

# Chapter 4

## Network Layer:

## The Data Plane

---

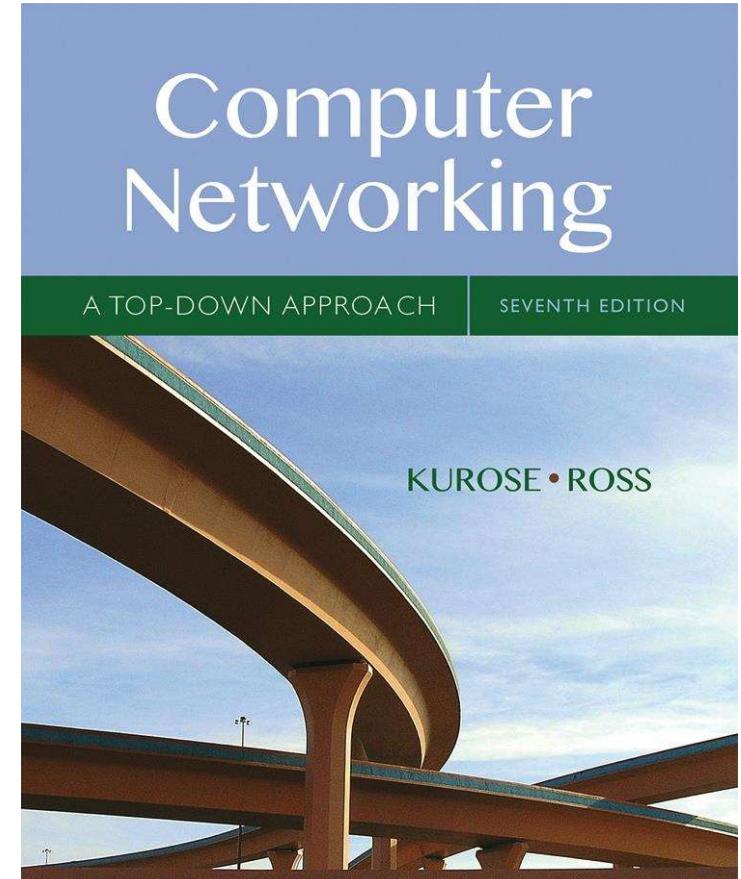
### A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016  
© J.F Kurose and K.W. Ross, All Rights Reserved



*Computer  
Networking: A Top  
Down Approach*

7<sup>th</sup> edition

Jim Kurose, Keith Ross  
Pearson/Addison Wesley  
April 2016

# Chapter 4: Network Layer

## Chapter goals:

- 理解网络层服务原理，主要关注数据面
  - 网络层服务模型
  - 网络层上的重要功能：转发和选路
  - 路由器工作原理
  - 编址
  - 因特网架构
- 因特网的网络层（数据面）
  - IP协议
  - NAT，中间件

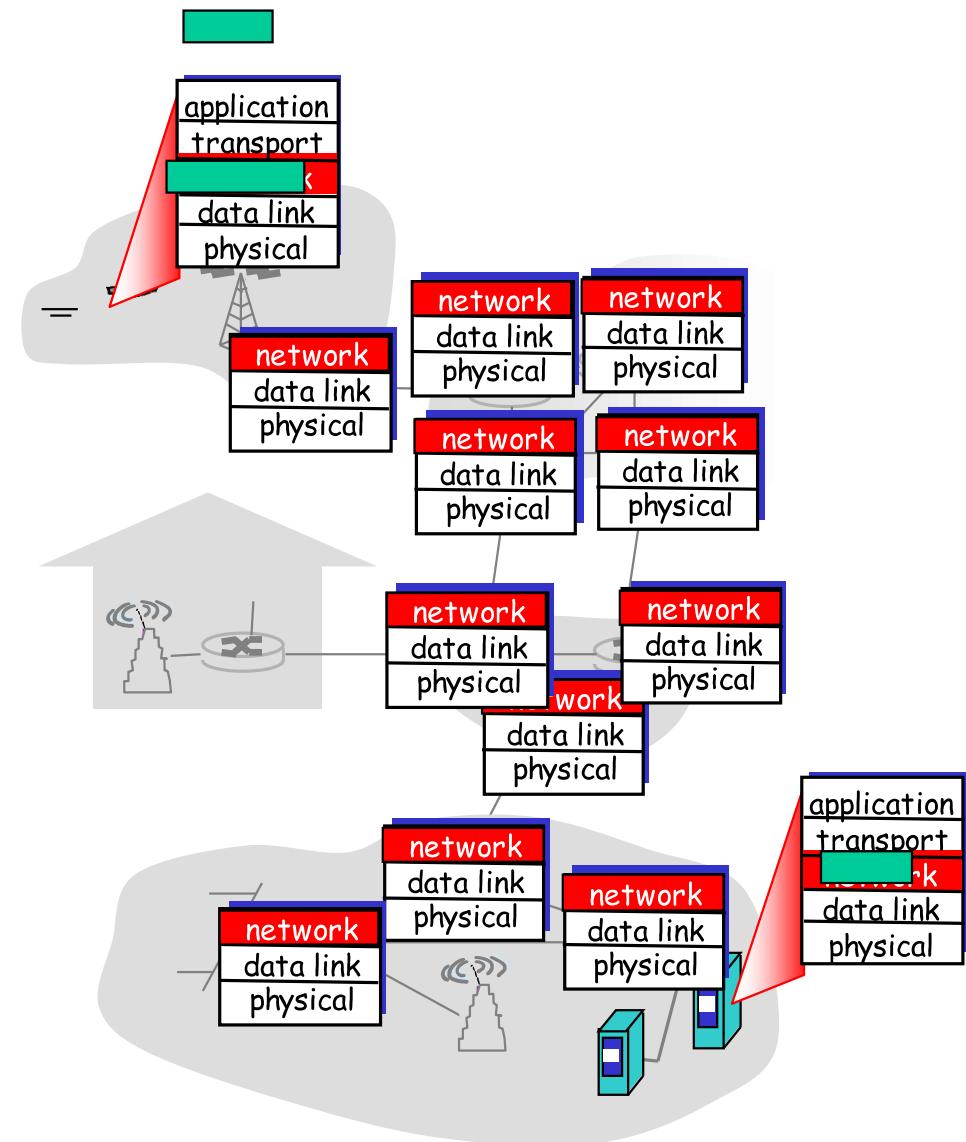
# Chapter 4: Network Layer

- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6
- 4.4 Generalized Forward and SDN
  - match
  - action
  - OpenFlow examples of match-plus-action in action



# 网络层服务

- 网络层为传输层提供主机到主机的通信服务
- 每一台主机和路由器都运行网络层协议
- 发送终端：将传输层报文段封装到网络层分组中，发送给边缘路由器
- 路由器：将分组从输入链路转发到输出链路
- 接收终端：从边缘路由器接收分组，取出报文段交付给传输层



# 网络层的两个主要功能

## 网络层的功能

- **选路**: 确定去往目的路由器的路由
- **转发**: 路由器根据选定的路由, 将分组从输入端口转移到输出端口



forwarding

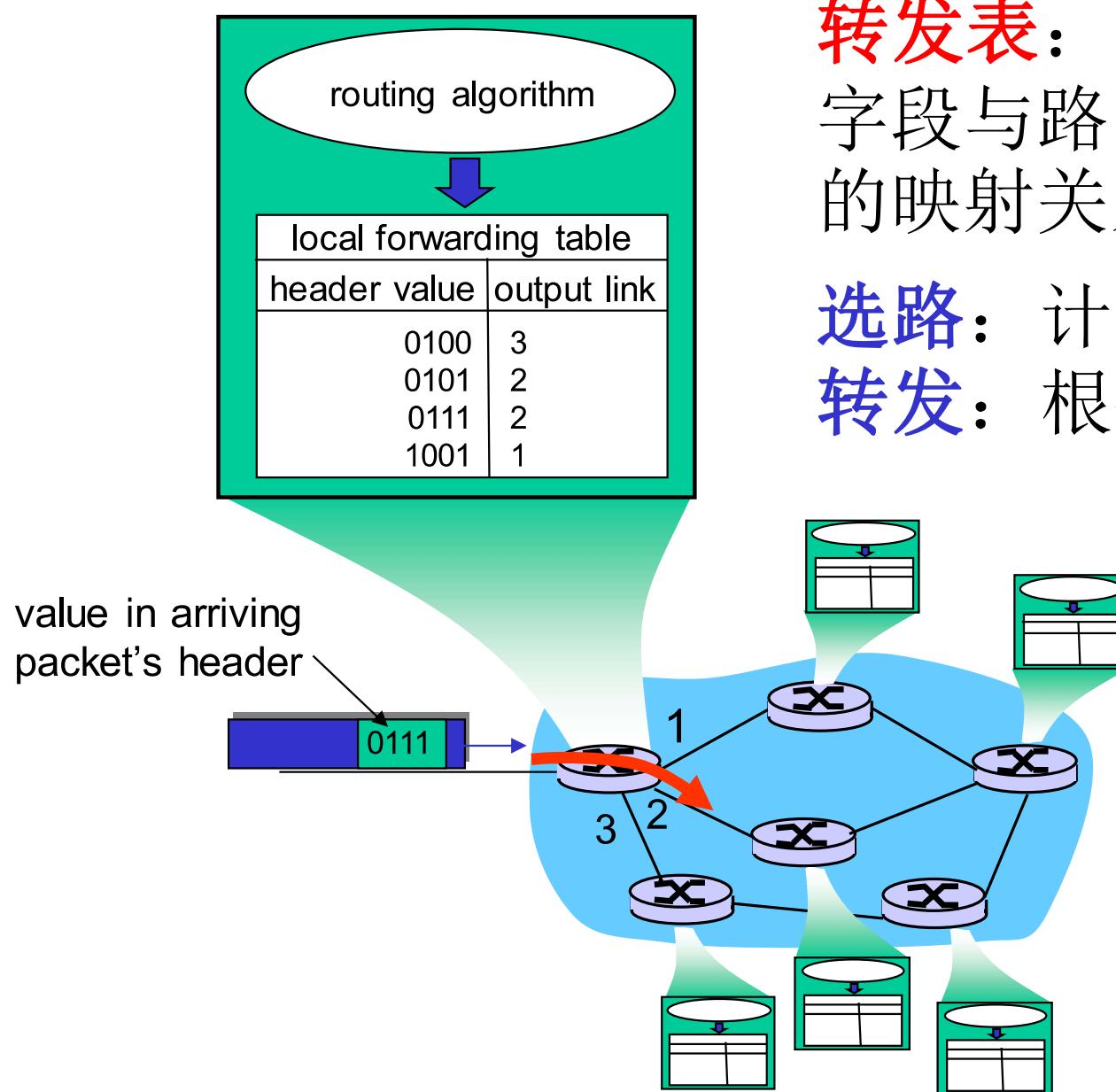
## 交通出行的类比

- **选路**: 规划到目的地的路线
- **转发**: 在到达路口时, 根据选好的路线转移到下一个路段



routing

# 选路和转发的关系



**转发表**: 记录分组头中某个字段与路由器输出端口之间的映射关系

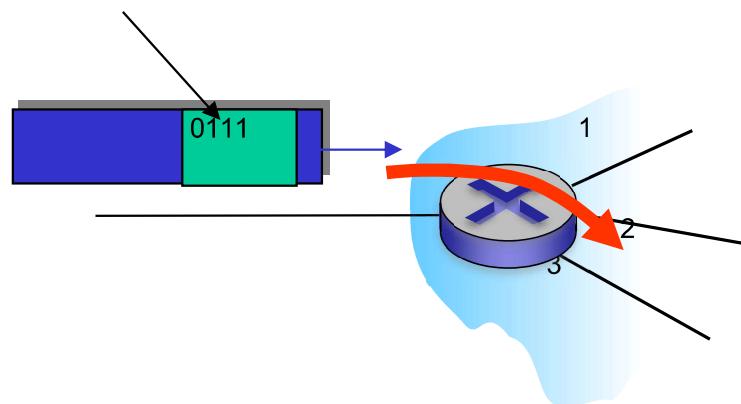
## 选路：计算转发表

**转发：**根据转发表转运分组

# 网络层：数据面和控制面

## 数据面 (Data plane)

- 执行数据传输的功能属于数据面
- 转发是数据面功能，在路由器内部实施分组转运
- 是路由器本地功能



## 控制面 (Control plane)

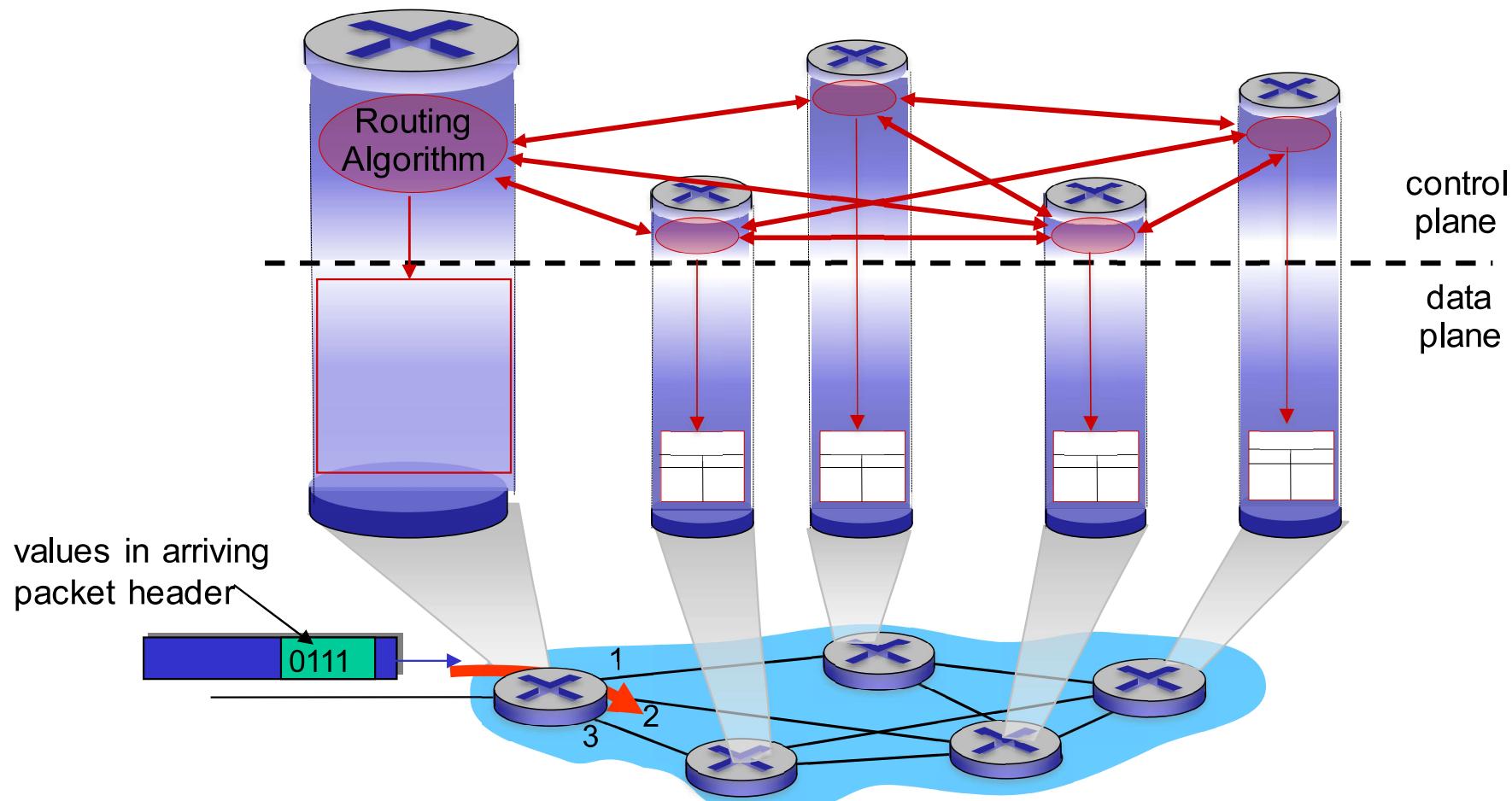
- 控制数据传输的功能属于控制面
- 选路是控制面功能，确定分组如何去往目的节点
- 是网络范围的功能

## 两种控制面实现方法

- 传统寻路算法：在路由器中实现
- 软件定义网络：在服务器中实现

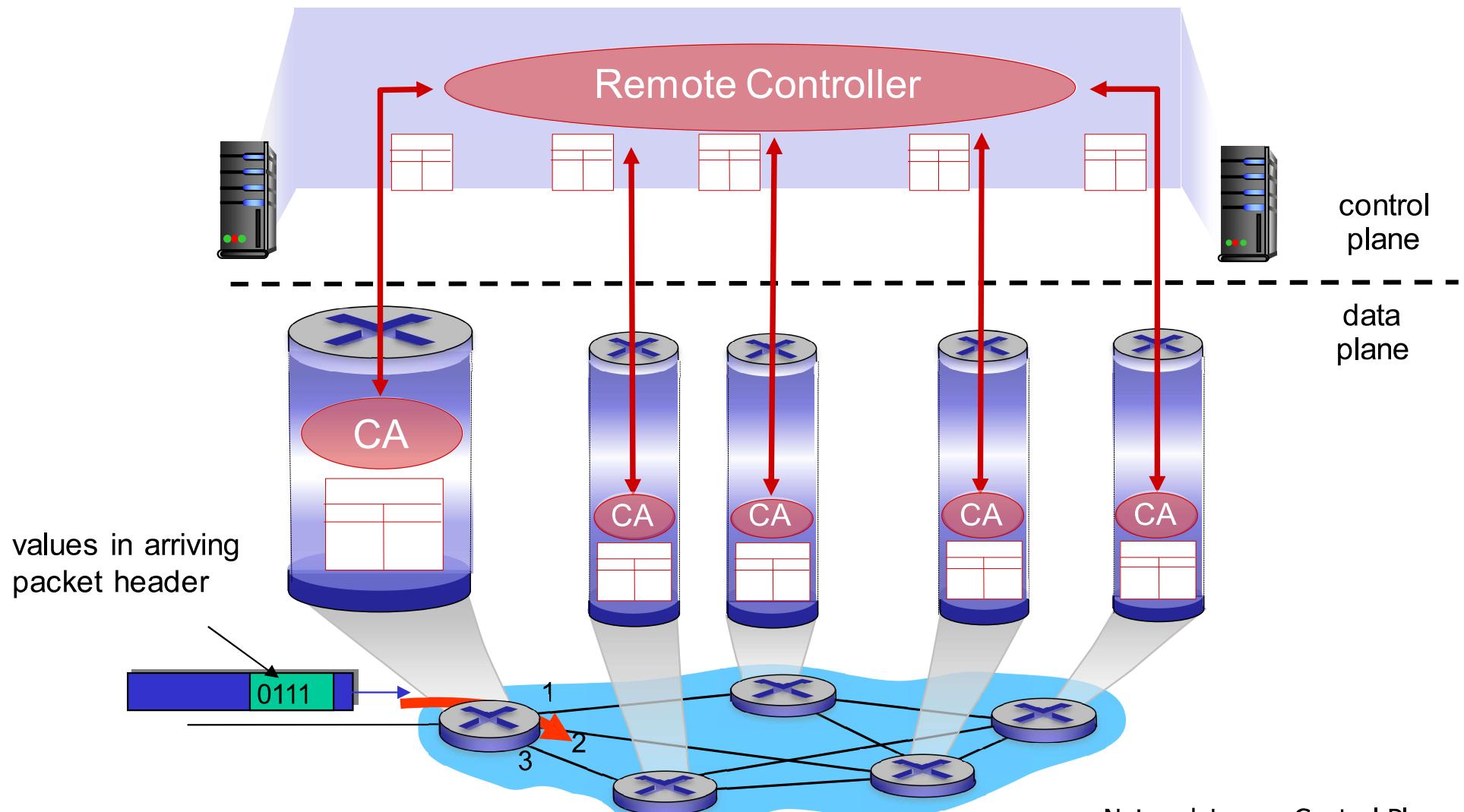
# Per-router control plane

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

网络服务模型：

- 定义了分组在发送终端与接收终端之间的传输特性

可能的网络服务：

- 保证交付
- 具有时延上界的保证交付
- 有序分组交付
- 保证最小带宽
- 安全性

# 网络层服务模型举例

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	no	no	no	no	no (inferred via loss)
ATM	CBR	恒定速率	yes	yes	yes	无拥塞发生
ATM	ABR	最小速率	no	yes	no	yes

- 不同架构的网络提供的网络层服务可能不同
- 同一个网络也可以提供不同的网络层服务

# Chapter 4: Network Layer

- 4.1 Introduction
- **\* Virtual circuit and datagram networks**
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

# 面向连接的服务，无连接服务

两种基本的网络类型：

数据报网络：提供网络层无连接服务

虚电路网络：提供网络层面向连接服务

## □ 网络层服务

- 主机-主机
- 一个网络不能同时提供无连接服务和面向连接的服务
- 在网络核心实现

## □ 传输层服务

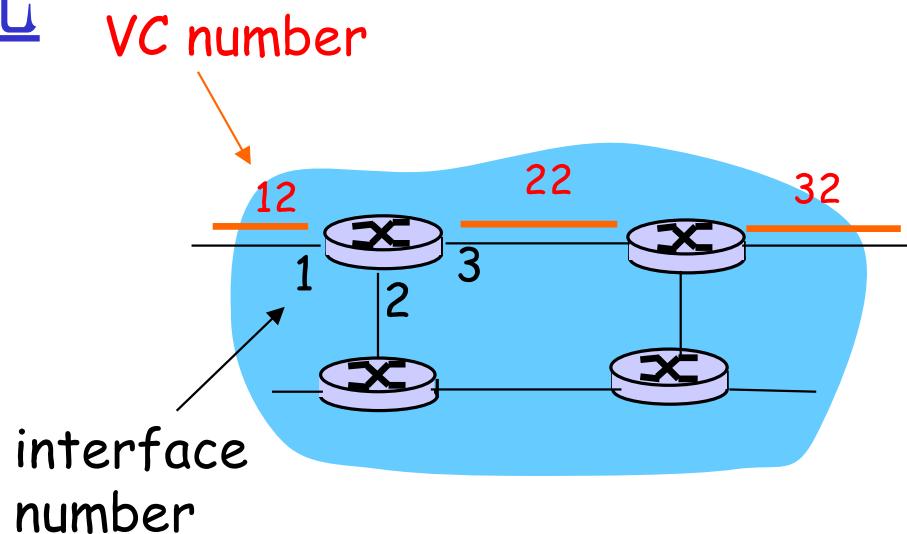
- 进程-进程
- 可同时提供无连接服务和面向连接的服务
- 在网络边缘实现

# 虚电路 (Virtual circuits)

- 虚电路网络在网络层上使用连接，这些网络层连接称为虚电路
- 虚电路是一条端到端路径：
  - 传输分组前需建立虚电路，传输结束后应拆除虚电路
  - 所有分组在这条虚电路上传输（分组传输是有序的）
  - 如果将路由器资源（带宽、缓存等）预先分配给虚电路，则虚电路可以提供可预期的网络服务
- 建立虚电路的本质：
  - 预先选好从源主机到目的主机的路径，此后分组仅沿选好的路径传输，是否分配资源是可选的



# 虚电路 (VC) 实现

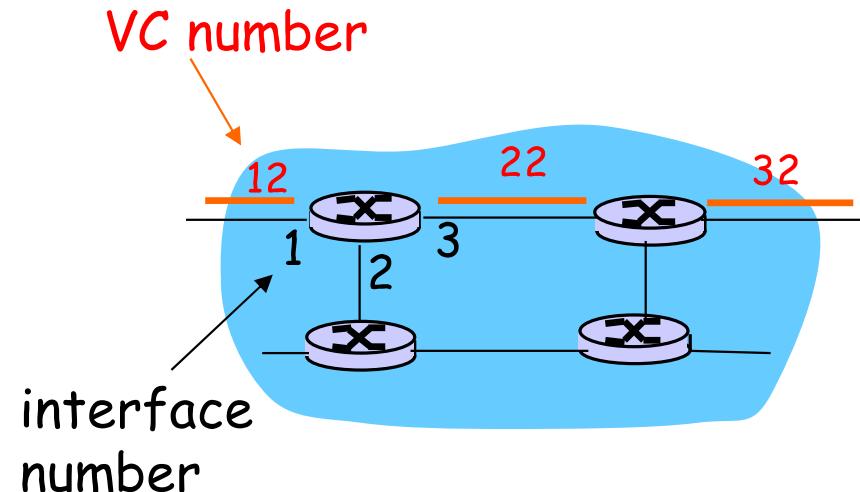


一条虚电路由以下几部分组成：

1. 从源主机到目的主机的端到端路径
2. 途经每条链路时的虚电路号（用于区分经过该链路的不同虚电路，仅有本地意义）
3. 沿途每个路由器中的转发表项：<进入端口，进入VC号> --> <输出端口，输出VC号>



# 虚电路转发



左上角路由器中的转发表

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

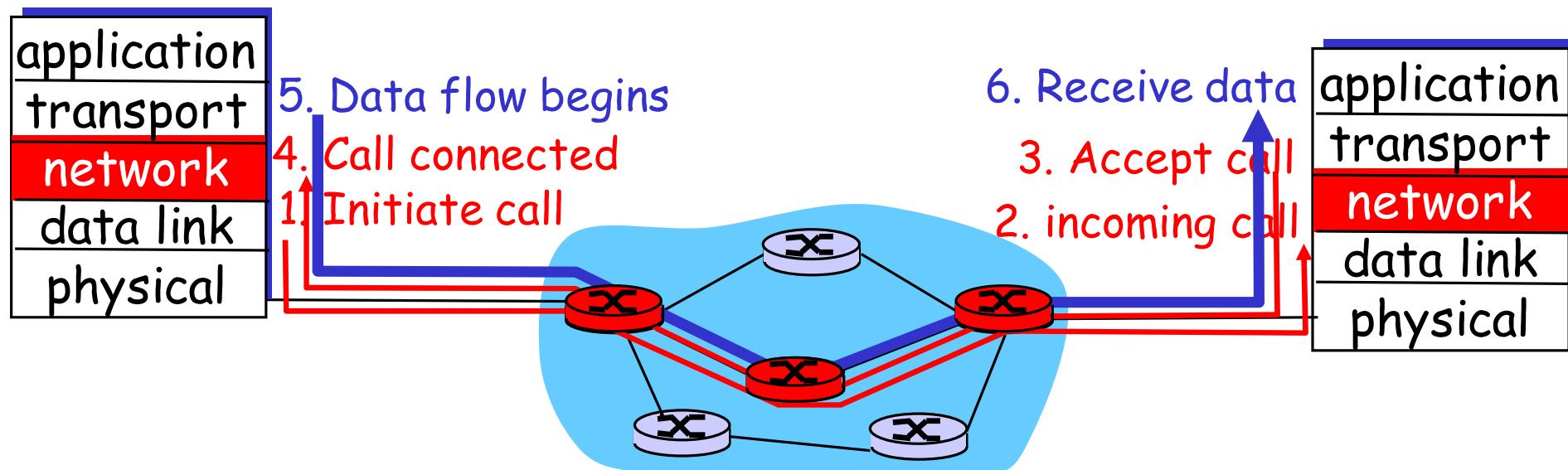
- 分组携带**VC**号，路由器利用输入端口和**VC**号查找转发表
- 转发前，路由器使用输出链路上的**VC**号替换分组中的**VC**号

注意：在虚电路上传输分组时，分组只需携带**VC**号，不需携带目的地址



# 虚电路: 信令 (signaling) 协议

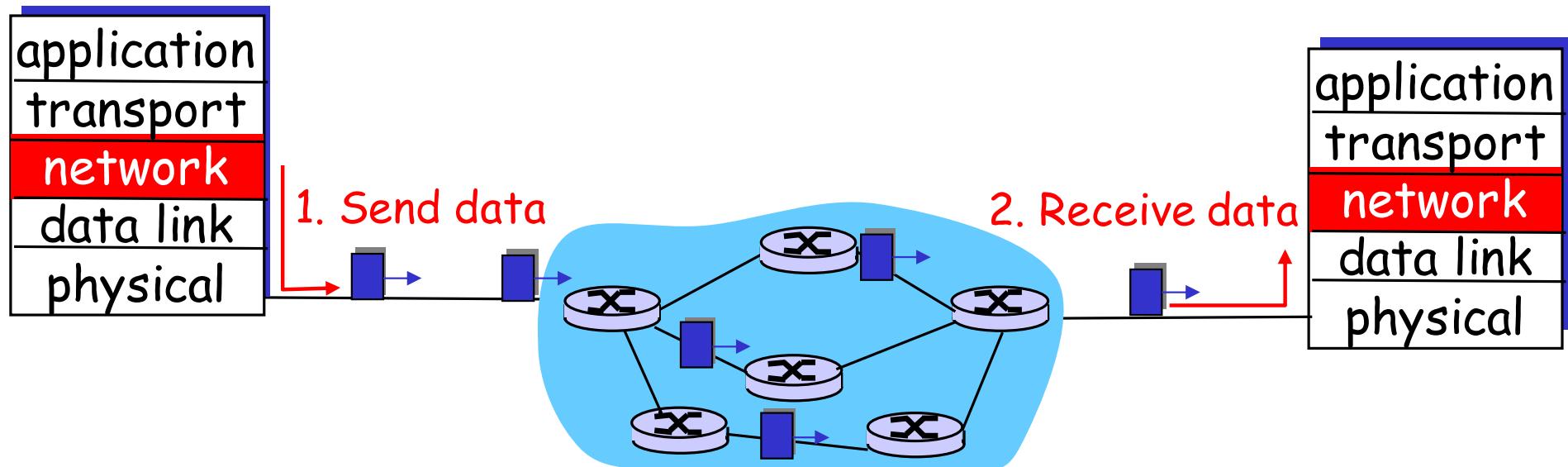
- 信令报文: 专门用于建立、维护、拆除虚电路的控制报文
- 信令协议: 交换信令报文的协议





# 数据报网络

- 分组携带目的主机地址，路由器按目的地址转发分组
- 路由器中的转发表记录**目的地址→输出端口**的映射
- 转发表被选路模块修改，约1~5分钟更新一次
- 同一对主机之间传输的分组可能走不同的路径，从而可能重排序



# Datagram or VC network: why?

## Internet (数据报网络)

- 为计算机通信而设计：
  - 早期的网络应用对网络服务没有严格要求（弹性应用）
  - 用户免费使用网络
- 终端（计算机）具有智能
  - 可将复杂的工作（如差错控制）推到网络边缘，以保持网络核心简单

## ATM (虚电路网络)

- 由电信网发展而来
  - 注重用户体验，追求高质量服务（用户付费了）
- 终端无智能或很少智能
  - 复杂工作由网络完成，以保持终端简单

# Datagram or VC network: why?

□ 数据报网络提供最小服务的好处：

- 可运行在各种链路上
- 为传输层服务的设计增加了灵活性，可简、可繁

□ 符合当初因特网设计的原则：

- 网络只提供尽力而为的服务
- 复杂的工作推到网络边缘（增加新服务只涉及终端）

□ 这两个好处带来了因特网今天的发展：

- **IP over everything** (IP可以运行在任何物理网络上)
- **Everything over IP** (任何应用可以运行在IP上)

# Chapter 4: Network Layer

- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

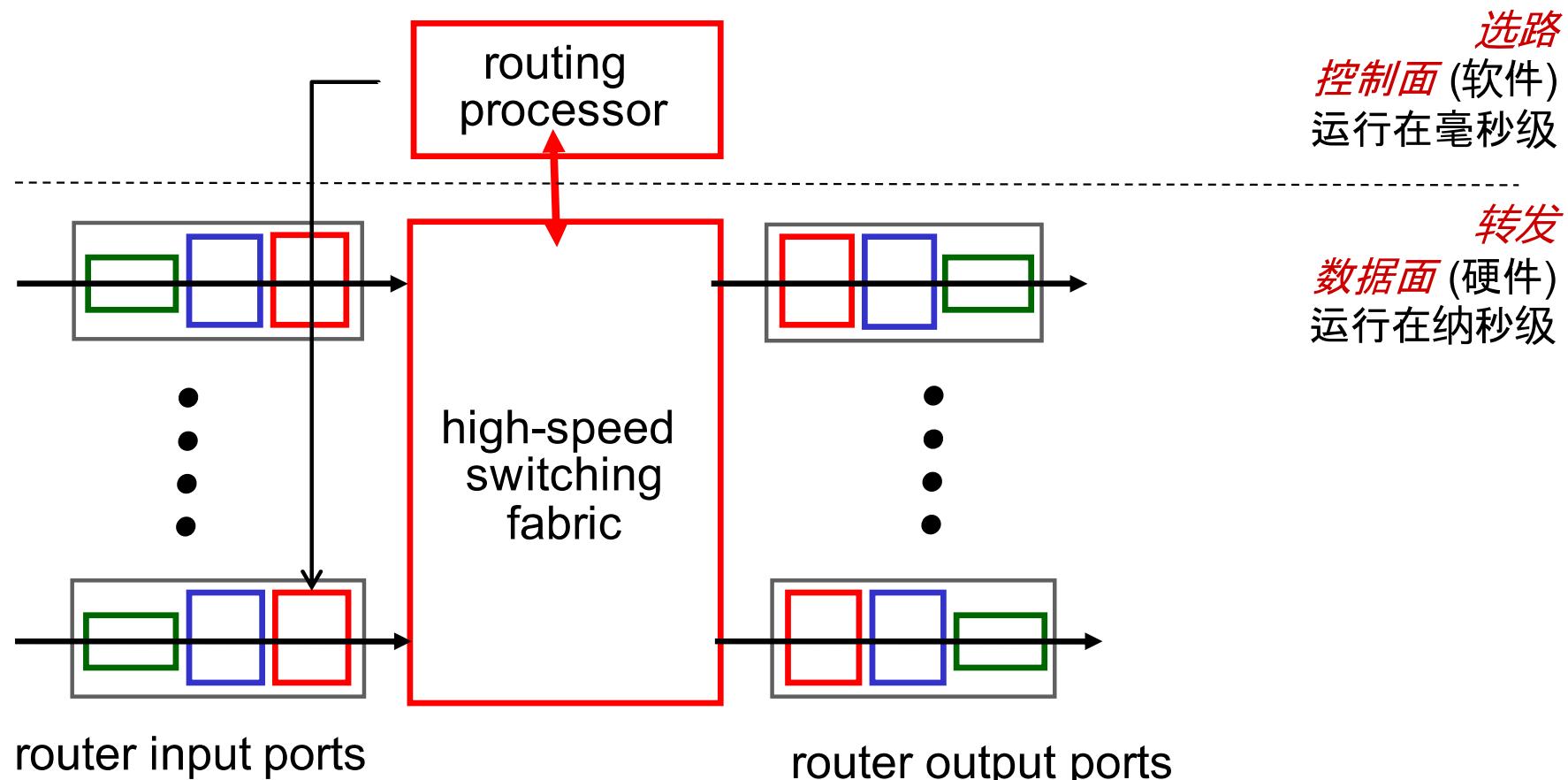
## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

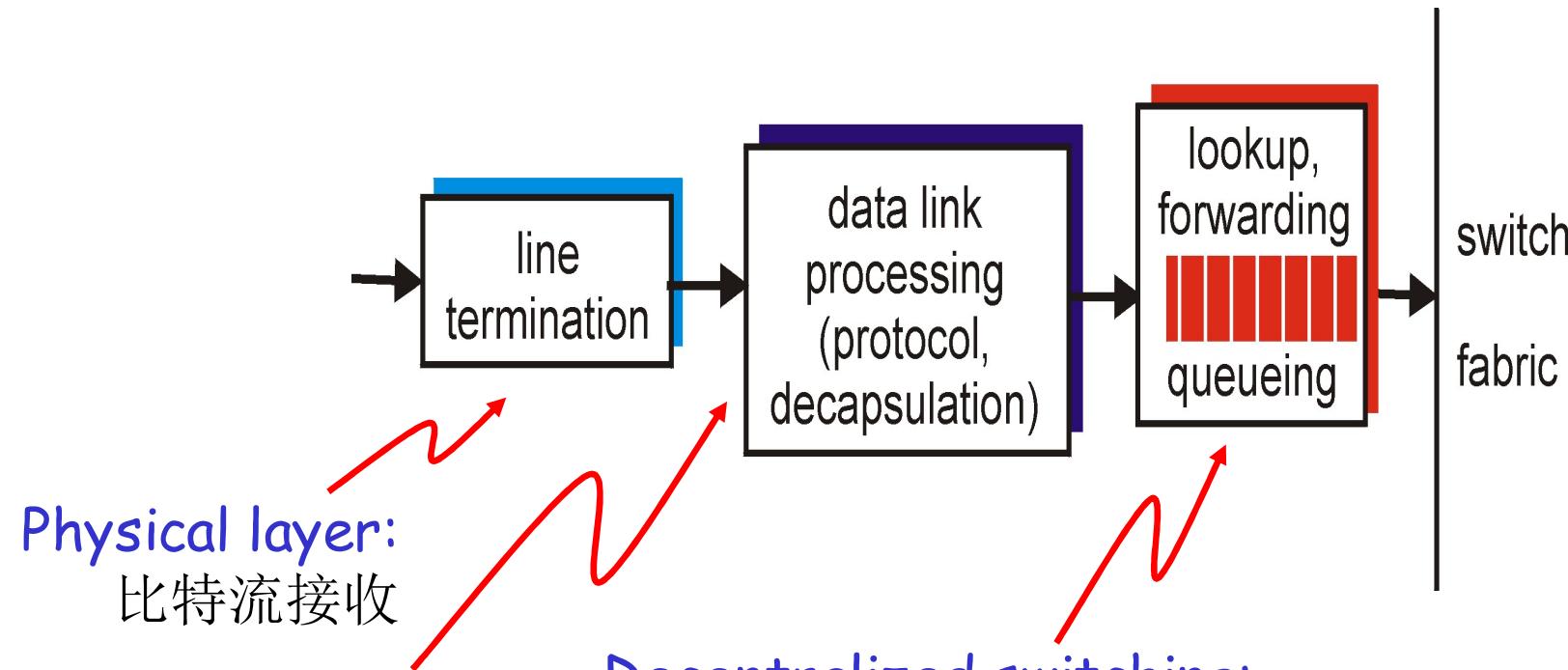
# 路由器架构概述

路由器的两个主要功能：

- 选路：运行选路协议，计算转发表
- 转发：依据转发表，从输入链路到输出链路转发数据报



# 输入端口功能



## Data link layer:

- 从比特流中提取帧
- 处理帧
- 提取IP数据报

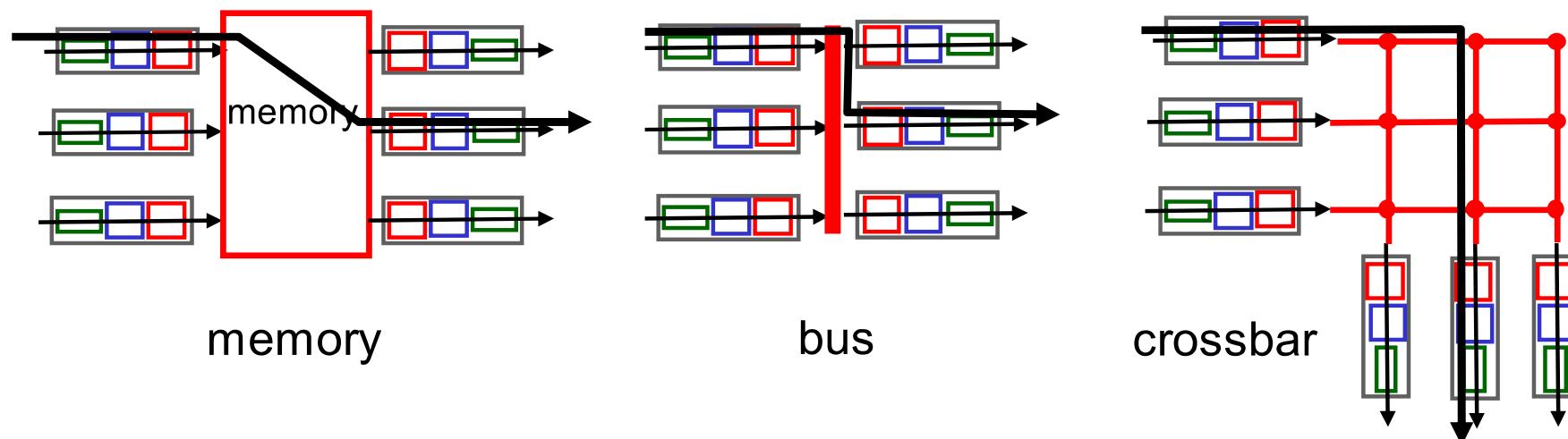
## Decentralized switching:

- 查表**: 每块线卡 (line card) 上都有转发表的一个镜像，查表仅在本地进行
- 排队**: 当交换结构阻塞时，分组需在此排队
- 转发**: 通过交换结构将分组发送到输出端口 (这个过程也称为交换，switch)

性能要求：以“线速”完成输入端口处理

# 交换结构

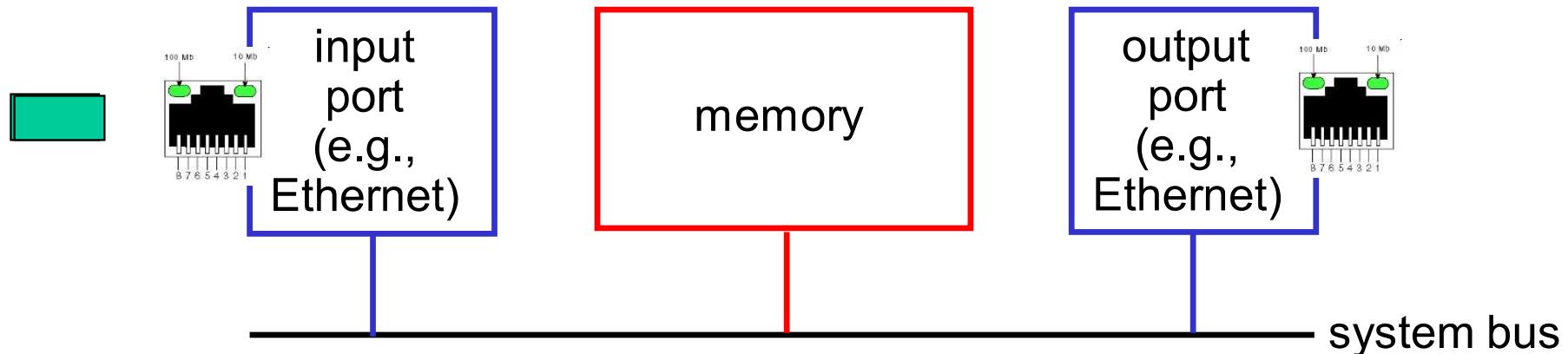
- 路由器中的互联网络，用于在输入端口、输出端口和选路处理器之间转运分组
- 交换速率：
  - 通常是输入/输出链路速率的若干倍
- 三种类型的交换结构



# 通过内存交换

第一代路由器:

- 即传统计算机，在**CPU**的直接控制下完成交换
- 数据包拷贝到系统内存中进行交换
- 交换速率受限于内存带宽：每个数据包穿过系统总线2次

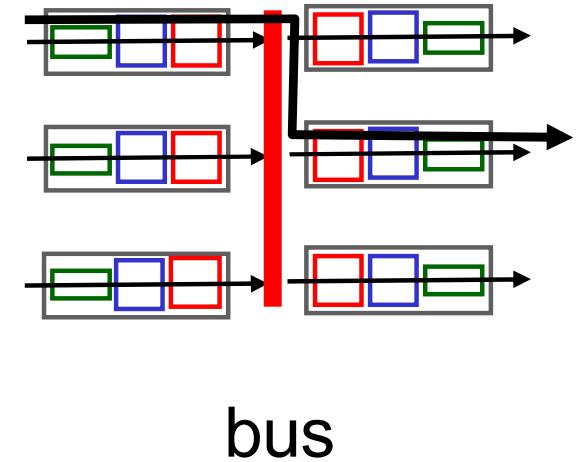


现代路由器的做法:

- 每个端口使用一个内存接口硬件连接到存储系统
- 一个控制器硬件在端口之间传输控制消息（不需要**CPU**参与）
- 输入端口将一个包放入内存后，接口硬件通过控制器向输出端口发送一个消息，输出端口从内存指定位置读取包，发回响应消息

# 通过总线交换

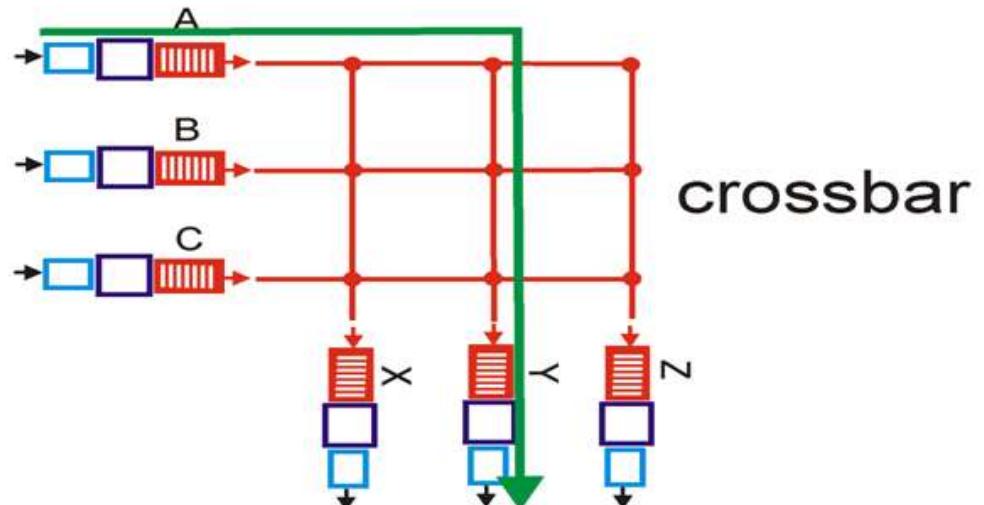
- 数据包通过一条共享总线，从输入端口缓存转移到输出端口缓存
- 每个输入和输出端口通过一个接口硬件连接到总线上，每个端口被分配一个内部标签
- **总线竞争：**
  - 总线协议防止多个端口同时传输，比如，采用时分多路复用的方法
  - 各个输入端口在总线上轮流广播分组，每个输出端口根据分组携带的内部标签接收发给本端口的分组
- 交换速率受限于总线带宽



Cisco 5600采用32Gbps总线，满足接入路由器和企业路由器的交换要求

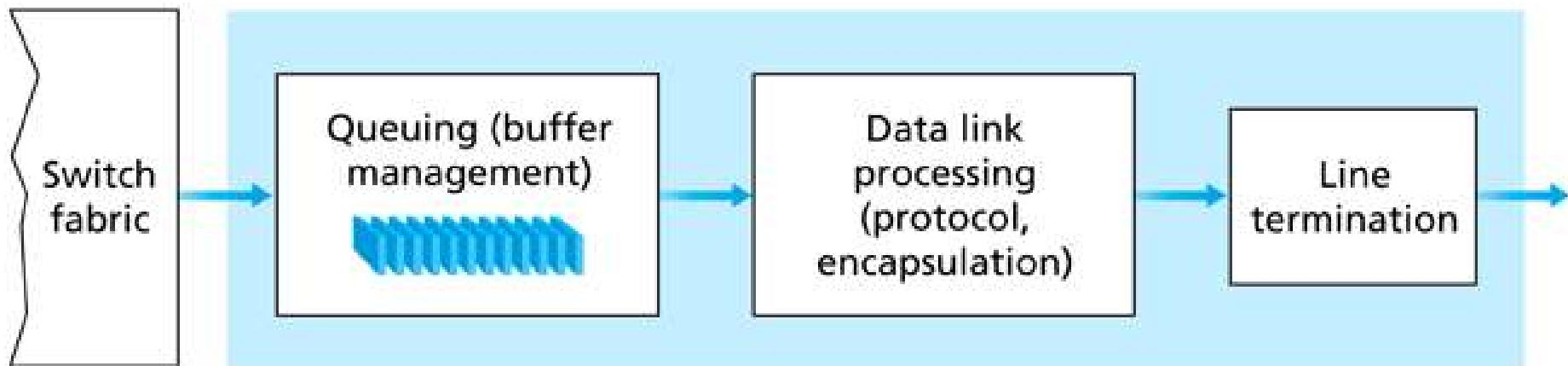
# 通过互联网络交换

- 交换结构控制器通过控制交叉点的开、闭，在输入端口与输出端口间建立内部专用电路
- 多对端口间可以并行传输
- 分阻塞型与非阻塞型，阻塞型互联网络会产生阻塞
- Cisco 12000通过互联网络获得60 Gbps的交换速度



- 先进设计：
  - 将输入端口和输出端口连接到N个并行运行的交换结构
  - 将分组划分成若干个固定长度的信元（cell），同时送入交换结构
  - 信元离开交换结构后再组装成分组

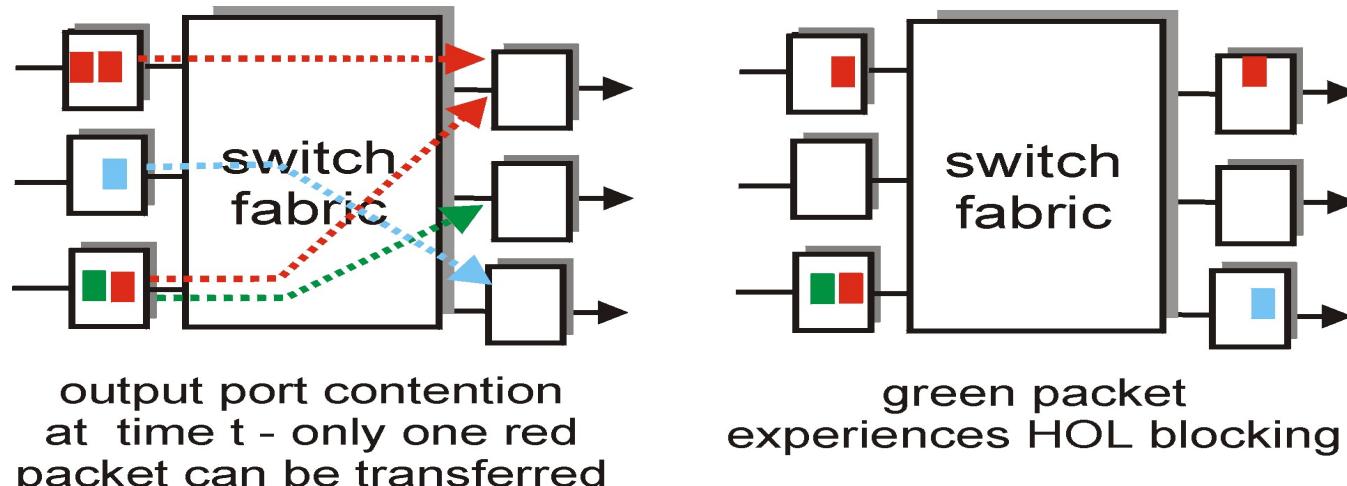
# 输出端口功能



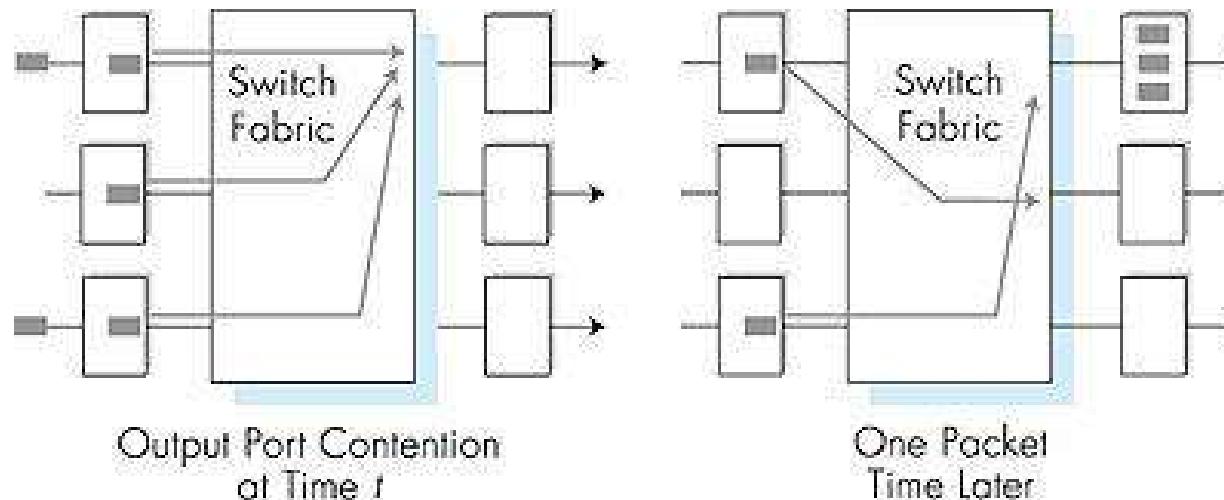
- 网络层处理：
  - 组装：若需要，将交换结构输出的信元组装成分组
  - 排队：若输出端口来不及发送，分组在此排队
  - 调度：输出端口每次选择一个分组发送
- 链路层处理：执行链路层协议，封装
- 物理层处理：将比特流转换成物理信号

# 输入端口排队与丢包

- 当交换结构不能及时将输入端口的分组转移到输出端口时，输入端口处形成排队
- 排队带来的问题：
  - 队头阻塞：队头分组阻塞其后分组的转发
  - 丢包：当输入队列溢出时，发生丢包
- 当交换结构速率至少为端口速率的n倍时（n为输入端口数），可以消除输入端口的排队，但路由器成本提高了



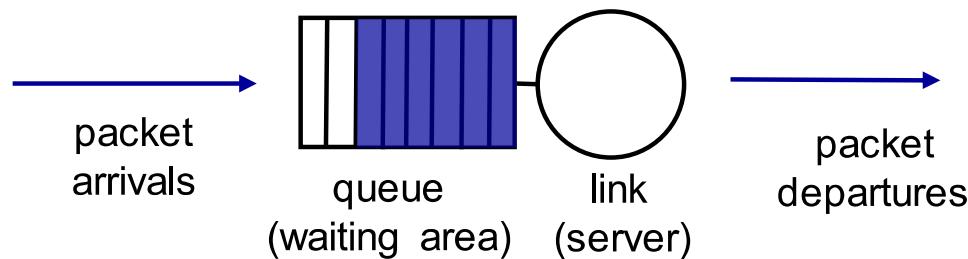
# 输出端口排队与丢包



- 多个输入端口同时向一个输出端口发送时，在输出端口形成排队
- 当输出队列满时，发生丢包
- **输出端口排队是不可避免的**，设置多大的输出队列是一个问题：
  - 增大输出队列：可以减少丢包的发生，但会增加内存消耗，并增大分组延迟（延迟太大的分组最终被重传，浪费资源）
  - **输出队列并不是越长越好！**

# 调度策略

- 调度：输出端口选择下一个要发送的分组
- 先来先服务：
  - 按照分组到达输出队列的次序发送
- 分组丢弃策略，当输出队列满时丢弃哪个分组？
  - 弃尾：丢弃到来的分组
  - 按照优先级丢弃：低优先级分组
  - 随机丢弃：随机选择一个分组丢弃，如Random Early Detection (RED)

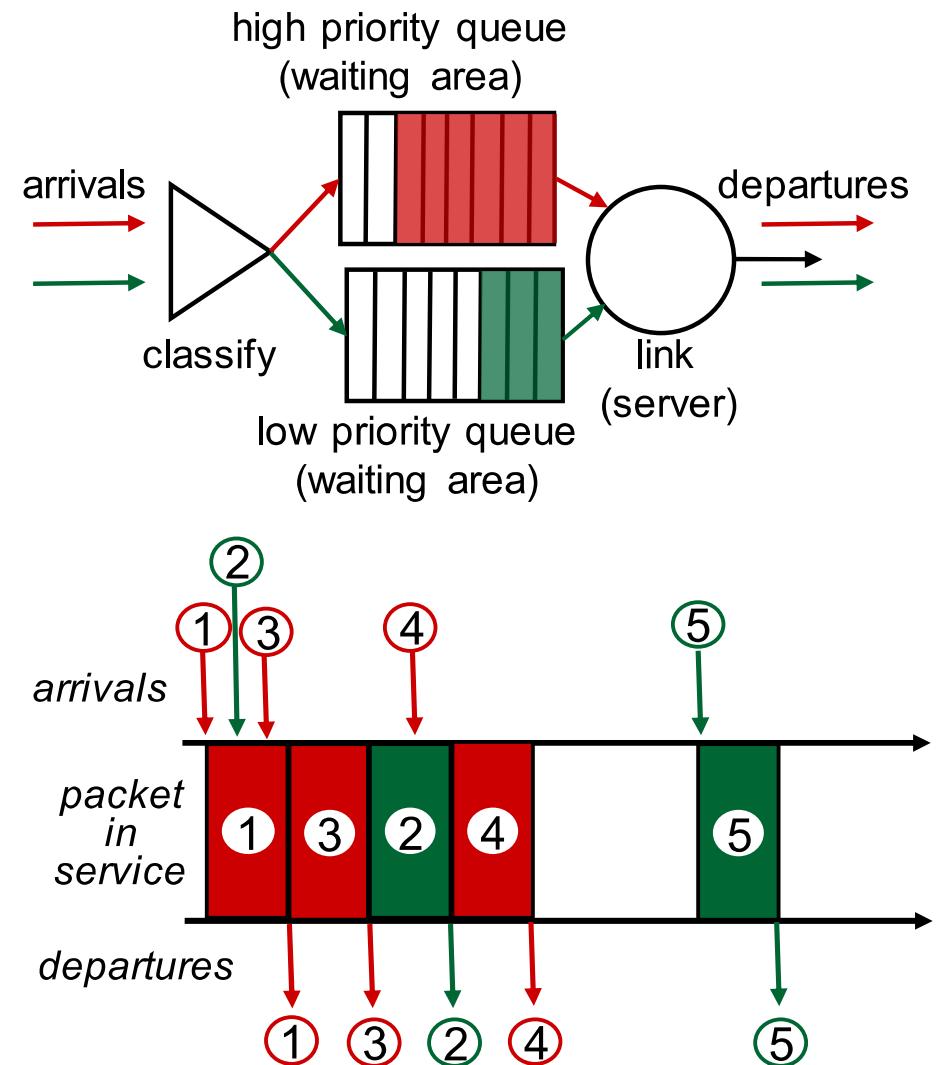


# 随机早检测 (RED)

- 在队列满之前就开始丢弃分组，设计为和TCP拥塞控制机制一起使用
- 丢弃策略：
  - 路由器在每个端口上记录输出队列的平均长度：
$$\text{AvgLen} = (1 - \text{Weight}) \times \text{AvgLen} + \text{Weight} \times \text{SampleLen}$$
  - 当平均队列长度达到第一个阈值 $\text{min}_{\text{th}}$ 时，按照丢弃概率 $p$ 丢弃到来的分组
  - 当平均队列长度达到第二个阈值 $\text{max}_{\text{th}}$ 时，丢弃每一个到达的分组
  - 概率 $p$ 是平均队列长度和上一次丢弃距当前时间的函数，分组队列长度越大，丢弃间隔越大， $p$ 越大

# 调度策略

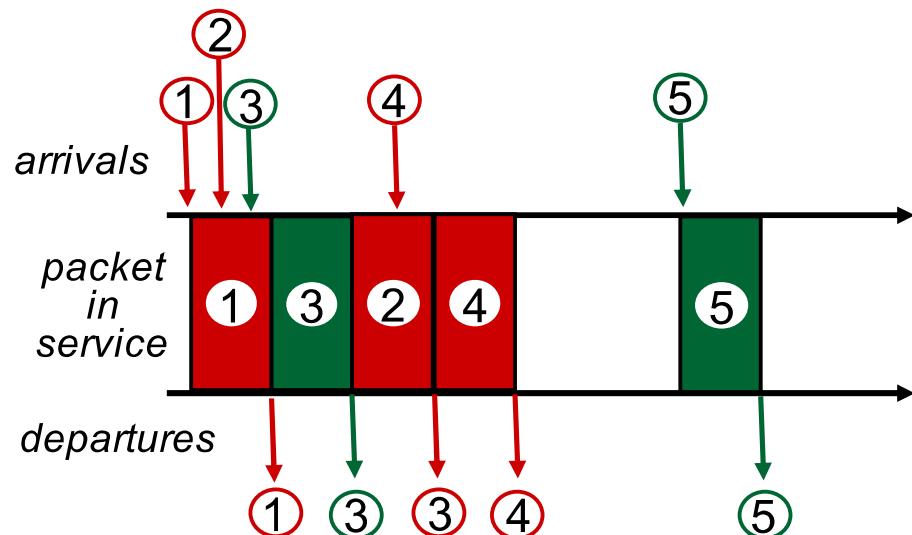
- 优先级调度：
  - 严格按优先级发送分组
  - 同一优先级的分组按先来无服务的顺序发送
- 设置多个优先级类，每个分组被标记一个优先级：
  - 可以根据分组头中的某些域，如IP地址、端口号等设置优先级
- 非抢占式优先级排队：
  - 分组一旦开始传输，就不能被中断



# 调度策略

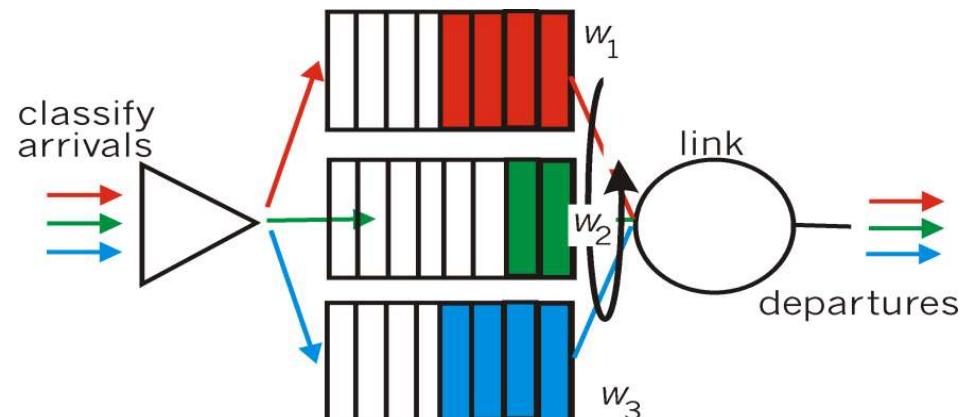
## □ 轮询调度：

- 在几个队列之间轮流提供服务，每次选择一个队头分组发送



## □ 加权公平排队：

- 每个队列分配一个带宽权重（比例）
- 按照每个队列的带宽权重，选择一定数量的分组发送



# Chapter 4: Network Layer

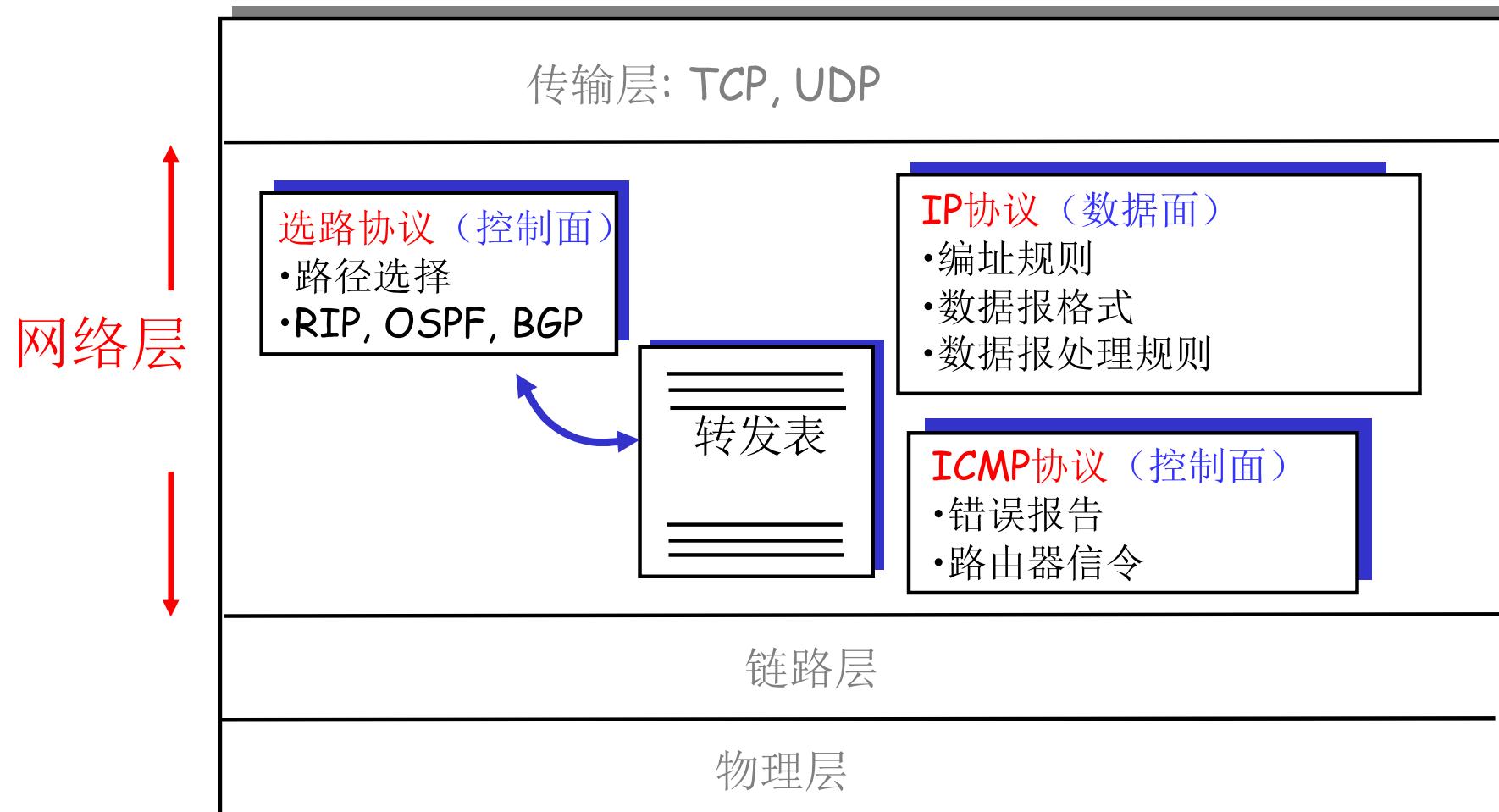
- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

# 因特网的网络层

主机和路由器的网络层功能包括：



# Chapter 4: Network Layer

- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

# IP只提供最小的传输服务

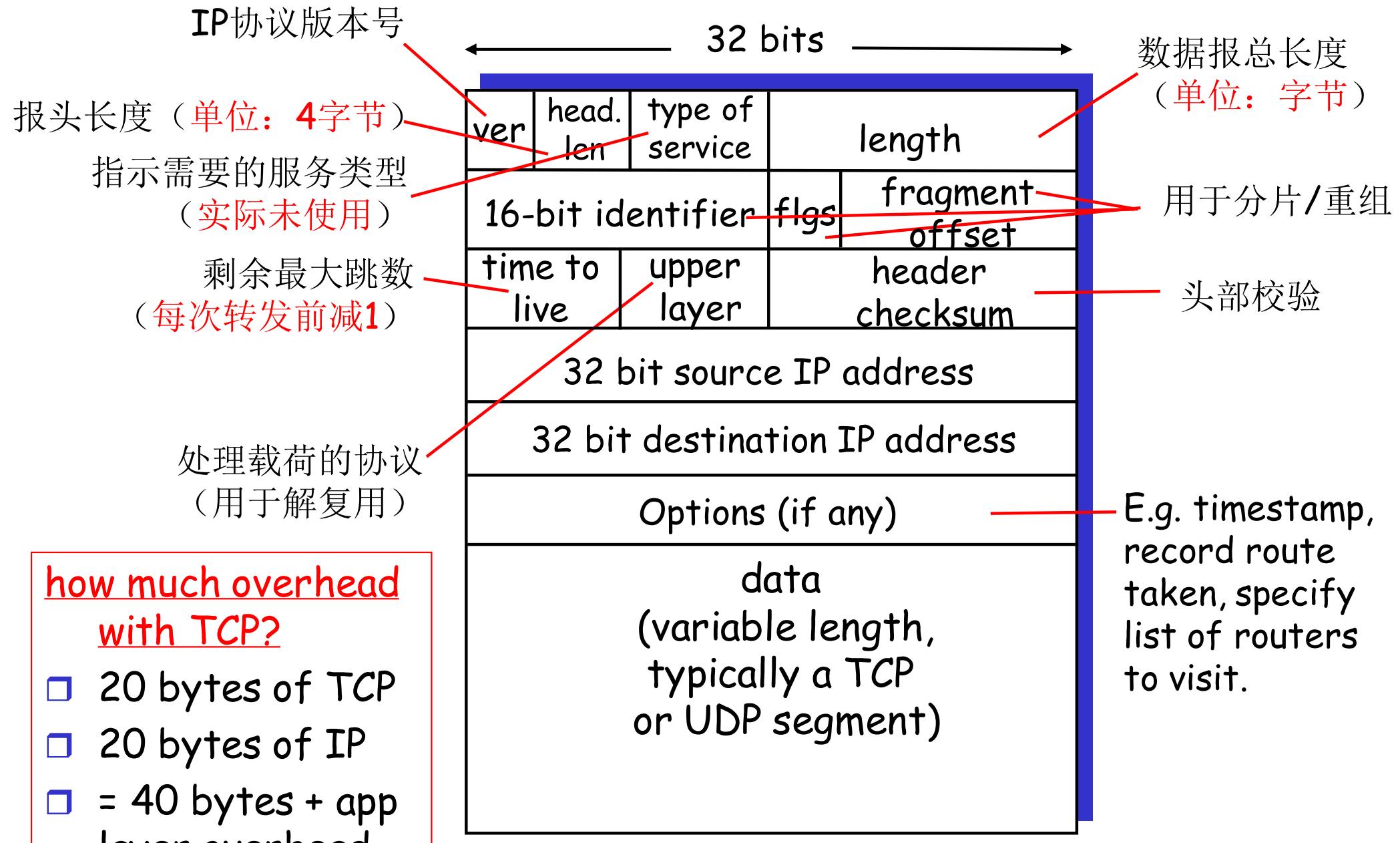
- IP提供的服务：

- 将数据报交付到目的地址和目的协议

- 尽管IP不提供任何服务承诺，但仍**尽最大努力**解决分组在网络中传输时可能遇到的一些问题：

- 数据报过大无法通过某个中间网络（措施：对数据报进行分片）
  - 数据报可能因寻路错误在网络中循环（措施：设定数据报的最大转发次数）
  - 报头出错可能导致误投递（措施：对报头检错）

# IP数据报格式



# Chapter 4: Network Layer

- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

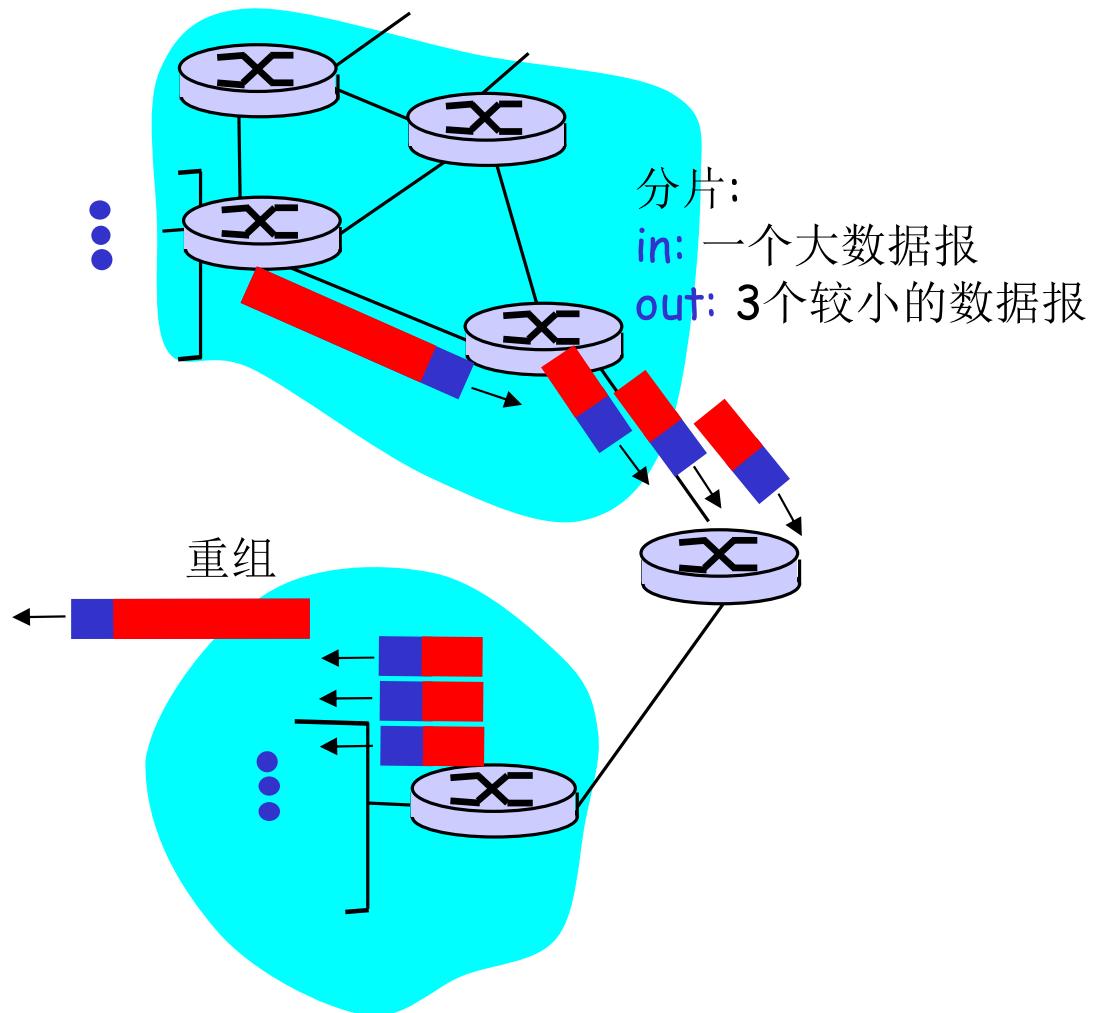
- ## 4.4 Generalized Forwarding and SDN
- match
  - action
  - OpenFlow examples of match-plus-action in action



# IP分片与重组

链路层帧能承载的最大数据字节数称为**MTU**（Max. Transmission Unit）

- 不同类型的链路可能具有不同的**MTU**
- 传输过程中，较大的**IP**数据报可以被分片：
  - 将**数据报载荷**划分为若干较小的数据块，每个数据块封装成一个独立的数据报传输
  - 数据报在传输的过程中可以被多次分片，但**仅在目的主机上重组**



# 分片的报头

- 分片的报头取自原始数据报
- 与分片有关的字段：
  - 标识：每个分片必须携带与原始数据报相同的标识
  - 偏移量：指示分片中的数据在原始数据报载荷中的位置
  - 标志位：
    - MF (more fragments) : 最后一个分片的MF=0，其余分片的MF=1
    - DF (don't fragment) : DF=1表示不允许对数据报分片
- 分片报头中的以下字段需要修改：
  - 总长度，偏移量，MF，TTL，头部检查和

# 分片的数据长度

- 假设原始数据报的报头长度为H，分片的数据长度N，应满足： $H+N \leq MTU$
- 由于偏移量只有13比特，除最后一个分片外，其余分片的数据长度应为8字节的整倍数
- 考虑到分组传输效率，除最后一个分片外，分片的数据长度N 应为满足以上两个条件的最大整数

# 数据报分片的处理过程

- 根据报头长度H和输出线路的MTU，确定分片的最大数据长度N
- 将数据报的载荷划分成长度为N的若干数据块（最后一个数据块可能不足N字节）
- 将原始报头加到每一个数据块的前面，修改报头中的以下字段：
  - 总长度 = H + 数据块长度
  - 最后一个报头的MF位置0，其余报头的MF位置1
  - 偏移量 = 数据块在原始数据报载荷中的字节序号/8
  - TTL=TTL-1
  - 计算头部检查和

# 分片的例子

- 例：要将一个总长度=4000字节的IP包发送到MTU=1500字节的链路上，IP报头长度H=20字节
- 数据块最大长度  $N = 1480$ 字节
- 原始数据报的载荷（3980字节）被分成三个数据块，长度分别为1480字节、1480字节和1020字节

分片序号	总长度	MF	偏移量
1	1500 ( $=1480+20$ )	1	0
2	1500 ( $=1480+20$ )	1	185 ( $=1480/8$ )
3	1040 ( $=1020+20$ )	0	370 ( $=185+185$ )

# 重组

- 将收到的分片重新组装成原始数据报的过程称为重组，重组在目的主机中进行：
  - 收集分片：目的主机使用 <源IP地址，标识> 确定属于同一个数据报的分片
  - 利用最后一个分片计算原始数据报长度：
$$\text{原始数据报长度} = \text{偏移量} \times 8 + \text{分片总长度}$$
$$\text{原始数据报载荷} = \text{偏移量} \times 8 + \text{分片总长度} - \text{报头长度}$$
  - 组装：将各分片中的数据块按照其在原始数据报载荷中的偏移量重组

# 分片的问题

## □ 分片的开销：

- 降低了路由器的吞吐量
- 消耗了目的主机的资源：每个重组的数据报需要一个重组缓冲区和一个重组定时器

## □ 针对分片的DoS攻击：

- 攻击者发送一系列奇怪的分片，消耗目的主机的资源

## □ IPv6取消了路由器分片的功能：

- 源主机发送探测报文，确定路径上的最小MTU
- 源主机构造的数据报大小不超过最小MTU
- 路由器丢弃超大的数据报，并发送错误报告

# 重要知识点

- 分片的方法：
  - 确定分片的大小
  - 修改报头域（尤其是偏移量）
  - 根据最后一个分片计算原始数据报的长度
- 数据报在传输过程中可以被多次分片，但仅在目的主机上组装，为什么？
- 注意报头中几个长度的单位：
  - head length: 4个字节
  - Length: 1个字节
  - fragment offset: 8个字节

# Chapter 4: Network Layer

- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

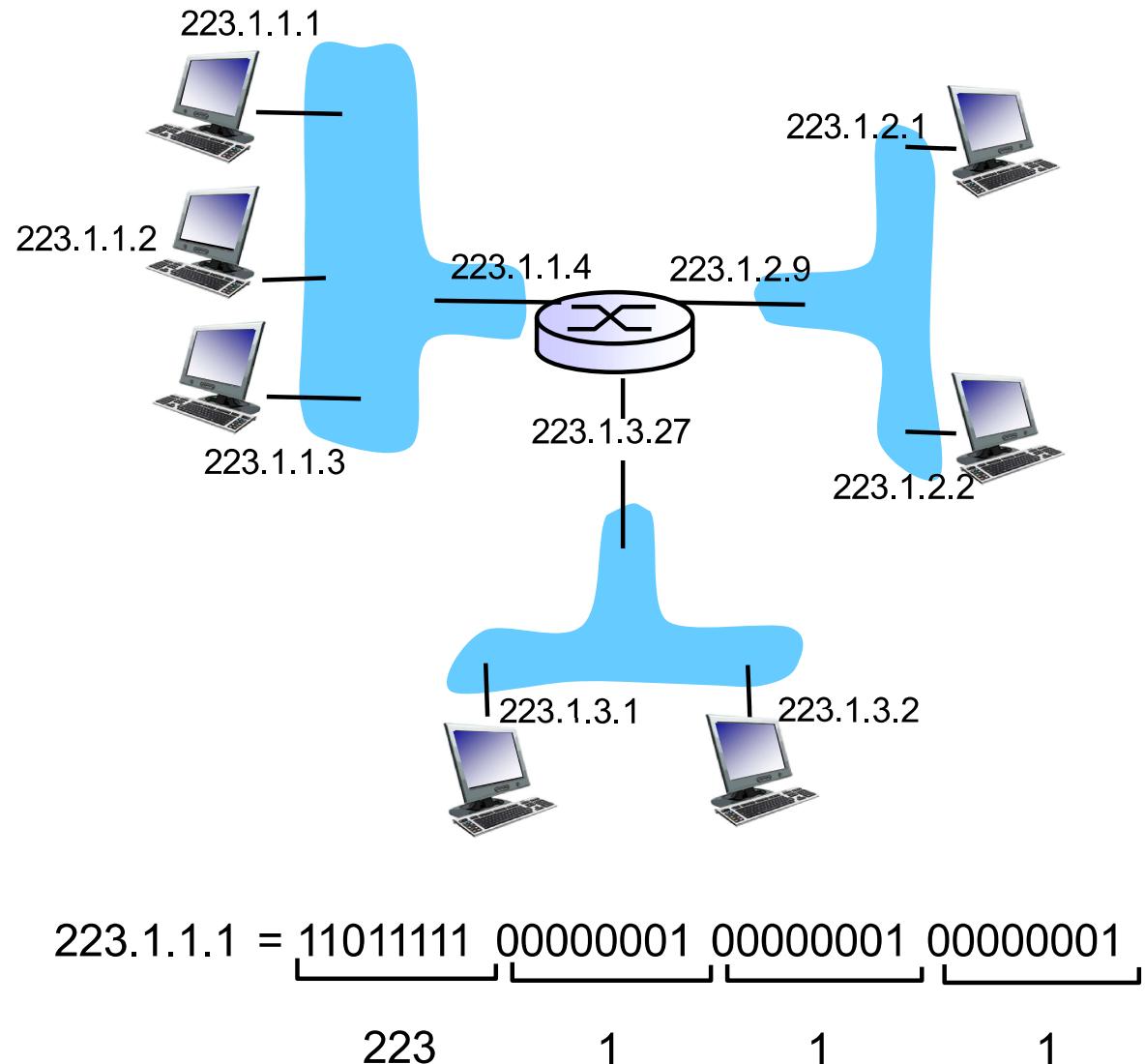
# 1. IP编址

## □ 接口 (interface) :

- 主机/路由器与物理链路的边界
- 路由器有多个接口
- 主机典型地有一个或两个接口（比如以太网接口、Wifi接口）

## □ IP address:

- 每个网络接口对应一个IP地址
- IP地址是一个32位的二进制数，通常用点分十进制数表示



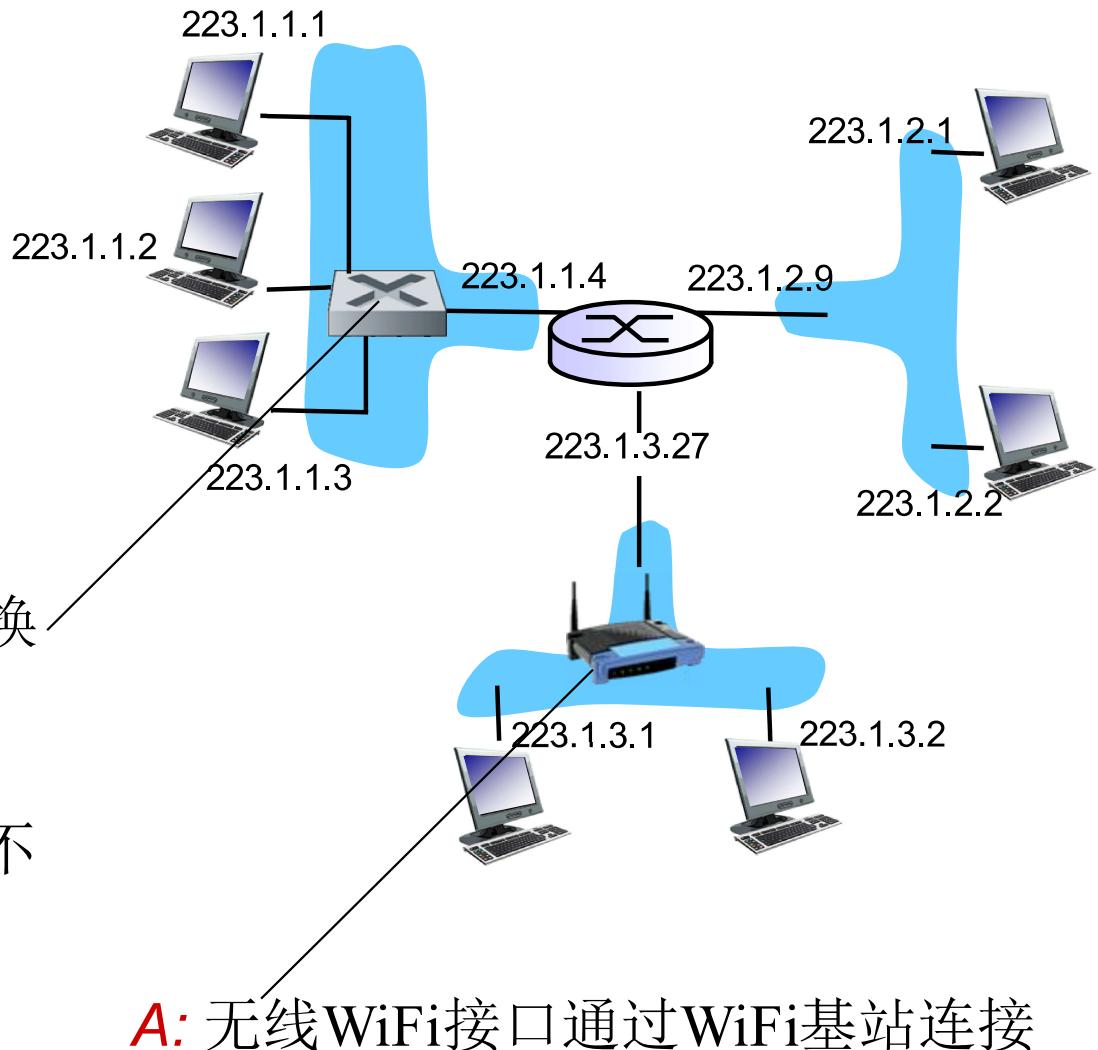
# IP编址

**Q:** 接口之间是怎么连接的？

**A:** 将在第6、7章介绍

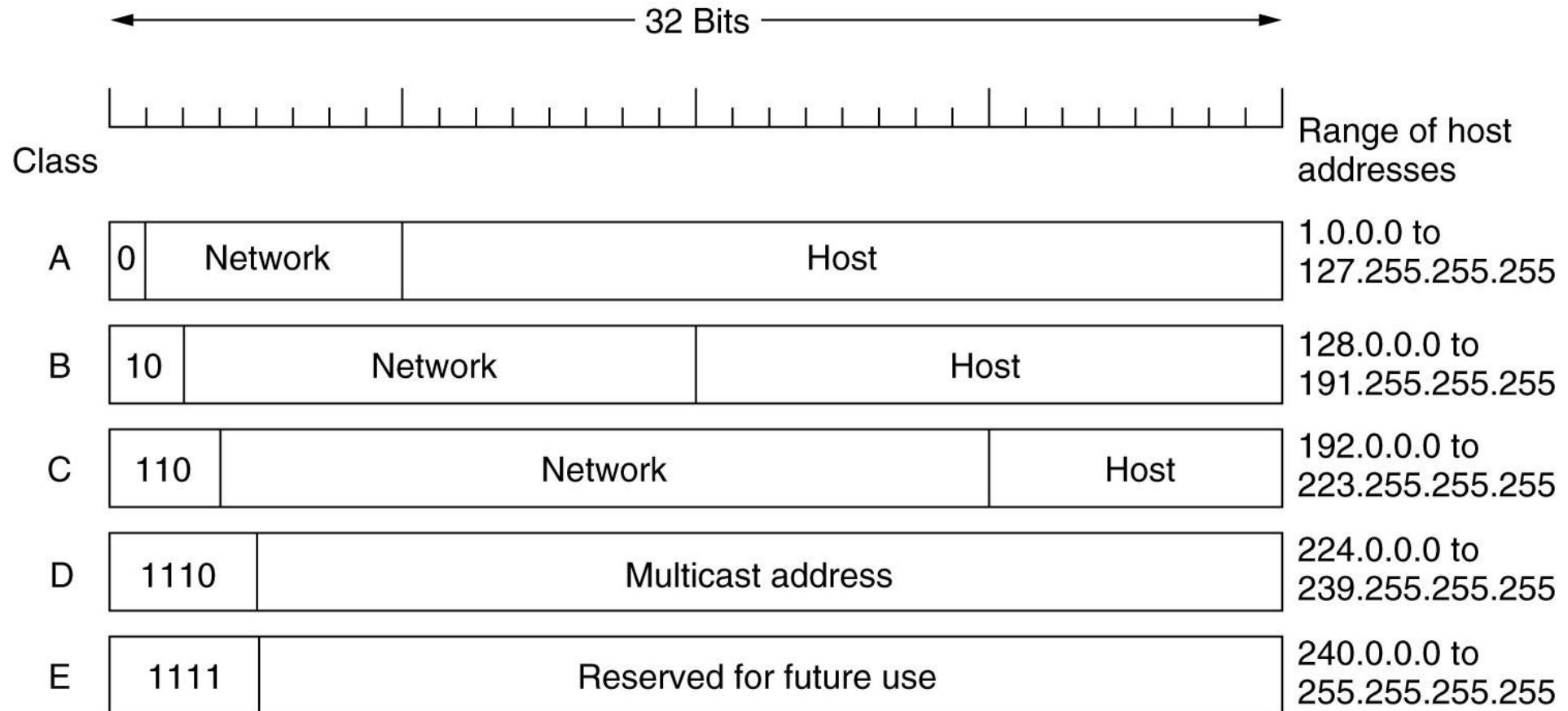
**A:** 以太网接口通过以太网交换机连接

**目前:** 暂不考虑一个接口如何不  
经由路由器连接到另一个接口



**A:** 无线WiFi接口通过WiFi基站连接

# 基于类的编址（早期）



# 单播地址结构

- 单播地址除类别标识外，其余比特被划分成网络号和主机号两部分：
  - 网络号：标识一个物理网络
  - 主机号：标识该物理网络上的一个网络接口
- 根据该编址方法可知，**同一个物理网络上的网络接口，它们的IP地址具有相同的网络号**

# 地址分配

- 因特网中的每个接口必须具有唯一的IP地址
- 为在因特网范围内保证IP地址的全局唯一性：
  - 网络号由ICANN统一分配
  - 主机号由网络管理员统一分配
- 建立私有网络的组织可以选择网络号，但同样必须保证每个网络号在私有网络内的唯一性

# 特殊的地址

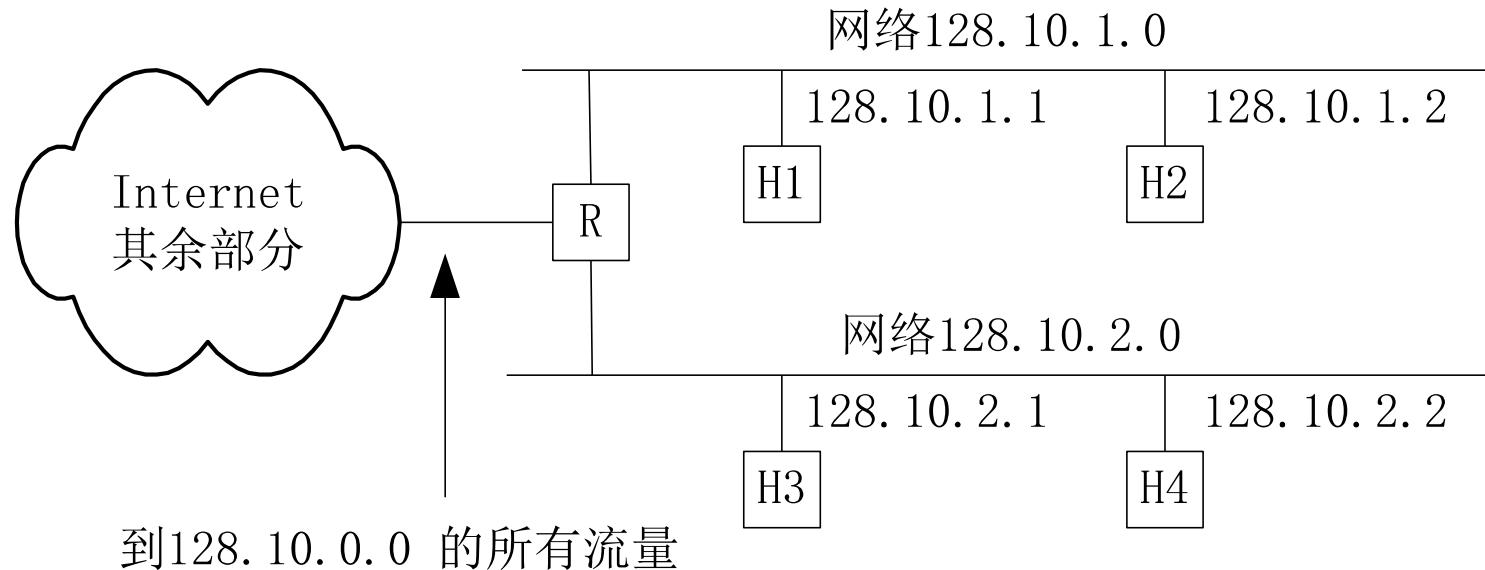
- 全0或全1的网络号及主机号是特殊地址，从不分配给特定的网络接口：
  - 网络号有效、主机号全为0的地址：保留给网络本身。
  - 网络号有效、主机号全为1的地址：保留作为定向广播，即在网络号指定的网络中广播（仅用作目的地址）
  - 32位全1的地址：本地广播地址，表示仅在发送节点所在的网络中广播（仅用作目的地址）
  - 32位全0的地址：指示本机（仅用作源地址）
  - 网络号为0、主机号有效的地址：指代本网中的主机
  - 形如127.xx.yy.xx的地址：保留作为回路测试，发送到这个地址的分组不输出到线路上，而是送回内部的接收端。

# 网络数量与地址数量

- A、B、C类地址对应不同规模的网络
- 一个A类、B类及C类地址可提供的接口地址数：
  - A类地址： $2^{24}-2 = 16777214$
  - B类地址： $2^{16}-2 = 65534$
  - C类地址： $2^8-2 = 254$
- A类、B类、C类地址的个数：
  - A类地址： $2^7-2 = 126$
  - B类地址： $2^{14}-2 = 16382$
  - C类地址： $2^{21}-2 = 2097152$

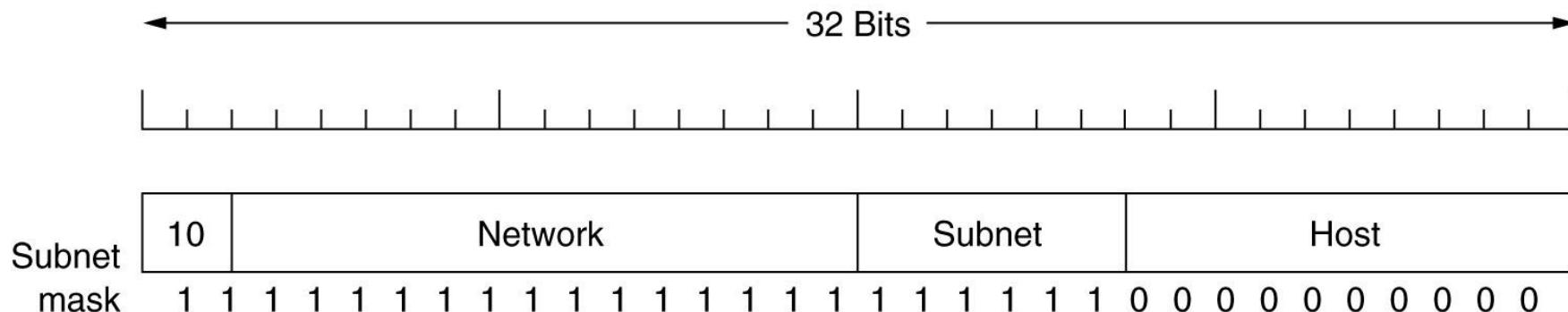
## 2. 子网 (subnet)

- 管理员通常利用路由器，将一个较大的网络划分成若干较小的网络（子网），每个网络使用一部分地址空间



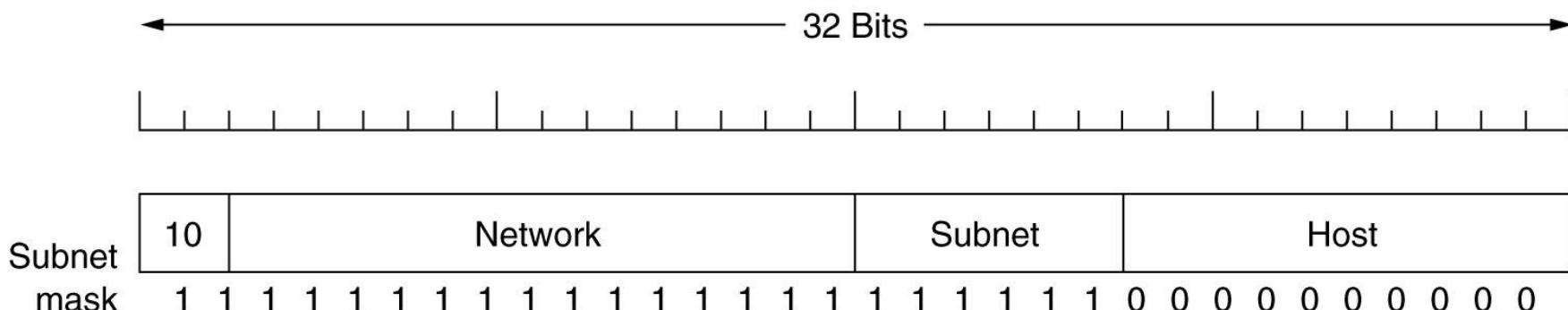
# 子网编址

- 从概念上说，引入子网仅略微改变了IP地址的解释：
  - 主机号被进一步划分成子网号和主机号两部分
  - 子网号标识网络内的一个子网，主机号标识子网中的一个网络接口
- 管理员可以根据需要，如子网的数量和规模，选择合适的子网号长度



# 子网掩码 (subnet mask)

- 子网掩码用于指示IP地址中子网号与主机号的边界：
  - 子网掩码是一个32比特的数，其中对应主机号的比特为0，其余比特为1
  - 子网掩码也采用点分十进制表示，如255.255.252.0
- 如何从IP地址中获取子网地址？
  - 将IP地址与子网掩码做“与”运算
  - 例如：128.10.1.1 AND 255.255.255.0 = 128.10.1.0
  - 注意：子网地址 ≠ 子网号，子网地址包括主机号之前的所有比特



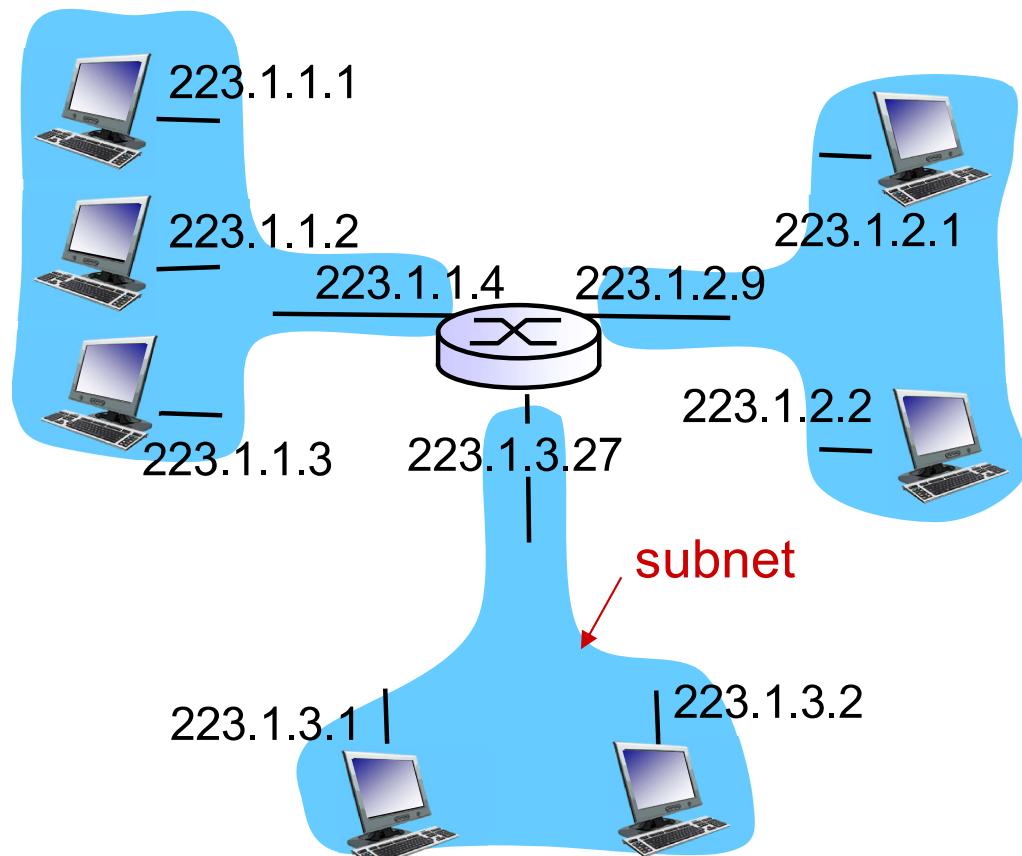
# 子网：更确切的含义

## □ 子网掩码将IP地址划分为两部分：

- 子网地址：对应子网掩码中“1”的部分
- 主机地址：对应子网掩码中“0”的部分

## □ 子网是什么？

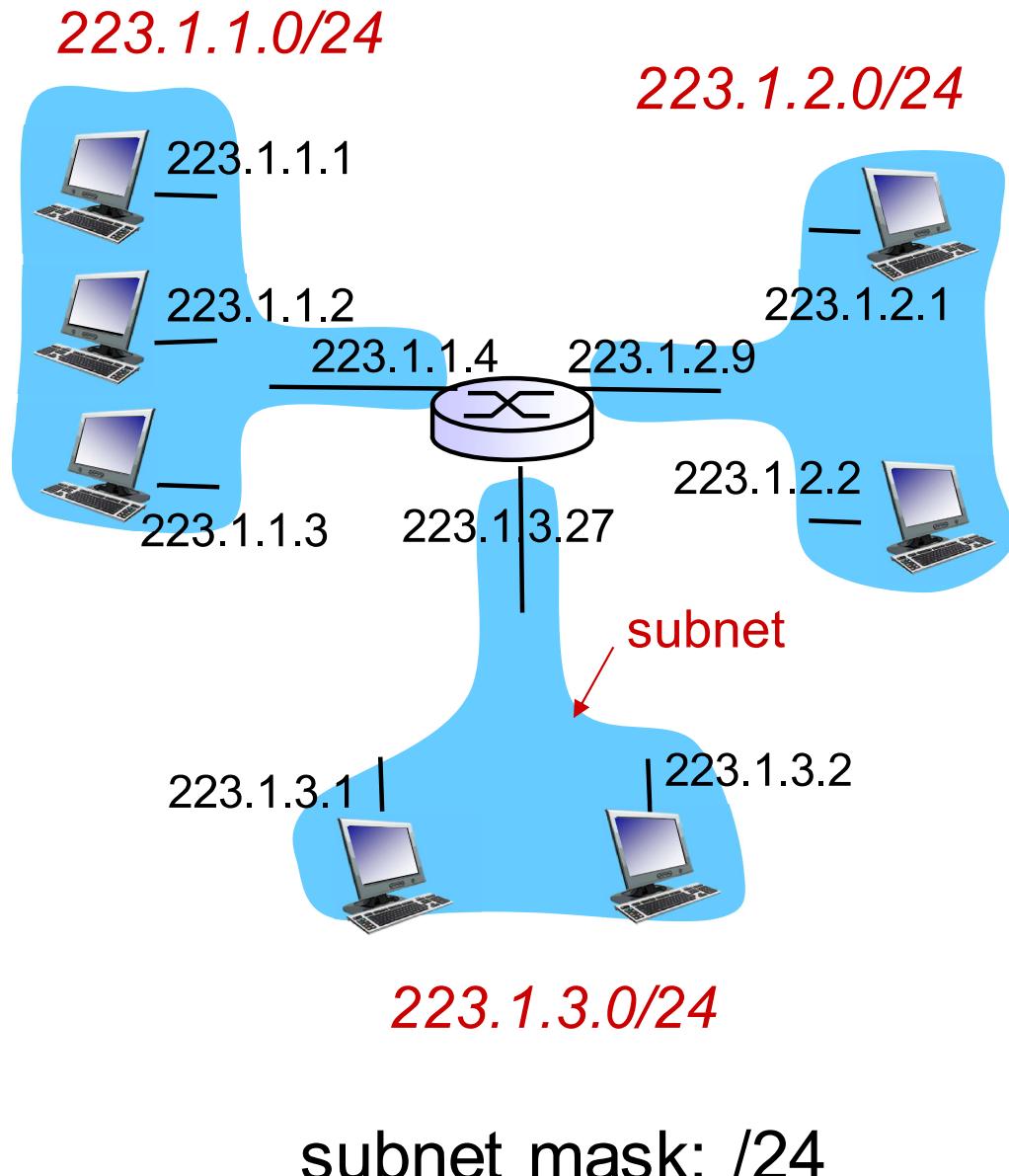
- 具有相同子网地址、且不需要通过路由器就可以相互到达的网络接口构成一个子网



network consisting of 3 subnets

# 如何确定子网？

- 将网络接口与主机/路由器分开，形成一些分离的网络岛
- 每个网络岛就是一个子网





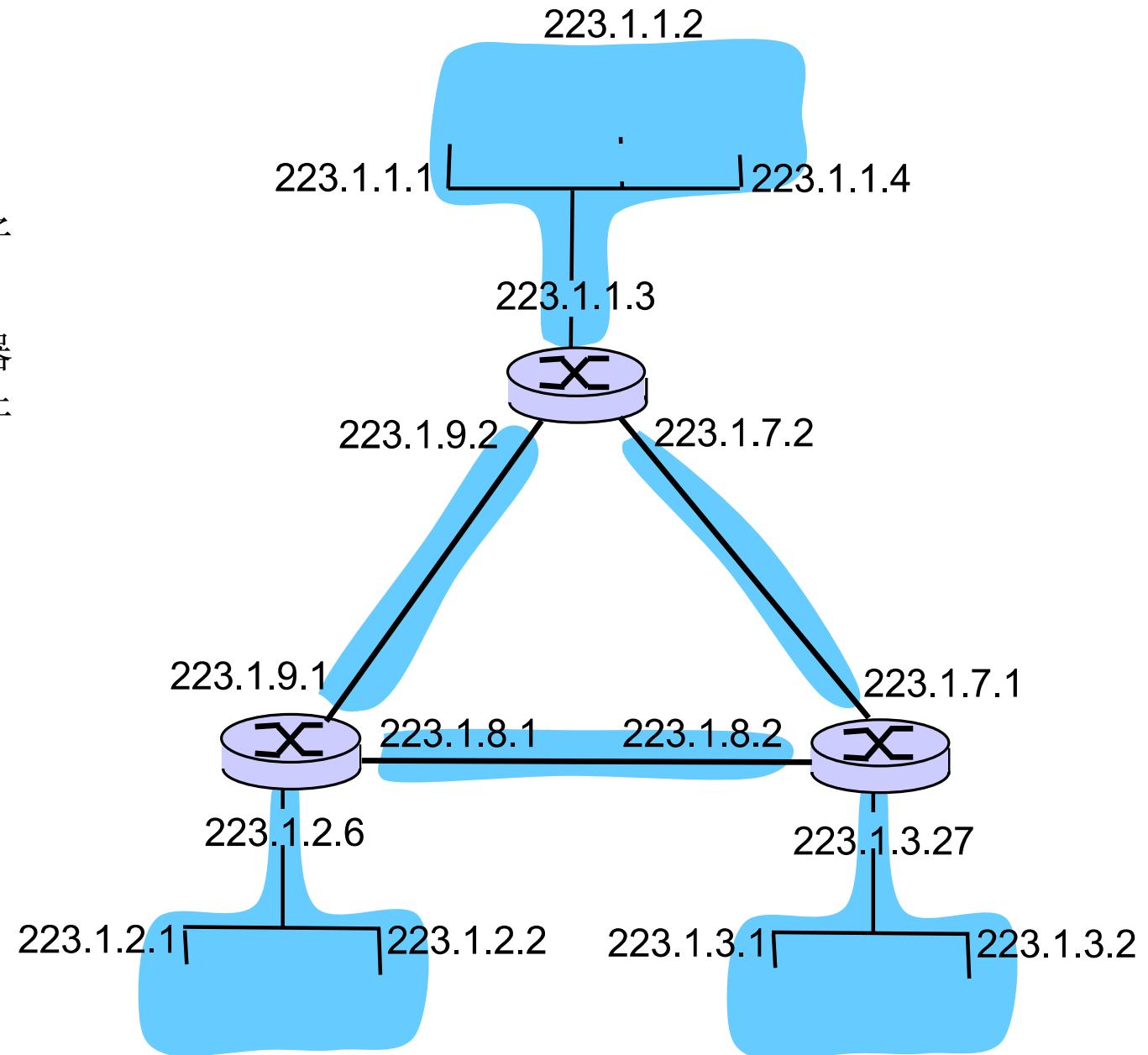
# 举例

□ 图示系统有6个子网：

- 每个子网具有相同的子网地址
- 子网内部不包含路由器
- 子网之间被路由器隔开

□ 重要的观察：

- 路由器的每个端口连接一个子网，不同的端口连接不同的子网
- 路由器是在子网之间转发数据包的设备
- 子网内部通信不需要通过路由器，子网之间通信必须通过路由器



# 要求理解的概念

## □ 子网：

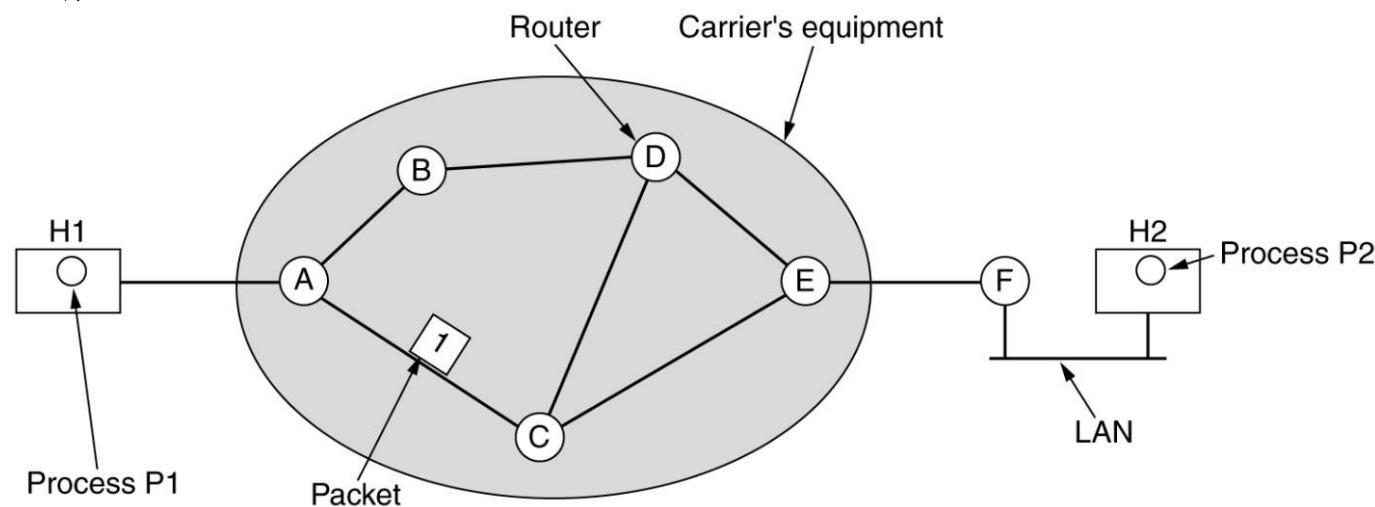
- 子网内部的接口具有相同的子网地址
- 子网内部的通信不需要经过路由器
- 子网之间通信一定要经过路由器

## □ 路由器：

- 在子网之间转发分组的设备，负责将分组从一个子网转发至另一个子网

### 3. IP数据报转发

- 网络层转发数据报的两种情形：
  - 直接交付 (direct delivery) : 节点将数据包直接发送给目的主机 (不需要其它路由器转发)
  - 间接交付 (indirect delivery) : 节点将数据包转发给一个路由器去处理

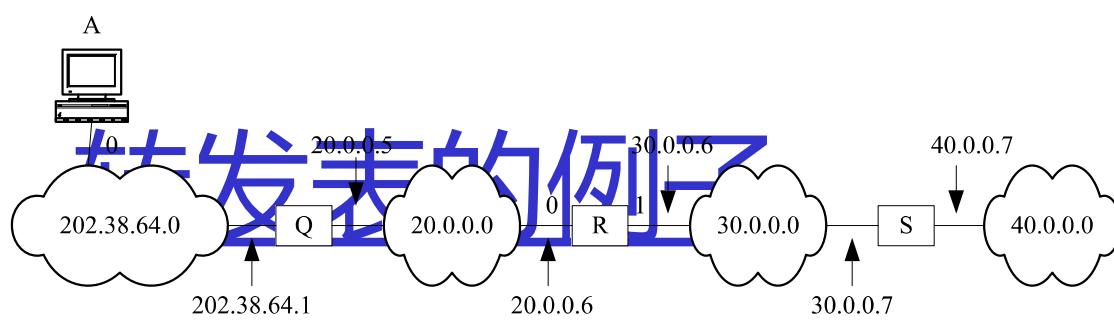


# 直接交付和间接交付

- 如何判断使用直接交付还是间接交付?
  - 直接交付：数据包的目的地址与本节点的某一端口在同一个子网中
  - 间接交付：数据包的目的地址不与本节点的任何一个端口在同一个子网中
- 直接交付的实现：
  - Chapter 6
- 间接交付的实现：
  - 节点查找转发表，将数据包发送给下一个路由器

# 转发表

- 转发表记录目的地址到输出端口的映射
- 取决于目的地址类型的不同，有三类转发表项：
  - 目的地址是一个子网地址：地址前缀表项
  - 目的地址是一个特定的网络接口地址：特定主机表项
  - 缺省项：不匹配所有其它表项的地址都被映射到一个默认的路由器端口
- IP采用逐跳选路：
  - 每个转发表项只记录去往目的地址的下一跳信息（下一个要到达的路由器端口），而不是一条完整的端到端路由
- 每个转发表项包括：
  - 目的地址/掩码、下一跳地址、输出端口等
  - 下一跳地址必须与输出端口在同一个子网中（不需要通过其它路由器就可以直接到达）



R的路由表

目的地址	掩码	下一跳	端口
20.0.0.0	255.0.0.0	直接交付	0
30.0.0.0	255.0.0.0	直接交付	1
202.38.64.0	255.255.255.0	20.0.0.5	0
40.0.0.0	255.0.0.0	30.0.0.7	1

A的路由表

目的地址	掩码	下一跳	端口
202.38.64.0	255.255.255.0	直接交付	0
default	0.0.0.0	202.38.64.1	0

# IP数据报的转发过程（主机/路由器）

从数据报中提取目的IP地址D，根据地址类别得到网络地址N；

if D与自己的任何一个IP地址匹配 //本节点是数据报的目的节点

    then 将数据包交给protocol域指定的协议实体处理

else if N与自己的任何一个直连网络的地址匹配 //直接交付

    then 通过该直连网络把数据包直接交付到目的节点D

else if 表中包含到D的特定主机表项 //间接交付

    then 把数据包发送到表中指定的下一跳

else if 表中包含到N的一个地址前缀表项 //间接交付

    then 把数据包发送到表中指定的下一跳

else if 表中包含一个缺省项 //间接交付

    then 把数据包发送到表中指定的默认路由器端口

else 宣告选路出错，向数据包的源地址发送一条错误报告消息（ICMP）

# 要求掌握的概念

## □ 直接交付:

- 不经过路由器就能到达（在同一个子网内）

## □ 间接交付:

- 需要路由器转发才能到达（不在同一个子网内）

## □ 逐跳转发:

- 每次只转发到下一跳，下一跳必须是直接可达的（在同一个子网内）

## □ 如何依据转发表进行转发决策

## 4. CIDR: Classless InterDomain Routing

### □ 分类编址的缺点：

- 只能按照三种固定的大小分配地址空间，地址浪费严重（尤其是A类、B类地址）
- 转发表必须记录每个已分配的网络，转发表规模爆炸式增长（C类地址网络非常多）

### □ CIDR：

- 按照实际需要的地址数量分配地址空间，提高地址使用效率
- 允许将若干条转发表项进行聚合，减小转发表规模

# 按照实际需要分配地址

- 举例：
  - 若一个网络需要2000个地址，可为其分配一个具有2048个连续地址的地址块；这些地址的前21位必须相同，从而可将其看成是一个具有21位子网地址的网络
- CIDR地址分配的原则：
  - 地址块的长度 L 必须是 2 的幂次
  - 所有地址的前  $(32 - \log_2 L)$  位必须相同
- 网络地址的表示方法：
  - 用掩码指示网络地址的长度，如194.24.0.0，255.255.248.0
  - 用 “/长度” 指示子网地址的长度，如194.24.0.0/21

# 机构如何获得网络地址？

- 机构通常从ISP的地址空间中分配地址
- 假设：ISP有一块地址200.23.16.0/20（共4096个地址），8个机构向该ISP申请地址，每个申请512个地址

ISP's block	<u>11001000 00010111 00010000 00000000</u>	200.23.16.0/20
Organization 0	<u>11001000 00010111 0001<u>000</u>0 00000000</u>	200.23.16.0/23
Organization 1	<u>11001000 00010111 0001<u>001</u>0 00000000</u>	200.23.18.0/23
Organization 2	<u>11001000 00010111 0001<u>010</u>0 00000000</u>	200.23.20.0/23
...	.....	....
Organization 7	<u>11001000 00010111 0001<u>111</u>0 00000000</u>	200.23.30.0/23

# CIDR地址分配的另一个例子

- ISP有一块从194.24.0.0/16 开始的地址块
  - 剑桥大学申请2048个地址
  - 牛津大学申请4096个地址
  - 爱丁堡大学申请1024个地址

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

# 地址分配过程

剑桥大学申请2048个地址

ISP's block	<u>11000000</u> <u>00011000</u>	00000000	00000000	194.24.0.0/16
Cambridge	<u>11000000</u> <u>00011000</u>	<u>00000</u> 000	00000000	194.24.0.0/21
(Available)	<u>11000000</u> <u>00011000</u>	<u>00001</u> 000	00000000	194.24.8.0/21

牛津大学申请4096个地址

ISP's block	<u>11000000</u> <u>00011000</u>	00000000	00000000	194.24.0.0/16
Cambridge	<u>11000000</u> <u>00011000</u>	<u>00000</u> 000	00000000	194.24.0.0/21
(Available)	<u>11000000</u> <u>00011000</u>	<u>00001</u> 000	00000000	194.24.8.0/21
Oxford	<u>11000000</u> <u>00011000</u>	<u>0001</u> 0000	00000000	194.24.16.0/20

爱丁堡大学申请1024个地址

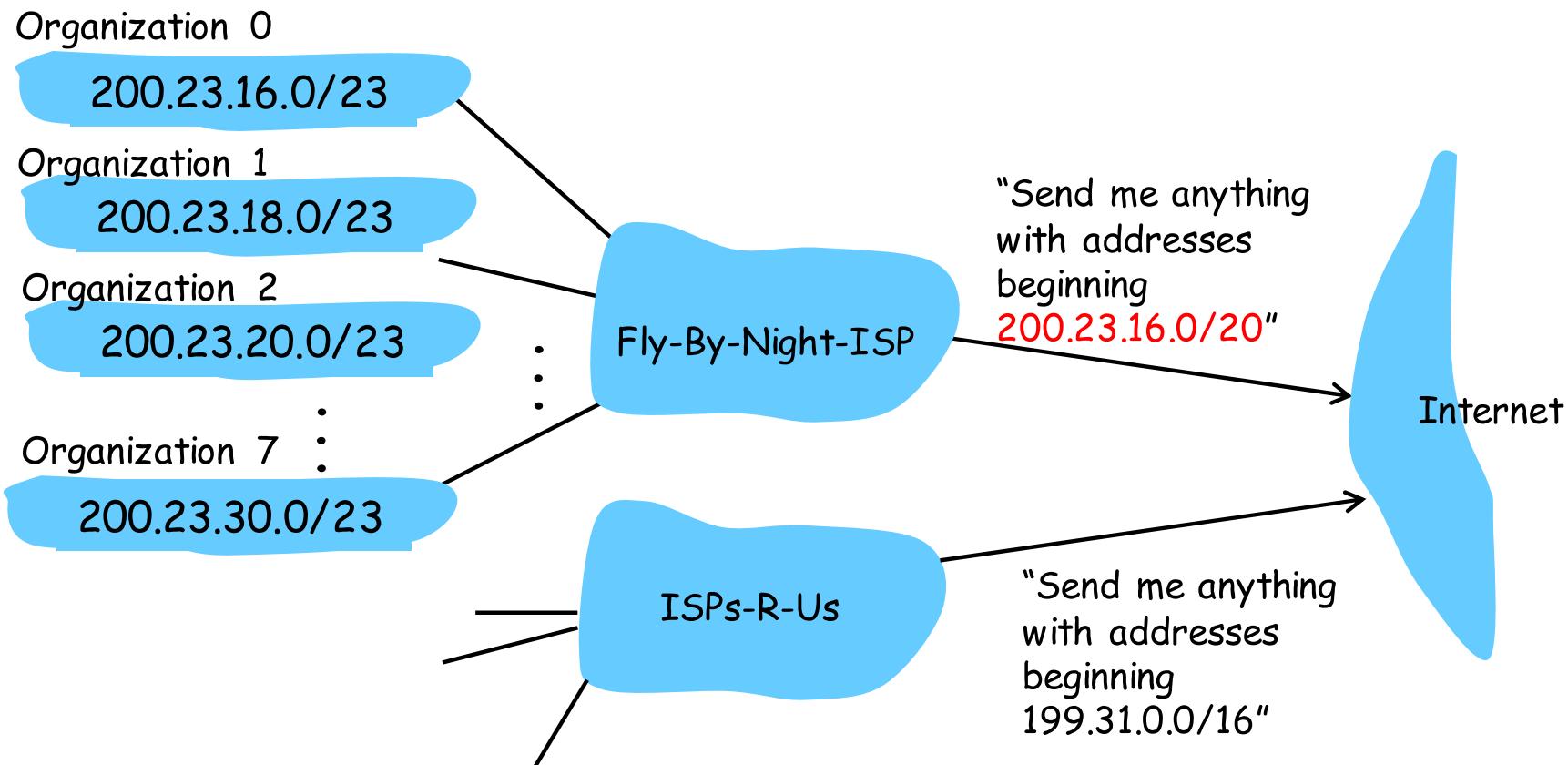
ISP's block	<u>11000000</u> <u>00011000</u>	00000000	00000000	194.24.0.0/16
Cambridge	<u>11000000</u> <u>00011000</u>	<u>00000</u> 000	00000000	194.24.0.0/21
Edinburgh	<u>11000000</u> <u>00011000</u>	<u>000010</u> 00	00000000	194.24.8.0/22
(Available)	<u>11000000</u> <u>00011000</u>	<u>000011</u> 00	00000000	194.24.12.0/22
Oxford	<u>11000000</u> <u>00011000</u>	<u>0001</u> 0000	00000000	194.24.16.0/20

# 更新转发表

- ISP在其转发表中添加三个表项：
  - 194.24.0.0/21 (掩码：255.255.248.0) // Cambridge
  - 194.24.8.0/22 (掩码：255.255.252.0) // Edinburgh
  - 194.24.16.0/20 (掩码：255.255.240.0) // Oxford
  
- 若路由器收到目的地址为194.24.17.4的数据包，其查表过程为：
  - 194.24.17.4 AND 255.255.248.0 = 194.24.16.0, 与194.24.0.0不匹配
  - 194.24.17.4 AND 255.255.252.0 = 194.24.16.0, 与194.24.8.0不匹配
  - 194.24.17.4 AND 255.255.240.0 = 194.24.16.0, 与194.24.16.0匹配  
选择该转发表项

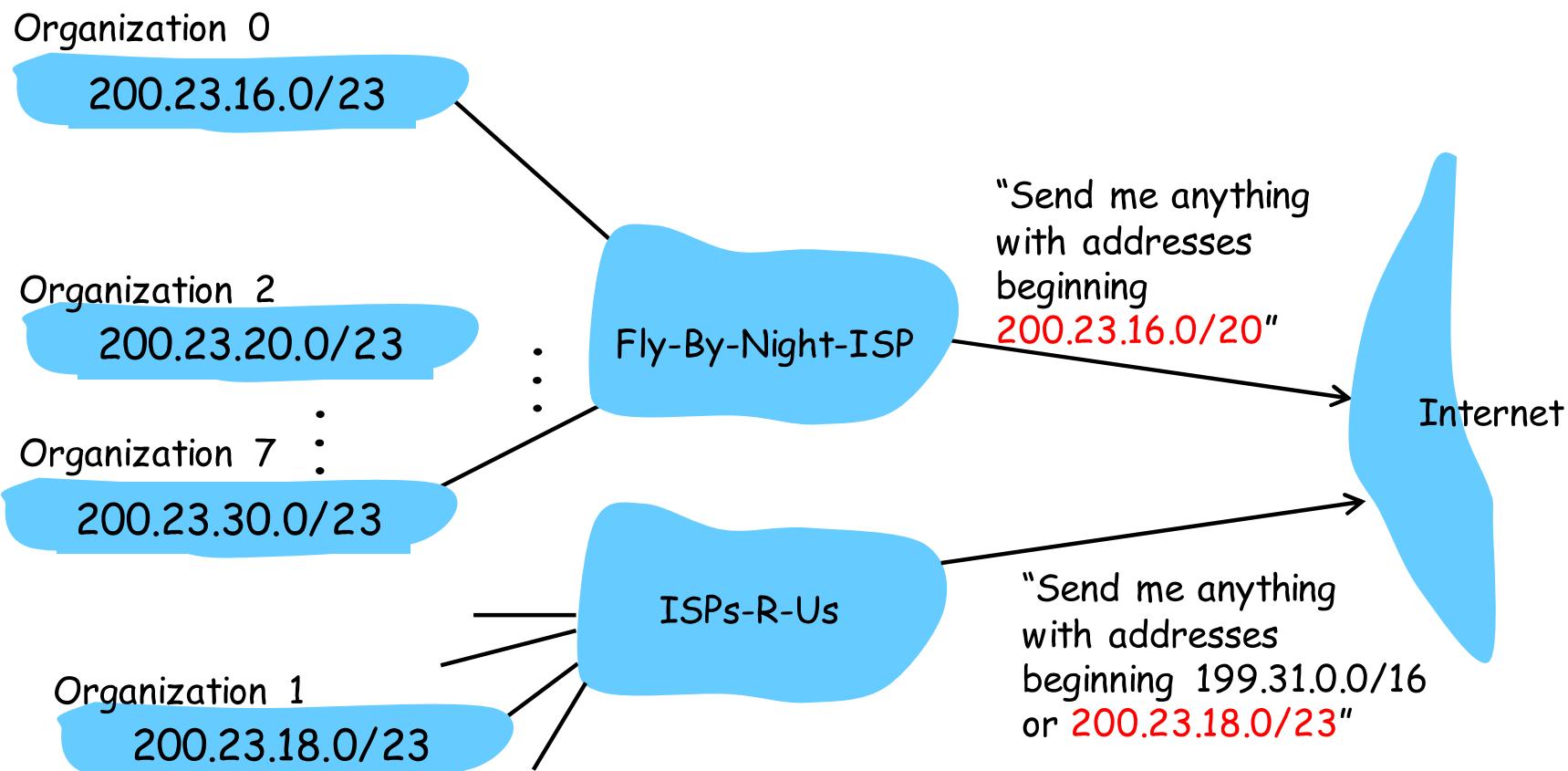
# 地址聚合

- 转发表中符合以下条件的若干个表项可以合并成一个表项：
  - 这些表项的目的地址可以聚合成一个前缀更短的地址
  - 这些表项使用相同的下一跳
- 地址聚合的过程可以递归进行



# 不能被聚合的转发表项

- 若个别表项不满足路由聚合的条件：
  - 仍然可以在转发表中给出一条聚合表项： **200.23.16.0/20**
  - 同时给出不能被聚合的表项： **200.23.18.0/23**
- **最长前缀匹配：**
  - 在所有匹配的路由表项中，选择前缀最长的表项



# 查找转发表：使用分类地址

- 转发表分为A、B、C三张表，分别记录A、B、C三类地址的转发表项，用哈希表组织
- 路由器收到数据报后：
  - 根据目的地址的类型确定要查找的转发表
  - 根据目的地址的类型提取网络地址
  - 用网络地址在相应的转发表中进行哈希查找（精确匹配）

# 采用CIDR后出现的问题

- 为与某个转发表项 Dest\_addr/prefix\_len进行匹配运算：
  - 路由器需要先从表项中读出地址掩码（或prefix-len值）
  - 计算包的目的地址前缀（用地址掩码和包的目的地址相与）
  - 与Dest\_addr的地址前缀（Dest\_addr与地址掩码相与）进行比较
- 引入的问题：
  - 地址前缀的长度prefix\_len可以是任意值
  - Prefix\_len无法从地址本身得到，只能从转发表项中得到
  - 必须从所有匹配的表项中选择前缀最长的表项
- 在大规模转发表中进行快速查找是一个难题（已经解决）

# IP编址的小结

- 基于类的编址存在两个问题：
  - 地址空间浪费
  - 转发表空间爆炸
- **CIDR**解决了这两个问题：
  - 按需分配解决了地址空间浪费的问题
  - 地址聚合解决了转发表空间爆炸的问题
- **CIDR**引入的问题：
  - 地址聚合要求采用最长前缀匹配的原则查找转发表
  - 最长前缀匹配给转发表的快速查找带来了困难

## 5. 主机/路由器如何获得IP地址？

### □ 路由器：

- 管理员手工配置路由器各个接口的IP地址

### □ 主机：

- 管理员手工配置主机IP地址，服务器通常采用这种方法
- 使用动态主机配置协议DHCP（**Dynamic Host Configuration Protocol**）获取IP地址、子网掩码、缺省路由器、本地DNS服务器等配置信息，个人终端通常采用这种方法

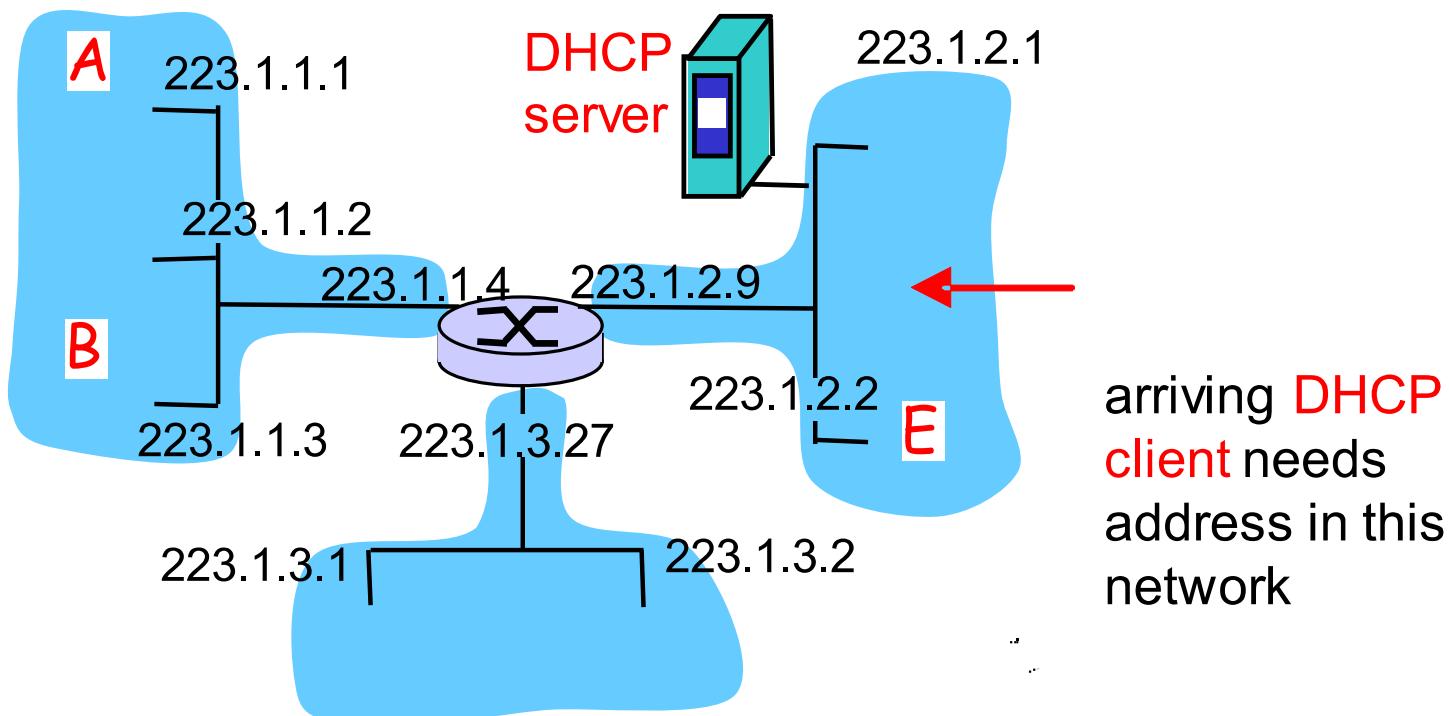
### □ 使用DHCP的好处：

- 免去手工配置的麻烦（**即插即用**）
- 可用少量的IP地址服务较多的客户（**地址重用**）



# DHCP

- **目标:** 允许主机加入网络时自动获取配置信息
- **DHCP**是一个客户-服务器模式的应用协议
  - 每个子网中须有一个**DHCP**服务器，或者一个**DHCP**代理
  - 新到达的主机上运行**DHCP client**

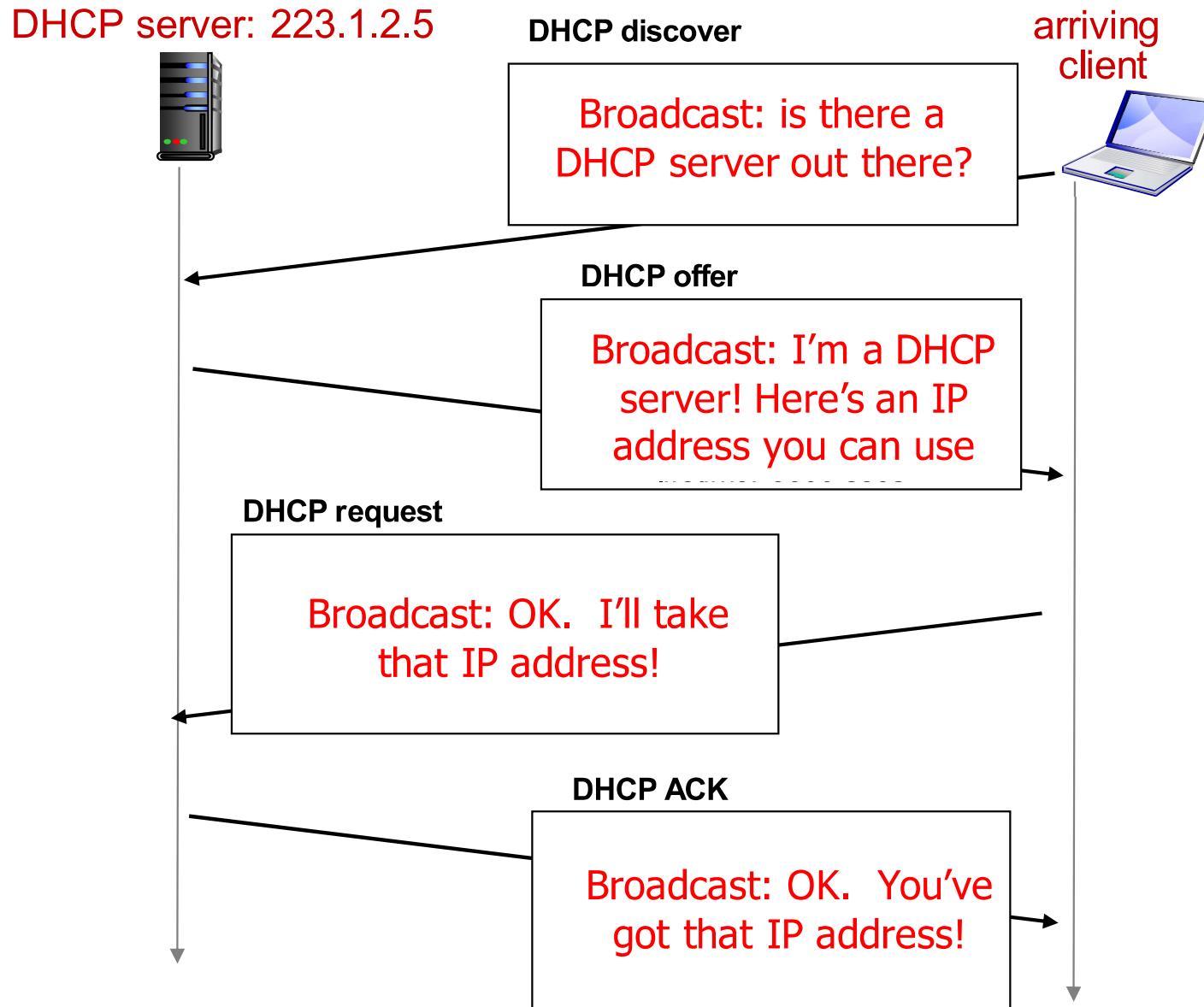


arriving **DHCP**  
**client** needs  
address in this  
network

# DHCP概述

- 主机广播 “**DHCP discover**” 报文
  - 寻找子网中的**DHCP**服务器
- **DHCP**服务器用 “**DHCP offer**” 报文进行响应
  - 给出推荐的**IP**地址及租期、其它配置信息
- 主机用 “**DHCP request**” 报文请求**IP**地址
  - 主机选择一个**DHCP**服务器，向其请求**IP**地址
- **DHCP**服务器用“**DHCP ack**” 报文发送**IP**地址
  - 服务器响应客户的请求，确认所要求的参数
- **DHCP**服务器使用**UDP**端口**67**，客户使用**UDP**端口**68**

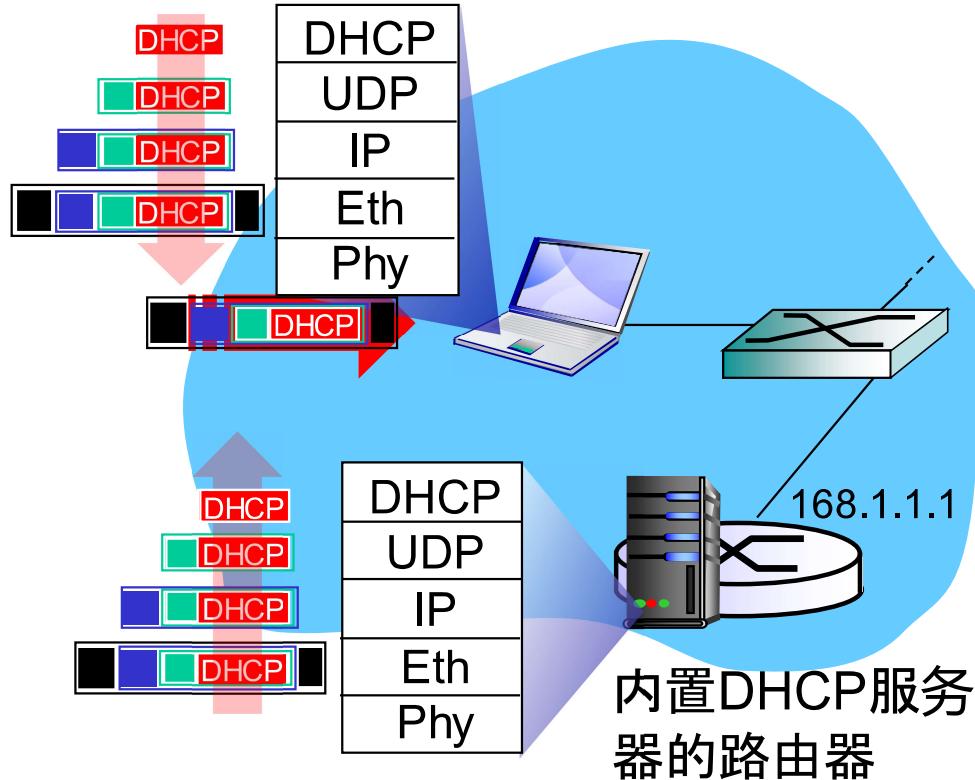
# DHCP客户-服务器交互



# DHCP: more than IP addresses

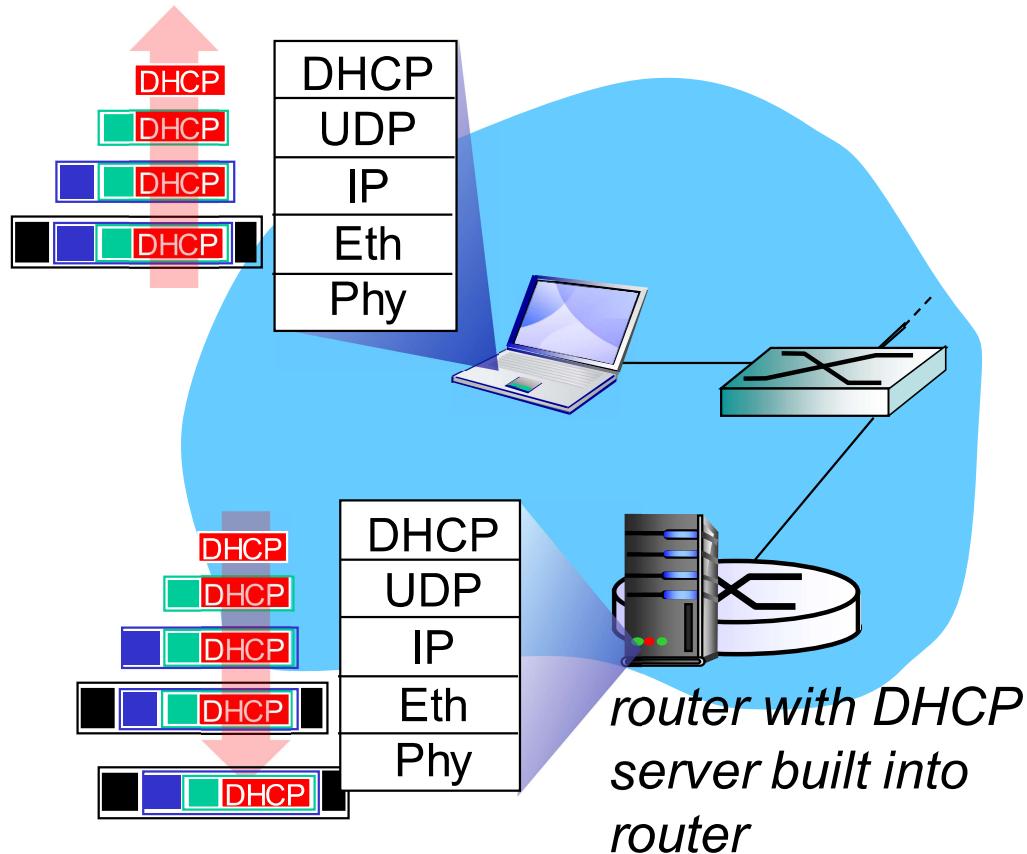
- DHCP不只返回分配的IP地址，还包括：
  - 第一跳路由器的IP地址（缺省网关、边缘路由器）
  - 本地DNS服务器的IP地址
  - 地址掩码

# DHCP报文传输



- 入网的笔记本电脑需要本机地址、第一跳路由器地址、本地DNS服务器地址
- DHCP请求被封装到 UDP/IP/以太帧中
- 以太帧在局域网中广播 (dest: FFFFFFFFFFFF), 被运行了DHCP服务器的路由器收到
- DHCP请求被提取出来, 送给DHCP服务器

# DHCP报文传输



- **DHCP**服务器构造**DHCP**响应，包含客户的**IP**地址、客户第一跳路由器的**IP**地址、本地**DNS**服务器的**IP**地址
- **DHCP**响应经过封装、传输、解封装，到达**DHCP**客户

# Chapter 4: Network Layer

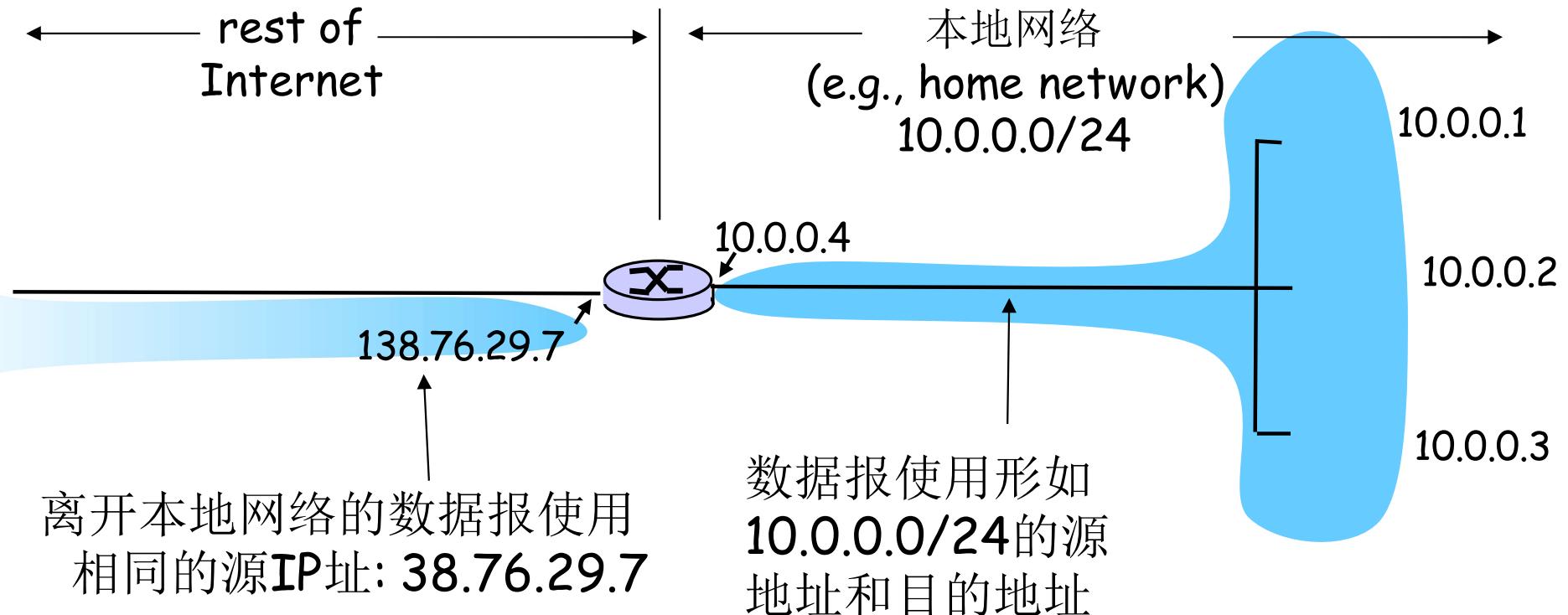
- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action



# 网络地址转换 (NAT)



## □ 动机:

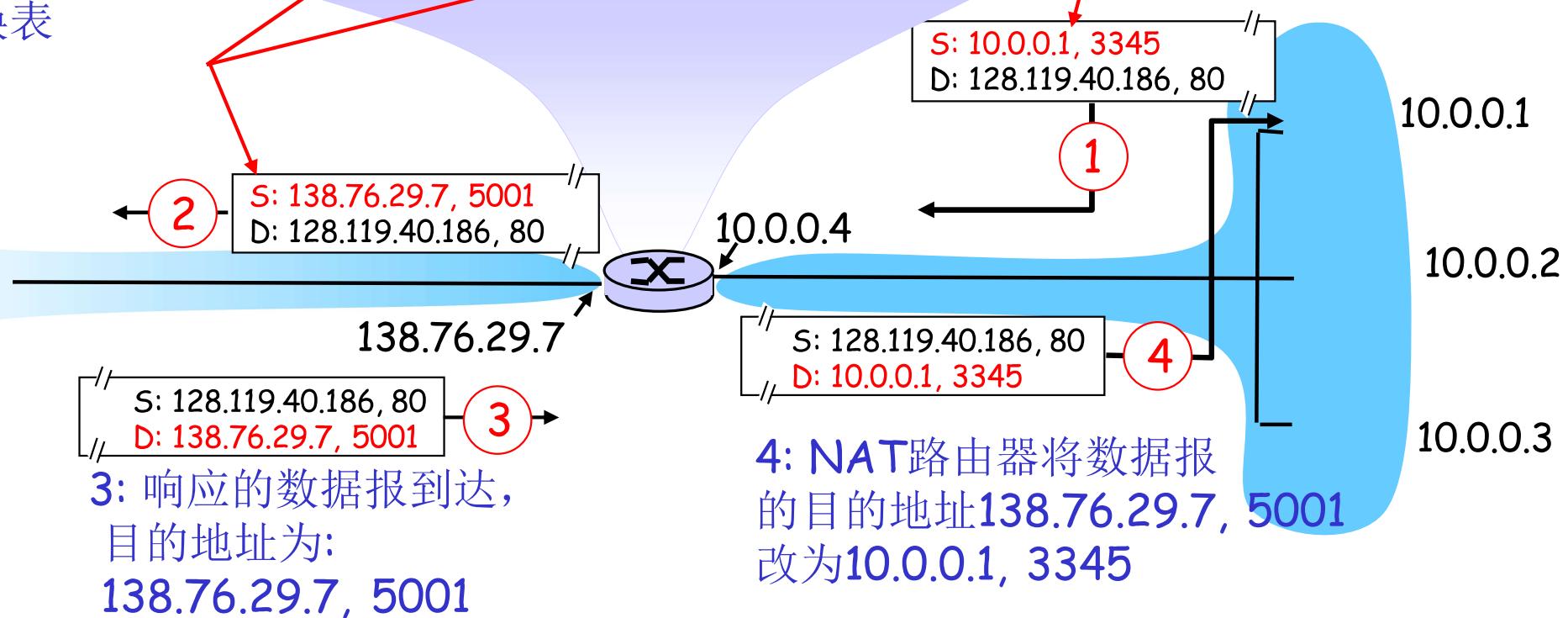
- 使用一个公用IP地址支持许多用户**同时**上网
- 仅为公共可访问的节点分配公用IP地址（减少需要的公用IP地址数）
- 网络内部节点对外是不可见的（安全考虑）



# NAT举例

2: NAT路由器将数据报源地址  
10.0.0.1, 3345改为  
138.76.29.7, 5001,  
更新转换表

NAT转换表	
因特网侧地址	本地地址
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....



# NAT实现

## □ 外出的数据报:

- 将数据报中的（源IP地址，源端口号）替换为（NAT IP地址，NAT端口号）

## □ NAT 转换表:

- 记录每个（源IP地址，源端口号）与（NAT IP地址，NAT端口号）的转换关系

## □ 进入的数据报:

- 取出数据报中的（目的IP地址，目的端口号）查找NAT转换表，然后用转换表中对应的（IP地址，端口号）进行替换

# NAT: Network Address Translation

- 16比特端口号:

- 允许一个NAT IP地址同时支持65535个对外连接

- NAT的使用有争议:

- 路由器应当只处理三层以下的包头（端口号在传输层）
  - 违反端到端原则（节点介入修改IP地址和端口号）

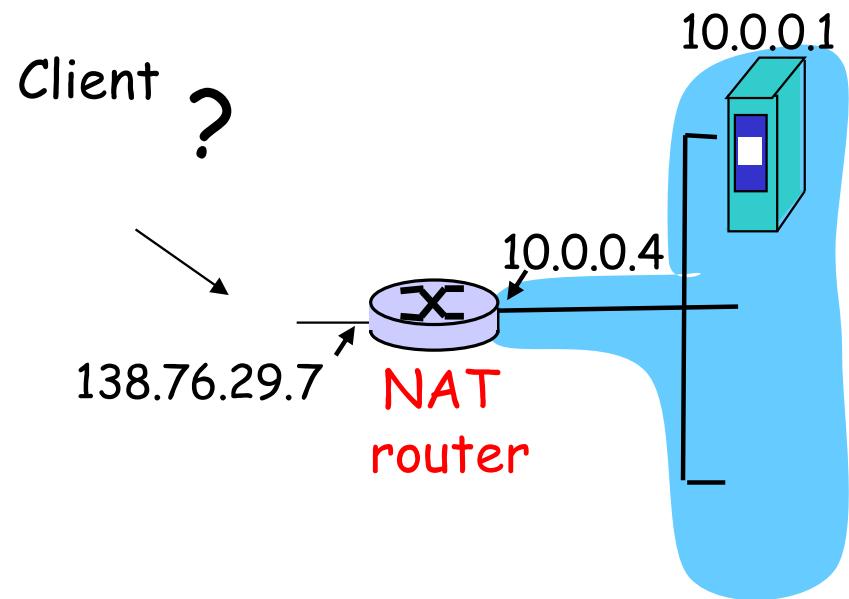
- NAT妨碍P2P应用:

- NAT只允许内部主动发起的通信（位于NAT后面的主机对外是不可见的）
  - 但P2P应用要求任何对等方可以向任何其它（参与的）对等方发起通信



# NAT 穿越问题

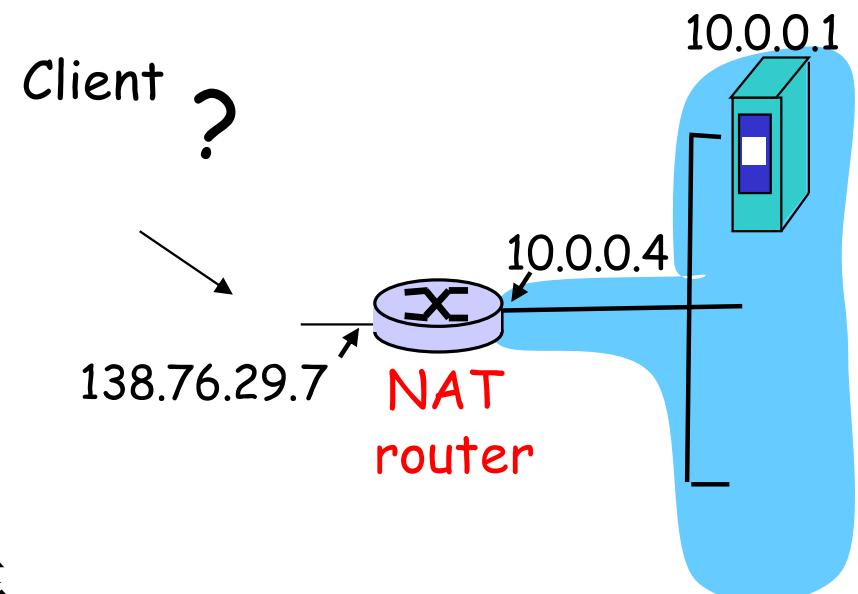
- Client希望连接服务器10.0.0.1，但是
  - 地址 10.0.0.1 为内部地址，对于client 不可见
  - Client 只能看到NAT路由器的公共IP地址138.76.29.7





# 使用UPnP实现NAT穿越

- 假设主机10.0.0.1在端口3345上运行一个BT程序：
  - BT程序请求NAT产生一个“洞”，将 $<10.0.0.1, 3345>$ 映射到 $<138.76.29.7, 5001>$ 上
  - BT程序向追踪器通告它在 $<138.76.29.7, 5001>$ 上可用
  - 其它主机通过追踪器可以看到该主机，并能向 $<138.76.29.7, 5001>$ 发起TCP连接
  - NAT将 $<138.76.29.7, 5001>$ 上收到的SYN包转发给主机10.0.0.1

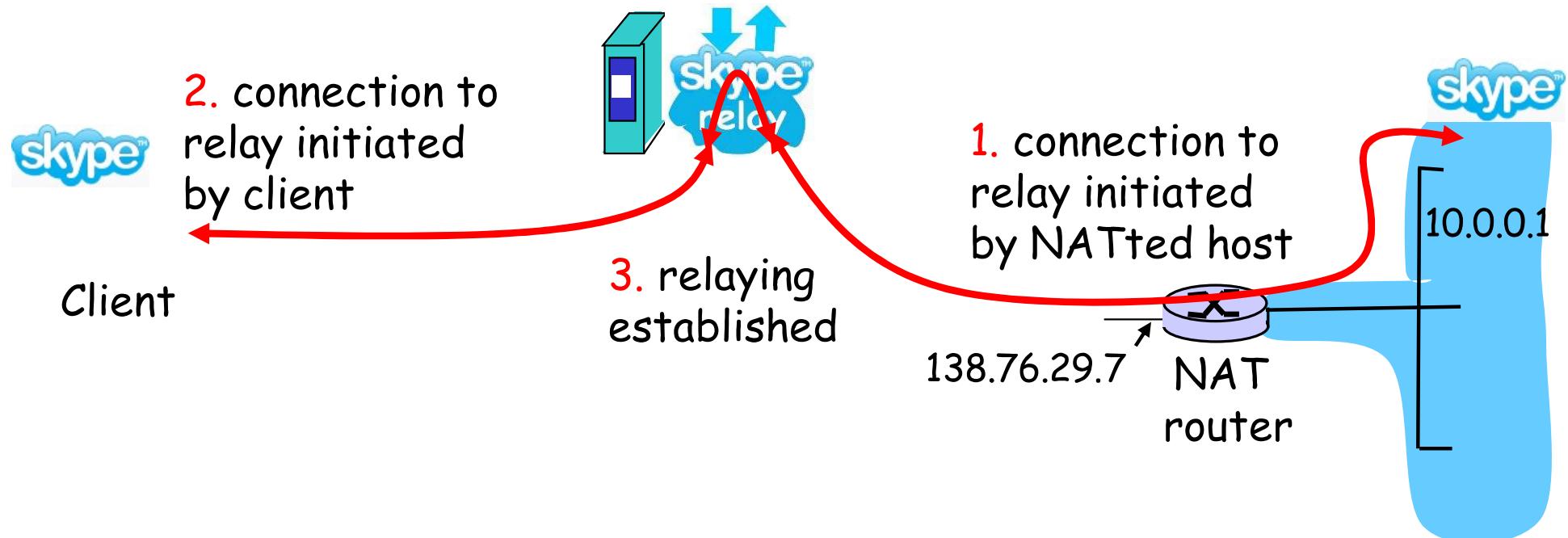




# 使用中继服务器实现NAT穿越

## □ 在 Skype 中使用：

- NAT 后面的服务器与中继器建立连接
- 外部客户与中继器建立连接
- 中继器在两个连接之间转发分组



# 重要知识点

- 子网、路由器与数据报转发：
  - 子网内部通信不需要经过路由器，子网之间通信必须经过路由器
  - 子网内部可直接交付，跨子网需间接交付
  - 逐跳转发：下一跳必须直接可达
- CIDR：
  - 地址分配，地址聚合，最长前缀匹配
- NAT的工作原理
- CIDR、DHCP、NAT均可以部分解决IP地址不足的问题，但角度不同

# Chapter 4: Network Layer

- 4.1 Introduction
- \*\* Virtual circuit and datagram networks
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6

## 4.4 Generalized Forwarding and SDN

- match
- action
- OpenFlow examples of match-plus-action in action

# IPv6

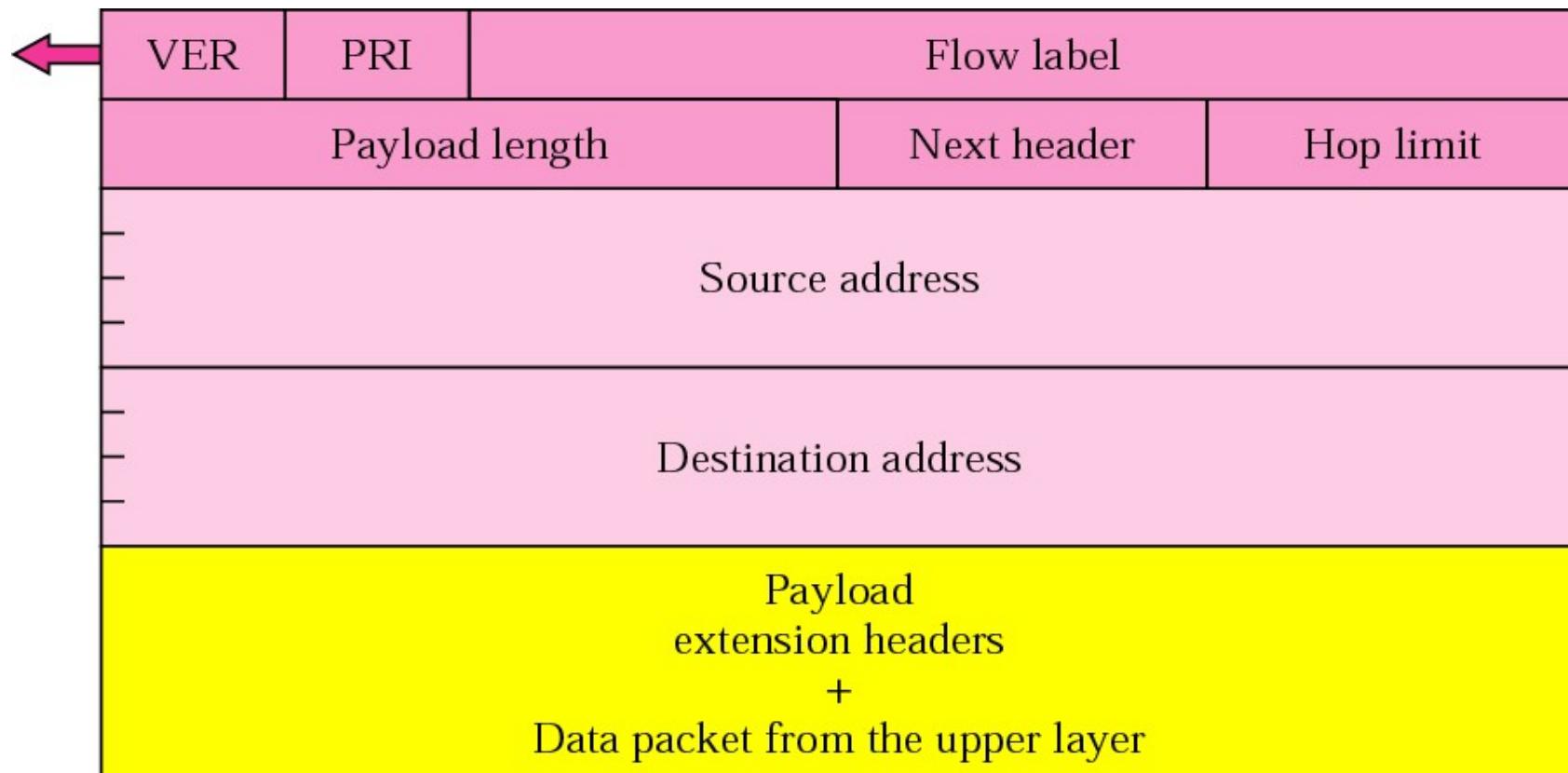
- 最初的动机: IPv4地址将很快耗尽
- 进一步的动机是借机改进IPv4的不足之处:
  - 简化头部格式, 加快数据报处理和转发
  - 支持服务质量
  - 支持多播
  - 支持移动性
  - 增强安全性
  - .....
- IPv6与IPv4不兼容, 但与其它所有因特网协议都兼容

# IPv6地址

- 128位，使用冒号十六进制表示，每16位以十六进制的形式写成一组，组之间用冒号分隔，如  
8000:**0:0:0**:0123:4567:89AB:CDEF
- 地址表示的零压缩技术：可将连续的多组0压缩为一对冒号，如以上地址可表示为：8000**::**0123:4567:89AB:CDEF
- IPv6定义了三种地址类型：
  - 单播地址：一个特定的网络接口
  - 多播地址：一组网络接口
  - 任播地址（anycast）：一组网络接口中的任意一个（通常是最最近的一个）

# IPv6数据报格式

- IPv6数据报以一个40字节的基本头开始，后面跟零个或多个扩展头，然后是数据



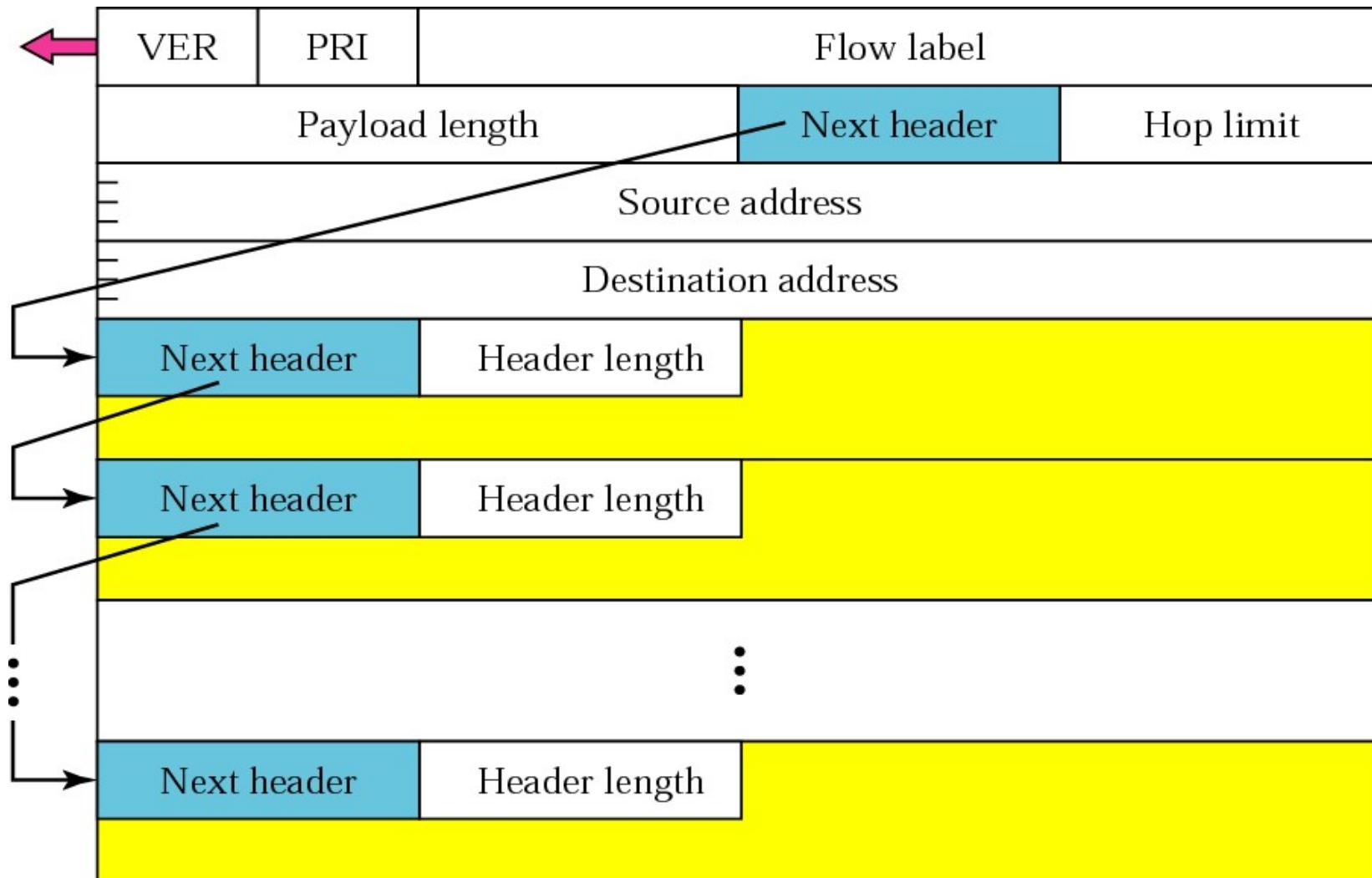
# PRI (或traffic class)

- 作用：
  - 发送方在该域定义数据报的优先级
  - 路由器发现网络拥塞时，按优先级从低到高的顺序丢弃包
- IPv6将网络流量划分为两大类：
  - 受拥塞控制的流：
    - 非实时流属于这一类，优先级0~7，按照重要性及用户体验设定
  - 不受拥塞控制的流：
    - 实时多媒体流属于这一类，优先级8~15，尚无标准，可以按照用户要求的服务质量等级定义

# 流标签（flow label）

- 流标签：用于标明属于同一个流（**flow**）的数据报，但流的概念并没有明确地定义
- 一般来说，流是**具有相同传输特性**（如源/目的、优先级、选项等）、**并要求相同处理**（如使用相同的路径和资源、具有相同的服务质量要求等）的一系列数据包
- 流标签由发送方分配，**<源地址，流标签>**唯一标识一个流：
  - 路由器维护一张流表（**flow table**），记录每一个流需要的处理；收到数据包后，根据源地址和流标签查找流表，进行相应的处理
- **流标签的引入使得IPv6具备了对数据包进行区分处理的能力：**
  - 流标签的使用极大地降低了数据包分类的复杂度（多元包分类->二元包分类）

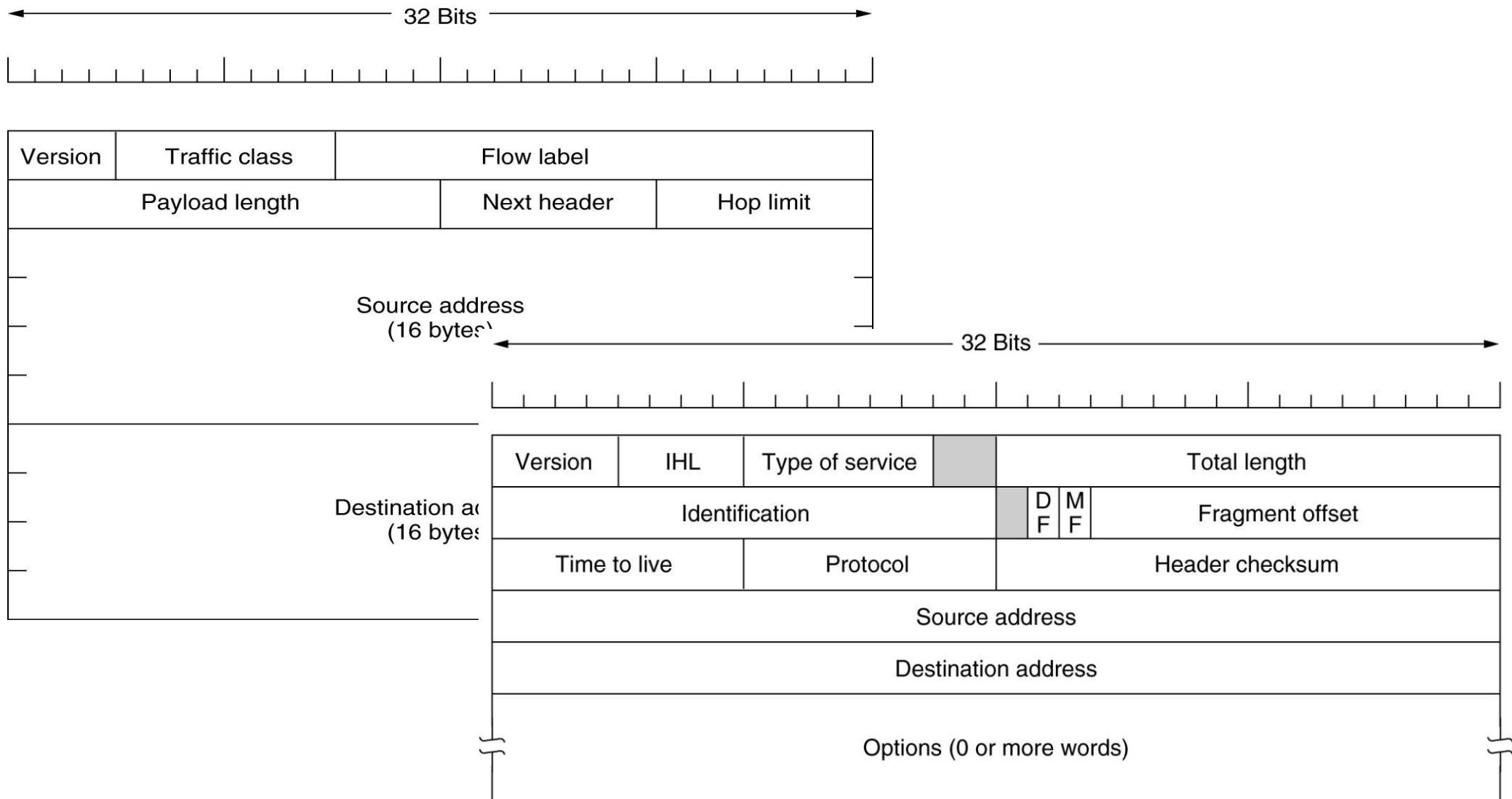
# 扩展头部



# IPv6包格式

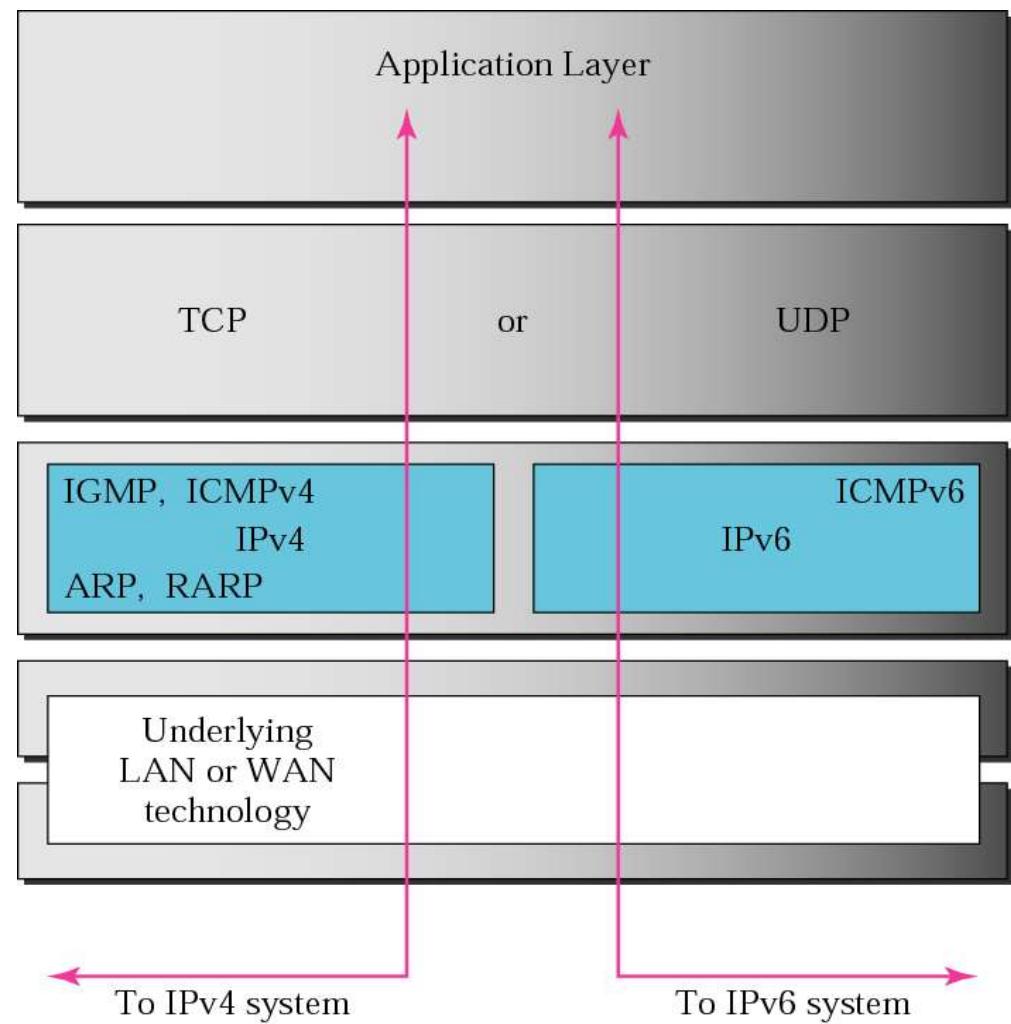
- 与IPv4固定头相比， IPv6的基本头中去掉了以下字段：
  - IHL： IPv6的基本头总是40字节长
  - 与分片相关的字段： IPv6路由器不负责分片
  - 头校验：计算校验和太花时间，现在的网络非常可靠，且链路层和传输层上往往都有差错检测
- IPv6基本头中增加了以下字段：
  - 流标签：支持对数据包区分处理
- IPv6基本头改变了以下字段的作用：
  - Type of Service：代之以Traffic Class
  - 总长度：代之以载荷长度
  - Protocol：代之以Next header，允许任意扩展选项

# IPv6的基本头与IPv4的固定头



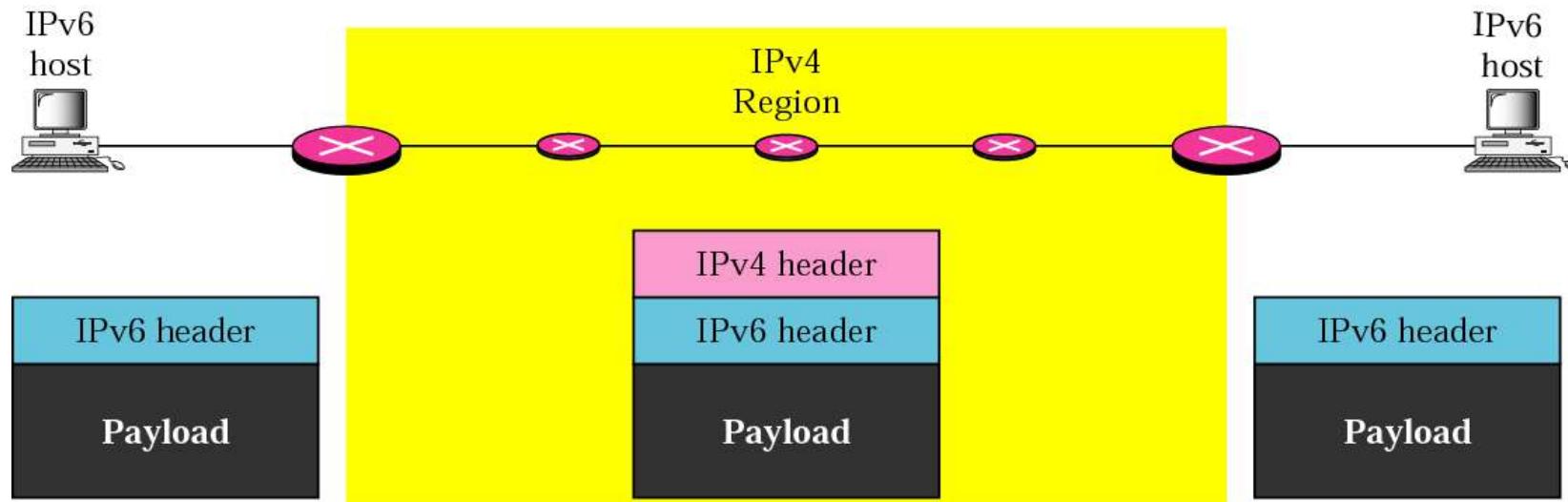
# 从IPv4过渡到IPv6

- 因特网不可能一夜之间升级到IPv6， IPv4与IPv6如何共存？
- 采用IPv4/IPv6双协议栈：
  - 支持IPv6的主机和路由器同时运行IPv4和IPv6
  - 运行双栈的源节点先对目的节点查询DNS：若DNS返回IPv4地址，发送IPv4分组；若返回IPv6地址，发送IPv6分组
  - 双栈节点同时拥有IPv4和IPv6地址

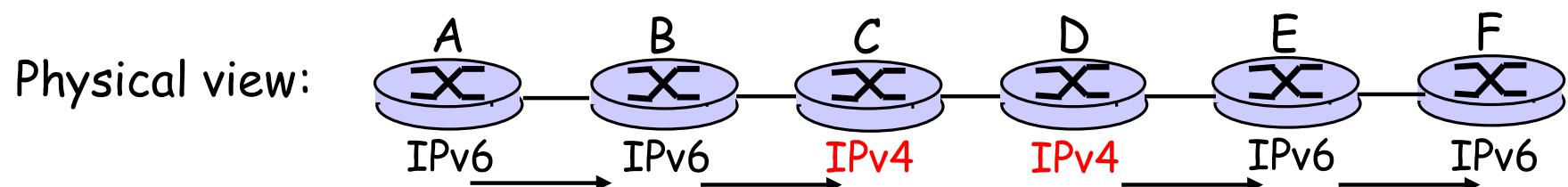
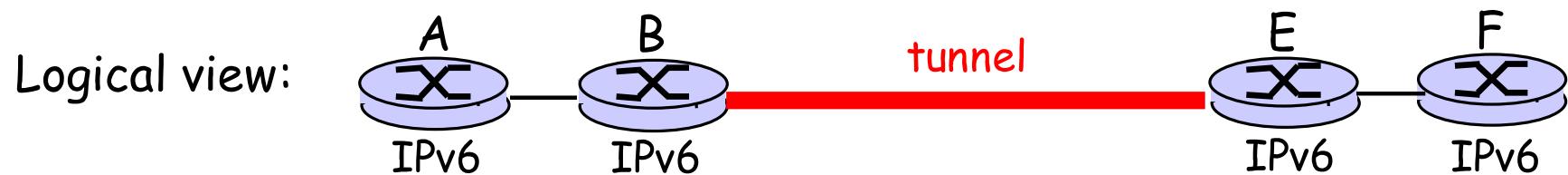


# IPv6数据包如何穿越IPv4网络？

- 两台支持**IPv6**的主机中间隔着一个运行**IPv4**的网络，**IPv6**数据报如何穿越**IPv4**网络？
- 一种有效的方法是建立隧道（**tunneling**）：
  - **IPv6/IPv4**边界路由器将**IPv6**包封装到一个**IPv4**包中，送入**IPv4**网络，目的边界路由器取出**IPv6**包继续传输

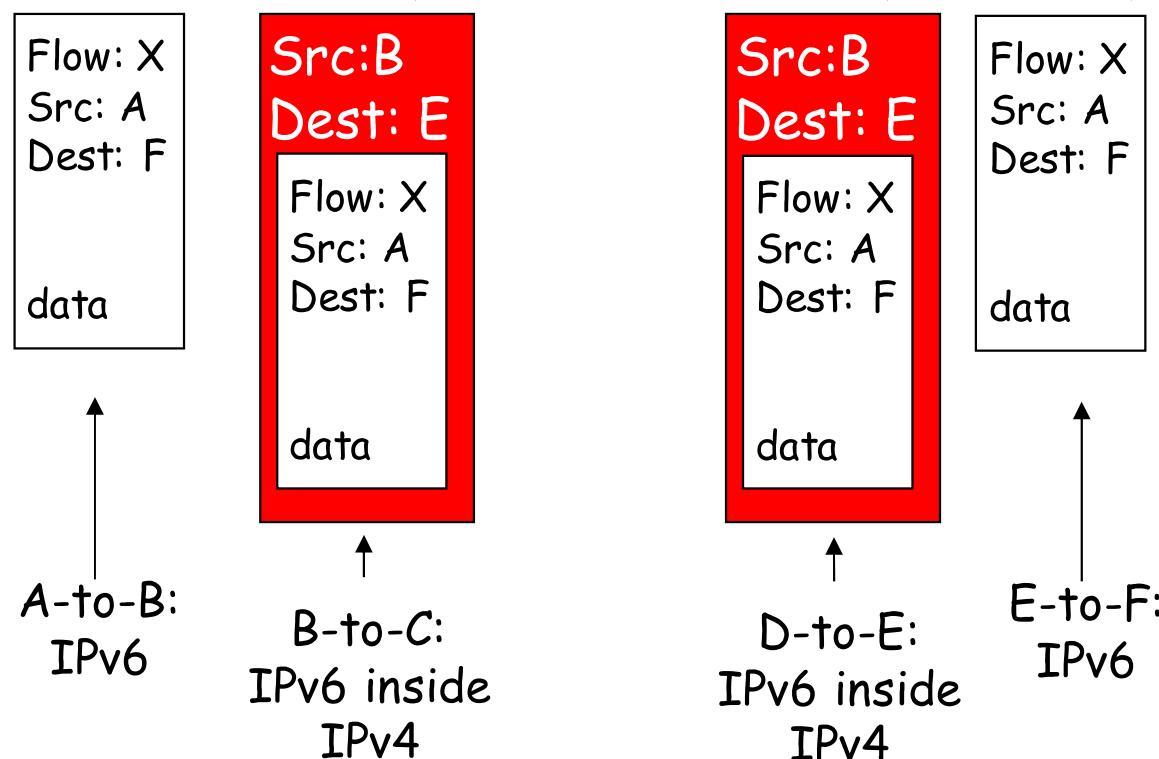


# 建立隧道的方法：封装数据包

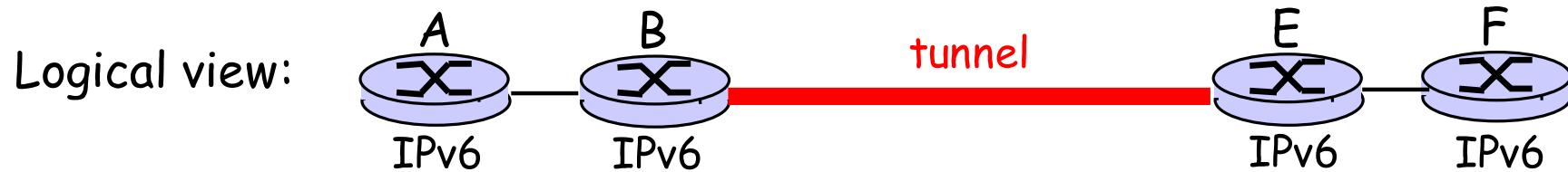


外层IPv4数据报：

- Src IP =  
    B的IPv4地址
- Dest IP =  
    E的IPv4地址



# 隧道技术的要点



□ 隧道技术的本质是封装数据包：

- 当分组需要穿过一个不同协议的网络时，将分组封装在中间网络的分组中传输
- 执行封装和解封装的路由器端口称为隧道的端点

□ 为什么称为隧道？

- 一个类比：IPv4路由器是检查站，每个经过的数据报都会被检查，但在隧道中传输的IPv6包逃避了检查
- IPv6包在IPv4网络中传输时，没有被IPv4路由器检查和处理，特别是，报头中的TTL未减1
- 在路由器E的IPv6看来，IPv6包是从路由器B直接到来的

# 重要知识点

- IPv4与IPv6不兼容
- 隧道技术的要点和应用
- 如何解决IPv4与IPv6共存：
  - 双协议栈 + 隧道技术

# 我国IPv6的发展现状

- IPv6网络高速公路已经建成：
  - 13个骨干网直联点全部实现IPv6互联互通
  - IPv6的国际出入口带宽（90Gbps）也已经开通
- 已建成规模最大的IPv6网络：
  - 网络基础设施全部支持IPv6，规模全球领先
  - 已申请的IPv6地址资源位居全球第一
  - 2020年覆盖50%的互联网用户
- 已建成业务形态最全的IPv6网络
  - 2018年底全面应用，从商业网站到政府公网全面支持IPv6
  - 数据中心、云产品、内容分发等应用基础设施，已初步具备全国全网IPv6的支持服务能力
- 2021-2025年为向以IPv6为核心的网络演进的阶段，下一代互联网将全面建成，IPv6流量在网络占主体

# 我国IPv6的发展现状

- 2015年6月开始实施“雪人计划”，面向全球招募了25个根服务器运营志愿单位，正式运营IPv6根服务器和域名解析系统
- 至2017年，已在全球部署了25台IPv6根服务器：
  - 3台主根服务器：分别位于中国、美国和日本
  - 22台辅根服务器：其中3台在中国
- 我国已初步形成端到端的下一代互联网体系，终结了美国主导30多年的互联网霸权
- 当今世界，传统互联网正进入IPv6时代，移动互联网正步入5G时代，我国在技术上完成了逆转，在规模上实现了超越，总体优势明显

# Chapter 4: done!

4.1 Overview of Network layer: data plane and control plane

\*\*Virtual circuit and datagram networks

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

4.4 Generalized Forward and SDN

- match plus action
- OpenFlow example

**Question:** 转发表是如何得到的？

**Answer:** 通过控制面（下一章）

# 作业

- 习题: 11月19日
  - 5, 7, 8, 10, 12, 14
- 实验: 11月24日
  - IP