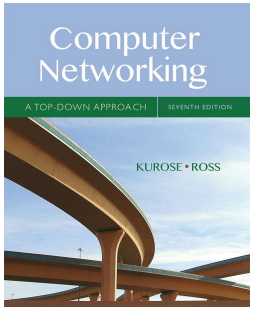
第 4 章
网络层：数据平面

中国科学技术大学
自动化系 郑烇
改编自Jim kurose,Keith Ross



Computer Networking: A Top Down Approach
7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

第4章 网络层：数据平面

- 4. 1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

第4章 网络层：数据平面

- 4. 1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

第4章 网络层：数据平面

本章目标:

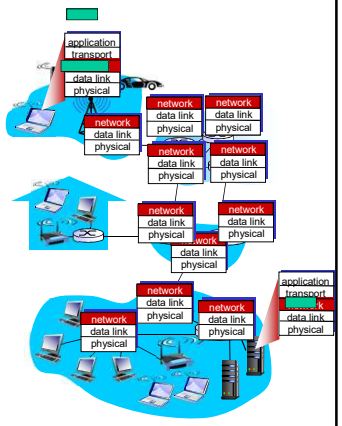
- 理解网络服务的基本原理，聚焦于其数据平面
  - 网络服务模型
  - 转发和路由
  - 路由器工作原理
  - 通用转发
- 互联网中网络层协议的实例和实现

网络层服务

网络层提供的服务

- 在发送主机和接收主机对之间传送段（segment）
- 在发送端将段封装到数据报中
- 在接收端，将段上交给传输层实体
- 网络层协议存在于每一个主机和路由器
- 路由器检查每一个经过它的IP数据报的头部

将段不断地进行封装解封装
最后进行一次大的解封装



网络层的关键功能

局部概念：

网络层功能：

- *转发*: 将分组从路由器的输入接口转发到合适的输出接口
- *路由*: 使用路由算法来决定分组从发送主机到目标接收主机的路径
  - 路由选择算法
  - 路由选择协议

全局的功能

旅行的类比：

- 转发: 通过单个路口的过程
- 路由: 从源到目的的路由路径规划过程

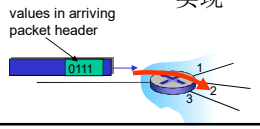
网络层：数据平面、控制平面

数据平面

- 本地，每个路由器功能
- 决定从路由器输入端口到达的分组如何转发到输出端口
- 转发功能：
  - 传统方式：基于目标地址+转发表
  - SDN方式：基于多个字段+流表

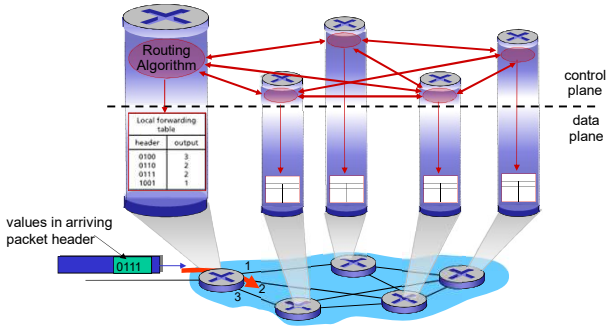
控制平面

- 网络范围内的逻辑
- 决定数据报如何在路由器之间路由，决定数据报从源到目标主机之间的端到端路径
- 2个控制平面方法：
  - 传统的路由算法: 在路由器中被实现
  - *software-defined networking (SDN)*: 在远程的服务器中实现

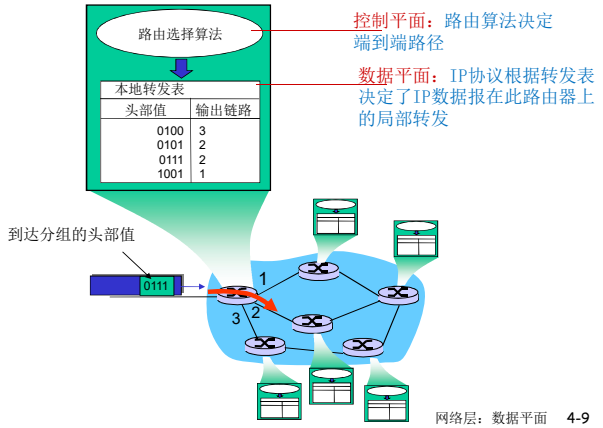


传统方式：每-路由器(Per-router)控制平面

在每一个路由器中的单独路由器算法元件，在控制平面进行交互



传统方式：路由和转发的相互作用

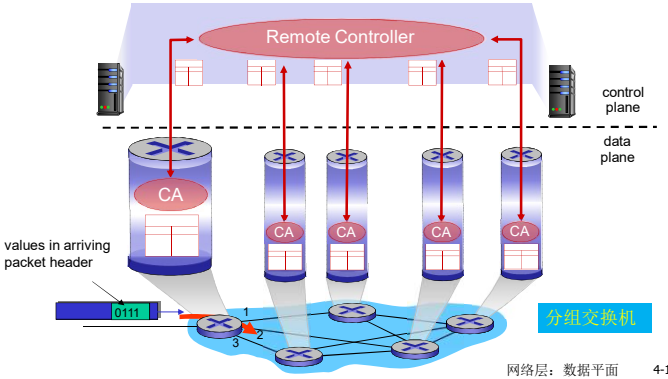


控制平面：路由算法决定端到端路径

数据平面：IP协议根据转发表决定了IP数据报在此路由器上的局部转发

SDN方式：逻辑集中的控制平面

一个不同的（通常是远程的）控制器与本地控制代理（CAs）交互



网络服务模型

Q: 从发送方主机到接收方主机传输数据报的“通道”，网络提供什么样的服务模型？

对于单个数据报的服务:

- 可靠传送
- 延迟保证，如：少于40ms的延迟

对于数据报流的服务:

- 保序数据报传送
- 保证流的最小带宽
- 分组之间的延迟差

连接建立

- 在某些网络架构中是第三个重要的功能
  - ATM, frame relay, X.25
- 在分组传输之前，在两个主机之间，在通过一些路由器所构成的路径上建立一个网络层连接
  - 涉及到路由器
- 网络层和传输层连接服务区别:
  - 网络层: 在2个主机之间，涉及到路径上的一些路由器
  - 传输层: 在2个进程之间，很可能只体现在端系统上(TCP连接)

网络层服务模型:

网络架构	服务模型	保证？ 带宽	丢失	保序	延迟	拥塞反馈
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR 恒定速率	constant rate	yes	yes	yes	no congestion
ATM	VBR 变化速率	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR 可用比特率	guaranteed minimum	no	yes	no	yes
ATM	UBR 不指名比特率	none	no	yes	no	no

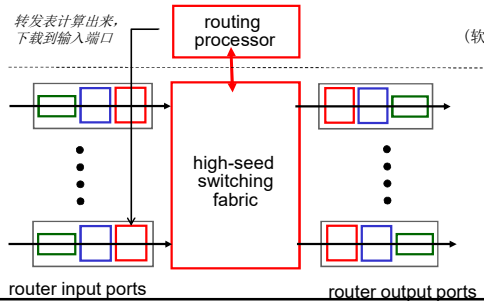
第4章 网络层：数据平面

- 4.1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

路由器结构概况

高层面(非常简化的)通用路由器体系架构

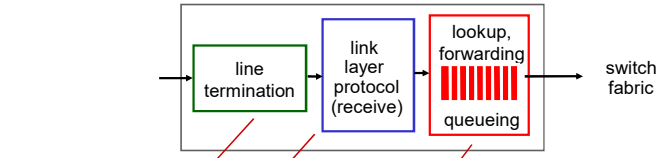
- 路由：运行**路由**选择算法 / 协议 (RIP, OSPF, BGP)-生成路由表
- 转发：从输入到输出链路交换数据报-根据路由表进行分组的**转发**



输入端口

输出端口

输入端口功能



物理层:
Bit级的接收

数据链路层:
链路层协议动作、解封装
e.g., Ethernet
见第五章

分布式交换：

- 根据数据报头部的信息如：目的地址，在输入端口内存中的转发表中查找合适的输出端口（匹配+行动）
- 基于目标的转发：仅仅依赖于IP数据报的目标IP地址（传统方法）
- 通用转发：基于头部字段的任意集合进行转发

queue ： 匹配输入和输出速度不一致性

基于目标的转发

转发表

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

Q: 但是如果地址范围如果没有划分的特别规整，会发生什么？

最长前缀匹配

longest prefix matching

当给定目标地址查找转发表时，采用最长地址前缀匹配的目标地址表项

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

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

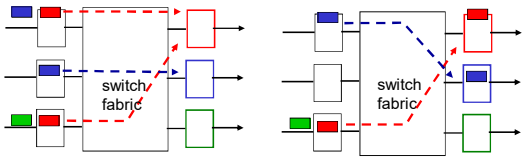
DA: 11001000 00010111 00011000 10101010 which interface?

最长前缀匹配

- 我们将会在学习IP地址时，简单讲解为什么要采用最长前缀匹配
- 最长前缀匹配：在路由器中经常采用TCAMs(ternary content addressable memories)硬件来完成
  - 内容可寻址：将地址交给TCAM，它可以在一个时钟周期内检索出地址，不管表空间有多大
  - Cisco Catalyst系列路由器: 在TCAM中可以存储多达约为1百万条路由表项

输入端口缓存

- 当交换机构的速率小于输入端口的汇聚速率时 ➡ 在输入端口可能要排队
  - *排队延迟以及由于输入缓存溢出造成丢失!*
- Head-of-the-Line (HOL) blocking: 排在队头的数据报阻止了队列中其他数据报向前移动

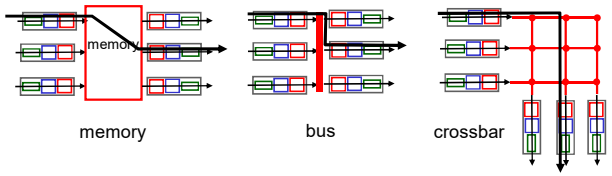


输出端口竞争：
只能有一个红色分组被传递，交换到一个输出端口.
下面红色的分组被阻塞

一个分组时间：
绿色分组经历了头端阻塞

交换结构

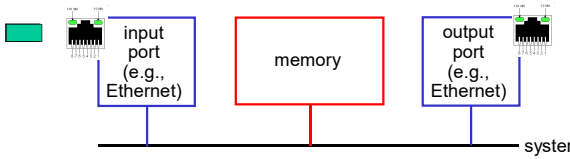
- 将分组从输入缓冲区传输到合适的输出端口
- 交换速率：分组可以按照该速率从输入传输到输出
  - 运行速度经常是输入/输出链路速率的若干倍
  - N 个输入端口：交换机构的交换速度是输入线路速度的N倍比较理想，才不会成为瓶颈
- 3种典型的交换机构



通过内存交换

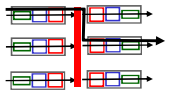
第一代路由器:

- 在CPU直接控制下的交换，采用传统的计算机
- 分组被拷贝到系统内存，CPU从分组的头部提取出目标地址，查找转发表，找到对应的输出端口，拷贝到输出端口
- 转发速率被内存的带宽限制 (数据报通过BUS两遍)
- 一次只能转发一个分组



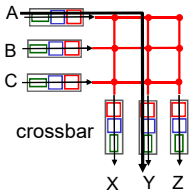
通过总线交换

- 数据报通过共享总线，从输入端口转发到输出端口
- 总线竞争: 交换速度受限于总线带宽
- 1次处理一个分组
- 1 Gbps bus, Cisco 1900； 32 Gbps bus, Cisco 5600：对于接入或企业级路由器，速度足够（但不适合区域或骨干网络）

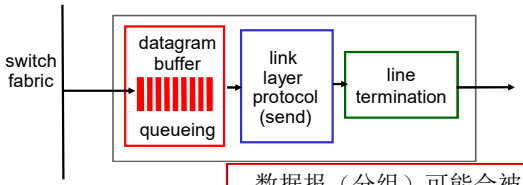


通过互联网络(crossbar等)的交换

- 同时并发转发多个分组，克服总线带宽限制
- Banyan（榕树）网络，crossbar(纵横)和其它的互联网络被开发，将多个处理器连接成多处理器
- 当分组从端口A到达，转给端口Y；控制器短接相应的两个总线
- 高级设计：将数据报分片为固定长度的信元，通过交换网络交换
- Cisco12000：以60Gbps的交换速率通过互联网络



输出端口

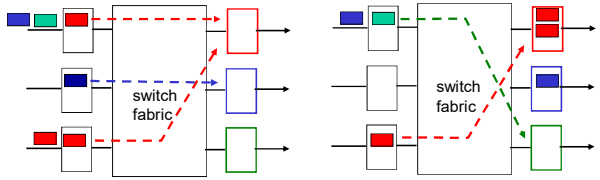


数据报（分组）可能会被丢弃，由于拥塞，缓冲区没有空间

- 当数据报从交换机构的到达速度比传输速率快就需要输出端口缓存
- 由调度规则选择排队的数据报进行传输

优先权调度-谁会获得最优性能，网络中立？

输出端口排队



- 假设交换速率R_{switch}是R_{line}的N倍（N：输入端口的数量）
- 当多个输入端口同时向输出端口发送时，缓冲该分组（当通过交换网络到达的速率超过输出速率则缓存）
- 排队带来延迟，由于输出端口缓存溢出则丢弃数据报！

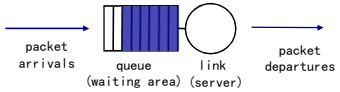
需要多少缓存?

- RFC 3439 拇指规则（经验性规则）：平均缓存大小=典型的RTT（例如：250ms）倍于链路容量C
  - e.g., C = 10 Gpbs link
  - 250ms*10Gbps=2.5 Gbit buffer
- 最近的一些推荐：有*N（非常大）*个流，缓存大小等于

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

调度机制

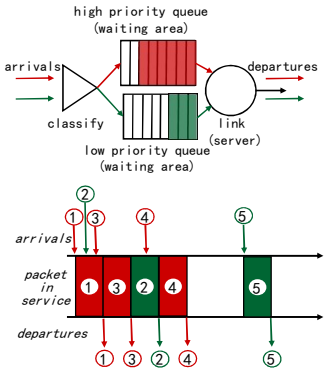
- *调度:* 选择下一个要通过链路传输的分组
- *FIFO (first in first out) scheduling:* 按照分组到来的次序发送
  - 现实例子?
  - *丢弃策略:* 如果分组到达一个满的队列，哪个分组将会被抛弃?
    - *tail drop:* 丢弃刚到达的分组
    - *priority:* 根据优先权丢失/移除分组
    - *random:* 随机地丢弃/移除



调度策略：优先权

*优先权调度：*发送最高优先权的分组

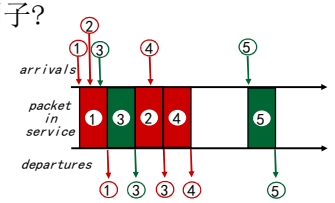
- 多类，不同类别有不同的优先权
  - 类别可能依赖于标记或者其他的头部字段，e.g. IP source/dest，port numbers，ds，etc.
  - 先传高优先级的队列中的分组，除非没有
  - 高（低）优先权中的分组传输次序：FIFO
  - 现实生活中的例子?



调度策略：其他的

Round Robin (RR) scheduling:

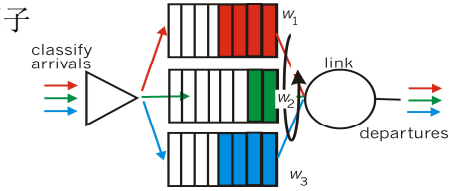
- 多类
- 循环扫描不同类型的队列，发送完一类的一个分组，再发送下一个类的一个分组，循环所有类
- 现实例子?



调度策略：其他的

Weighted Fair Queuing (WFQ):

- 一般化的Round Robin
- 在一段时间内，每个队列得到的服务时间是：$W_i/(XIGMA(W_i))$ *t，和权重成正比
- 每个类在每一个循环中获得不同权重的服务量
- 现实例子

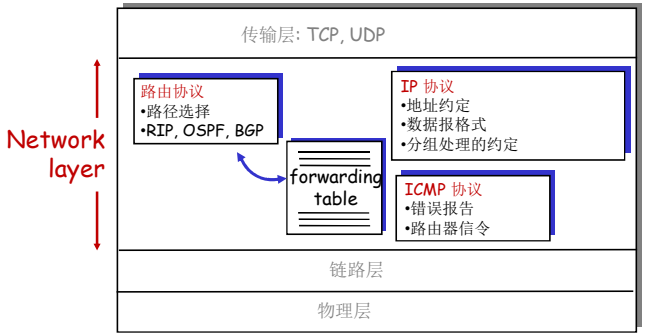


第4章 网络层：数据平面

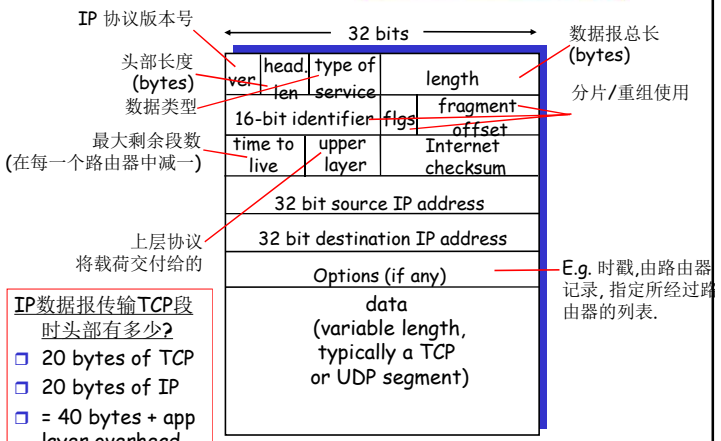
- 4.1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

互联网的网络层

主机,路由器中的网络层功能：



IP 数据报格式

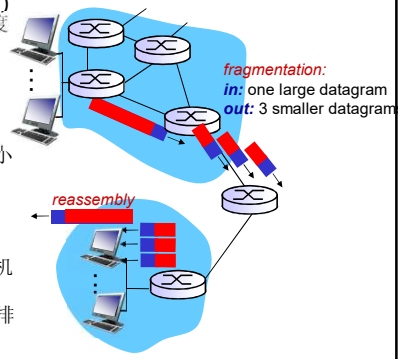


IP数据报传输TCP段时头部有多少?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

IP 分片和重组(Fragmentation & Reassembly)

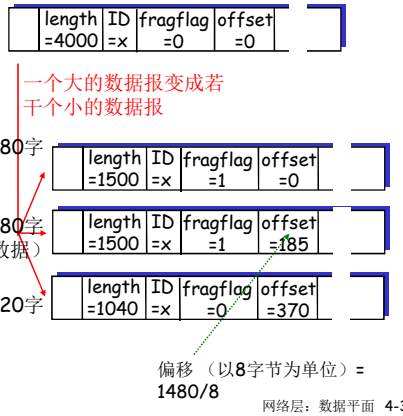
- 网络链路有MTU (最大传输单元)
  -链路层帧所携带的最大数据长度
  - 不同的链路类型
  - 不同的MTU
- 大的IP数据报在网络上被分片 (“fragmented”)
  - 一个数据报被分割成若干个小的数据报
    - 相同的ID
    - 不同的偏移量
    - 最后一个分片标记为0
  - “重组”只在最终的目标主机进行
  - IP头部的信息被用于标识，排序相关分片



IP 分片和重组

例子

- 4000 字节数据报
  - 20字节头部
  - 3980字节数据
- MTU = 1500 bytes
- 第一片：20字节头部+1480字节数据
  - 偏移量：0
- 第二片：20字节头部+1480字节数据（1480字节应用数据）
  - 偏移量：1480/8=185
- 第三片：20字节头部+1020字节数据（应用数据）
  - 偏移量：2960/8=370

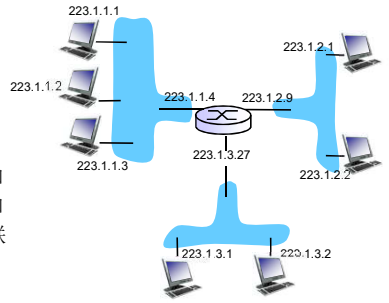


第4章 网络层：数据平面

- 4.1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

IP 编址：引论

- IP 地址：32位标示，对主机或者路由器的接口编址
- *接口*：主机/路由器和物理链路的连接处
  - 路由器通常拥有多个接口
  - 主机也有可能有多个接口
  - IP地址和每一个接口关联
- 一个IP地址和一个接口相关联



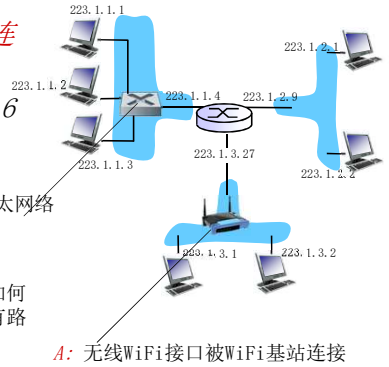
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP 编址：引论

Q：这些接口是如何连接的?

A：我们将会在第5，6章学习



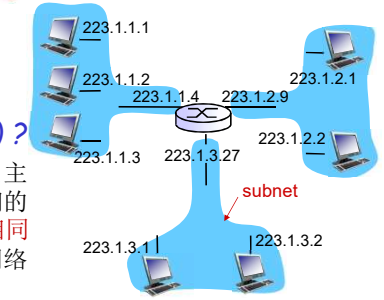
*A：*有线以太网网口链接到以太网络交换机连接

*目前：*无需担心一个接口是如何接到另外一个接口（中间没有路由器）

*A：*无线WiFi接口被WiFi基站连接

子网(Subnets)

- IP地址：
  - 子网部分(高位bits)
  - 主机部分(地位bits)
- *什么是子网(subnet)？*
  - 一个子网内的节点（主机或者路由器）它们的IP地址的高位部分相同，这些节点构成的网络的一部分叫做子网
  - 无需路由器介入，子网内各主机可以在物理上相互直接到达

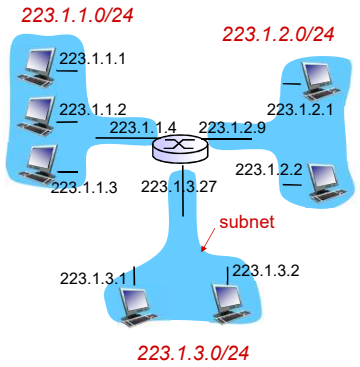


包括3个子网的网络

子网

方法：

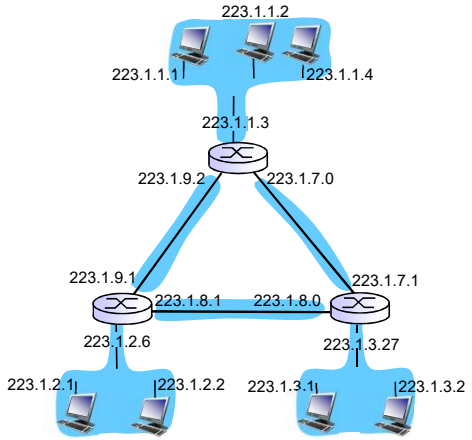
- 要判断一个子网，将每一个接口从主机或者路由器上分开，构成了一个个网络的孤岛
- 每一个孤岛（网络）都是一个都可以被称之为subnet.



子网掩码：11111111 11111111 11111111 00000000
Subnet mask: /24

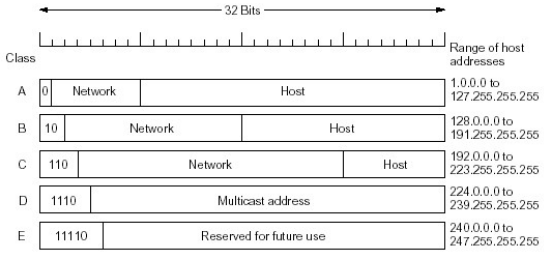
子网

几个?



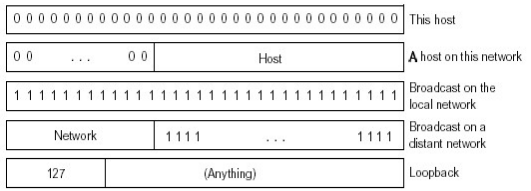
IP 地址分类

- Class A：126 networks， 16 million hosts
- Class B：16382networks， 64 K hosts
- Class C：2 million networks， 254 host
- Class D：multicast
- Class E：reserved for future



特殊IP地址

- 一些约定：
  - 子网部分：全为 0---本网络
  - 主机部分：全为0---本主机
  - 主机部分：全为1--广播地址，这个网络的所有主机
- 特殊IP地址



内网(专用)IP地址

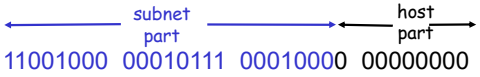
- 专用地址：地址空间的一部份供专用地址使用
- 永远不会被当做公用地址来分配，不会与公用地址重复
  - 只在局部网络中有意义，区分不同的设备
- 路由器不对目标地址是专用地址的分组进行转发
- 专用地址范围
  - Class A 10.0.0.0-10.255.255.255 MASK 255.0.0.0
  - Class B 172.16.0.0-172.31.255.255 MASK 255.255.0.0
  - Class C 192.168.0.0-192.168.255.255 MASK 255.255.255.0

IP 编址: CIDR

CIDR: Classless InterDomain Routing

（无类域间路由）

- 子网部分可以在任意的位置
- 地址格式: a.b.c.d/x, 其中 x 是 地址中子网号的长度



200.23.16.0/23

子网掩码：11111111 11111111 11111110 00000000

子网掩码(subnet mask)

- 32bits , 0 or 1 in each bit
  - 1: bit位置表示子网部分
  - 0:bit位置表示主机部分
- 原始的A、B、C类网络的子网掩码分别是
  - A：255.0.0.0 ：11111111 00000000 0000000 00000000
  - B：255.255.0.0：11111111 11111111 0000000 00000000
  - C：255.255.255.0：11111111 11111111 11111111 00000000
- CIDR下的子网掩码例子：
  - 11111111 11111111 11111100 00000000
- 另外的一种表示子网掩码的表达方式
  - /#
  - 例：/22：表示前面22个bit为子网部分

转发表和转发算法

Destination Subnet Num	Mask	Next hop	Interface
202.38.73.0	255.255.255.192	IPx	Lan1
202.38.64.0	255.255.255.192	IPy	Lan2
.....			
Default	-	IPz	Lan0

- 获得IP数据报的目标地址
- 对于转发表中的每一个表项
  - ✓如 (IP Des addr) & (mask)== destination, 则按照表项对应的接口转发该数据报
  - ✓ 如果都没有找到,则使用默认表项转发数据报

如何获得一个IP地址

Q: 主机如何获得一个IP地址?

- 系统管理员将地址配置在一个文件中
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: 从服务器中动态获得一个IP地址
  - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

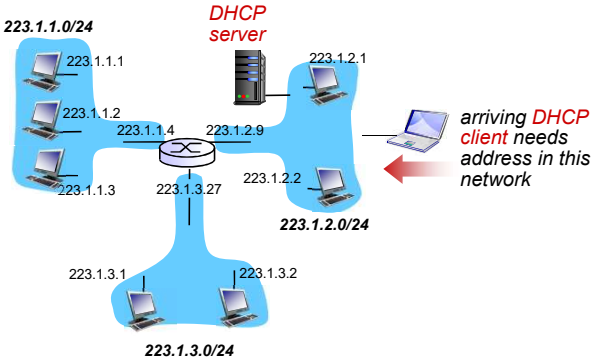
目标: 允许主机在加入网络的时候，动态地从服务器那里获得IP地址：

- 可以更新对主机在用IP地址的租用期-租期快到了
- 重新启动时，允许重新使用以前用过的IP地址
- 支持移动用户加入到该网络（短期在网）

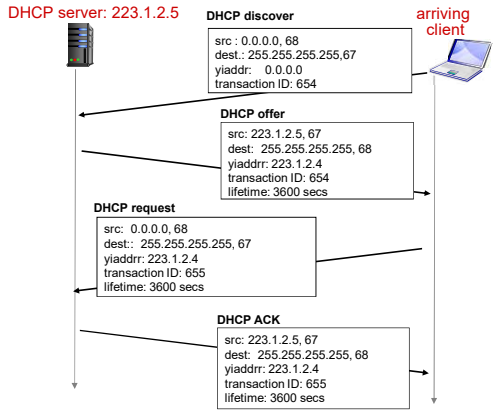
DHCP工作概况:

- 主机广播“DHCP discover”报文[可选]
- DHCP 服务器用 “DHCP offer”提供报文响应[可选]
- 主机请求IP地址：发送 “DHCP request” 报文
- DHCP服务器发送地址：“DHCP ack” 报文

DHCP client-server scenario



DHCP client-server scenario

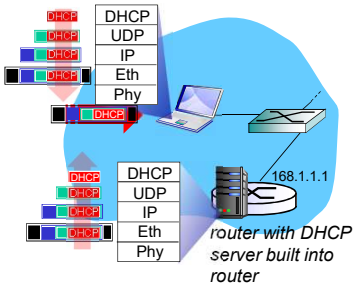


DHCP: 不仅仅是IP addresses

DHCP 返回:

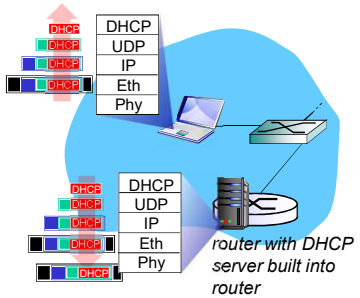
- IP 地址
- 第一跳路由器的IP地址（默认网关）
- DNS服务器的域名和IP地址
- 子网掩码 (指示地址部分的网络号和主机号)

DHCP: 实例



- 联网笔记本需要获取自己的IP地址，第一跳路由器地址和DNS服务器：采用DHCP协议
- DHCP 请求被封装在UDP段中，封装在IP数据报中，封装在以太网的帧中
- 以太网帧在局域网范围内广播（dest: FFFFFFFFFFFF），被运行DHCP服务的路由器收到
- 以太网帧解封装成IP，IP解封装成UDP，解封装成DHCP

DHCP: 实例



- DHCP服务器生成DHCP ACK，包含客户端的IP地址，第一跳路由器的IP地址和DNS域名服务器的IP地址
- DHCP服务器封装的报文所在的帧转发到客户端，在客户端解封装成DHCP报文
- 客户端知道它自己的IP地址，DNS服务器的名字和IP地址，第一跳路由器的IP地址

DHCP: Wireshark 输出(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."
```

如何获得一个IP地址

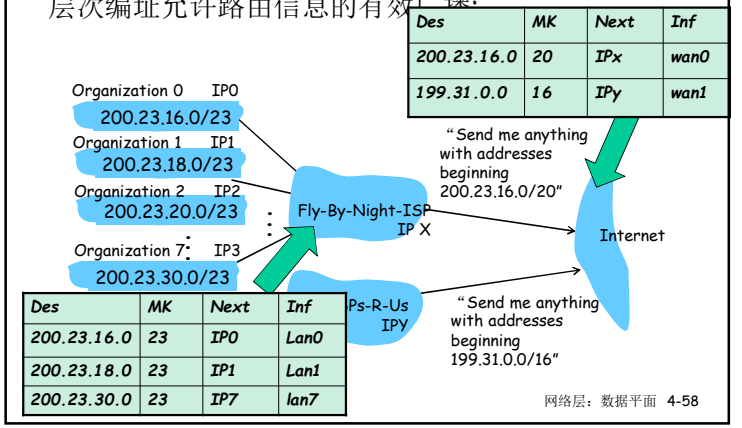
Q: 如何获得一个网络的子网部分?

A: 从ISP获得地址块中分配一个小地址块

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

层次编址: 路由聚集（route aggregation）

层次编址允许路由信息的有效广播:



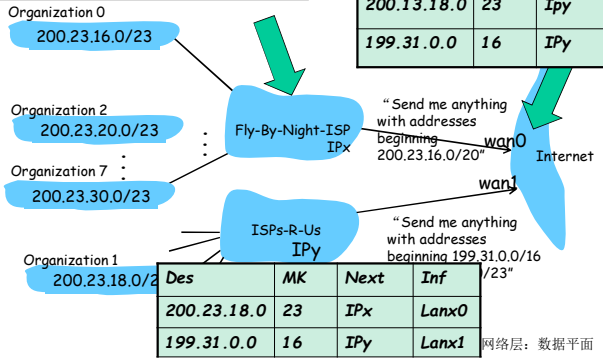
Des	MK	Next	Inf
200.23.16.0	20	IPx	wan0
199.31.0.0	16	IPy	wan1

Des	MK	Next	Inf
200.23.16.0	23	IP0	Lan0
200.23.18.0	23	IP1	Lan1
200.23.30.0	23	IP7	lan7

层次编址: 特殊路由信息(more specific routes)

Des	MK	Next	Inf
200.23.16.0	23	IP0	Lan0
200.23.20.0	23	IP2	Lan1
200.23.30.0	23	IP7	lan7

Des	MK	Next	Inf
200.23.16.0	20	IPx	wan0
200.13.18.0	23	Ipy	wan1
199.31.0.0	16	IPy	wan1



Des	MK	Next	Inf
200.23.18.0	23	IPx	Lanx0
199.31.0.0	16	IPy	Lanx1

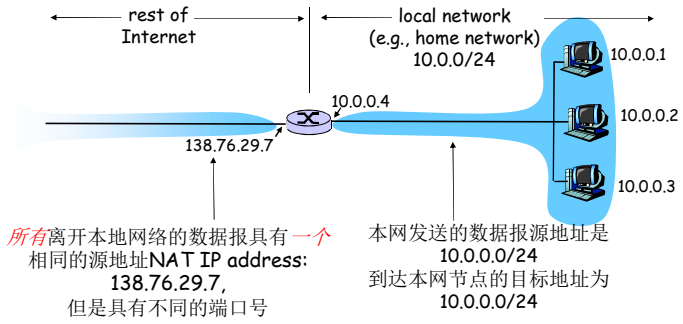
IP 编址: 如何获得一块地址

Q: 一个ISP如何获得一个地址块?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- 分配地址
- 管理DNS
- 分配域名，解决冲突

NAT: Network Address Translation



NAT: Network Address Translation

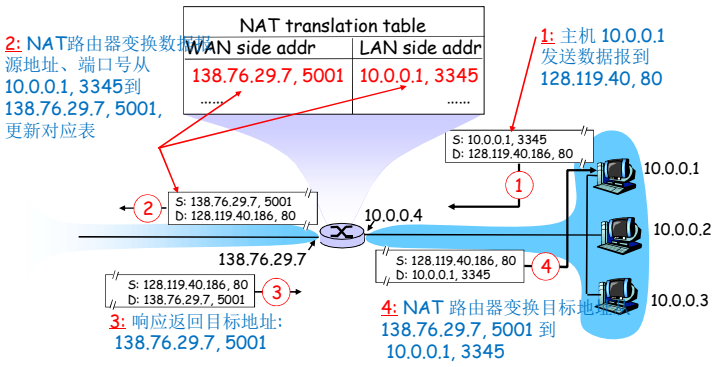
- 动机：本地网络只有一个有效IP地址：
  - 不需要从ISP分配一块地址，可用一个IP地址用于所有的（局域网）设备--省钱
  - 可以在局域网改变设备的地址情况下而无须通知外界
  - 可以改变ISP（地址变化）而不需要改变内部的设备地址
  - 局域网内部的设备没有明确的地址，对外是不可见的--安全

NAT: Network Address Translation

实现：NAT 路由器必须：

- **外出数据包**：替换源地址和端口号为NAT IP地址和新的端口号，目标IP和端口不变
  ...远端的C/S将会用NAP IP地址，新端口号作为目标地址
- 记住每个转换替换对（在NAT转换表中）
  .. 源IP，端口 vs NAP IP，新端口
- **进入数据包**：替换目标IP地址和端口号，采用存储在NAT表中的mapping表项，用（源IP，端口）

NAT: Network Address Translation

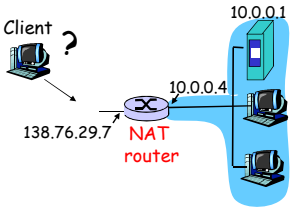


NAT: Network Address Translation

- 16-bit端口字段:
  - 6万多个同时连接，一个局域网!
- 对NAT是有争议的:
  - 路由器只应该对第3层做信息处理，而这里对端口号（4层）作了处理
  - 违反了end-to-end 原则
    - 端到端原则：复杂性放到网络边缘
      - 无需借助中转和变换，就可以直接传送到目标主机
    - NAT可能要被一些应用设计者考虑, eg, P2P applications
    - 外网的机器无法主动连接到内网的机器上
  - 地址短缺问题可以被IPv6 解决
  - NAT穿越： 如果客户端需要连接在NAT后面的服务器，如何操作

NAT 穿越问题（略）

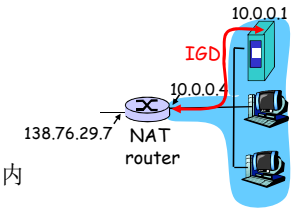
- 客户端需要连接地址为10.0.0.1的服务器
  - 服务器地址10.0.0.1 LAN本地地址 (客户端不能够使用其作为目标地址)
  - 整网只有一个外部可见地址: 138.76.29.7
- 方案1: 静态配置NAT：转发进来的对服务器特定端口连接请求
  - e.g., (123.76.29.7, port 2500) 总是转发到10.0.0.1 port 25000



NAT 穿越问题（略）

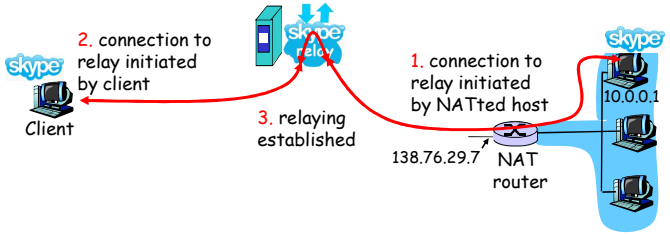
- 方案2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) 协议. 允许NATted主机可以:
  - 获知网络的公共 IP地址 (138.76.29.7)
  - 列举存在的端口映射
  - 增/删端口映射 (在租用时间内)

  i.e., 自动化静态NAT端口映射配置



NAT 穿越问题（略）

- 方案 3: 中继 (used in Skype)
  - NAT后面的服务器建立和中继的连接
  - 外部的客户端链接到中继
  - 中继在2个连接之间桥接



第4章 网络层：数据平面

- 4.1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

IPv6：动机

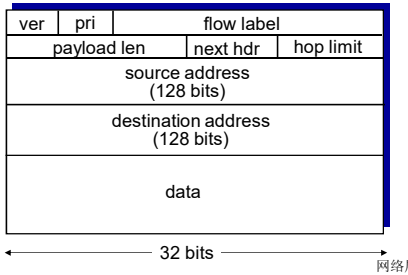
- 初始动机：32-bit地址空间将会被很快用完
- 另外的动机：
  - 头部格式改变帮助加速处理和转发
    - TTL-1
    - 头部checksum
    - 分片
  - 头部格式改变帮助QoS

IPv6 数据报格式：

- 固定的40 字节头部
- 数据报传输过程中，不允许分片

IPv6 头部 (Cont)

Priority: 标示流中数据报的优先级
Flow Label: 标示数据报在一个“flow.”
（“flow”的概念没有被严格的定义）
Next header: 标示上层协议

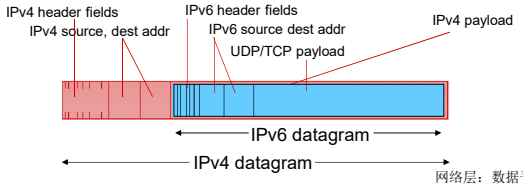


和IPv4的其它变化

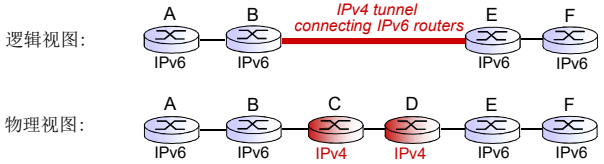
- *Checksum: 被移除掉，降低在每一段中的处理速度*
- *Options:* 允许，但是在头部之外，被 “Next Header” 字段标示
- *ICMPv6:* ICMP的新版本
  - 附加了报文类型，e.g. “Packet Too Big”
  - 多播组管理功能

从IPv4到IPv6的平移

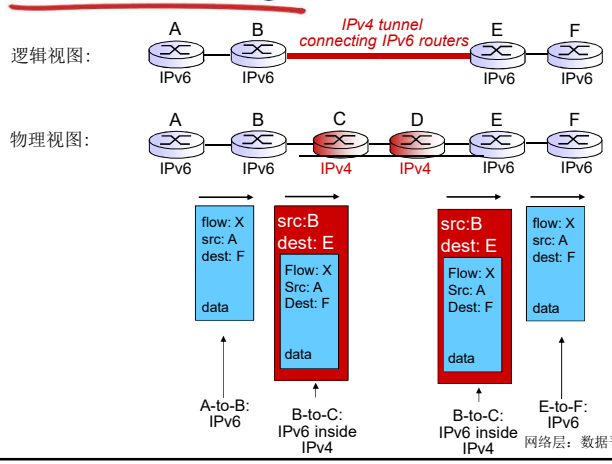
- 不是所有的路由器都能够同时升级的
  - 没有一个标记日 “flag days”
  - 在IPv4和IPv6路由器混合时，网络如何运转?
- *隧道*: 在IPv4路由器之间传输的IPv4数据报中携带IPv6数据报



隧道(Tunneling)



隧道(Tunneling)



IPv6：应用

- Google: 8% 的客户通过IPv6访问谷歌服务
- NIST: 全美国1/3的政府域支持IPv6
- *估计还需要很长时间进行部署*
  - 20年以上!
  - 看看过去20年来应用层面的变化: WWW, Facebook, streaming media, Skype, …
  - *为什么?*

第4章 网络层：数据平面

- 4.1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

网络层功能为例的数据平面和控制平面

网络层功能：

- *转发：* 对于从某个端口到来的分组转发到合适的输出端口
- *路由：* 决定分组从源端到目标端的路径
  - *路由算法*

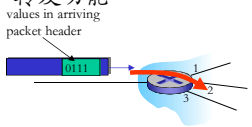
类比: 旅行

- *转发：* 一个多岔路口的进入和转出过程
- *路由:* 规划从源到目标的旅行路径

网络层: 数据平面和控制平面

数据平面

- 本地的、每个路由器的功能
- 决定某个从某个端口进入的分组从从哪个端口输出
- 转发功能

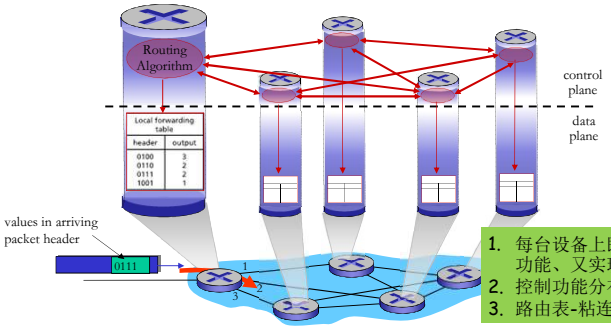


控制平面

- 网络范围的逻辑
- 决定分组端到端穿行于各个路由器的路径

每个路由器(Per Route)的控制平面

每个路由器上都有实现路由算法元件（它们之间需要相互交互）- 形成传统IP实现方式的控制平面



1. 每台设备上既实现控制功能、又实现数据平面
2. 控制功能分布式实现
3. 路由表-粘连

数量众多、功能各异的中间盒

- 路由器的网络层功能：
  - IP转发：对于到来的分组按照路由表决定如何转发，数据平面
  - 路由：决定路径，计算**路由表**；处在控制平面
- 还有其他种类繁多网络设备（中间盒）：
  - 交换机；防火墙；NAT；IDS；负载均衡设备
  - 未来：不断增加的需求和相应的网络设备
  - 需要不同的设备去实现不同的网络功能
    - 每台设备集成了控制平面和数据平面的功能
    - 控制平面分布式地实现了各种控制平面功能
    - 升级和部署网络设备非常困难

网络设备控制平面的实现方式特点

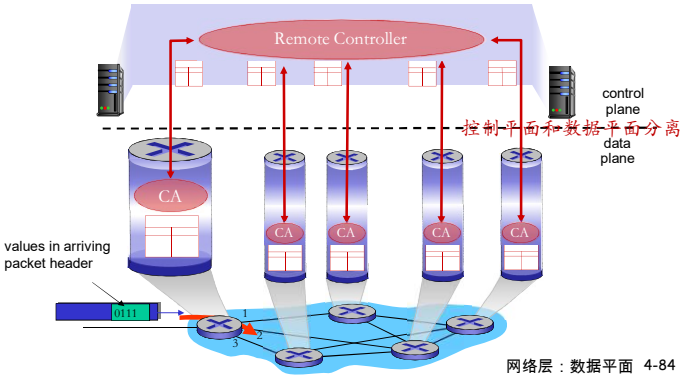
- 互联网网络设备：传统方式都是通过分布式，每台设备的方法来实现数据平面和控制平面功能
  - *垂直集成*：*每台*路由器或其他网络设备，包括：
    - 1）硬件、在私有的操作系统；
    - 2）互联网标准协议(IP, RIP, IS-IS, OSPF, BGP)的私有实现
    - 从上到下都由一个厂商提供（代价大、被设备上“绑架”“）
  - 每个设备都实现了数据平面和控制平面的事情
    - 控制平面的功能是分布式实现的
  - 设备基本上只能（分布式升级困难）按照固定方式工作，控制逻辑固化。不同的网络功能需要不同的“middleboxes”：防火墙、负载均衡设备、NAT boxes,..
- （数据+控制平面）集成>（控制逻辑）分布->固化
  - 代价大；升级困难；管理困难等

传统方式实现网络功能的问题

- 问题：
  - 垂直集成>昂贵、不便于创新的生态
  - 分布式、固化设备功能==网络设备种类繁多
    - 无法改变路由等工作逻辑，无法实现流量工程等高级特性
    - 配置错误影响全网运行；升级和维护会涉及到全网设备：**管理困难**
    - 要增加新的网络功能，需要设计、实现以及部署新的特定设备，设备种类繁多
- ~2005: 开始重新思考网络控制平面的处理方式
  - 集中：远程的控制器集中实现控制逻辑
  - 远程：数据平面和控制平面的分离

SDN：逻辑上集中的控制平面

一个不同的（通常是远程）控制器和CA交互，控制器决定分组转发的逻辑（可编程），CA所在设备执行逻辑。



SDN的主要思路

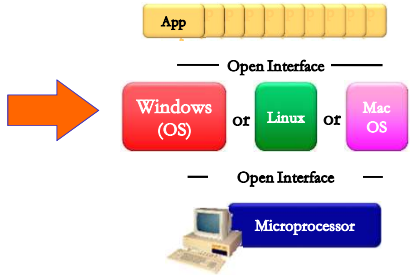
- 网络设备数据平面和控制平面分离
- 数据平面-分组交换机
  - 将路由器、交换机和目前大多数网络设备的功能进一步**抽象**成：按照流表（由控制平面设置的控制逻辑）进行PDU（帧、分组）的动作（包括转发、丢弃、拷贝、泛洪、阻塞）
  - **统一**化设备功能：SDN交换机（分组交换机），执行控制逻辑
- 控制平面-控制器+网络应用
  - 分离、集中
  - 计算和下发控制逻辑：流表

SDN控制平面和数据平面分离的优势

- 水平集成控制平面的开放实现（而非私有实现），创造出好的产业生态，促进发展
  - 分组交换机、控制器和各种控制逻辑网络应用app可由不同厂商生产，专业化，引入竞争形成良好生态
- 集中式实现控制逻辑，网络管理容易：
  - 集中式控制器了解网络状况，编程简单，传统方式困难
  - 避免路由器的误配置
- 基于流表的**匹配+行动**的工作方式允许“可编程的”分组交换机
  - 实现流量工程等高级特性
  - 在此框架下实现各种新型（未来）的网络设备

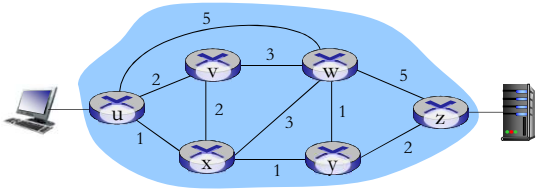
类比: 主框架到PC的演变

垂直集成
封闭，私有
创新缓慢
产业规模小



水平集成
开放接口
快速创新
产业巨大

流量工程: 传统路由比较困难

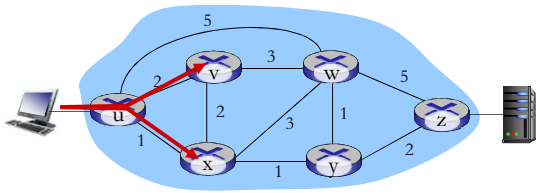


Q: 网管如果需要u到z的流量走uvwz,x到z的流量走xwyz，怎么办?

A: 需要定义链路的代价，流量路由算法以此运算（IP路由面向目标，无法操作）(或者需要新的路由算法)!

链路权重只是控制旋钮，错!

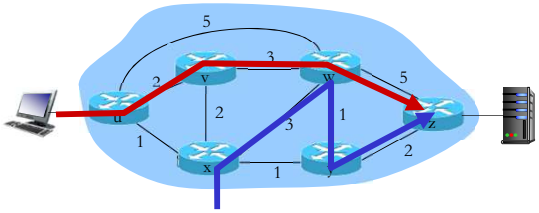
流量工程：困难



Q: 如果网管需要将u到z的流量分成2路：uvwz 和uxyz（负载均衡），怎么办？（IP路由面向目标）

A: 无法完成(在原有体系下只有使用新的路由选择算法，而在全网部署新的路由算法是个大的事情)

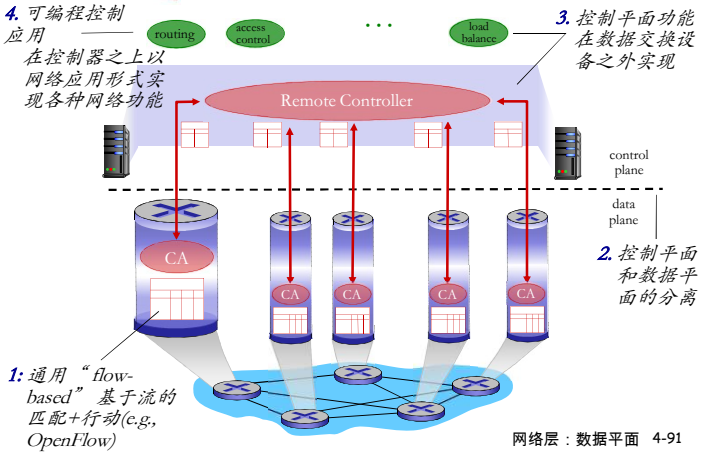
流量工程：困难



Q: 如果需要w对蓝色的和红色的流量采用不同的路由，怎么办？

A: 无法操作 (基于目标的转发，采用LS, DV 路由)

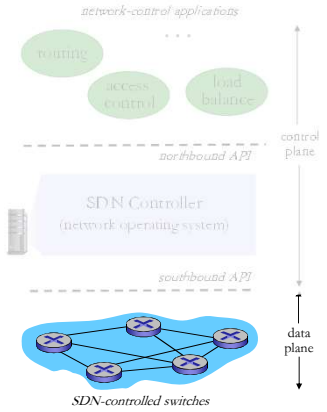
SDN特点



SDN 架构: 数据平面交换机

数据平面交换机

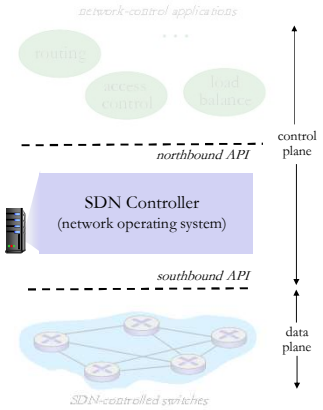
- 快速，简单，商业化交换设备采用硬件实现通用转发功能
- 流表被控制器计算和安装
- 基于南向API（例如OpenFlow），SDN控制器访问基于流的交换机
  - 定义了哪些可以被控制哪些不能
- 也定义了和控制器的协议 (e.g., OpenFlow)



SDN 架构: SDN控制器

SDN 控制器(网络OS):

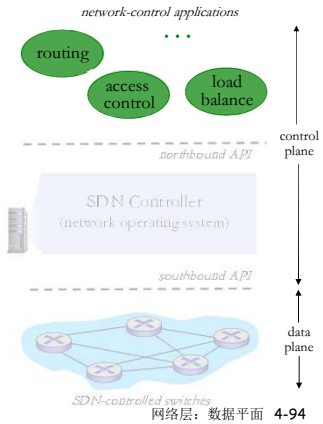
- 维护网络状态信息
- 通过上面的北向API和网络控制应用交互
- 通过下面的南向API和网络交换机交互
- 逻辑上集中，但是在实现上通常由于性能、可扩展性、容错性以及鲁棒性采用分布式方法



SDN 架构: 控制应用

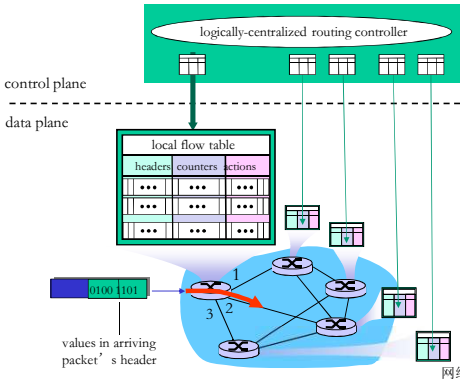
网络控制应用:

- 控制的大脑： 采用下层提供的服务（SDN控制器提供的API），实现网络功能
  - 路由器 交换机
  - 接入控制 防火墙
  - 负载均衡
  - 其他功能
- *非绑定:* 可以被第三方提供，与控制器厂商以通常上不同，与分组交换机厂商也可以不同



通用转发和SDN

每个路由器包含一个流表（被逻辑上集中的控制器计算和分发）



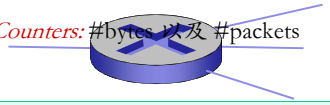
OpenFlow 数据平面抽象

- *流:* 由分组（帧）头部字段所定义
- 通用转发: 简单的分组处理规则
  - *模式:* 将分组头部字段和流表进行匹配
  - *行动:* 对于匹配上的分组，可以是丢弃、转发、修改、将匹配的分组发送给控制器
  - *优先权Priority:* 几个模式匹配了，优先采用哪个，消除歧义
  - *计数器Counters:* #bytes 以及 #packets

路由器中的流表定义了路由器的匹配+行动规则
（流表由控制器计算并下发）

OpenFlow 数据平面抽象

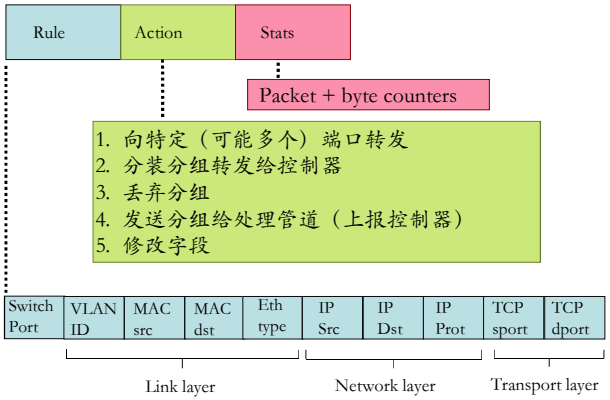
- *流*: 由头部字段所定义
- 通用转发: 简单的分组处理规则
  - *模式Pattern*: 将分组头部字段和流表进行匹配
  - *行动*: *对于匹配上的分组，可以是丢弃、转发、修改、将匹配的分组发送给控制器*
  - *优先权Priority*: 几个模式匹配了，优先采用哪个，消除歧义
  - *计数器Counters*: #bytes 以及 #packets



* : 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: 流表的表项结构



例子

基于目标的转发

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 数据报目标地址是51.6.0.8 应该被通过端口6转发

防火墙:

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

不转发（阻塞）所有具有目标TCP端口号 是22的分组

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

所有由128.119.1.1发送的分组都应该被阻塞

例子

基于层2目标的转发:

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

所有层2源MAC地址是 22:A7:23:11:E1:02都应该被向端口3转发

OpenFlow 抽象

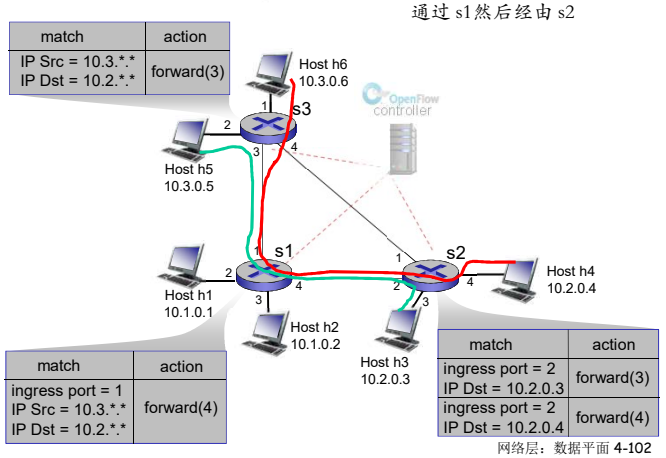
- *match+action:* 统一化各种网络设备提供的功能

- 路由器
  - *match:* 最长前缀匹配
  - *action:* 通过一条链路转发
- 交换机
  - *match:* 目标MAC地址
  - *action:* 转发或者泛
- 防火墙
  - *match:* IP地址和TCP/UDP端口号
  - *action:* 允许或者禁止
- NAT
  - *match:* IP地址和端口号
  - *action:* 重写地址和

目前几乎所有的网络设备都可以在这个匹配+行动模式框架进行描述，具体化为各种网络设备包括未来的网络设备

OpenFlow 例子

例子: 来自H5和H6的数据报应该被发向H3或者H4通过 s1然后经由 s2



第4章 网络层：数据平面

- 4.1 导论
  - 数据平面
  - 控制平面
- 4.2 路由器组成
- 4.3 IP: Internet Protocol
  - 数据报格式
  - 分片
  - IPv4地址
  - NAT：网络地址转换
  - IPv6
- 4.4 通用转发和SDN
  - SDN架构
  - 匹配
  - 行动
  - OpenFLow有关“匹配+行动”的运行实例

问题: 转发表(基于目标的转发)和流表（通用转发）是如何计算出来的？

答案: 通过控制平面(下一章)