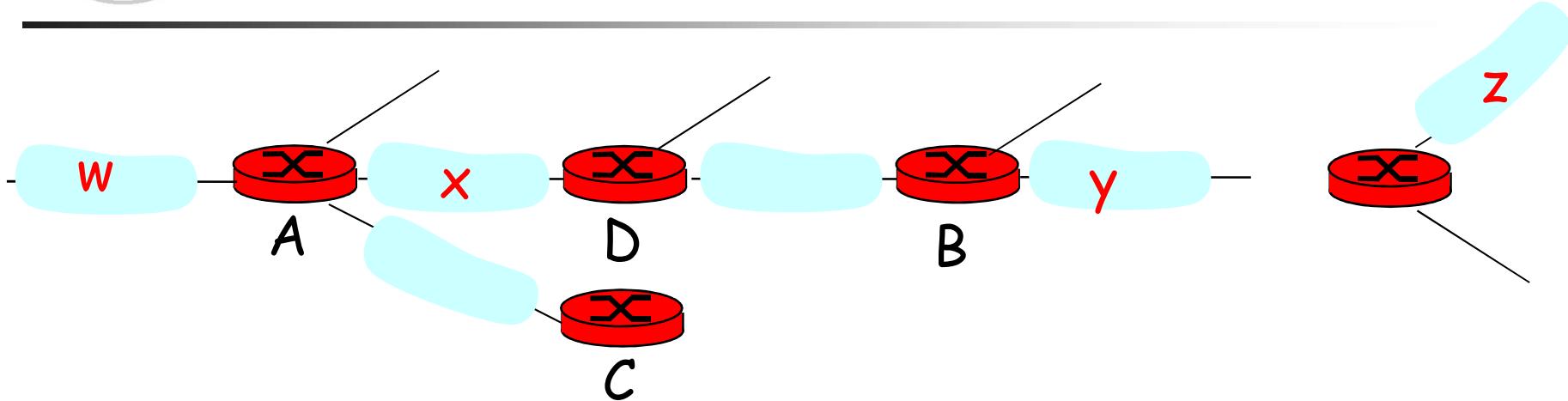


RIP协议

- **RIP (Routing Information Protocol)**
- 属于内部网关协议**IGP**
- **1982年，BSD-UNIX实现了RIP协议**
- **RIPv1, RFC1058; RIPv2, RFC1723**
- 采用距离向量算法
- 距离的衡量采用跳数 (**max = 15 hops**)
- 距离向量：每**30**秒交换一次



RIP协议（续）



Destination Network	Next Router	Num. of hops to dest.
W	A	2
Y	B	2
Z	B	7
X	--	1
....

Routing table in D



RIP协议（续）

- 故障处理
 - 如果180秒（6个路由声明周期）内没有收到来自邻居的路由声明，则认为邻居/链路失效
 - 经过该邻居的路由无效
 - 新的路由声明发往其他邻居
 - 邻居依次发出新的路由声明
 - 链路失效的信息迅速传播到全网
 - 使用poison reverse避免ping-pong loops
(infinite distance = 16 hops)



OSPF协议

- 开放最短路径优先**OSPF** (**Open Shortest Path First**)
- 属于内部网关协议**IGP**
- 支持多种距离衡量尺度，例如，物理距离、延迟、带宽等
- 采用链路状态算法
- 支持基于服务类型的路由
- 支持负载平衡
- 支持分层路由
- 适量的安全措施
- 支持隧道技术



OSPF协议（续）

- 分层路由
 - 自治系统可以划分成区域 (areas)
 - 每个AS有一个主干 (backbone) 区域，称为区域0
 - 所有其它区域都与主干区域相连
 - 其它区域之间不能相连
 - 一般情况下，有三种路由
 - 区域内
 - 区域间
 - 从源路由器到主干区域
 - 穿越主干区域到达目的区域
 - 到达目的路由器
 - 自治系统间



OSPF协议（续）

- 四类路由器，允许重叠
 - 完全在一个区域内的内部路由器
 - 连接多个区域的区域边界路由器**ABR**
 - 主干路由器
 - 自治系统边界路由器**ASBR**
- 信息类型

Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

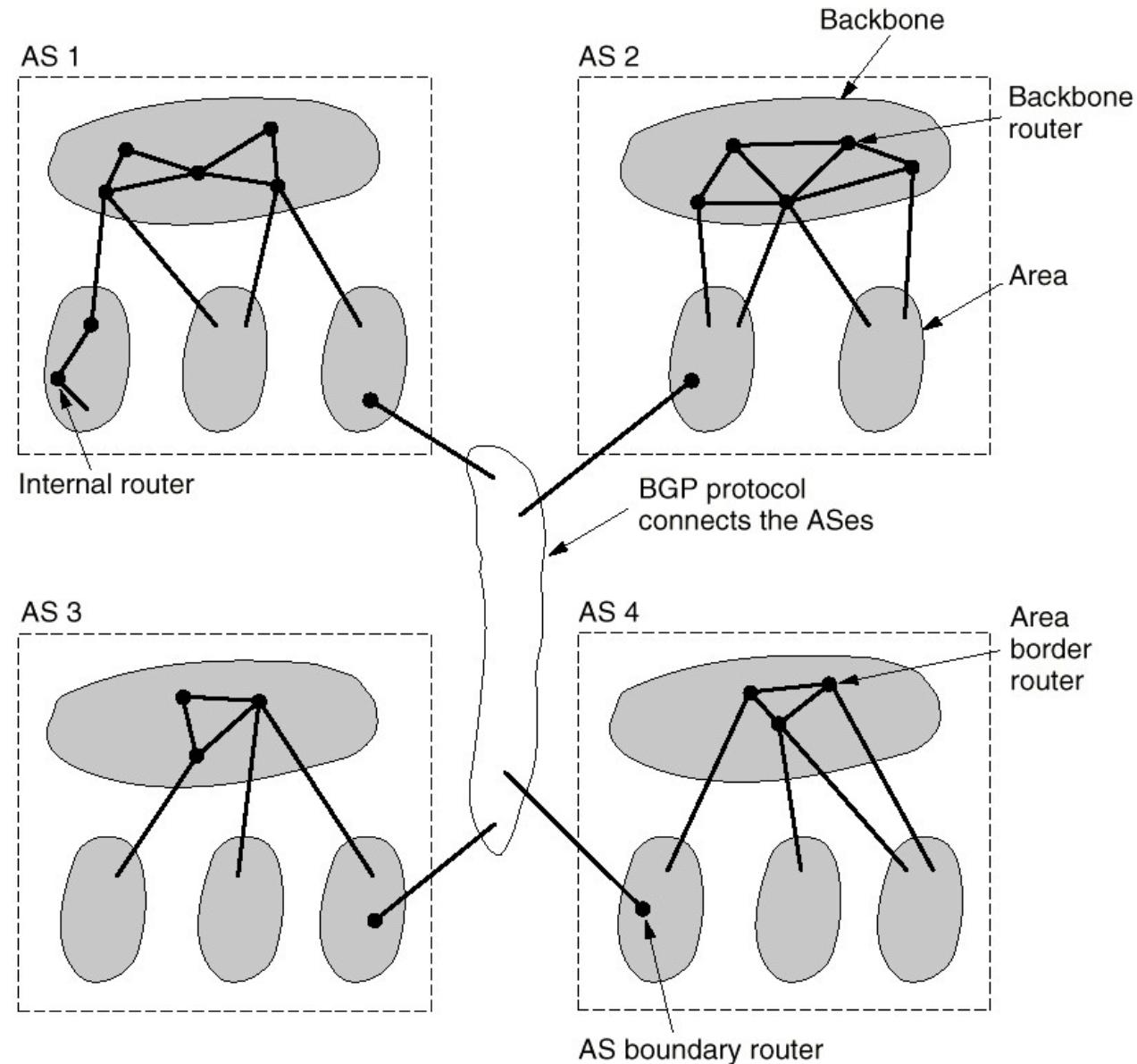


Fig. 5-53. The relation between ASes, backbones, and areas in OSPF.



BGP协议

- 为什么域间和域内的路由有所不同？
 - 策略
 - 域间路由跨越不同管理域，要控制流量如何路由
 - 域内路由属于同一管理域，不需要定义策略
 - 规模
 - 分层路由降低了路由表的大小，减小了路由更新的流量
 - 性能
 - 域内路由：着重于性能
 - 域间路由：策略更为重要



BGP协议（续）

- 边界网关协议**BGP** (**Border Gateway Protocol**)
- 通过**TCP**连接传送路由信息
- 采用路径向量 (**path vector**) 算法， 路由信息中记录路径的轨迹
 - 与距离向量协议相似
 - 每个BGP网关向邻居广播所有通往目的地的路径
 - 比如， **Gateway X** 发送它通往Z的路径
 - $\text{Path (X,Z)} = \text{X,Y}_1,\text{Y}_2,\text{Y}_3,\dots,\text{Z}$



BGP协议（续）

- 假设：网关**X**把它的路径发给**peer**网关**W**
- **W**可以选择是否使用**X**提供的路径，依据是开销、策略（例如：不使用通过竞争对手网络的路径）、防止出现路由循环等
- 如果**W**使用**X**提供的路径，则
 - **Path (W,Z) = w, Path (X,Z)**
- 注意：**X**能通过其发往**peers**的路径声明来控制进入的流量。不想转发到**Z**的流量，可以通过不通告通往**Z**的路径来实现



BGP协议（续）

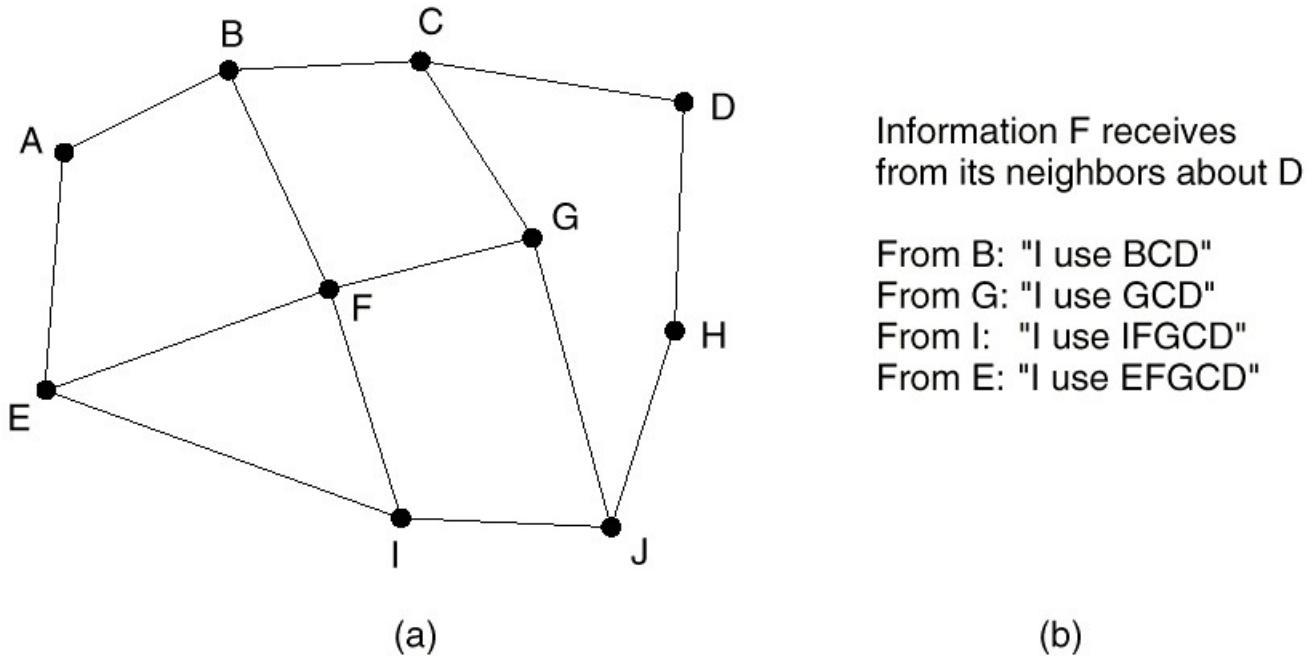


Fig. 5-55. (a) A set of BGP routers. (b) Information sent to F.



BGP协议（续）

■ BGP 消息

- **OPEN**: 建立与对等方的TCP连接并认证身份
- **UPDATE**: 声明新的路径或撤销旧的路径
- **KEEPALIVE**: 在没有UPDATE消息的时候保持连接有效。也用来回应OPEN请求
- **NOTIFICATION**: 报告上一个消息的错误，也用来关闭连接



无类域间路由CIDR

- CIDR (Classless InterDomain Routing) 的提出
 - Internet指数增长, IPv4地址已经分配完毕
 - IP is rapidly becoming a victim of its own popularity
 - 基于分类的IP地址空间的组织浪费了大量的地址
 - 罪魁祸首是B类地址
- CIDR
 - RFC 1519
 - 基本思想: 将剩余的C类地址分成大小可变的地址空间
 - 已推广到所有单播地址, 用掩码确定前缀长度



无类域间路由CIDR（续）

- 例如，需要**2000**个地址，则分配一个有**2048**个地址（**8**个**C**类地址）的连续地址块，而不是一个**B**类地址
- **RFC 1519** 改变了过去**C**类地址的分配规则，将世界分成**4**个区，每个区分配一块连续的**C**类地址空间
 - 欧洲：194.0.0.0 ~ 195.255.255.255
 - 北美：198.0.0.0 ~ 199.255.255.255
 - 中、南美：200.0.0.0 ~ 201.255.255.255
 - 亚太：202.0.0.0 ~ 203.255.255.255

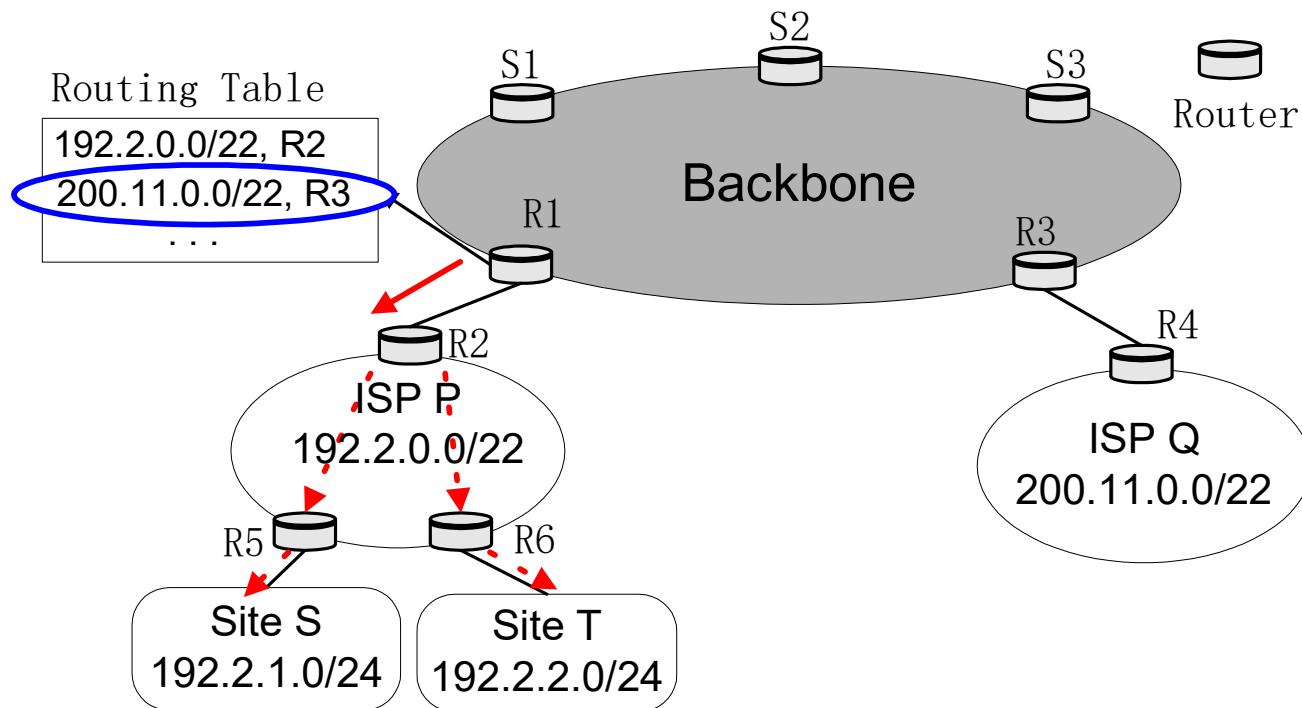


无类域间路由CIDR（续）

- 路由表中增加一个**32位**的掩码（**mask**）域
- 最长前缀匹配原则：路由查找时，若多个路由表项匹配成功，选择掩码最长（**1比特数多**）的路由表项
- **CIDR**思想可用于所有**IP**地址，没有**A、B、C**类之分。
- 地址格式：**a.b.c.d/x**，其中**x** 地址中网络号的位数

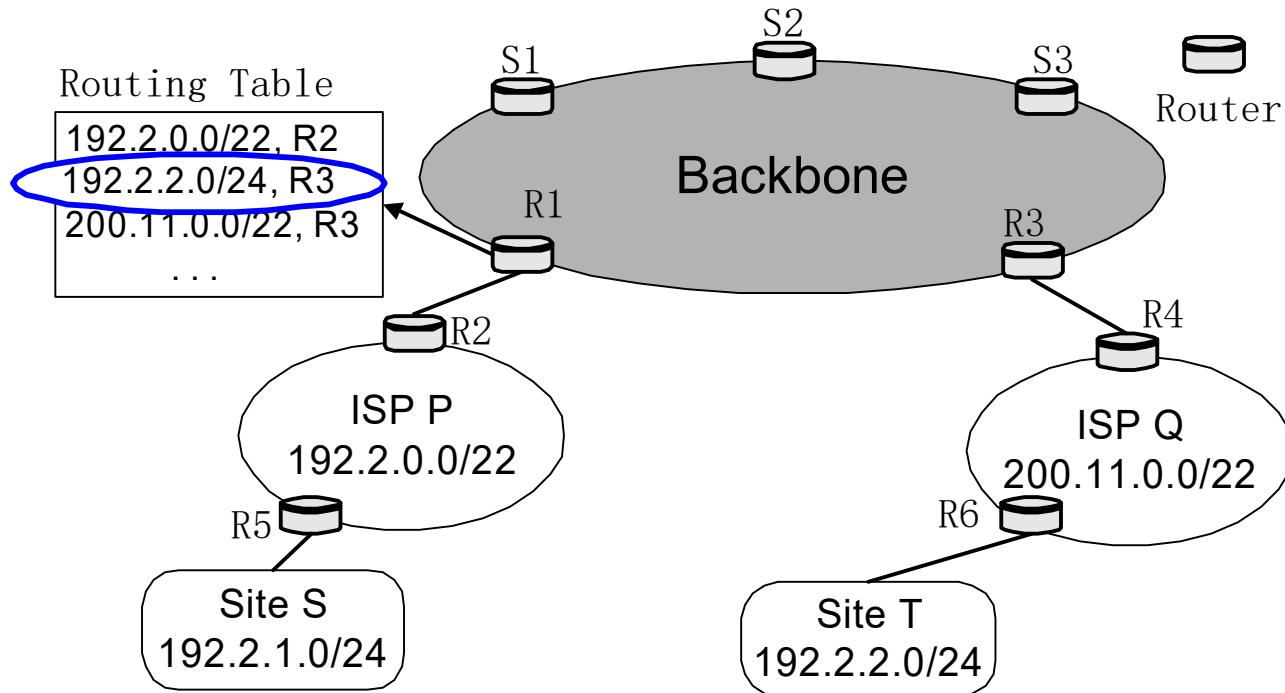


最长前缀匹配



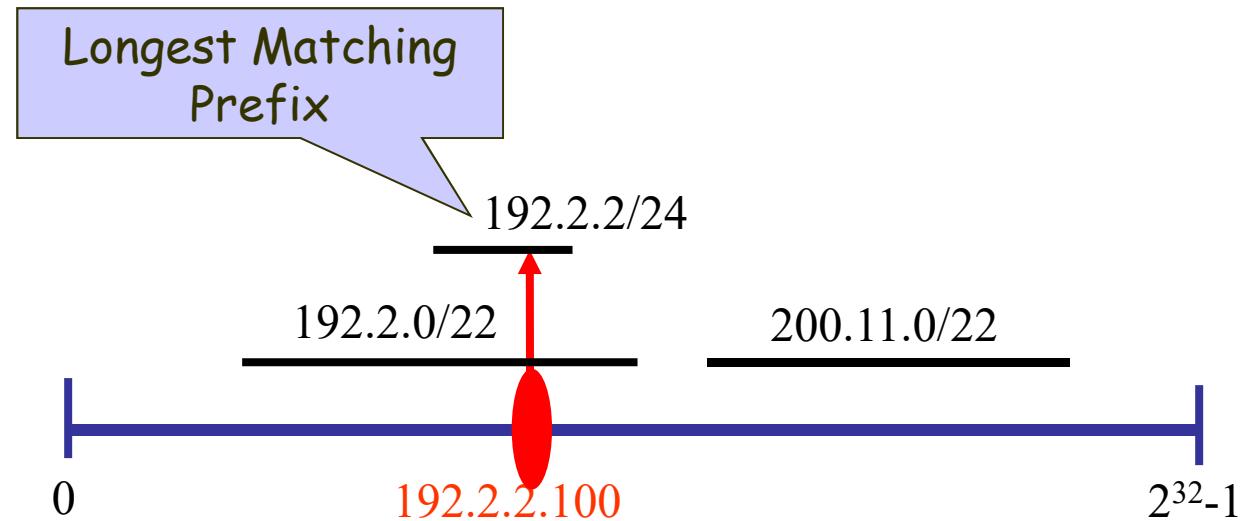


最长前缀匹配（续）





最长前缀匹配（续）



路由查找：在所有前缀中寻找最长匹配



无类域间路由CIDR（续）

■ 例

- Cambridge University 需要2048个地址,
194.24.0.0 ~ 194.24.7.255, 掩码**255.255.248.0**
- Oxford University 需要4096个地址, **194.24.16.0 ~ 194.24.31.255**, 掩码**255.255.240.0**;
- Edinburgh University 需要1024个地址,
194.24.8.0 ~ 194.24.11.255, 掩码**255.255.252.0**
- 路由表内容

地址	掩码
11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000
11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000

- 或者表示成

- **194.24.0.0/21, 194.24.16.0/20, 194.24.8.0/22**



无类域间路由CIDR（续）

- 一个目的地址为194.24.17.4的分组到达路由器，路由表查找过程如下：
 - 194.24.17.4的二进制表示
11000010 00011000 00010001 00000100
 - 该地址与第一项Cambridge的掩码做 AND 操作，得到
11000010 00011000 00010000 00000000
与Cambridge的地址不同，匹配不成功
 - 该地址与下一项Oxford的掩码做AND操作，得到
11000010 00011000 00010000 00000000
与Oxford的地址相同，匹配成功
 - 继续匹配，最后选择前缀最长的路由表项



IPv6

■ IPv6的提出

- CIDR仅仅是一种临时补救措施，不能从根本上解决IP地址空间匮乏的问题
- 移动电话、家电上网需要大量的IP地址
- 1990年，IETF开始IPv6的工作，收到21个建议
- 1992年，7个建议被讨论
- 1993年，3个比较好的建议发表在IEEE Network上，进一步讨论、修改、结合后，形成IPv6
- IPv6与IPv4不兼容，但与其它Internet协议兼容，如TCP、UDP、OSPF、BGP、DNS等，但实际上还是需要开发另外一套协议栈



IPv6 (续)

■ IPv6的目标

- 即使在不能有效分配地址空间的情况下，也能支持数十亿的主机
- 减少路由表的大小
- 简化协议，使得路由器能够更快的处理分组
- 提供比IPv4更好的安全性
- 更多的关注服务类型，特别是实时数据
- 支持Multicast
- 支持移动功能
- 协议具有很好的可扩展性
- 增强安全性
- 在一段时间内，允许IPv4与IPv6共存



IPv6 vs IPv4

- 与**IPv4**相比，**IPv6**的主要变化
 - 地址变长，由32位变成128位
 - IP头简化，由13个域减少为7个域，提高路由器处理速度
 - 由于**IPv6**分组头定长，取消IHL域
 - Protocol域取消，用Next header域表示
 - 取消与分片有关的域，**IPv6**的分片方法：所有主机和路由器必须支持1280字节的分组，当主机发送一个大分组时，路由器不做分片，而是给主机发一个错误信息，由主机做分片
 - 取消Checksum域
 - 更好的支持选项功能
 - 安全性提高



IPv6 分组头

■ IPv6分组头

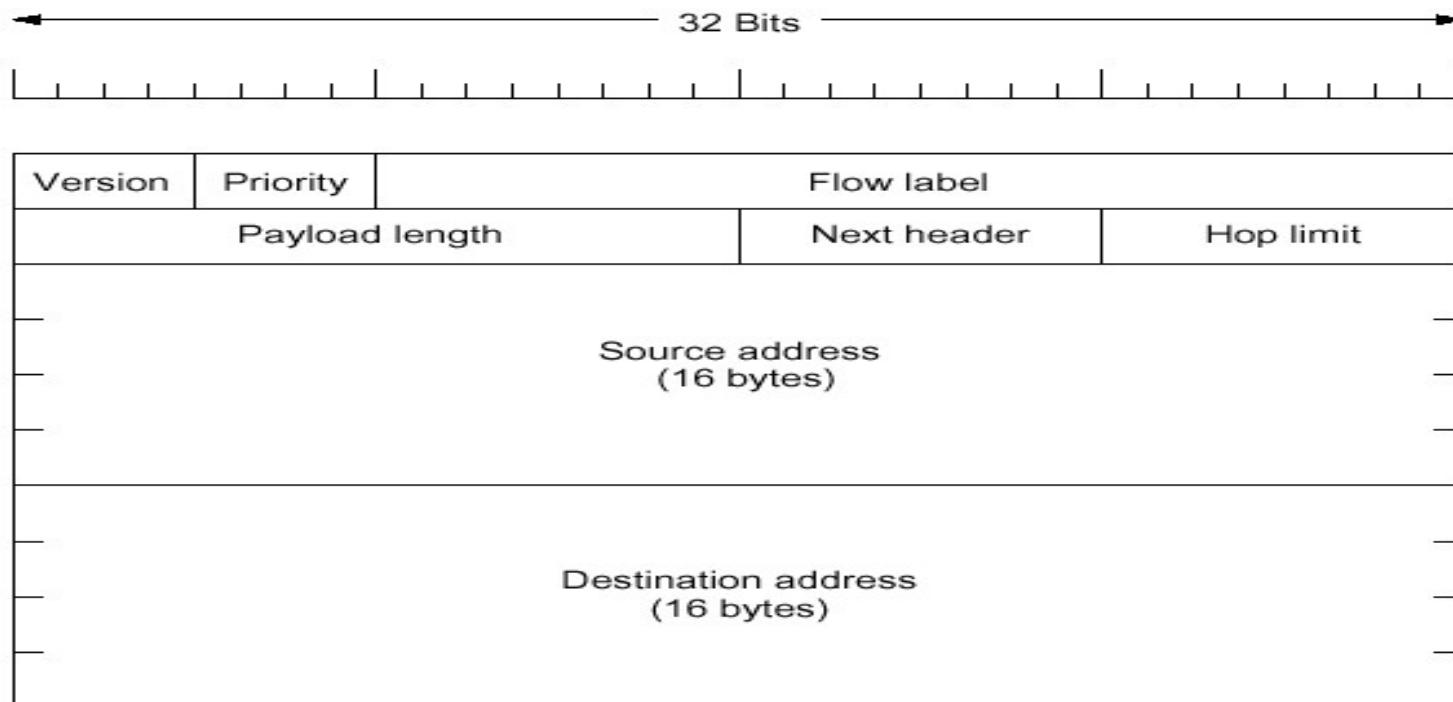


Fig. 5-56. The IPv6 fixed header (required).



IPv6分组头（续）

- **Version**, 值为**6**
- **Priority**, 用来区分源端可以流控或不能流控的分组, 值越小优先级越低
- **Flow label**, 用来允许源和目的建立一条具有特殊属性和需求的伪连接
- **Payload length**, 用来指示**IP**分组中**40**字节分组头后一部分的长度, 与**IPv4**的**total length**域不同
- **Next header**, 指示扩展分组头, 若是最后一个分组头, 则指示传输协议类型 (**TCP/UDP**)
- **Hop limit**, **IP**分组的最大跳数
- **Source address, destination address**, **16**字节定长地址



IPv6 地址表示

- **16字节地址表示成用冒号(:) 隔开的8组，每组4个16进制位，例如，**
**8000:0000:0000:0000:0123:4567:89AB:CD
EF**
- 由于有很多“0”，有三种优化表示
 - 打头的“0”可以省略，0123可以写成123
 - 一组或多组16个“0”可以被一对冒号替代，但是一对冒号只能出现一次。上面的地址可以表示成8000::123:
4567:89AB:CDEF
 - IPv4地址可以写成一对冒号和用“.”分隔的十进制数，例如::192.31.20.46



IPv6 扩展分组头

- 目前定义了六种类型的扩展分组头
- **hop-by-hop header**, 用来指示路径上所有路由器必须检查的信息
- **routing header**, 列出路径上必须要经过的路由器
- **fragmentation header**, 与**IPv4**相似, 扩展头中分组包括**IP**分组标识号、片段号和判断是否还有片段的位, 只有源主机可以分片

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

Fig. 5-58. IPv6 extension headers.



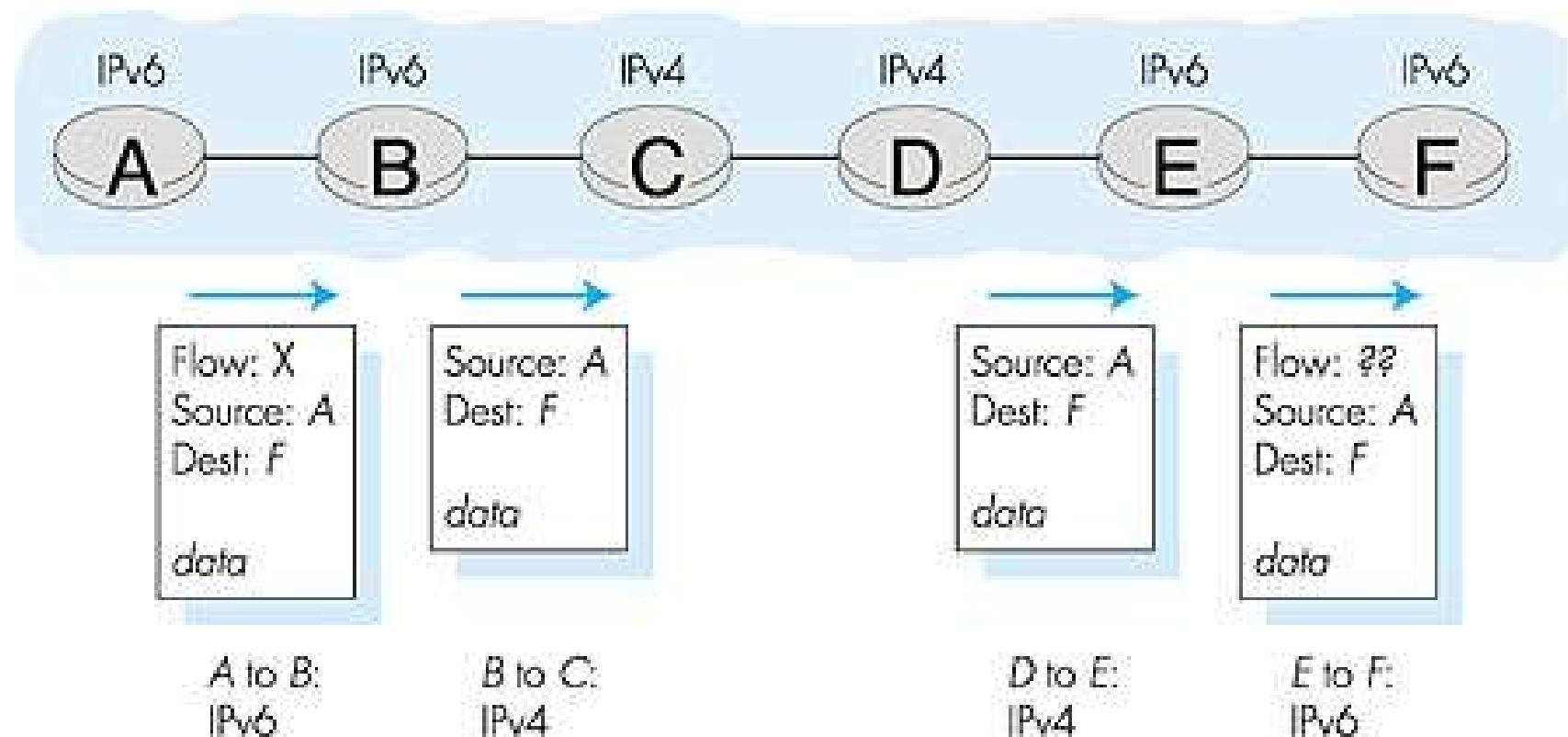
从IPv4往IPv6的过渡

- 所有路由器不可能同时升级
 - 没有一个确定的期限 (not like Y2K)
 - IPv4和IPv6路由器混合的网络如何操作?
- 有三种方案 (两种技术)
 - 双栈: 实现IPv4/v6两套协议栈, 主机根据DNS返回的结果或对方发来报文的版本号决定采用哪个协议, 路由器根据收到IP分组的版本号决定采用哪个协议
 - 翻译: 有些路由器同时运行两套协议栈, 可以在这两套之间进行协议翻译和地址翻译, 例如NAT-PT (已废弃)
 - 隧道: IPv6的报文作为IPv4报文的净负荷在IPv4网络中传输



翻译

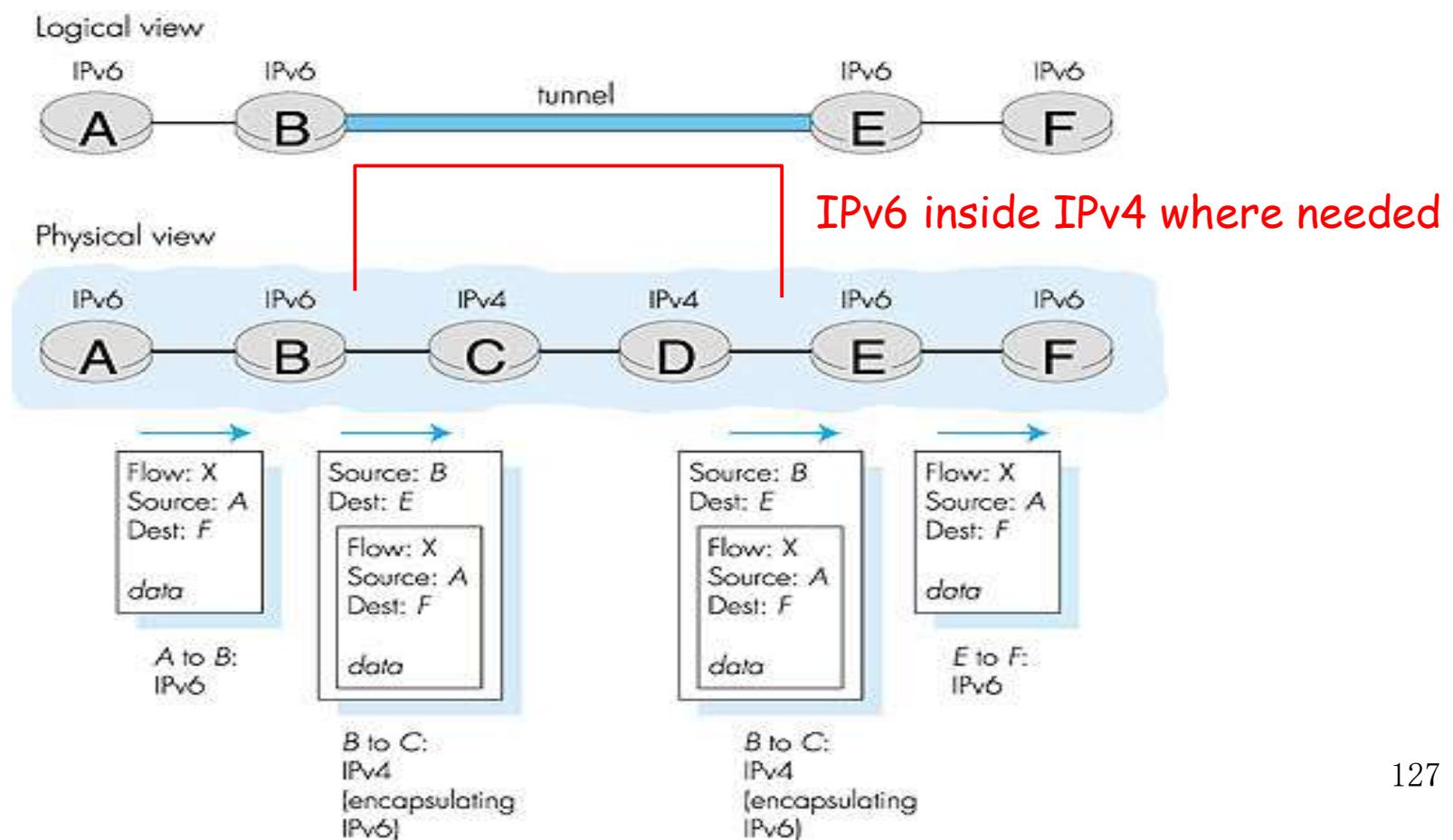
■ NAT-PT





隧道

■ 隧道



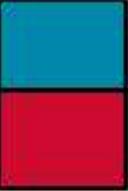


互联网协议小结

- **IPv4协议**
 - 分片与重组
 - IPv4地址
- **ICMP协议**
 - ICMP报文封装在IP分组中
- **ARP协议**
 - ARP协议工作过程
- **RARP协议**
 - RARP与BOOTP的区别
- **RIP**
 - 内部网关协议
 - 距离向量算法
 - 基于UDP
- **OSPF**
 - 内部网关协议
 - 链路状态算法
 - 分层思想
 - 四种类型路由器
- **BGP**
 - 外部网关协议
 - 路径向量算法
 - 基于TCP
- **CIDR**
 - CIDR的思想
 - IP地址分配和掩码配置
 - 最长匹配原则
- **IPv6**
 - IPv6与IPv4的主要区别
 - IPv6地址
 - IPv4/IPv6过渡

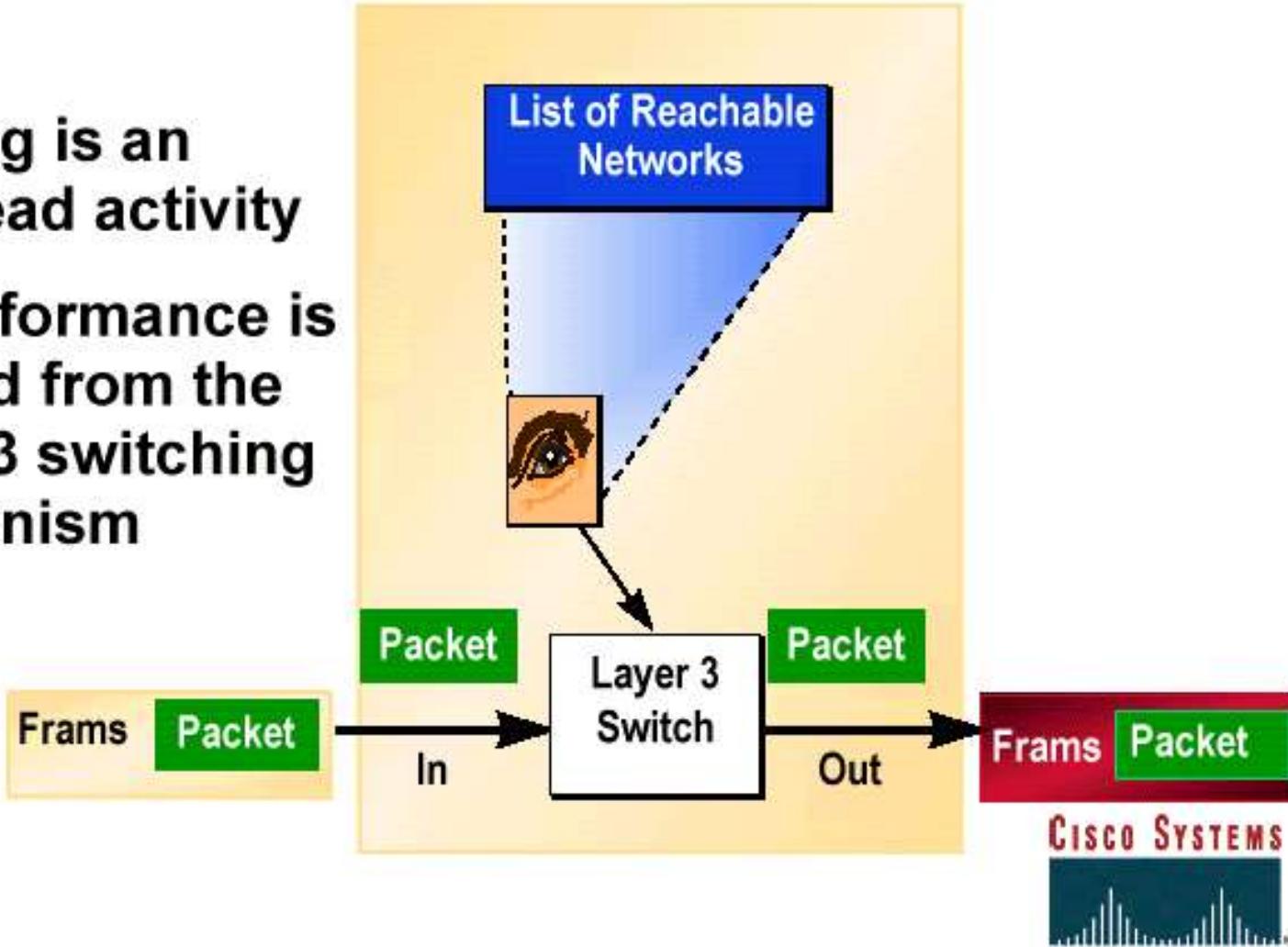
路由器体系结构和关键技术





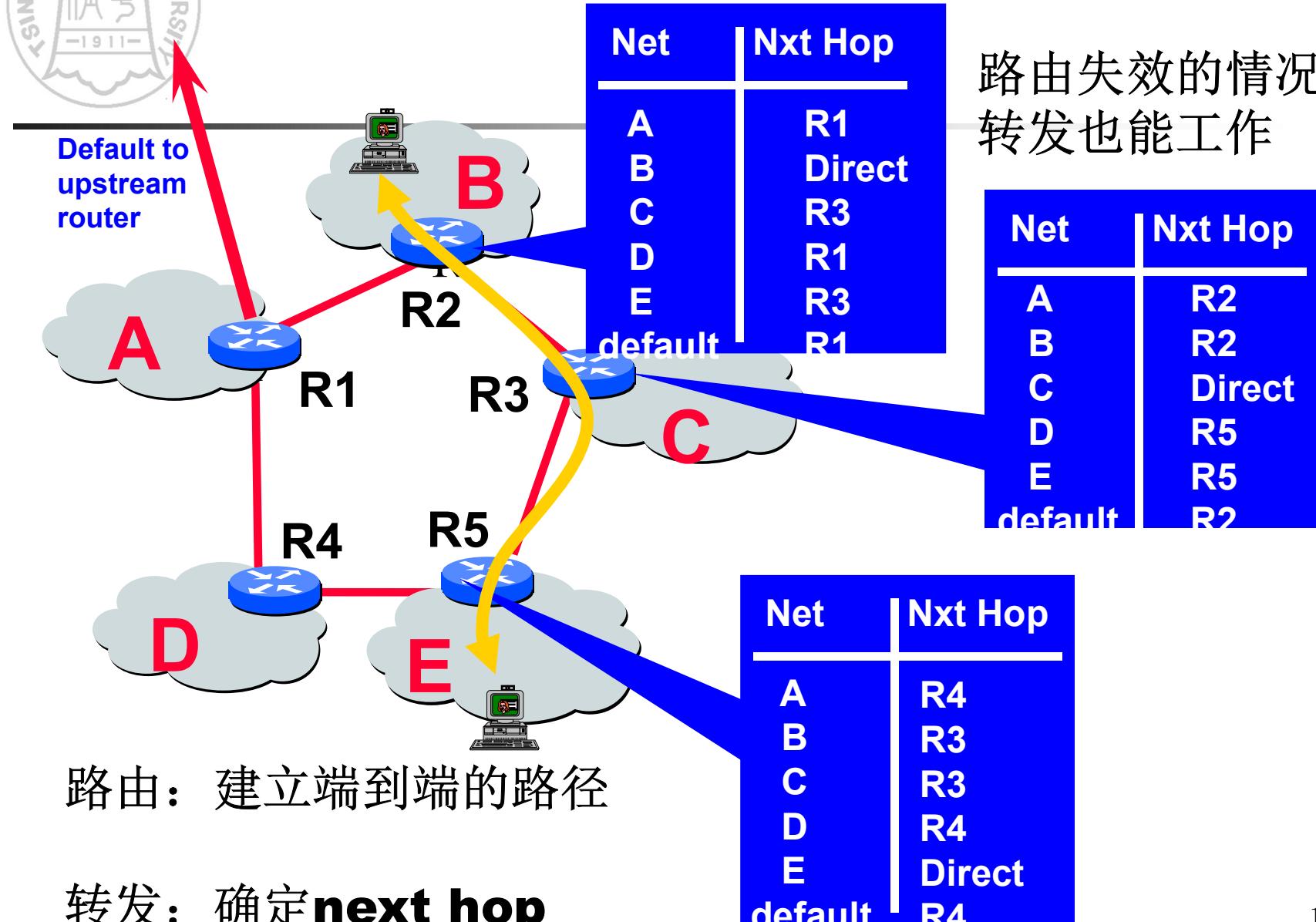
What Is a Router?

- Routing is an overhead activity
- All performance is derived from the Layer 3 switching mechanism





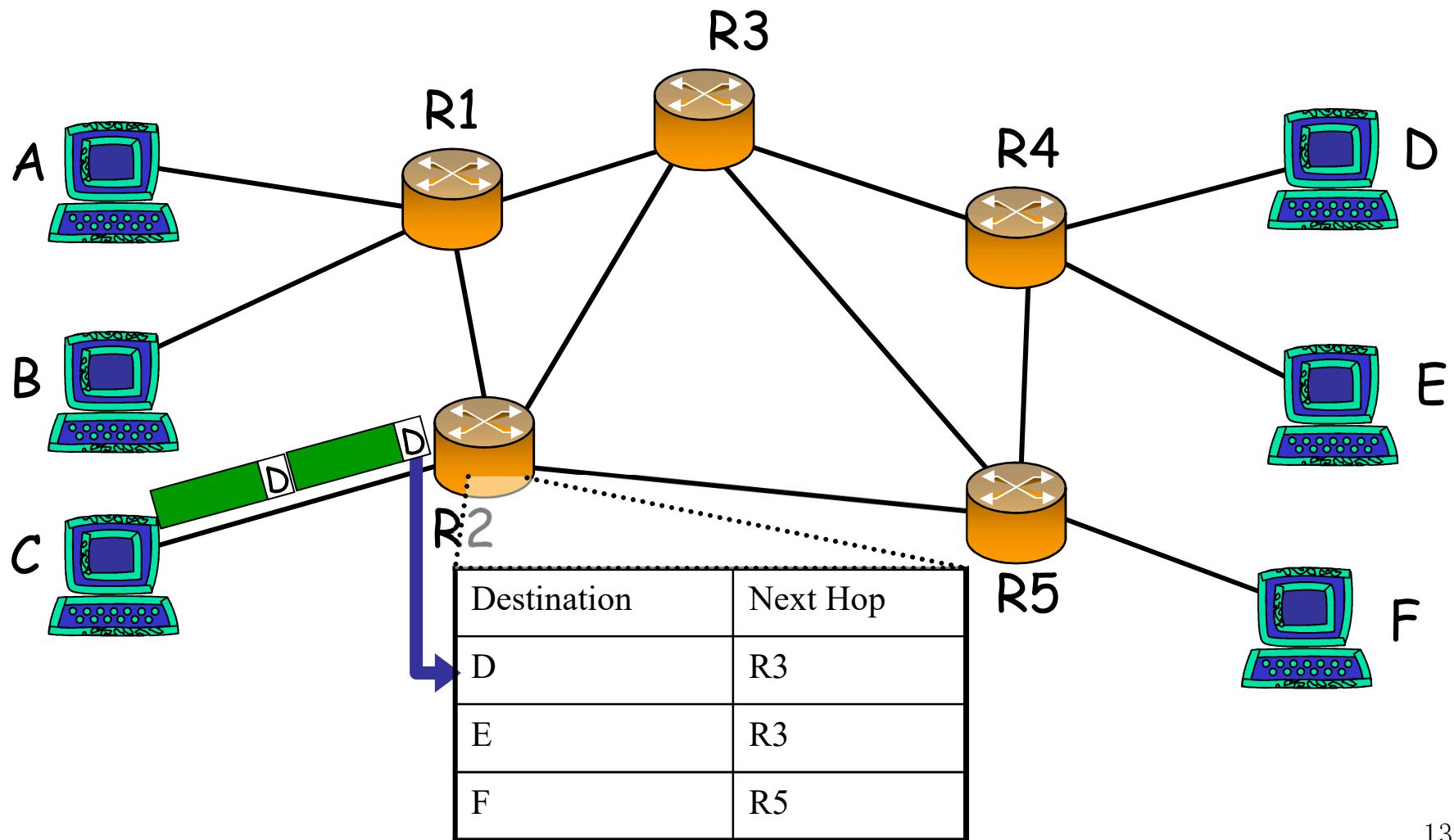
什么是路由？什么是转发？



路由失效的情况下，
转发也能工作

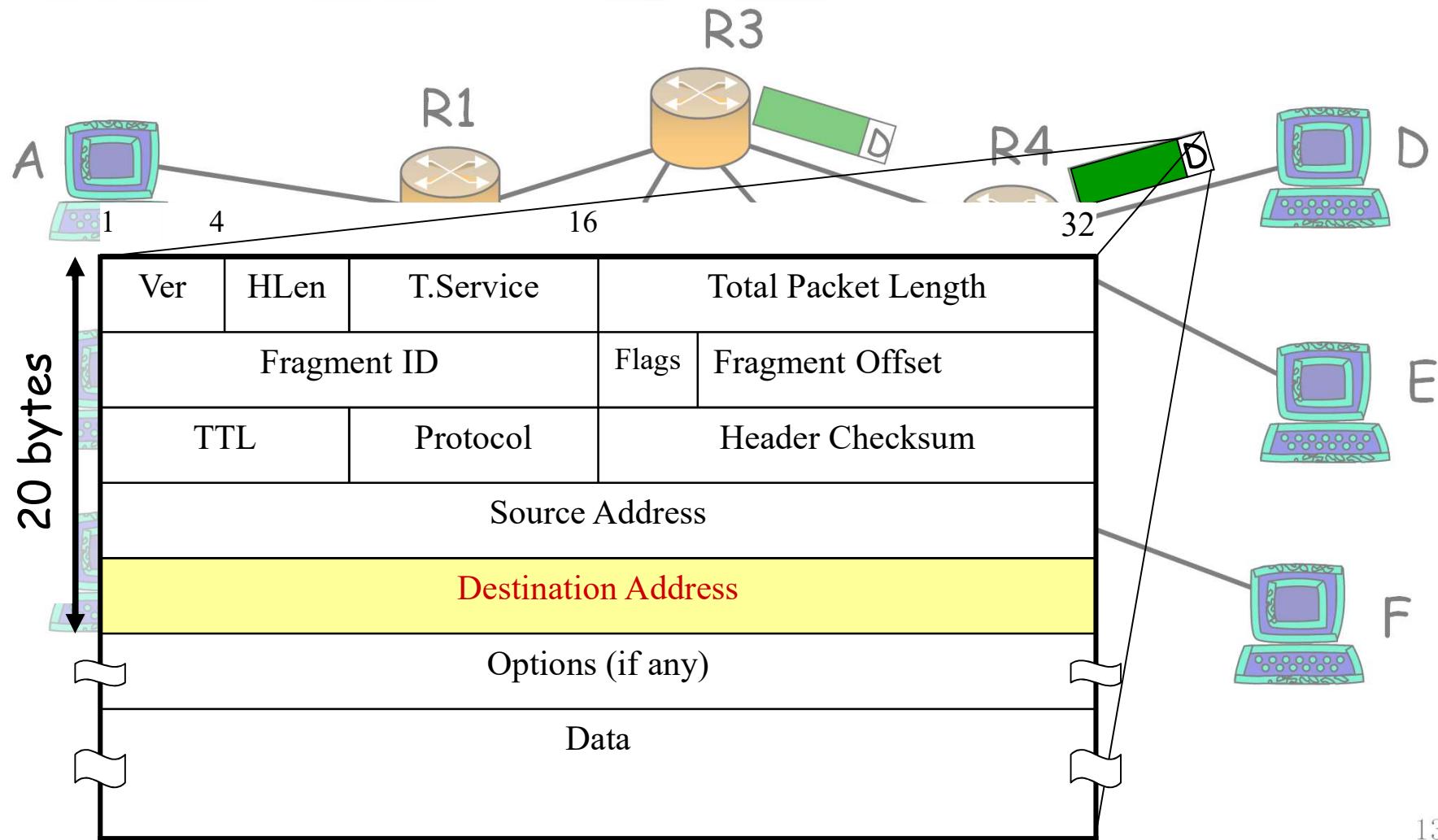


什么是路由？什么是转发？（续）



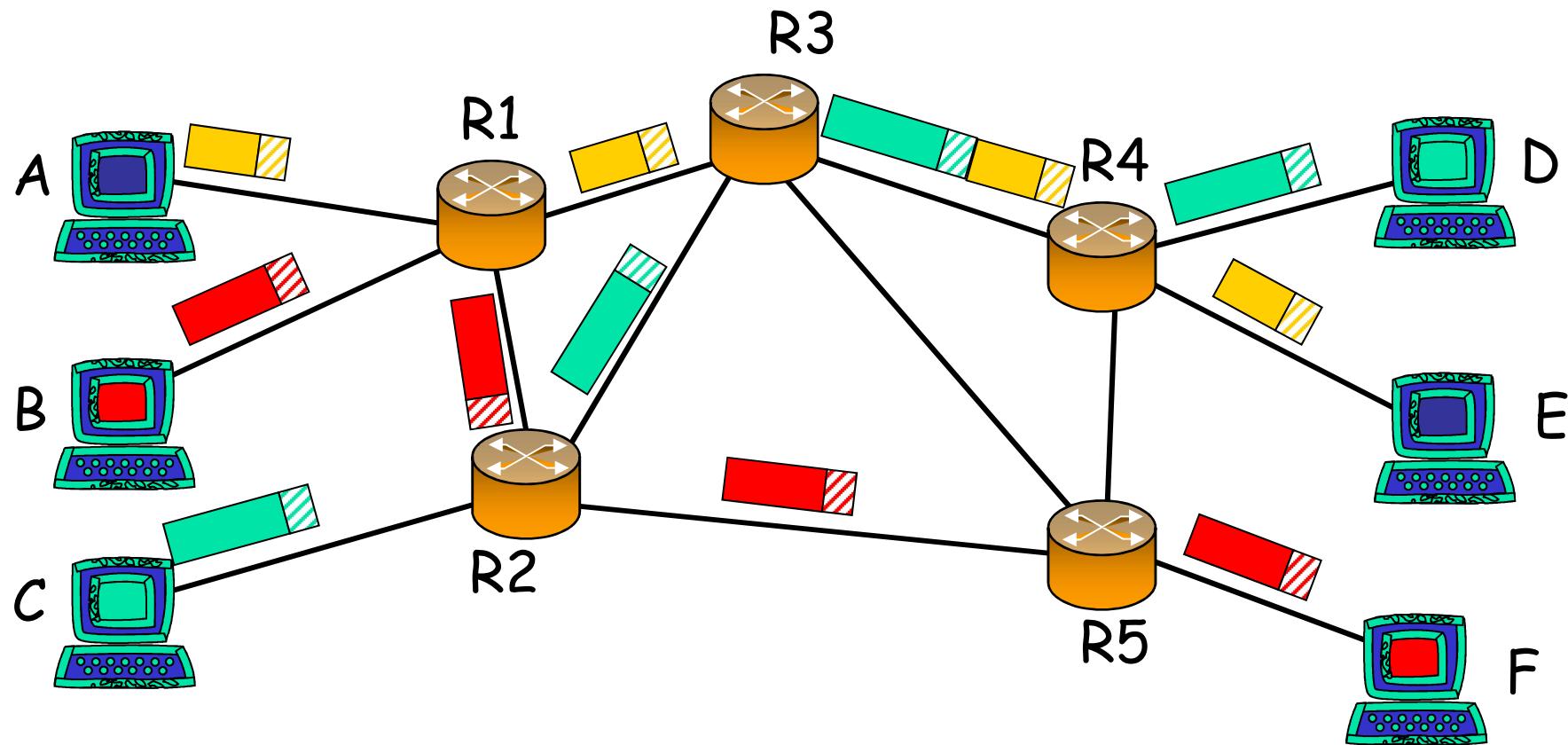


什么是路由？什么是转发？（续）





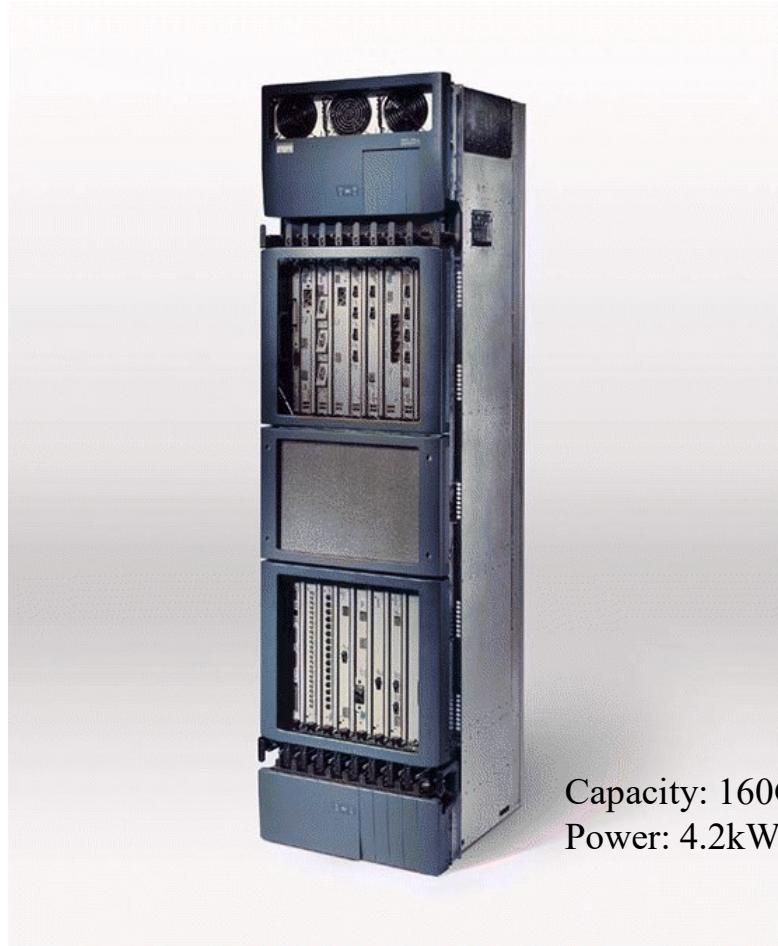
什么是路由？什么是转发？（续）





路由器是什么样子

Cisco GSR 12416

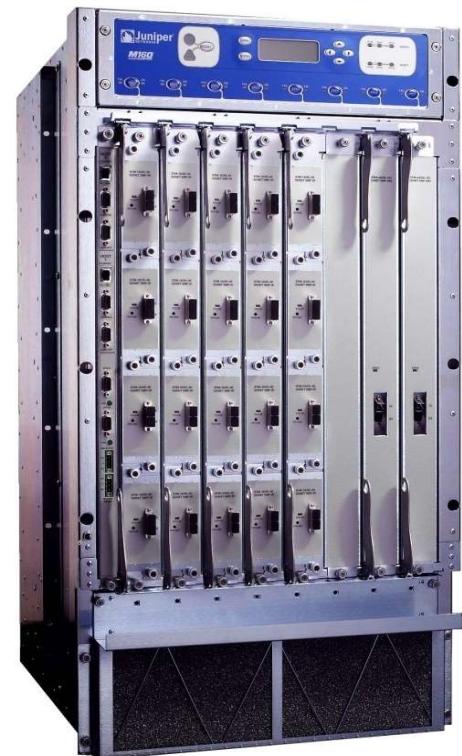


Capacity: 160Gb/s
Power: 4.2kW

Cisco CRS1



Juniper M160



Capacity: 80Gb/s
Power: 2.6kW



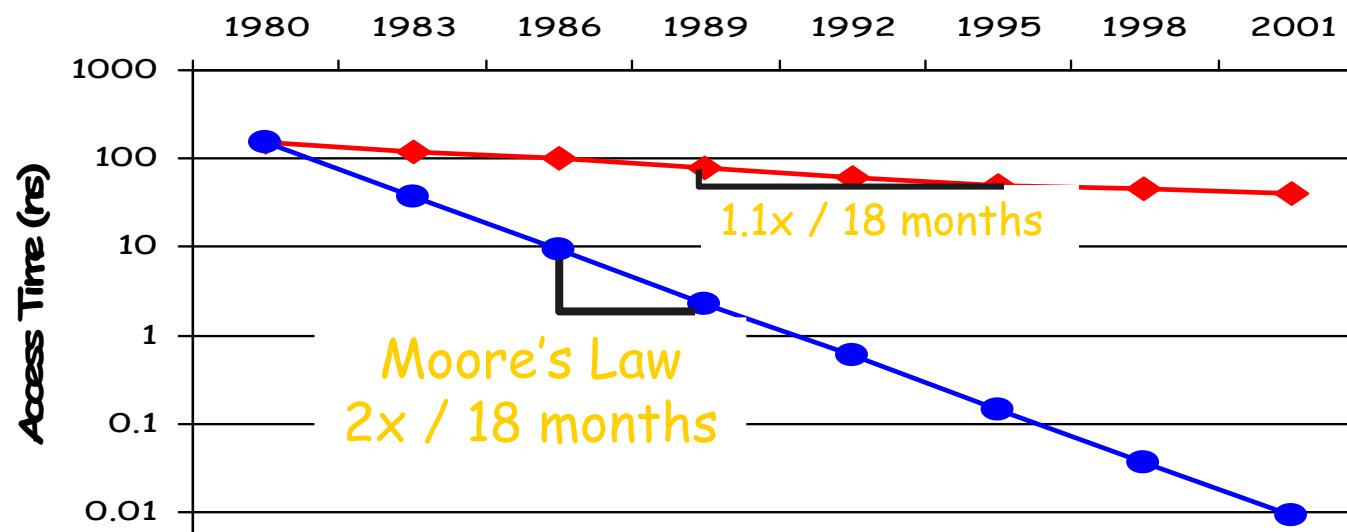
为什么设计制造高速路由器有 很大的难度？

-
1. 内存速度是瓶颈，没法跟上摩尔定律
 2. 对路由器性能要求的提升速度超过了摩尔定律，带宽增加速度超过了处理能力增加的速度



为什么设计制造高速路由器有 很大的难度？（续）

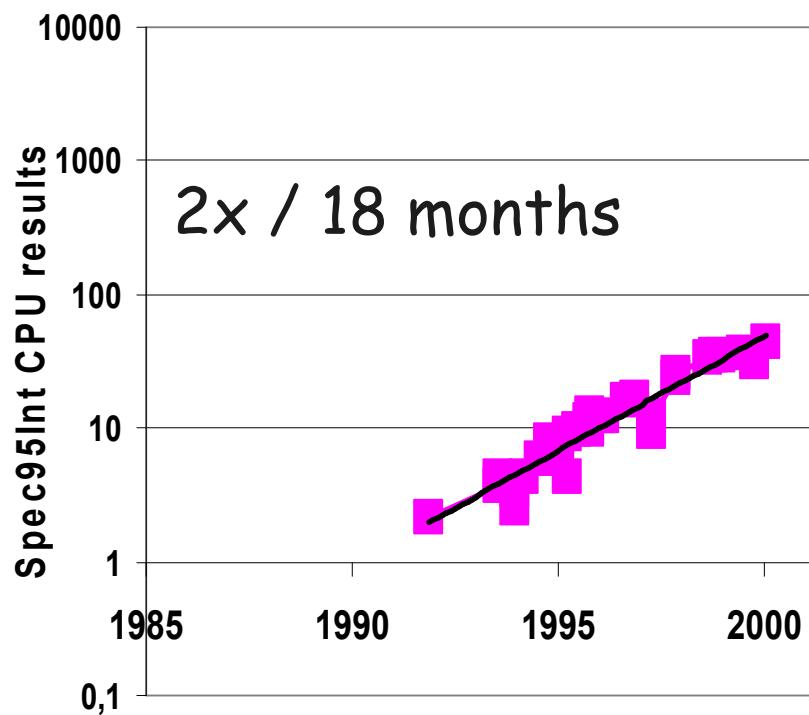
1. **It's hard to keep up with Moore's Law:**
 - The bottleneck is memory speed.
 - Memory speed is not keeping up with Moore's Law



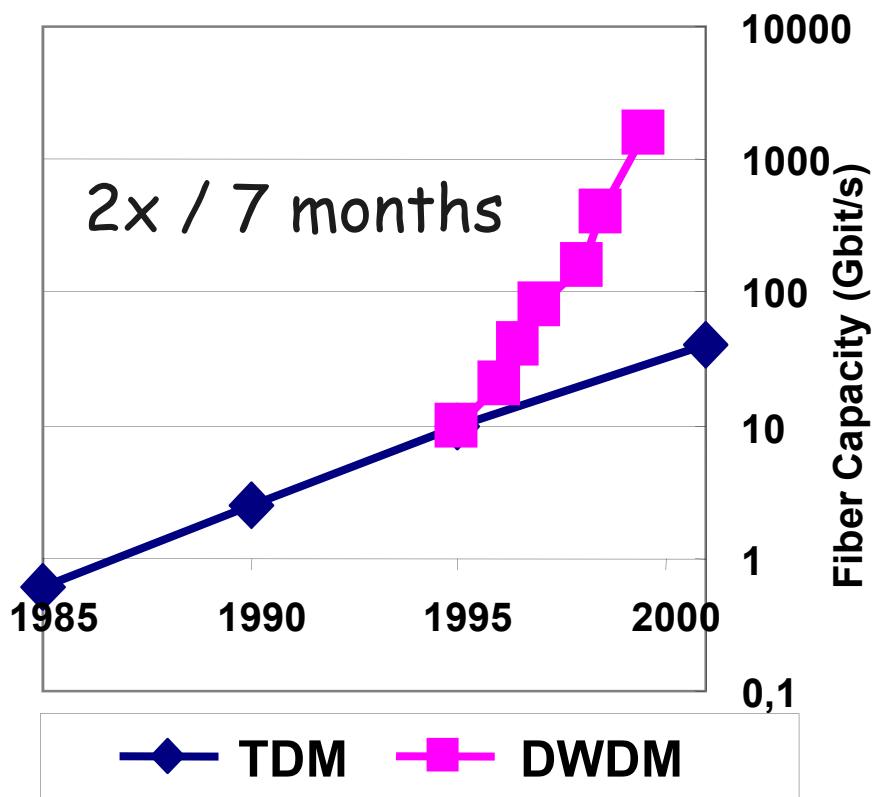


摩尔定律和光纤定律

分组处理能力



链路速度



Source: SPEC95Int & David Miller, Stanford.



路由器的性能增长超过了摩尔定律

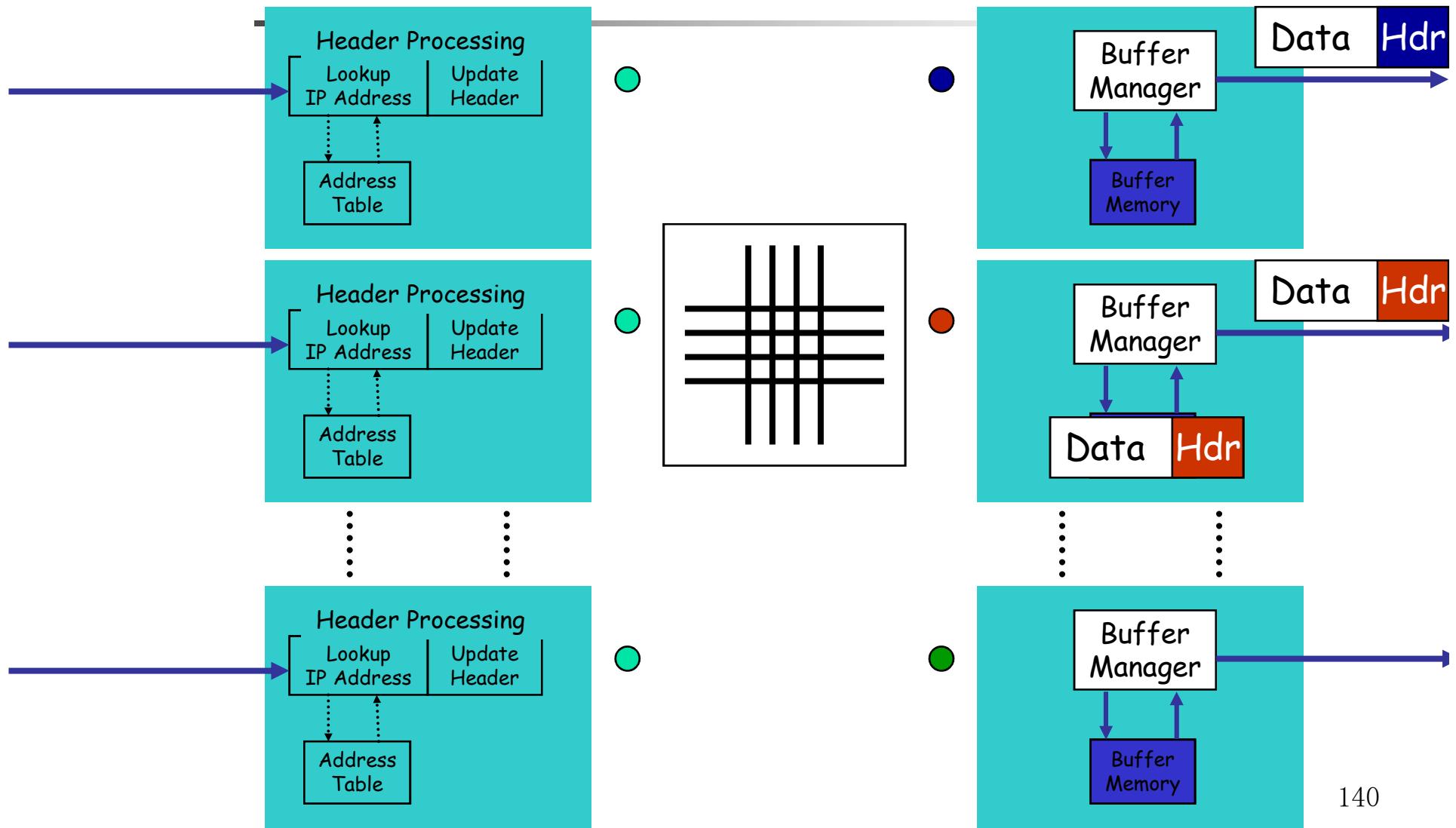
商业路由器的容量增长

- Capacity 1992 ~ 2Gb/s
- Capacity 1995 ~ 10Gb/s
- Capacity 1998 ~ 40Gb/s
- Capacity 2001 ~ 160Gb/s
- Capacity 2002 ~ 640Gb/s (T640)
- Capacity 2004 ~ 92Tb/s (CRS-1)
- Capacity 2010 ~ 322Tb/s (CRS-3)

平均增长率: **2.2x / 18 months.**



通用路由器框架



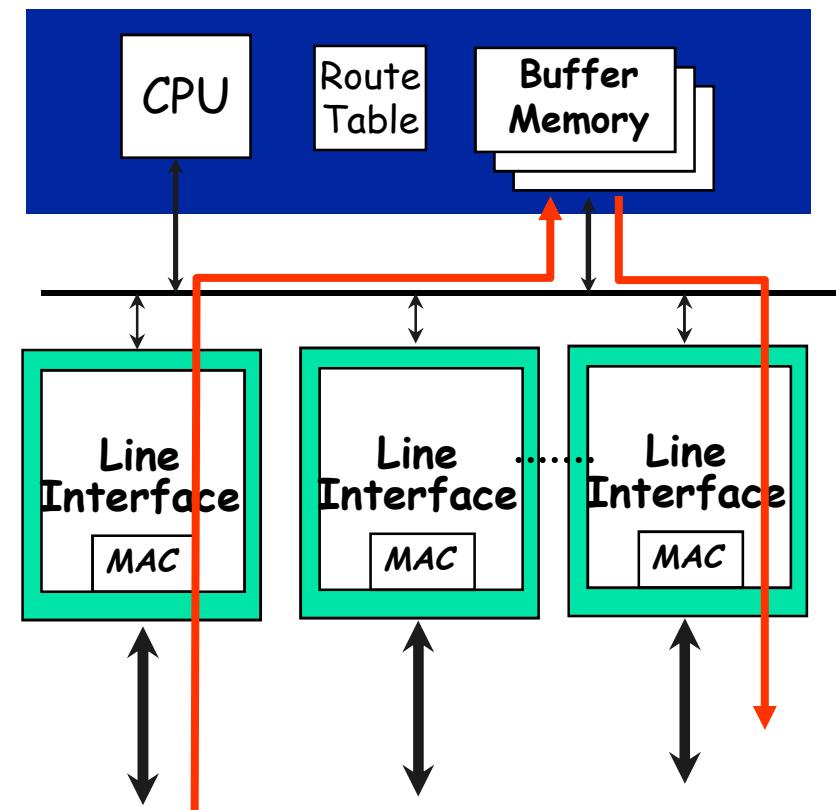
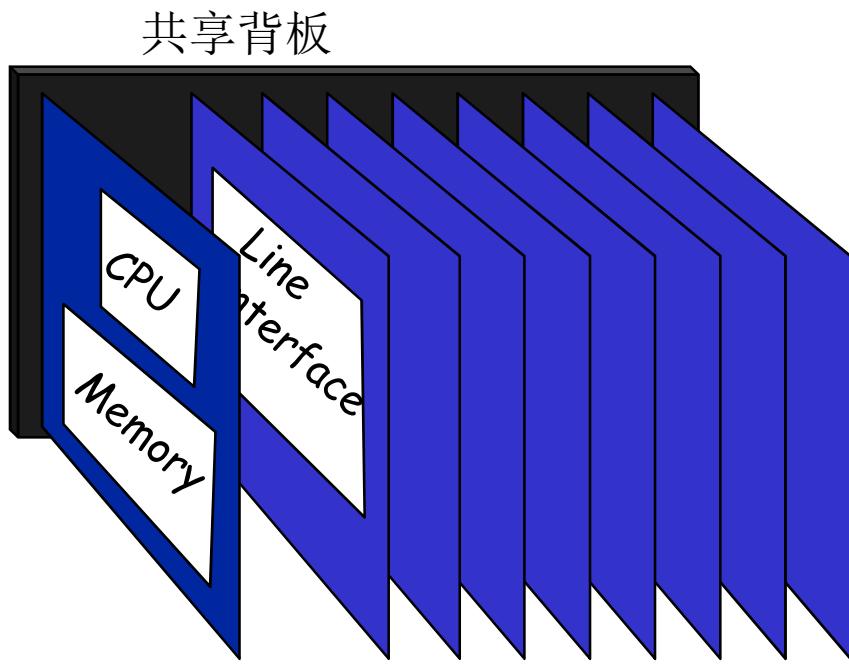


路由器体系结构的发展

- 单总线，单处理器
 - 传统的计算机结构，处理器成为处理瓶颈
- 单总线，多处理器，每个接口卡上有一个处理器，主处理器负责协调。
 - 分组转发由本地处理器完成，减少总线负担
- 交换结构 + 专用硬件（ASIC, FPGA, NP）
 - 交换结构实现无阻塞转发
 - 硬件实现对IP分组的处理和路由查找
- 可扩展路由器（Internet Routing Matrix）
 - 多结点级连，类似并行计算机



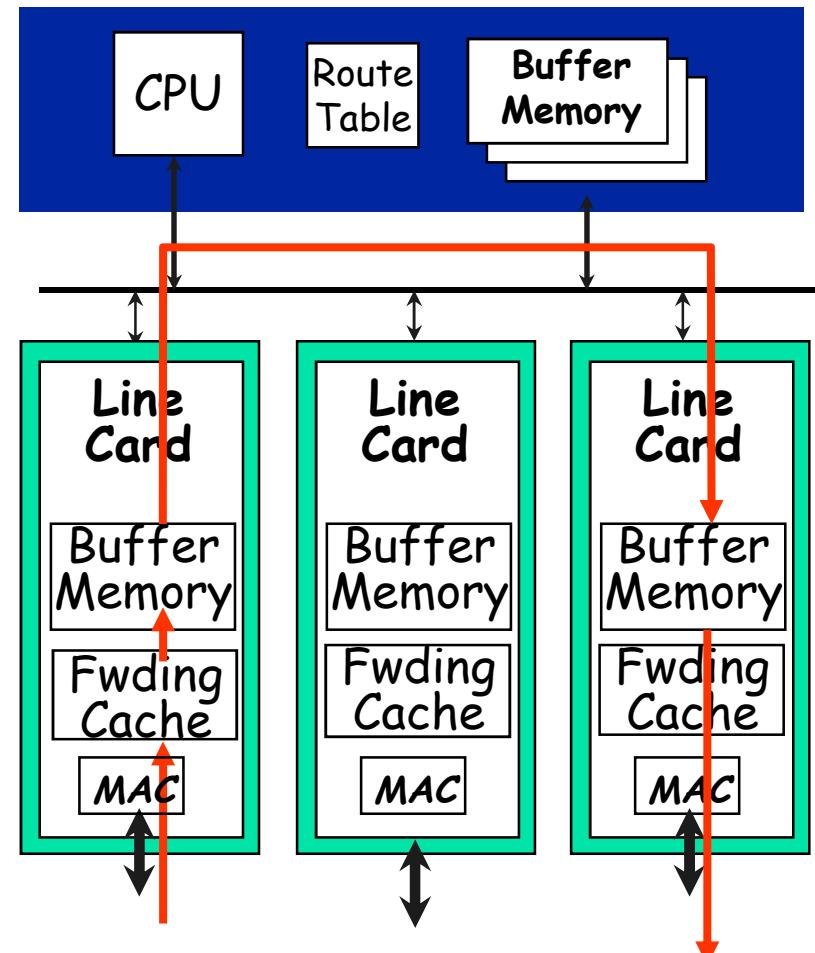
第一代路由器



Typically <0.5Gb/s aggregate capacity



第二代路由器

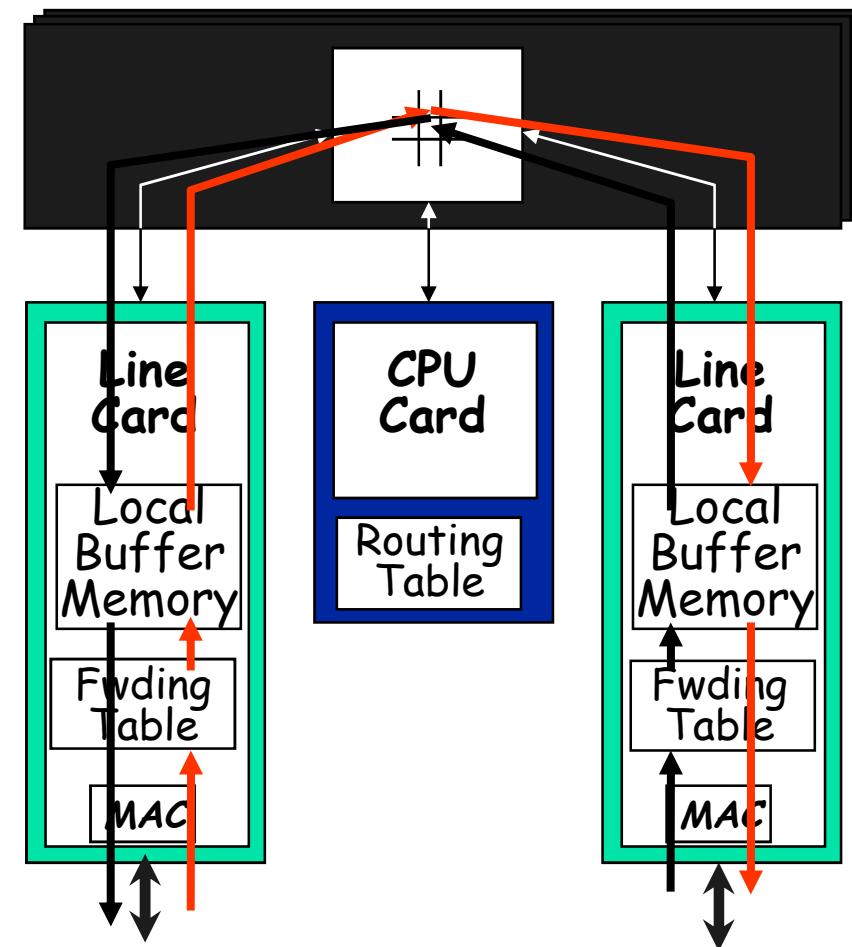
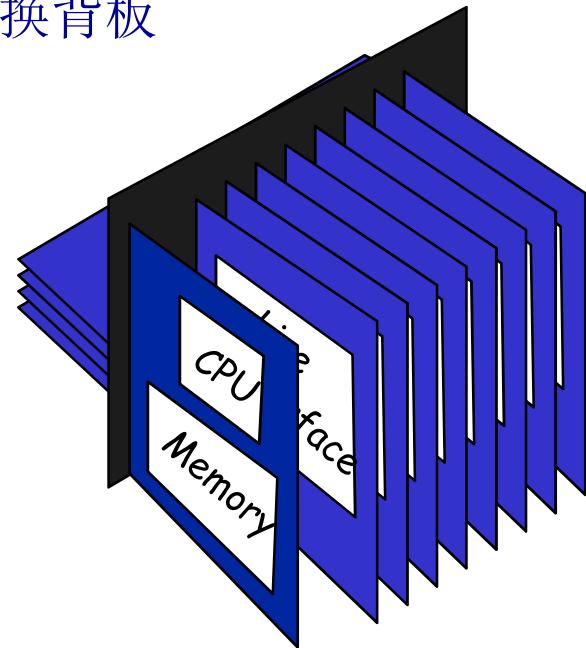


Typically <5Gb/s aggregate capacity



第三代路由器

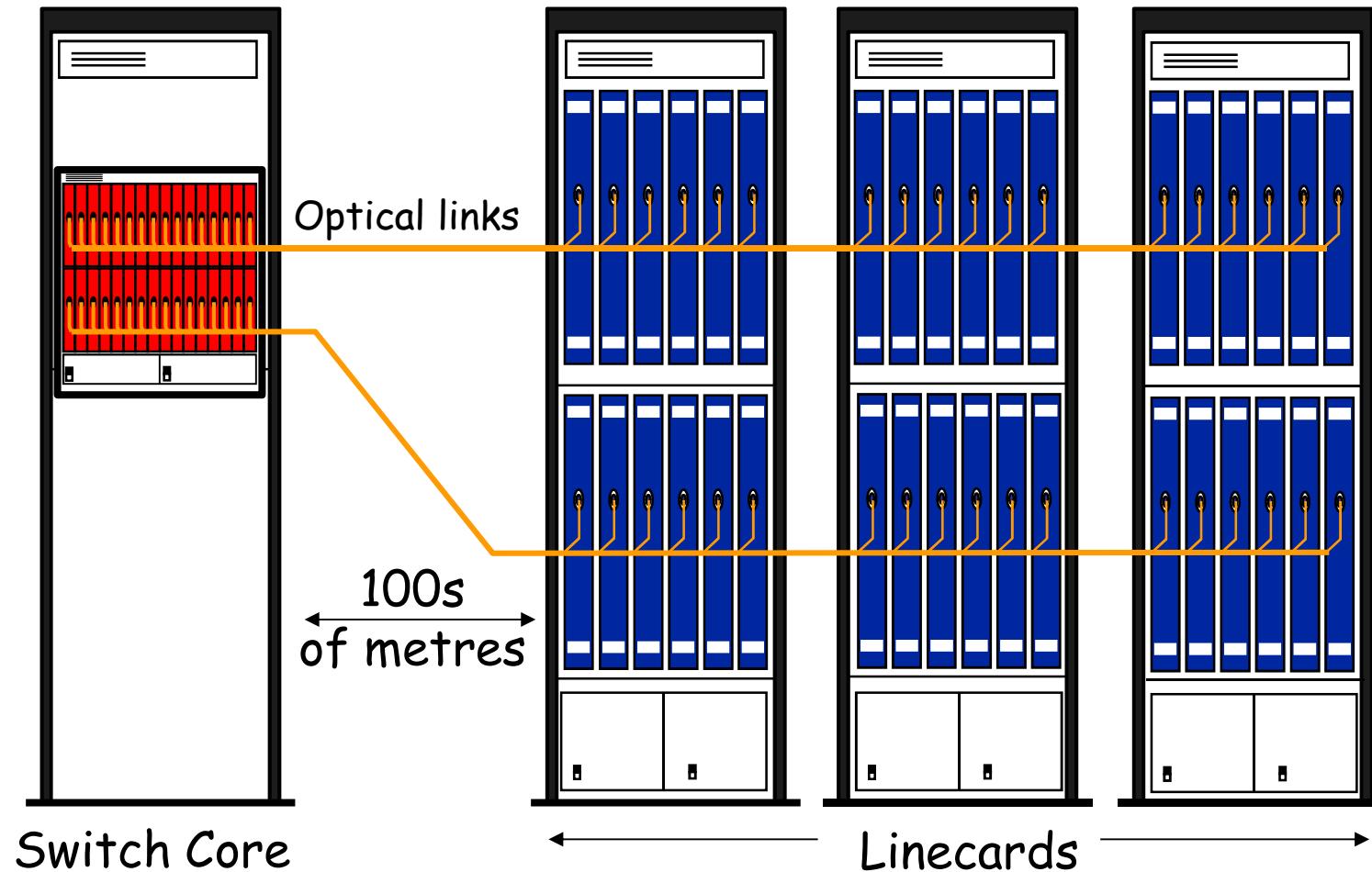
交换背板



Typically <1Tb/s aggregate capacity



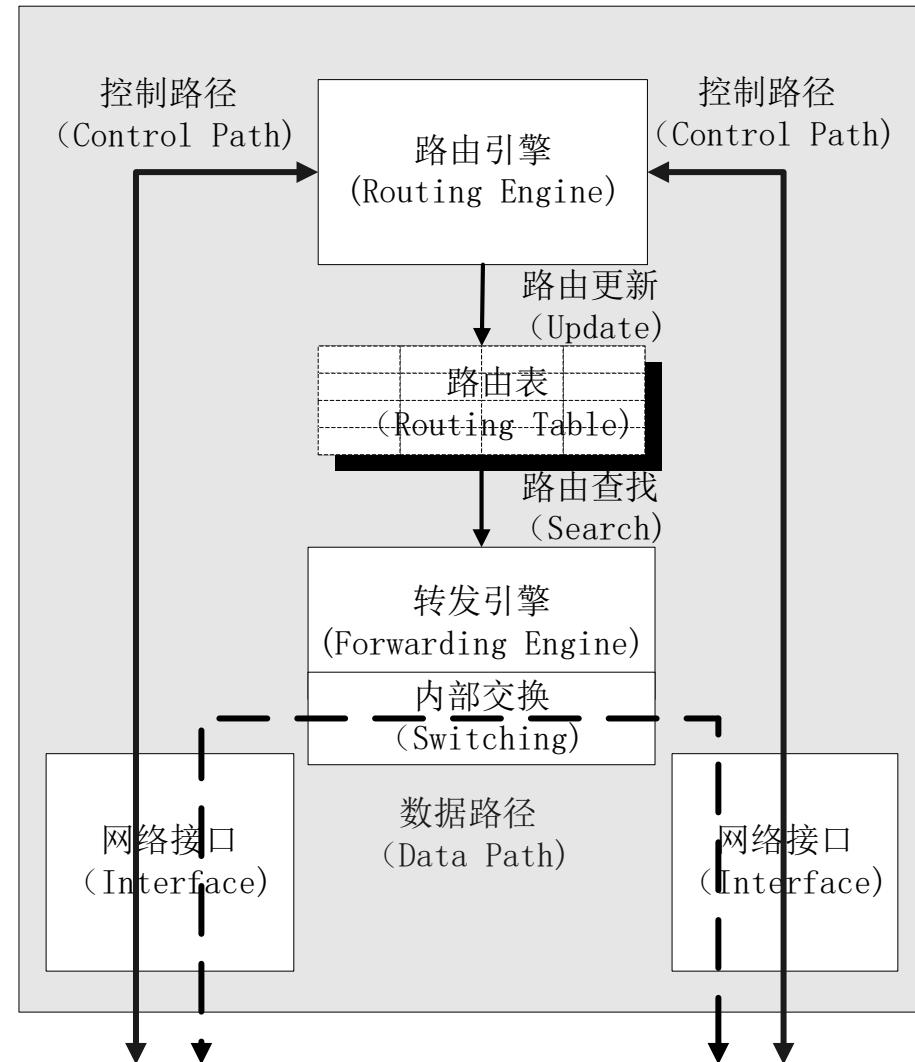
第四代路由器



1 - 几百 Tb/s routers in development



路由器基本结构



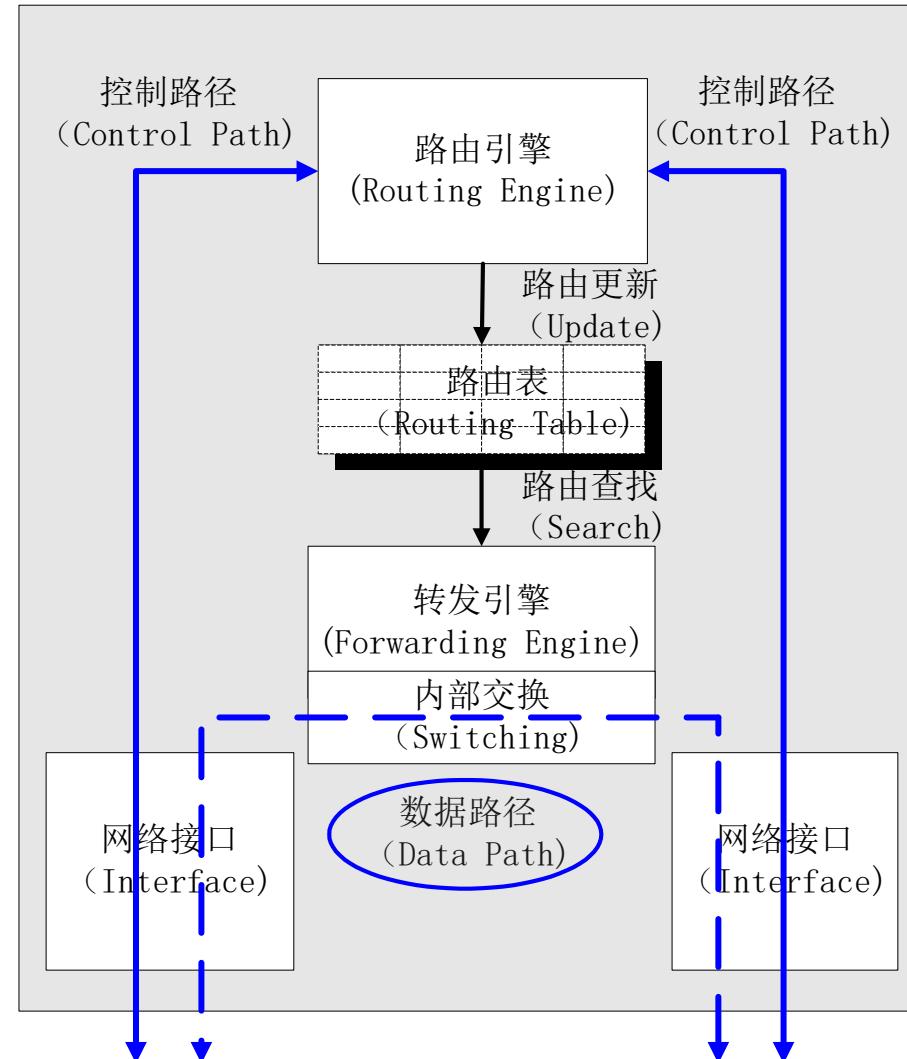


路由器基本结构（续）

- 网络接口
 - 完成网络报文的接收和发送
- 转发引擎
 - 负责决定报文的转发路径
- 内部交换
 - 为多个网络接口以及路由引擎模块之间的报文数据传送提供高速的数据通路
- 路由引擎
 - 由运行高层协议（特别是路由协议）的内部处理模块组成
- 路由表
 - 路由表分组含了能够完成网络报文正确转发的所有路由信息，它在整个路由器系统中起着承上启下的作用



分组处理路径



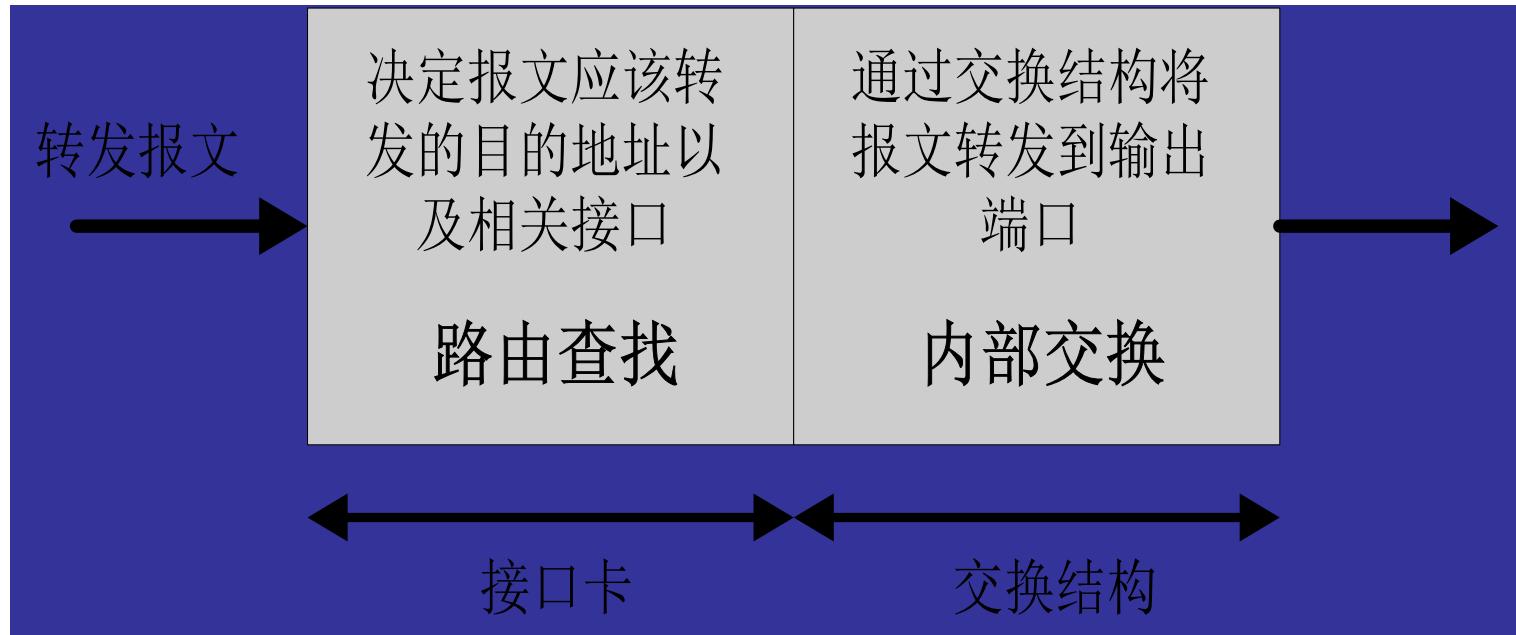


分组处理路径（续）

- 路由器提供了两种不同的报文处理路径：
 - **数据路径：**处理目的地址不是本路由器而需要转发的报文，因此数据路径是整个路由器的关键路径，它的实现好坏直接影响着路由器的整体性能
 - **控制路径：**处理目的地址是本路由器的高层协议报文，特别是各种路由协议报文。虽然控制路径不是路由器的关键路径，但是它负责完成路由信息的交互，从而保证了数据路径上的报文沿着最优的路径转发。



数据路径的工作流程



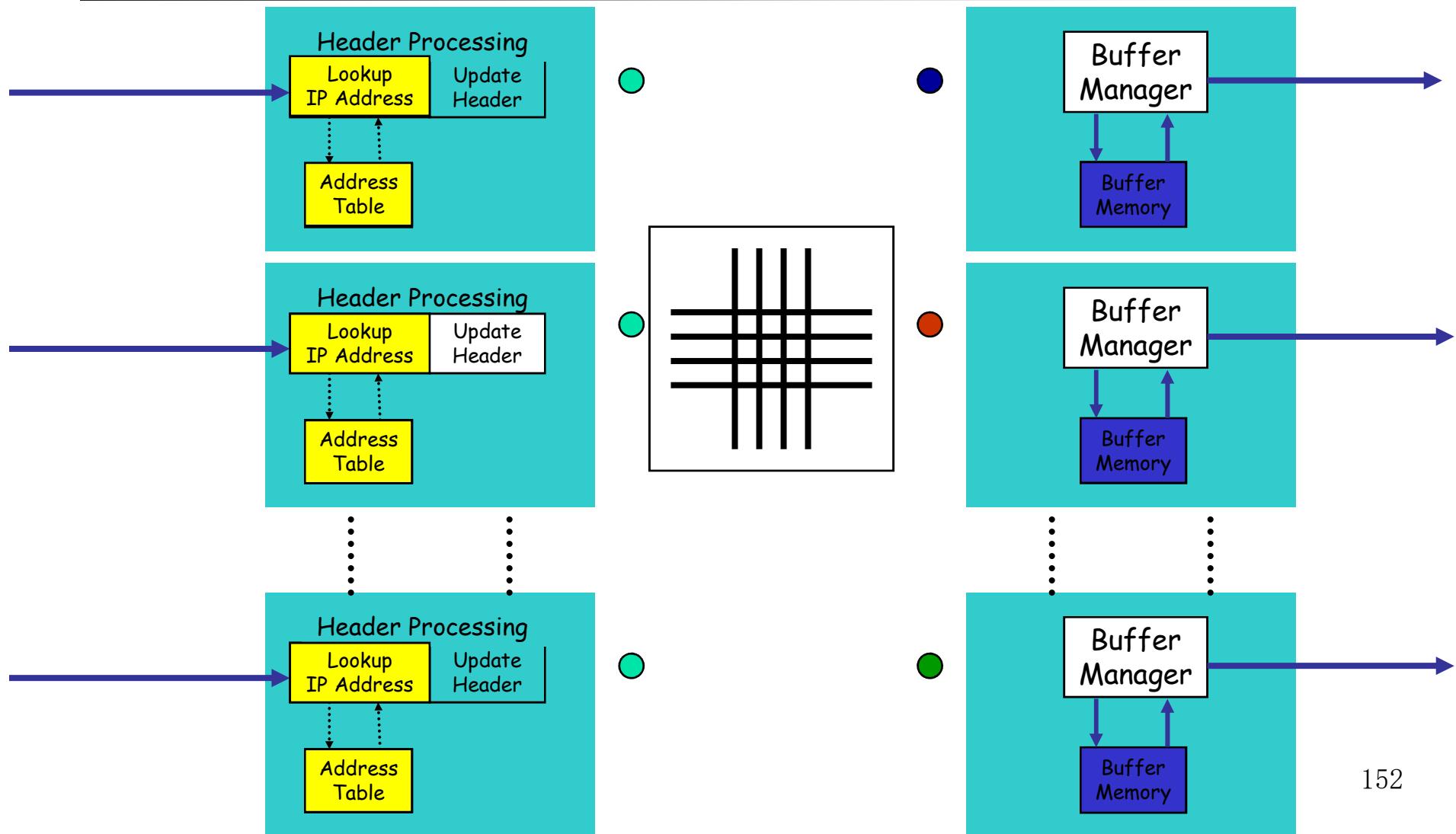


数据路径的工作流程

- RFC1812规定IP路由器必须完成两个基本功能
 - 首先路由器必须能够对每个到达本路由器的报文做出正确的转发决策，决定报文向哪一个下一跳路由器转发。为了进行正确的转发决策，路由器需要在转发表中查找能够与转发报文目的地址最佳匹配的表项，这个查找过程被称为路由查找（Route Lookup）
 - 其次路由器在得到了正确的转发决策之后必须能够将报文从输入接口向相应的输出接口传送，这个过程被称为内部交换过程（Switching）

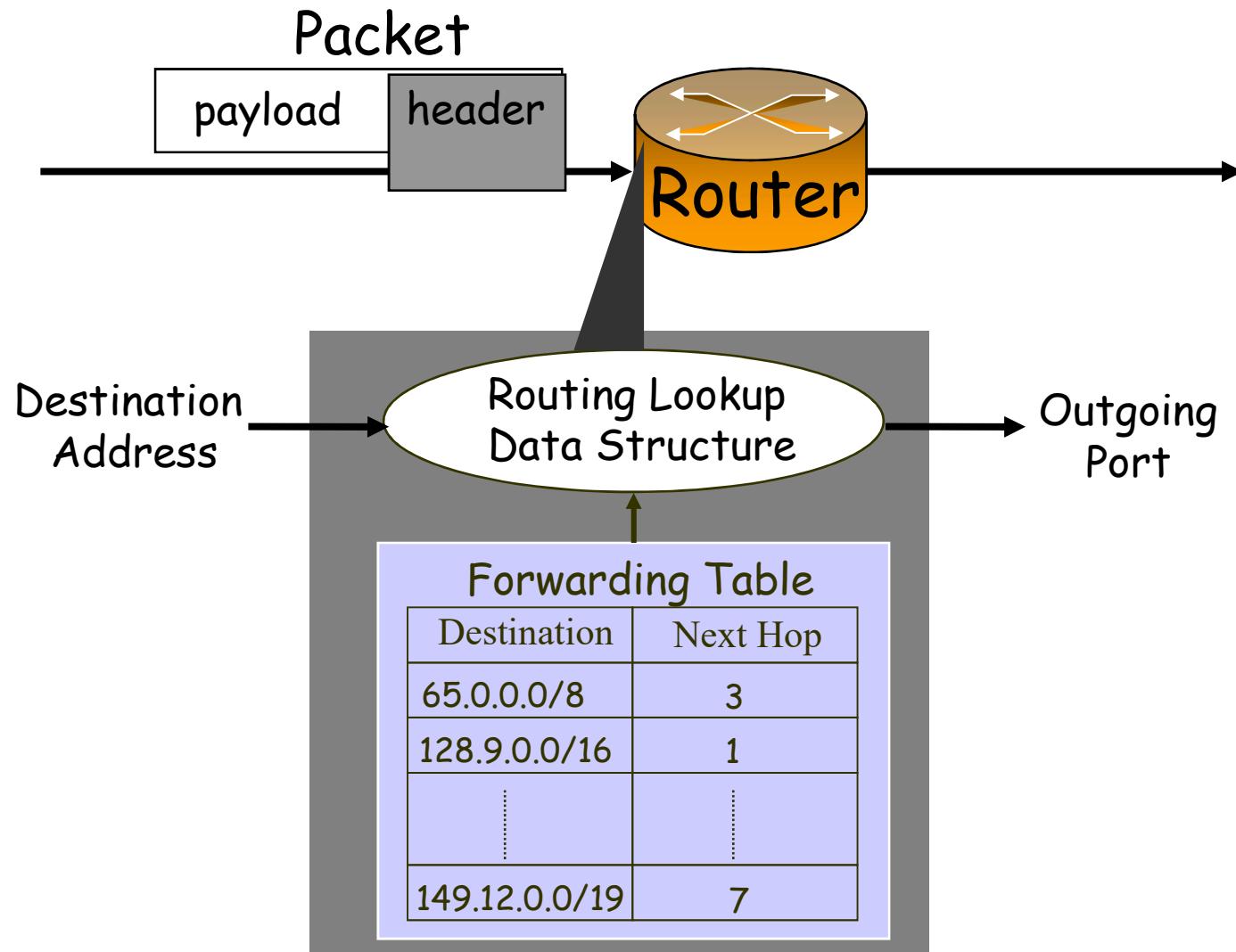


路由查找





路由查找过程示意图





路由查找

路由查找的难点在于：

1. 不是精确匹配，是最长匹配 .
2. 路由表很大，目前超过70万条表项，并且还在不断增长 .
3. 路由查找必须很快：对应10Gb/S的链路，速度要求是30ns .



路由查找算法的分类

- 基于地址前缀值的路由查找算法
 - 通过对整个地址前缀空间进行地址关键字穷举法来避免对地址前缀长度进行考虑。
 - 线性查找法、地址区间的二分查找法、TCAM硬件查找法
- 基于地址前缀长度的路由查找算法
 - 从前缀长度的角度入手进行路由查找
 - trie树(包括二分支、多分支)、前缀长度空间的二分查找法



线性查找法

Prefixes

a 0*

b 01000*

c 011*

d 1*

e 100*

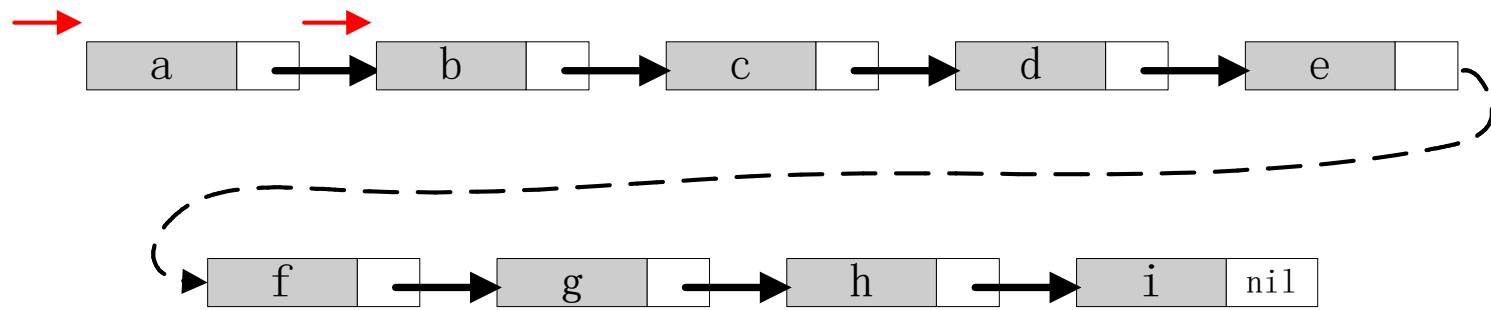
f 1100*

g 1101*

h 1110*

i 1111*

01000



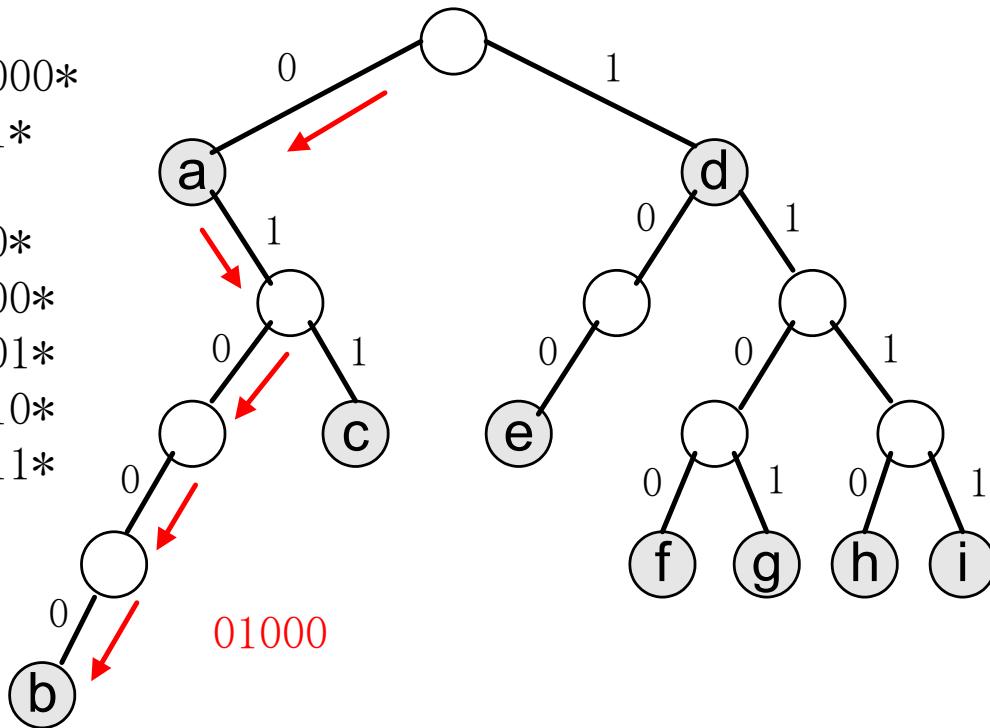
- 实现简单
- 查找效率低，查找过程的算法复杂度为 $O(N)$
- 存储空间复杂度为 $O(N)$ ，插入删除复杂度为 $O(1)$
- 一种改进：将前缀长度大的路由表项放在链表的前列，平均查找性能会有改进，但是路由更新复杂度变为 $O(N)$
- 只适用于路由表项比较少的早期低端路由器



二分支trie树查找法 (A Binary Trie)

Prefixes

a 0*
b 01000*
c 011*
d 1*
e 100*
f 1100*
g 1101*
h 1110*
i 1111*



- 数字查找树的每个结点不是关键字，而是组成关键字的符号
- 实现简单
- 适用性好，可以用于任何长度关键字的查找
- 查找效率低，最差情况下 $O(W)$ ， $W = \log N$
- 存储效率较低
- Trie树查找过程实际上就是在长度空间内的顺序查找操作



路径压缩trie树

(Path-Compressed Trie)

Prefixes

a 0*

b 01000*

c 011*

d 1*

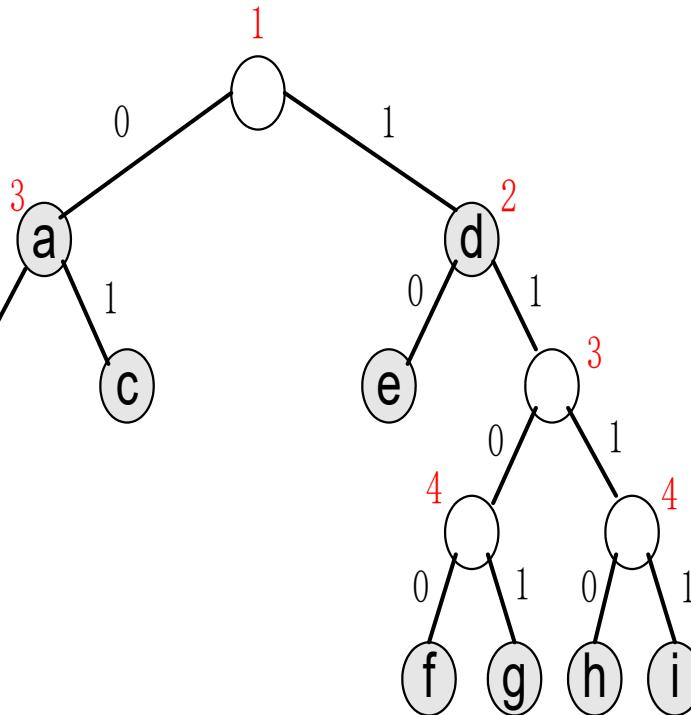
e 100*

f 1100*

g 1101*

h 1110*

i 1111*



- **Trie**树中单分支结点的存在既增加了搜索的深度，又增加了存储空间
- 由于忽略了地址中某些位的匹配操作，结点处需要有一个变量指示下一个要检查的位
- 路径压缩**trie**树前缀结点保存的是地址前缀
- 当到达叶子结点或前缀匹配失败时，查找结束
- 最早在**PATRICIA**算法中提出，并在**BSD UNIX**中实现。

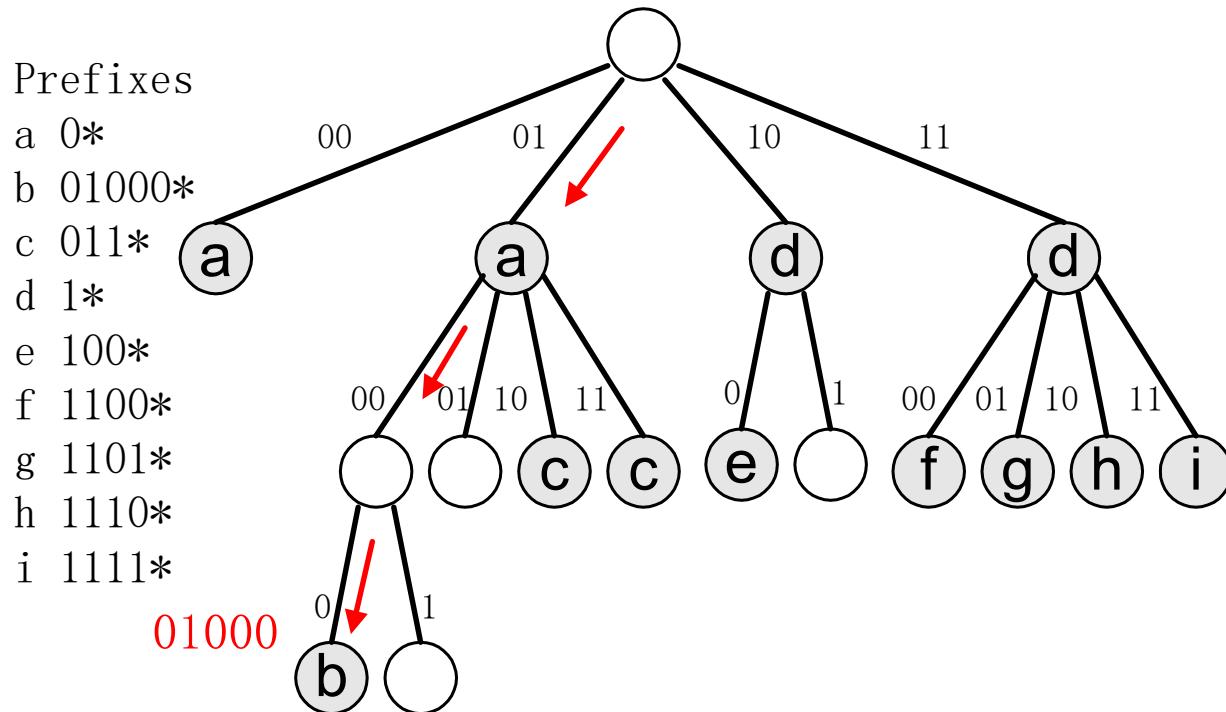


查找算法使用的辅助策略

- 前缀扩展 (Prefix Expansion)
 - 将一条长度较短的前缀展开成多条长度较长的前缀集
 - 前缀扩展技术可以把包含各种前缀长度的前缀集转化为只包含较少前缀长度的前缀集
- 独立前缀转化 (Disjoint Prefix Transformation)
 - 为了解决前缀地址空间的重叠和最长匹配问题，可以将地址前缀集转化为完全独立的前缀集
 - 根据独立前缀集构造的trie树中所有前缀结点都出现在叶子上，也称为叶子扩展 (leaf pushing) 技术
- 压缩技术 (Compression Techniques)
- 优化技术 (Optimization Techniques)
- 存储层次 (Memory Hierarchy)



多分支trie树查找法



- 优点

- 提高了查找效率，复杂度为 $O(W/k)$ ，其中 k 为trie树步宽

- 缺点

- 存储空间的需求大，利用率不高
- 更新效率低

- 前缀扩展

- 图中a(0*)扩展成a(00*)以及a(01*)



基于大容量RAM的快速路由查找算法

Table16表项结构

0	路由转发信息
1	TableNext表指针

TableNext表项结构

路由转发信息

表Table16

...		...
165.255	-	--
166.0	0	A
166.1	0	A
...		...
166.110	0	A
166.111	1	
166.112	0	A
...		...
166.255	0	A
167.0	-	--
...		...

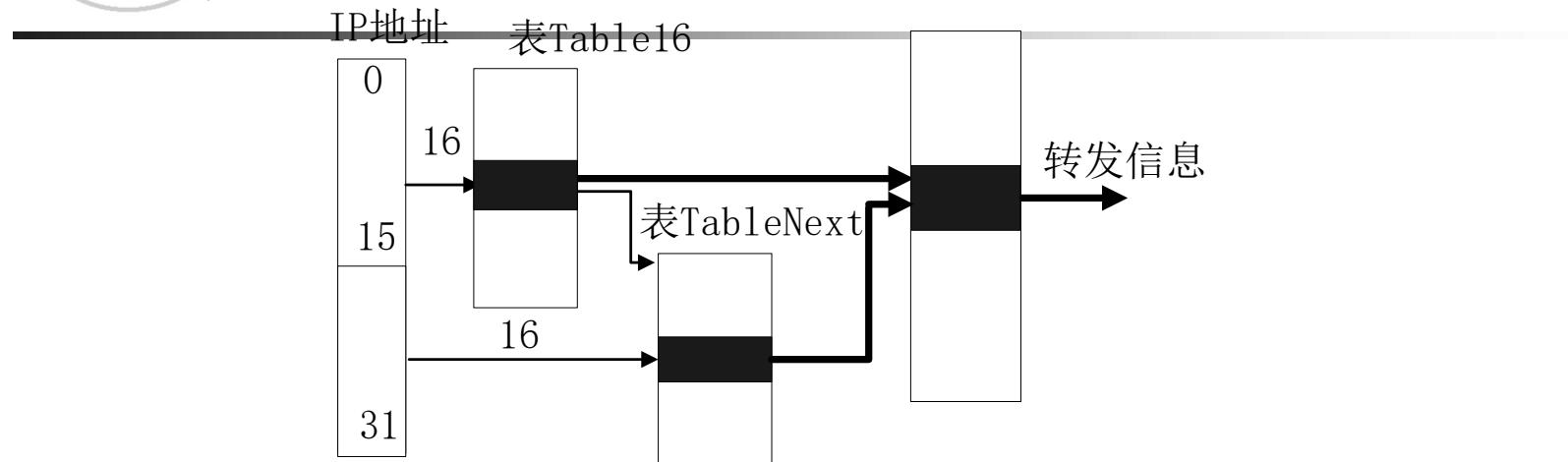
表TableNext

0*256+0	B
...	..
0*256+255	B
...	..
68*256+0	C
...	..
68*256+255	C
69*256+0	B
...	..
69*256+255	B
...	..

- 采用多分支trie树查找算法思想，trie树的深度为**2**，步宽分别为**16/16**（或**24/8**）
 - 。
- 算法采用两种查找表结构，分别为**Table16**和**TableNext**。**Table16**表相当于多分支trie树的第一层结点，它主要保存那些路由地址前缀小于等于16的表项；**TableNext**表相当于多分支trie树的第二层结点，主要保存那些路由地址前缀大于16的表项。
- 例如加入路由项**166/8(A)**，**166.111/16(B)**，**166.111.68/24(C)**，查找表如左图所示。
- 在查找表中查找地址**166.1.2.3**和**166.111.1.2**



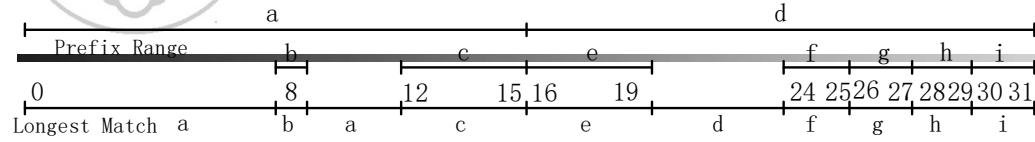
基于大容量RAM的快速路由查找算法



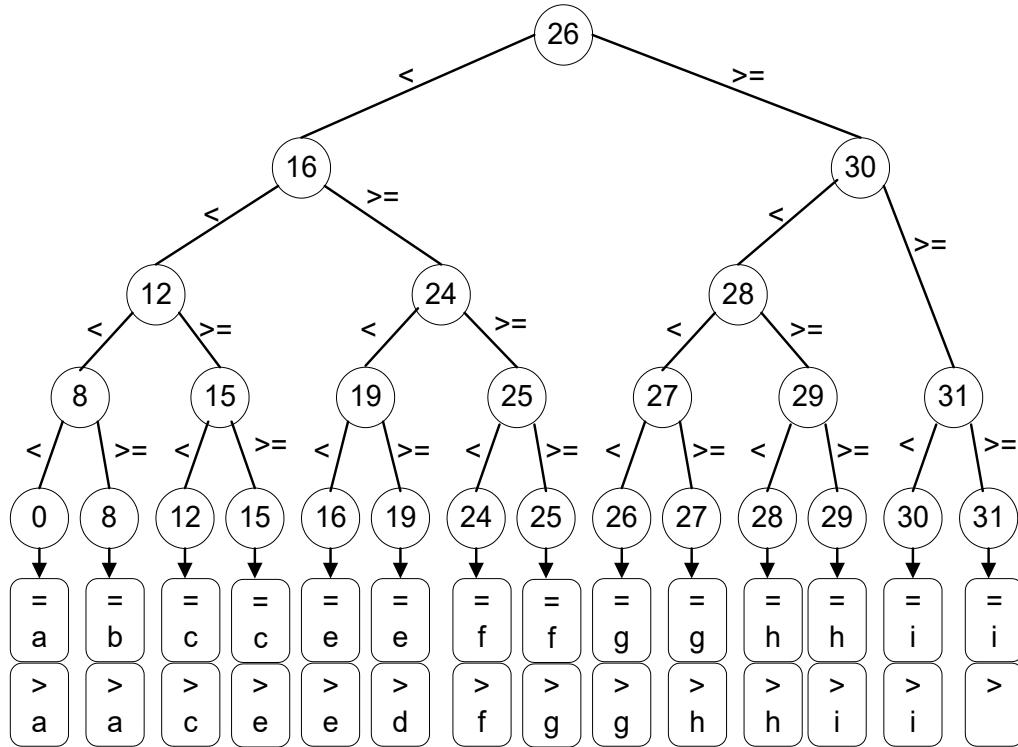
- 对于任何一个IP地址，查找过程最多只需要访问两次查找表（Table16和TableNext）就可以得到转发信息，大大加快了路由查找的速度。
- 如果我们在实现中将Table16和TableNext表从物理上分开（比如说采用两个不同的RAM），那么访问不同的表就可以同时进行，从而可以使用流水线查找。在查找过程完全满足流水线操作的条件下，查找只需要一次存储器访问，达到了查找算法的最高性能。



地址区间的二分查找法



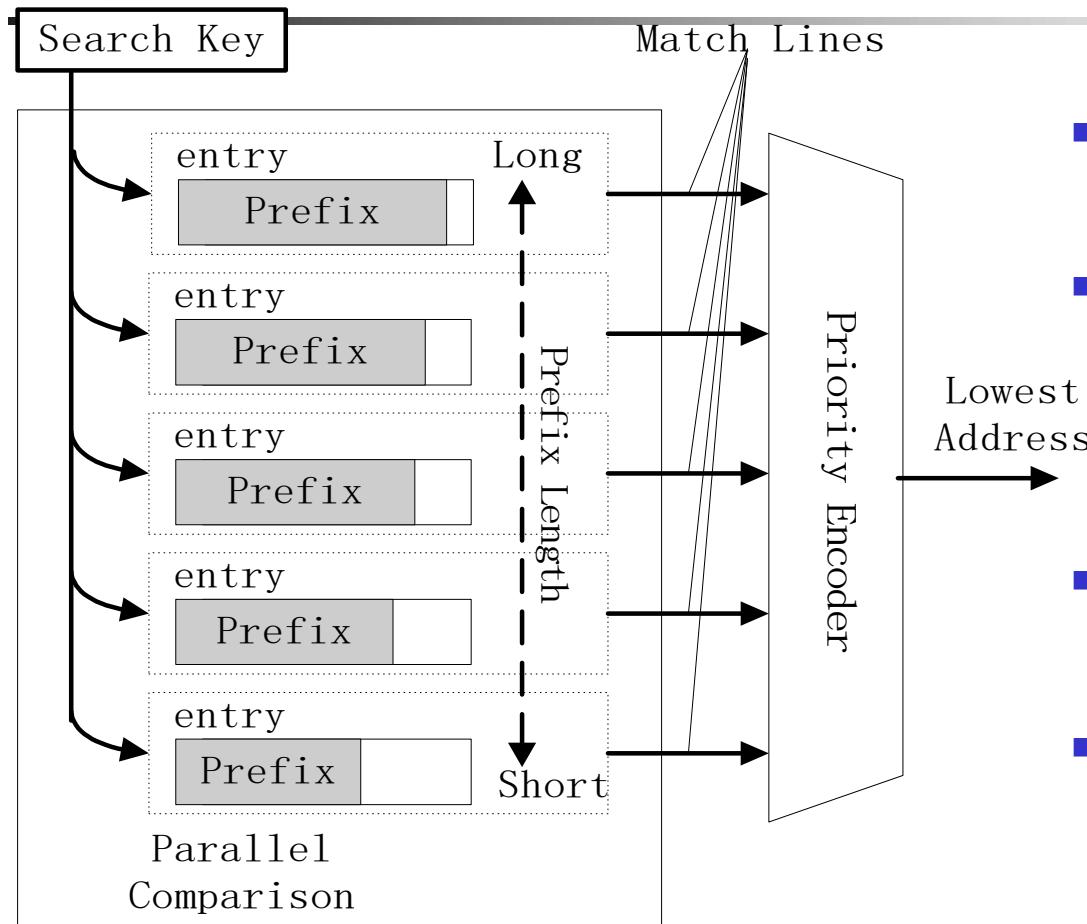
前缀对应的地址区间图



- 地址前缀在整个地址空间内代表了一段连续的地址区间
- 任何地址区间对应的最长前缀应该是包含此区间的前缀中地址范围最窄的一项
- 例: **10110 (22)**
- N个地址前缀, 查找算法复杂度为O ($\log_2 N$)
- 改进: 多路查找算法, 复杂度降为O ($\log_k N$)
- 区间二分查找法的最大问题是地址前缀的更新



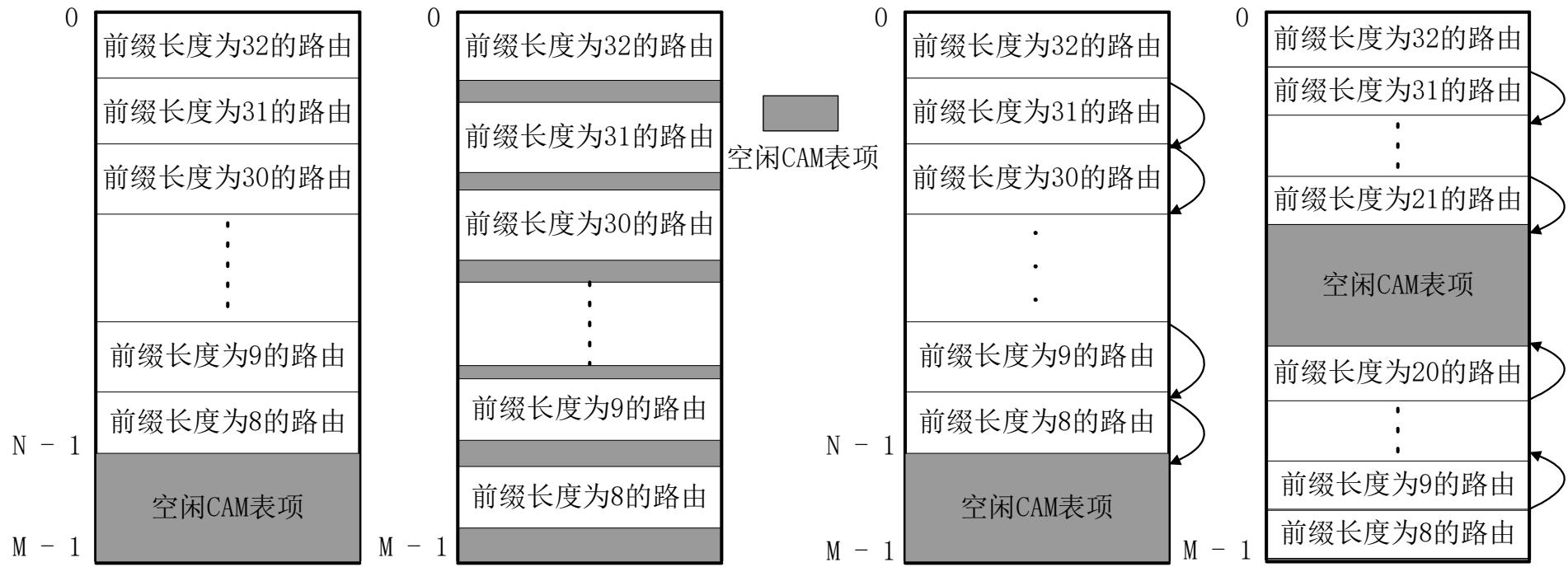
TCAM (Ternary Content Addressable Memory)



- 在一个硬件时钟周期内完成关键字的精确匹配查找
- TCAM规定在所有匹配的表项中选取地址最低的表项作为最后结果，因此需要保证在低地址存储前缀较长的前缀
- 优点
 - **查找速度快(15-20ns)**
- 缺点
 - **单位容量的芯片价格高**
 - **功耗较大**
 - **更新效率低**



TCAM 路由更新



顺序移动法 -- $O(N)$

预留表项空间的顺序移动法 -- $O(N)$

选择移动法 -- $O(W)$

改进的选择移动法 -- $O(W/2)$

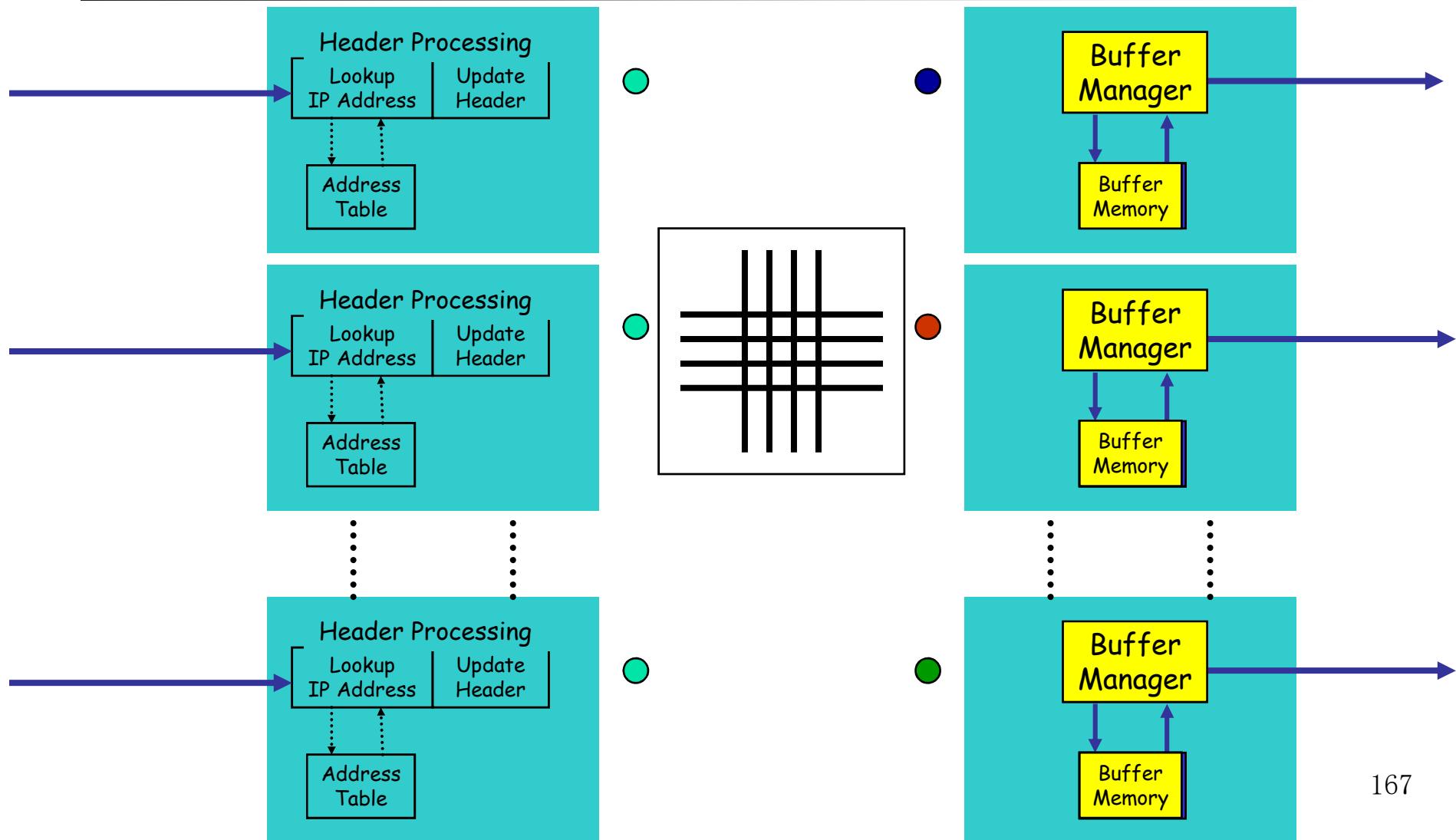


路由查找算法的评价标准

- 查找速度 (Speed)
- 存储容量 (Storage)
- 预处理和更新速度(Preprocessing and Update Speed)
- 算法实现的灵活性 (Flexibility in Implementation)
- 算法的可扩展性 (Scalability)



报文缓存

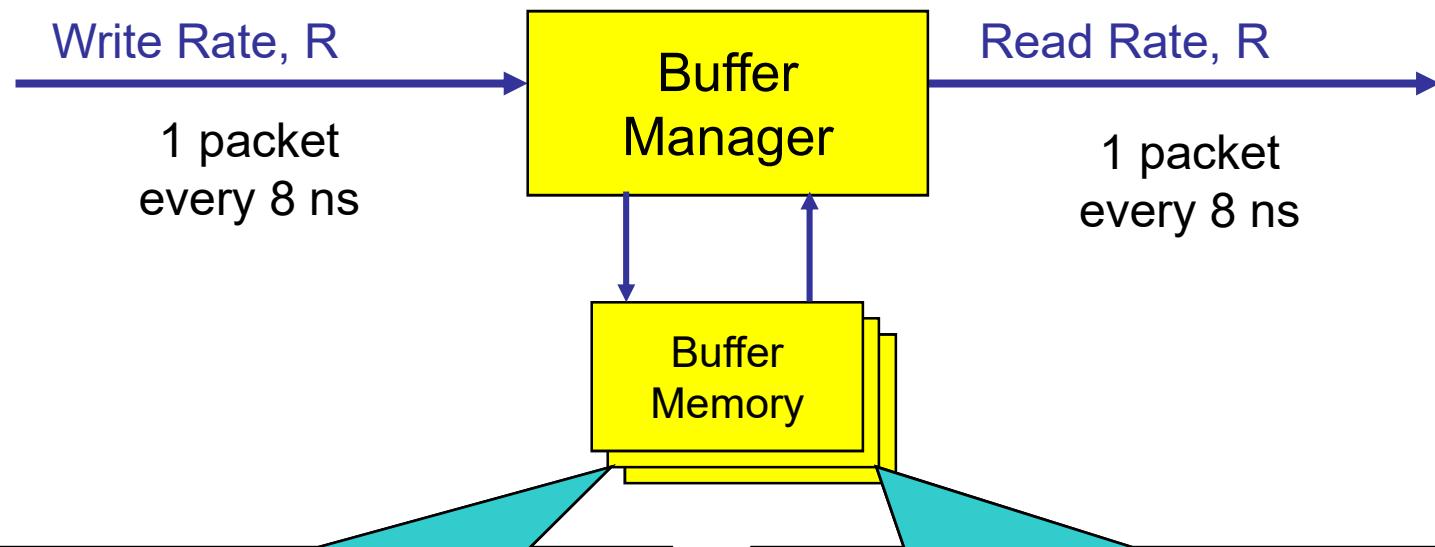




快速报文缓存

Example: 40Gb/s packet buffer

40 byte packets



使用SRAM?

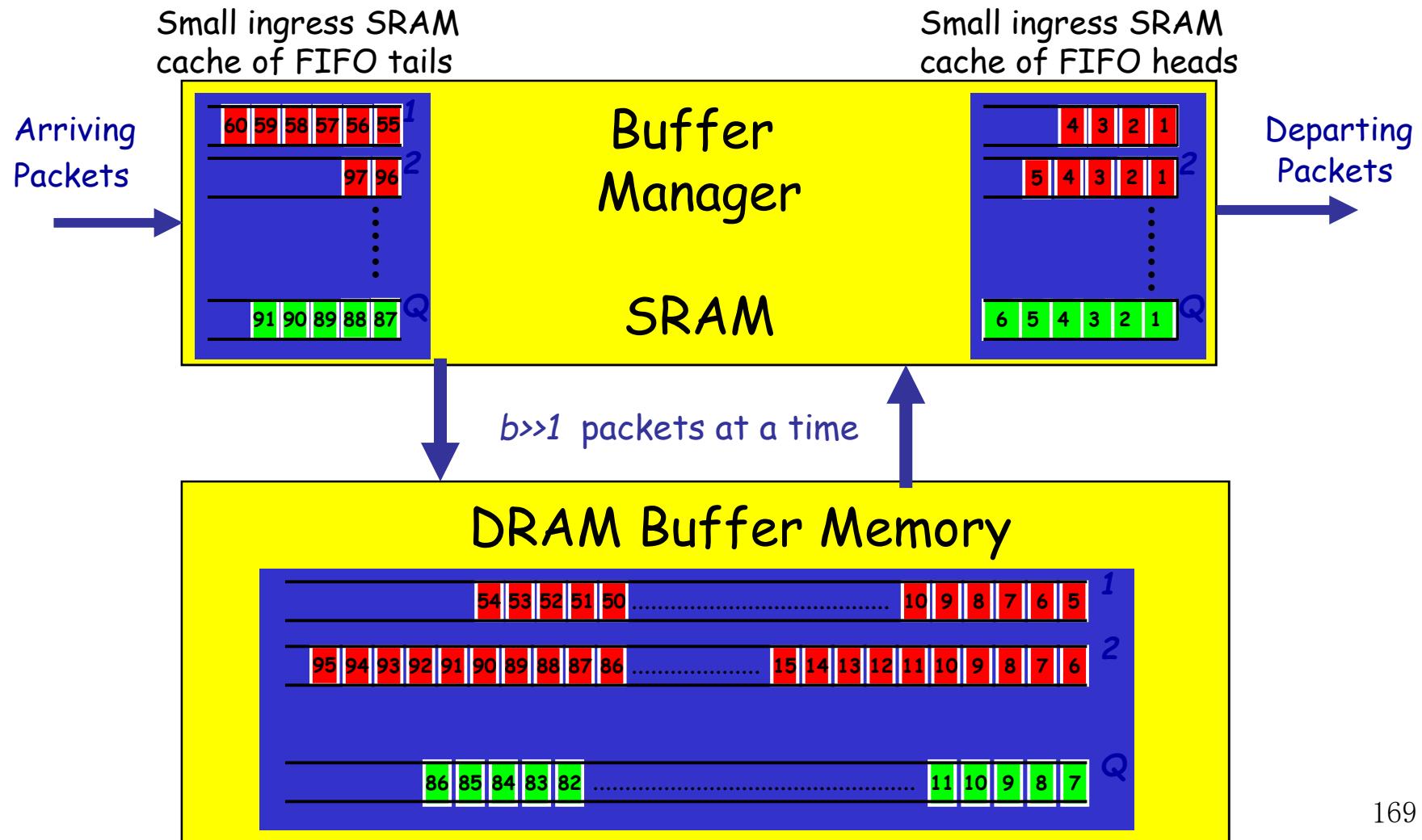
- + 速度够快
- 但容量不够.

使用DRAM?

- + 容量足够
- 但速度太慢.

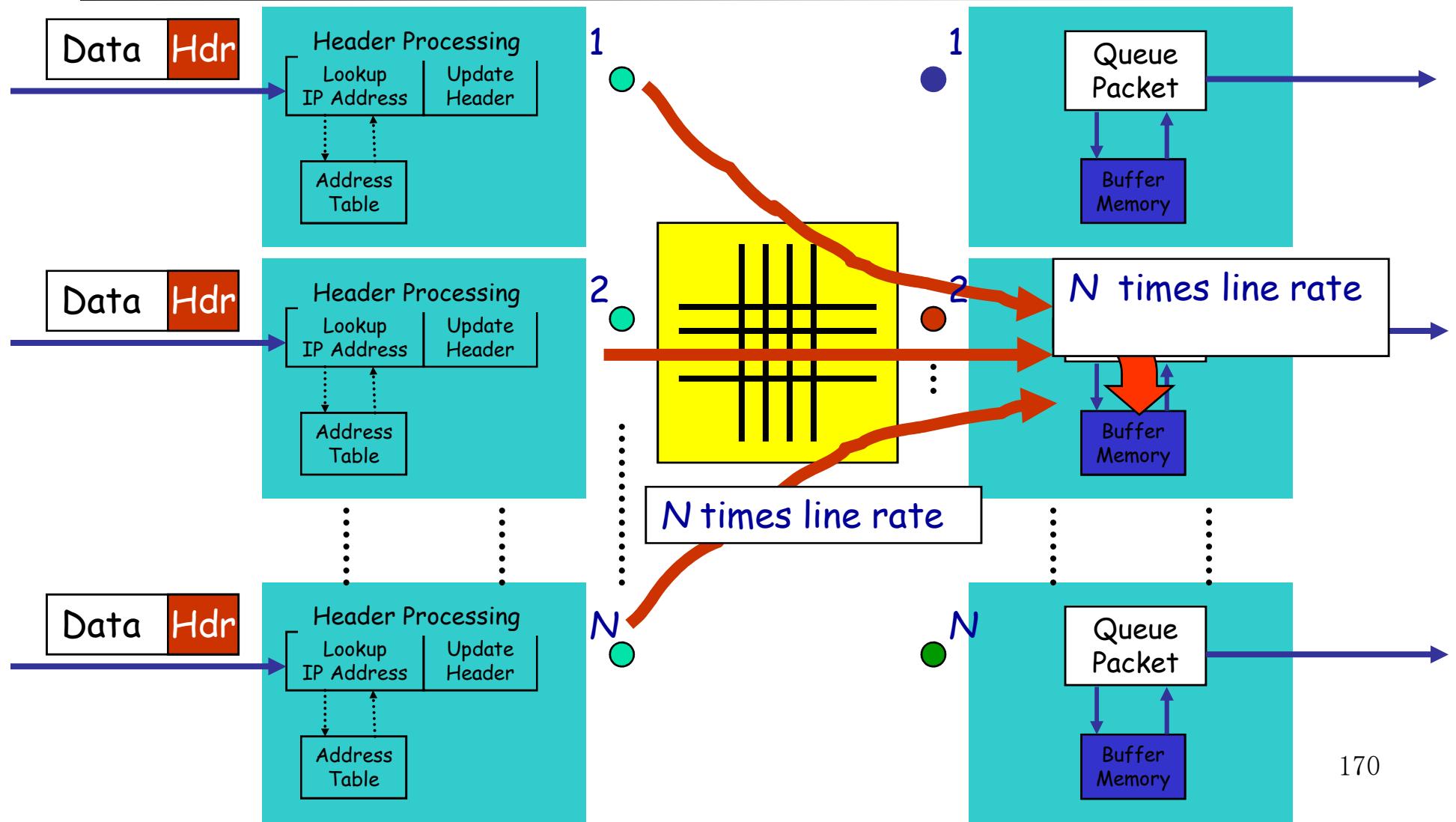


快速报文缓存（续）



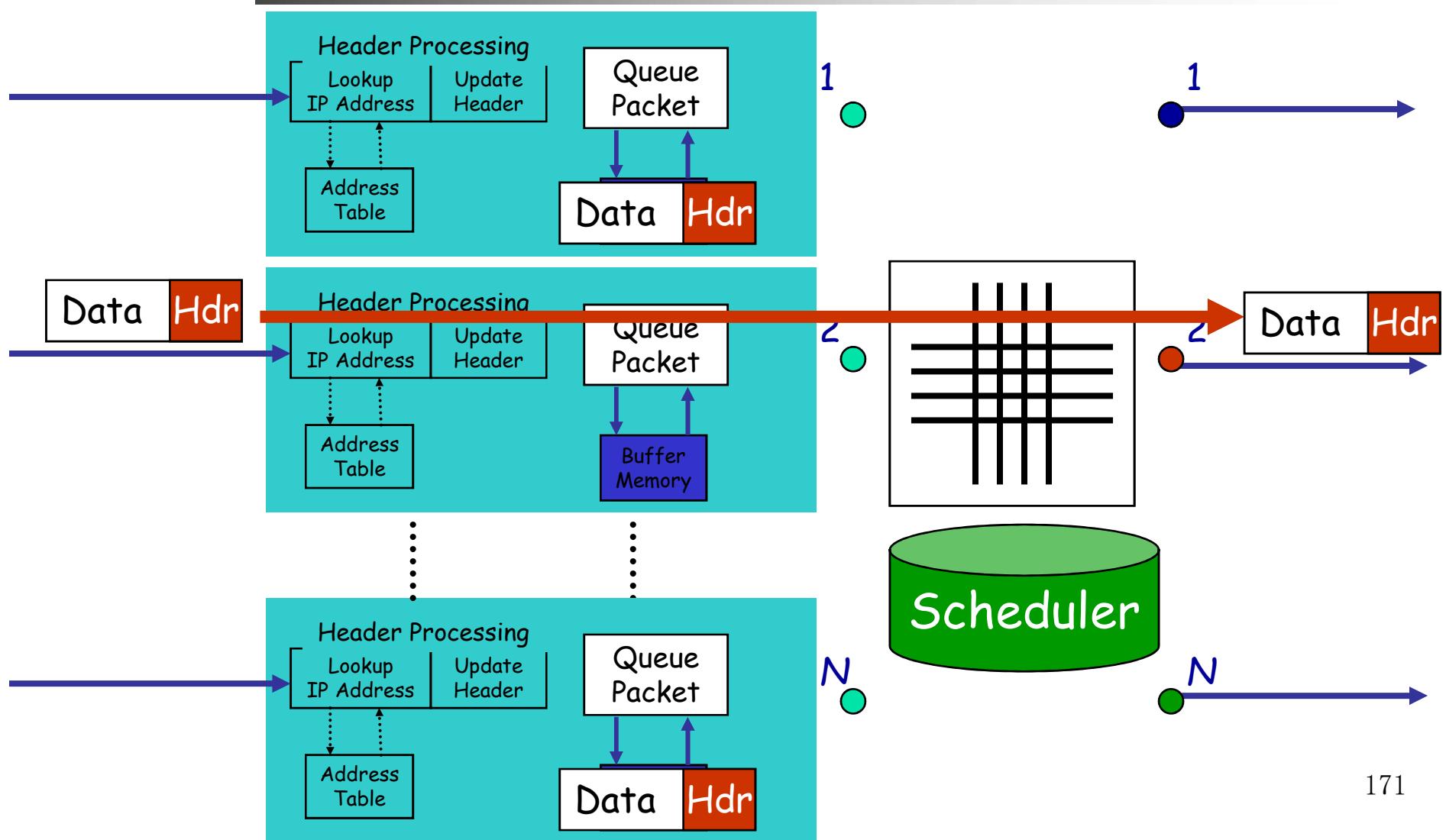


交換

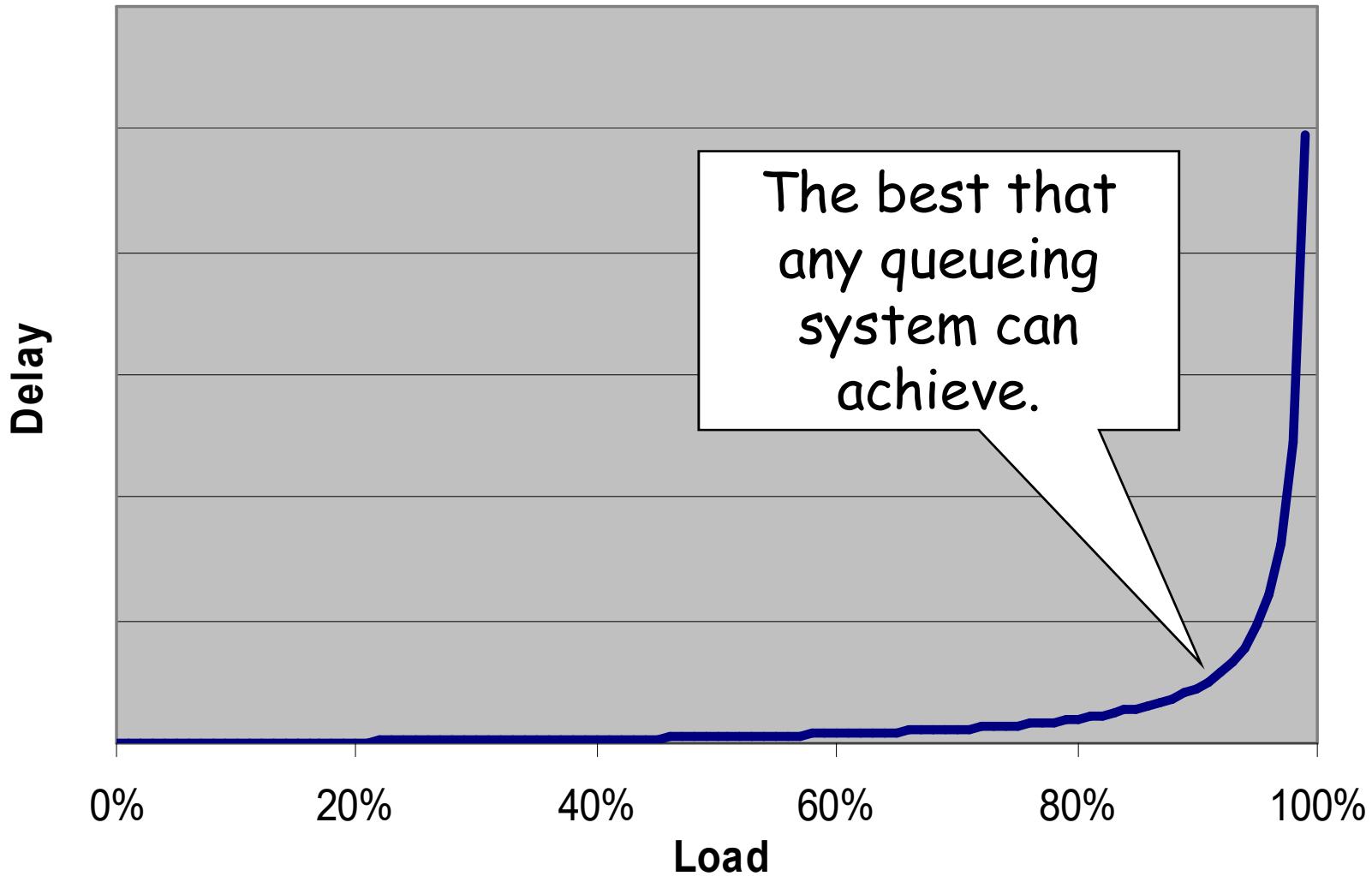




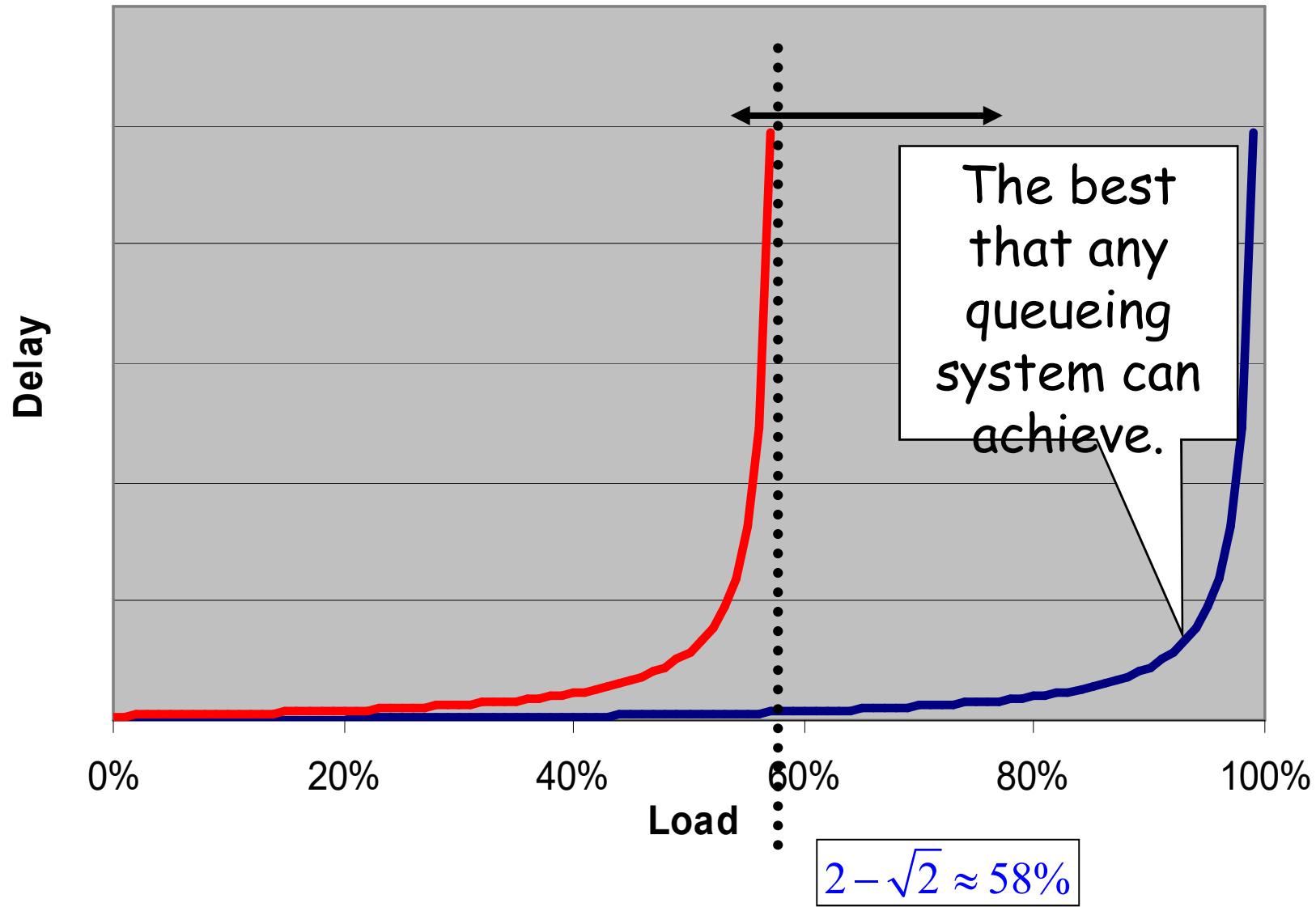
交换



使用输入队列的路由器

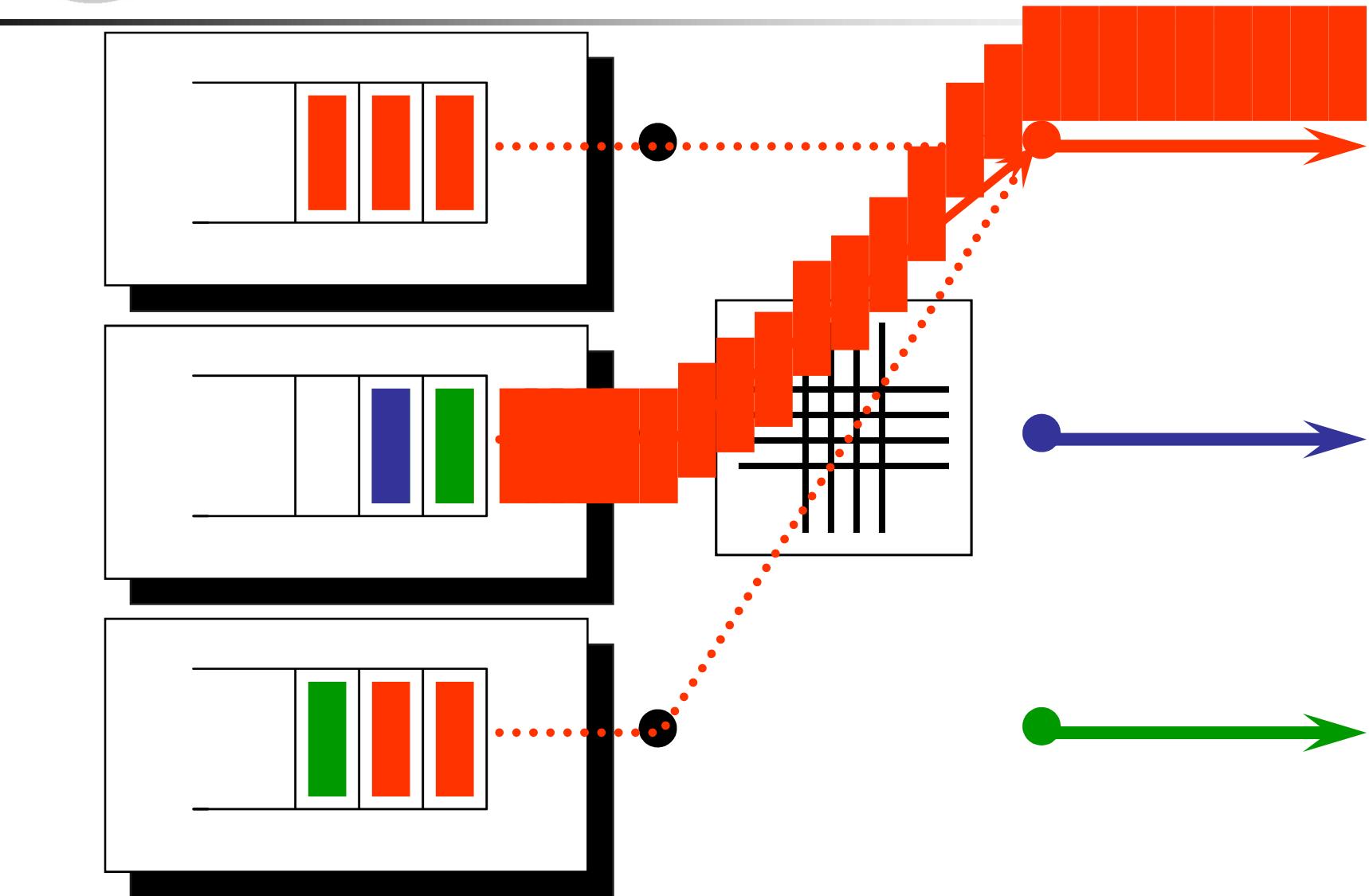


使用输入队列的路由器队头阻塞



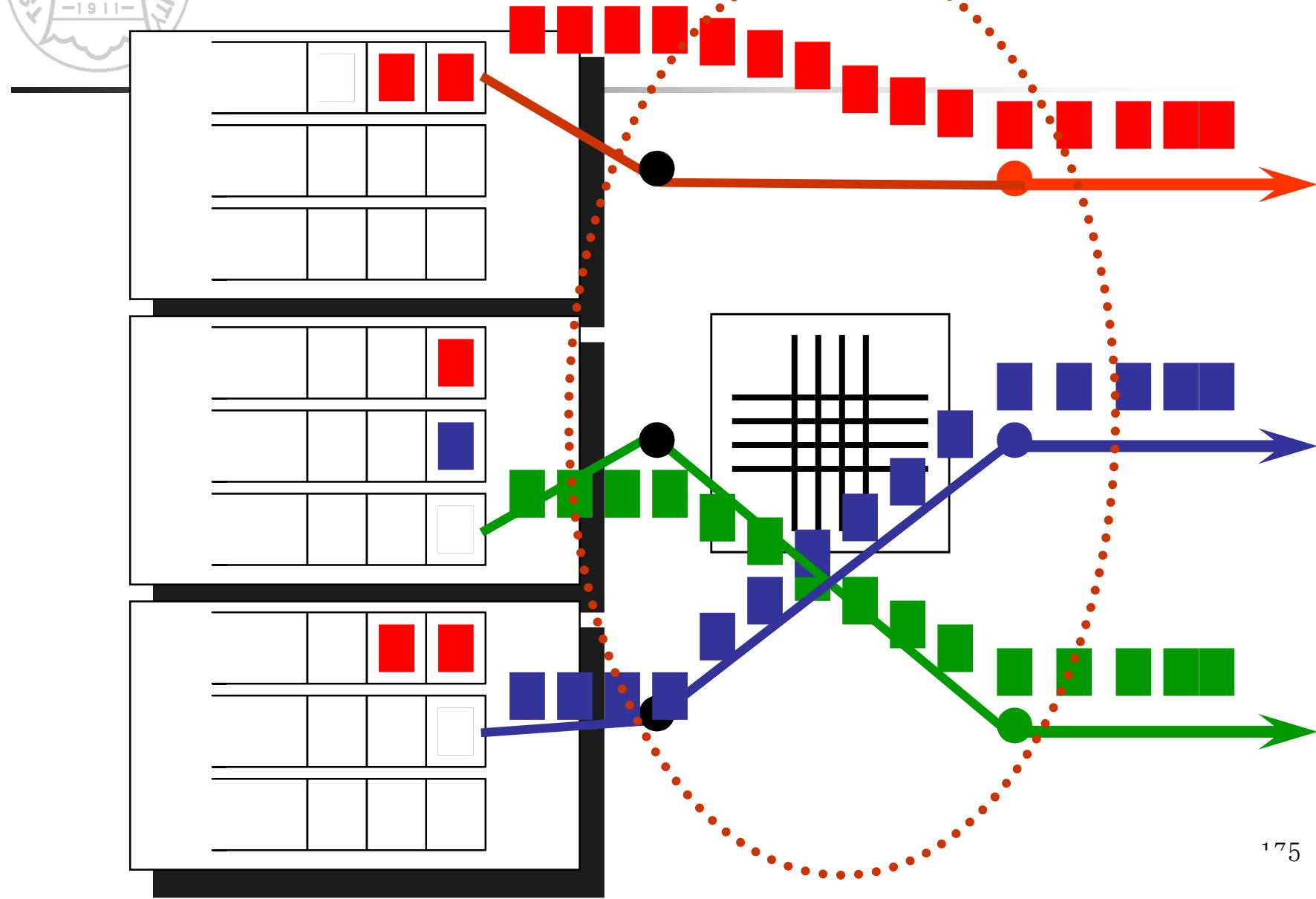


队头阻塞





虚拟输出队列



使用虚拟输出队列的路由器

