

计算机网络:自顶向下方法

肖周芳

计算机学院 1教606

计算机网络:自顶向下方法（第7版）

J.F.Kurose, K.W.Ross著,陈鸣译,机械工业出版社,2018.

Computer Networking: A Top-Down Approach(Sixth Edition)

J.F.Kurose, K.W.Ross,2017.

本PPT改编自英文版教材附带的PPT。

Network Layer 4-1

Chapter 4: Network Layer

目标:

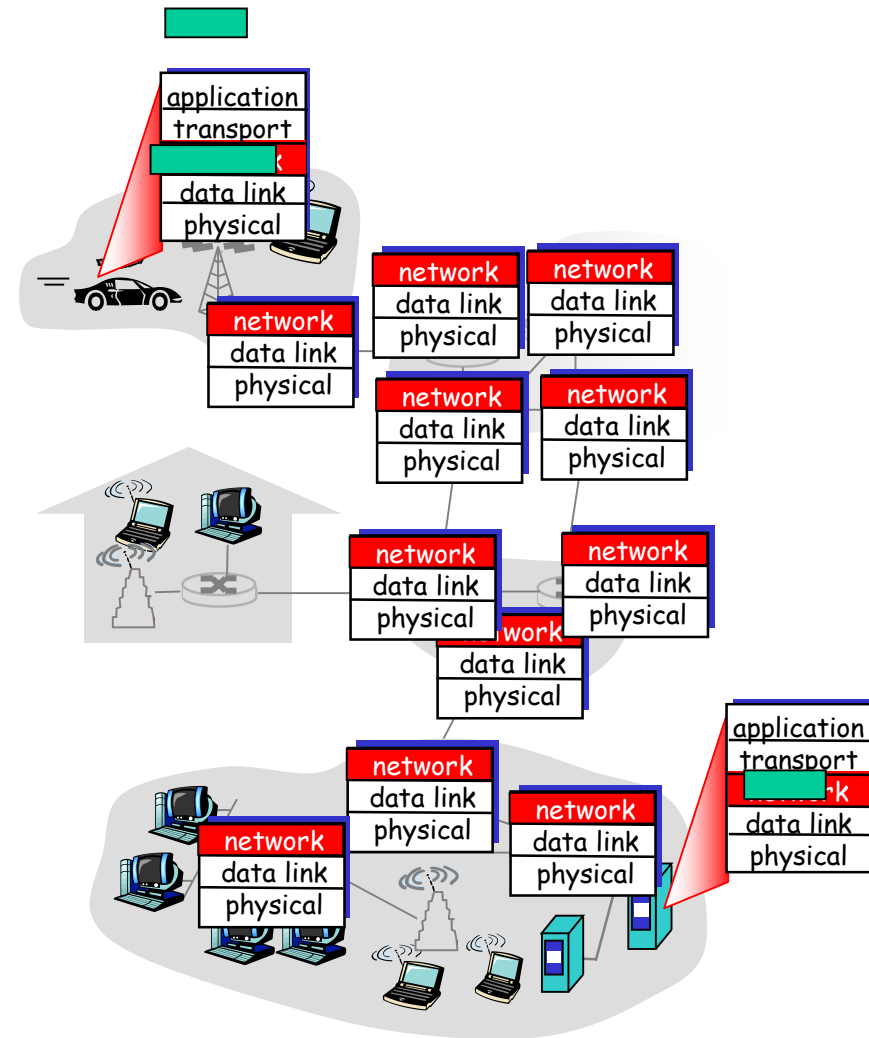
- ❑ 理解网络层服务的原理:
 - 网络层服务模型
 - 前向转发和路由
 - 路由如何工作
 - 路由(路径选择)
 - 如何扩展
 - 最新技术: IPv6, 移动性
- ❑ 实例, IP技术在 Internet 中的运用

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由

Network layer

- ❑ 从发送主机向接收主机传输报文段segment
- ❑ 发送端主机, 封装段segments成为数据报 datagrams
- ❑ 接收端主机, 分发段segments至传输层
- ❑ 网络层主机工作于 **每一个** 主机, 路由
- ❑ 路由检查经过该点的所有IP数据报头部(header fields)



网络层的两个关键功能

□ 转发:

forwarding: 将数据报从路由的输入端口前向转发到相应的输出端口

□ 路由:

routing: 决定数据报从源到目的地所经过的路径

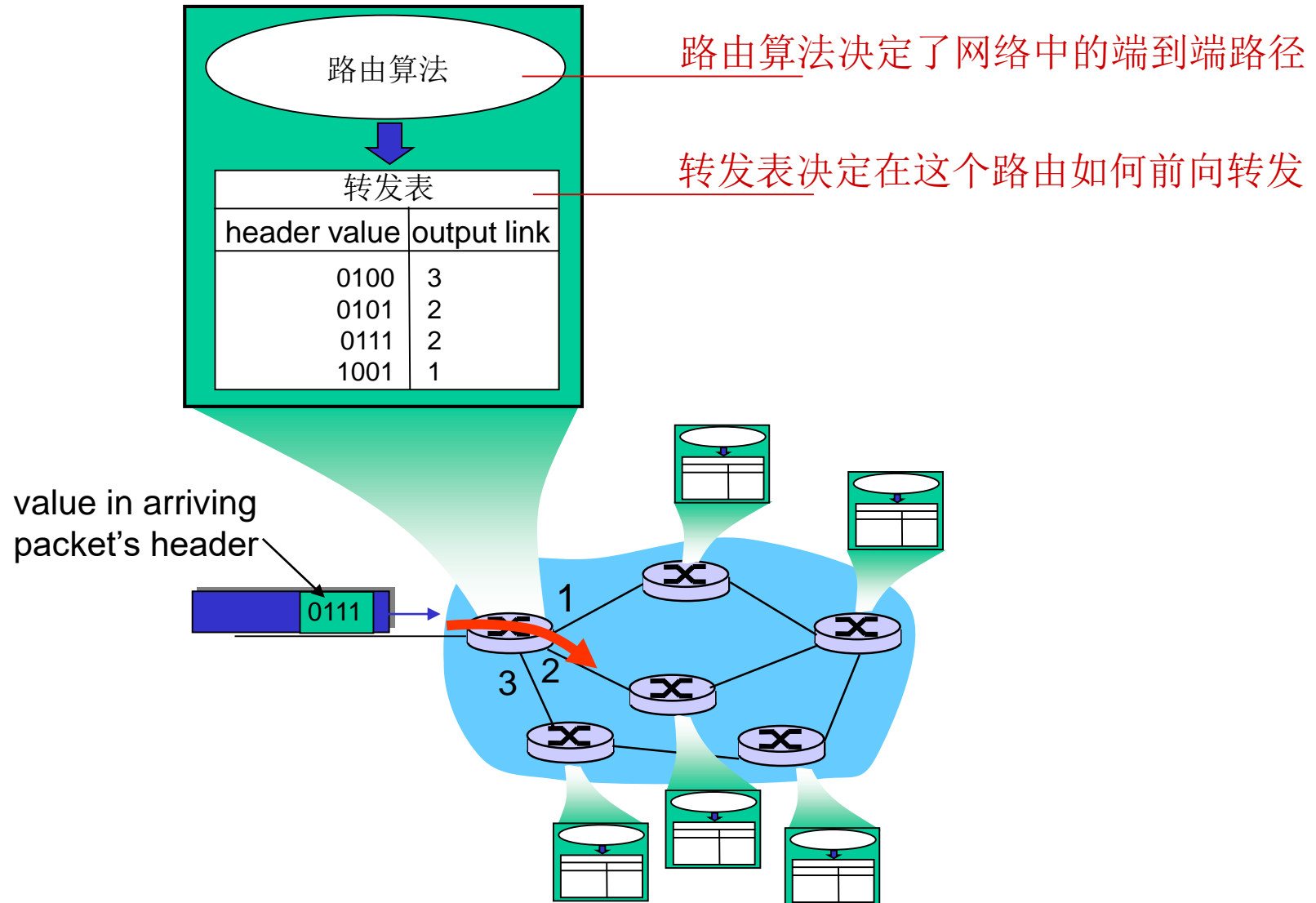
路由算法: *routing algorithms*

analogy:

□ *routing*: 从源到目的地路径的计划过程

□ *forwarding*: 经过单个交换结构-路由的过程

路由和转发的关系

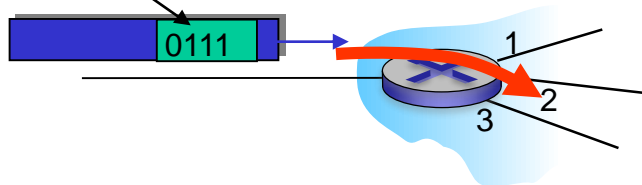


网络层: 数据平面, 控制平面

数据平面

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

values in arriving packet header

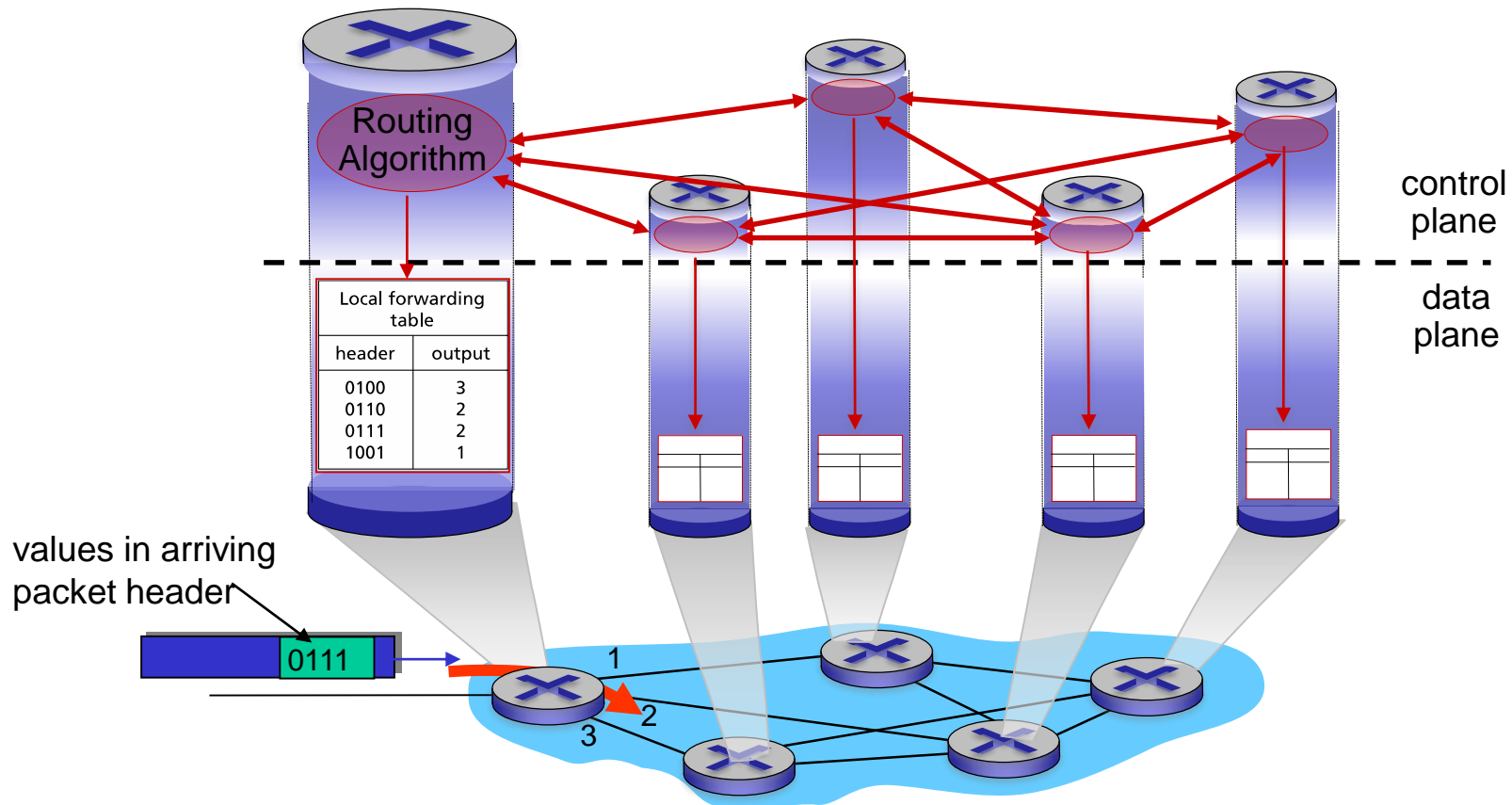


控制平面

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
 - *traditional routing algorithms*: implemented in routers
 - *software-defined networking (SDN)*: implemented in (remote) servers

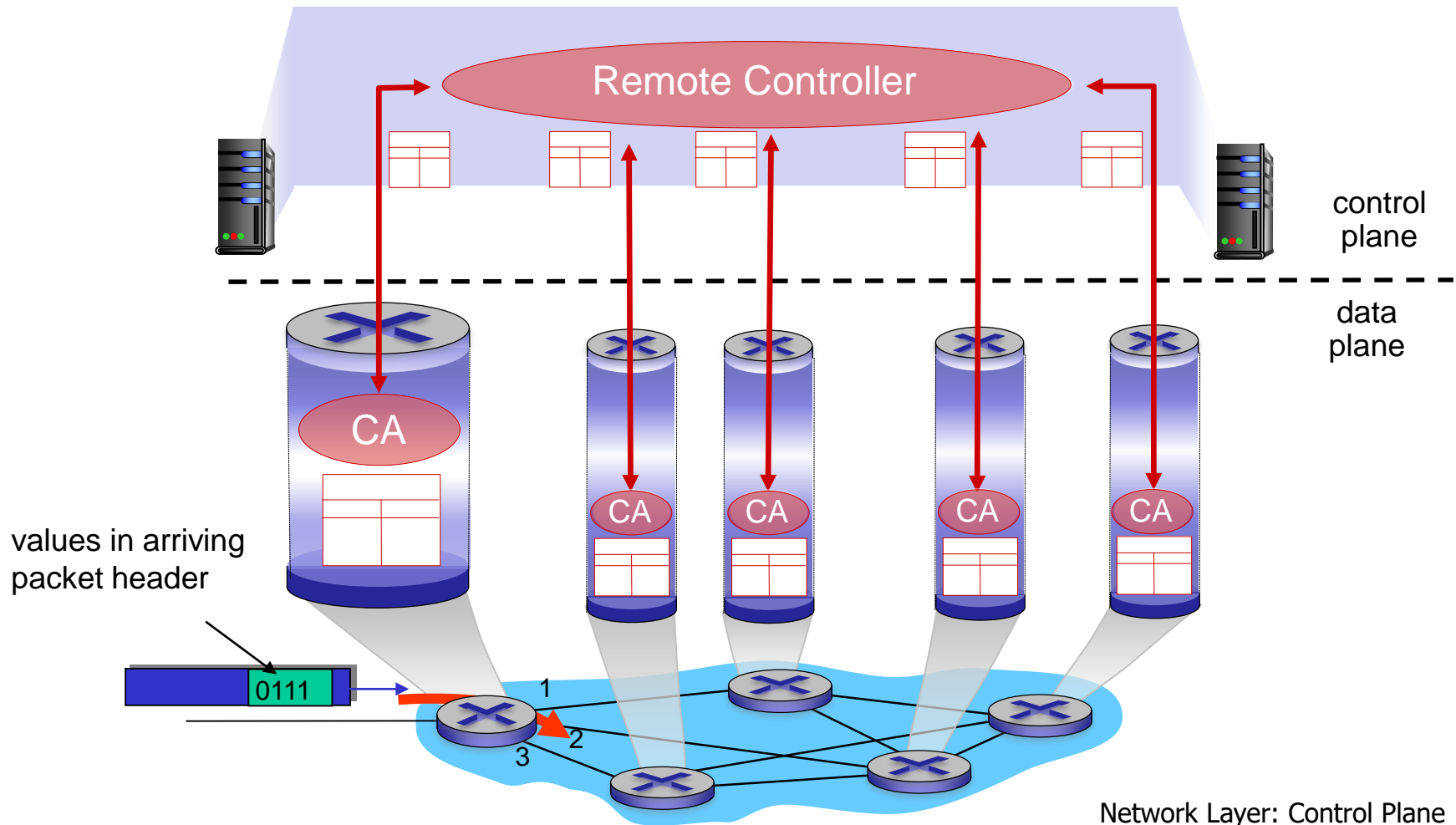
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



网络服务模型

Q: 什么是信道为发送者和接收者提供的服务模型(*service model*)?

网络服务模型定义网络的一端到另一端之间分组的端到端传输特性。

数据报的服务模型:

- ❑ 保证交付
- ❑ 在某个时间内的保证交付

数据报流的服务模型:

- ❑ 有序的数据报交付
- ❑ 数据流的最小带宽保证
- ❑ 确保最大时延抖动

网络层服务模型:

网络 体系结构	服务 模型	Guarantees ?				拥塞指示
		带宽	丢包	顺序	定时	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

网络层连接和无连接服务

- ❑ 数据报网络提供了网络层无连接的服务
- ❑ **VC** 网络提供了网络层面向连接的服务

- ❑ 网络层提供的类似传输层的连接服务：
不同点
 - **service**: 主机到主机
 - **no choice**: 网络提供, 没有其它选择
 - **implementation**: 在网络的核心实施

VC网络

“源到目的地路径的建立过程类似电话电路”

- 性能较好
 - 从源路由到目的路由、中间的相关路由协同工作
-
- 在传输数据前存在呼叫建立和拆除的过程
 - 每个数据报携带**VC**号(不是目的主机地址)
 - 从源到目的路由之间的每个路由都为每一个数据流维护状态
 - 链路,路由资源(带宽、缓冲区)分配给**VC**
 - (专属的资源= 可预见的服务)

VC 实施

VC 组成:

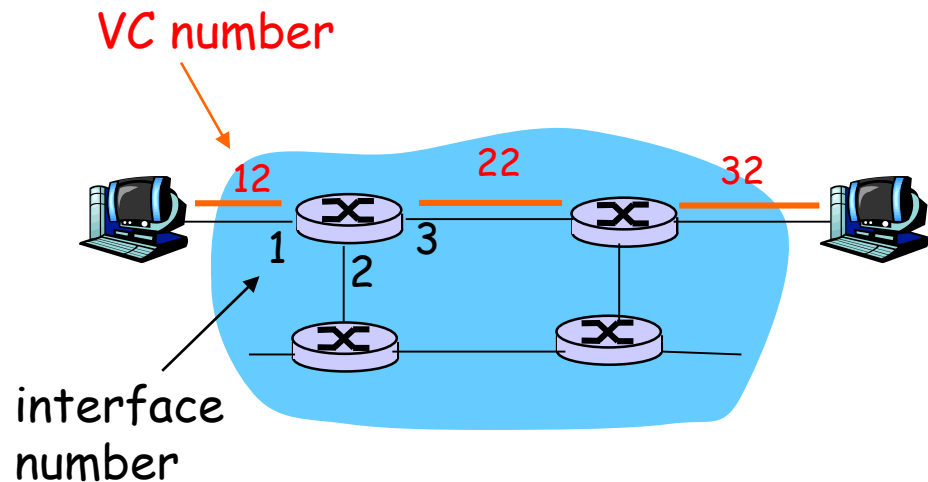
1. 源到目的主机的路径
 2. **VC**号, 路径上的每段链路一个号码
 3. 路由表中的前向转发表项
- 每个数据报都带有 **VC**号码
(不是目的主机地址!)
 - **VC** 号在每段链路上都有可能变化.
 - 从转发表获得新的**VC** 号码代替旧的**VC** 号码

VC前向转发表

Forwarding table in
northwest router:

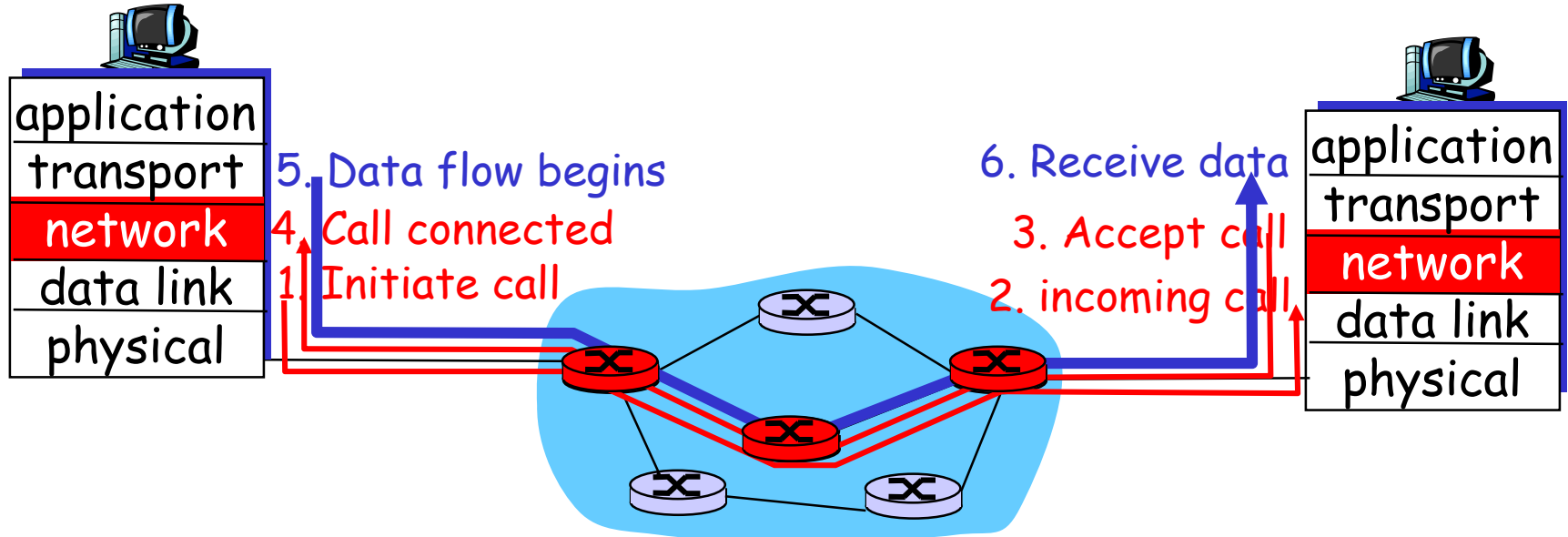
入接口	入VC #	出接口	出VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...

每个VC路由都维护连接状态信息！



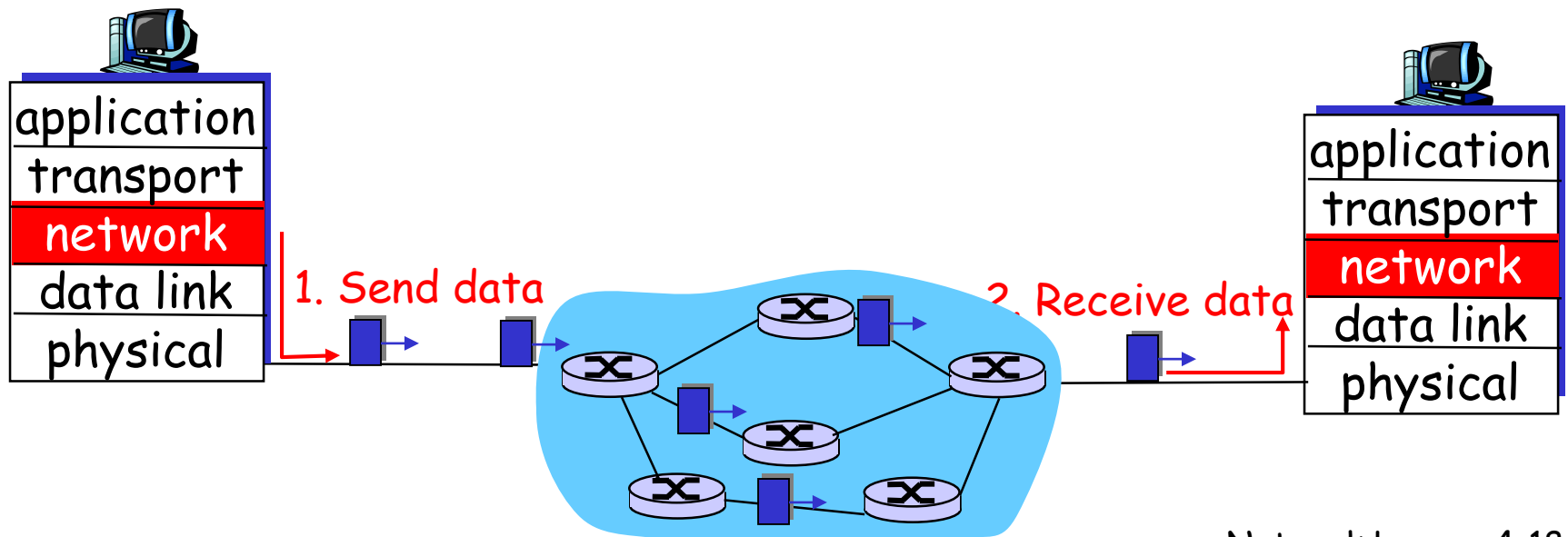
虚电路：信令协议

- 用于建立、维护、拆除 VC
- 在 **ATM**, **frame-relay**, **X.25** 网络中使用
- 没有在 **Internet** 使用了！

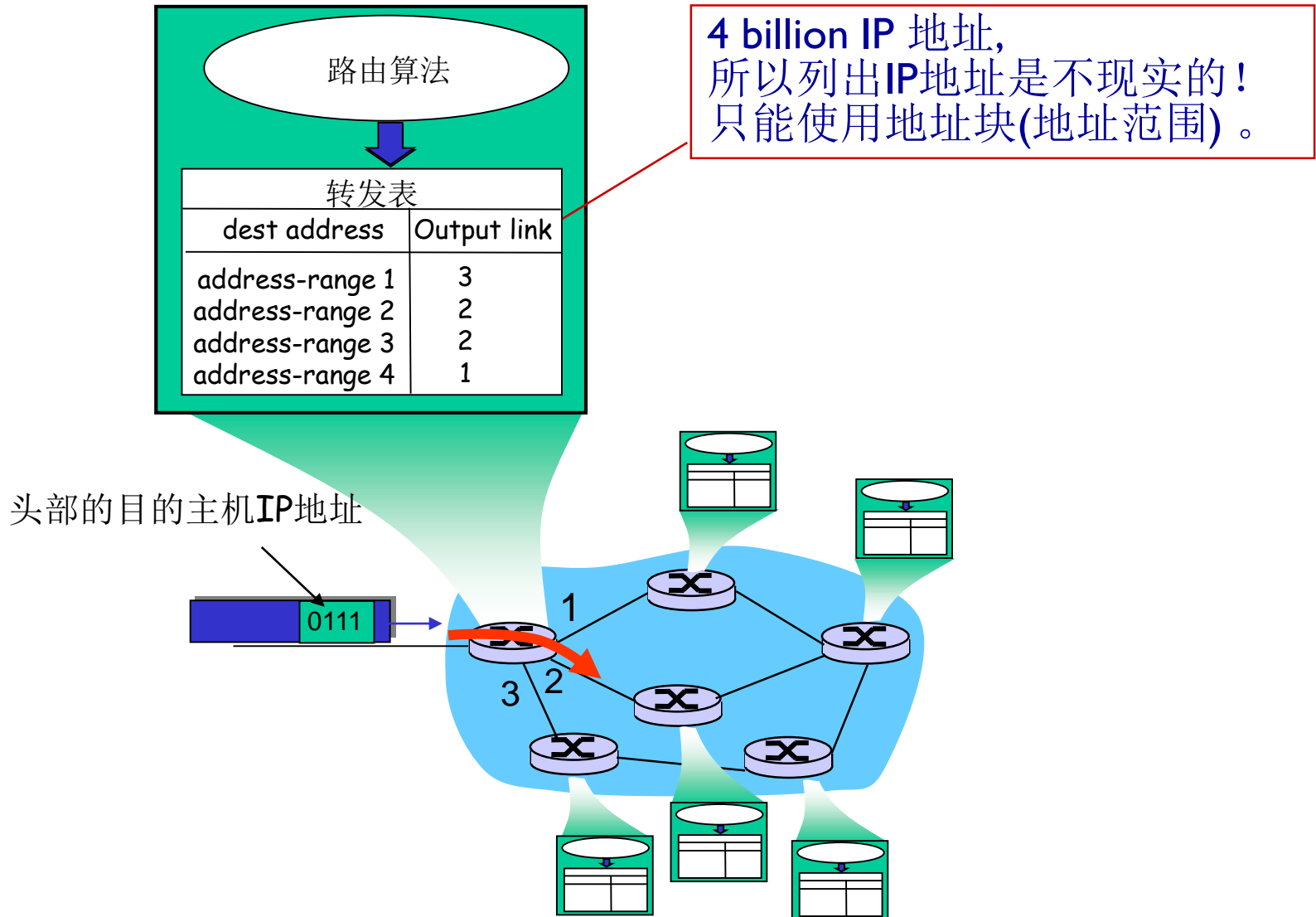


数据报网络

- ❑ 网络层没有呼叫建立的过程
- ❑ **routers**: 没有端到端连接的状态
 - 没有网络层概念的“连接”
- ❑ 数据报转发使用目的主机**IP**地址
 - 相同源-目的地址对的数据报可能沿着不同的路径向前转发



前向转发表



数据报网络 vs VC网络

Internet (datagram)

- ❑ 计算机之间的数据交换
 - “弹性”服务, 无严格时间要求
- ❑ “smart” 端系统
 - 能够自适应, 控制, 错误恢复
 - 网络内部简单,
网络边缘复杂
- ❑ 链路类型多样
 - 不同特征
 - 统一服务困难

ATM (VC)

- ❑ 从电话网络发展而来
- ❑ 满足人类交流需要:
 - 严格的时间, 可靠性要求
 - 保证服务
- ❑ “dumb” 端系统
 - 例如电话
 - 网络内部复杂

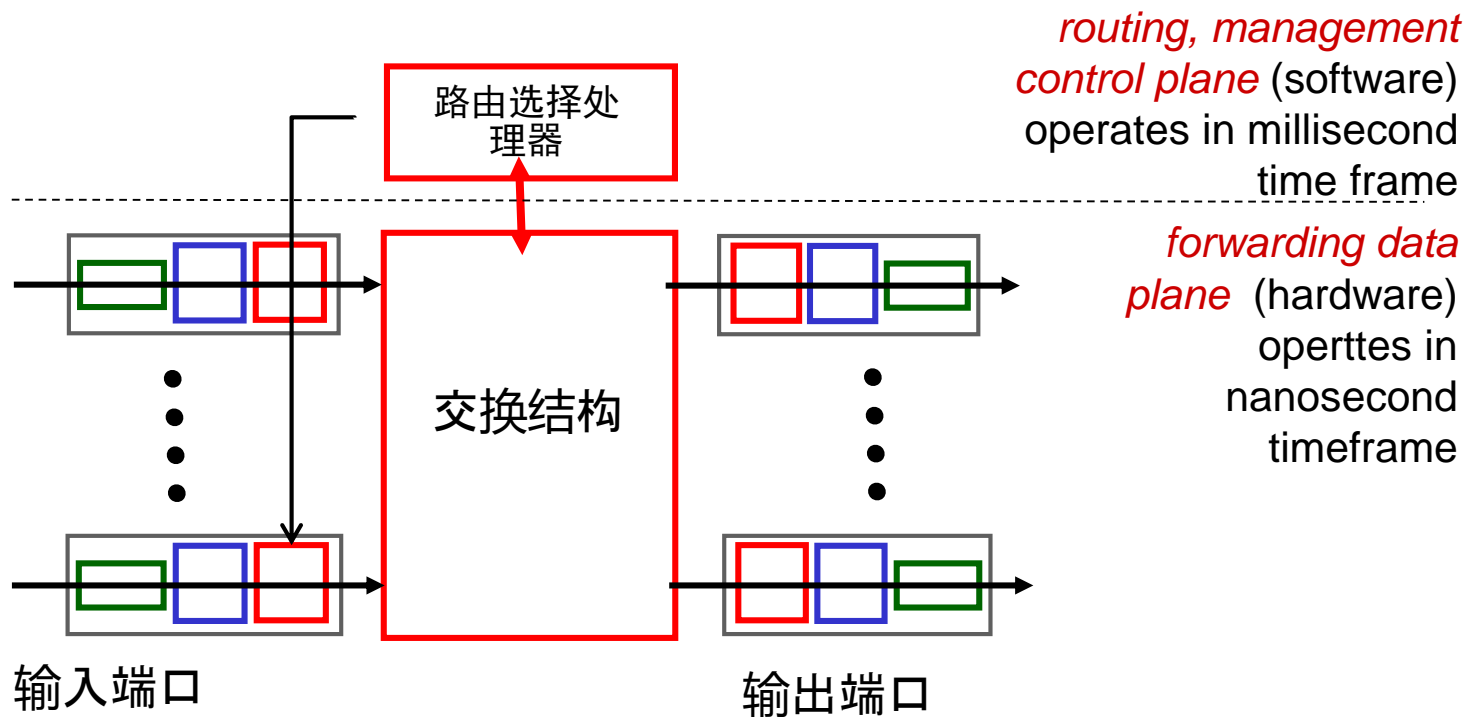
Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

路由结构概览

两个关键的路由功能:

- ❑ 运行路由和算法协议 (RIP, OSPF, BGP)
- ❑ 从入端口到出端口链路 前向转发 数据报



基于目的的转发表

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

最长前缀匹配

<u>Prefix Match</u>	<u>Link Interface</u>
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Examples

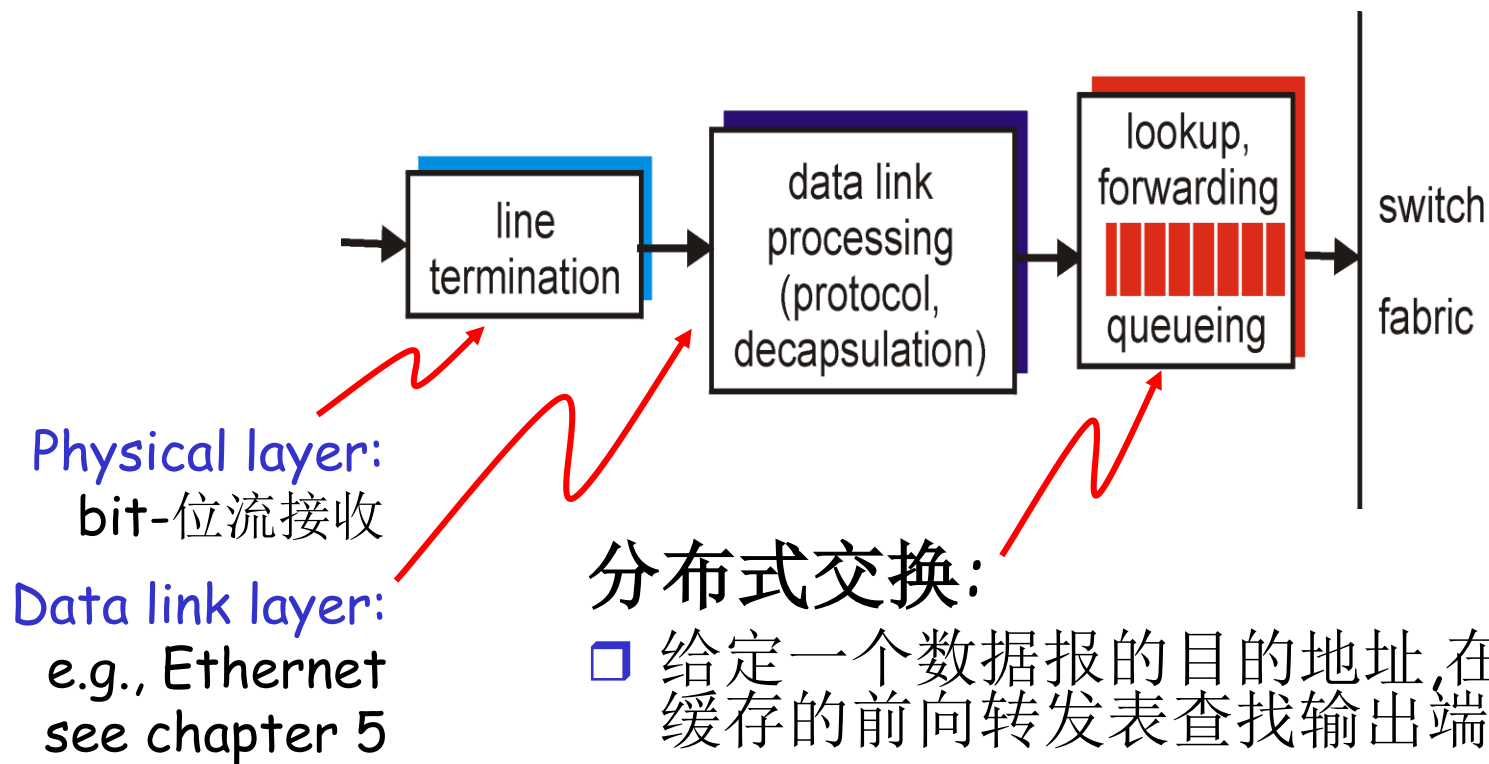
DA: 11001000 00010111 00010110 10100001

Which interface?

DA: 11001000 00010111 00011000 10101010

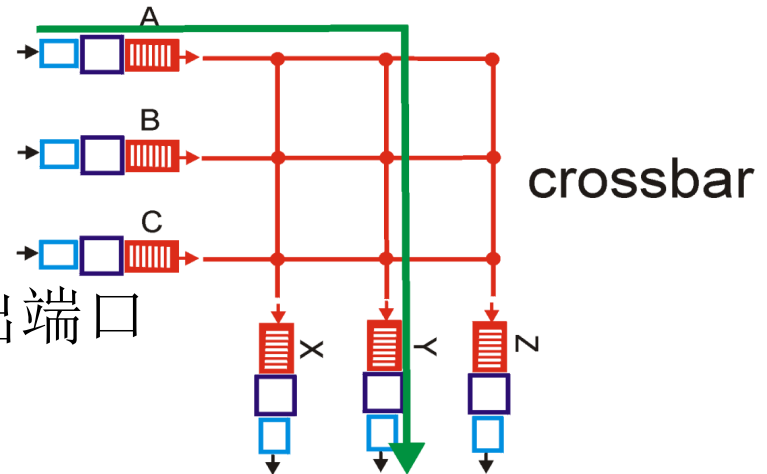
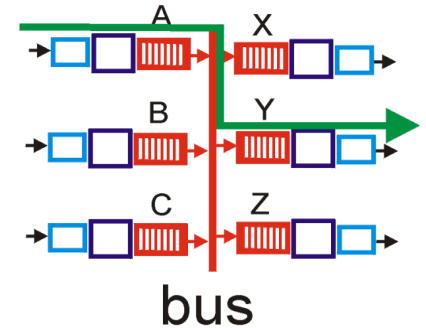
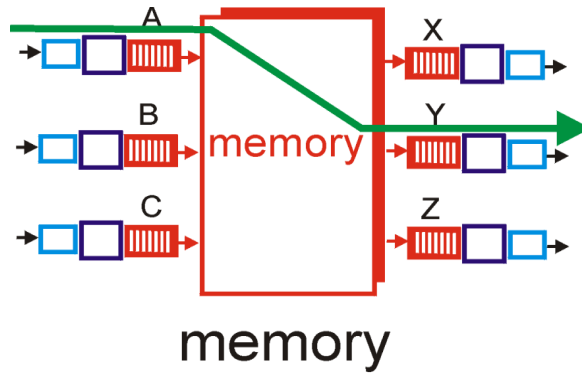
Which interface?

输入端口功能



- 给定一个数据报的目的地址,在输入端口缓存的前向转发表查找输出端口
- 目标:
输入端口的处理速度能够达到线路速度
- 排队:
如果数据到达速度大于前向转发速度

三种类型的交换结构

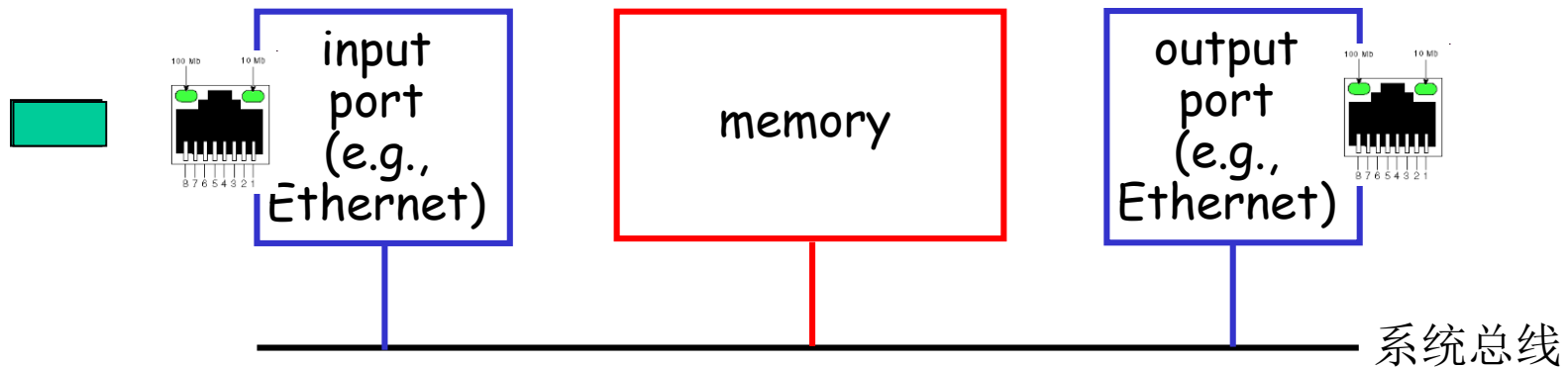


- 从输入端口传输数据至输出端口
- 交换速率：能够传的多快

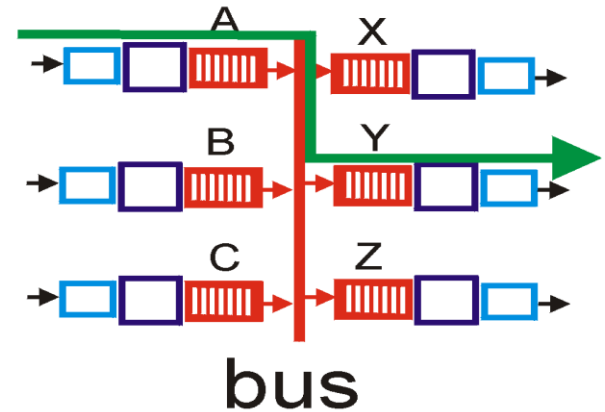
通过内存交换

第一代路由器:

- ❑ 普通的计算机，在 **CPU** 的控制下完成交换功能
- ❑ 数据报首先拷贝到系统的内存
- ❑ 速度受到内存带宽的限制
(每个数据报都两次经过总线)

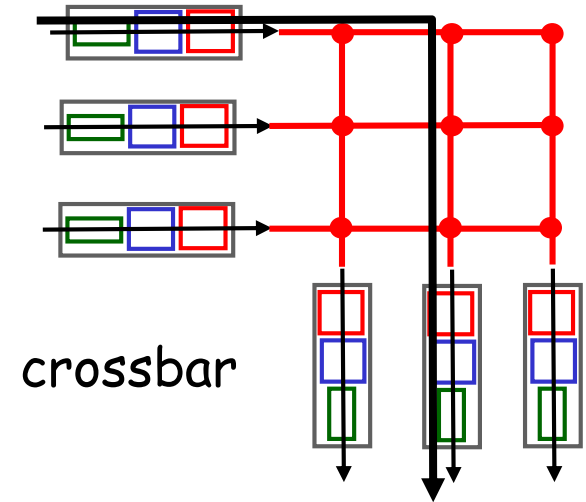


通过总线交换



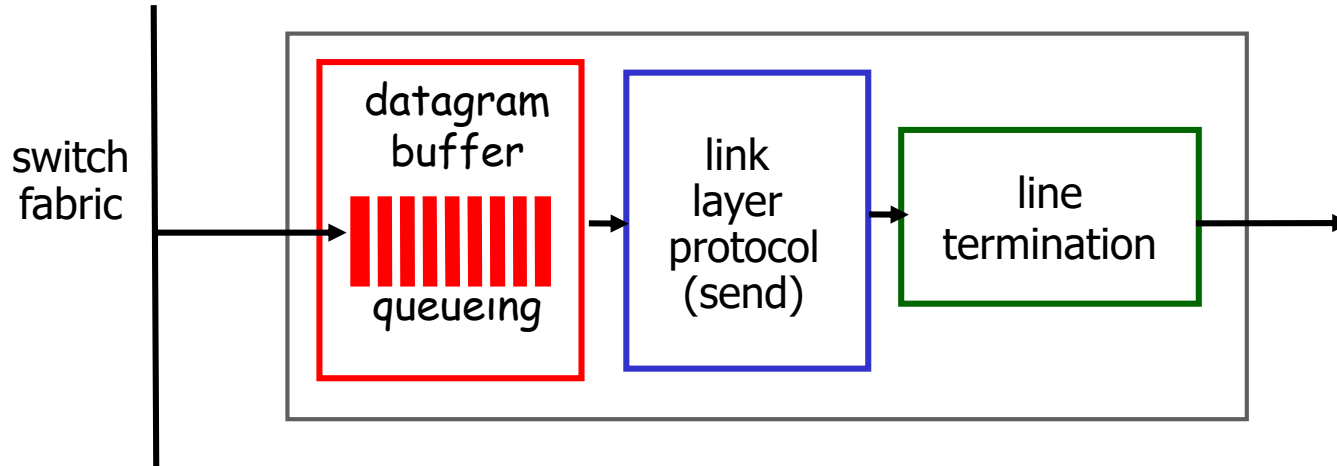
- ❑ 输入端口经总线将数据报传送到输出端口
- ❑ 总线竞争(bus contention):
交换速度受到总线速度的限制
- ❑ 32 Gbps 总线, Cisco 5600: 适用于一般的接入路由和企业路由

通过互联网络交换



- ❑ 突破了总线带宽的限制
- ❑ Banyan 网络,
以及其它曾用于多处理器系统的交换网络
- ❑ 发展方向: 将长度变化的**IP**分组分片成固定的长度的信元, 通过互联网络进行交换
- ❑ Cisco 12000: 交换速度达到 60 Gbps

输出端口

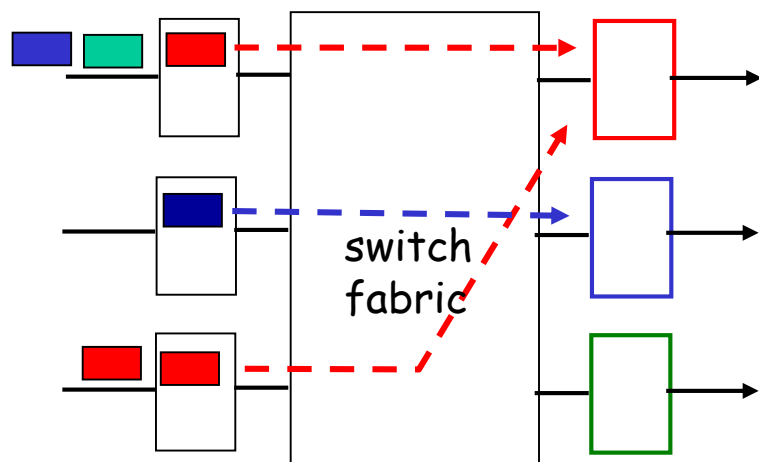


□ 缓存(*buffering*) :

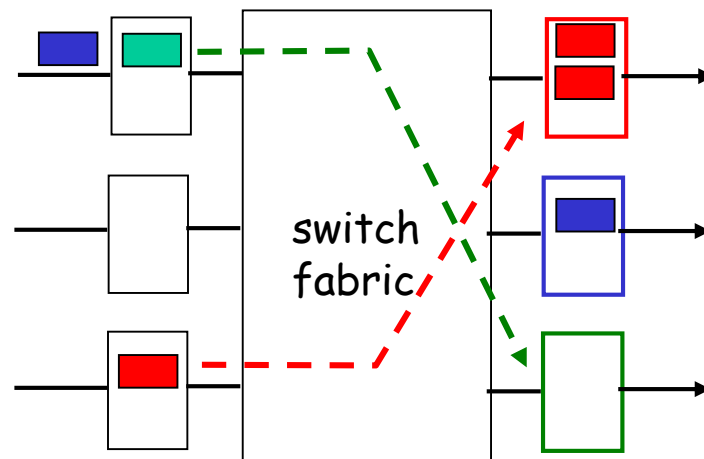
当数据报到达 数据报可能由于拥塞和缓存溢出丢失！

□ 调度策略(*scheduling*) : 优先级调度— 优先调度某些数据报！ 选择哪个数据报优先进行传送

输出端口排队



at t , packets move
from input to output



one packet time
later

- 缓冲：通过交换结构的数据报到达速率超过线路传输速率
- 排队(延时)和丢包 产生于缓冲区溢出

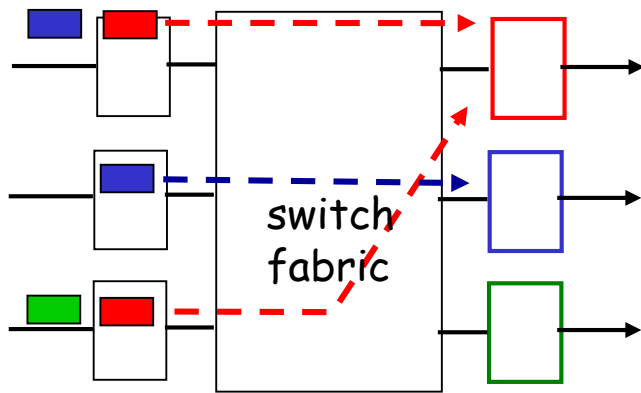
缓冲区大小的选择?

- ❑ RFC 3439 :
- ❑ 平均缓冲区应该是链路容量 C 的“典型” RTT (例如 250 msec) 倍
 - e.g., $C = 10$ Gps link: 2.5 Gbit buffer
- ❑ 研究表明 N 个数据流, 缓冲区大小应该是

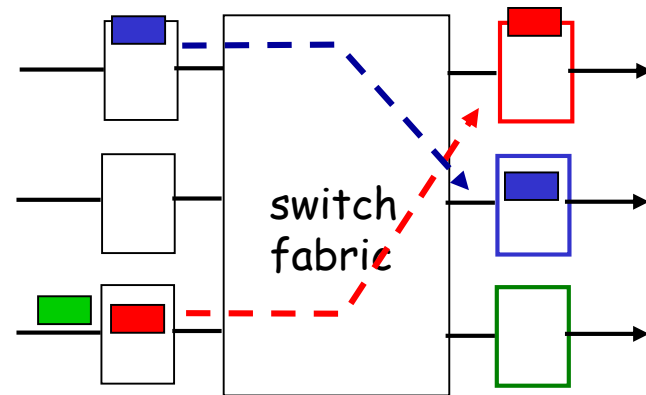
$$\frac{RTT \cdot C}{\sqrt{N}}$$

输入端口排队

- Fabric 处理速度小于输入端口的速度会导致输入端口排队
- Head-of-the-Line (HOL) blocking:
- 线路前部阻塞：一个输入队列中排队的数据报必须等待通过交换结构发送，由于排在前面的数据报将它阻塞了。



output port contention:
only one red datagram can be
transferred.
lower red packet is blocked



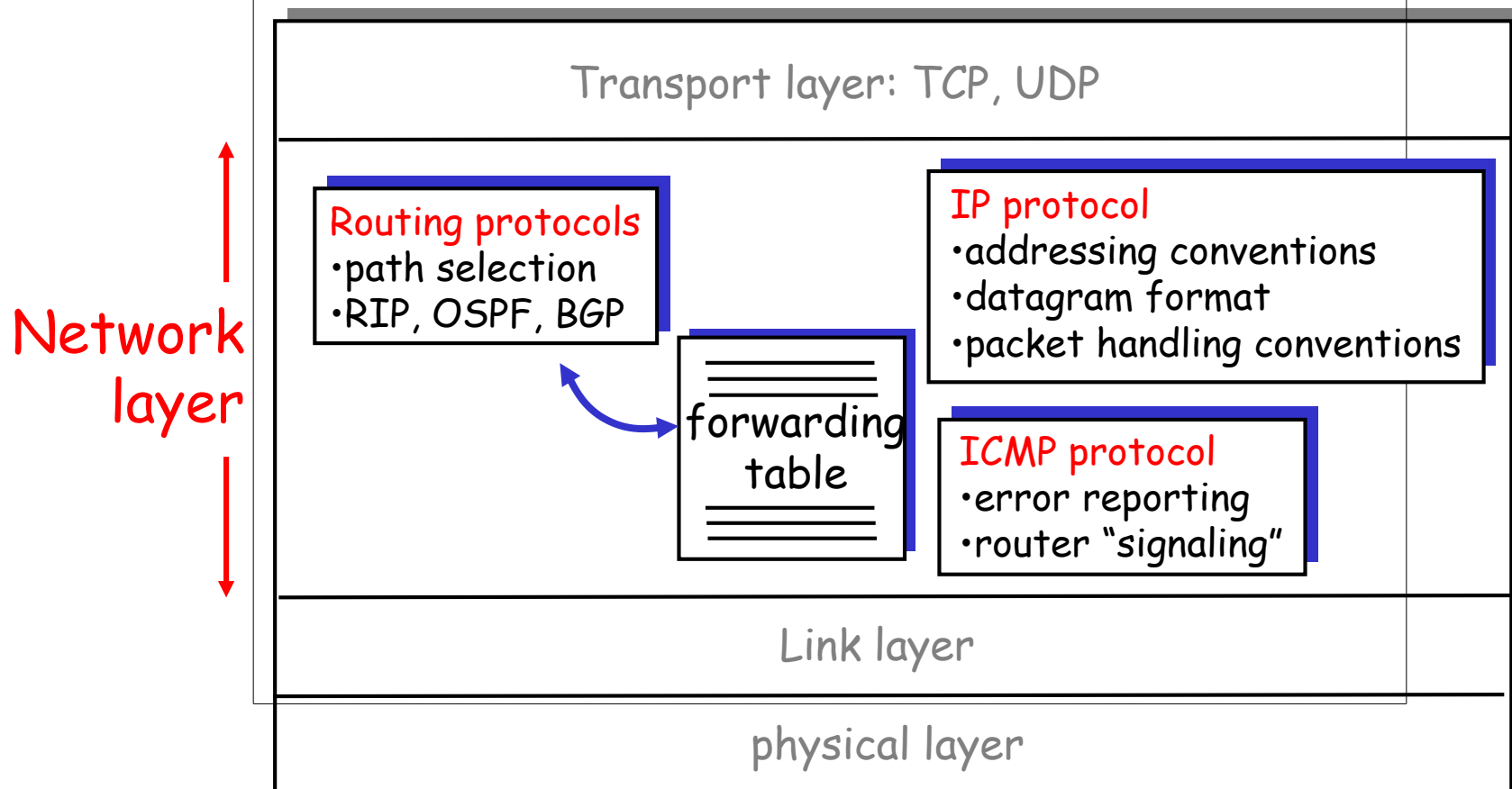
one packet time later:
green packet
experiences HOL
blocking

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

Internet 网络层

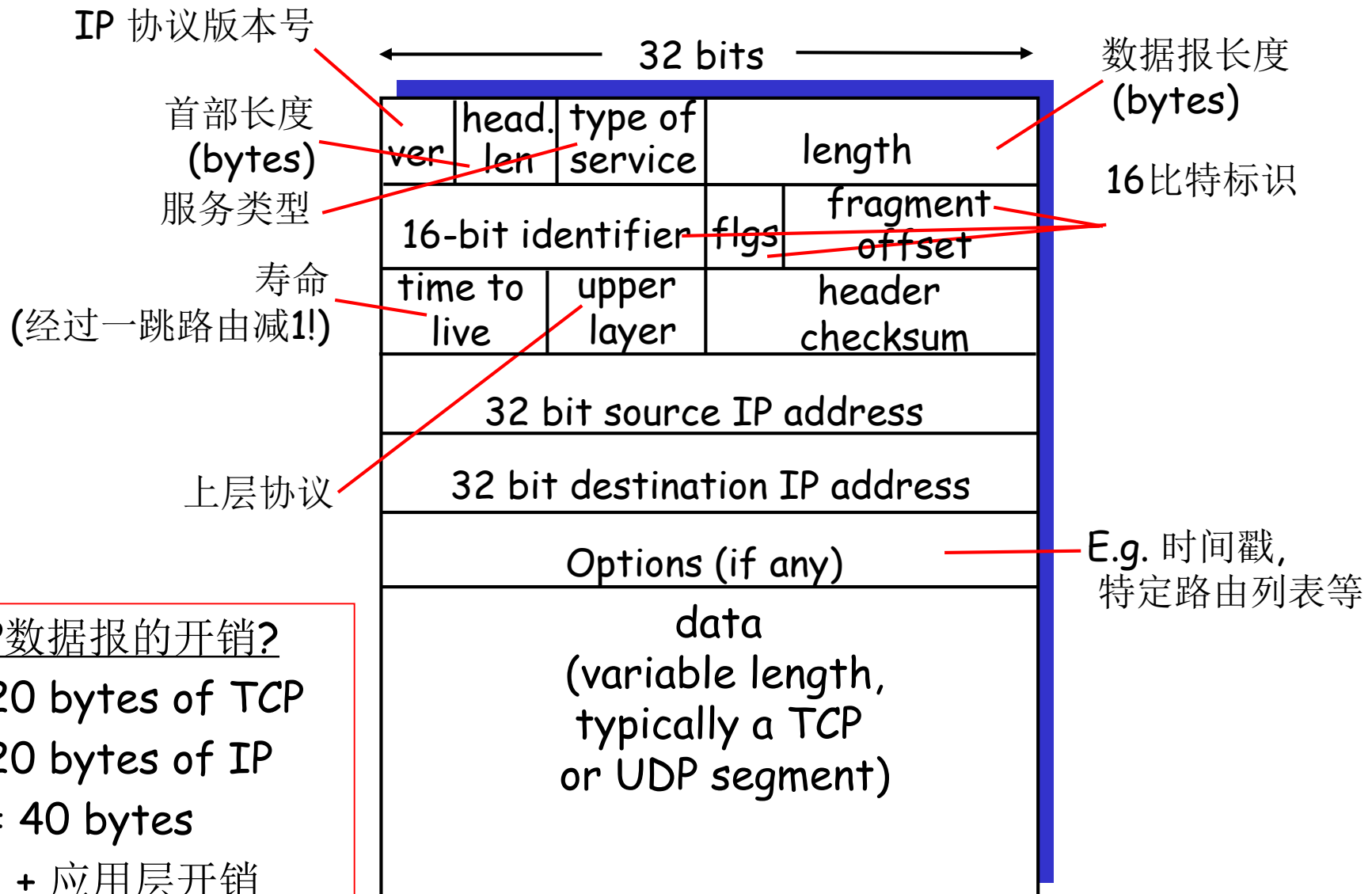
主机路由网络层功能:



Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

IP 数据报格式



TCP数据报的开销?

- ❑ 20 bytes of TCP
- ❑ 20 bytes of IP
- ❑ = 40 bytes
- + 应用层开销

IP 分片 & 重组

❑ 最大传输单元

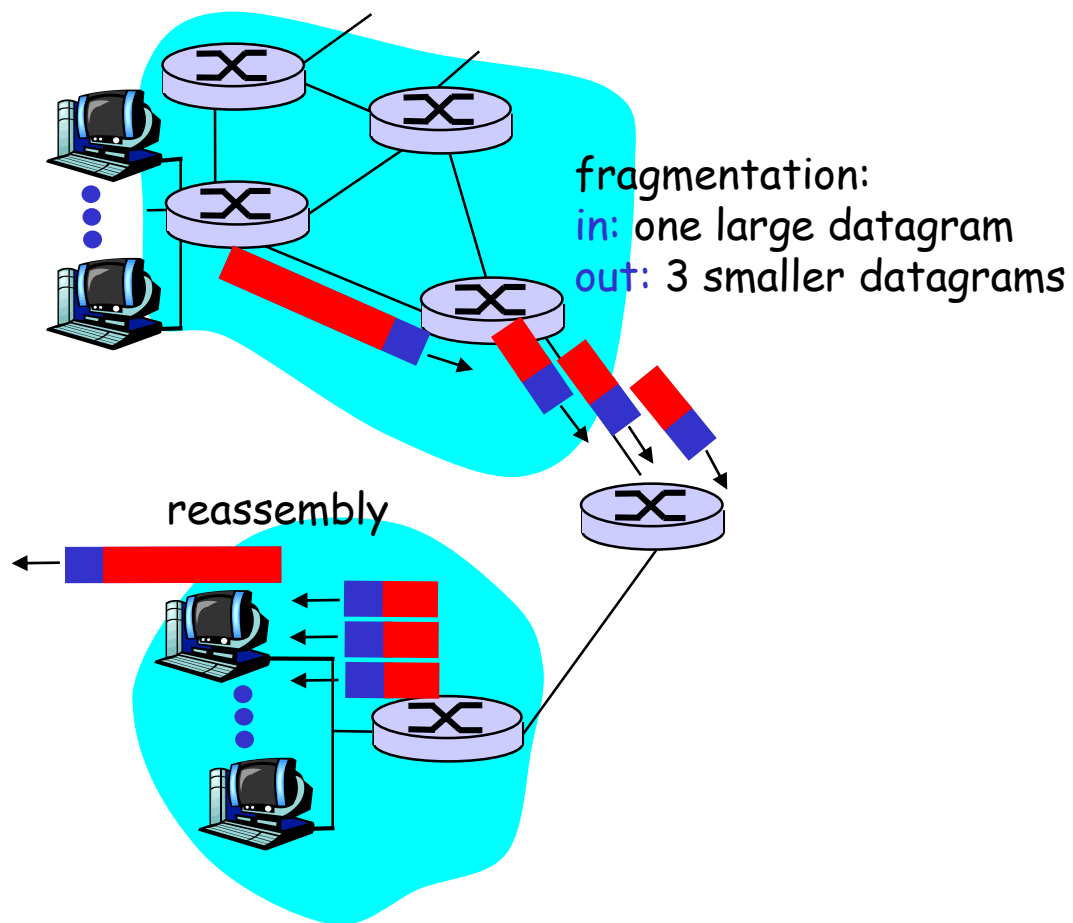
MTU (max. transfer size)

链路层帧能承载的最大数据量.

- 不同链路类型, 不同 MTUs

❑ 大的 IP 数据报分片成两个或多个较小的数据报 ("fragmented")

- 一个数据报变成多个数据报
- 在目的主机重组
- IP 头部字段用于标识和重装



IP分片 & 重组

Example

- ❑ 4000 byte datagram
- ❑ MTU = 1500 bytes

1480 bytes in data field

offset = $1480/8$

	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	=185

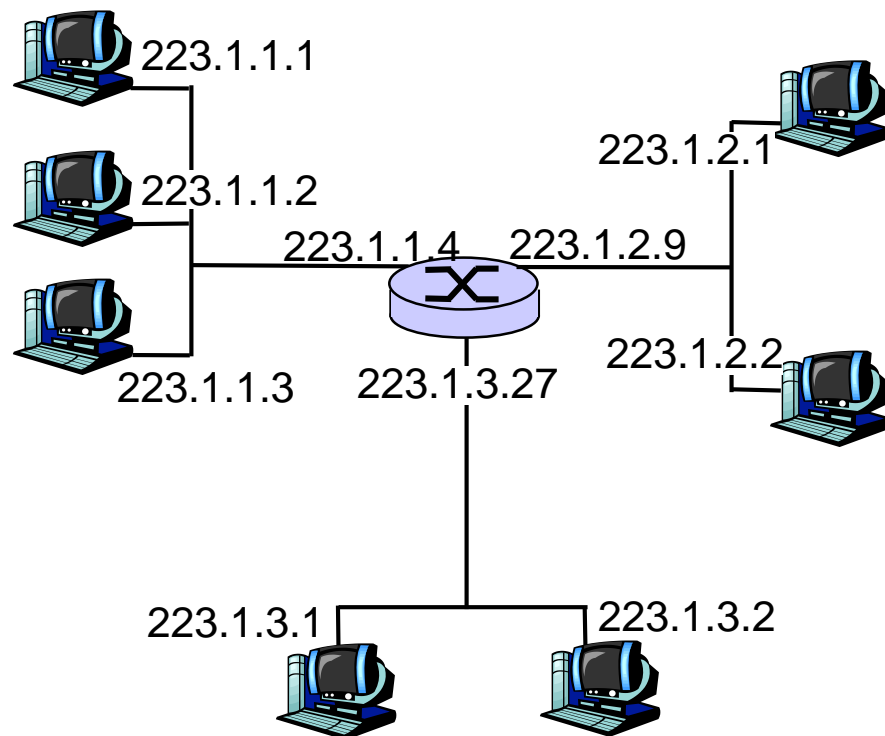
	length	ID	fragflag	offset
	=1040	=x	=0	=370

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

IP 编址: 介绍

- ❑ IP 地址: 32-bit 主机、路由器接口的标识
- ❑ 接口(interface): host/router和物理链路的连接
 - Router
 - 通常具有多个接口
 - host
 - 通常具有一个接口
 - IP 地址和每个接口关联



$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

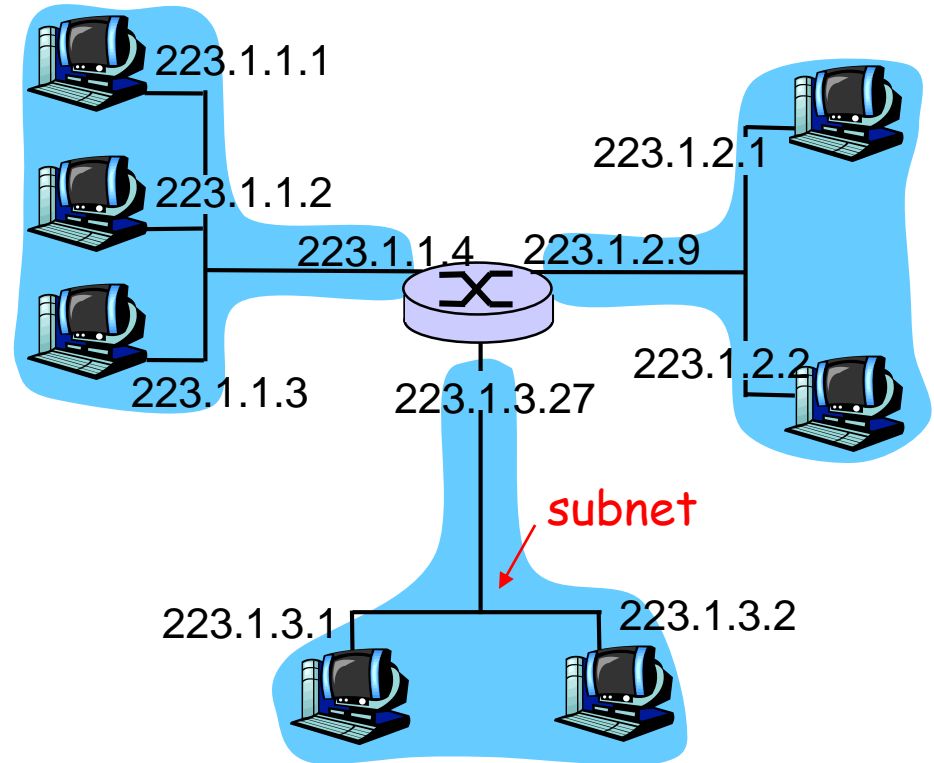
子网

□ IP 地址:

- 子网部分
(high order bits)
- 主机部分
(low order bits)

□ *What's a subnet ?*

- 具有相同的子网地址
- 物理上直接相连

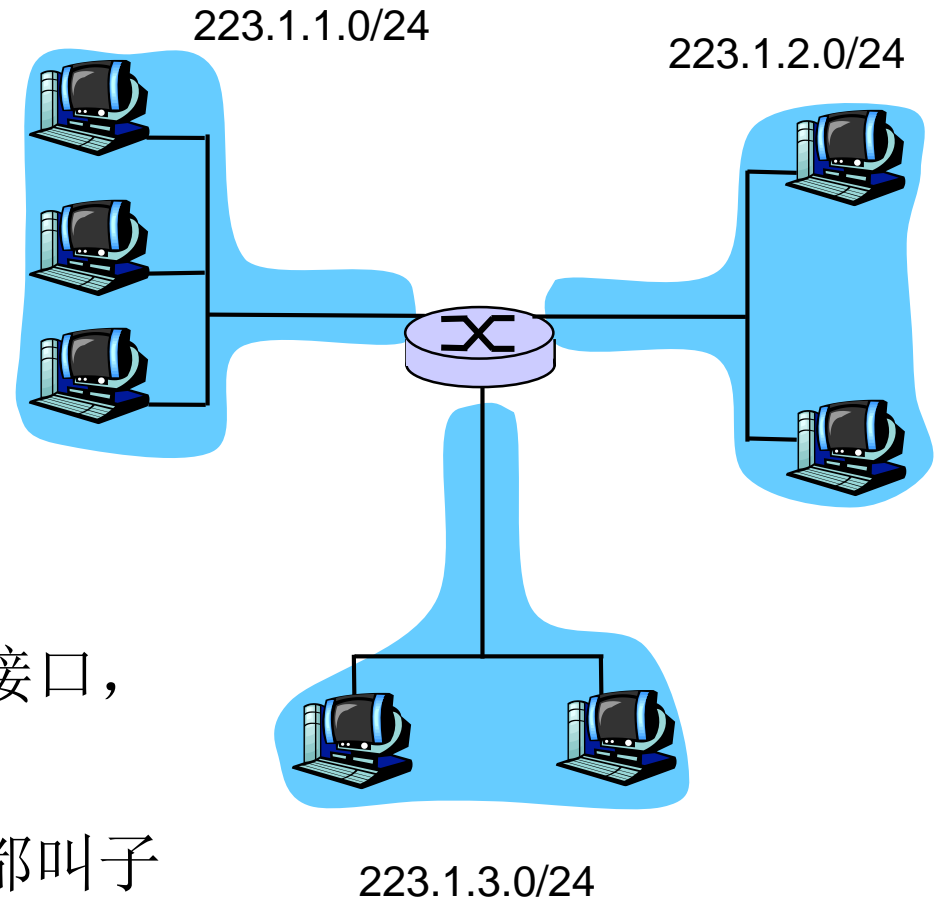


network consisting of 3 subnets

子网

Recipe

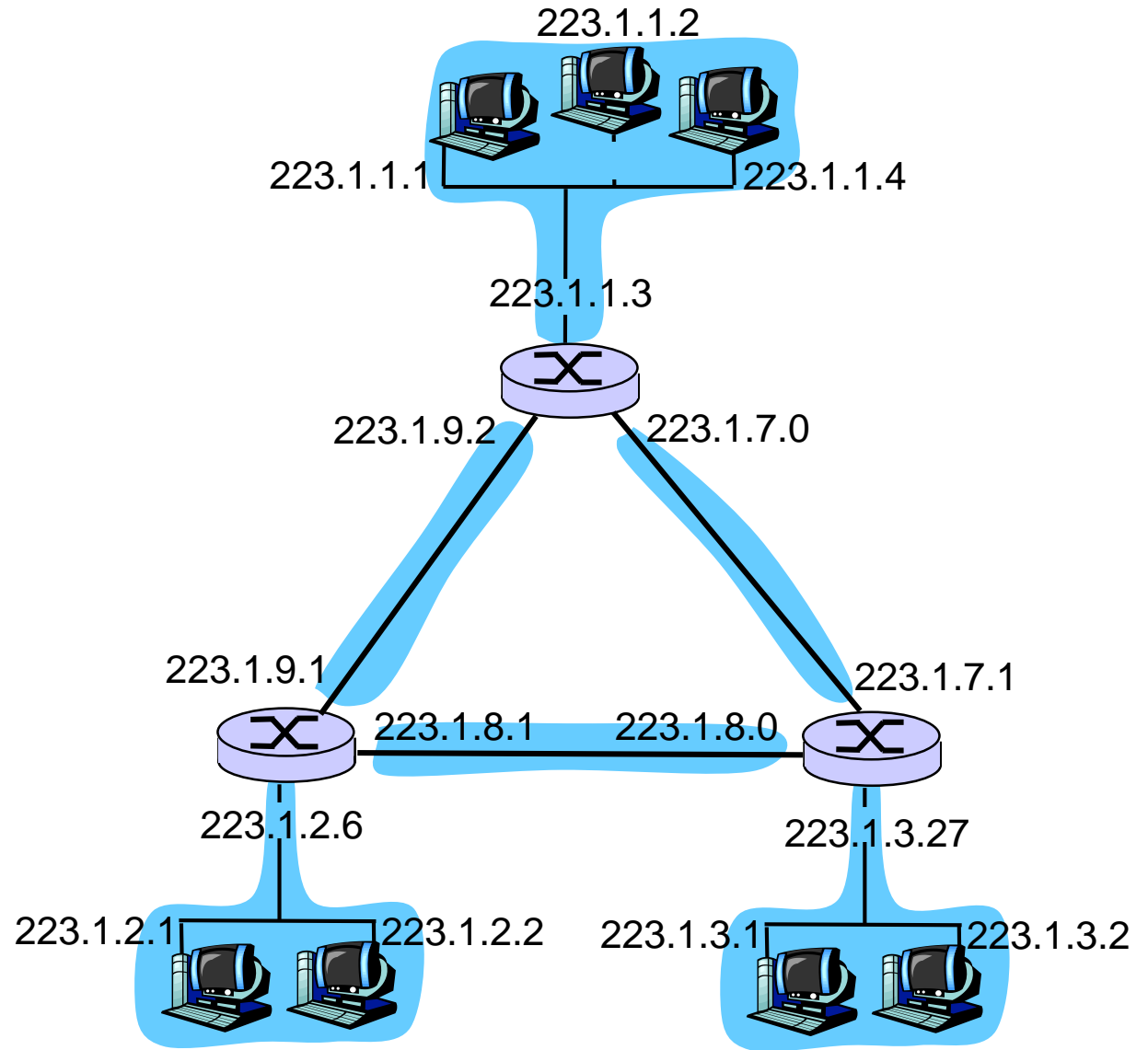
- ❑ 为了确定子网，
- ❑ 分开主机和路由器的每个接口，
- ❑ 从而产生几个分离的网络，
- ❑ 这些独立的网络中每一个都叫子网 (subnet).



Subnet mask: /24

子网

How many?

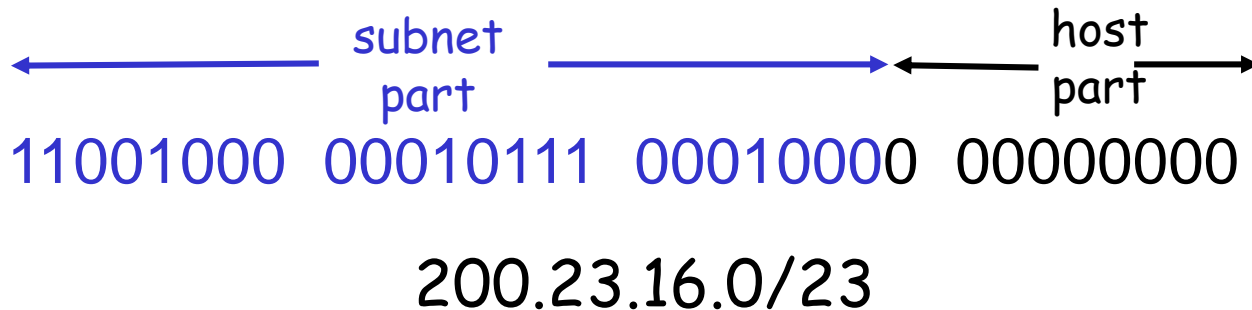


IP 地址: CIDR

CIDR: Classless InterDomain Routing

无类别域间选路

- IP地址子网部分具有一定的长度
- 地址格式: **a.b.c.d/x**, 其中**x**指示了在地址的第一部分中的比特数



IP 地址: 如何获取?

Q: 主机如何获得**IP**地址?

□ 网络管理员分配

- Windows:

- control-panel->network->configuration->tcp/ip->properties

- UNIX:

- /etc/rc.config

□ 动态主机配置协议**DHCP**(**D**ynamic **H**ost **C**onfiguration **P**rotocol):

从服务器动态获得**IP**地址

- “plug-and-play”

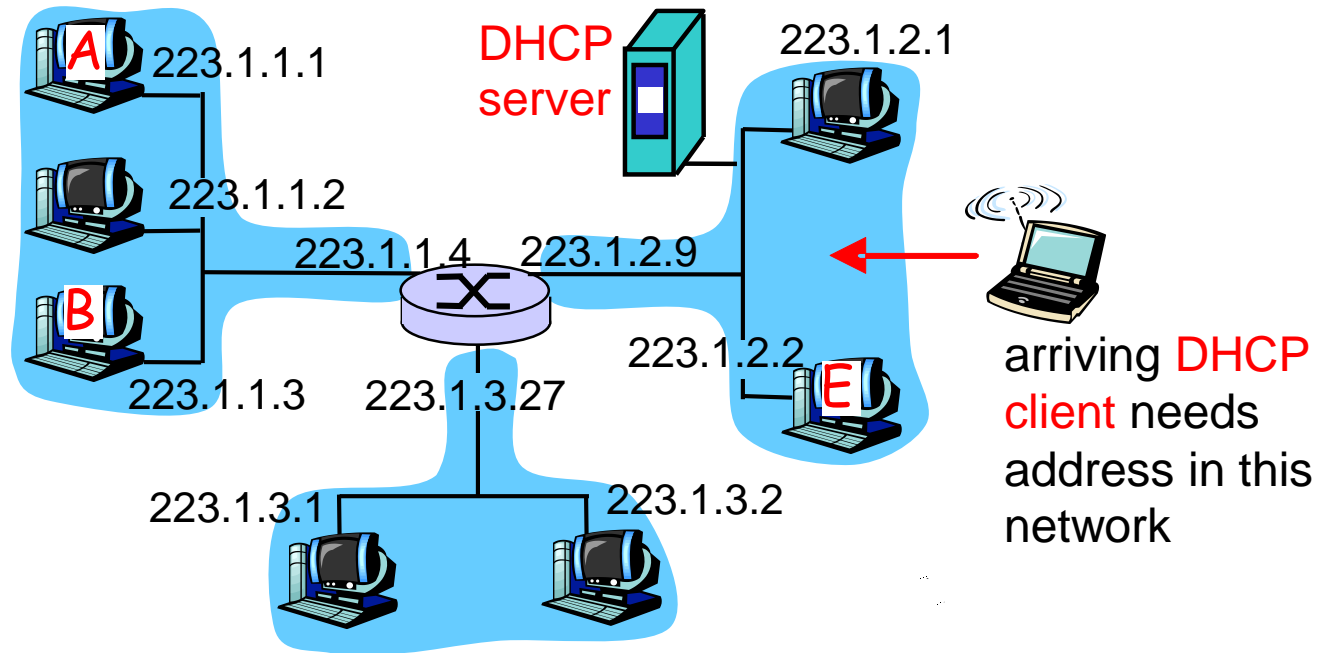
DHCP:动态主机配置协议

Goal: 当主机加入一个网络的时候，自动的获得一个地址。
可以续租；
可以重用地址(配置成每次都获得相同的地址)。

DHCP overview:

- host 广播 “DHCP discover” msg
- DHCP 响应 “DHCP offer” msg
- host 请求: “DHCP request” msg 请求IP地址
- DHCP 发送: “DHCP ack” msg 包含IP地址

DHCP 场景



DHCP 场景

DHCP server: 223.1.2.5

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 0.0.0.0
transaction ID: 654

arriving
client



DHCP offer

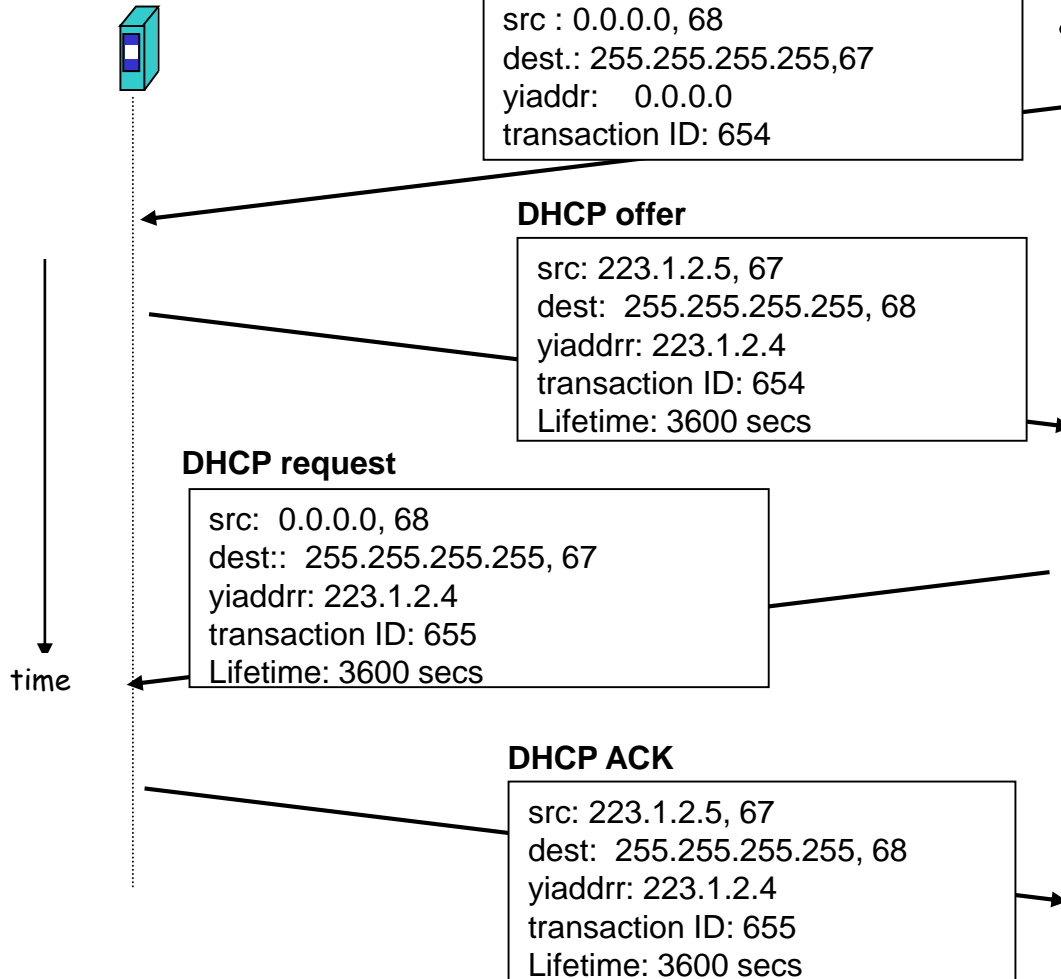
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs



DHCP: 同时获得其它配置

DHCP 除了获得所在子网的一个**IP**地址:

- 还有网关地址: 进入网络的第一跳路由;
- 还有**DNS** 服务器**IP**地址;
- 还有网络掩码(用于指示网络地址)!

IP 地址: 如何获取?

Q: 网络如何获得相应的地址块?

A: 在 **ISP's** 地址空间中申请

ISP's block	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/20
Organization 0	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/23
Organization 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000	200.23.18.0/23
Organization 2	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000	200.23.20.0/23
...
Organization 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000	200.23.30.0/23

IP 寻址：如何获取？

Q: ISP 如何获得地址空间？

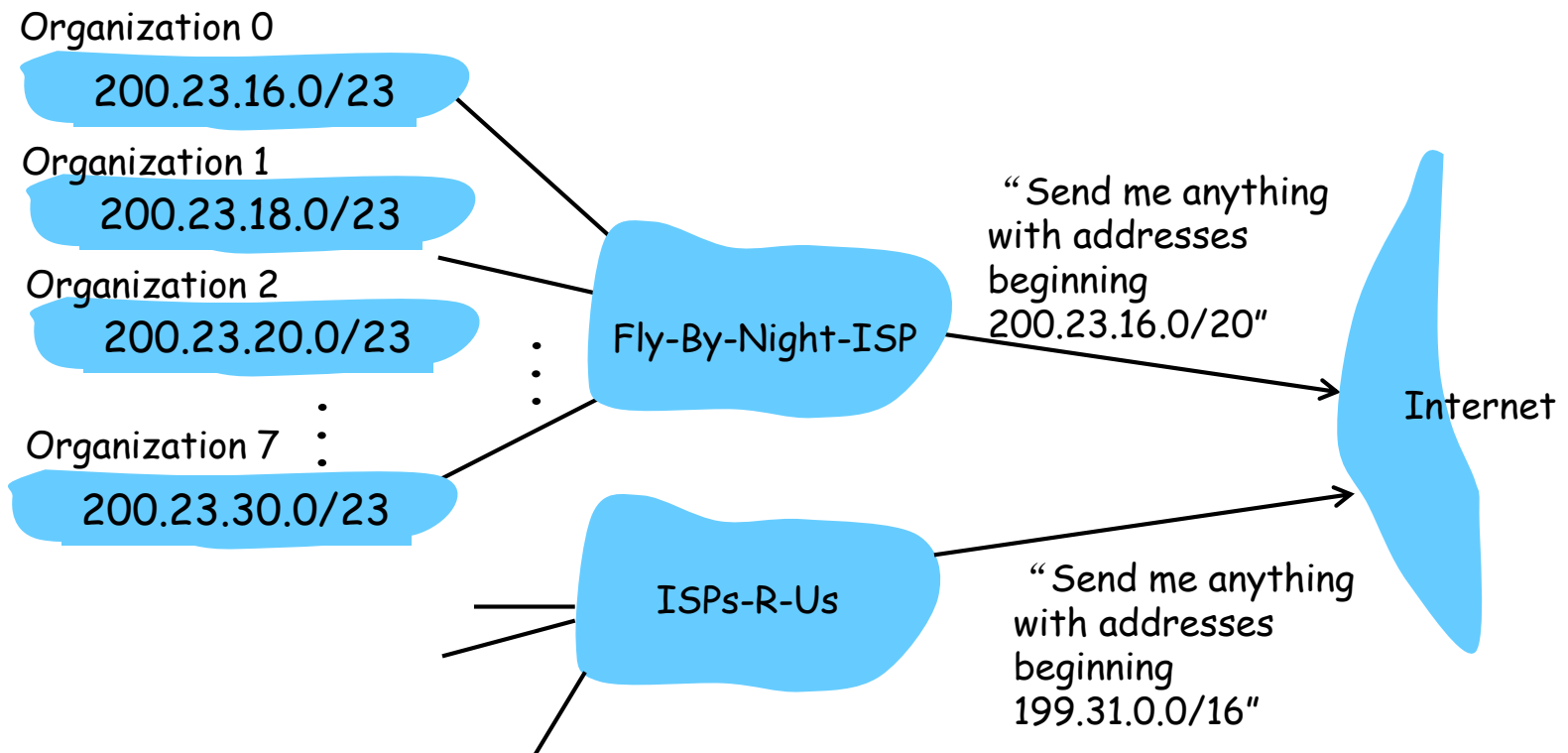
A: ICANN:

Internet Corporation for Assigned Names and Numbers

- 分配地址
- 管理DNS
- 解决纠纷

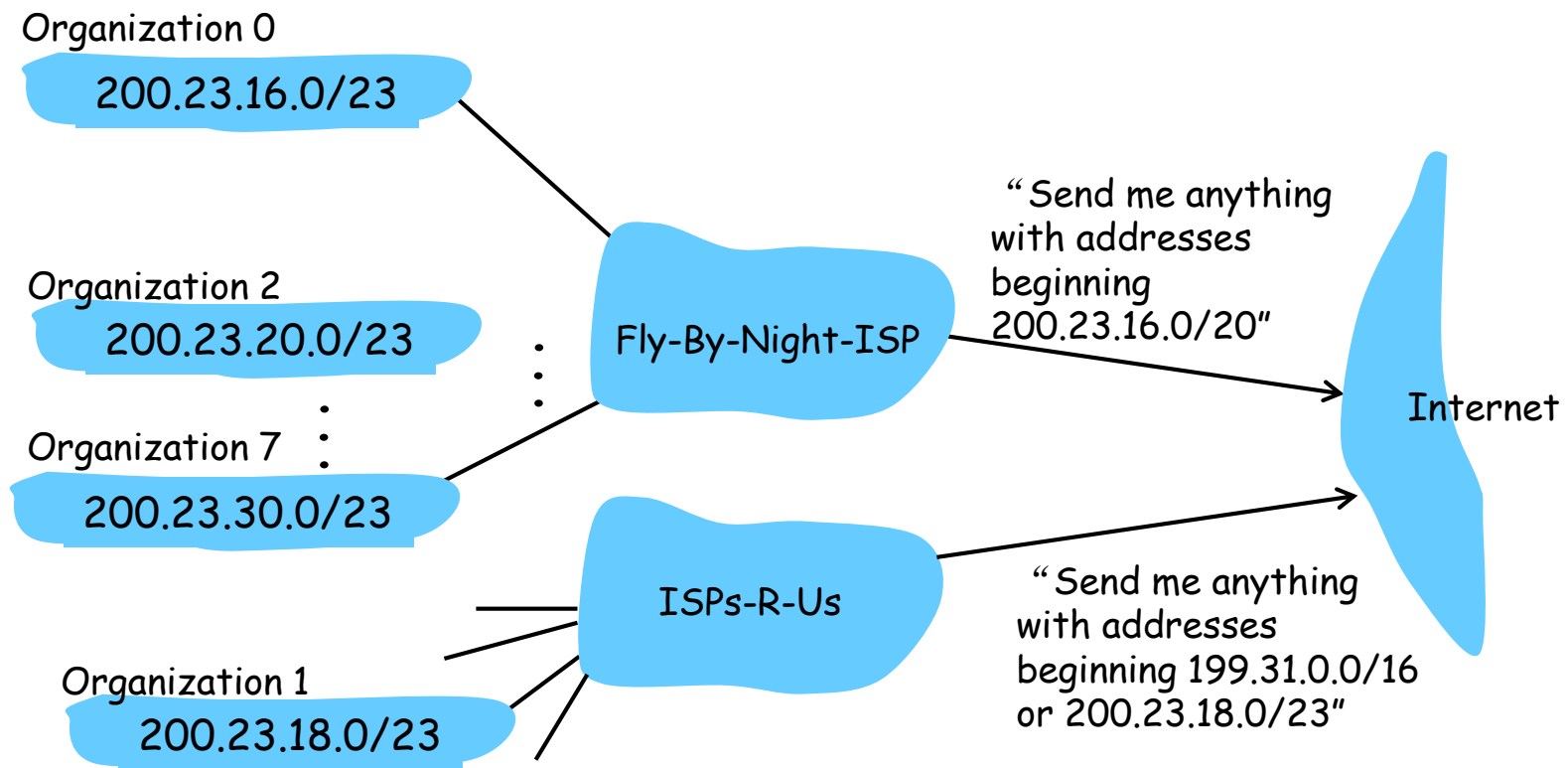
层次寻址：路由汇聚

层次寻址允许有效地地址广播：

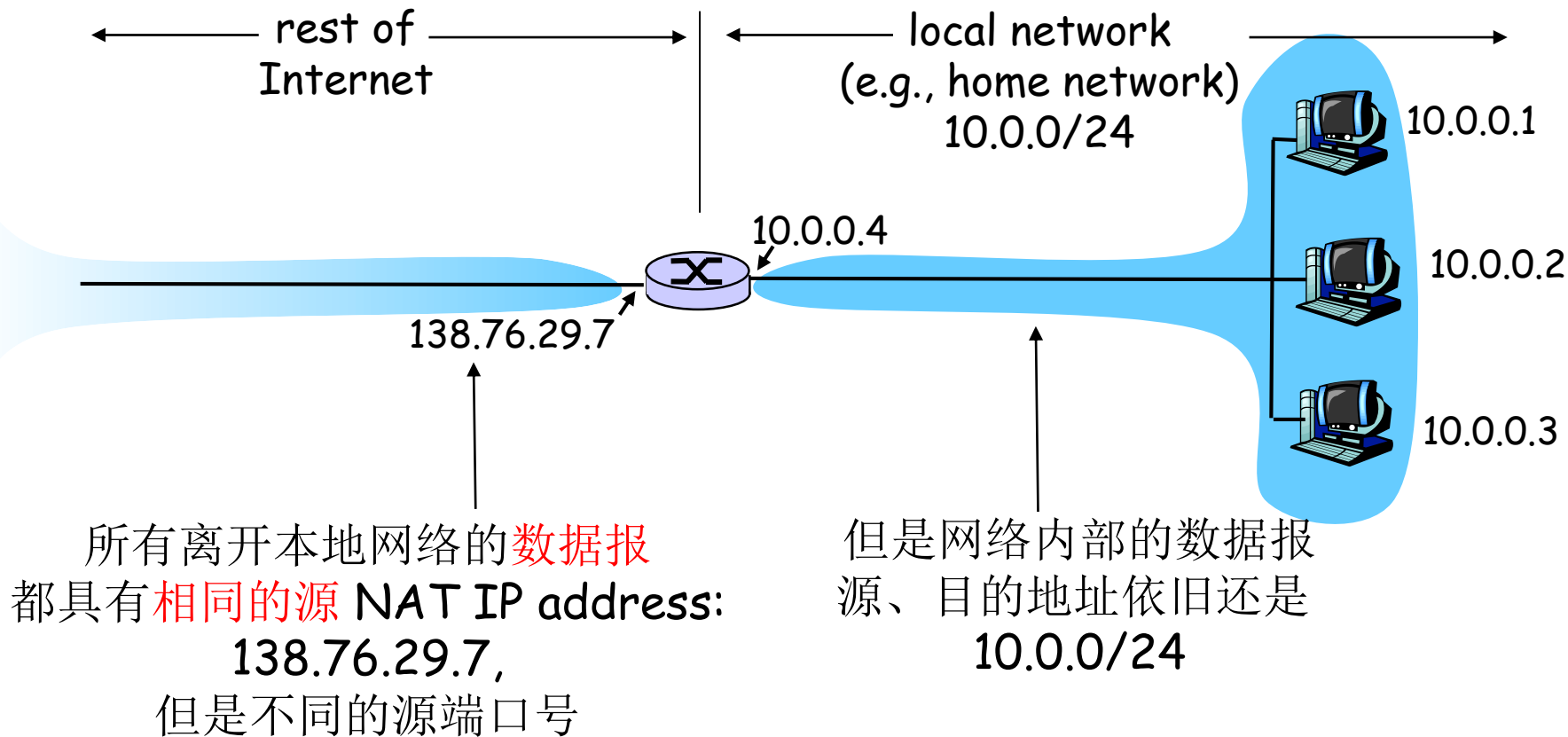


层次寻址：路由汇聚

ISPs-R-Us 对 Organization 1 做的一个路由汇聚



NAT网络地址转换: (Network Address Translation)



NAT:网络地址转换

- **Motivation:** 内部网络只使用一个外部**IP**地址:
 - **ISP**只需要一个地址:
 - 所有内部网络的设备共享一个 **IP**地址;
 - 可以在机构内部更改**IP**地址
 - 而不需要告诉外部相关部门;
 - 可以更换 **ISP**
 - 而修改内部网络设备的**IP**地址 ;
 - 内部网络设备无法直接寻址而更加安全
 - 对于外面的计算机来说不可见.

NAT:网络地址转换

Implementation: NAT router must:

- 对每个外出的数据报:

replace (源 IP 地址, 端口 #) 为 (NAT IP地址, 新端口 #)

... 远处的主机和服务器将使用(NAT IP地址, 新端口 #) 作为目的地地址进行相应.

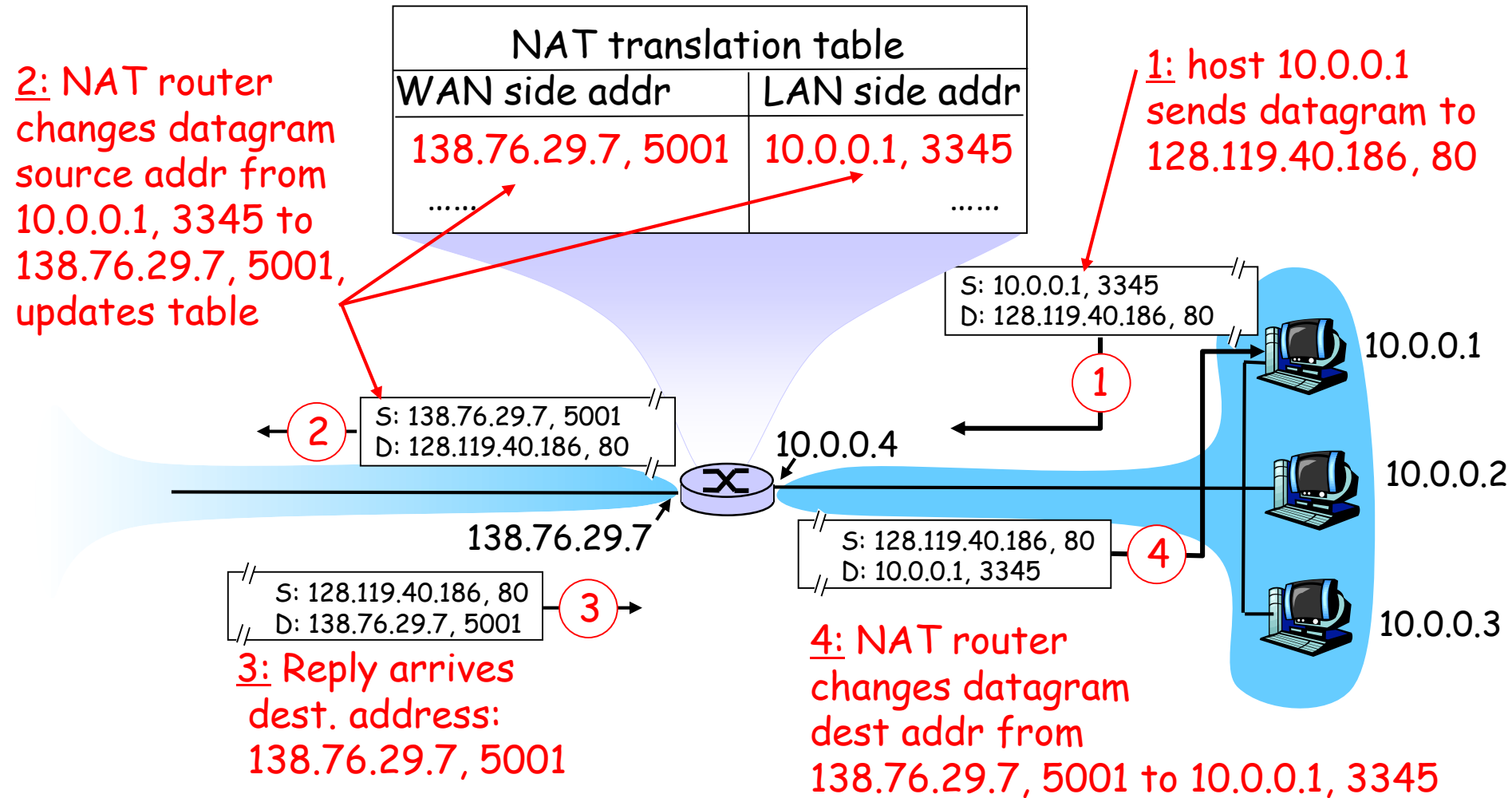
- 记录(*NAT* 转换表)

(源 IP 地址, 端口 #) (NAT IP地址, 新端口 #)

- 对每个进来的数据报:

replace (NAT IP地址, 新端口 #) (源 IP 地址, 端口 #) 存储在 *NAT* 表中

NAT:网络地址转换



NAT:网络地址转换

□ 16-bit 端口字段:

- 高达60,000 多个连接共享路由器广域网侧**IP**地址!

□ NAT 的不足:

- 路由器访问数据报 **layer 3** 头部
- 违反了**end-to-end** 原则
 - NAT 在应用层程序设计时应该加以考虑, *eg*, P2P applications
- 地址的短缺应该由**IPv6**来解决

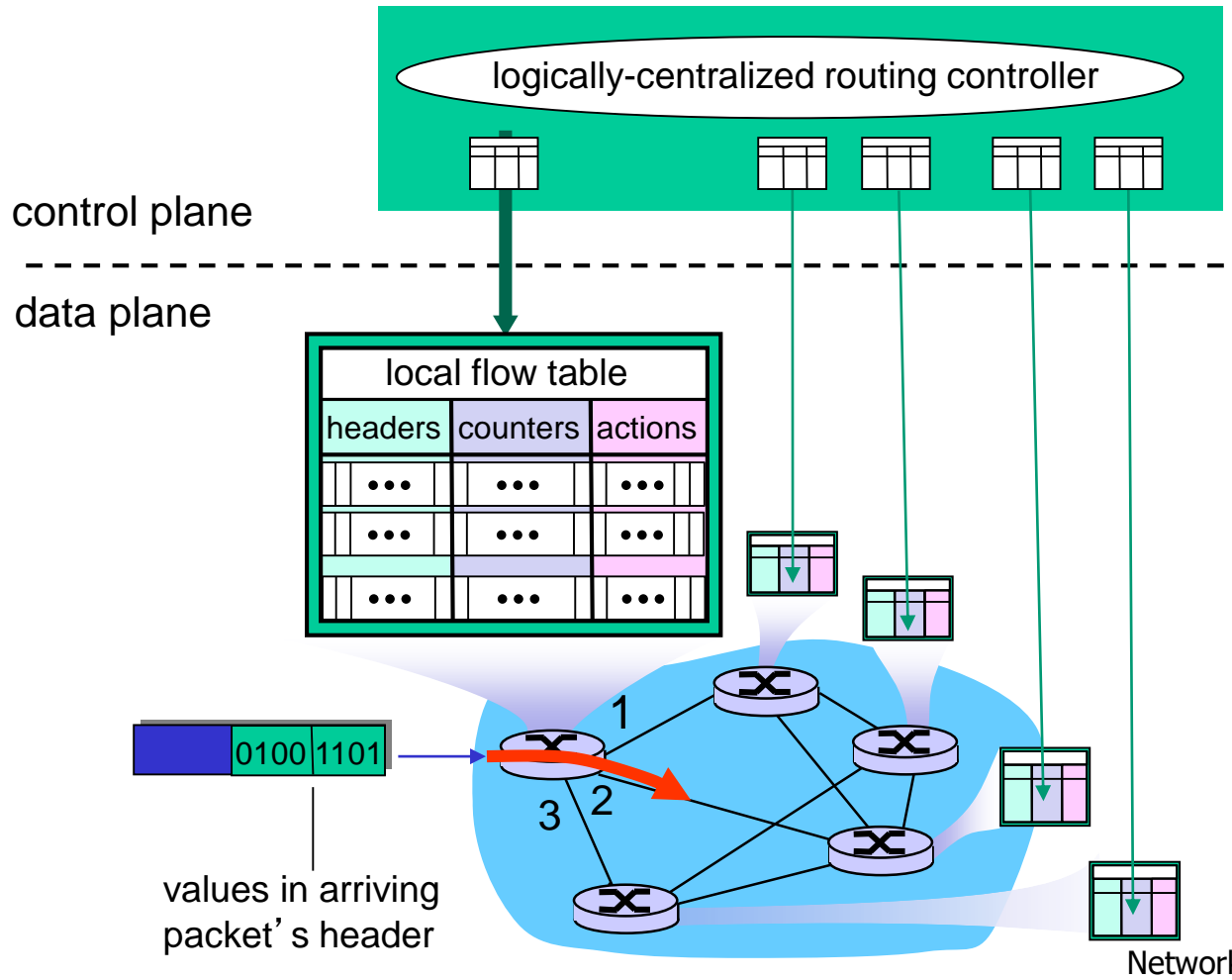
一般转发和SDN

- match
- action
- OpenFlow 实例
- atch-plus-action 一般过程

一般化转发 和 SDN

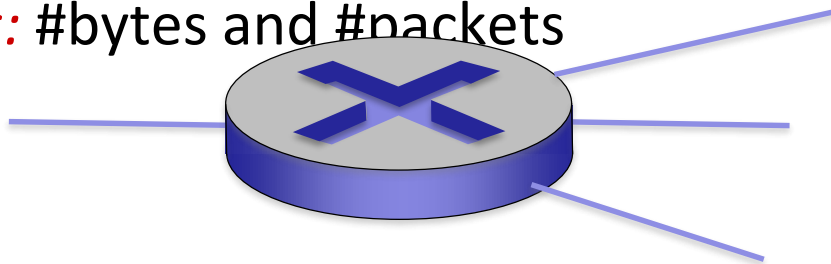
每个路由包含 *flow table*

由逻辑上集中的路由控制器进行计算和分发



OpenFlow 数据平面抽象

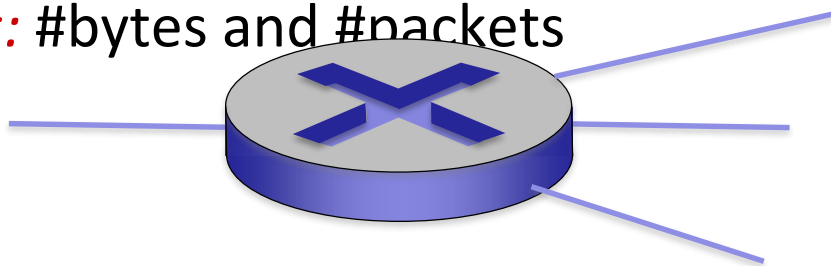
- *flow*: 由头部字段(header fields)定义
- 一般化前向转发: simple packet-handling rules
 - *Pattern*: 与头部字段值匹配
 - *Actions*: (匹配的数据报): drop, forward, modify, matched packet or send matched packet to controller
 - *Priority*: 清楚重叠模式
 - *Counters*: #bytes and #packets



Flow table (computed and distributed by controller)
定义路由match+action 规则

OpenFlow数据平面抽象

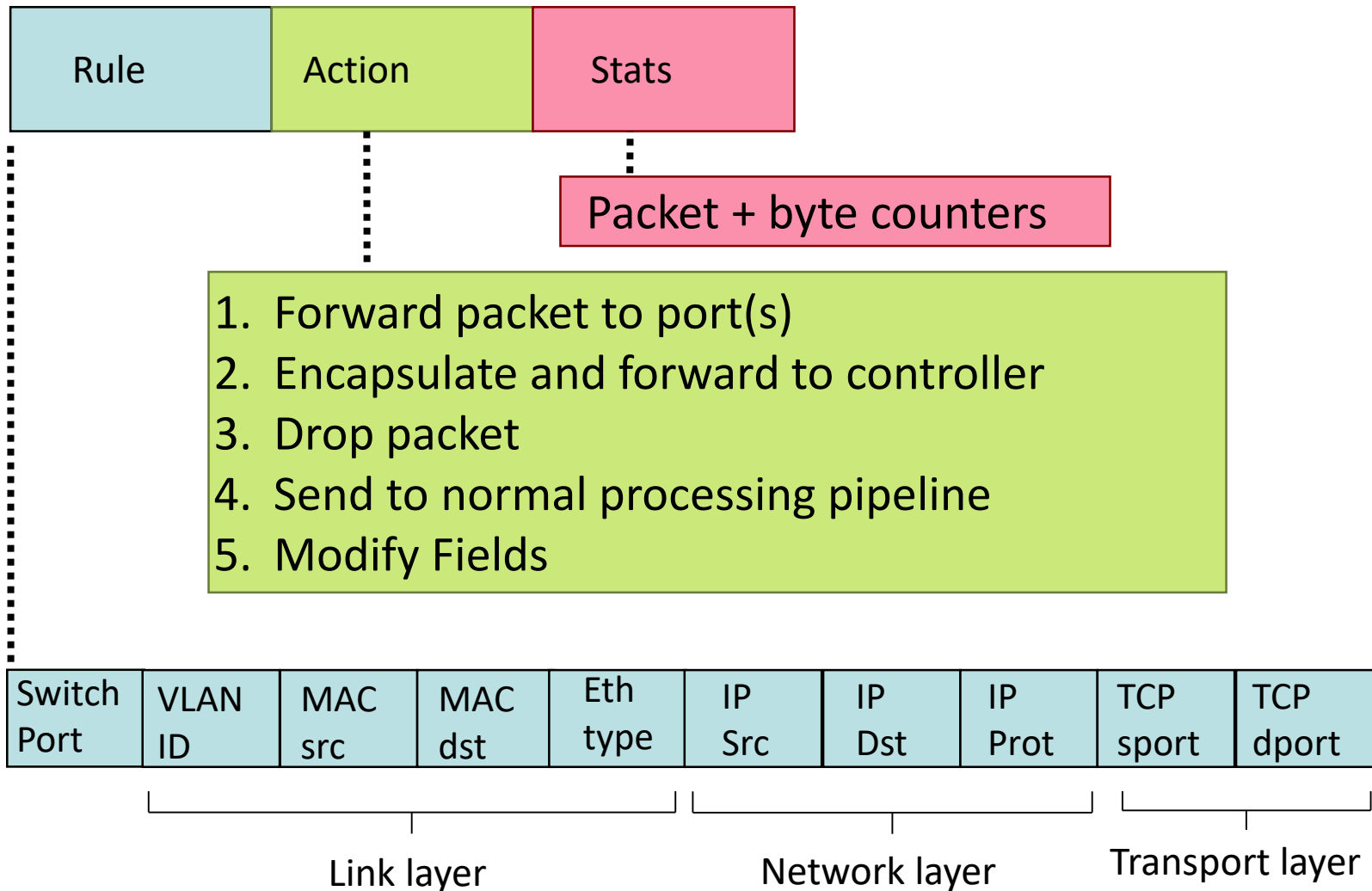
- *flow*: 由头部字段(header fields)定义
- 一般化前向转发: simple packet-handling rules
 - *Pattern*: 与头部字段值匹配
 - *Actions*: (匹配的数据报): drop, forward, modify, matched packet or send matched packet to controller
 - *Priority*: 清楚重叠模式
 - *Counters*: #bytes and #packets



* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table 条目



实例

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

实例

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

*layer 2 frames from MAC address 22:A7:23:11:E1:02
should be forwarded to output port 6*

OpenFlow 抽象

- *match+action*: unifies different kinds of devices

- Router

- *match*:

- 最长目标IP前缀

- *action*: forward out a link

- Switch

- *match*:

- 目标MAC 地址

- *action*: forward or flood

- Firewall

- *match*:

- IP 地址、

- TCP/UDP 端口号

- *action*: permit or deny

- NAT

- *match*:

- IP 地址、端口号

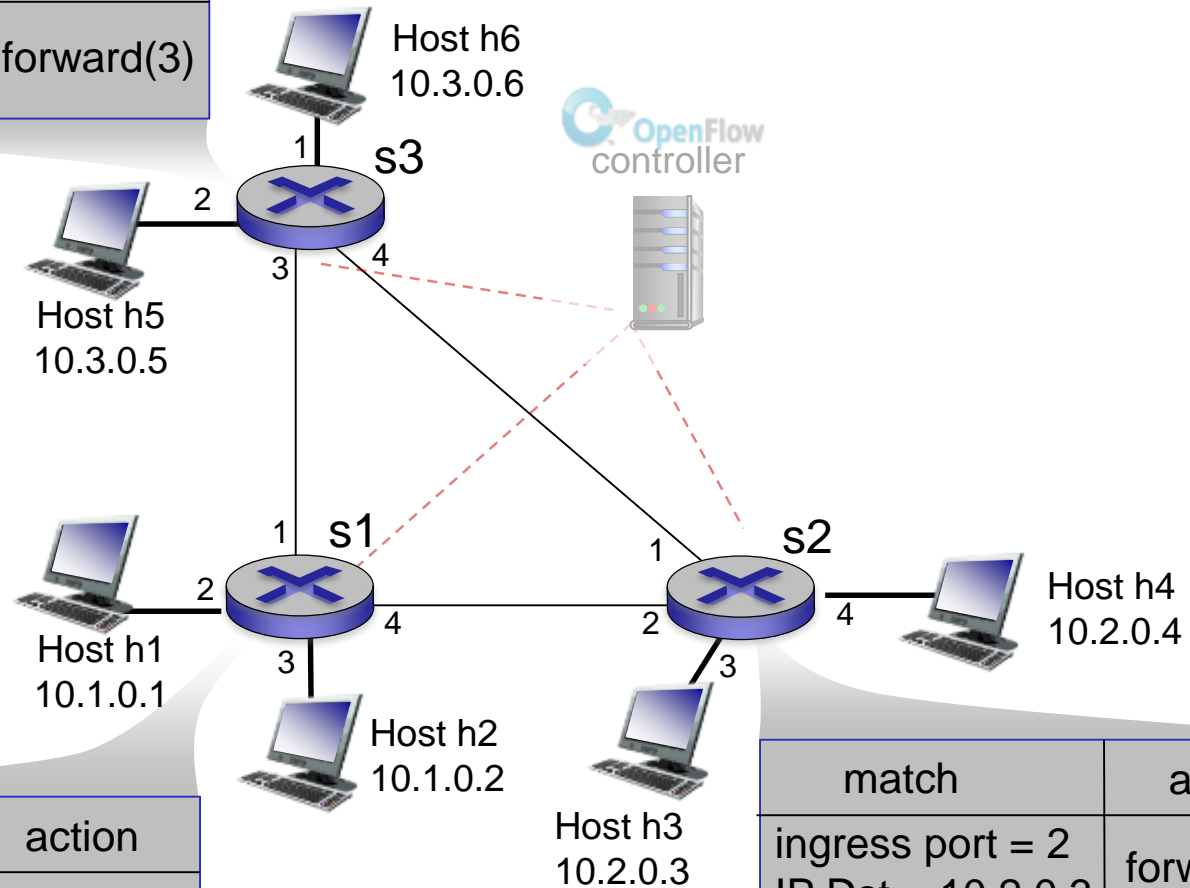
- *action*: rewrite address and port

OpenFlow 实例

Example:

自h5和h6的数据报发送至
h3 or h4, 通过 s1、s2到达
终点

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

ICMP互联网控制报文协议: (Internet Control Message Protocol)

□ 用于主机和路由器交换网络层信息

○ 错误报告:

不可达主机, 网络, 端口, 协议

○ echo request/reply (ping)

□ 层次“above” IP:

○ ICMP msgs 承载于IP数据报

□ ICMP message:

○ 错误类型,

○ 代码,

○ 引起错误IP数据报前8个字节

<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	echo回答 (ping)
3	0	目的网络不可达
3	1	目的主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的网络未知
3	7	目的主机未知
4	0	源抑制 (congestion control - not used)
8	0	echo请求 (ping)
9	0	路由通告
10	0	路由发现
11	0	TTL 过期
12	0	IP 首部错误

Traceroute and ICMP

- ❑ Source 发送不可达 UDP 端口号的报文至 dest
 - 第一个TTL =1
 - 第二个TTL=2, etc.
 - 不可端口号
- ❑ 当 nth 数据报抵达 路由:
 - 路由丢弃数据报
 - 发送 source 一个 ICMP 消息 (type 11, code 0)
 - 消息包含路由名字& IP 地址

- ❑ 当 ICMP 消息回来后 source 计算 RTT
- ❑ Traceroute 发送 3次

Stopping criterion

- ❑ UDP 段最终到达主机
- ❑ ICMP 返回“host unreachable”报文 (type 3, code 3)
- ❑ 源主机得到ICMP消息, 停止.

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

IPv6

□ Initial motivation:

32-bit 地址空间很快将被用完（已经用完）。

□ Additional motivation:

- 首部格式帮助加速 处理/前向转发
- 首部格式帮助支持 QoS

IPv6 datagram format:

- 固定 40 byte 首部
- 不允许分片

IPv6 Header (Cont)

流量类型(Priority):

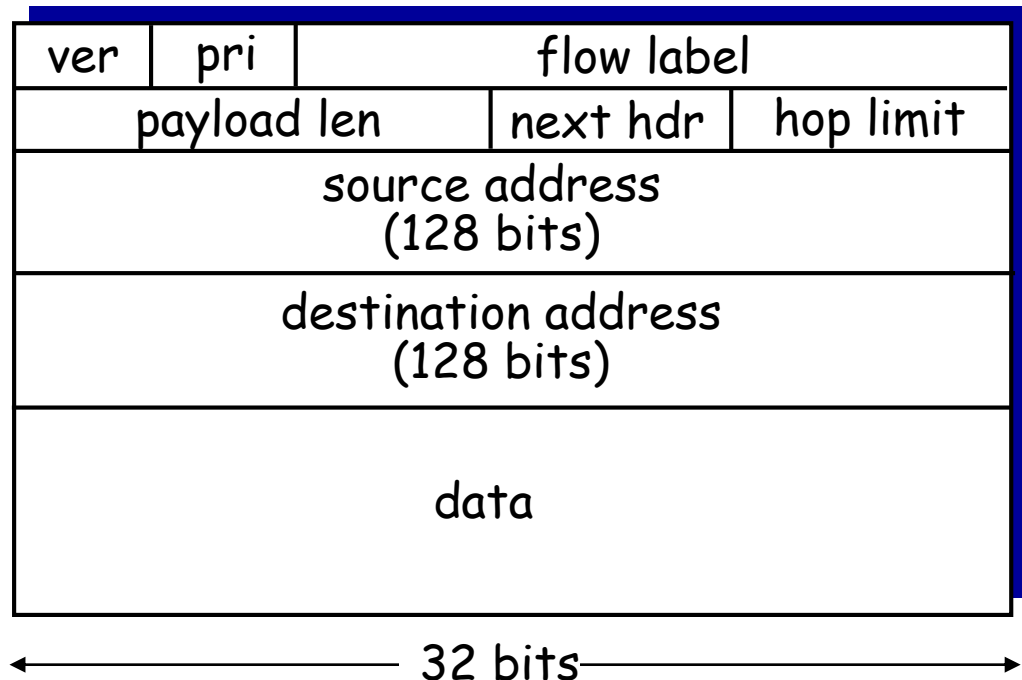
标识“流”内数据报优先级

流标签(Flow Label):

标识不同数据所属“流”(“流”没有准确定义).

下一首部(Next header):

标识上层协议



与 IPv4 的不同

❑ 检验和(*Checksum*):

- ❑ 为了加快处理, 没有设置。

❑ 选项(*Options*):

- ❑ 在头部意外进行设置("Next Header" field)

❑ *ICMPv6*:

❑ 新版ICMP

- 新的消息类型, e.g. "Packet Too Big"
- 多播组管理功能

IPv4 To IPv6迁移

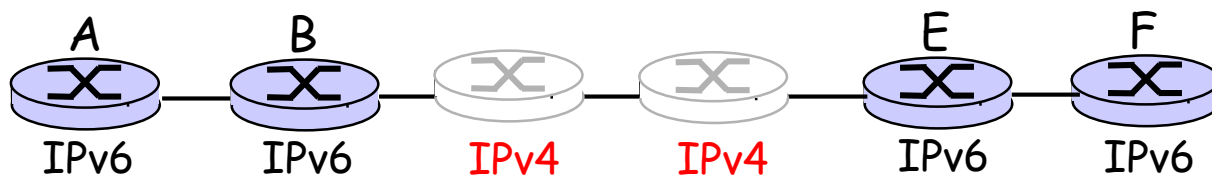
- ❑ 不可能所有的路由同时迁移
 - no “flag days”
 - IPv4 和 IPv6 共存?
- ❑ 隧道(*Tunneling*):
- ❑ IPv6作为IPv4数据报的负载在IPv4路由器间流动

隧道 Tunneling

Logical view:



Physical view:

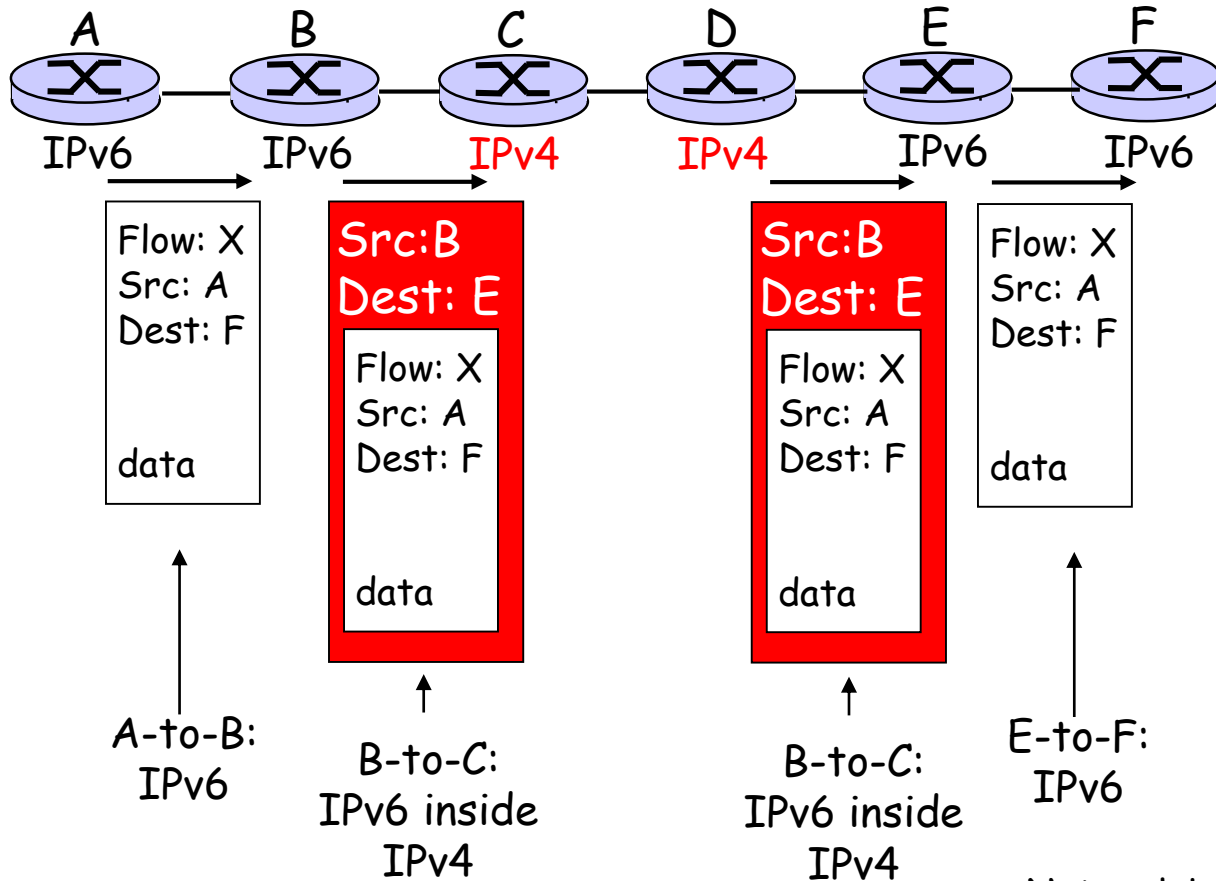


隧道

Logical view:



Physical view:



IPv6: adoption

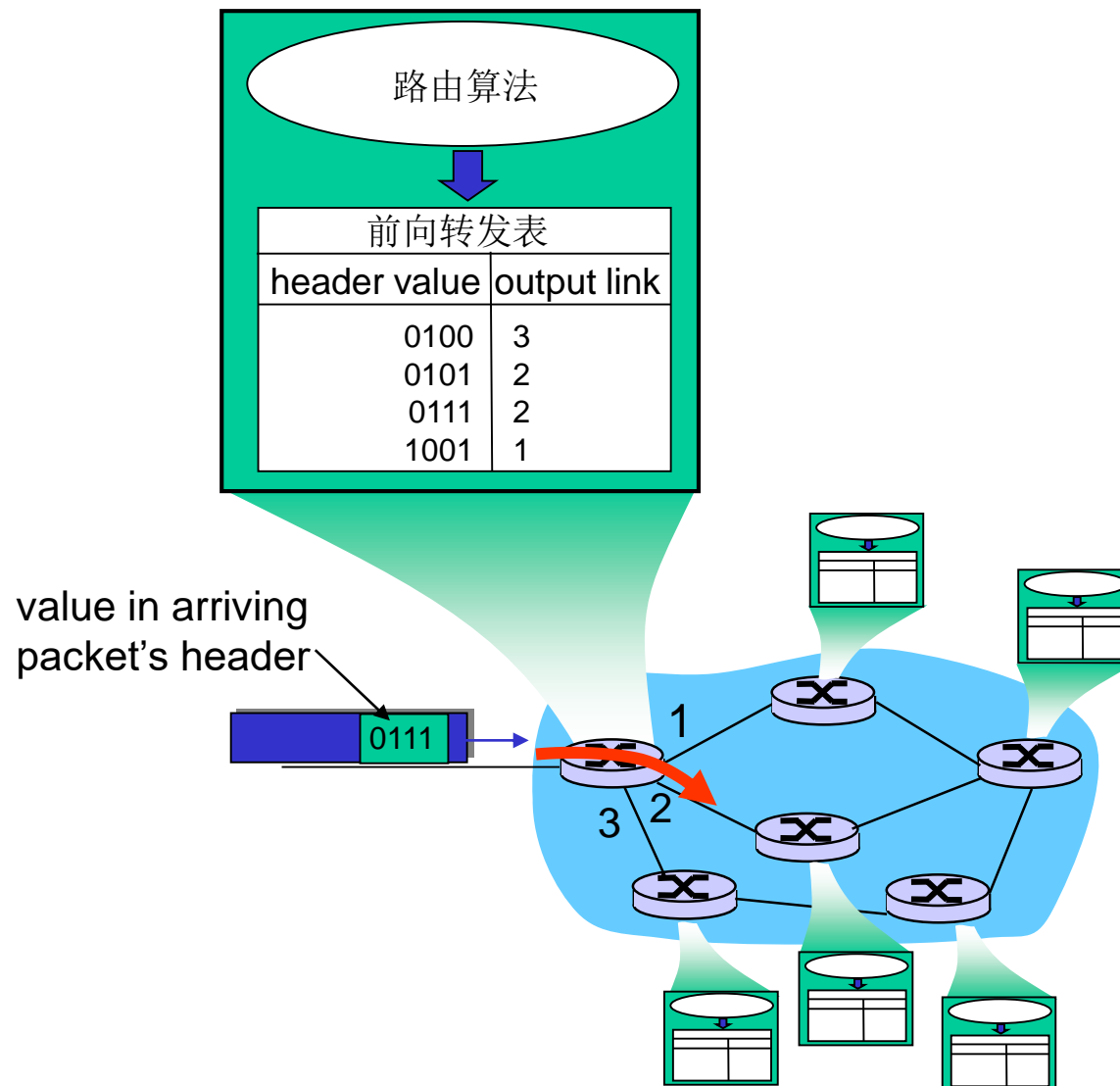
- ❑ US National Institutes of Standards estimate [2013]:
 - ~3% 工业IP路由
 - ~11% US 政府路由

- ❑ *Long (long!) time for deployment, use*
 - 20 years and counting!
 - *Why?*

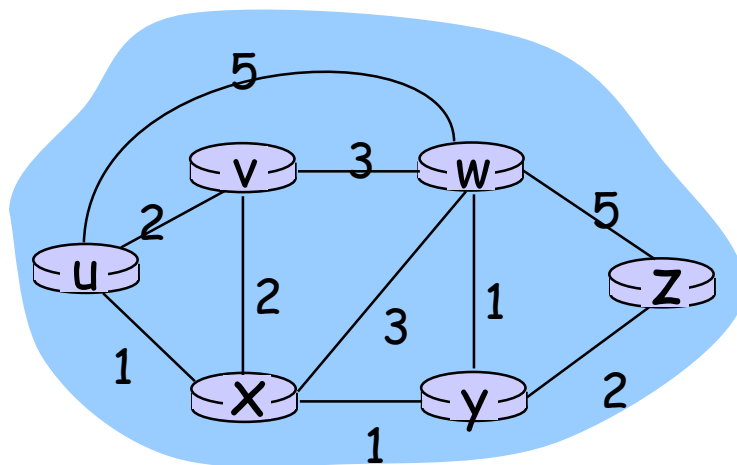
Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由

路由, 前向转发的相互作用



Graph 抽象



Graph: $G = (N, E)$

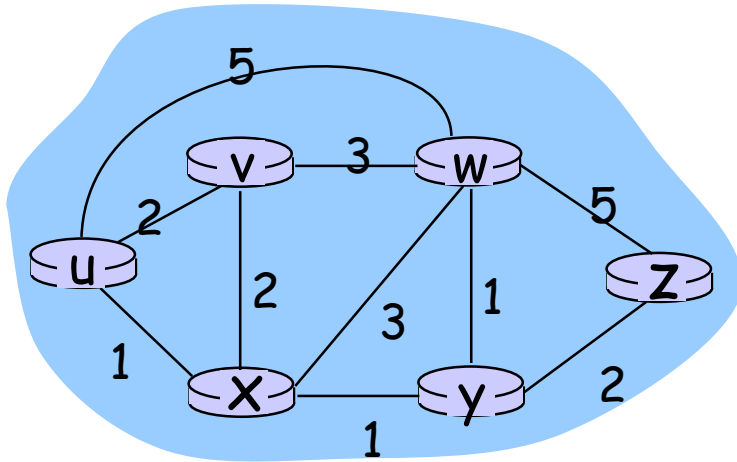
$N =$ 路由集合 $= \{ u, v, w, x, y, z \}$

$E =$ 链路集合 $= \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Remark: 计算机网络中图抽象非常有用！

Example: P2P, N 是端节点数目 and E 是TCP连接的条数。

Graph 抽象: 费用(costs)



- $c(x,x')$ = 链路(x,x')的开销

- e.g., $c(w,z) = 5$

- 开销可以是1, 或者是带宽的反相关函数, 或者是拥塞的反相关函数

路径的开销($x_1, x_2, x_3, \dots, x_p$) = $c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Question: u 和 z 之间的最小开销路径?

Routing algorithm: 找到最低费用的路径

路由算法分类

全局或分布式选路算法?

全局:

- ❑ 所有的路由都有全局的拓扑(topology), 链路开销信息
- ❑ “链路状态” 算法

分布式:

- ❑ 路由仅了解物理相连的邻居和相应的链路开销
- ❑ 迭代的计算过程,
相邻路由器之间路由信息交换
- ❑ “距离矢量” 算法

静态或动态?

静态:

- ❑ 路由变化缓慢

动态:

- ❑ 路由变化较快慢
 - 周期性更新
 - 会对链路开销变化做出响应

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

链路状态选路算法

Dijkstra's 算法

- 网络拓扑和所有的链路费用都是已知的
 - 通过 “链路状态广播”
 - 所有节点都有相同信息
- 计算每个节点 (**source**) 到所有其他节点的最低费用路径
 - 根据最短路径更新 转发表
- 迭代:
 - 在 k 次迭代后, 可知道到 k 目的节点的最低费用路径

Notation:

- $c(x,y)$: x 到 y 节点的费用;
 $= \infty$ 如果不存在直接链路
- $D(v)$: 当前从源节点到目的节点的最低费用路径的费用
- $p(v)$: 从源节点到目的节点 v 沿着当前最低费用路径的前一节点
- N' : 节点子集:

Dijkstra's 算法

1 **Initialization:**

2 $N' = \{u\}$

3 for all nodes v

4 if v adjacent to u

5 then $D(v) = c(u,v)$

6 else $D(v) = \infty$

7

8 **Loop**

9 find w not in N' such that $D(w)$ is a minimum

10 add w to N'

11 update $D(v)$ for all v adjacent to w and not in N' :

12 $D(v) = \min(D(v), D(w) + c(w,v))$

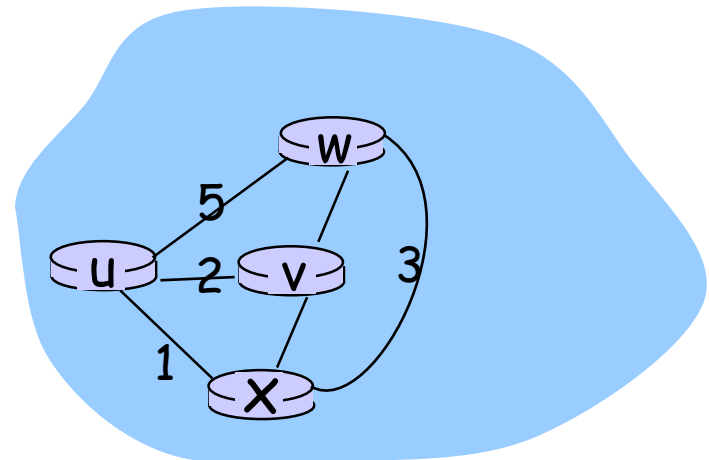
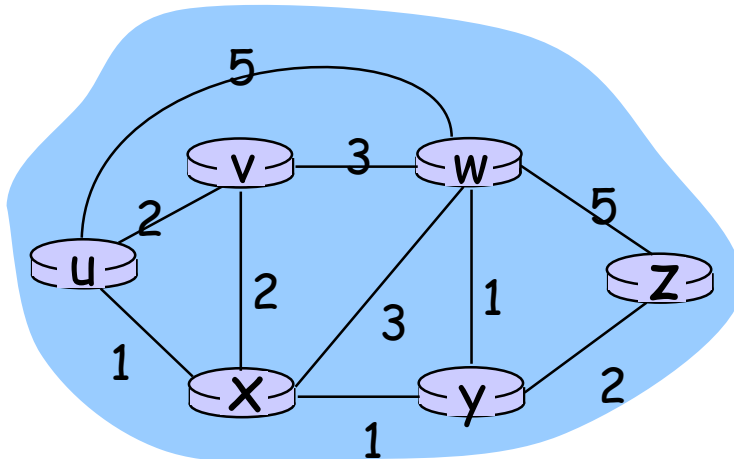
13/* 到 v 节点的新 开销等于 旧的开销 或者

14 最短路径开销加上 w 到 v 的开销*/

15 **until all nodes in N'**

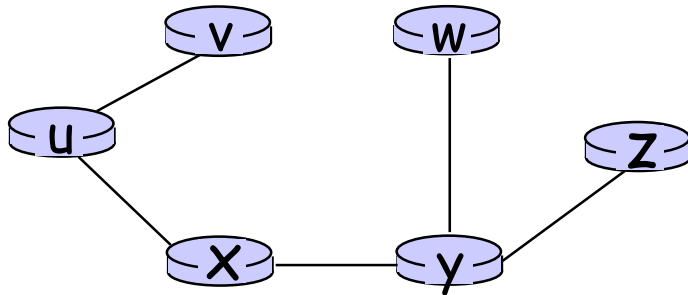
Dijkstra's算法: 举例

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's算法: 举例(2)

u出发的最短路径树:



u的转发表:

目的地	路径
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

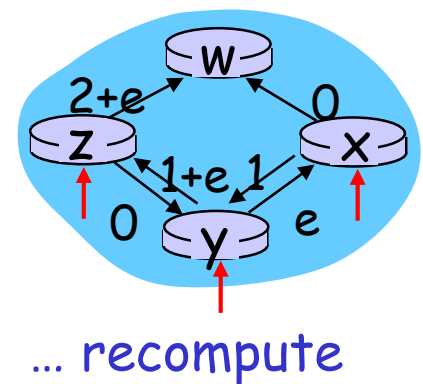
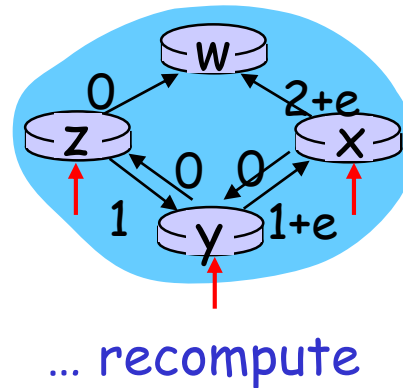
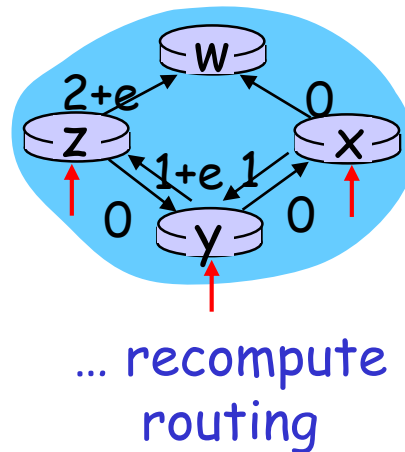
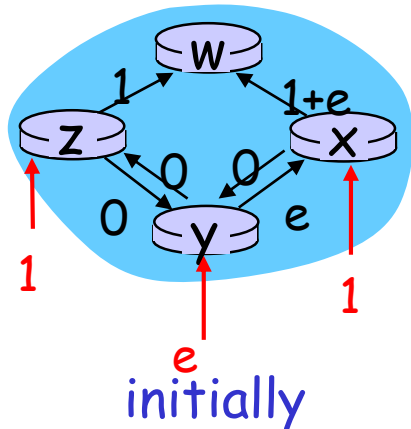
Dijkstra's 算法, 讨论

算法复杂性: n 个节点

- 每次迭代: 需要检查所有没有在 N 中的节点
- $n(n+1)/2$ 次比较: $O(n^2)$
- 改进的算法可以达到: $O(n \log n)$

可能发生的振荡:

- e.g., link cost = 流量负载



Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由

距离矢量算法

Distance Vector Algorithm

Bellman-Ford 等式 (动态规划)

Define

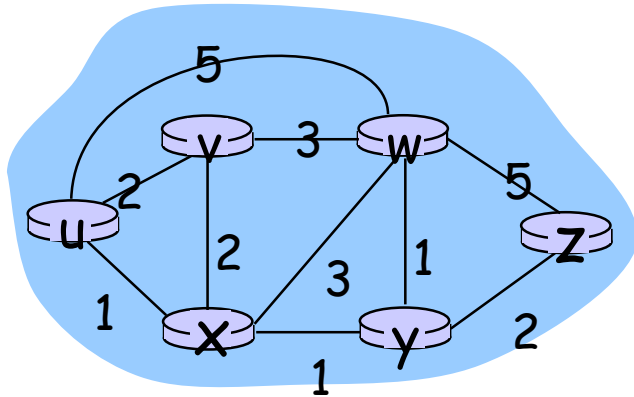
$d_x(y) :=$ 从 x 到 y 的最小开销路径的开销

Then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

\min 取自 x 的所有相邻节点 v

Bellman-Ford等式例子

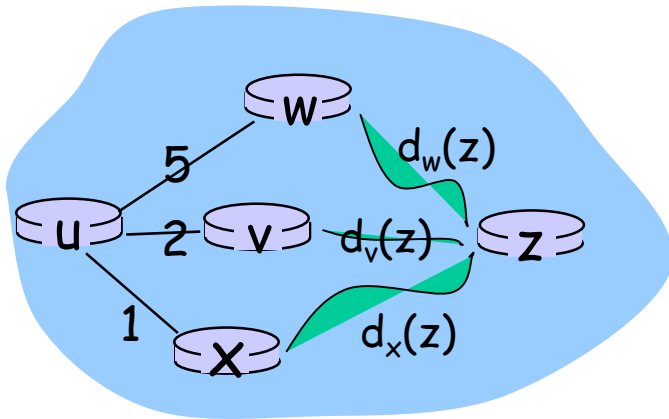


显然, $d_w(z) = 3$, $d_v(z) = 5$, $d_x(z) = 3$

B-F 等式就有:

$$d_u(z) = \min \{ c(u,w) + d_w(z), \\ c(u,v) + d_v(z), \\ c(u,x) + d_x(z) \}$$

$$= \min \{ 5 + 3, \\ 2 + 5, \\ 1 + 3 \} = 4$$



Bellman-Ford 计算得到的最短路径 → forwarding table

距离矢量算法

- $D_x(y)$ = 从 x 到 y 最小开销的估计
- $c(x,v)$ 节点 x 到相邻节点 v 的开销
- $D_x = [D_x(y): y \in N]$ 节点 x 维护的距离矢量
- 节点 x 维护它相邻节点的距离矢量
 - 对于每个相邻节点 v , x 维护
$$D_v = [D_v(y): y \in N]$$

距离矢量算法

Basic idea:

- ❑ 每个节点不时的向邻居发送它的距离矢量拷贝
- ❑ 异步的
- ❑ 当节点 x 从它的一个邻居接收到一个新的 DV 估计时, 它保存 v 的距离矢量, 然后用 **B-F** 等式更新 DV:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- ❑ 只要所有的节点继续以异步方式交换他们的 DV 每个 $D_x(y)$ 就会收敛到 $d_x(y)$

距离矢量算法(5)

迭代, 异步:

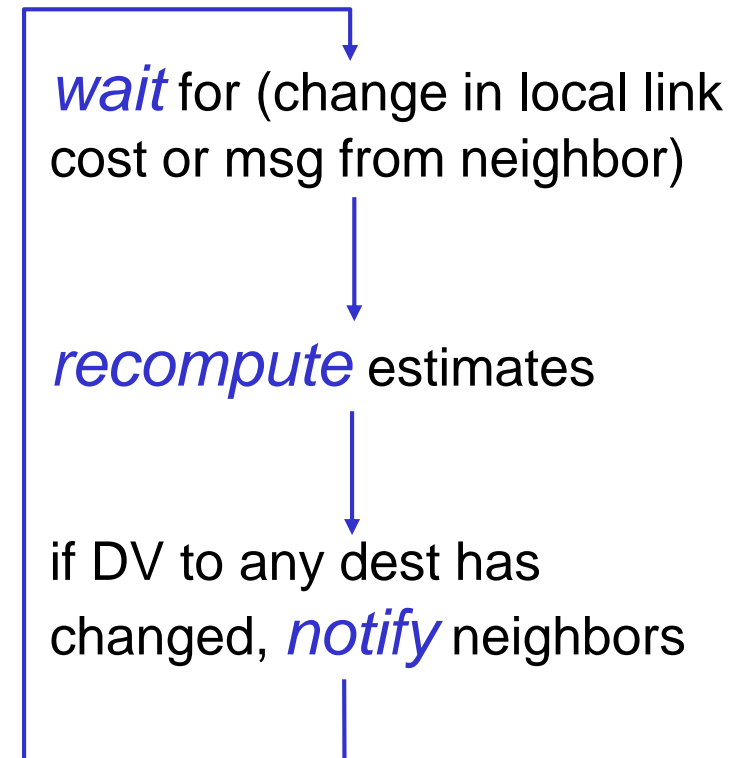
本地迭代的触发:

- ❑ 本地链路开销发生变化
- ❑ 来自相邻节点DV 更新消息

分布式:

- ❑ 节点DV发生变化时通知相邻节点
 - 有必要时相邻节点又会通知它的相邻节点

Each node:



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \\ = \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} \\ = \min\{2+1, 7+0\} = 3$$

node x table

		cost to		
from		x	y	z
	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

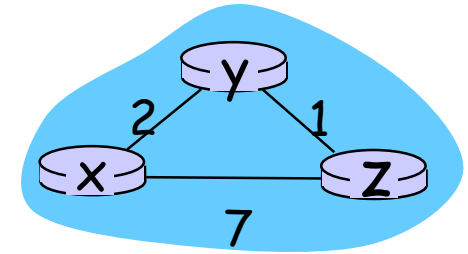
node y table

		cost to		
from		x	y	z
	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

node z table

		cost to		
from		x	y	z
	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

		cost to		
from		x	y	z
	x	0	2	3
	y	2	0	1
	z	7	1	0



time

Network Layer 4-100

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \\ = \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} \\ = \min\{2+1, 7+0\} = 3$$

node x table

		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

node y table

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

node z table

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

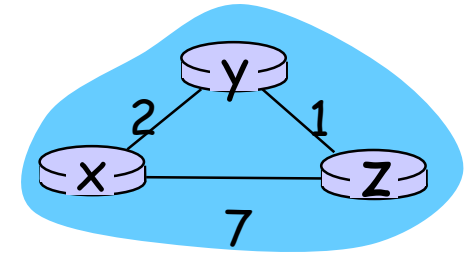
		cost to		
from		x	y	z
	x	0	2	7
	y	2	0	1
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
from		x	y	z
	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

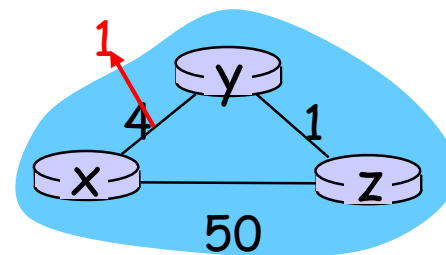


time

Distance Vector:链路费用变化

链路费用变化:

- ❑ 节点检测到本地链路费用发生变化
- ❑ 更新路由信息, 重新计算DV
- ❑ 如果DV 发生变化, 则向邻居通知最新DV



“好消息”传的快

At time t_0 , y检测到本地链路费用发生变化, 更新 DV, 向邻居通知.

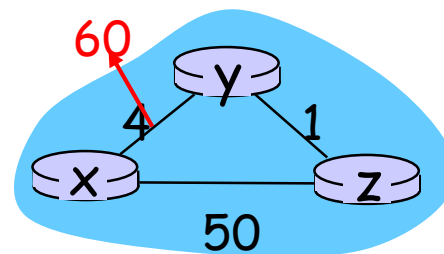
At time t_1 , z 从 y接收到更新, 更新自己的距离表, 计算出到x的新最低费用, 向另据发送它的DV

At time t_2 , y 从z接收到更新, 更新自己的距离表, 计算出y的最低费用没有变化, 不发送报文, 算法进入静止状态

Distance Vector: 链路费用变化

链路费用变化

- ❑ 坏消息传的慢
- ❑ 44 次迭代，算法才稳定下来



毒性逆转:

- ❑ 如果z路由通过y到达x:
 - z告诉y它通过z到x的距离无穷大 (所以y不会选择z到x)
- ❑ 毒性逆转不能完全解决问题，涉及到3个或更多节点的环路将无法使用该方法。

LS 与DV 算法比较

消息的复杂性

- LS: 节点数 n , 链路数 E , 消息数为 $O(nE)$
- DV: 只在相邻节点间交换消息
 - 收敛时间不一定

收敛速度

- LS: $O(n^2)$ $O(nE)$ 个消息
 - 可能会有波动
- DV: 收敛时间不一定
 - 可能会有路由环
 - 计数到无穷(count-to-infinity) 问题

健壮性: 路由器发生故障情况下的情况?

LS:

- 节点通告不正确的链路开销
- 每个节点计算自己的路由表

DV:

- 节点广播不正确的路径开销
- 每个节点的路由表都向相邻节点通报
 - 错误通过网络传播

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由

层次路由

Hierarchical Routing

前面都是理想化的网络环境

- ❑ 所有路由平等

- ❑ 网络是扁平的

... 与真实网络环境有区别

规模:

600 million 目的网络:

- ❑ 路由表不可能存储所有网络!
- ❑ 路由交换的信息可能瘫痪网络!

管理自治

- ❑ internet = network of networks
- ❑ 每个网络管理员都希望能够控制自己的网络!

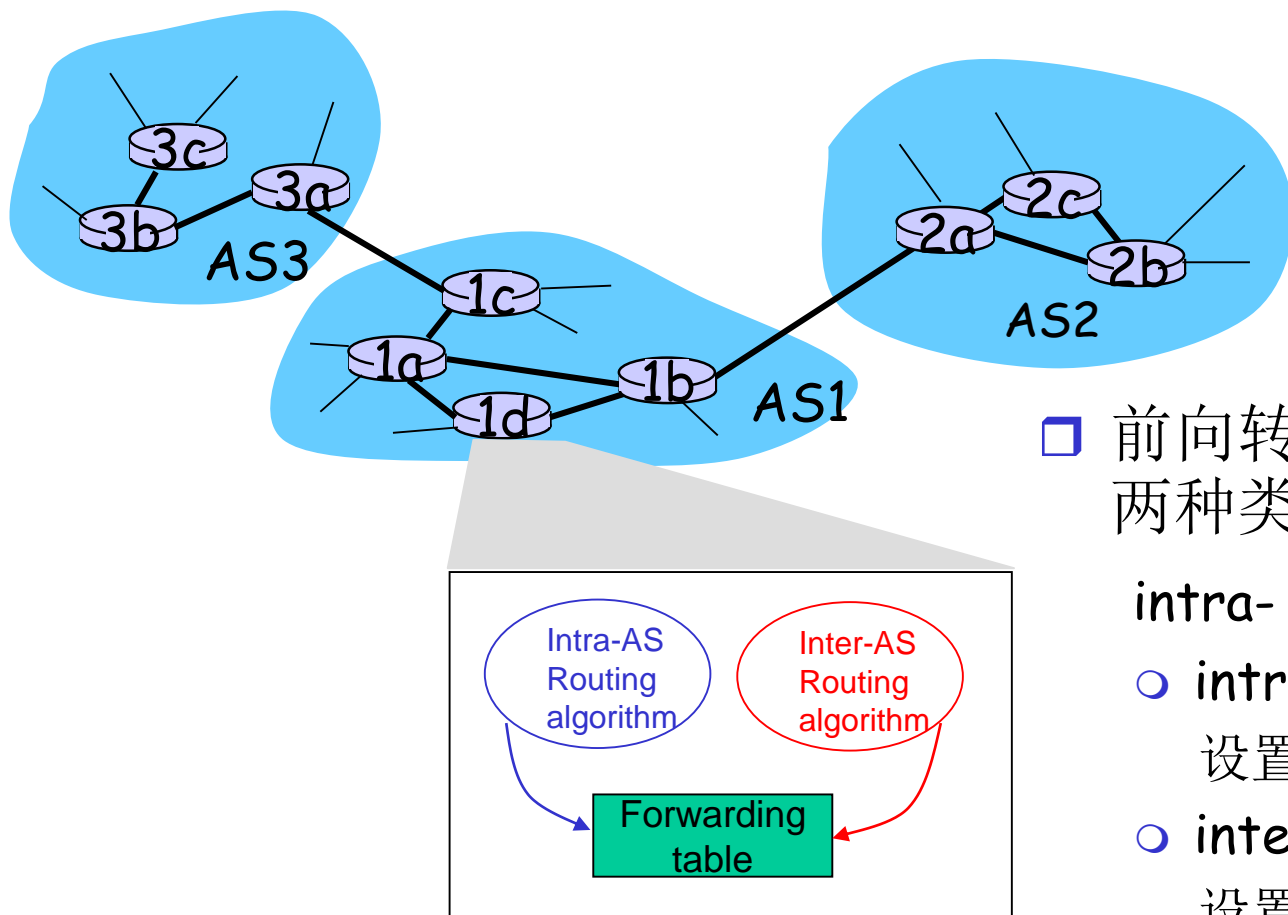
层次路由

- 通过将路由器组织进自治系统, “autonomous systems” (AS)
- 自治系统由一组通常在相同管理控制下的路由器组成
 - “intra-AS” 路由协议
 - 不同AS 系统运行不同 intra-AS 路由协议

网关路由

- 负责向本 AS之外的目的地转发分组

互联的 ASes



□ 前向转发表通常会配置两种类型的协议：

intra- 和inter-AS路由协议

- intra-AS

设置内部目的网络条目

- inter-AS & intra-As

设置外部目的网络条目

Inter-AS 任务

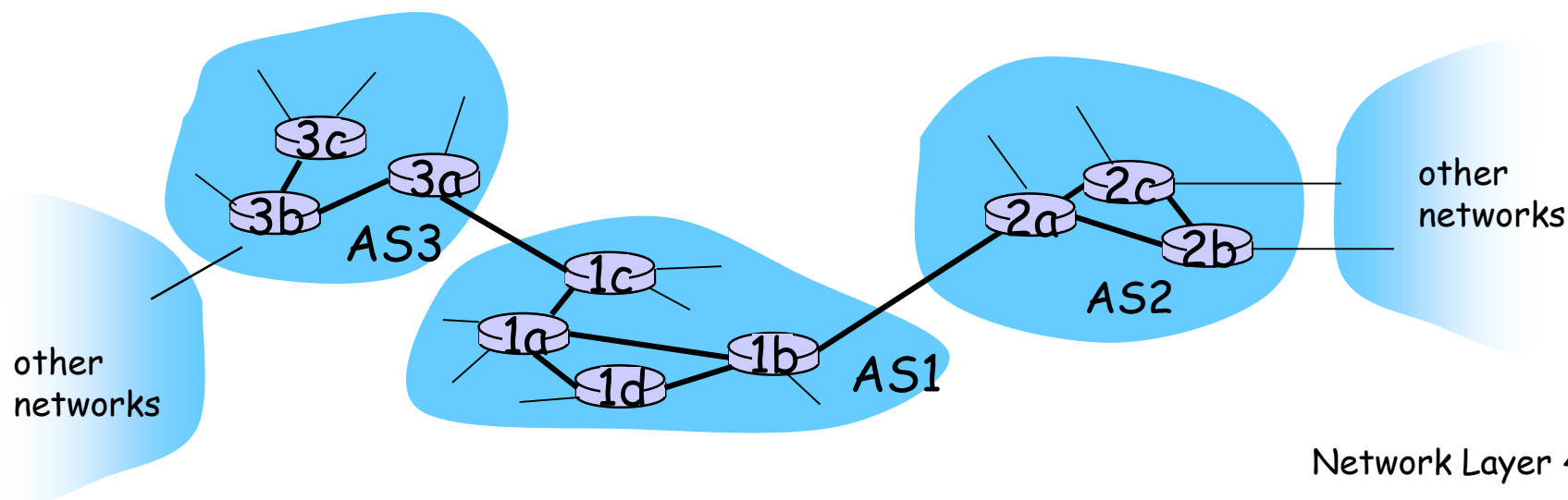
□ **AS1** 路由器接收到来自 **AS1**外部的数据报:

- 路由前向转发数据给网关路由,
- 但是哪一个呢?

AS1 must:

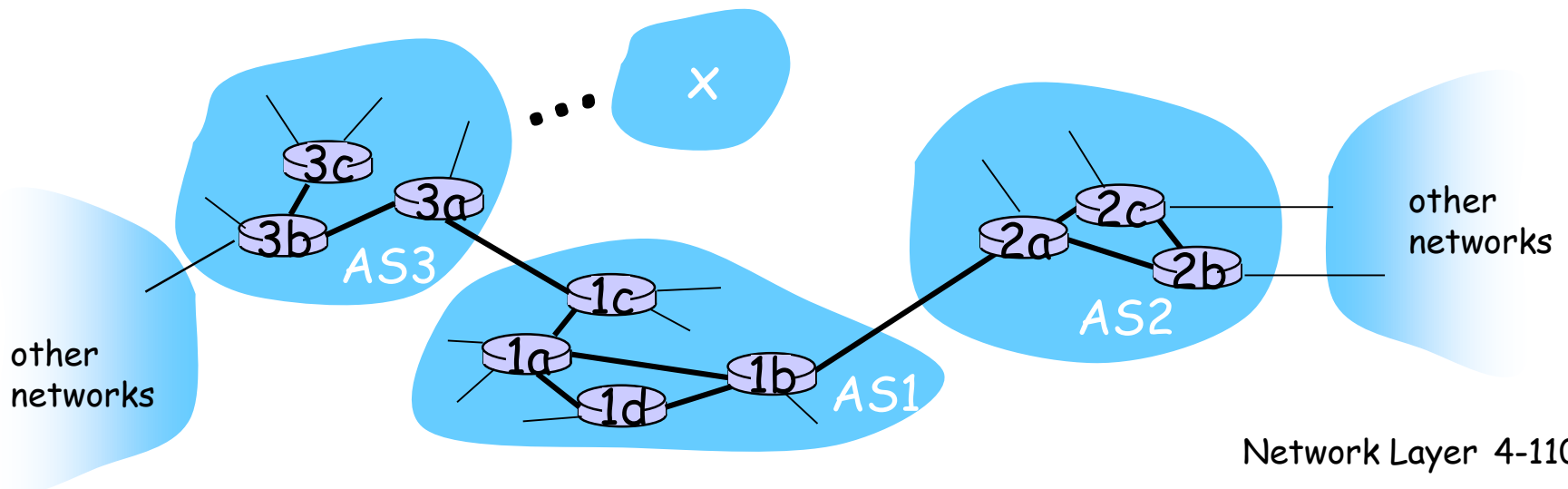
1. 需要了解经 **AS2**, **AS3** 分别都能到什么地方
2. 向 **AS1**中的所有路由器传播这些信息

Job of inter-AS routing!



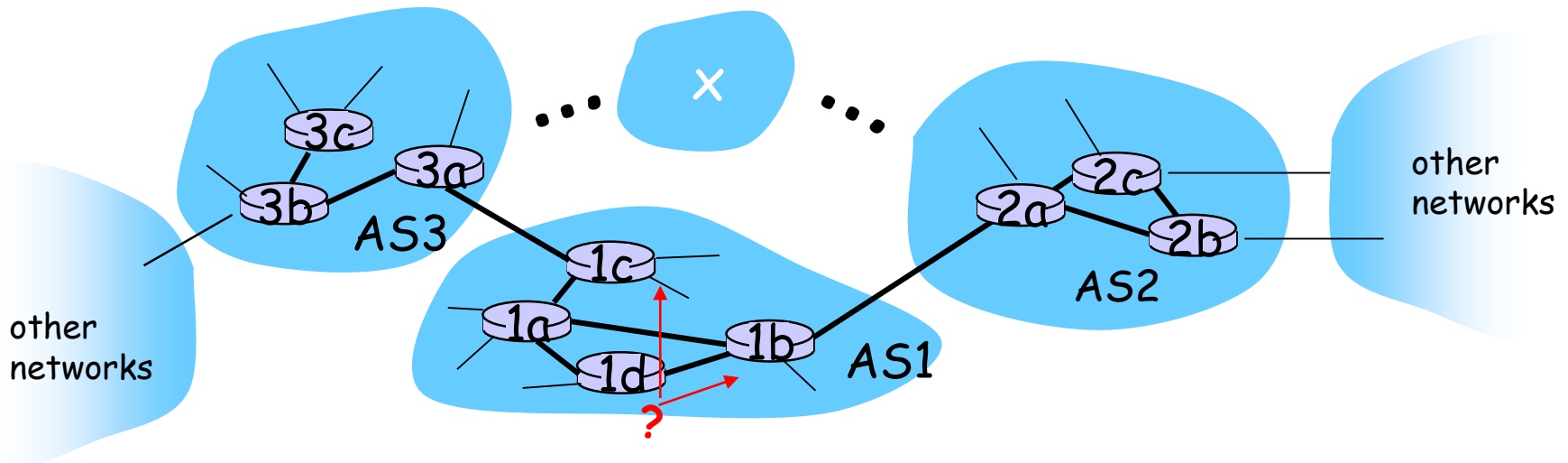
Example: 路由器 1d 前向转发表的设置

- ❑ 假设 AS1 了解到 (via inter-AS protocol) 子网 **x** 通过 AS3 (gateway 1c) 可达, 但是通过 AS2 不可达.
- ❑ AS1 广播可达信息.
- ❑ 路由器 1d 找到其接口 **I** 是到达 1c 最低费用路径上的路由器接口.
 - 将 (**x, I**) 放入其转发表



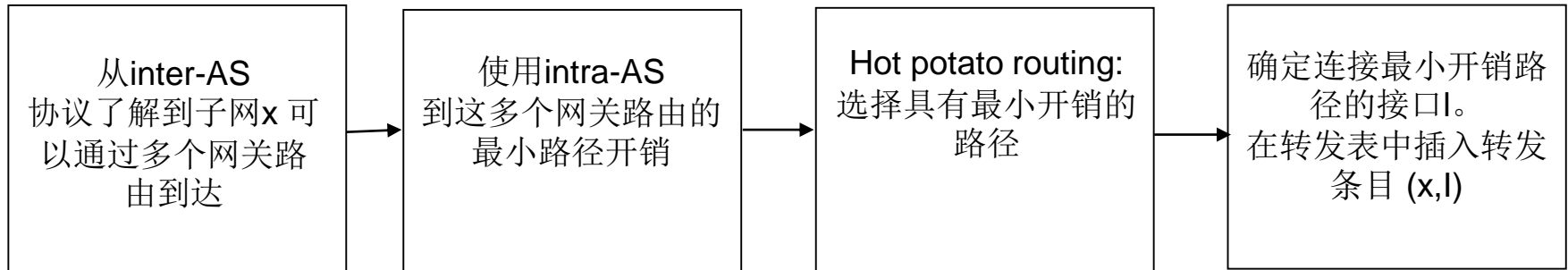
Example: 在多个 AS 中选择

- ❑ 假设 **AS1** 了解到 (通过 **inter-AS** 协议) 子网 **x** 通过 **AS3** (网关 **1c**) 可达, 通过 **AS2** 也可达.
- ❑ 路由器 **1d** 配置时需要找到到 **x** 子网的最佳路径后配置其转发表.
 - 这也是 **inter-AS** 路由协议的工作!



Example:在多个 AS中选择

- ❑ 热土豆选路(hot potato routing):
AS应该尽快的摆脱分组.



Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由

AS 内部选路

- ❑ 内部网关协议 **Interior Gateway Protocols (IGP)**
- ❑ 最常用**Intra-AS** 路由协议:
 - RIP: Routing Information Protocol
 - OSPF: Open Shortest Path First
 - IGRP: Interior Gateway Routing Protocol
(Cisco proprietary)

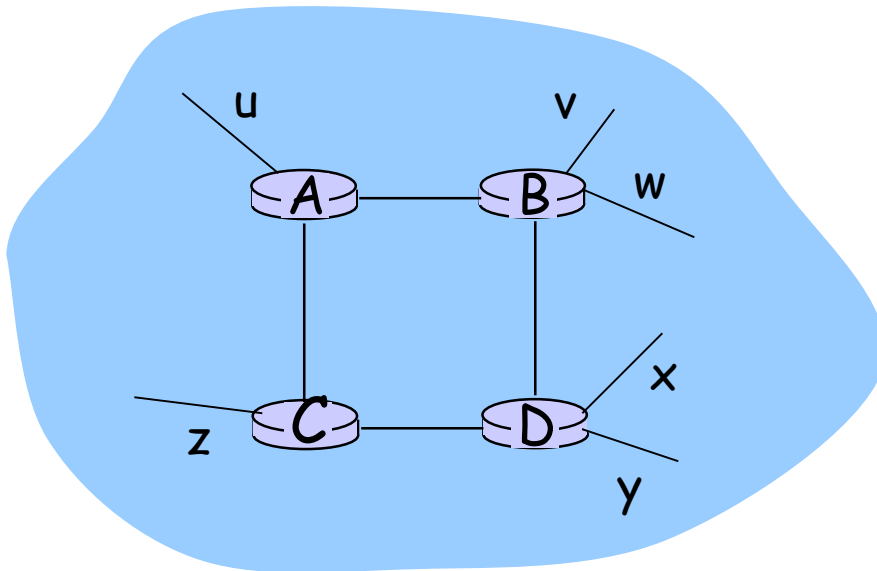
Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - **RIP**
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
Broadcast and
multicast routing

选路信息协议

RIP (Routing Information Protocol)

- ❑ 使用距离向量算法
- ❑ 包含在BSD-UNIX分发中， 1982年
- ❑ 距离标准: # hops (max = 15 hops)



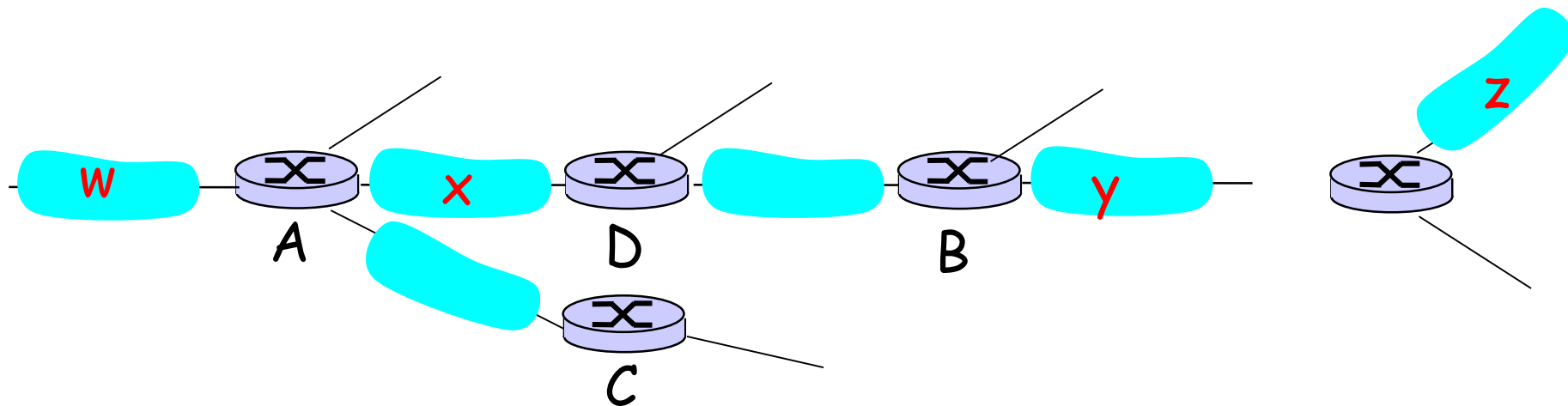
From router A to subnets:

<u>destination</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2

RIP 通告

- ❑ Distance Vectors: 30 秒交换一次
- ❑ Response Message (也叫通告 **advertisement**)
- ❑ 每个通告: 包含了一个由多达**25**个**AS**内的目的子网列表

RIP: 例子



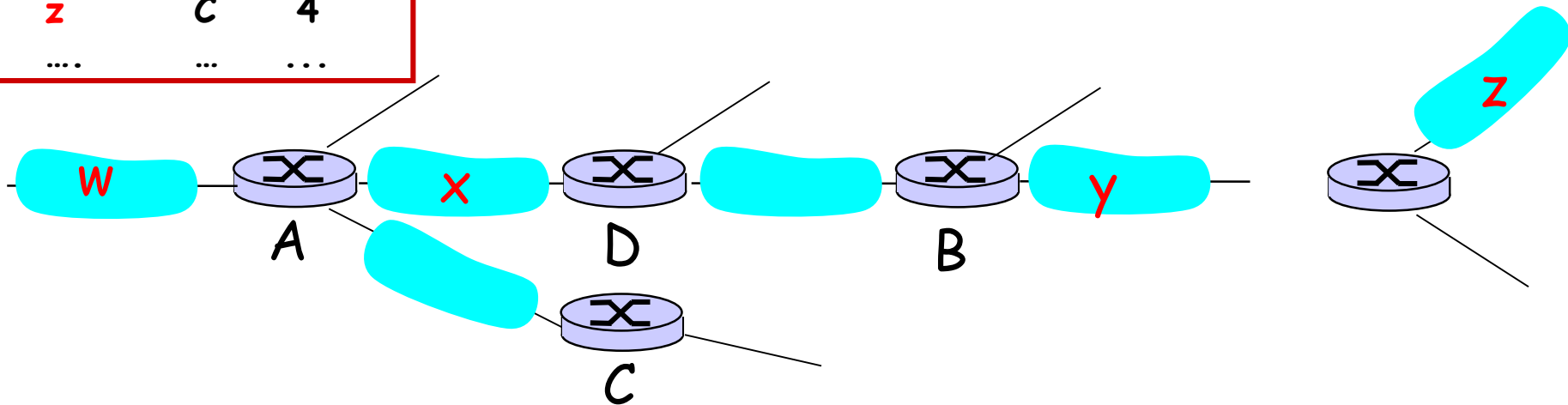
目的子网	下一跳路由	到目的地跳数 .
W	A	2
Y	B	2
Z	B	7
X	--	1
....

D的路由/转发表

RIP: 例子

A到D的一个通告

Dest	Next	hops
w	-	1
x	-	1
z	C	4
....



目的子网	下一跳路由	到目的地跳数 .
w	A	2
y	B	2
z	B A	7 5
x	--	1
....

Routing/Forwarding table in D

Network Layer 4-119

RIP: 链路错误恢复

路由器一旦超过 **180** 秒没有监听到邻居-->

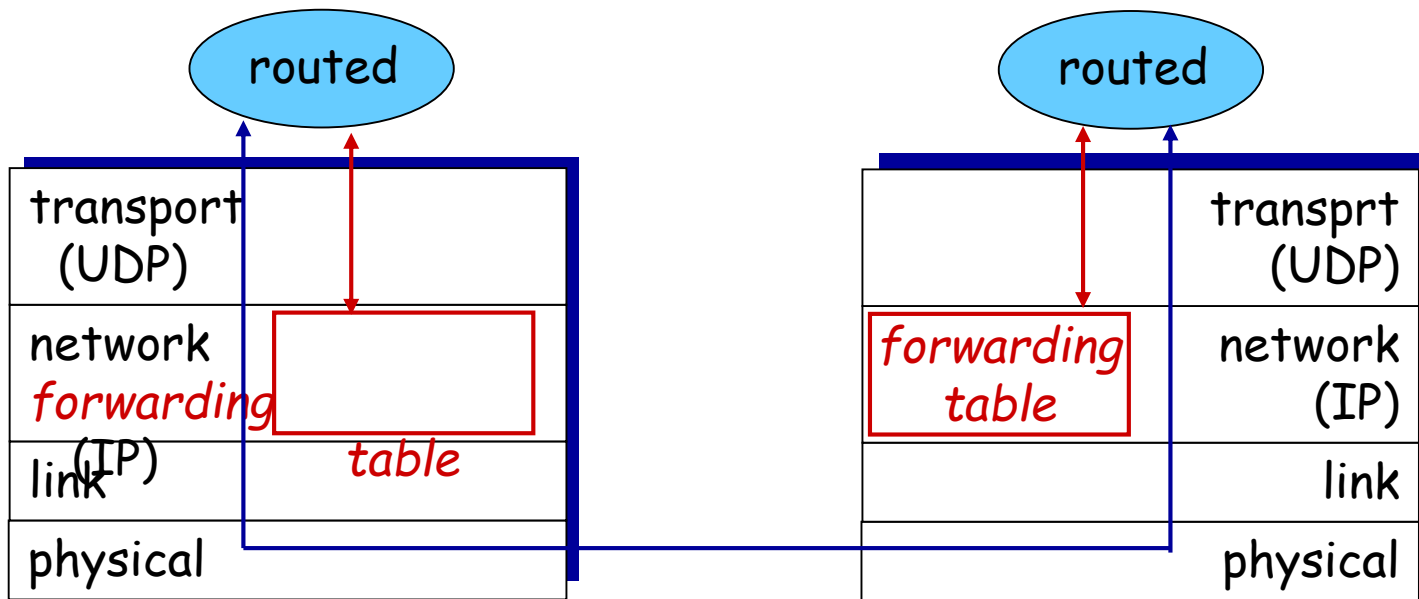
认为该邻居不可达/或者链路中断

- **RIP**修改本地路由表
- 发送通告广播消息
- 邻居也可能发送新的通告 (如果路由表改变)
- 链路中断信息能够很快传播到整个网络
- 毒性逆转(*poison reverse*)

用于阻止 ping-pong 循环(infinite distance = 16 hops)

RIP 路由表处理

- ❑ RIP 路由表由应用层进程管理 **route-d (daemon)**
- ❑ 通告信息封装在 **UDP** 数据报中, 周期性重复



Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

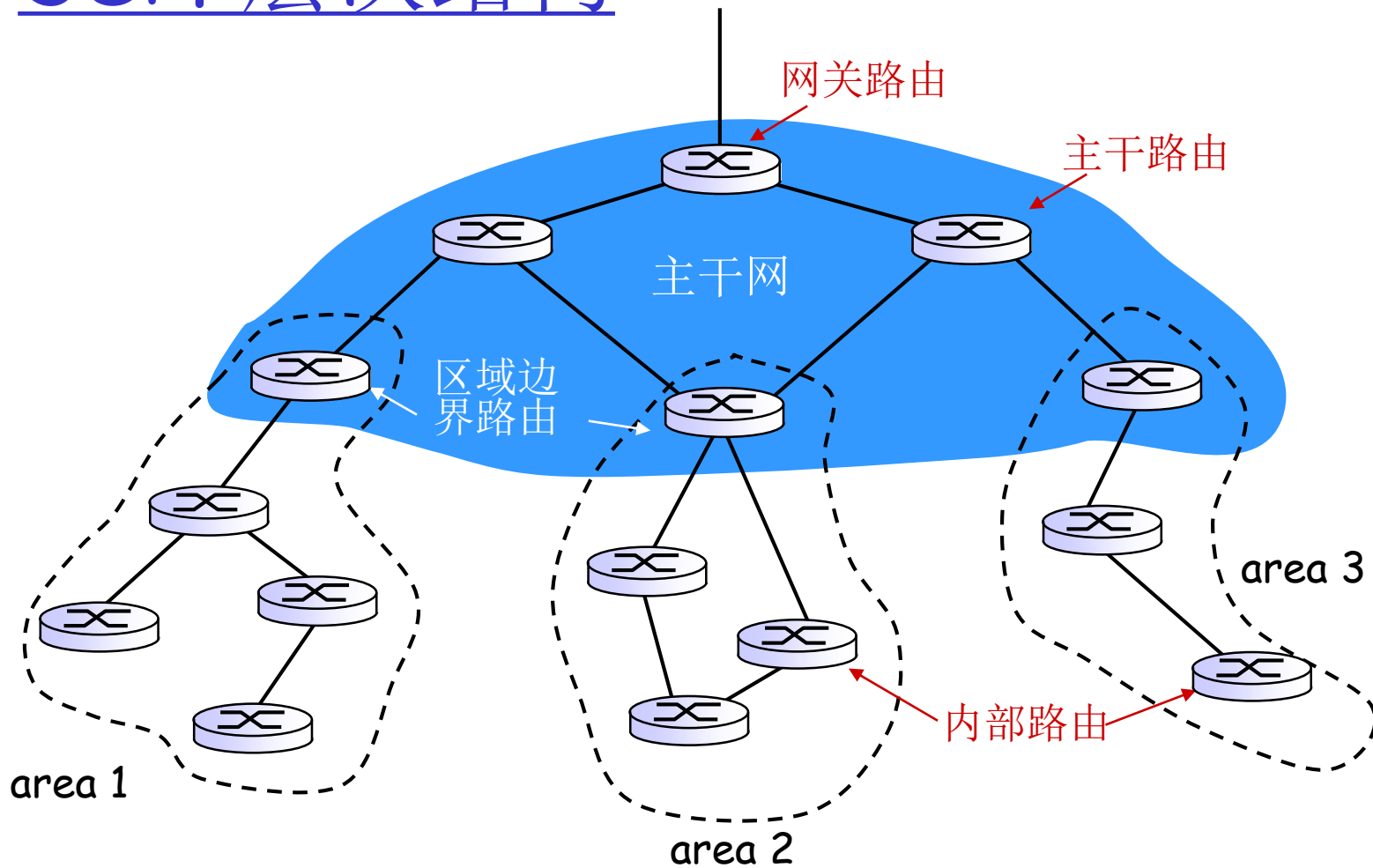
OSPF (Open Shortest Path First)

- ❑ “open”: 公开的
- ❑ 使用链路状态协议
 - LS 数据报分发
 - 每个节点拥有网络的拓扑
 - 使用Dijkstra's 算法进行路由计算
- ❑ OSPF 周期性的广播链路状态信息
- ❑ 向 AS 内所有路由器广播选路信息(通过flooding)
 - OSPF 通告包含在OSPF报文中，直接承载在 IP数据报中
(不是TCP or UDP)

OSPF 优点

- ❑ 安全： 所有 OSPF 都是经过鉴别（可以防止恶意入侵）
- ❑ 允许多条相同费用路径（RIP只允许一条）
- ❑ 对单播选路与多播选路的支持
 - Multicast OSPF 与OSPF 使用相同的拓扑数据
- ❑ 支持单个选路域内的层次结构

OSPF层次结构



OSPF层次结构

- ❑ **两个层次：**本地区域, 主干.
 - 仅在本地区域进行链路状态广播
 - 每个节点都仅知道本地区域的拓扑; 和通往其他区域的直接最短路径.
- ❑ **区域边界路由器：**同时属于区域与主干两个层次.
- ❑ **主干路由器：**执行主干中的选路.
- ❑ **网关路由器：**与其他自治系统的路由器交换选路消息.

Chapter 4: Network Layer

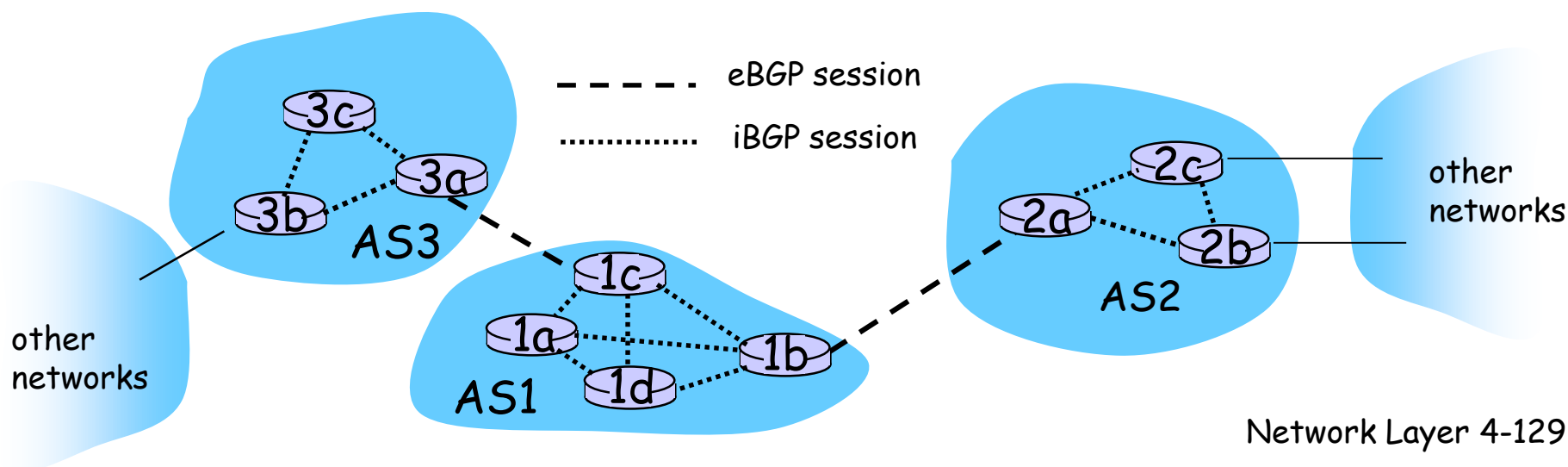
- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - **BGP**
- ❑ 4.7 广播和多播路由
Broadcast and
multicast routing

自治系统间选路: BGP

- ❑ **BGP (Border Gateway Protocol):** 事实标准
- ❑ **BGP** 提供了一种手段:
 1. 从相邻 **ASs**处获得子网可达性信息.
 2. 向该 **AS**内部的所有路由器广播这些可达性信息.
 3. 基于可达性信息和**AS**策略, 决定到达子网的路由.
- ❑ **BGP**允许每个子网向**Internet**的其余部分通告它的存在: *"I am here"*

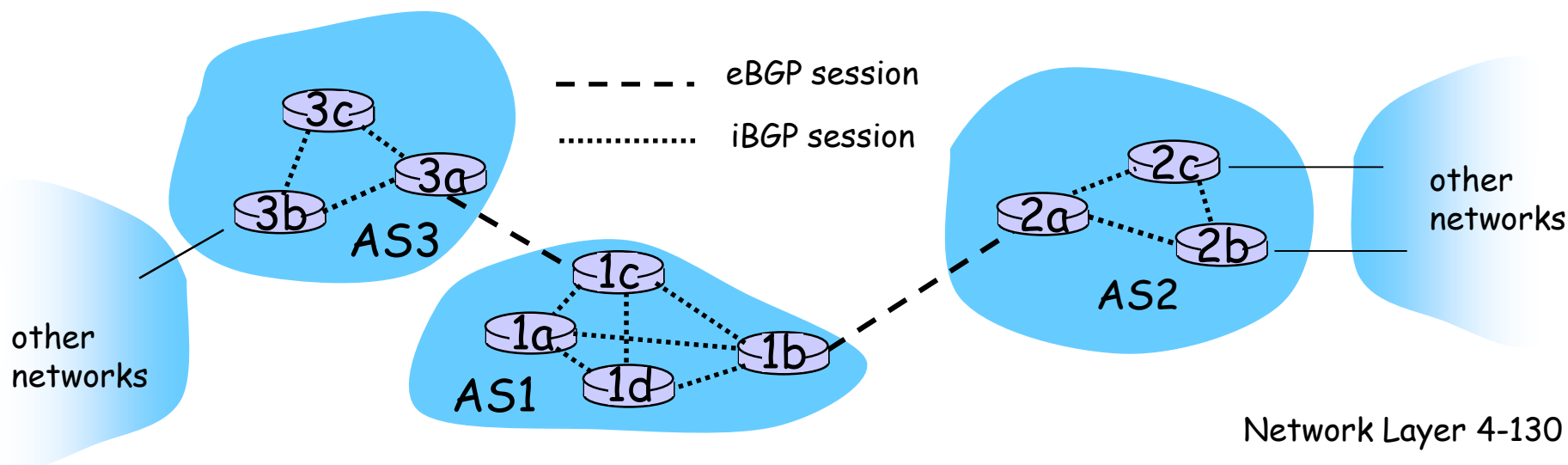
BGP 基础

- ❑ 路由器对 (BGP peers) 使用179端口的TCP连接交换选路信息:
 - BGP sessions 不总是与物理链路对应.
- ❑ BGP使每个AS知道经过其相邻AS, 哪些目的地是可到达的。目的地不是主机而是CIDR化前缀:
 - AS 保证前向转发这些前缀可达性信息.
 - AS 在他的通告里面能够聚合这些前缀.



分布式可达信息

- 在3a和1c之间使用eBGP会话，AS3向AS1发送一个自AS3可达的前缀列表；AS1向AS2发送一个经AS1可达的前缀列表
 - 1c 使用iBGP 向AS1内部路由转发前缀可达信息
 - 1b 向AS2，通过1b-to-2a eBGP session通告新的可达性信息
- 当路由器得知一个新前缀时，它为该前缀在其转发表中创建一个项.



BGP 路由&路径属性

- ❑ 路由器通告前缀时，包括一些 **BGP**属性。
 - **prefix + attributes = "route"**
- ❑ 两种最重要的属性：
 - **AS-PATH**: 包含了前缀通告已经经过的那些**AS**,
用于检测和防止循环通告。
 - **NEXT-HOP**: 是一个开始某**AS-PATH**的路由器接口。
- ❑ 当网关路由器接收到路由器通告时, 使用输入策略来决定是否接受或者过滤该路由。

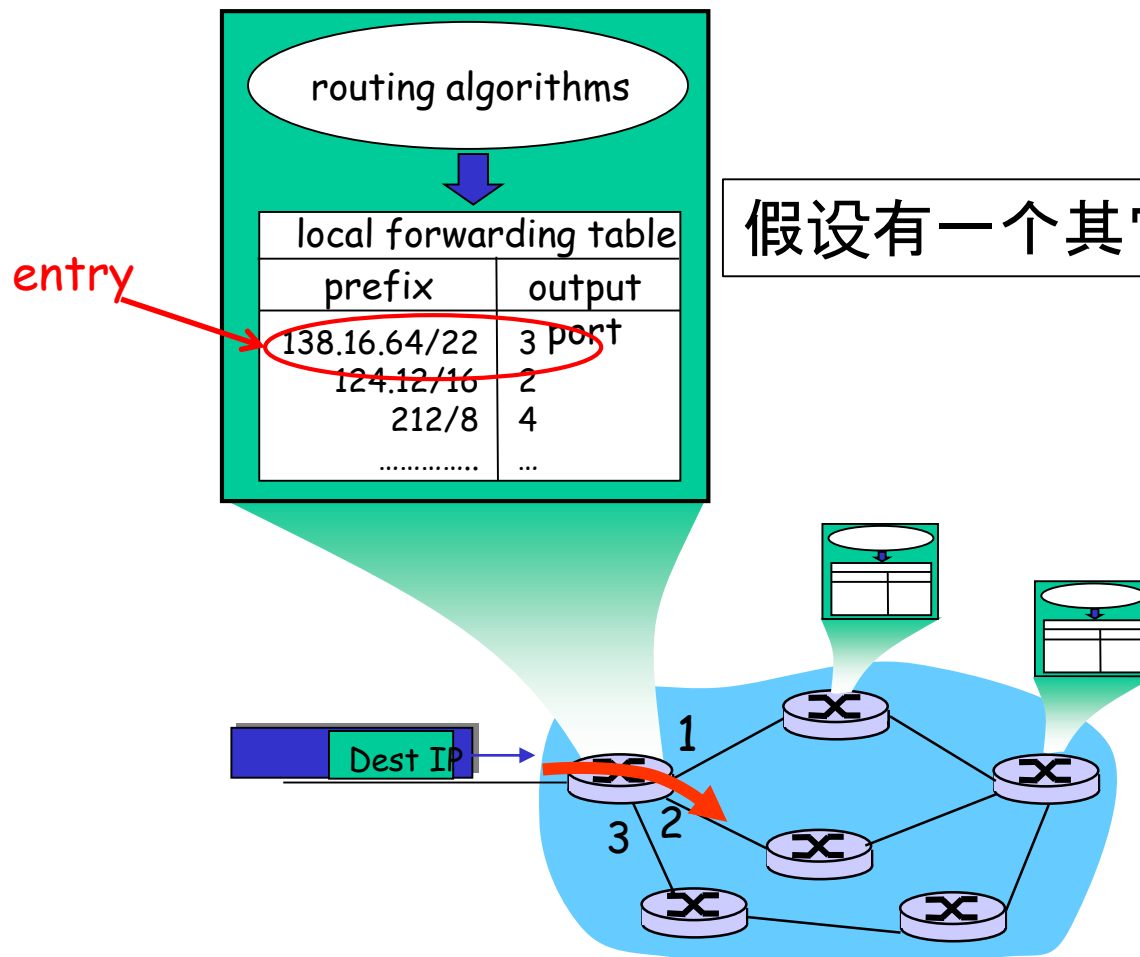
BGP 路由选择

- 路由器可能知道到达任何一条前缀的多条路由. 路由器必须在可能的路由中选择一条:
- 消除规则:
 1. 路由被指派一个本地偏好值作为它们的属性之一
 2. 在余下路由中选择具有最短 **AS-PATH** 的路由
 3. 选择最靠经 **NEXT-HOP** 的路由
 4. 使用**BGP**标识符来选择路由

BGP 消息

- ❑ BGP 使用 TCP 交换消息.
- ❑ BGP 消息:
 - OPEN: 开始 TCP 连接并认证;
 - UPDATE: 广播新路径 (撤回旧的路径);
 - KEEPALIVE 中update间隔期保持连接活性, 也用于ACKs OPEN 请求;
 - NOTIFICATION: 报告前述消息错误, 也用于关闭连接。

转发表的构建?

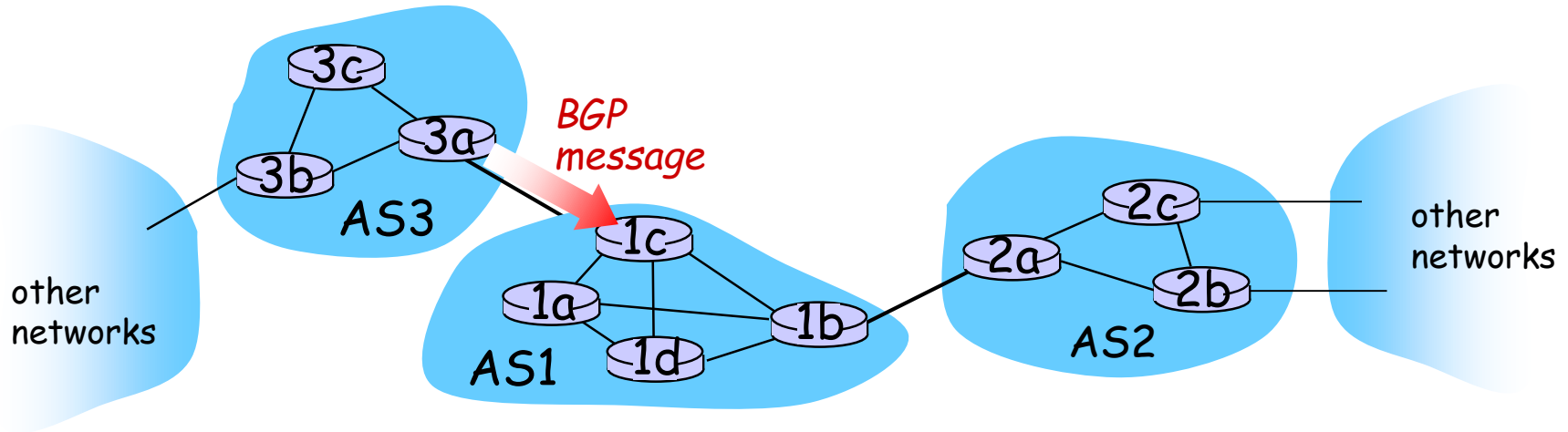


转发表的构建?

High-level overview

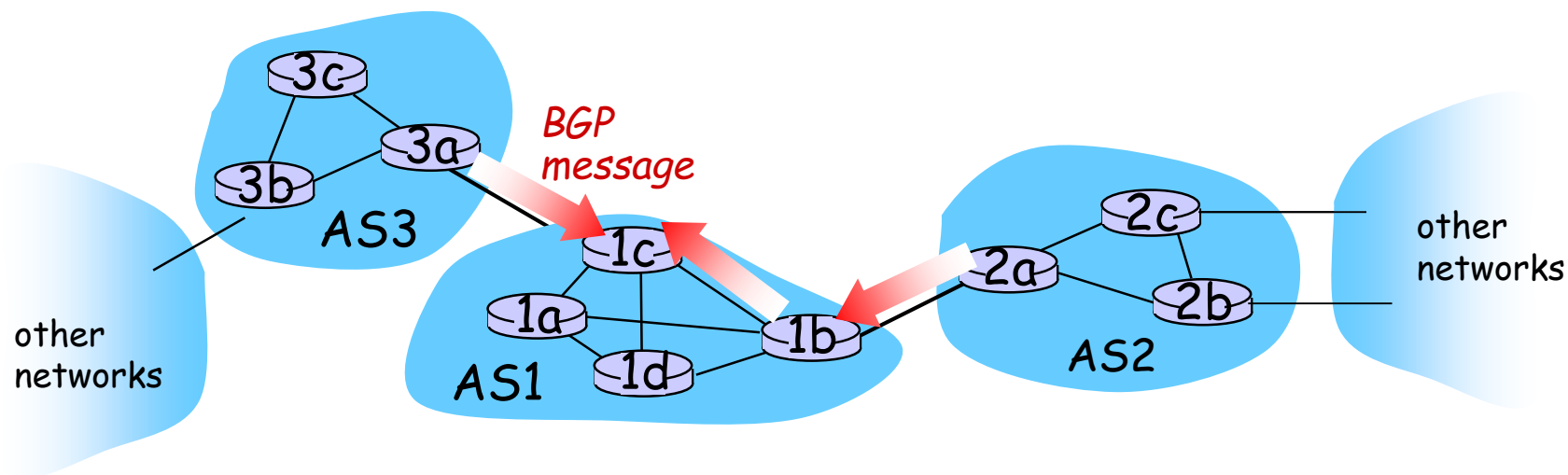
1. 路由了解到前缀
2. 路由确定该前缀的输出端口
3. 路由在转发表中插入前缀-端口条目

路由接收前缀



- ❖ BGP 包含了“路径”
- ❖ “路径” 是一个前缀(prefix)和属性(attributes):
AS-PATH, NEXT-HOP,...
- ❖ Example: route:
 - ❖ Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ;
NEXT-HOP: 201.44.13.125

路由可能接收到多条路径

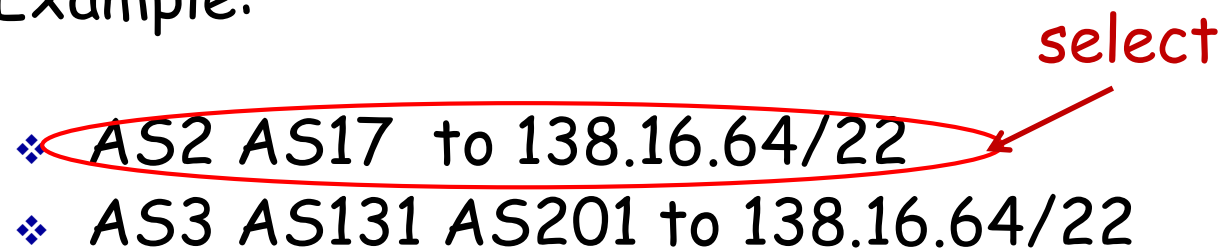


- ❖ 路由可能接收到关于同一个前缀的多条路径
- ❖ 必须选择其中一条

选择前往前缀最好的BGP路由

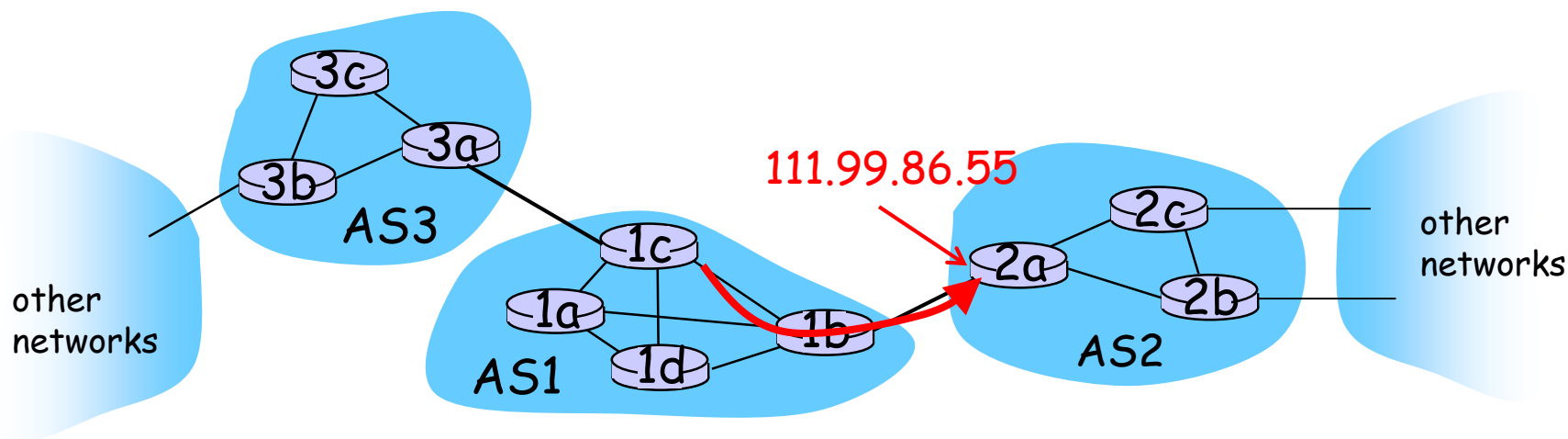
□ 从最短的AS-PATH中选择路由

❖ Example:

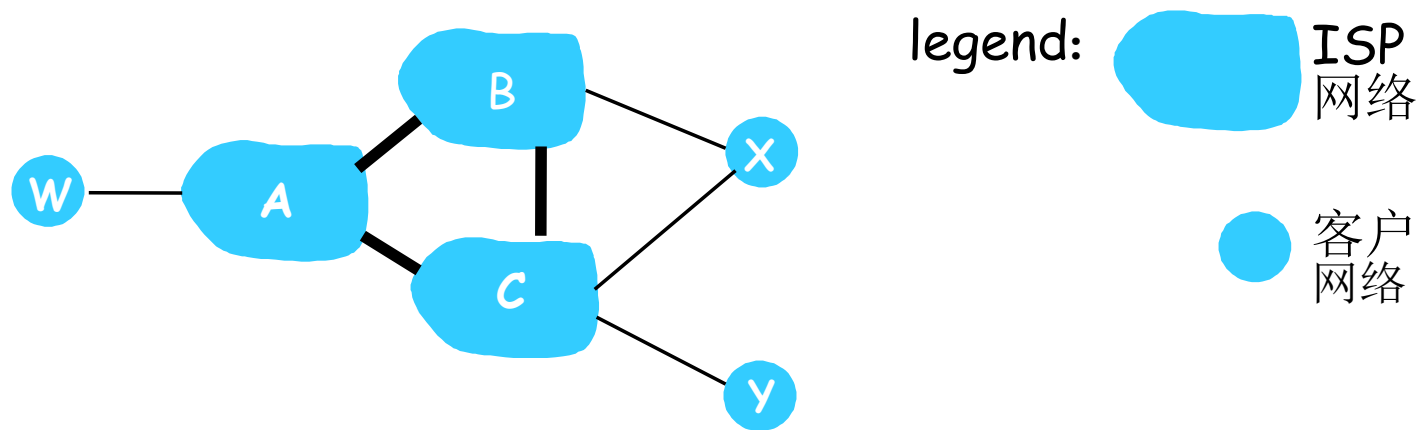
- ❖ AS2 AS17 to 138.16.64/22
 - ❖ AS3 AS131 AS201 to 138.16.64/22
- select
- 

选择intra-路径前往BGP 路由

- 根据路由的NEXT-HOP 属性
 - 路由NEXT-HOP 包含AS PATH路由接口的IP地址.
- Example:
 - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- 路由使用OSPF 发现从1c 到 111.99.86.55的最短路径

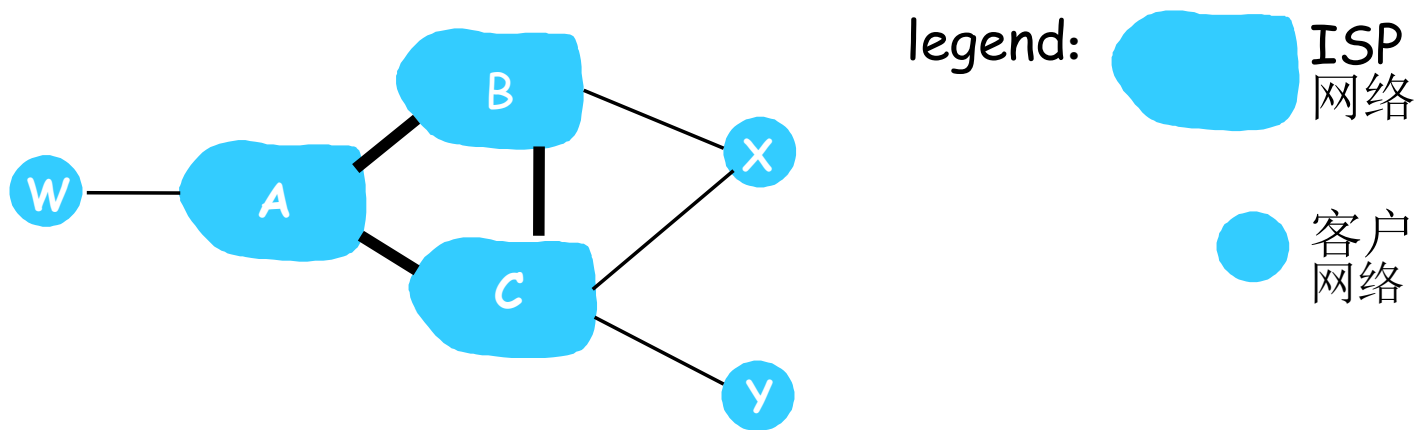


BGP 路由策略



- A,B,C ISP 网络
- X,W,Y 是客户(ISP网络)
- X 是多宿接入ISP: 与两个网络相连
 - X 当然不希望B 通过 X 路由到 C
 - .. 所以 X 向B 广播其没有到C的路由

BGP路由策略(2)-到w的路径



- ❑ A 广播路径 **AW** to B
- ❑ B 广播路径 **BAW** to X
- ❑ B 应该广播路径 **BAW** to C?
 - No way! B 认为确保C经过A和C之间连接引导A客户的流量，是A和C的工作，不管它的事。
 - B 仅希望为其客户进行路由!

为什么使用不同的 Intra- 和 Inter-AS 路由?

Policy:

- ❑ Inter-AS: 管理员希望控制网络流动路径, 以及哪些流量经过网络.
- ❑ Intra-AS: 单个域内, 不需要采取什么策略.

Scale:

- ❑ 层次路由减少路由表条目数量, 减少路由更新流量

Performance:

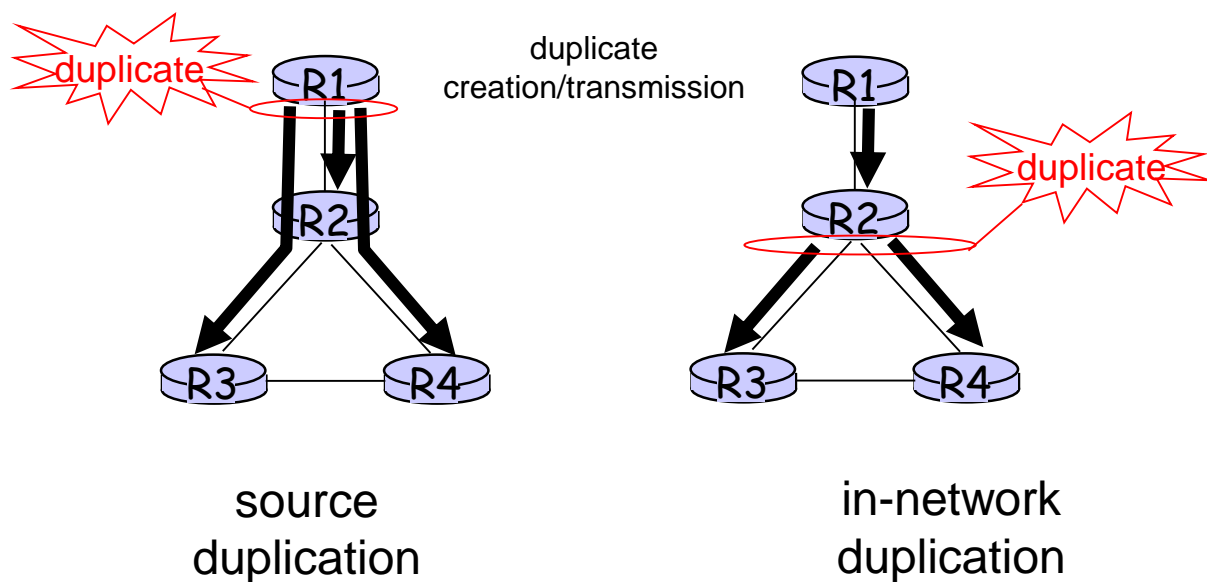
- ❑ Intra-AS: 着眼于性能
- ❑ Inter-AS: 策略更重要

Chapter 4: Network Layer

- ❑ 4.1 概览
- ❑ 4.2 虚电路和数据报网络
- ❑ 4.3 路由结构
- ❑ 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- ❑ 4.5 路由算法
 - Link state
 - Distance Vector
 - Hierarchical routing
- ❑ 4.6 Internet中的路由
 - RIP
 - OSPF
 - BGP
- ❑ 4.7 广播和多播路由
 - Broadcast and multicast routing

广播路由

- ❑ 从源节点到网络中所有其他节点交付分组的服务
- ❑ 发送节点向目的地分别发送拷贝：



- ❑ 低效；
如何知道目的地的地址？

广播

❑ 泛洪(flooding):

泛洪要求源节点向它所有邻居发送分组的拷贝

- Problems: 循环 & 广播风暴

❑ 受控泛洪(controlled flooding):

受控泛洪选择何时泛洪，何时不泛洪；

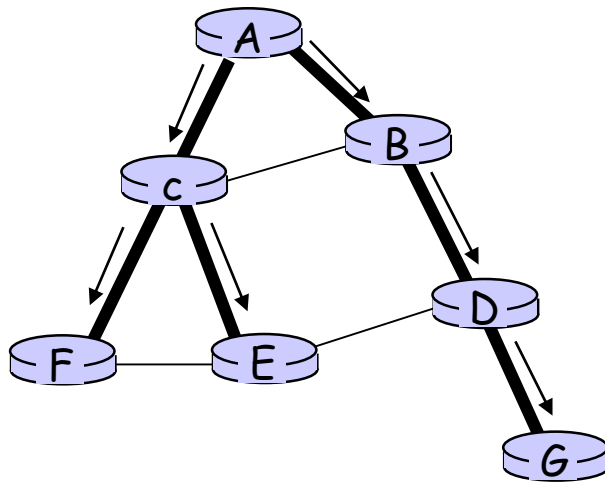
- 序号控制泛洪：将其地址以及广播序号放入分组，再向其邻居发送分组
- 反向路径转发泛洪 (RPF): 仅当自己位于它自己到其源的最短单播路径上，才向其所有出链接广播分组

❑ 生成树广播(spanning tree):

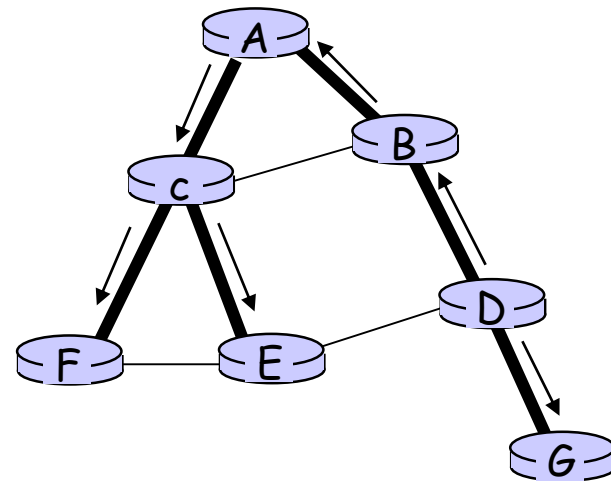
- 任何节点都不会收到冗余的广播数据

生成树(Spanning Tree)

- 构造一棵生成树
- 节点延生成树发送分组



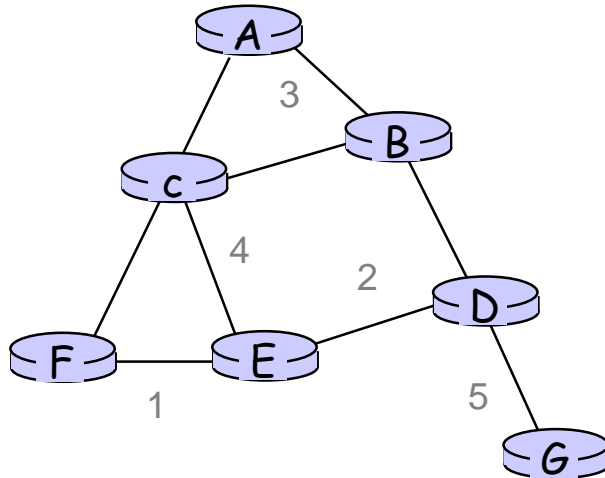
(a) Broadcast initiated at A



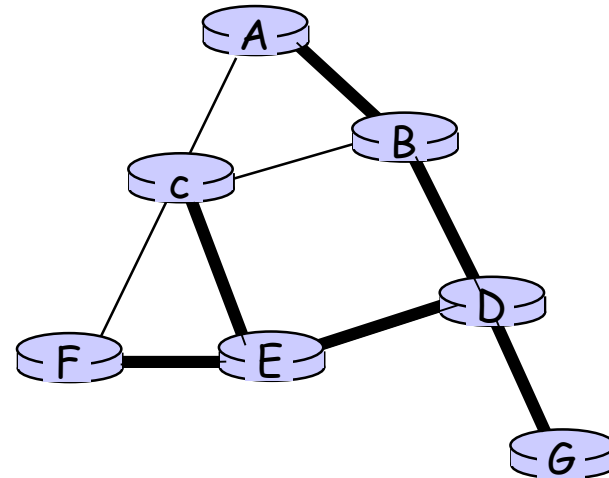
(b) Broadcast initiated at D

生成树: 构造

- 中心节点
- 所有节点向中心节点单播加入树报文
 - **Message** 使用单播选路朝着中心节点转发, 知道它到达一个已经属于生成树的节点或中心节点



(a) Stepwise construction of spanning tree



(b) Constructed spanning tree

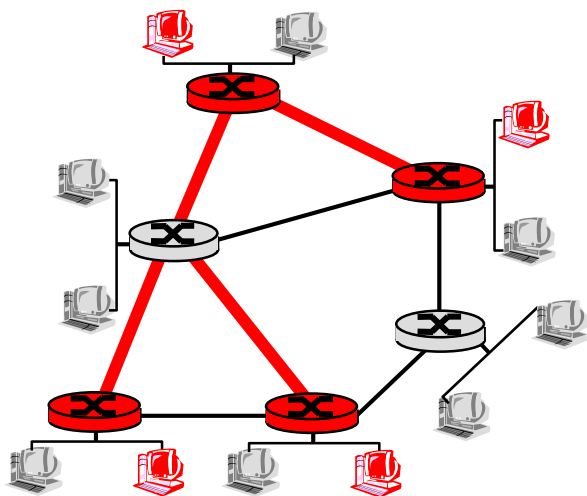
多播

□ Goal:

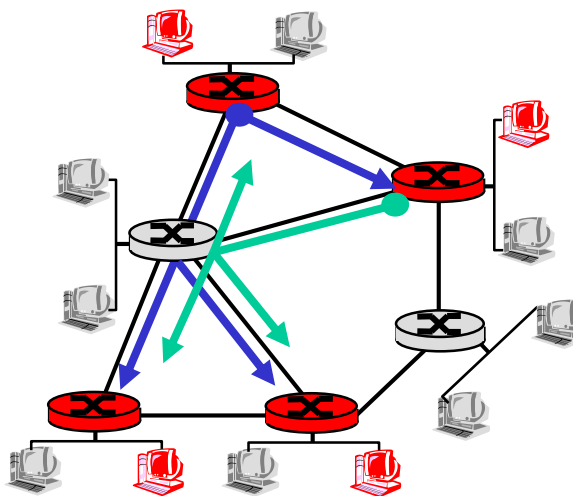
多播分组仅交付给网络节点的一个子集。

□ 互联网组管理协议

□ 多播选路算法



Shared tree



Source-based trees

多播选路算法

策略:

- ❑ 基于源的树(**source-based tree**):
 - shortest path trees
 - reverse path forwarding
- ❑ 基于组共享的树(**group-shared tree**):
 - minimal spanning (Steiner)
 - center-based trees