# 第4章 网络层：数据平面

**目的:**

- 了解网络层服务背后的原则，专注于数据平面：
  - 网络层服务模型
  - 转发 vs 路由
  - 路由器的工作原理
  - 网际协议
- 因特网的网络层的实现

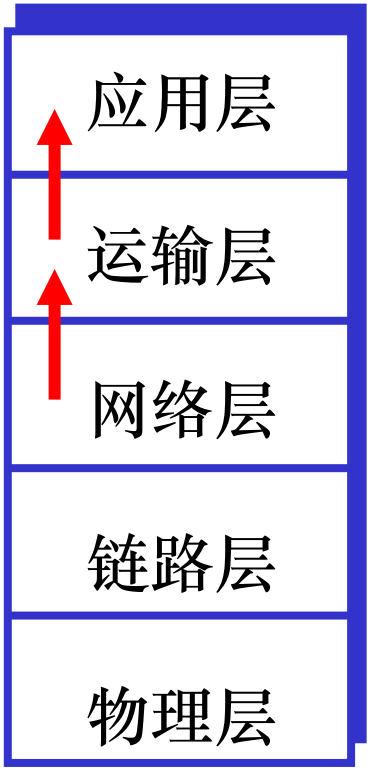
# 第4章 主要内容

- 4.1 网络层概述
  - 转发和路由选择：数据平面和控制平面
  - 网络服务模型
- 4.2 路由器工作原理
  - 输入端口和基于目的地的转发
  - 交换
  - 输出端口处理
  - 何处出现排队
  - 分组调度
- 4.3 网际协议：IPv4、寻址、IPv6
  - IPv4数据报格式
  - IPv4数据报分片
  - IPv4编址
  - 网络地址转换(NAT)
  - IPv6

# 4.1 网络层概述

- 运输层和网络层提供服务的区别
  - 运输层为主机上的两个进程之间提供通信服务；
  - 运输层只在网络的主机中出现；
  - 网络层提供的主机到主机的通信服务；
  - 网络层在网络的主机和路由器中出现。

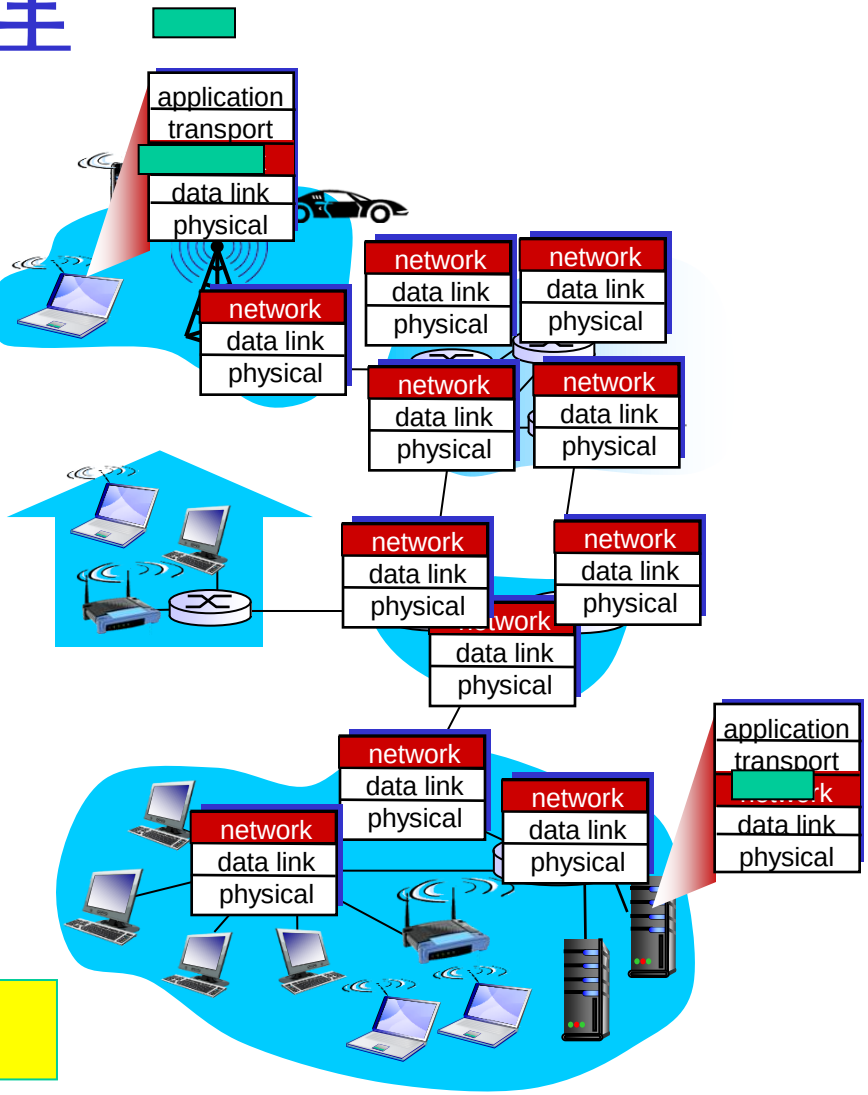
本章主要讨论网络层如何实现主机到主机的通信服务



# 4.1.1 转发和路由选择

- 传输报文段(segment)从发送主机到接收主机
- 发送方封装报文段(segments)为数据报(datagrams)
- 接收方递交数据报给运输层
- 网络层协议在每个主机(host)和路由器(router)上实现
- 路由器检查所有IP数据报的头部字段，根据目的IP地址转发数据报

**通常路由器只实现下三层部分**



# 路由器的两个核心功能

*网络层功能：*

- 转发(forwarding)：将分组(packet)从路由器的输入端口转移到输出端口
- 路由(routing)：确定分组从发送方到接收方的路由(route)
  - 路由算法

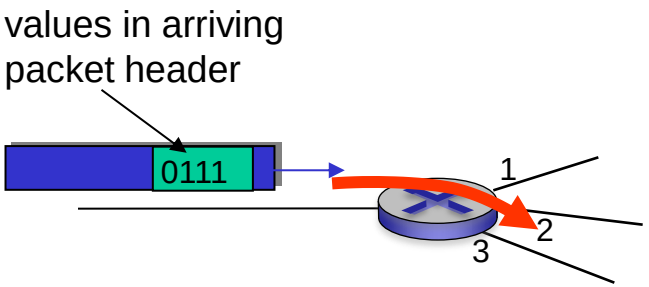
*类比：taking a trip*

- 路由器的转发过程类比于通过立交桥；单个路由器
- 路由过程类比于从电子科大沙河校区开车到清水河校区的旅程，需要通过若干道路和立交桥（确定怎么走的线路）；网络中的所有路由器，集体经选路协议交互，决定分组从源到目的地的路径。

# 数据平面和控制平面

## *数据平面*

- 本地、每个路由器的功能
- 将数据报从路由器的输入端口转发到路由器的输出端口
- forwarding功能

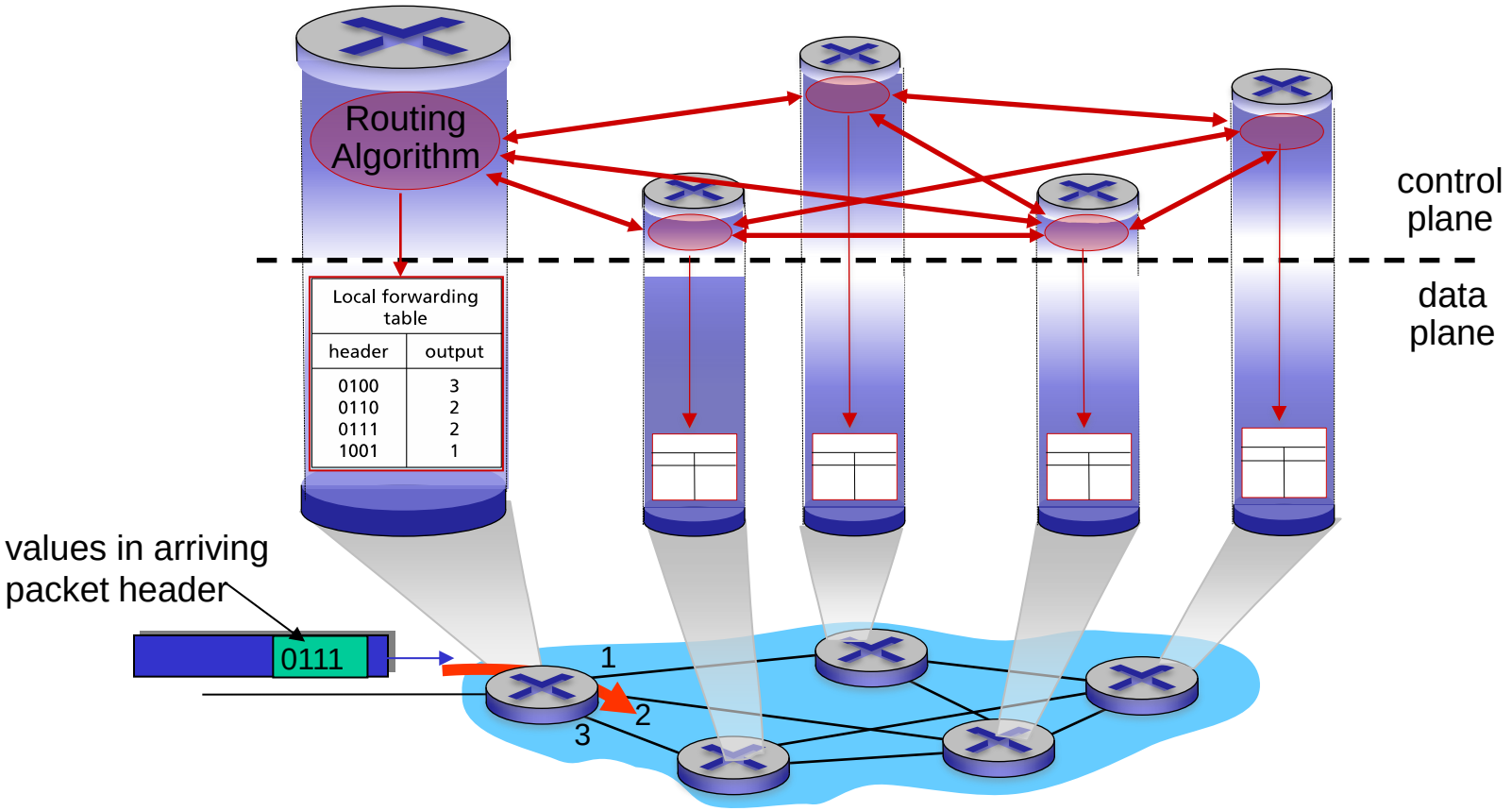



## *控制平面*

- 网络范围的逻辑
- 确定数据报从发送主机到接收主机的端到端路径上通过的路由器是哪些
- 两个控制平面的方案：
  - 传统路由算法：在路由器中实现
  - 软件定义网络(software-defined networking，SDN)：在远程的服务器上实现

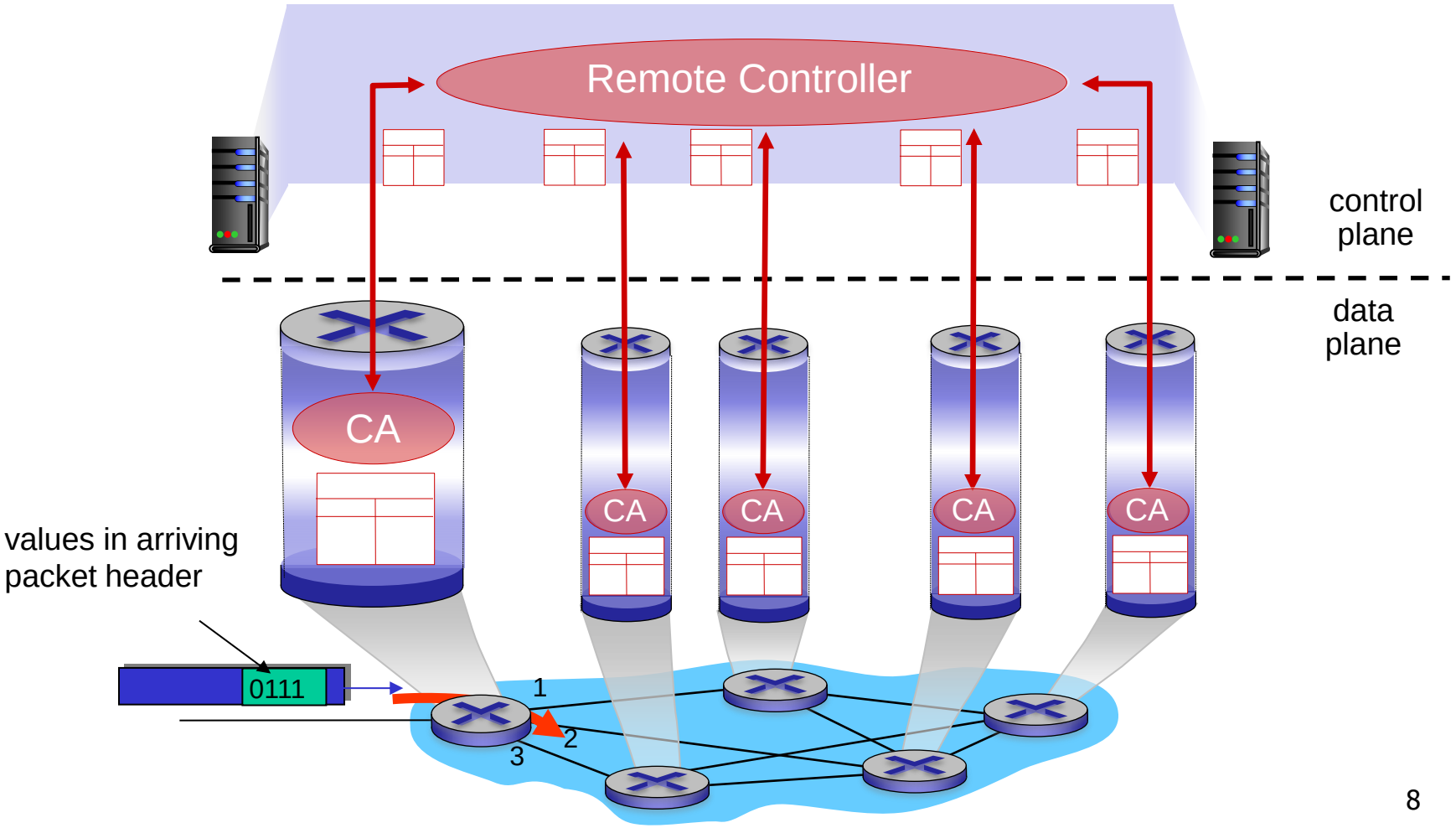
# 每个路由器的控制平面

每个路由器中的各个路由算法组件在控制平面中交互



# 逻辑上集中式的控制平面

一个特别的(通常是远程的)控制器与本地控制代理(CA)交互

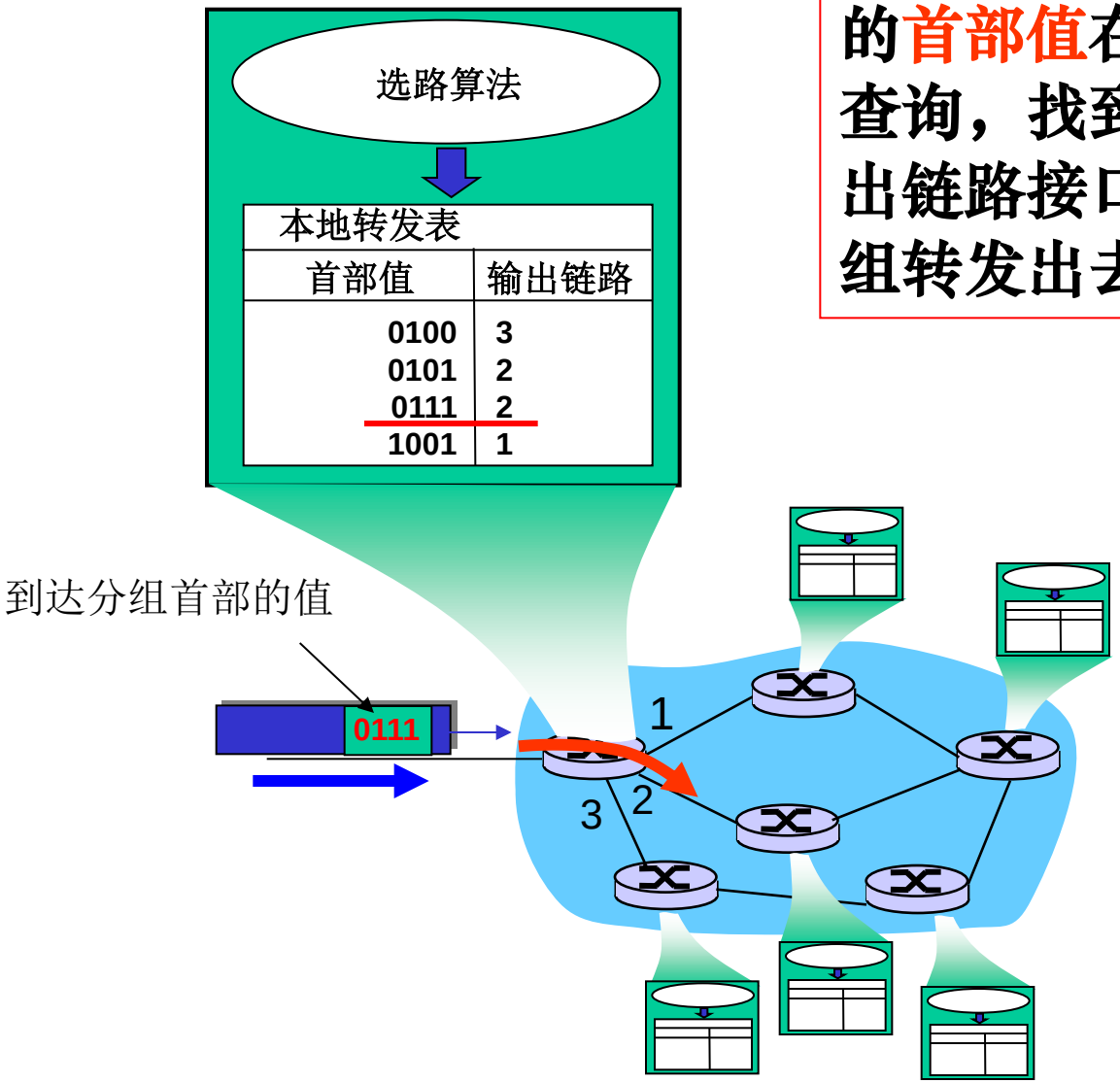


# 路由器如何转发分组

- 转发表：每台路由器有一张转发表
  - 分组首部(目的地址或某个连接标识)和相应输出链路的对照表;
  - 转发表的内容由选路算法决定(集中式或分布式)。
- 分组交换机(packet switcher)：一台分组交换设备，根据分组首部值，从输入链路接口到输出链路接口传送分组。
  - 链路层交换机：根据链路层字段值作转发决定的分组交换机。
  - 路由器：根据网络层字段值作转发决定的分组交换机。

# 转发方法

**路由器根据到达分组的首部值在转发表中查询，找到相应的输出链路接口，并将分组转发出去。**



# 讨论：是否在网络层建立连接

- 运输层连接：如TCP协议，在数据实际传输之前，需要发送方和接收方经过三次握手建立所需的状态信息。
- 网络连接：指网络层数据分组开始传输前，在所选择的从源到目的地路径上的各路由器之间相互握手，建立连接状态。
- 在网络层建立连接的例子：ATM、帧中继、MPLS
- 因特网的网络层不建立连接！路由器上不需要维护状态

# 4.1.2 网络服务模型

**问题:**什么样的服务模型可以用于将数据报从发送方传输到接收方?

- 网络层可能提供的服务
  - 确保交付：确保分组到达目的地。
  - 具有时延上界的确保交付：主机到主机的时延。
  - 有序分组交付：按发送顺序到达。
  - 确保最小带宽：当发送主机以低于特定比特率的速率发送比特，分组不会丢失，在一定时延到达。
  - 确保最大时延抖动：发送方发送两个连续分组的时间间隔与接收到的间隔相同。
- 因特网的网络层提供的服务
  - 单一服务，即尽力而为服务(best-effort service) 。
  - 分组间的定时不能被保证;
  - 分组的接收顺序与发送顺序不一定相同;
  - 传送的分组不能保证最终交付，即网络可能未向目的地交付分组。

# 其他网络服务模型

- ATM网络体系结构提供了确保按序时延、有界时延和确保最小带宽。
- 因特网的基本尽力而为服务模型与适当带宽供给相结合已经被证明超过“足够好”，能够满足各种网络应用。
- ATM是基于虚电路网络，在发送数据前需要在网络层建立连接(路径上的路由器之间)；[虚电路可参考本书第6版]
- 因特网是基于数据报网络，在发送数据前不需要在网络层建立连接。

# 虚电路网络

- 数据报网络提供网络层的 无连接 服务
- 虚电路网络提供网络层的 连接 服务
- 任何网络中的网络层只提供两种服务之一，不会同时提供
  - 虚电路网络：提供连接服务。
  - 数据报网络：提供无连接服务。
- 传输层的面向连接服务是在网络边缘的**端系统**中实现
- 网络层的面向连接服务是在**端系统及路由器**中实现。

# 虚电路(Virtual Circuits)

“源主机-目的主机路径的行为类似于电话网络的行为”

- 性能上类似
- 沿着源-目的路径的网络行为类似

- 在数据传输之前，需要为每个呼叫建立连接
- 每个分组携带VC标识符(不是目的主机地址)
- 位于“源-目的路径”上的每个路由器会维护经过它的每条连接的“状态”
- 链路和路由器的资源(带宽、缓存)可以被分配给VC(专用资源)

# 虚电路的实现

- 一条虚电路(VC)包括：
  - 一条从发送方到接收方的**路径；**
  - VC号，沿路径的每条链路都有一个**VC号；**
  - 沿路径的每个路由器上都有**转发表条目；**
- 属于同一个VC的分组携带相同的VC号(而不是目的地址)
- 在每条链路上VC号可以被修改
  - 新的VC号来自于转发表

# 主机A与主机B通信过程

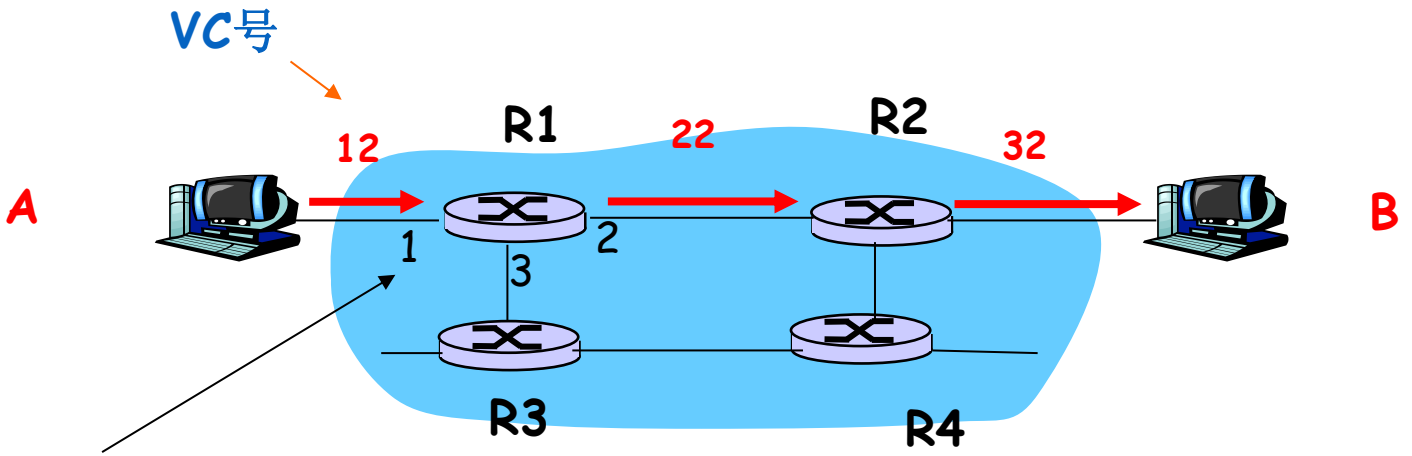
- 主机A与主机B之间创建一条VC：

路径为A → R1 → R2 → B，3条链路分配VC号12、22和32

- 传输时，分组VC号变化过程：

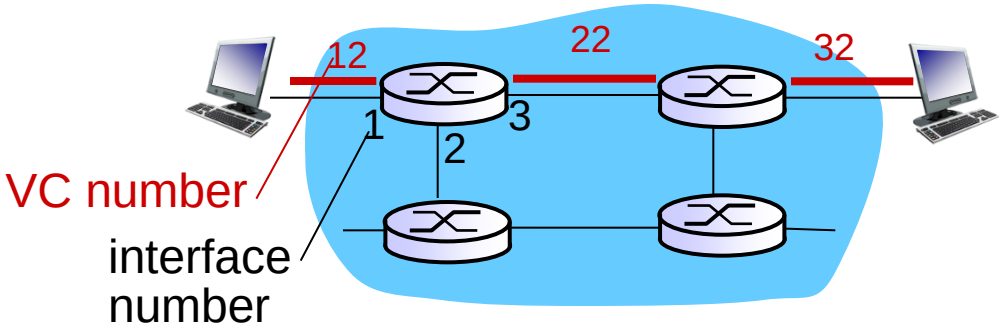
12　　22　　32

A ⟶ R1 ⟶ R2 ⟶ B




**接口号: 路由器连接链路的接口号码**

# VC转发表



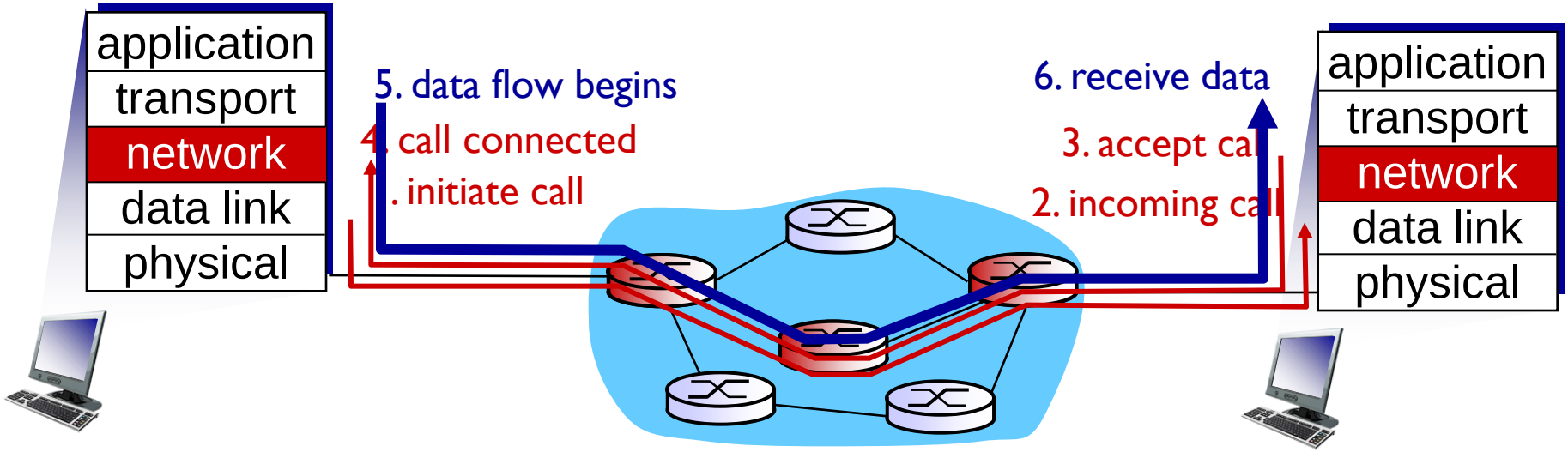

*forwarding table in northwest router:*

| Incoming interface | Incoming VC # | Outgoing interface | Outgoing VC # |
|---|---|---|---|
| 1 | 12 | 3 | 22 |
| 2 | 63 | 1 | 18 |
| 3 | 7 | 2 | 17 |
| 1 | 97 | 3 | 87 |
| … | … | … | … |

*VC routers maintain connection state information!*

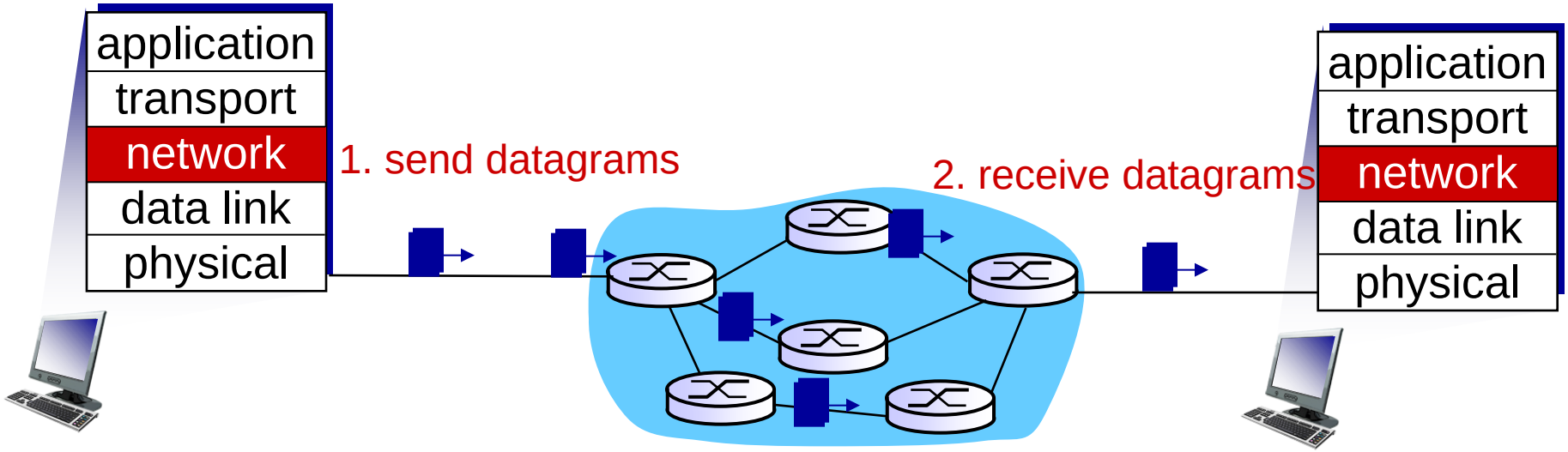
# 虚电路：信令协议

- 用于建立，维护，拆除虚电路
- 用于ATM, frame-relay, X.25网络
- 不能用于目前的因特网



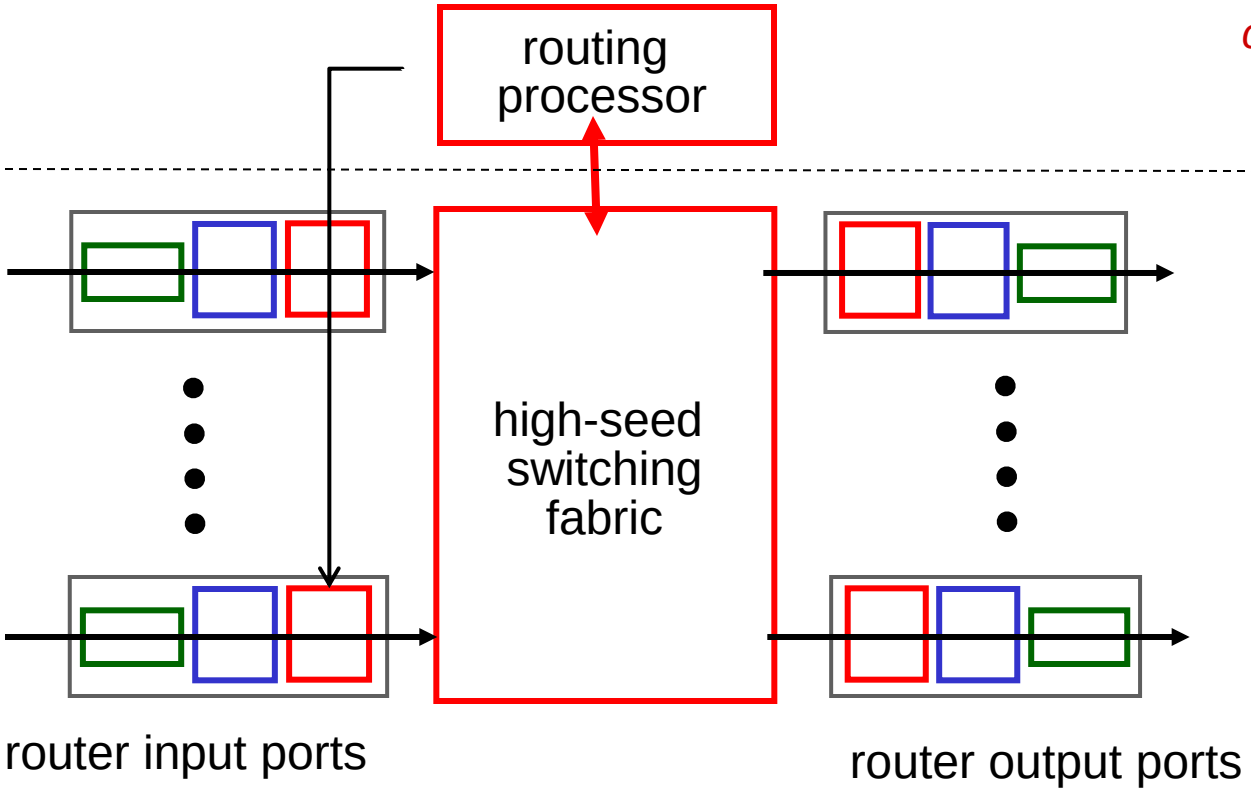
# 数据报网络

- 在网络层无呼叫的过程
- 路由器： 不需要维护端到端连接的状态
- 没有网络等级的“连接”的概念
- 使用目的主机的地址进行分组转发



# 4.2 路由器工作原理

- 路由器的两个核心功能：
  - 运行路由算法/协议(RIP, OSPF, BGP)
  - 将分组从路由器的输入链路传送到正确的输出链路。
- 路由器的体系结构：

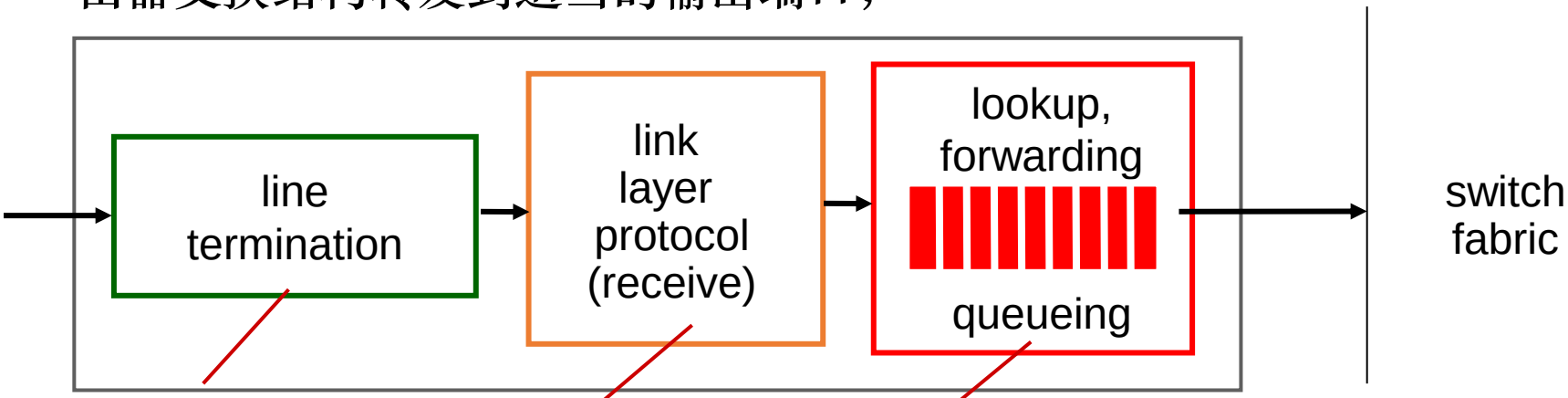



*routing, management control plane* (software) operates in millisecond time frame

*forwarding data plane* (hardware) operttes in nanosecond timeframe

# 4.2.1 输入端口处理和基于目的地转发

- 第一个线路端接模块：将一条物理链路端接到路由器的物理层；
- 第二个数据链路处理模块：实现路由器的数据链路层功能；
- 第三个查找与转发模块：实现查找与转发功能，以便分组通过路由器交换结构转发到适当的输出端口；



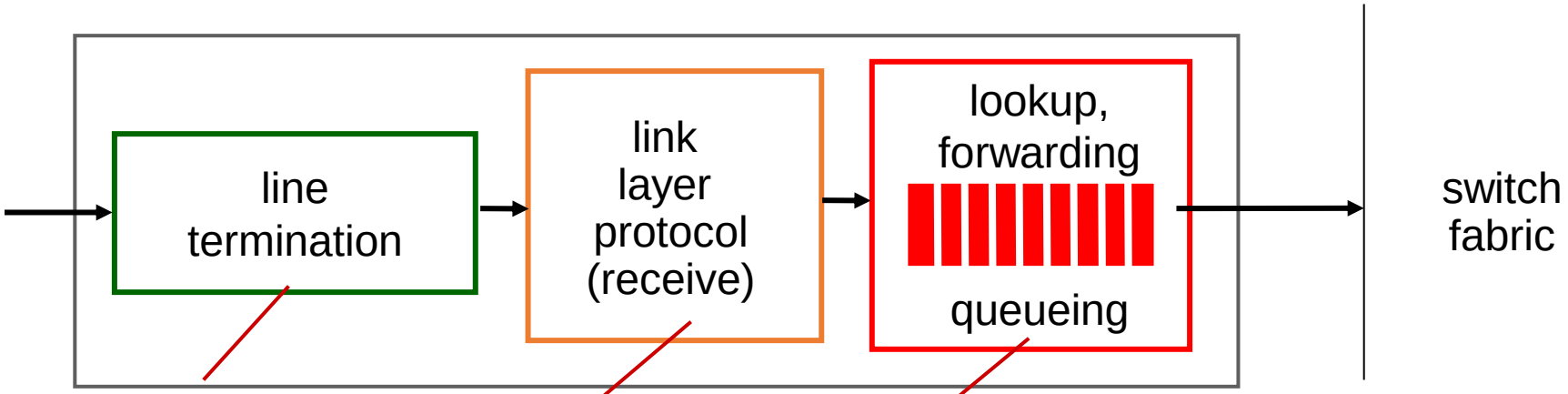

物理层:
比特等级的接收

数据链路层:
解封装进行分析
e.g., Ethernet

*分布式交换:*

- 根据数据报的头部字段的值，使用输入端口内存中缓存的转发表查找输出端口
- 目标：以线速完成输入端口处理
- 排队：当数据报到达的速率快于转发到交换结构的速率

# 4.2.1 输入端口处理和基于目的地转发




物理层:
比特等级的接收

数据链路层:
解封装进行分析
e.g., Ethernet

*分布式交换:*

- 根据数据报的头部字段的值，使用输入端口内存中缓存的转发表查找输出端口
- 基于目的IP地址的转发（传统）
- 通用转发：基于任意设置的头部字段的值的组合

# 基于目的IP地址的转发

- 根据分组的目的IP地址在转发表中查询，找到相应的输出链路接口，并将分组转发出去
- 转发表：每台路由器维护一张路由表，目的地址与链路接口的映射表。
  - 转发表中的表项数与地址位数有关，如果每个可能的IP地址对应一项，转发表的条目可以达到40亿条(目的地址32位)

# 转发表

| **Destination Address Range** | Link Interface |
|---|---|
| 11001000 00010111 00010000 00000000<br>through<br>11001000 00010111 00010111 11111111 | 0 |
| 11001000 00010111 00011000 00000000<br>through<br>11001000 00010111 00011000 11111111 | 1 |
| 11001000 00010111 00011001 00000000<br>through<br>11001000 00010111 00011111 11111111 | 2 |
| otherwise | 3 |

问题: 如果地址范围分配没有很合理，会发生什么问题？

# 最长前缀匹配规则

*longest prefix matching*

对于给定的目的地址，使用最长地址前缀匹配来完成输出端口的查找

| Destination Address Range | Link interface |
|---|---|
| 11001000 00010111 00010*** ********* | 0 |
| 11001000 00010111 00011000 ********* | 1 |
| 11001000 00010111 00011*** ********* | 2 |
| otherwise | 3 |

例子:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

# 路由器查表方法

| 前缀匹配 | 链路接口 |
| --- | --- |
| 11001000 00010111 00010 | 0 |
| 11001000 00010111 00011000 | 1 |
| 11001000 00010111 00011 | 2 |
| 其他 | 3 |

用目的地址前缀与转发表的前缀匹

- 存在匹配：向对应链路转发。

如，目的地址

11001000 00010111 00010110 10100001 → **0#**

- 不存在匹配：选择“其他”项对应的链路转发。
- 存在多个匹配：使用最长前缀匹配规则，即向与最长前缀匹配的链路接口转发分组。

如，目的地址

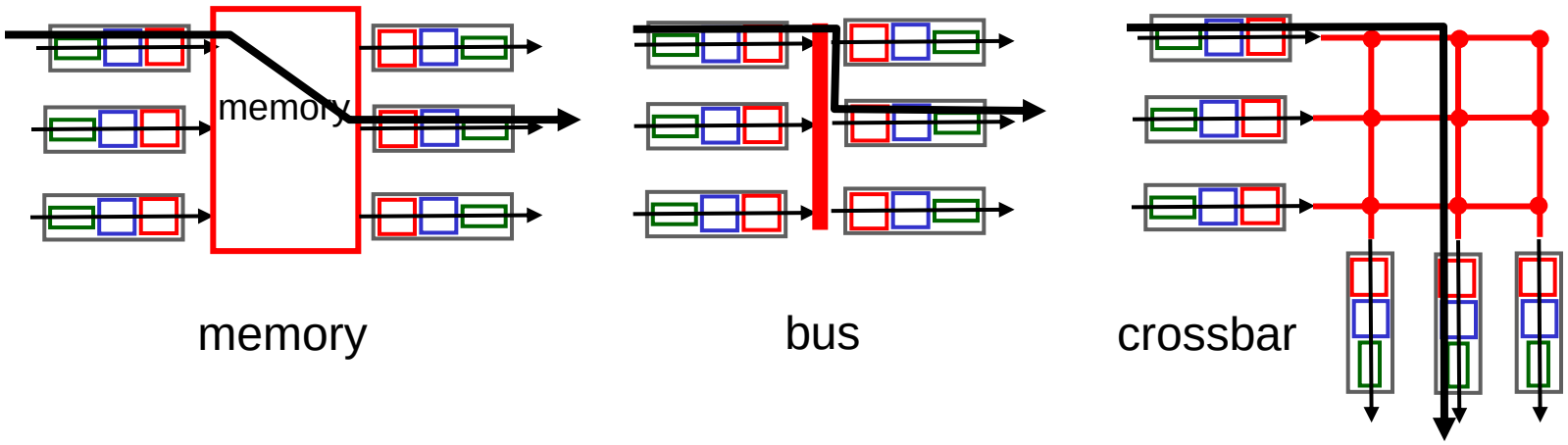
11001000 00010111 00011000 10101010 → **1#**

# 4.2.1 输入端口处理和基于目的地转发

- 确定将一个到达的分组通过交换结构转发给哪个输出端口。 通过查找转发表实现，这里的转发表是存储在输入端口的内存中。
- 分布式交换：
  - 选路处理器计算转发表，给每个输入端口存放一份转发表拷贝。
  - 在每个输入端口本地做出交换决策，无须激活中央选路处理器。
  - 可避免在路由器中某个单点产生转发处理瓶颈。
  - 目的：以线速完成输入端口的处理
  - 排队：如果数据报到达输入端口的速度快于输入端口将数据报转发到交换结构的速度，就会发生排队

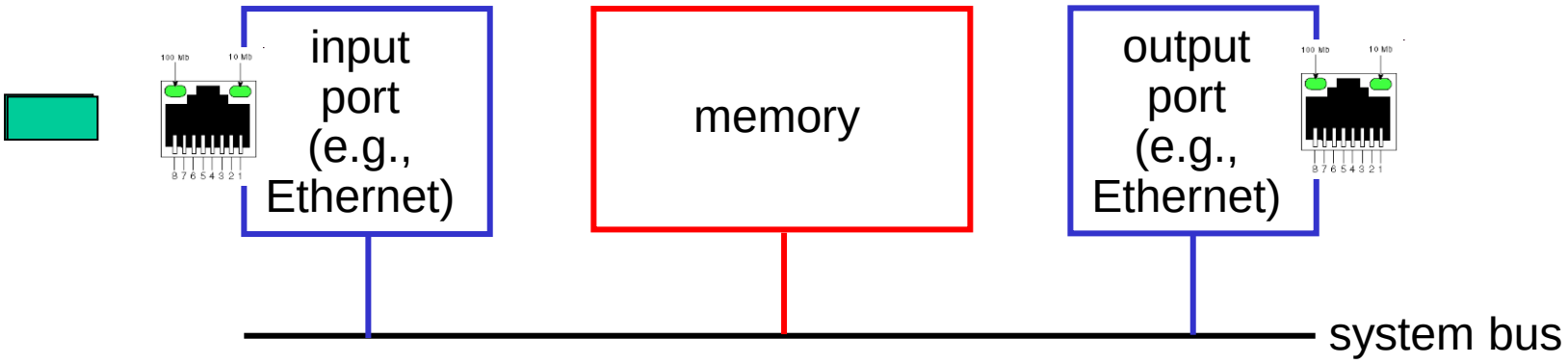
# 4.2.2 交换结构

- 将分组从输入端口缓存交换（转发）到恰当的输出端口缓存中
- 交换速率：分组从输入端口到输出端口的速率
  - 通常用输入/输出线速的倍数来表示
  - N inputs: switching rate N times line rate desirable
- 三种类型的交换结构



# 经内存交换

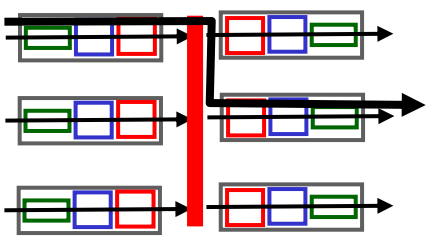
- 早期用计算机作为路由器时采用的结构(第一代)
  - 输入端口与输出端口之间的交换由CPU(选路处理器)控制完成；
  - 输入端口与输出端口类似I/O设备：
    - 当分组到达输入端口时，通过中断向选路处理器发出信号，将分组拷贝到处理器内存中；
    - 选路处理器根据分组中的目的地址查表找出适当的输出端口，将该分组拷贝到输出端口的缓存中。




交换速度受总线带宽的速度限制 (每个分组穿过两次总线)

# 经总线交换

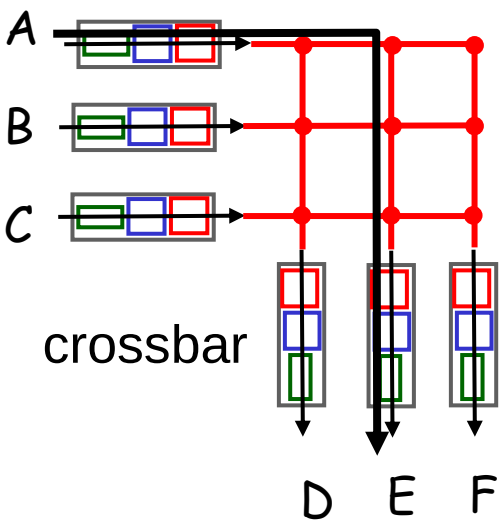
- 输入端口通过一条共享总线将分组直接传送到输出端口，不需要选路处理器的干预
- 每次只能有一个分组通过总线传送，分组可以被总线上的每个端口接收，只有标签正确的端口将其存储；
- 分组到达一个输入端口时，若总线正忙，会被暂时阻塞，在输入端口排队
- 路由器交换带宽受总线速率限制。



bus

# 经互联网络交换

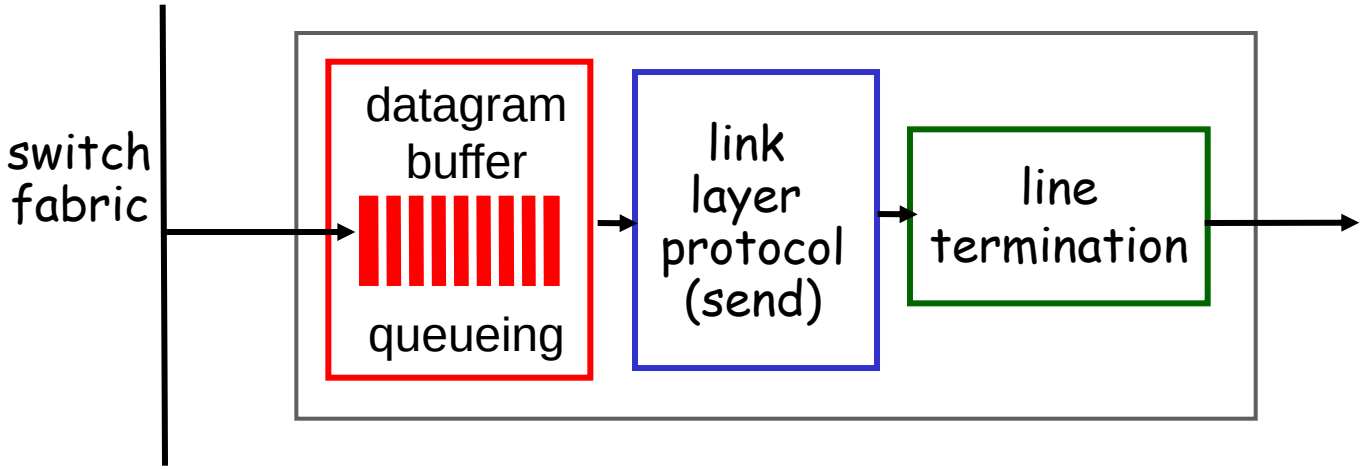
- 纵横式(crossbar)交换机：由2n条总线组成，n个输入端口与n个输出端口连接。
- 到达输入端口的分组沿水平总线穿行，直至与所希望的输出端口的垂直总线交叉点：
  - 若该条垂直总线空闲，则分组被传送到输出端口；
  - 否则，该到达的分组被阻塞，必须在输入端口排队。




从A到E端口转发分组，从B到D端口转发分组可以同时进行，因此速率高于经总线的交换结构

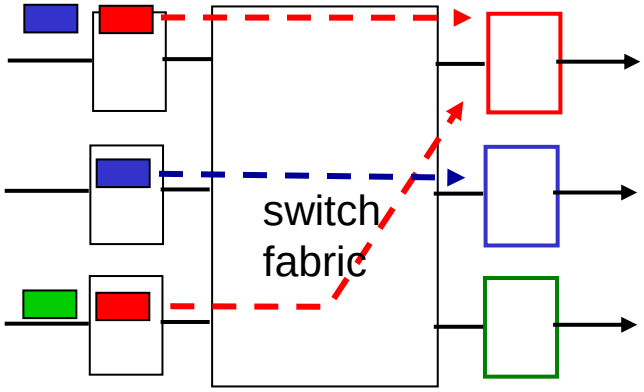
# 4.2.3 输出端口处理

- 取出存放在输出端口内存中的分组，并将其传输到输出链路上。
- 当交换结构将分组交付给输出端口的速率超过输出链路速率，就需要排队与缓存管理功能。当输出端口的缓冲区溢出时，就会出现延时和丢包。

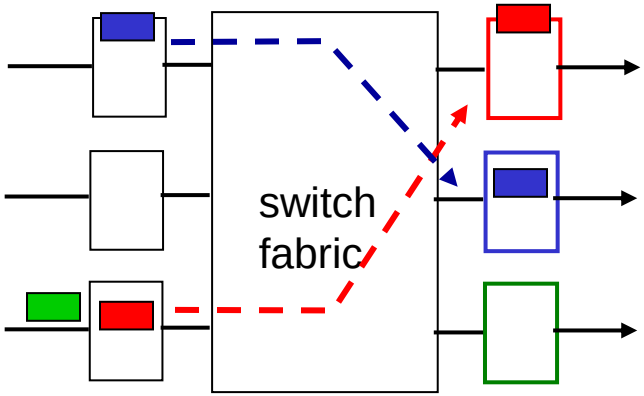


# 4.2.4 何时出现排队

- 输入端口排队
  - 当交换结构的速度慢于输入端口的速度，就会在输入端口的缓冲区发生排队
  - 会导致排队延时和由于输入缓冲区溢出导致的丢包!
- 线头阻塞（Head-of-the-Line (HOL) blocking）：在队列前面的被阻塞的数据报会阻止队列中的其他数据报被转发。



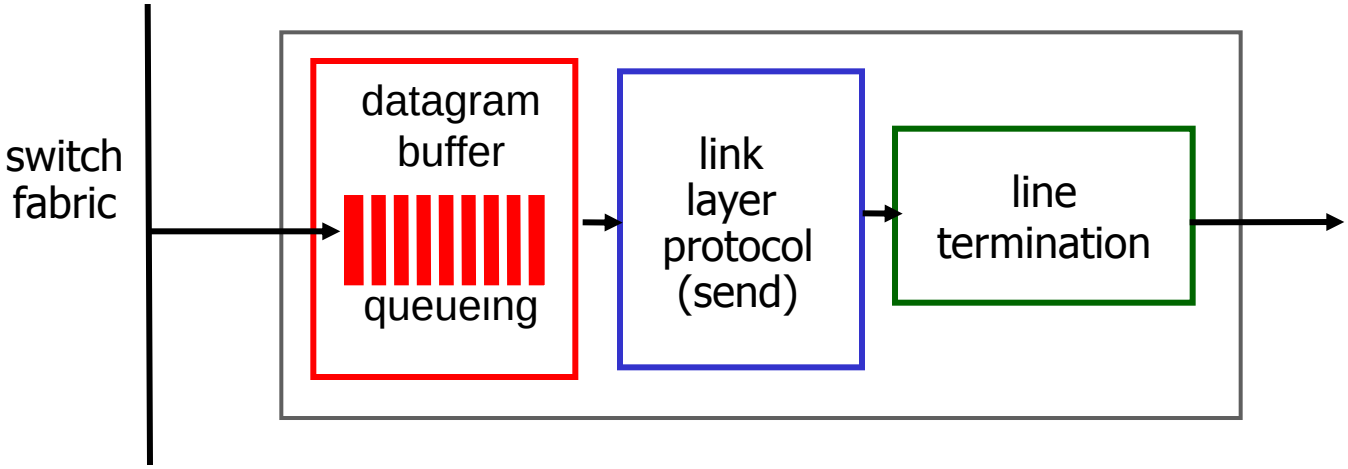

输出端口竞争:
只有一个红色的数据报可以被转移
左下角的红色分组被阻塞




一个分组时间之后：绿色分组经历了HOL阻塞

# 输出端口排队

*This slide in HUGELY important!*

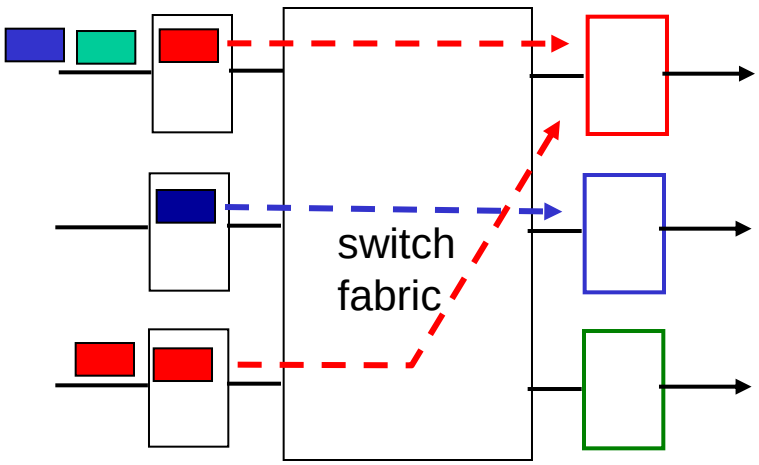



Datagram (packets) can be lost due to congestion, lack of buffers

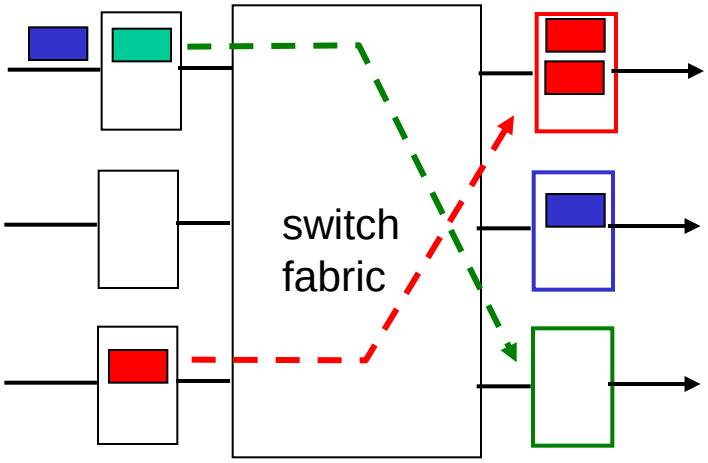
- *当数据报从交换结构到达输出端口的速率超过输出端口的传输速率，则需要缓存数据报*
- *调度价值用来在队列中选择传输的数据报*

Priority scheduling – who gets best performance, network neutrality

# 输出端口排队




at *t*, packets more from input to output

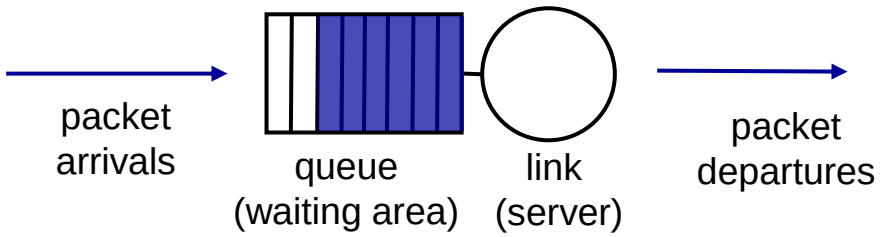



one packet time later

- 当经过交换结构到达的速度超过了输出端口的处理线速就会发生排队
- 当输出端口的缓冲区溢出时就会发生丢包!

# 4.2.5 分组调度

- 调度：选择该链路上下一个发送的分组
- FIFO (first in first out) 调度: 根据到达队列的顺序发送
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority basis
    - *random:* drop/remove randomly



# 三种调度策略的实例——FIFO

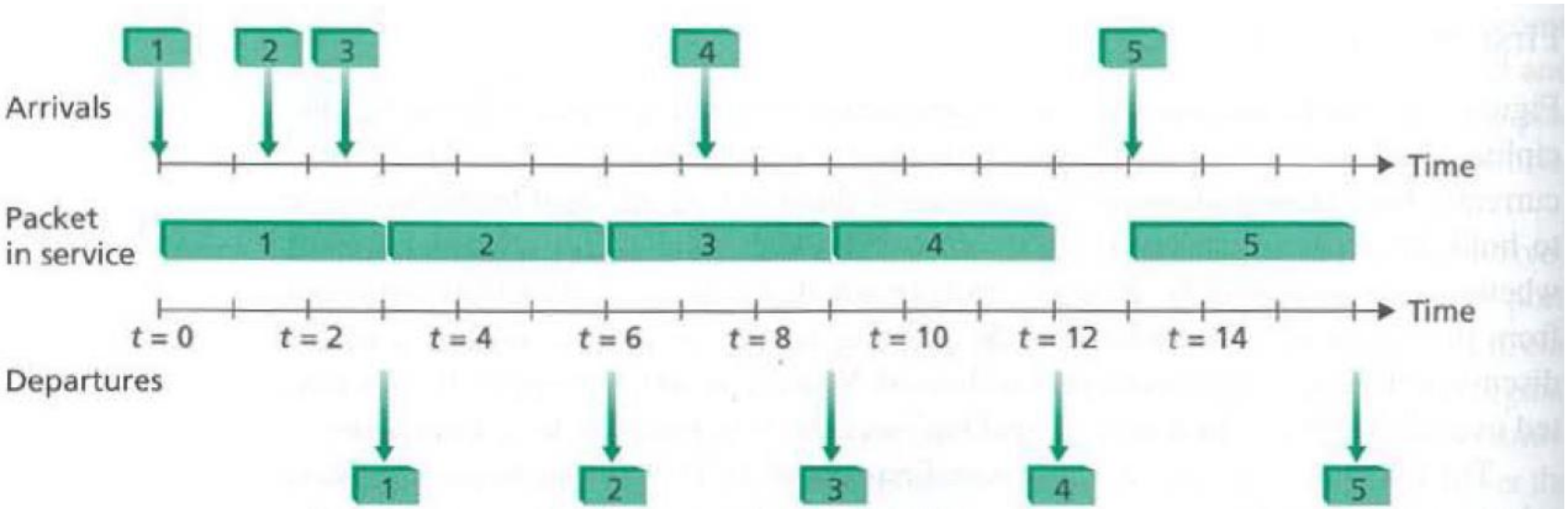
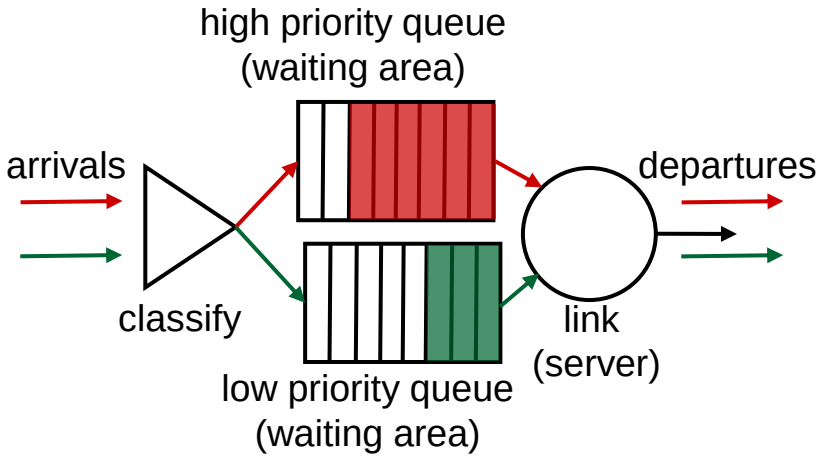



**Figure 4.11** ♦ The FIFO queue in operation

# 三种调度策略的实例——优先级



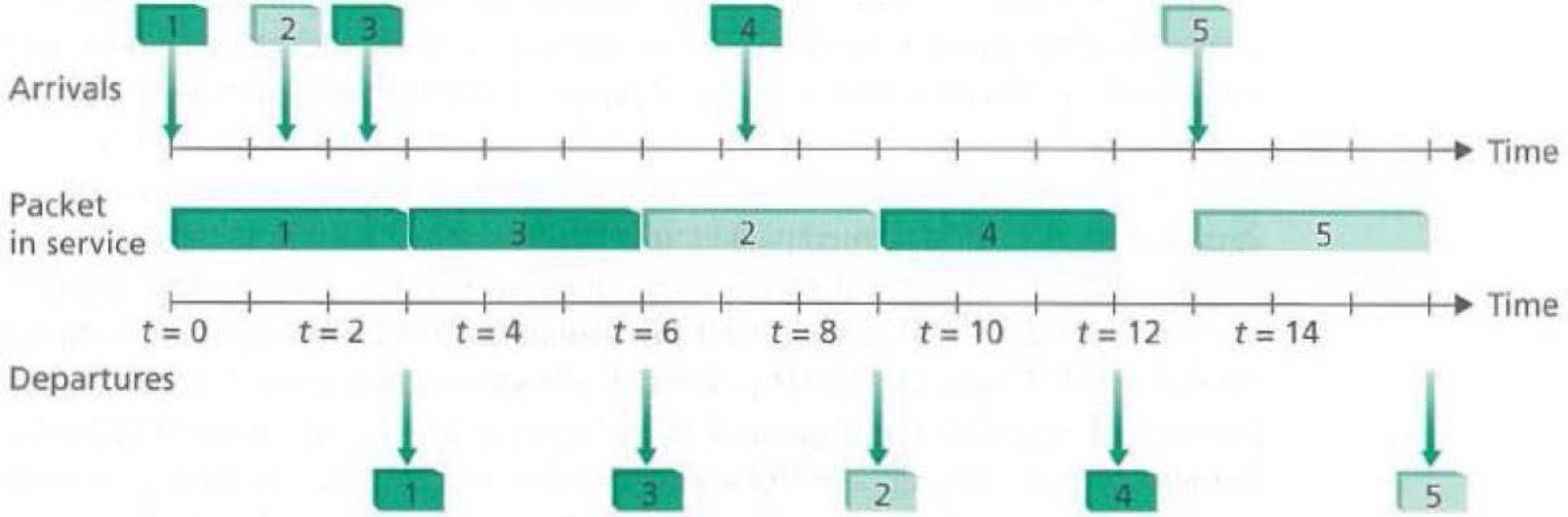




**Figure 4.13** ♦ The priority queue in operation

# 三种调度策略的实例—Round Robin

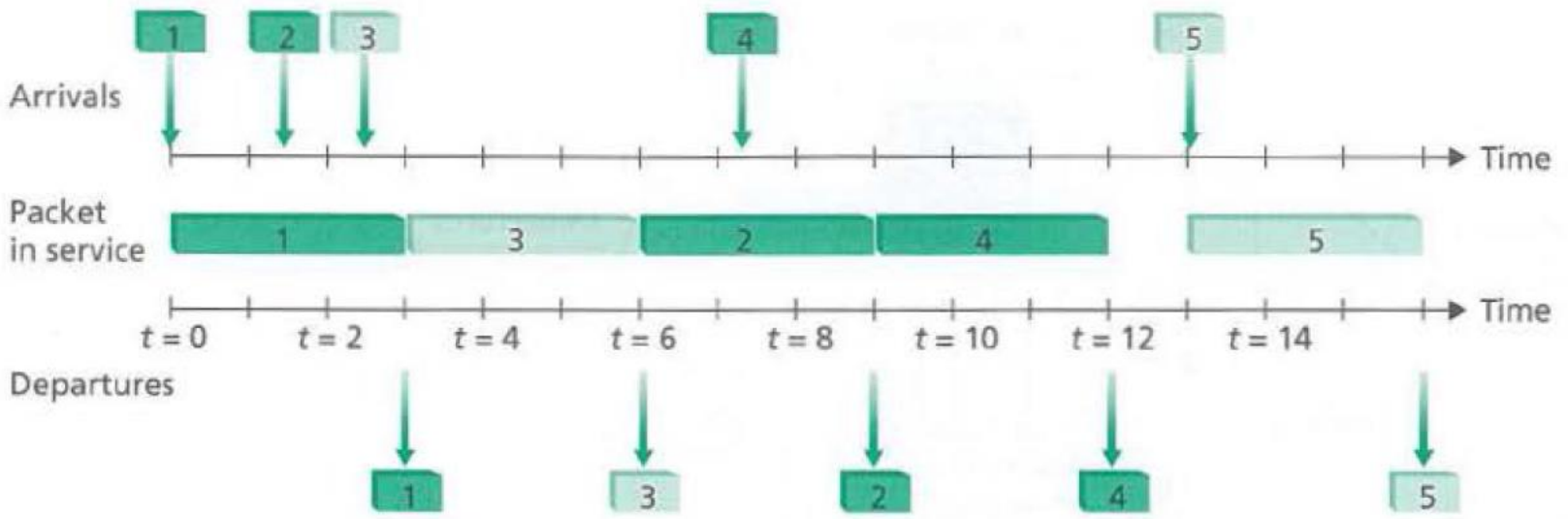
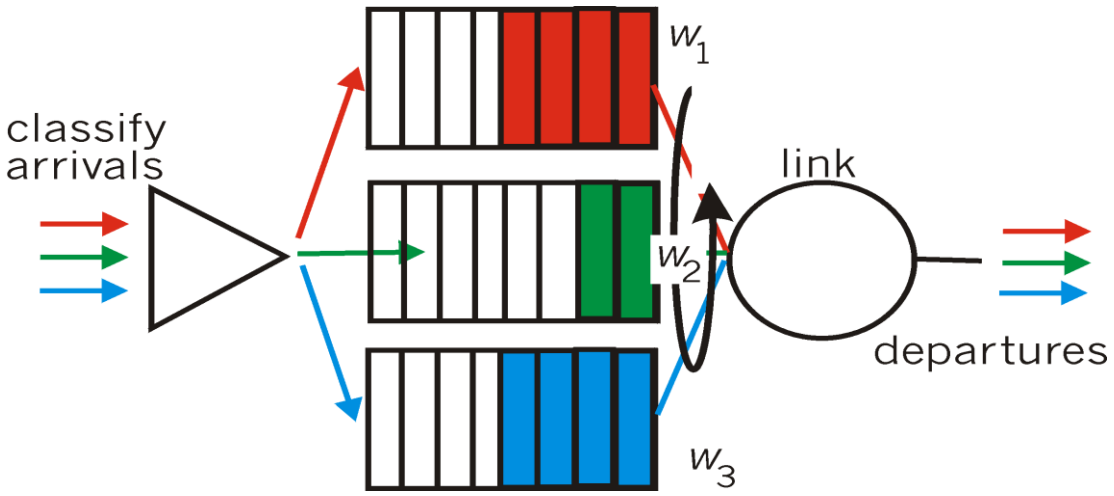



**Figure 4.14** ♦ The two-class robin queue in operation

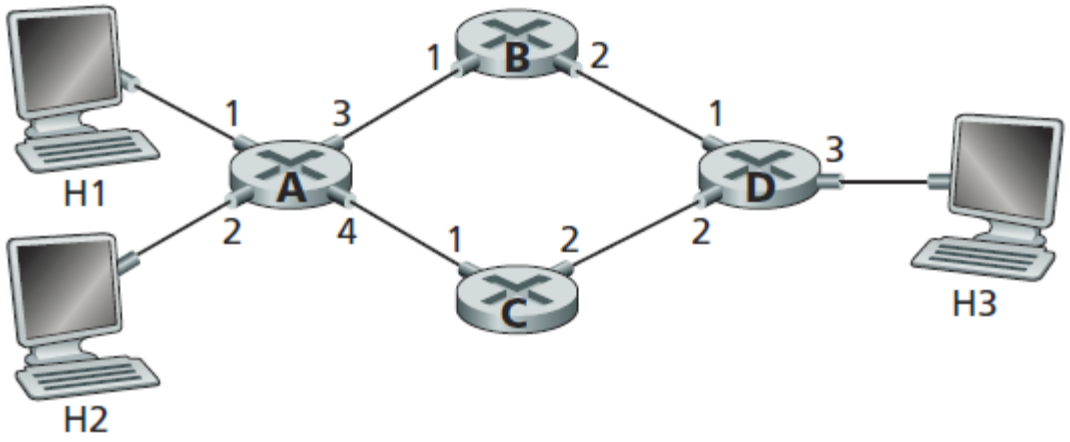
# 三种调度策略的实例——加权公平队列

*Weighted Fair Queuing (WFQ):*

- generalized Round Robin
- 在循环中每个队列有自己的权重
- real-world example?



# 课堂练习




P4. Consider the network below.

a. Suppose that this network is a datagram network. Show the forwarding table in router A, such that all traffic destined to host H3 is forwarded through interface 3.

b. Suppose that this network is a datagram network. Can you write down a forwarding table in router A, such that all traffic from H1 destined to host H3 is forwarded through interface 3, while all traffic from H2 destined to host H3 is forwarded through interface 4? (Hint: this is a trick question.)

c. Now suppose that this network is a virtual circuit network and that there is one ongoing call between H1 and H3, and another ongoing call between H2 and H3. Write down a forwarding table in router A, such that all traffic from H1 destined to host H3 is forwarded through interface 3, while all traffic from H2 destined to host H3 is forwarded through interface 4.

d. Assuming the same scenario as (c), write down the forwarding tables in nodes B, C, and D.

# 课堂练习

| Destination Address Range | Link Interface |
|---|---|
| 11100000 00000000 00000000 00000000<br>through<br>11100000 00111111 11111111 11111111 | 0 |
| 11100000 01000000 00000000 00000000<br>through<br>11100000 01000000 11111111 11111111 | 1 |
| 11100000 01000001 00000000 00000000<br>through<br>11100001 01111111 11111111 11111111 | 2 |
| otherwise | 3 |

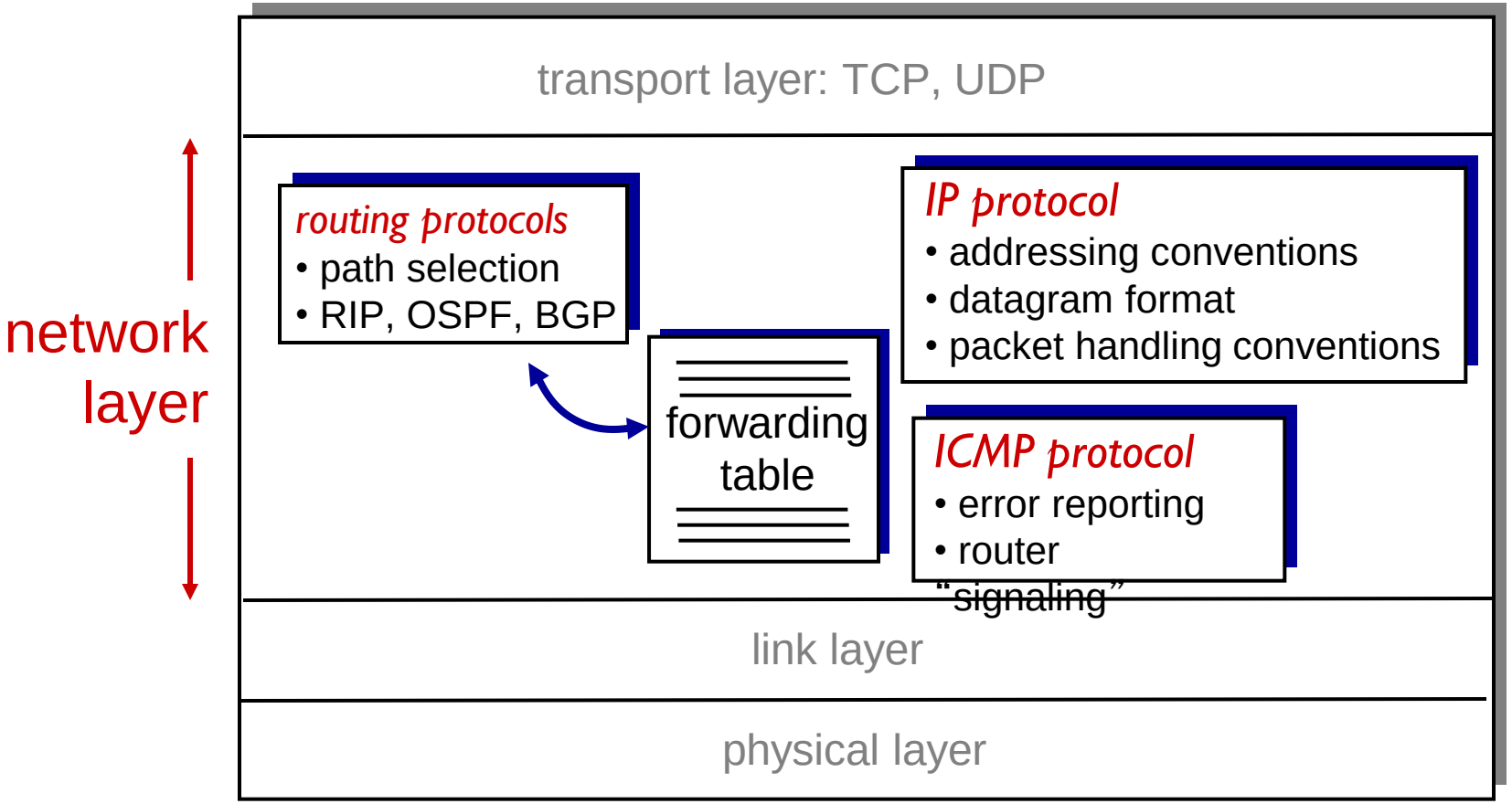
a. Provide a forwarding table that has five entries, uses longest prefix matching, and forwards packets to the correct link interfaces.

b. Describe how your forwarding table determines the appropriate link interface for datagrams with destination addresses:

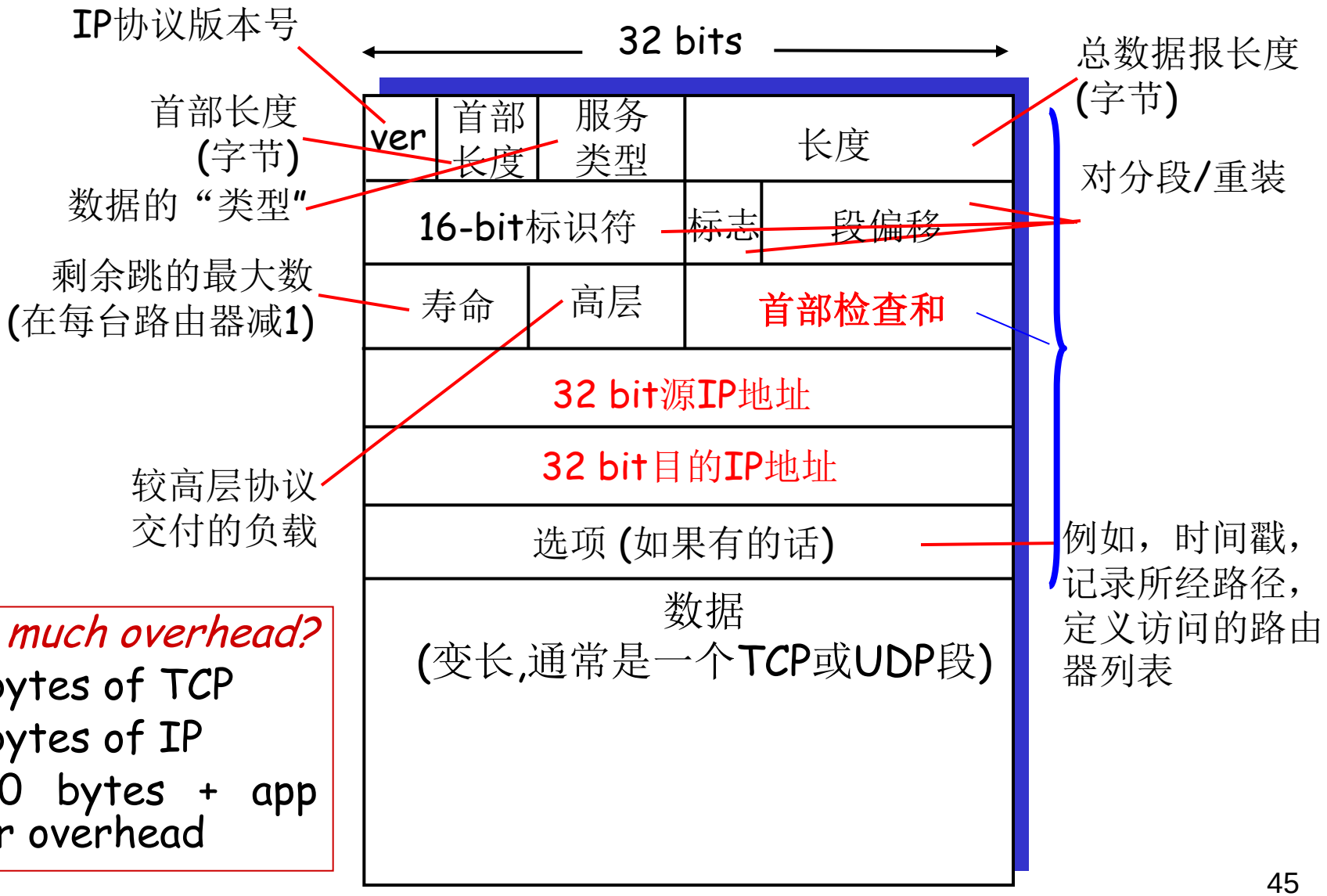
11001000 10010001 01010001 01010101
11100001 01000000 11000011 00111100
11100001 10000000 00010001 01110111

# 4.3：网际协议：IPv4、寻址、IPv6

host, router network layer functions:



# 4.3.1 IP 数据报格式（IPv4）




*how much overhead?*

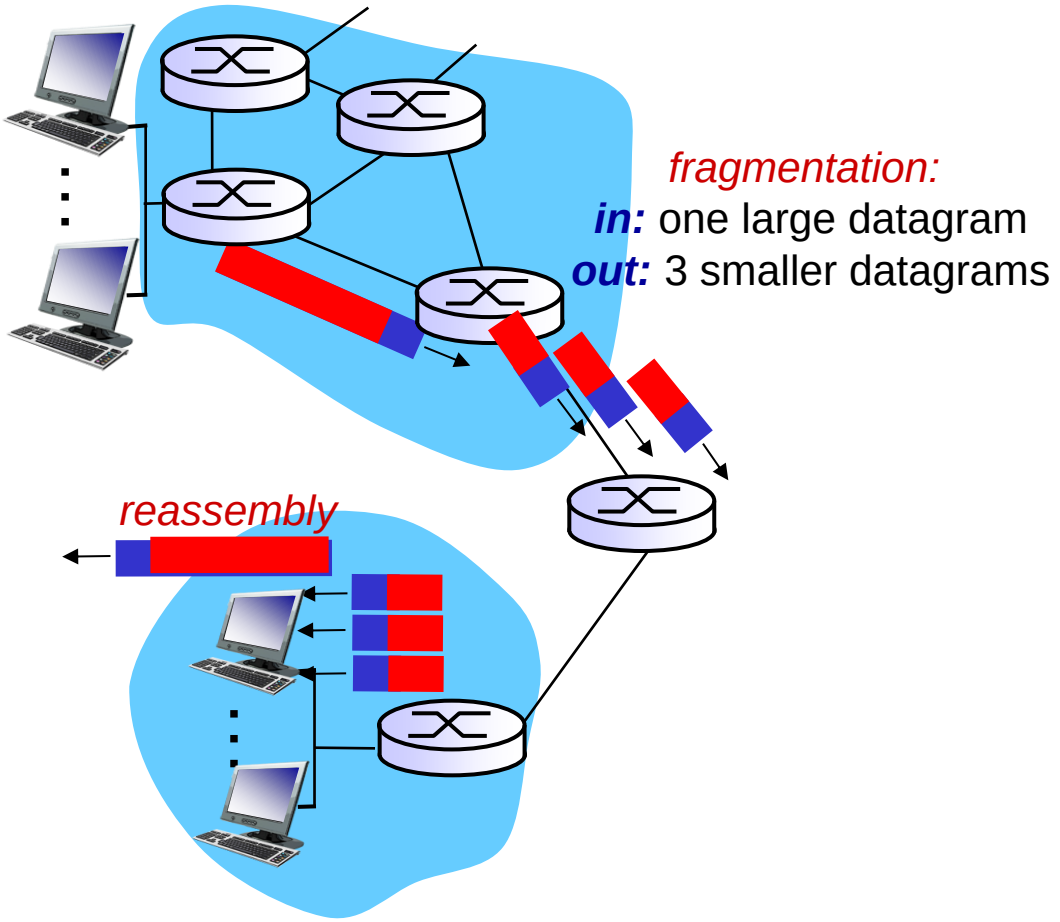
20 bytes of TCP

20 bytes of IP

= 40 bytes + app layer overhead

# 4.3.2 IPv4数据报分片和重组

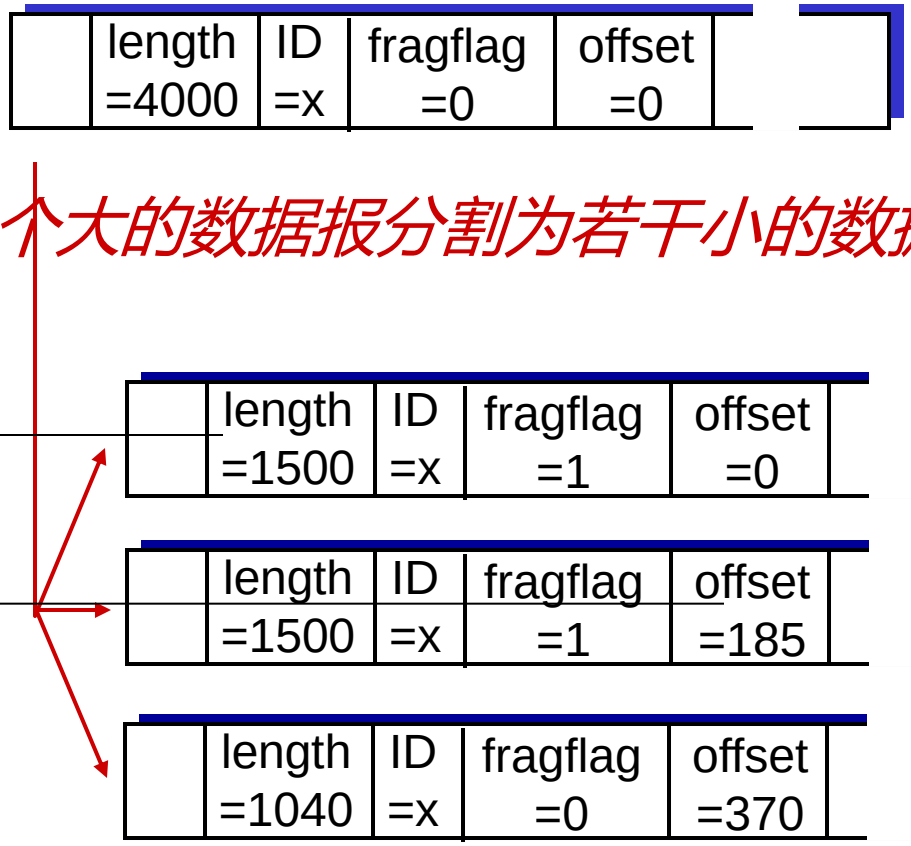
- 每个数据链路有自己的MTU，链路类型不同，MTU的值也不同，这里MTU指的是数据链路帧的数据区的最大字节数
- 在因特网中，一个大的分组可能在路由器中被分割为几个分片，在最终的目的主机上，将这些分片重新组装成一个大的分组
- 为了进一步识别出这些分组，需要对分片进行标识



# 分片的例子

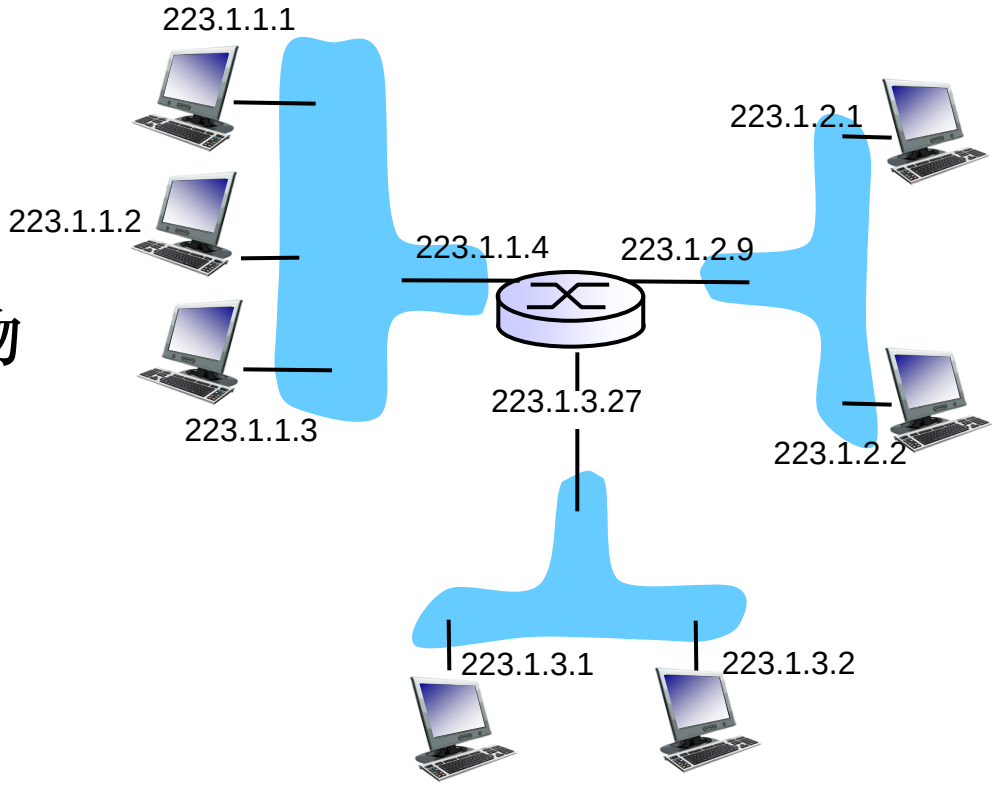
*example:*

- 4000 byte datagram
- MTU = 1500 bytes



# 4.3.3 IPv4编址

- IPv4地址: 分配给主机或路由器接口的32-bit标识符
- 接口: 主机/路由器与物理链路之间的边界
  - 路由器有多个接口
  - 主机可以有多个接口
  - 每个接口有一个IP地址




223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

# IPv4地址(补充)

- 早期的分类编址：根据不同的取值范围，早期将IP地址分为五类
  - IP地址中前5位用于标识IP地址的类别
  - A类地址的第一位为“0”，
  - B类地址的前两位为“10”，
  - C类地址的前三位为“110”，
  - D类地址的前四位为“1110”，
  - E类地址的前五位为“11110”。
  - A类、B类与C类地址为基本的IP地址。

# IPv4地址(补充)

- IP地址由两部分组成：
  - 网络号：指明主机所在物理网络的编号。
  - 主机号：主机在物理网络中的编号。

# 早期分类地址导致的问题

- 一个A类的IP地址，可以有24bit用于分配主机地址，因此可以支持(2的24次方)个主机，但是一个家庭或者组织往往不需要这么多的地址空间，造成浪费。
- 一个C类的IP地址，只有8bit用于分配主机地址，因此只能支持256个主机，又不太够用。
- 因此，逐渐按类别进行地址分配被CIDR技术取代

# 常用的特殊IP地址

- 127.0.0.1-127.255.255.254 (127.0.0.0和127.255.255.255除外)
  - 这是预留的一组IP地址，主要是用来识别主机本身的地址。也叫做“localhost”一般用来测试使用的。做开发的人比较熟悉。
- 10.x.x.x, 172.16.x.x-172.31.x.x, 192.168.x.x.
  - 这三个地址段主要是我们私有的内网地址。也就是我们平时企业或者家里局域网所使用的地址段，我们比较熟悉的应该就是192.168.x.x 这个地址段了

# 常用的特殊IP地址

- 0.0.0.0

这个地址严格上来说都不是真正意义上的IP地址。主要是用来标识不清楚的网络和主机的。系统遇到无法识别的网络或主机的时候会统一的归纳到这个地址

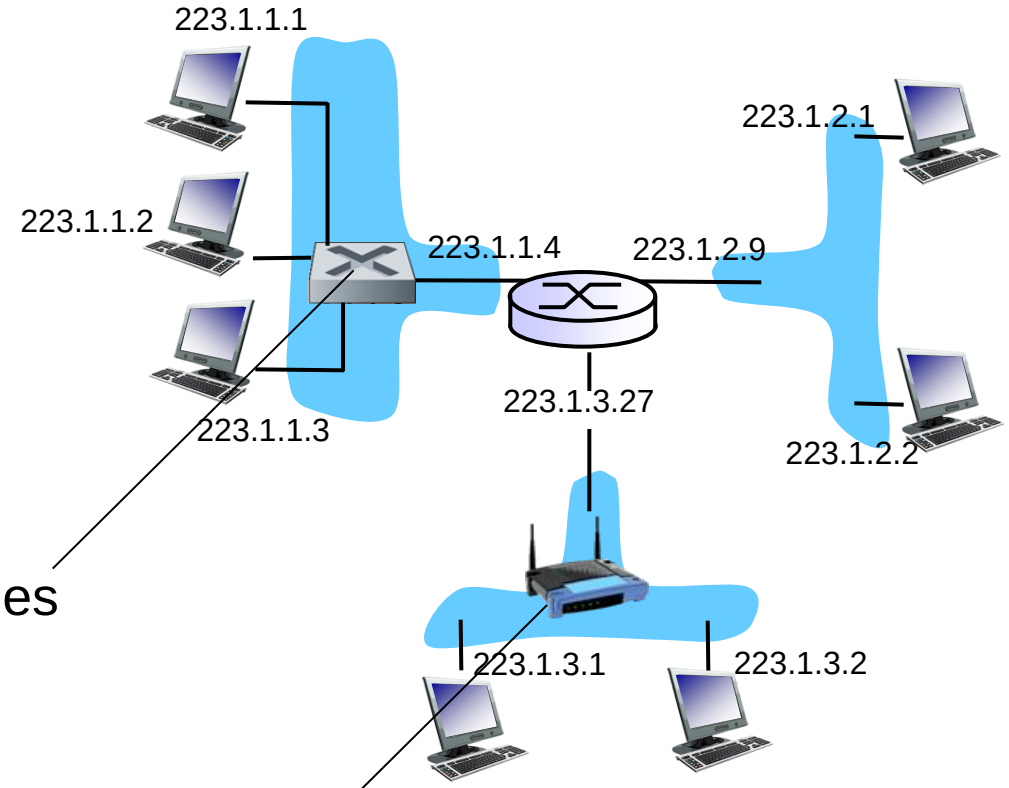
- 255.255.255.255

这个地址是受限的广播地址。主要指一个网段内的所有主机

# IPv4寻址

*Q: how are interfaces actually connected?*

*A: we'll learn about that in chapter 5, 6.*



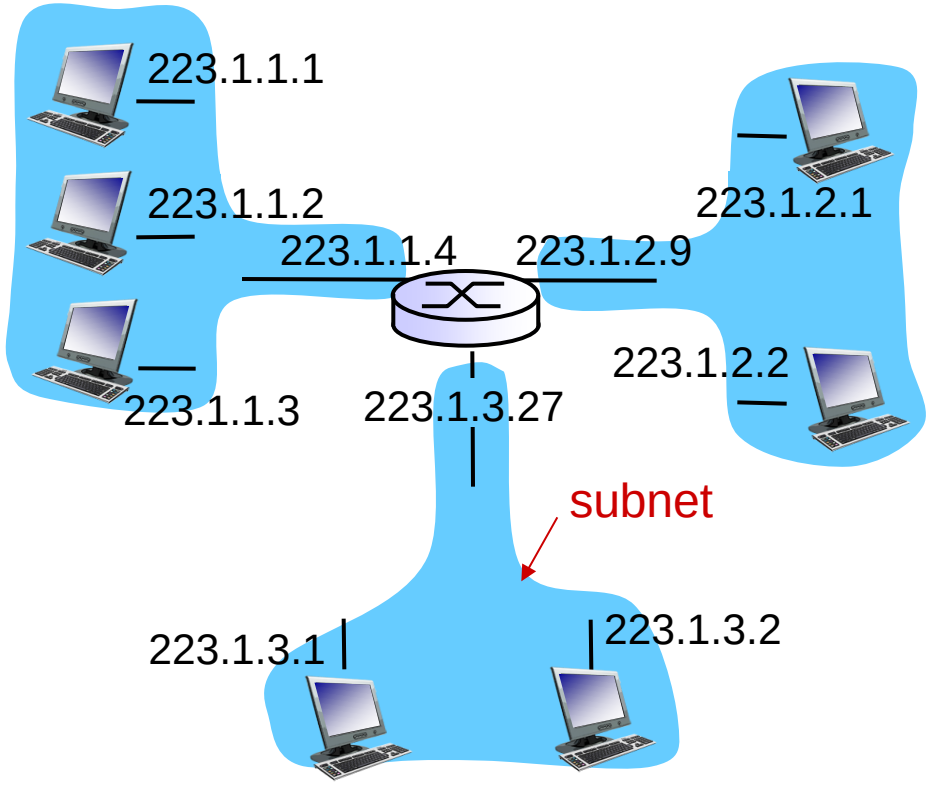

*A:* wired Ethernet interfaces connected by Ethernet switches

*For now:* don't need to worry about how one interface is connected to another (with no intervening router)

*A:* wireless WiFi interfaces connected by WiFi base station

# 子网的概念

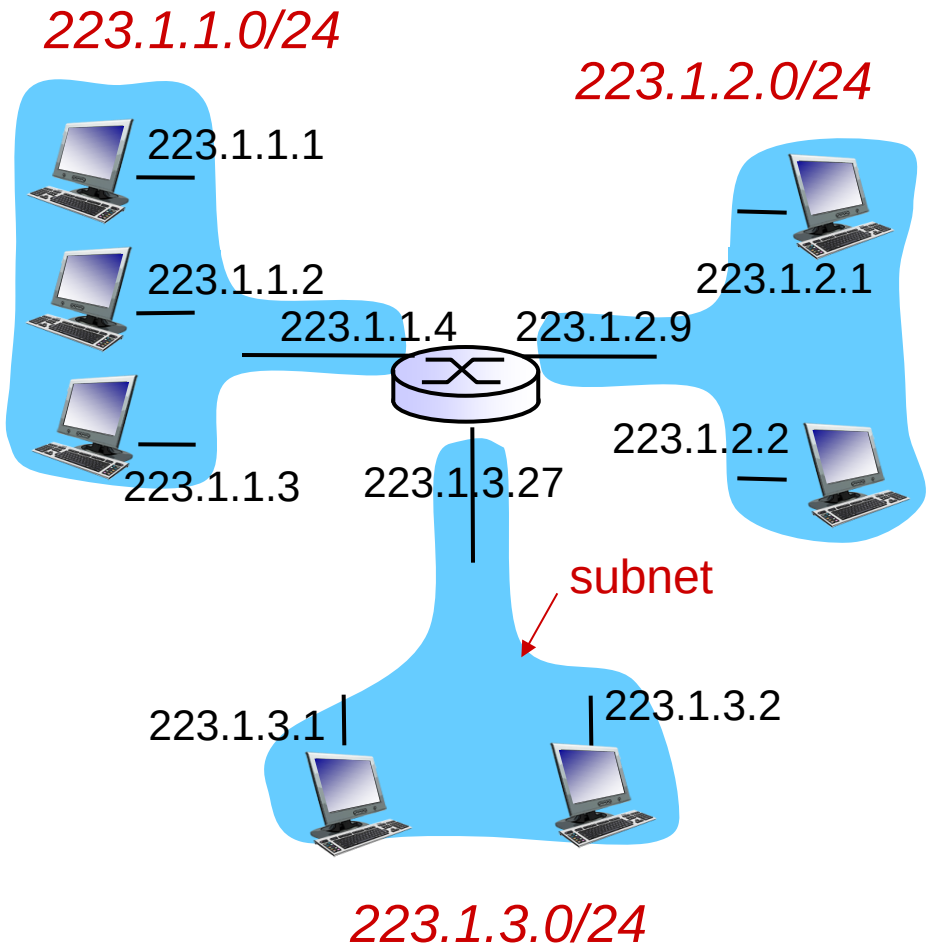
- IP 地址:
  - 子网部分 (高位 bits)
  - 主机部分 (低位 bits)
- 什么是一个子网？
  (从IP地址的观点来看)
  - 设备接口的IP地址具有同样的网络部分
  - 没有路由器的介入，物理上能够相互到达




network consisting of 3 subnets

# 子网的概念

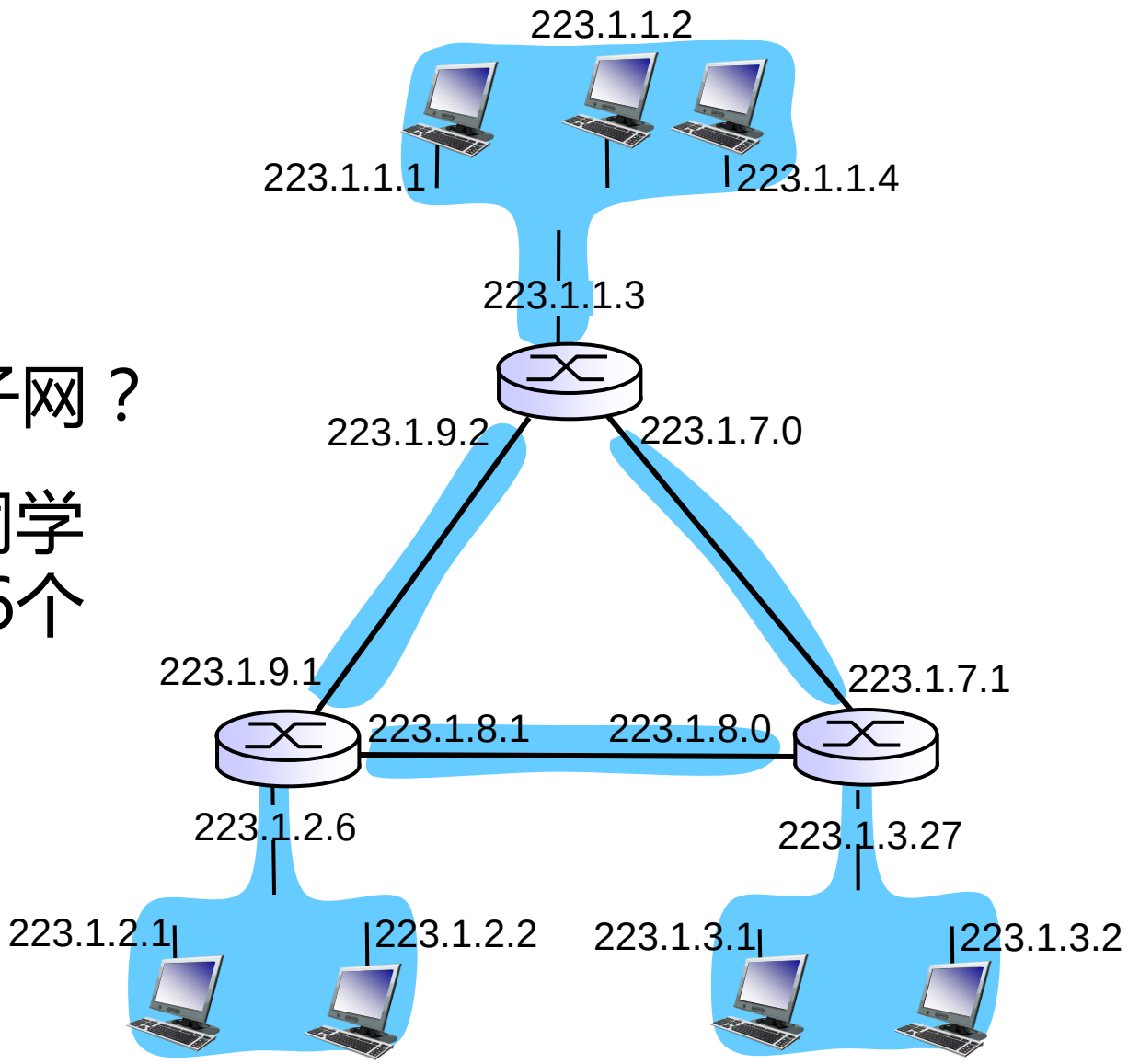
- 定义：为了确定子网，分开主机和路由器的每个接口，从而产生了几个分离的网络岛，这些独立网络中的每一个叫做一个子网(subnet).




subnet mask: /24

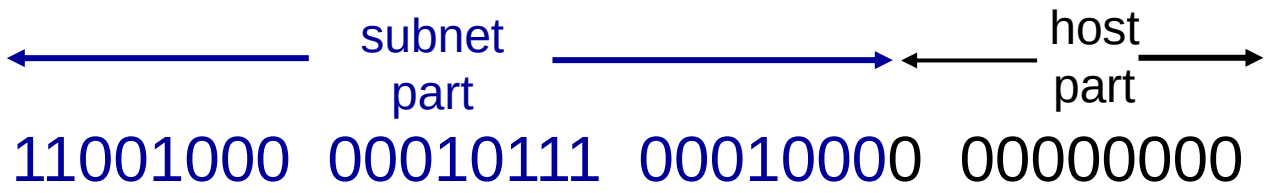
图中有多少个子网？

6个子网，请同学们分别列出这6个子网



# 无类别域间路由选择CIDR

- CIDR: Classless InterDomain Routing
  - 地址中的网络部分可以任意长
  - 地址格式: a.b.c.d/x, 这里 x 是地址网络部分的bit数
  - 子网掩码：用全1表示网络位，全0表示主机位




子网掩码　　200.23.16.0/23

| | | | |
|---|---|---|---|
| 11111111 | 11111111 | 11111110 | 00000000 |
| 255 | 255 | 254 | 0 |

# IPv4寻址

- IPv4寻址是根据网络号(子网部分)，首先在路由器的转发表中查找是否有该网络号对应的条目，将分组转发到该网络号对应的输出端口；
- 公司(个人)的机器如果配置了某个IP地址段，最初只有接入ISP的路由器知道这个IP地址段的信息，如何让其他路由器知道这个信息呢?
- 我们说路由信息的扩散是通过BGP协议(路由协议)向与该本地ISP相连的其他ISP的路由器扩散，直到所有因特网上的路由器都知道这个IP地址段的前缀信息。
- 怎么建立转发表条目，我们将在第5章进一步学习!

# 关于IP地址及域名的获得

- **ISP获得地址块的方法**

**ICANN**: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

  - 分配IP地址段
  - 管理根DNS服务器
  - 分配域名及解决纠纷

# 如何获得IP addresses

***Q:*** **如何获得IP地址的网络部分**

***A:*** **由ISP的地址空间中分配**

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

**注意每个子网的网络前缀变为了23比特?**

# 如何分配子网(重要)

- 给定一个地址块，怎么进行子网划分，以及为每个子网分配地址段
- 某单位从ISP处购买了一个IP地址块202.40.70.0/24。如果该单位有3个部门，第1个部门78台计算机，第2个和第3个部门各有48台计算机，请你为3个部门划分子网，并给出各部门的网络号、子网掩码和主机可分配的IP地址范围。

子网1的最后8比特的设置：0xxxxxxx；这里后7比特表示主机号，/25；
子网2的最后8比特的设置：10xxxxxx；这里后6比特表示主机号，/26；
子网3的最后8比特的设置：11xxxxxx；这里后6比特表示主机号，/26；

# 如何分配子网(重要)

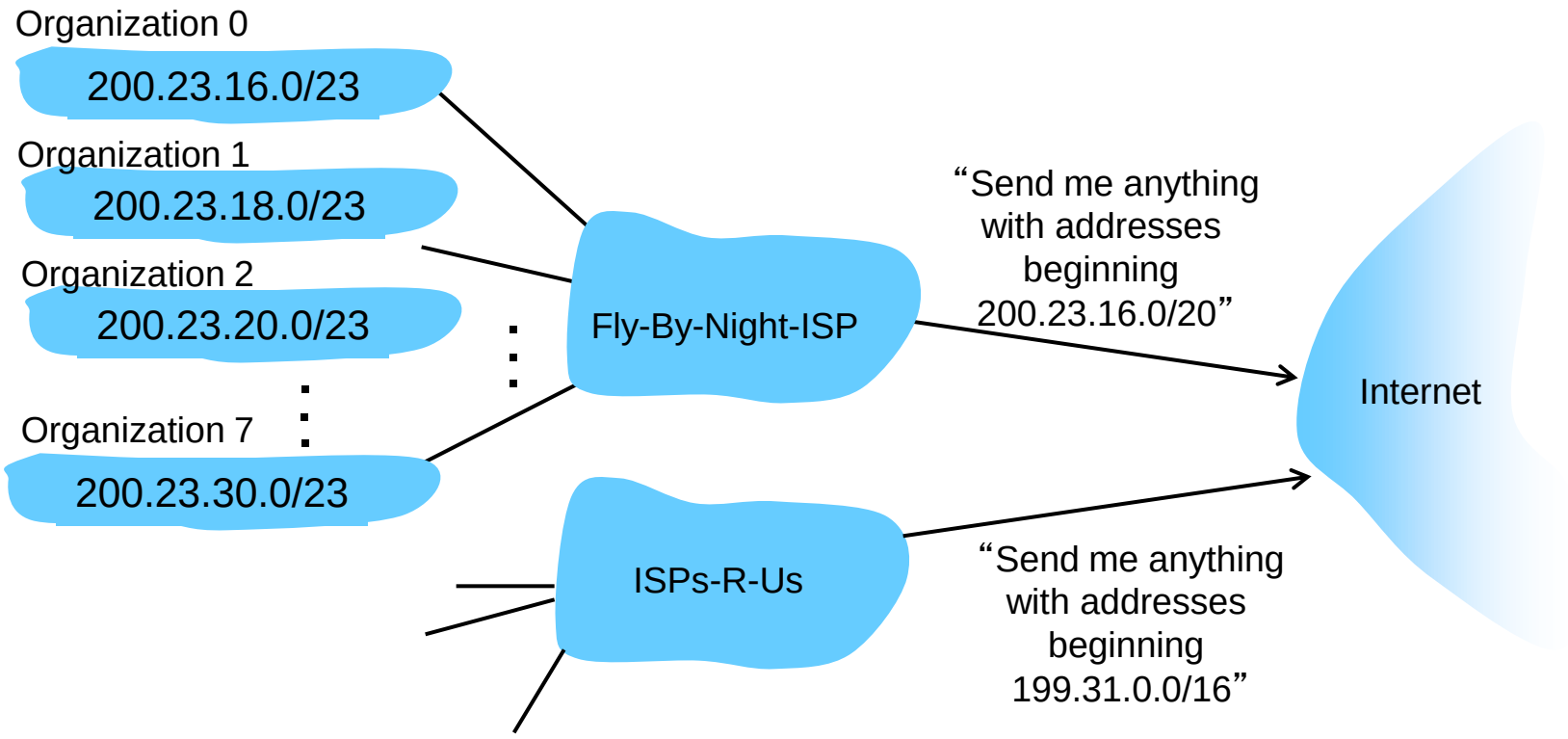
子网1：202.40.70.00000001 ~202.40.70.01111110
(202.40.70. 1 ~ 202.40.70.126)
网络号为：202.40.70.0　子网掩码为：255.255.255.128

子网2：202.40.70.10000001 ~ 202.40.70.10111110
(202.40.70. 129~202.40.70.190)
网络号为：202.40.70.128　子网掩码为：255.255.255.192

子网3：202.40.70.11000001 ~ 202.40.70.11111110
(202.40.70. 193~202.40.70.254)
网络号为：202.40.70.192　子网掩码为：255.255.255.192

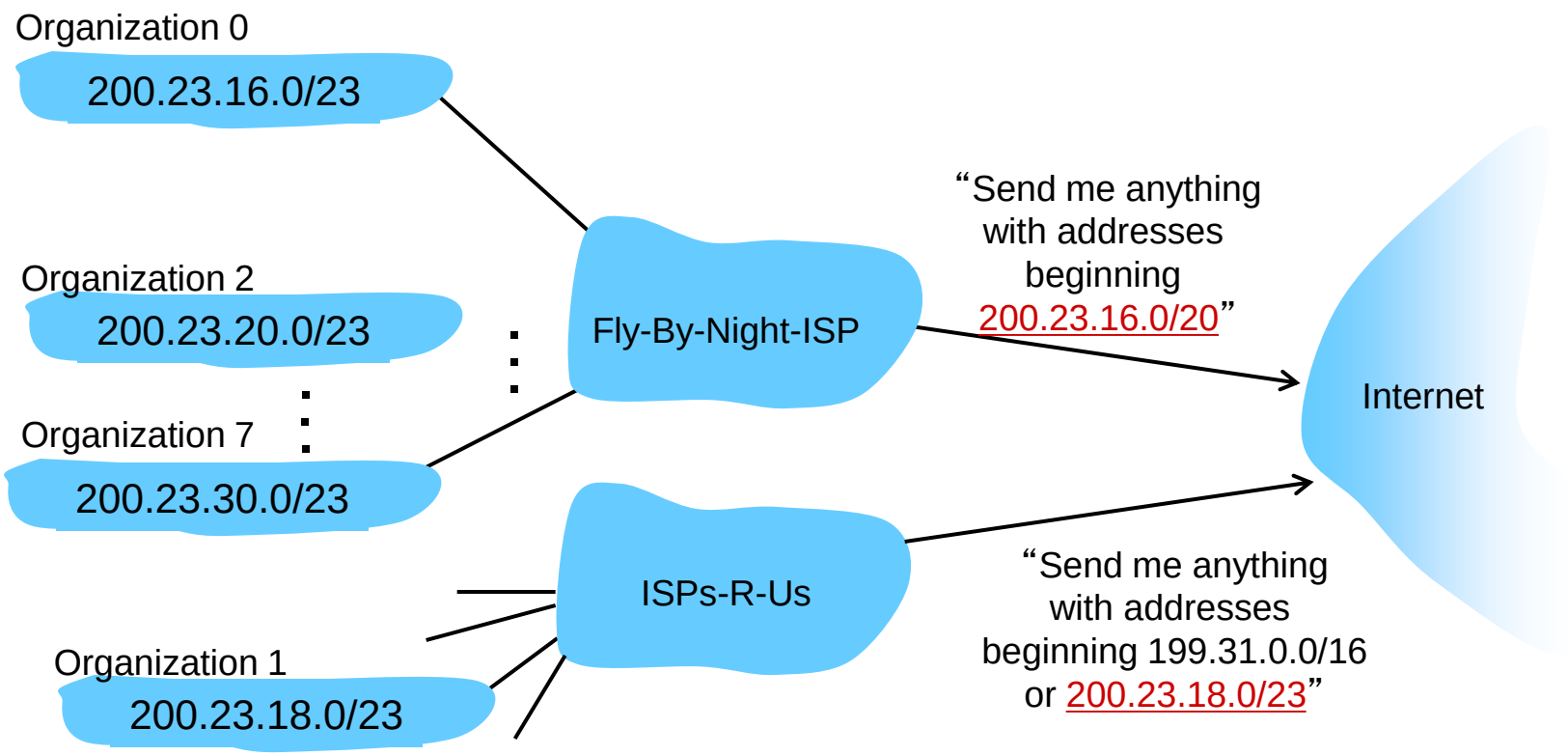
# 层次化地址: 路由聚合

hierarchical addressing allows efficient advertisement of routing information:



# 层次化地址:更详细的信息

ISPs-R-Us has a more specific route to Organization 1



# 课堂练习

P13. Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

某单位从ISP处购买了一个IP地址块202.40.70.0/24。如果该单位有3个部门，第1个部门78台计算机，第2个和第3个部门各有48台计算机，请你为3个部门划分子网，并给出各部门的网络号、子网掩码和主机可分配的IP地址范围。

# 配置主机IP地址

- 手工获取
- 动态主机配置协议获取（**DHCP**）：允许一个主机自动获取以下信息：
  - **IP**地址、子网掩码、默认网关、**DNS**地址
  - **plug-and-play**（即插即用）

hard-coded by system admin in a file

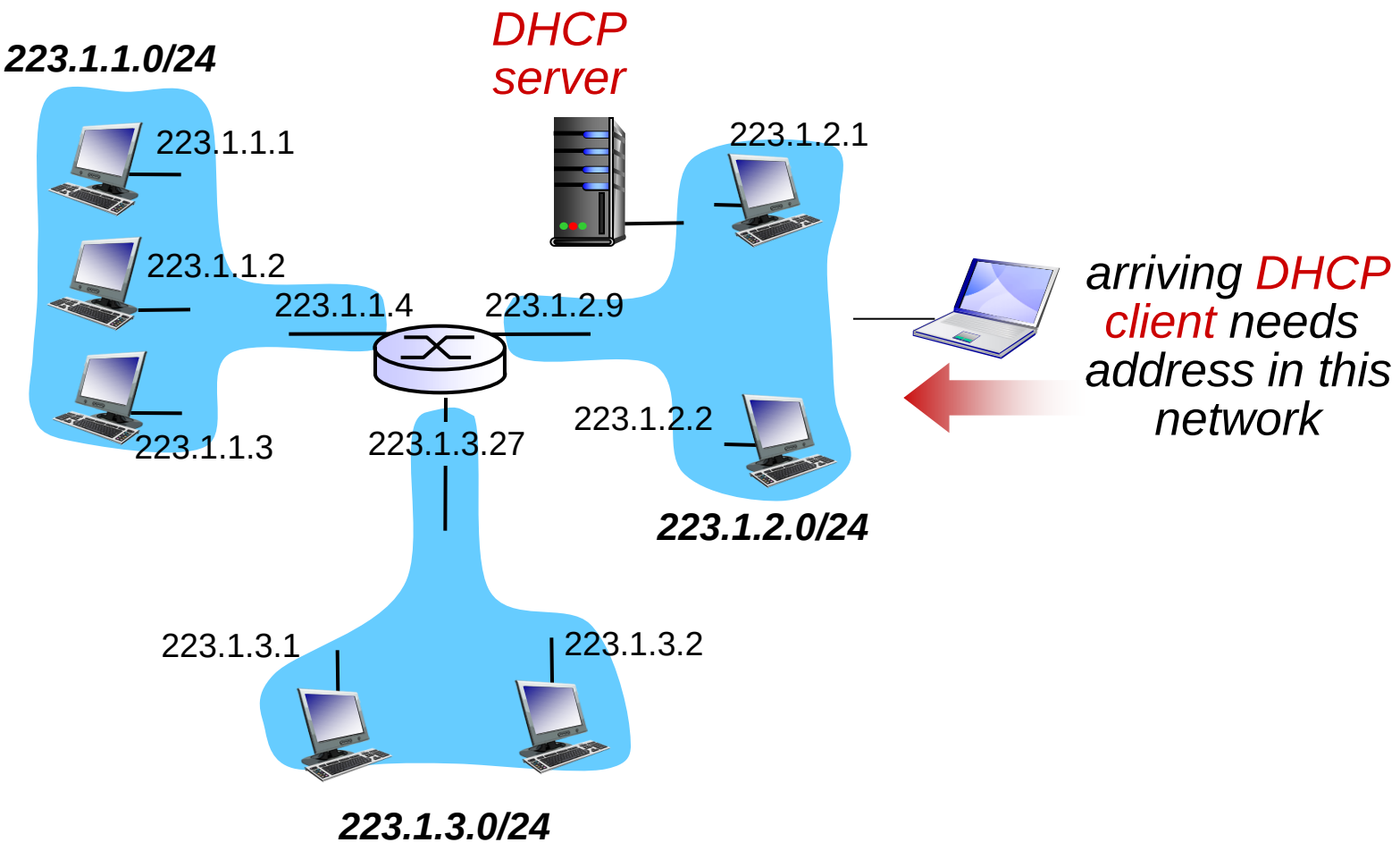
- Windows: control-panel->network->configuration->tcp/ip->properties
- UNIX: /etc/rc.config

DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server

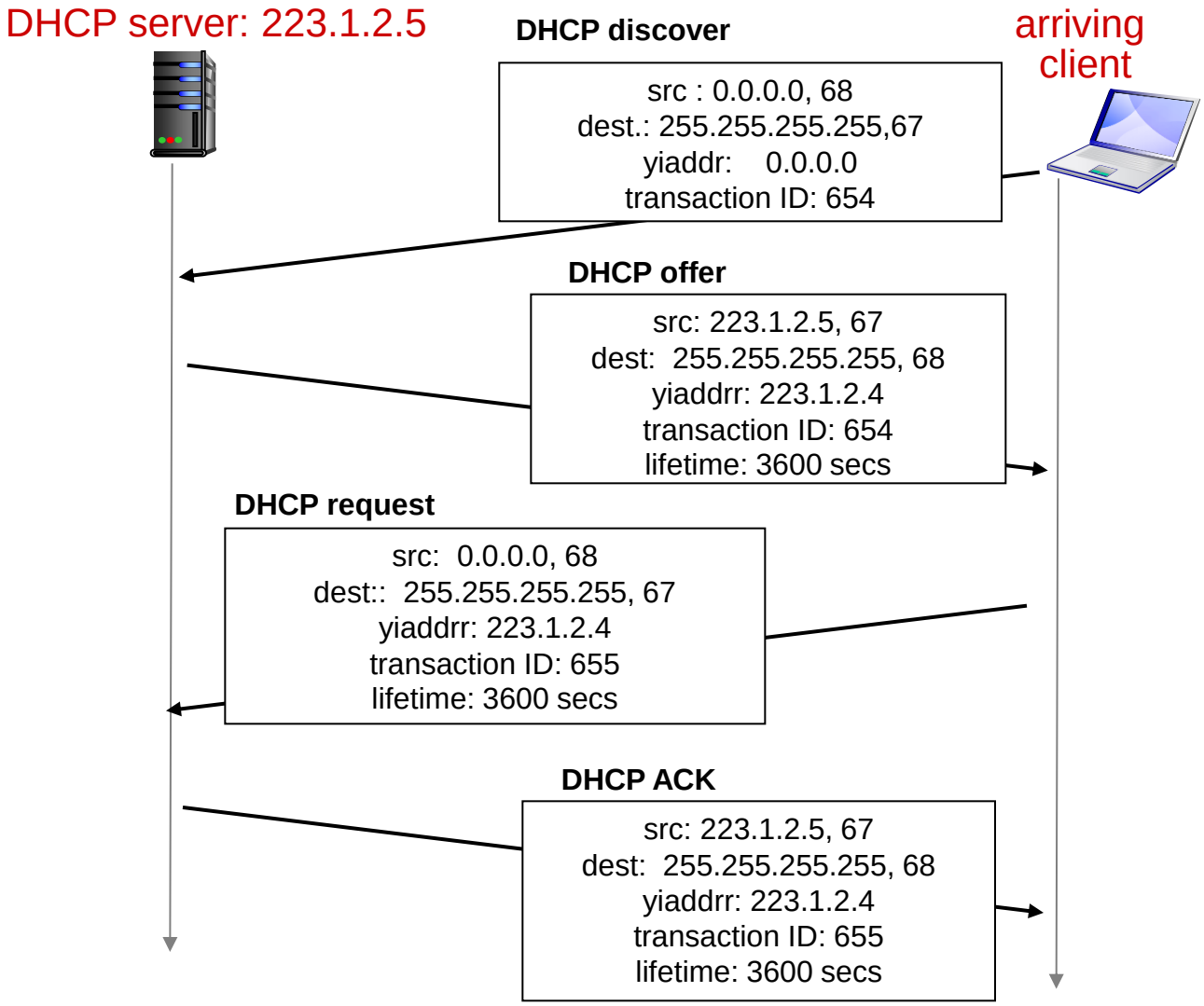
# 动态地址配置

- DHCP概述：
  - 主机广播“DHCP discover” msg [optional]
  - DHCP服务器响应“DHCP offer” msg [optional]
  - 主机请求IP地址：“DHCP request” msg
  - DHCP服务器发送地址：“DHCP ack” msg

# DHCP client-server scenario



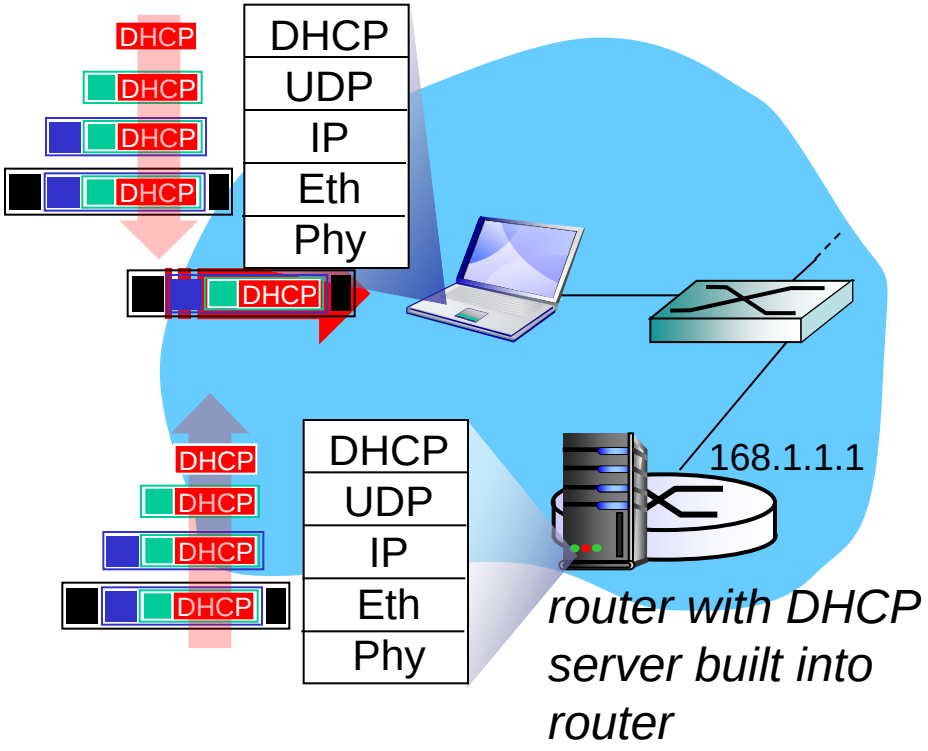
# DHCP客户-服务器场景



# DHCP：不仅获得IP地址

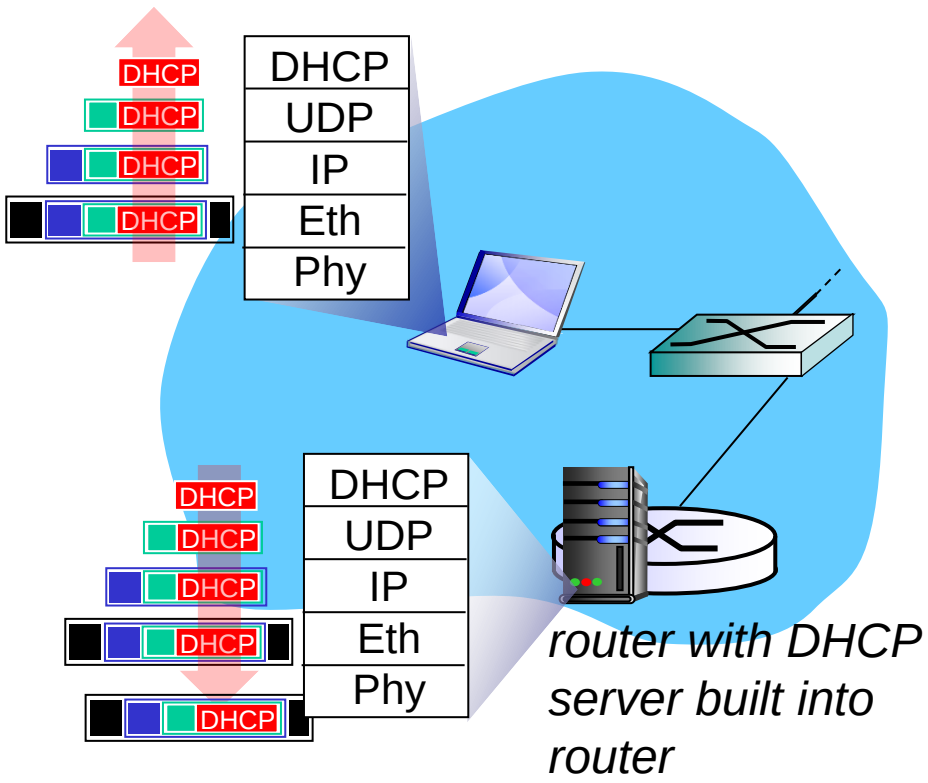
- DHCP返回的不仅仅是IP地址：
  - 客户的第一跳路由器的地址
  - DNS服务器的IP地址和域名
  - 网络掩码（用于指示网络的网络号部分和主机号部分）

# DHCP: 例子




- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

# DHCP: 例子




- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

# DHCP: Wireshark output (home LAN)

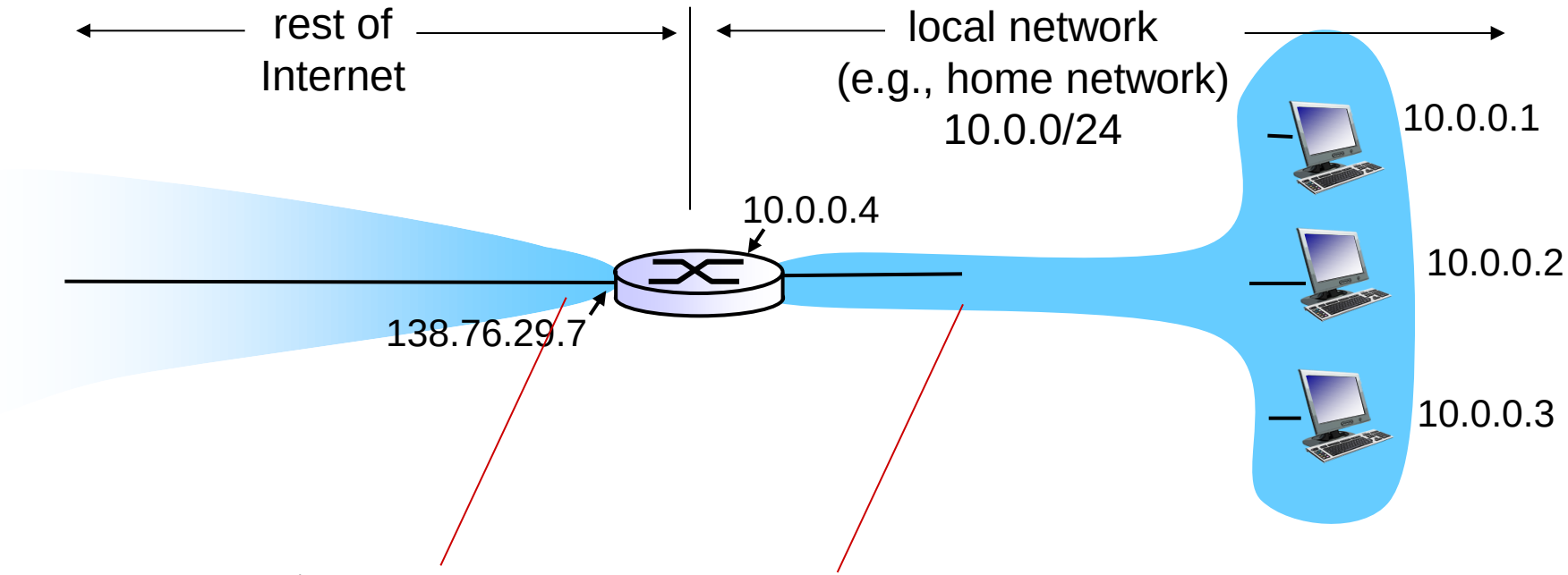
request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID: 0x6b3a11b7**
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
**Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)**
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
**Option: (55) Parameter Request List**
    Length: 11; Value: 010F03062C2E2F1F21F92B
    **1 = Subnet Mask; 15 = Domain Name**
    **3 = Router; 6 = Domain Name Server**
    44 = NetBIOS over TCP/IP Name Server
    ……

reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID: 0x6b3a11b7**
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
**Client IP address: 192.168.1.101 (192.168.1.101)**
Your (client) IP address: 0.0.0.0 (0.0.0.0)
**Next server IP address: 192.168.1.1 (192.168.1.1)**
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
**Option: (t=53,l=1) DHCP Message Type = DHCP ACK**
**Option: (t=54,l=4) Server Identifier = 192.168.1.1**
**Option: (t=1,l=4) Subnet Mask = 255.255.255.0**
**Option: (t=3,l=4) Router = 192.168.1.1**
**Option: (6) Domain Name Server**
    **Length: 12; Value: 445747E2445749F244574092;**
    **IP Address: 68.87.71.226;**
    **IP Address: 68.87.73.242;**
    **IP Address: 68.87.64.146**
**Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."**

# 4.3.4 网络地址转换(NAT)




所有分组离开本地网络有同样的 单个源端 NAT IP 地址: 138.76.29.7, 不同的源端口号

源端或目的端是这个网络的分组有10.0.0/24 地址
用于指示源地址或目的地址

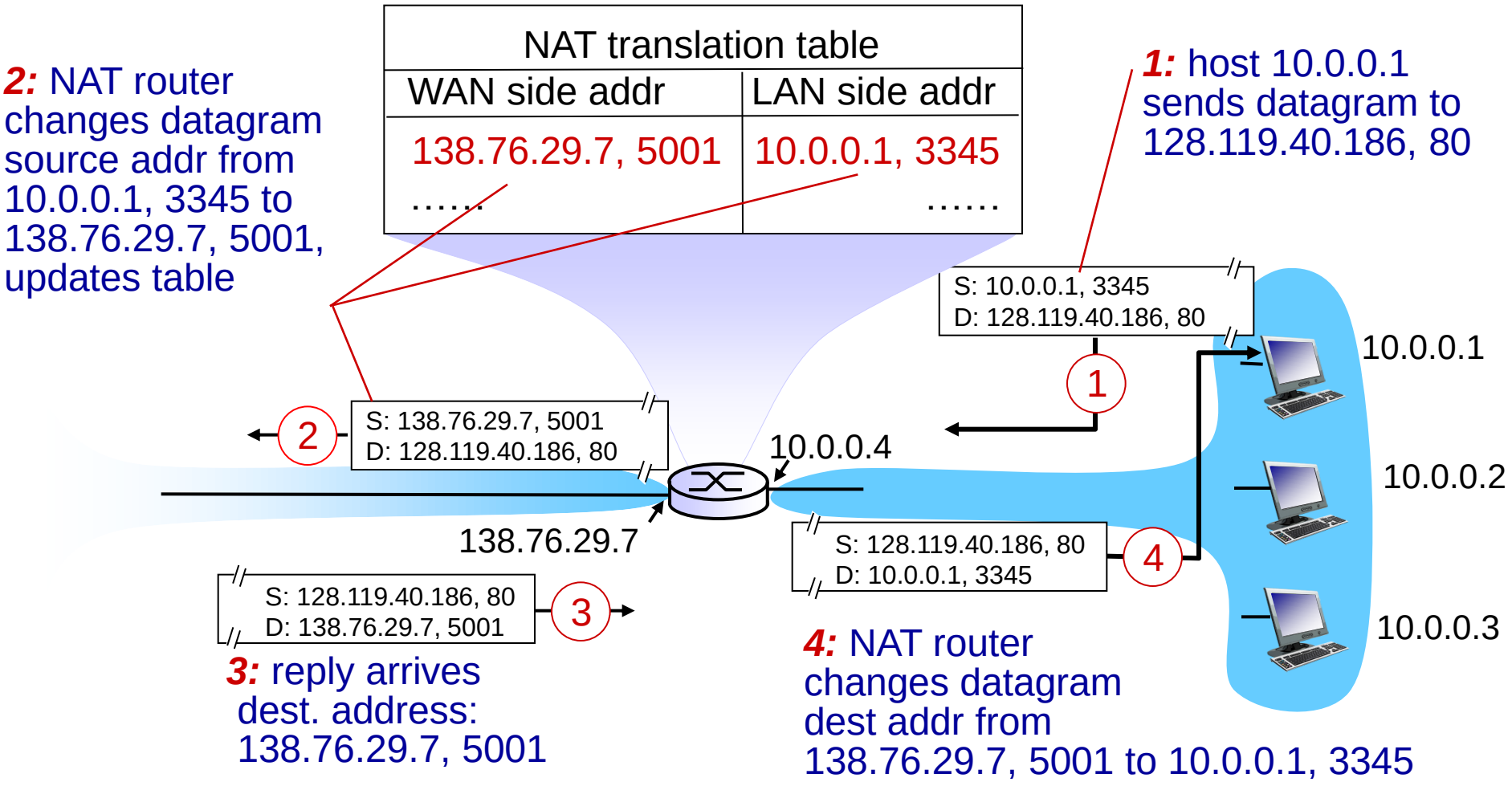
# 4.3.4 网络地址转换(NAT)

- 动机: 对外部网络来讲，本地网络只用一个IP地址
  - 不需要从 ISP分配一系列地址，只要一个IP地址用于所有设备
  - 在本地网络改变设备的IP地址不用通知外部世界
  - 可以变更 ISP 而不用改变本地网络的设备的地址
  - 本地网络内部设备不能被外部世界明确寻址,或是不可见 (增加了安全性).

# 4.3.4 网络地址转换(NAT)

- 执行NAT，路由器必须做到:
  - 外出的分组: 替换每个外出的分组的 (源IP 地址, 端口号) 为 (NAT IP 地址, 新端口号)，远程客户/服务器用(NAT IP地址, 新端口号)作为目的地来响应。
  - (在NAT转换表中)记录每个(源IP 地址, 端口号)到(NAT IP地址, 新端口号) 转换配对
  - 进来的分组: 对每个进来的分组，用保存在NAT表中的对应的(源IP 地址, 端口号) 替换分组中的目的域(NAT IP 地址, 新端口号)

# 4.3.4 网络地址转换(NAT)




* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

# 4.3.4 网络地址转换(NAT)

- 16-bit 端口号:
  - 一个局域网地址可以同时支持60,000 个并发连接!
- NAT存在争议
  - 路由器只应该处理到第三层
  - 违反了端到端主张

    应用程序设计者在设计时不得不将NAT加以考虑

    如P2P应用程序
  - 应使用IPv6来解决地址短缺问题
  - NAT穿透：如果客户机想要连接到NAT后面的服务器呢?

# 4.3.5 IPv6：动机

- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

*IPv6 datagram format:*

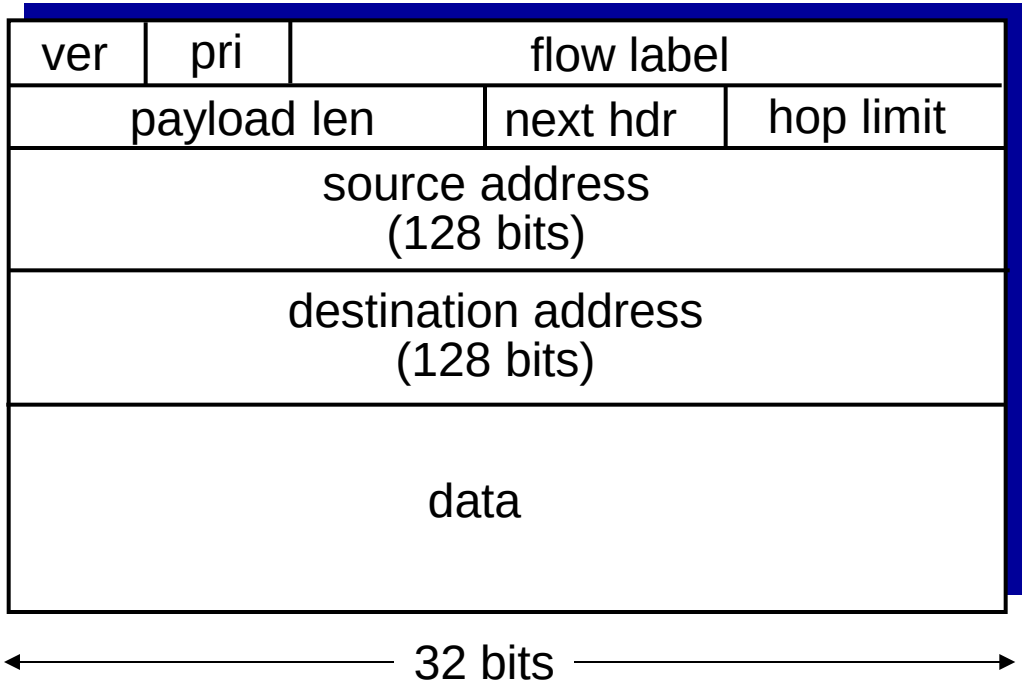
  - fixed-length 40 byte header
  - no fragmentation allowed

# IPv6 数据报格式

*priority:* identify priority among datagrams in flow

*flow Label:* identify datagrams in same “flow.”
(concept of“flow” not well defined).

*next header:* identify upper layer protocol for data

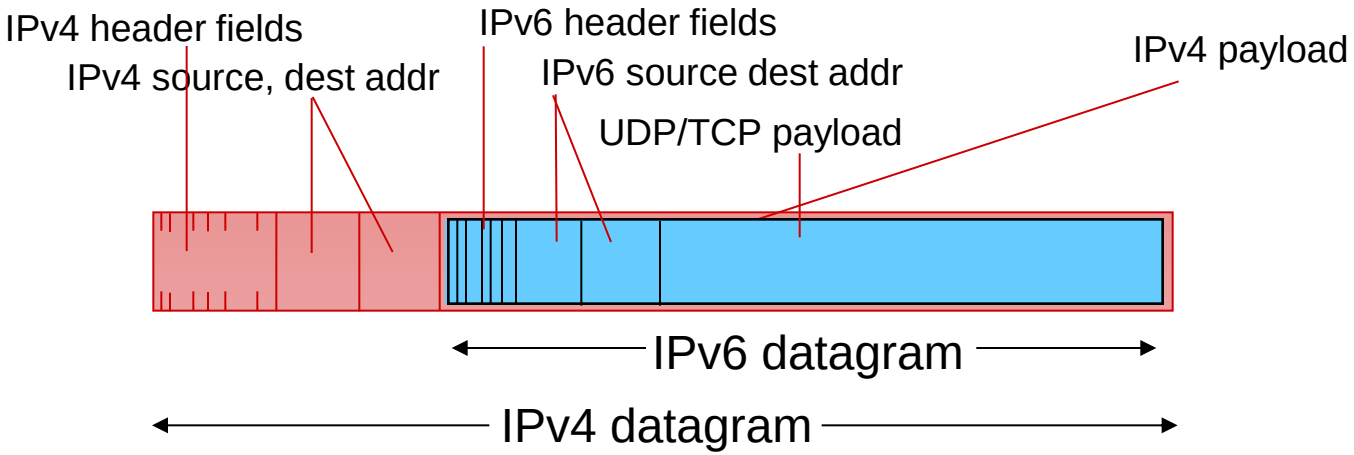


# Other changes from IPv4

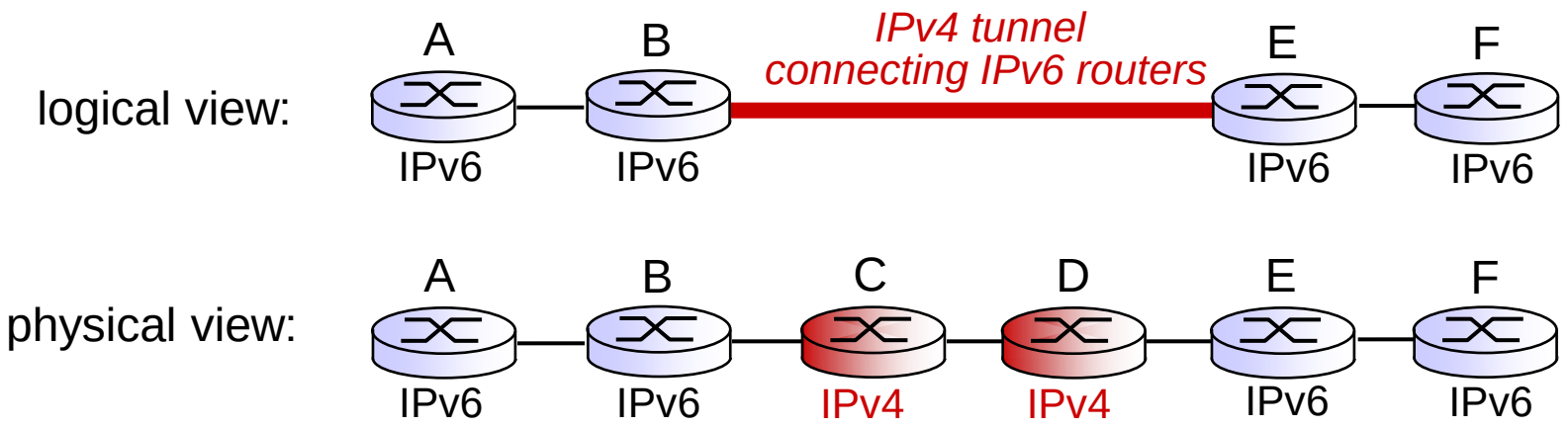
- *checksum*: removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

# Transition from IPv4 to IPv6

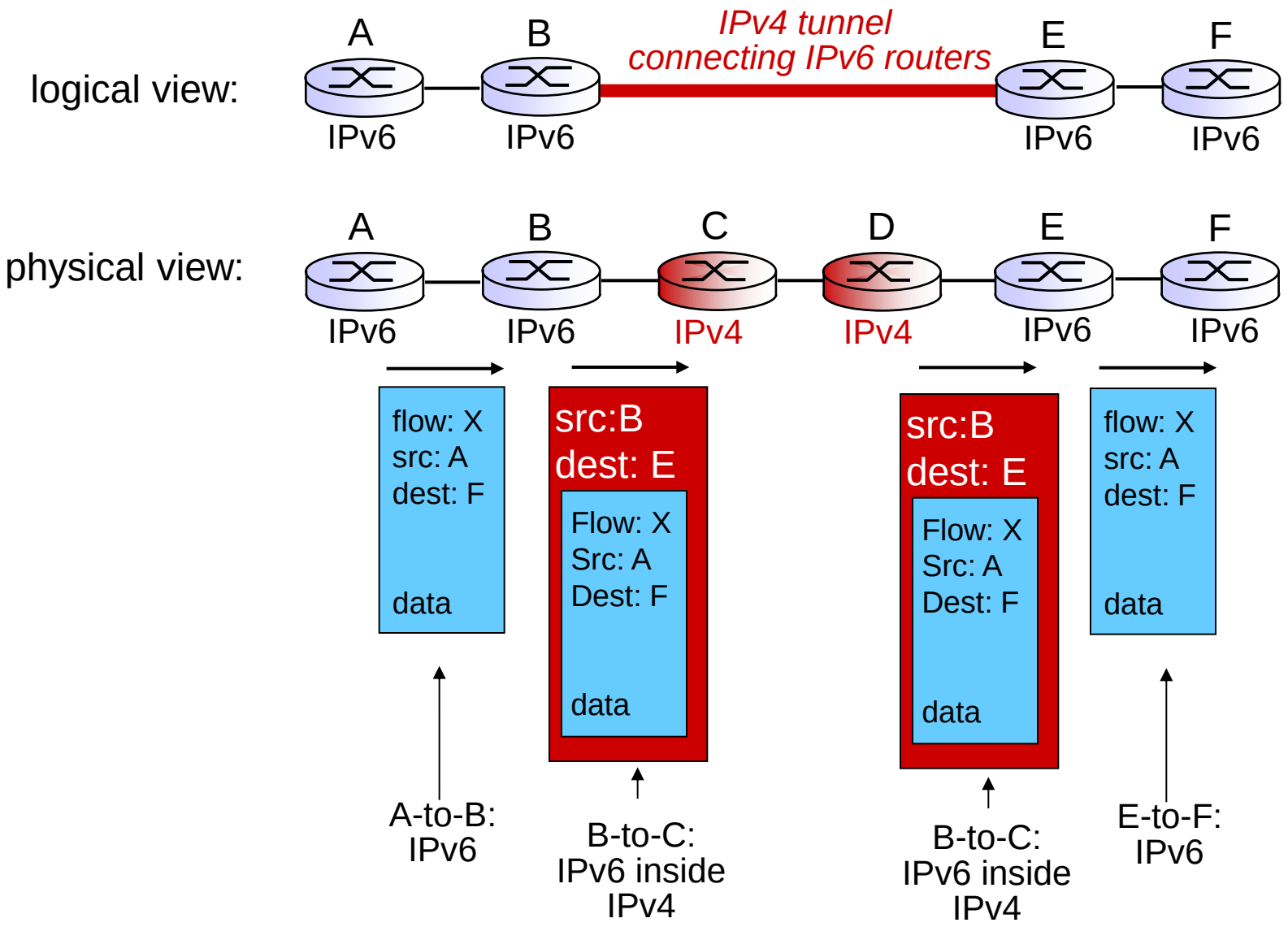
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



# Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

# Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

# IPv6: adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
  - *Why?*