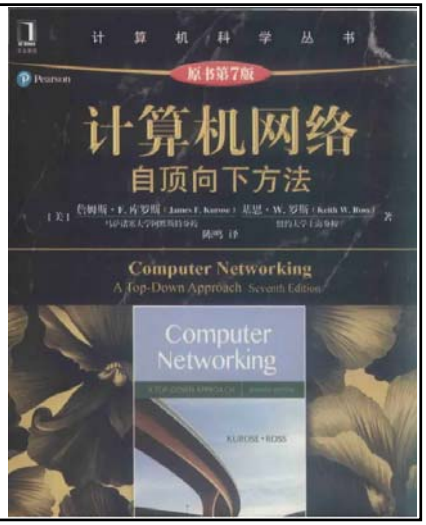
Chapter 4
网络层：数据平面
Network Layer: The Data Plane



第四章：网络层数据平面

章节目标:

- 理解网络层服务背后的原理，关注数据平面:
  - 网络层服务模型
  - 转发与路由
  - 路由器工作原理
  - 通用转发
- 实例化，在网络中的实现

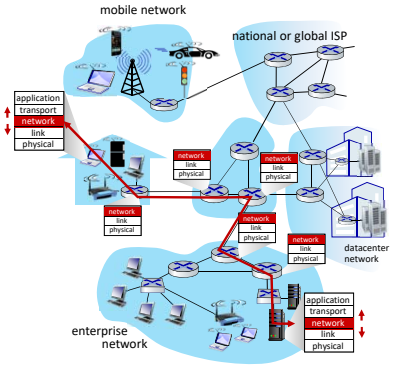
网络层：“数据平面”路线图

- 网络层：概述
  - 数据平面（ data plane ）
  - 控制平面（ control plane ）
- 路由器内部工作原理
  - 输入端口处理、交换、输出端口处理
  - 缓冲区管理、分组调度
- 网际协议IP(Internet Protocol)
  - 数据报格式
  - 编址
  - 网络地址转换
  - IPv6

- 通用转发和SDN
  - 匹配和动作
  - OpenFlow：匹配加动作
- 中间盒子(Middleboxes)

网络层服务和协议

- 负责从发送主机到接收主机的报文段传输
  - 发送方:将报文段封装为数据报，然后传递到链接层
  - 接收方:将报文段传送到传输层协议
- 每个网络设备均有网络层协议: 主机、路由器
- 路由器:
  - 检查通过它的所有IP数据报中的头部字段
  - 将数据报从输入端口移动到输出端口，沿着端到端路径传输数据报



两种重要的网络层功能

- 转发(forwarding):将数据包从路由器的输入链路移动到适当的路由器输出链路。
- 路由选择(routing):确定数据包从源到目的地所采用的路由或路径
  - 路由选择算法(routing algorithm)

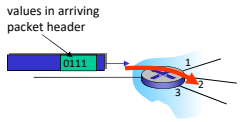
类比: 旅行

- 转发: 通过单个立交桥的过程
- 路由选择: 规划旅途从出发地到目的地行程的过程

网络层：数据平面和控制平面

数据平面(Data plane):

- 本地(local), 每个路由器功能
- 确定到达路由器输入端口的数据报如何转发到路由器输出端口的路由器本地动作

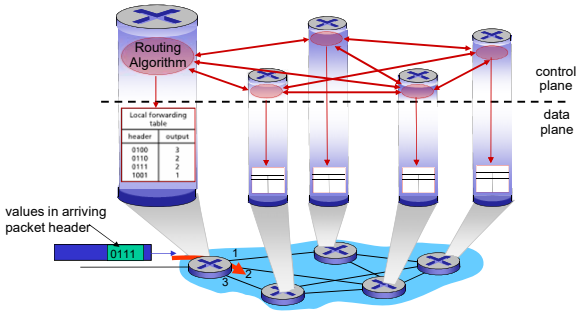


控制平面(Control plane):

- 全网(network-wide) 逻辑
- 确定数据报从源到目的所采取的端到端路径的网络范围处理过程
- 两种控制平面方法:
  - 传统路由算法: 路由选择算法运行在每台路由器中，且每台路由器包含转发和路由选择两种功能
  - 软件定义网络 (SDN)方法: 远程控制器计算和分发转发表以供每台路由器使用

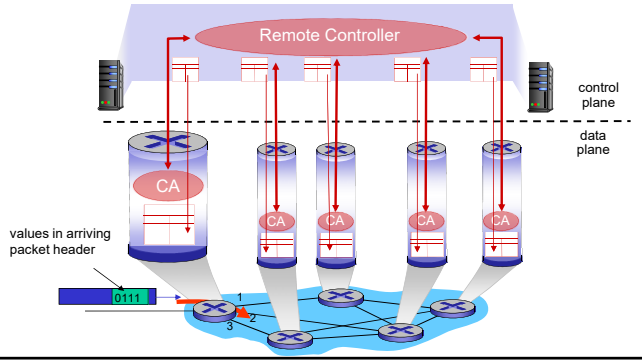
每台路由器的控制平面

每台路由器中的路由算法组件在控制平面中进行交互



软件定义网络(SDN) 控制平面

控制平面的路由选择功能与物理的路由器是分离的，即路由选择设备仅执行转发，远程控制器计算和分发转发表以供每台路由器所使用。



网络服务模型

问：网络服务模型是什么？若你来对该服务提要求，你会怎样设计？

传输层是否能指望网路层将分组交付给目的地？

当发送多个分组时，会按照发送顺序交付给接收主机的传输层吗？

发送两个连续分组的时间间隔与接收这两个分组的时间间隔相同吗？

发送主机与接收主机之间连接传输层的通道的抽象视图是什么？

也许，你希望的网络服务模型应提供如下这样的服务：

单个分组：

- 确保交付
- 具有时延上界（小于40 msec）的确保交付

多个分组：

- 有序分组交付
- 确保最小流量带宽
- 限制数据分组间距的变化

网络层服务模型-Internet的网络层

网络架构	服务模型	服务质量（QoS）保证？			
		带宽	无损	有序	实时
因特网	尽力而为	无	无	无	无

因特网“尽力而为(best effort)”服务模型

无法保证：

i. 成功将分组传送到目的地

ii. 端到端的时延和顺序

iii. 端到端流量的最小带宽

网络服务模型

网络架构	服务模型	服务质量 (QoS) 保证？			
		带宽	无损	有序	实时
因特网	尽力而为	无	无	无	无
ATM	恒定比特率(CBR)	恒定速率	有	有	有
ATM	可用比特率(ABR)	保证最小值	无	有	无
因特网	集成服务体系结构 Intserv Guaranteed (RFC 1633)	有	有	有	有
因特网	区分服务Diffserv (RFC 2475)	可能	可能	可能	无

关于尽力而为服务的思考：

- 简单的机制使互联网得以广泛部署和采用
- 充足的带宽供应允许实时应用程序（例如，交互式语音、视频）的性能在“大部分时间”内“足够好”
- 重复的、应用层的分布式服务（数据中心，内容分发网络）靠近客户的接入网，进而允许从多个位置提供服务
- “弹性”拥塞控制有一定的帮助

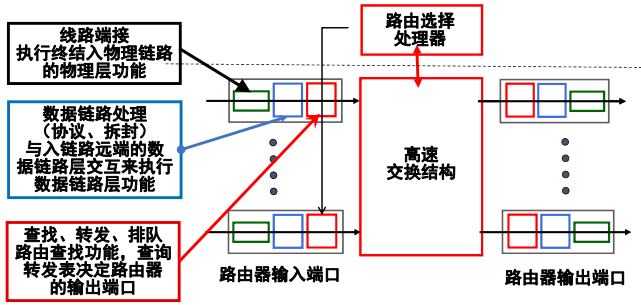
尽力而为服务模式的成功是有目共睹的

网络层：“数据平面”路线图

- 网络层：概述
  - 数据平面data plane
  - 控制平面(control plane)
- 路由器内部工作原理
  - 输入端口处理、交换、输出端口处理
  - 缓冲区管理、分组调度
- 网际协议IP(Internet Protocol)
  - 数据报格式
  - 编址
  - 网络地址转换
  - IPv6
- 通用转发和SDN
  - 匹配和动作
  - OpenFlow：匹配加动作
- 中间盒子(Middleboxes)

路由器架构概述

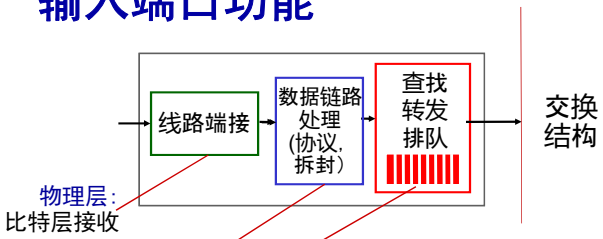
通用路由器体系结构的总体视图：



路由选择、管理
控制平面
（软件）毫秒级操作

转发
数据平面
（硬件）纳秒级操作

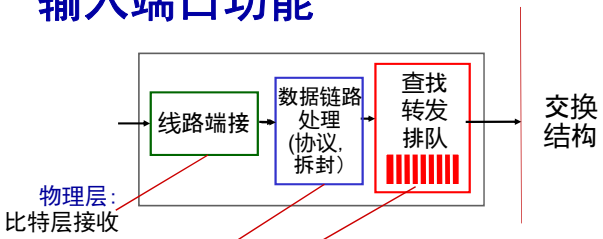
输入端口功能



分布式的交换：

- 使用头字段值，使用输入端口内存中的转发表查找输出端口（“*匹配加操作*”）
- 目标：以“线速”完成输入端口的处理，纳秒级执行
- 输入端口排队：如果数据报到达的速度快于进入交换结构的转发速度

输入端口功能



分布式的交换：

- 使用头字段值，使用输入端口内存中的转发表查找输出端口（“*匹配加操作*”）
- 基于目的地转发：仅基于目标IP地址（传统）进行转发
- 通用转发：基于任何一组头字段值进行转发

基于目的地转发

forwarding table

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

基于目的地转发

forwarding table

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

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

子集

最长前缀匹配

最长前缀匹配(longest prefix match)

在查找给定目标地址的转发表条目时，使用与目标地址匹配的最长地址前缀。

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

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

最长前缀匹配

最长前缀匹配(longest prefix match)

在查找给定目标地址的转发表条目时，使用与目标地址匹配的最长地址前缀。

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

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

最长前缀匹配

最长前缀匹配(longest prefix match)

在查找给定目标地址的转发表条目时，使用与目标地址匹配的最长地址前缀。

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

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

最长前缀匹配

最长前缀匹配(longest prefix match)

在查找给定目标地址的转发表条目时，使用与目标地址匹配的最长地址前缀。

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

match!

examples:

11001000 00010111 00010110 10100001 which interface?

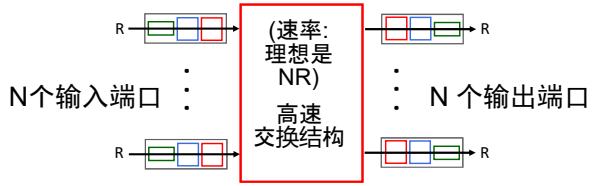
11001000 00010111 00011000 10101010 which interface?

最长前缀匹配

- 问：为什么使用最长前缀匹配？
- 最长前缀匹配：通常使用三态内容可寻址存储器（Tenary Content Address Memory, TCAM）来查找
  - 内容可寻址：向TCAM寻找地址：在一个时钟周期内检索地址，与表大小无关
  - Cisco Catalyst：能够保存100多万TCAM转发表项

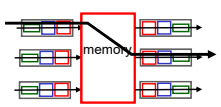
交换结构

- 将分组从输入链路传输到适当的输出链路
- 交换速率：分组从输入传输到输出的速率
  - 通常为输入/输出线速的倍数
  - N个输入：交换速率是线速的N倍

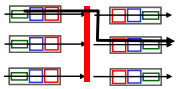


交换结构

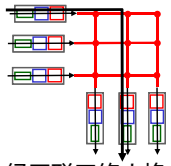
- 将数据包从输入链路传输到适当的输出链路
- 交换速率：数据包可以从输入传输到输出的速率
  - 通常为输入/输出线速的倍数
  - N个输入：交换速率是线速的N倍
- 交换结构的三种主要类型：



经内存交换



经总线交换

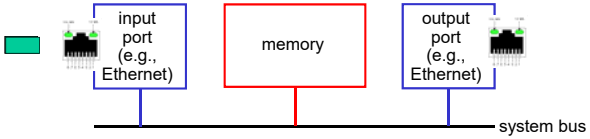


经互联网络交换

经内存交换

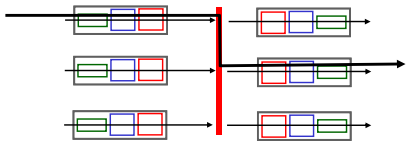
第一代路由器：

- 最简单、最早的路由器是传统的计算机，在CPU的直接控制下进行交换
- 分组被复制到处理器内存中
- 速度受到内存带宽的限制（每个分组需2次穿越系统总线）



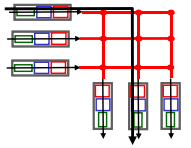
经总线交换

- 分组通过一条共享的总线从输入端口的内存传递到输出端口的内存
- 总线竞争：交换速率受限于总线的带宽
- 32 Gbps总线，Cisco 5600：对于接入网，该路由器的速度足够了

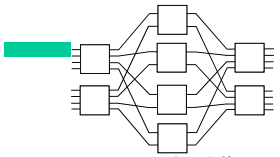


经互联网络交换

- 发展初期来自多处理器计算机体系结构中用来互联多个处理器的网络
- 纵横式交换机是由2N条总线组成的互联网络，N个输入，N个输出
- 多级交换：从多级更小的交换构建nxn交换
- 利用并行性：
  - 一个输入端口将一个分组分成K个较小的块，通过N个交换结构中的K个发送（喷射）这些块到所选择的输出端口，输出端口再将K个块装配还原成初始的分组



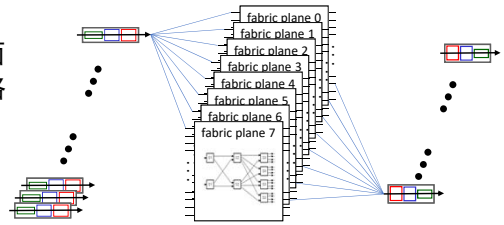
3x3 纵横交换机



8x8 多级交换
由多个更小尺寸交换建立

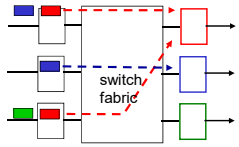
经互联网络交换

- 可扩展，并行使用多个交换“平面”：
  - 加速，并行扩展
- Cisco CRS路由器：
  - 基本单位：8个交换平面
  - 每个平面：3级内联网络
  - 高达100 Tbps的交换能力

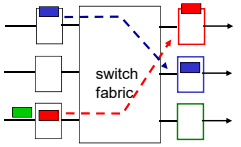


输入端口排队

- 交换网络的处理速度低于所有输入端口之和 -> 导致分组在输入端口的队列中排队
  - 由于输入缓冲区溢出导致的排队延迟和数据丢失！
- 线路前部（Head-of-the-Line，HOL）阻塞：在队列的排头上的分组挡住了其他分组的前移

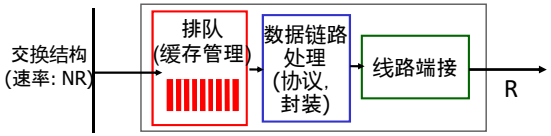


输入端口排队导致了输出端口的争用：只有一个红色分组能被传输。比如，下面的红色分组要等待（或者称为被阻塞）



一个分组时间过后：绿色的分组要等待，即使右侧输出端口无竞争，即绿色分组经历了线路前部阻塞

输出端口排队



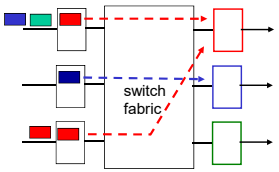
- 缓存：当来自交换网络的分组到达速度高于传输速率时，需要进行缓存。
- 丢弃策略：如果没有空闲的缓存，要丢弃哪些分组？

→ 分组可能会由于拥塞、缺少缓存而丢失

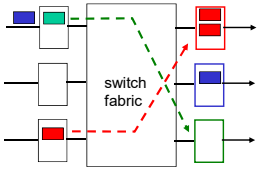
- 调度原则(Scheduling discipline) 从队列中选择分组进行传输

→ 优先级调度—谁获得最佳性能

输出端口排队



at *t*, packets more from input to output



one packet time later

- 当交换速度*超过*输出线路的速率时，需要进行缓存
- 输出端口的溢出会造成排队（延迟）和数据丢失！

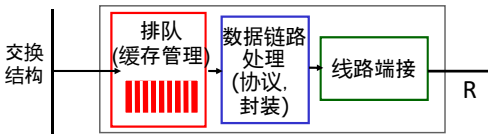
缓存容量多少合适？

- RFC 3439 经验法则：平均缓存等于“典型” RTT（例如250毫秒）乘以链路容量C
  - 例如，C = 10 Gbps 链路：2.5 Gbit 缓存
- 最近的理论和试验研究表明：在有N个流的情况下，缓存等于

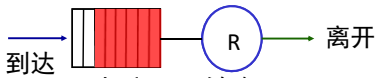
$$\frac{RTT \cdot C}{\sqrt{N}}$$

- 但是过多的缓存会增加延迟(特别是在家庭路由器中)
  - 较长的RTT：实时应用程序的性能较差，TCP响应缓慢
  - TCP协议中基于延迟的拥塞控制：“保持瓶颈链路刚满（忙），但没有更满。。”

缓存管理



抽象：队列



缓存管理：

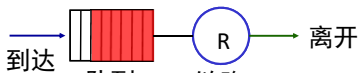
- 丢弃：缓存已满时要添加或丢弃哪些数据包
  - 弃尾(tail drop)：丢弃到达的分组
  - 优先级(priority)：按优先级丢弃/删除
- 标记：标记哪些分组以指示拥塞（ECN）

分组调度：FCFS

分组调度：决定下一个在链路上发送哪个数据包

- 先来先服务优先权排队
- 循环排队
- 加权公平排队

抽象：队列

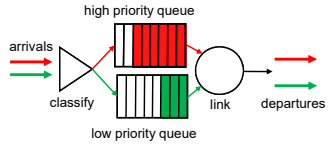


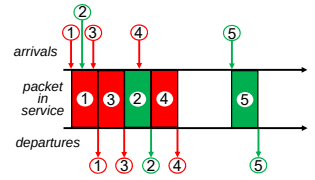
FCFS：先来先服务，分组按到达顺序传输到输出端口

- 也称为：先进先出(First-in-first-out,FIFO)
- 真实的例子？

分组调度：优先权排队

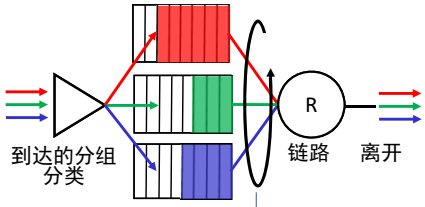
- 到达输出链路的分组被分类，按类别排队
  - 任何头部字段都可以用于分类
- 从具有缓存分组的最高优先权类中传输一个分组
  - 同一优先权类的分组之间的选择通常以FIFO方式完成





分组调度：循环排队
Round Robin Queuing

- 到达输出链路的分组被分类，按类别排队
  - 任何头部字段都可以用于分类
- 循环调度器周期性地重复扫描类队列，依次从每个类（如果可用）发送一个完整的分组

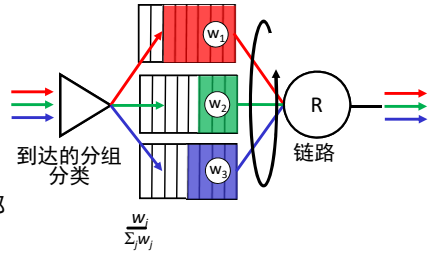


分组调度：加权公平排队
Weighted Fair Queuing

- 通用形式的循环排队
- 每个类 i具有权重w_i，并在每个周期中获得加权的服务部分：

$$\frac{w_i}{\sum_j w_j}$$

- 即最坏的情况下，即使所有的类都有分组在排队，第i类总能获得至少为R*$\frac{w_i}{\sum_j w_j}$ 最小带宽保证

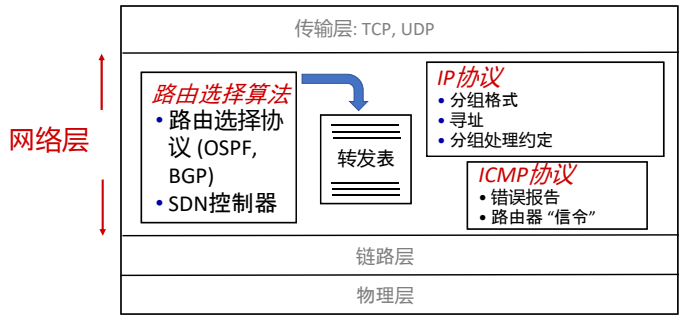


网络层：“数据平面”路线图

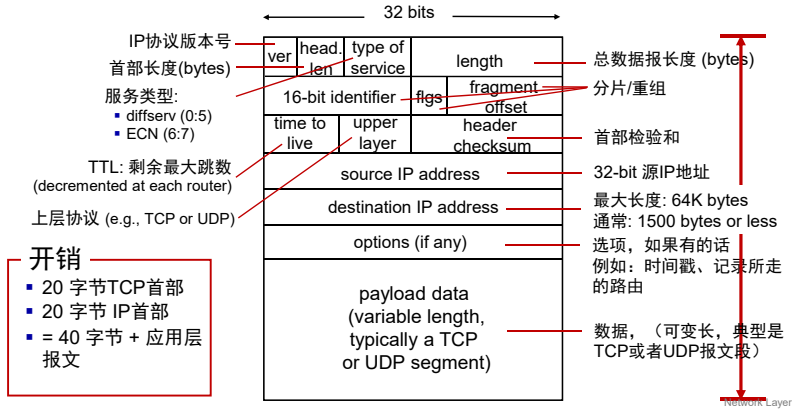
- 网络层：概述
  - 数据平面data plane
  - 控制平面(control plane)
- 路由器内部工作原理
  - 输入端口处理、交换、输出端口处理
  - 缓冲区管理、分组调度
- 网际协议IP(Internet Protocol)
  - 数据报格式
  - 编址
  - 网络地址转换
  - IPv6
- 通用转发和SDN
  - 匹配和动作
  - OpenFlow：匹配加动作
- 中间盒子(Middleboxes)

网络层

主机、路由器网络层功能：



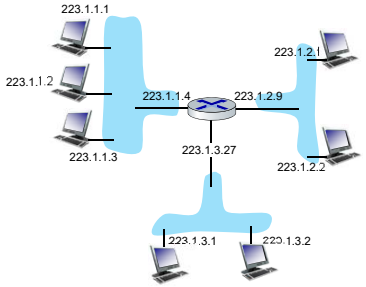
IPv4数据报格式



开销
- 20 字节TCP首部
- 20 字节 IP首部
- = 40 字节 + 应用层报文

IPv4编址

- IP 地址：与每个主机或路由器接口（interface）关联的32位标识符
- 接口：主机/路由器和物理链路之间的边界
  - 路由器通常有多个接口
  - 主机通常有一个或两个接口（例如，有线以太网、无线802. 11）



点分十进制IP地址表示法：

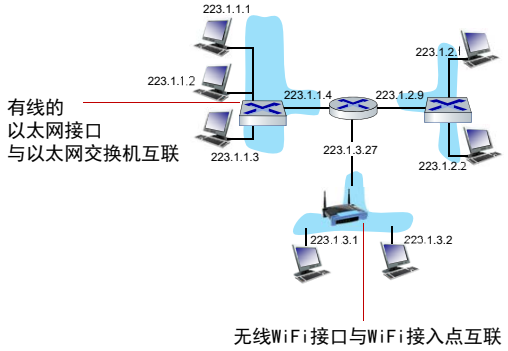
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IPv4编址

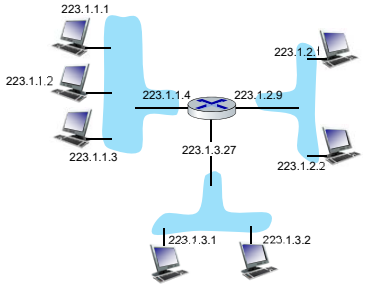
问：接口是如何连接的？

答：我们将在第6、7章学习到



子网

- 什么是子网？
  - 无需通过中间路由器即可物理连接到对方的设备接口
- IP地址具有以下结构：
  - 子网部分：同一子网中的设备具有共同的高位
  - 主机部分：剩余低位

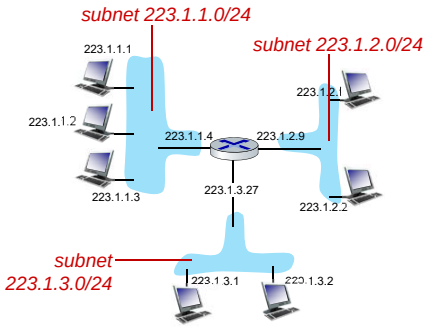


3个子网组成的网络

子网

定义子网的方法：

- 将每个接口与其主机或路由器分离，创建隔离网络的“孤岛”
- 每个隔离的网络称为子网

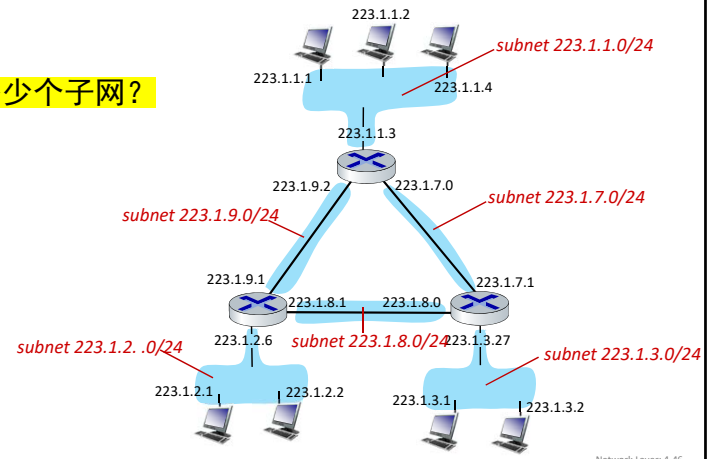


子网掩码: /24
(最左侧的24 比特: IP地址的子网部分)

子网

问：有多少个子网？

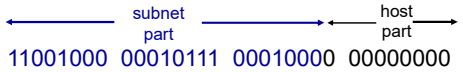
答：6个



IPv4编址：CIDR

无类别域间路由选择（ Classless InterDomain Routing, CIDR）（发音为 “cider”）

- IP地址中的任意长比特数作为网络部分
- 地址格式：a.b.c.d/x，其中x是网络地址的比特数



200.23.16.0/23

IP地址：如何获取？

这实际上是两个问题：

1. 问：主机如何在其网络中获得IP地址（地址的主机部分）？
2. 问：网络如何获取自身的IP地址（地址的网络部分）

主机如何获得IP地址？

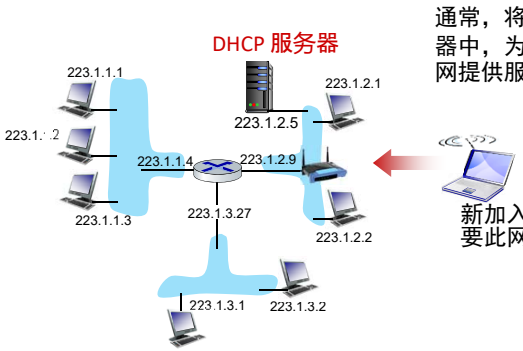
- 由sysadmin硬编码在配置文件中（例如，在UNIX中为/etc/rc.config）
- 动态主机配置协议(Dynamic Host Configuration Protocol, DHCP)：从服务器动态获取地址
  - “即插即用”

DHCP: 动态主机配置协议

目标：当主机“加入”网络时，动态地从网络服务器获取IP地址
- 可以更新(renew)其使用的地址
- 允许重复使用(reuse)地址（仅在连接/打开时保留地址）
- 支持加入/离开网络的移动用户

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

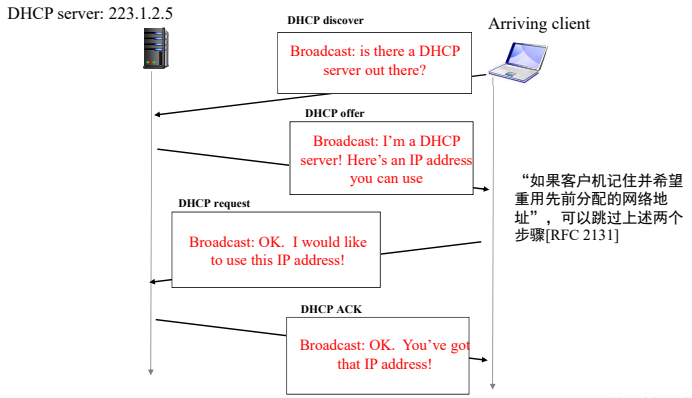
DHCP客户-服务器方案



通常，将DHCP服务器部署在路由器中，为路由器所连接的所有子网提供服务

新加入的DHCP客户端需要此网络中的地址

DHCP协议简要流程

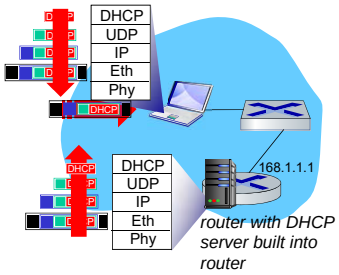


“如果客户机记住并希望重用先前分配的网络地址”，可以跳过上述两个步骤[RFC 2131]

DHCP可以返回的不仅仅是IP地址

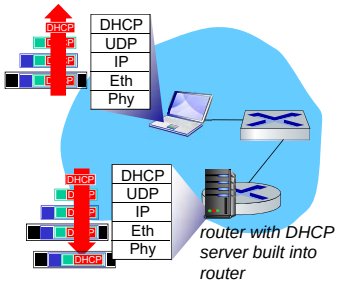
- 客户机的第一跳路由器地址
- DNS服务器的名称和IP地址
- 网络掩码（指示IP地址的网络和主机部分）

DHCP: 例子



- 新加入的笔记本电脑，使用DHCP获取IP地址、第一跳路由器地址、DNS服务器地址。
- DHCP请求消息封装在UDP中，再封装在IP中，再封装在以太网帧中
- 以太网帧局域网上广播（dest:ffffffffffff），该广播包被运行DHCP服务器的路由器接收
- 以太网帧多路分解到IP，再多路分解到UDP，再多路分解到DHCP

DHCP: 例子



- DHCP服务器给出包含客户端IP地址、客户端第一跳路由器IP地址、DNS服务器名称和IP地址的DHCP ACK
- 封装的DHCP服务器报文转发到客户端，在客户端多路分解到DHCP
- 客户机现在知道它的IP地址，DNS服务器的名称和IP地址，第一跳路由器的IP地址

IPv4地址：如何获取？

问：网络如何获取IP地址的子网部分？

答：ISP分配其地址空间的一部分

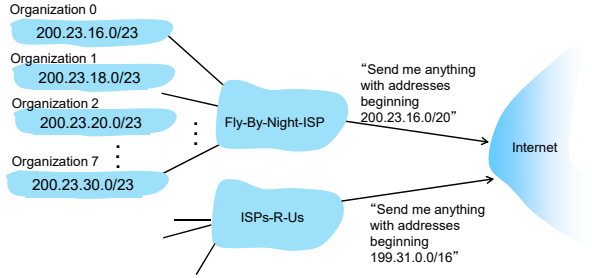
ISP's block 11001000 00010111 00010000 00000000 200.23.16.0/20

然后ISP可以按8个块分配其地址空间：

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

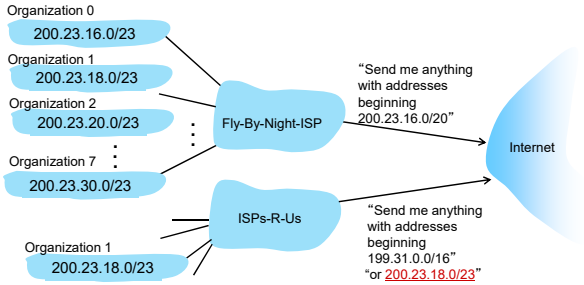
分层寻址：路由聚合

分层寻址可以有效地发布路由信息：



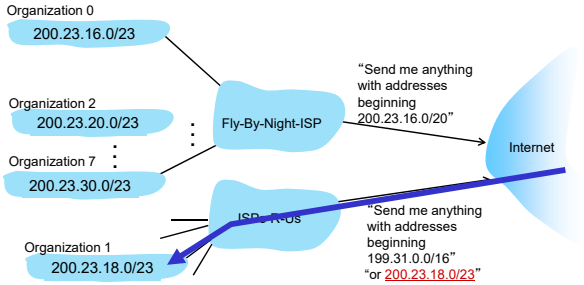
分层寻址：更具体的路由

- 组织1从Fly-By-Night-ISP迁移到ISPs-R-Us
- ISPs-R-Us发布了一条更为具体的到组织1的路由



分层寻址：更具体的路由

- 组织1从Fly-By-Night-ISP迁移到ISPs-R-Us
- ISPs-R-Us发布了一条更为具体的到组织1的路由



IPv4编址：最后...

问：ISP如何获取地址块？

答：ICANN：互联网名称与数字地址分配机构http://www.icann.org/

负责IP地址的空间分配、协议标识符的指派、顶级域名（gTLD、ccTLD）系统的管理、以及根服务器系统的管理

- 全球现有5个区域互联网注册管理机构(Regional Internet Registries, RIRs)，负责该区域的IP地址等的登记注册服务：ARIN,RIPE,APNIC,LACNIC,AfriNIC
- 在RIP之下还有一些注册机构，如国家注册机构NIR, 普通地区级注册机构LIR,负责下级的分配：我国的国家级注册机构是CNNIC

问：是否有足够的32位IP地址？

- ICANN在2011年将最后一部分IPv4地址分配给了区域互联网注册管理机构
- NAT有助于缓解IPv4地址空间耗尽
- IPv6有128位地址空间

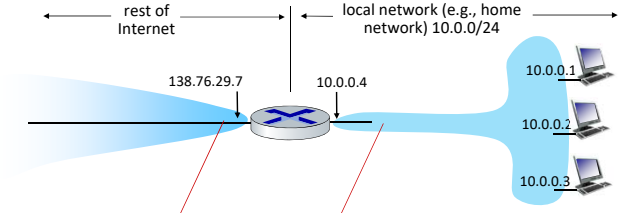
"谁知道我们需要多少地址空间？" Vint Cerf（反思将IPv4地址设置为32位长的决定）

网络层："数据平面"路线图

- 网络层：概述
  - 数据平面data plane
  - 控制平面(control plane)
- 路由器内部工作原理
  - 输入端口处理、交换、输出端口处理
  - 缓冲区管理、分组调度
- 网际协议IP(Internet Protocol)
  - 数据报格式
  - 编址
  - 网络地址转换
  - IPv6
- 通用转发和SDN
  - 匹配和动作
  - OpenFlow：匹配加动作
- 中间盒子(Middleboxes)

NAT: 网络地址转换

NAT(Network Address Translation)：就外部世界而言，本地网络中的所有设备仅共享一个IPv4地址



所有离开本地网络的数据报都有相同的源NAT IP地址：138.76.29.7，但源端口号不同

此网络中带有源或目地的数据报的源、目的地址为10.0.0 / 24（通常）

NAT: 网络地址转换

- 本地网络中的所有设备在“专用”IP地址空间（10/8、172.16/12、192.168/16前缀）中都有32位地址，这些地址只能在本地网络中使用
- 优点:
  - 供应商ISP只需为所有设备提供一个IP地址
  - 可以在不通知外界的情况下更改本地网络中主机的地址
  - 可以更改ISP而无需更改本地网络中设备的地址
  - 安全：本地网络中的设备无法直接被寻址，外界看不到

NAT: 网络地址转换

实现：NAT路由器必须透明：

- 传出数据报：将每个传出数据报的（源IP地址，端口#）替换为（NAT IP地址，新端口#）
  - 远程客户端/服务器将使用（NAT IP地址，新端口#）作为目标地址进行响应
- 记住（在NAT转换表中）每个（源IP地址，端口号）到（NAT IP地址，新端口号）的转换对
- 传入数据报：将每个传入数据报的目标字段中的（NAT IP地址，新端口号）替换为存储在NAT表中的相应的（源IP地址，端口号）

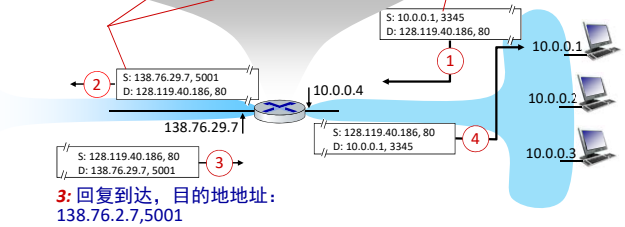
NAT: 网络地址转换

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

NAT 转换表

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

1: 主机10.0.0.1向128.119.40.186, 80发送数据报



3: 回复到达，目的地地址：138.76.2.7,5001

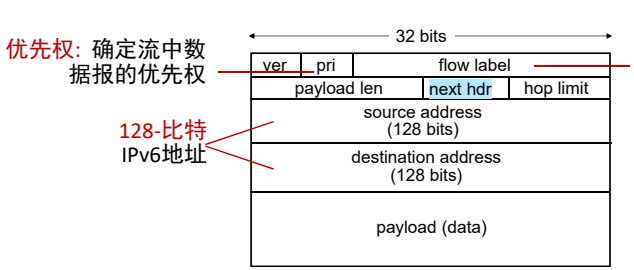
NAT: 网络地址转换

- NAT一直备受争议:
  - 路由器“应该”只能处理最高到第3层
  - 地址“短缺”应通过IPv6解决
  - 违反端到端原则（NAT中网络层设备要进行端口处理）
  - 如何进行NAT穿越：如果客户端要连接到NAT后面的服务器怎么办？
- 但是NAT仍然存在：
  - 广泛用于家庭网、机构网，4G / 5G蜂窝网

IPv6: 动机

- 最初的动机：32位IPv4地址空间将全部被分配完
- 其他动机：
  - 处理/转发速度更快：40字节固定长度的首部
  - 支持对“流”进行不同的网络层处理

IPv6 数据报格式

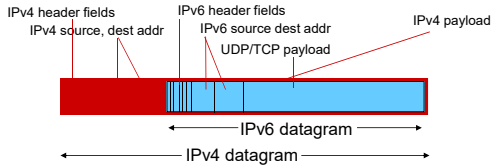


流标签: 用于识别相同“流”中的数据报。
- “流”的概念没有明确定义，例如: 音频和视频传输可能被当做一个流,而普通的邮件传输就不被当做流
- 给属于特殊流的分组加上标签，如一种非默认服务质量或者需要实时服务的流

与IPv4相比缺少什么:
- 没有校验和（以加快路由器的处理速度）
- 没有分片/重组
- 无选项（可能出现在 IPv6 首部中由 “下一个首部” 指出的位置上）

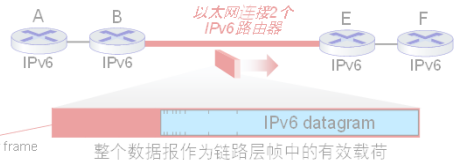
从 IPv4 到 IPv6 的迁移

- 并非所有路由器都可以同时升级
  - 没有“标志日”
  - 混合IPv4和IPv6路由器的网络将如何运行？
- 建隧道： IPv6数据报作为IPv4数据报的载荷、在IPv4路由器之间传输（“分组中的分组”）
- 在其他情况下建隧道方法也被广泛使用（4G / 5G）

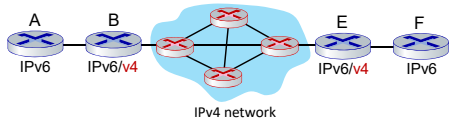


建隧道和封装

以太网连接2个IPv6路由器

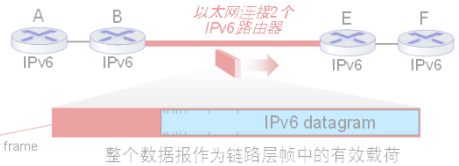


连接2个IPv6路由器的IPv4网络

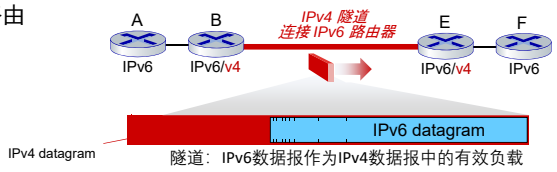


建隧道和封装

以太网连接2个IPv6路由器



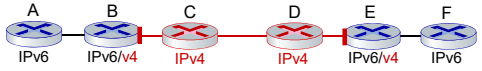
连接2个IPv6路由器的IPv4隧道



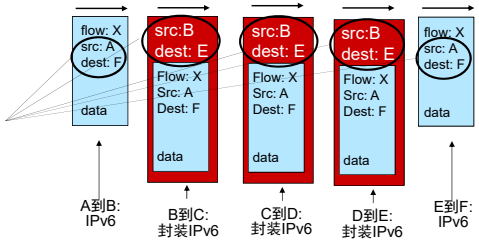
建隧道

逻辑视图：

物理视图：

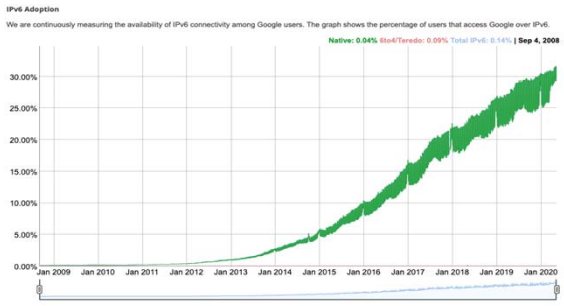


注意源地址和目的地址！



IPv6的部署

- Google[1]: 约30%的客户端通过IPv6访问服务(2020)
- NIST: 1/3的美国政府域名支持IPv6（NIST IPv6 2015）



[1] https://www.google.com/intl/en/ipv6/statistics.html

IPv6部署

- Google[1]: 从2015年8%到2020年约30%的客户端通过IPv6访问服务
- IPv6长时间（很长！）的部署
  - 26年了！（从1996 RFC1883计算的话）
  - 考虑一下过去26年中应用程序的变化：WWW，社交媒体，流媒体，游戏，远程呈现，…
  - 为什么部署经历了这么长的时间？
    改变网络层协议是十分困难的
    如同替换一栋房子的基石，在不拆掉整栋房子、或至少临时重新安置房屋住户的情况下是很难完成替换房子的基石的；新的应用层协议就像给一栋房子重新刷了一层漆，相对容易。

[1] https://www.google.com/intl/en/ipv6/statistics.html

网络层："数据平面"路线图

- 网络层：概述
  - 数据平面data plane
  - 控制平面(control plane)
- 路由器内部工作原理
  - 输入端口处理、交换、输出端口处理
  - 缓冲区管理、分组调度
- 网际协议IP(Internet Protocol)
  - 数据报格式
  - 编址
  - 网络地址转换
  - IPv6

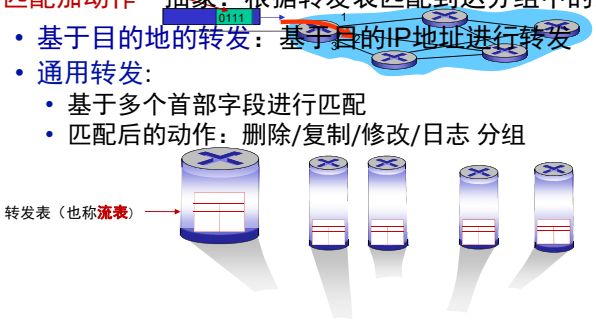
- 通用转发和SDN
  - 匹配和动作
  - OpenFlow：匹配加动作
- 中间盒子(Middleboxes)

通用转发：匹配加动作

回顾：每个路由器的转发

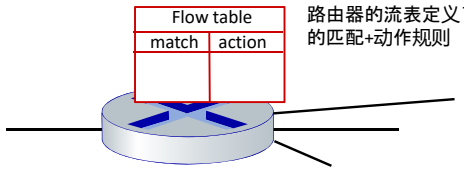
"匹配加动作"抽象：根据转发表匹配到达分组中的比特，执行操作

- 基于目的地的转发：基于目的IP地址进行转发
- 通用转发:
  - 基于多个首部字段进行匹配
  - 匹配后的动作：删除/复制/修改/日志 分组



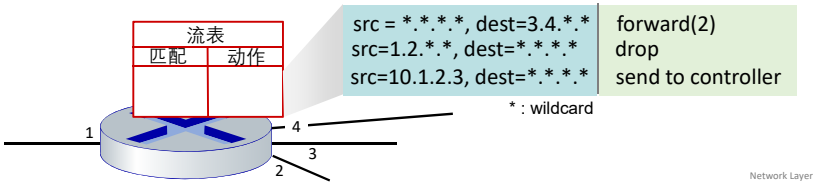
流表抽象

- 流：由头部字段值定义（在链接，网络，传输层字段中）
- 通用转发: 简单的数据包处理规则
  - 匹配：数据包头字段中的值
  - 动作：对于匹配的数据包：丢弃、转发、修改、匹配数据包或将匹配的数据包发送给控制器
  - 优先权：当多个匹配时，消除歧义
  - 计数器：#字节和#数据包

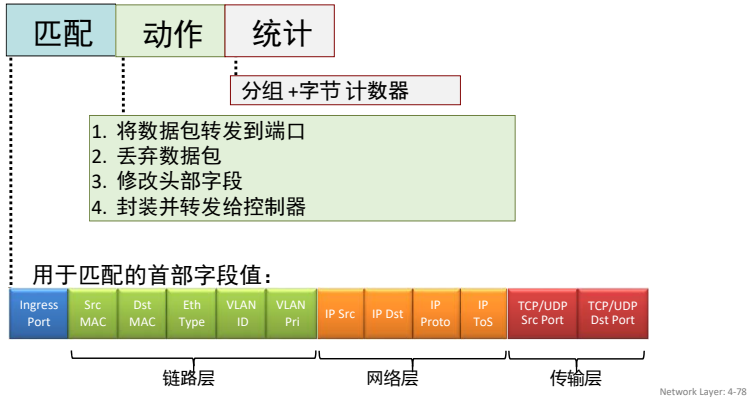


流表抽象

- 流：由首部字段值定义（在链路层、网络层、传输层的字段）
- 通用转发: 简单的数据包处理规则
  - 匹配：分组首部的字段的值
  - 动作：对于匹配的分组：丢弃、转发、修改，或将匹配的分组发送给控制器
  - 优先权：当有多个匹配时，消除歧义
  - 计数器：与表项匹配的分组数量，自从该表项上次更新以来的时间



OpenFlow:流表条目



OpenFlow: 例子

基于目的地的转发：

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

发往IP地址51.6.0.8的IP数据报应转发到路由器输出端口6

防火墙：

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

阻止（不转发）发往TCP端口22（ssh端口号）的所有数据报

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

阻止（不转发）主机128.119.1.1发送的所有数据报

OpenFlow: 例子

第2层基于目的地的转发：

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

具有目标MAC地址为22:A7:23:11:E1:02的第2层帧应转发到输出端口3

OpenFlow 抽象

- 匹配+动作: 抽象统一了不同类型的设备

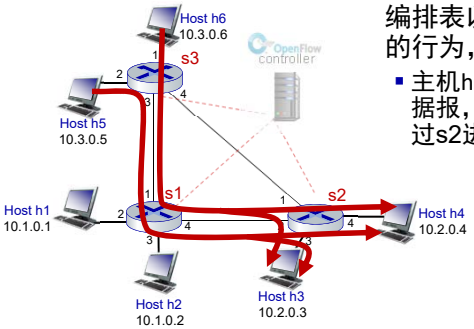
路由器
- 匹配: 目的IP的最长前缀
- 动作: 转发到某一输出端口

交换机
- 匹配: 目的MAC地址
- 动作: 转发或洪泛（flood）

防火墙
- 匹配: IP地址和TCP / UDP端口号
- 动作: 允许或拒绝

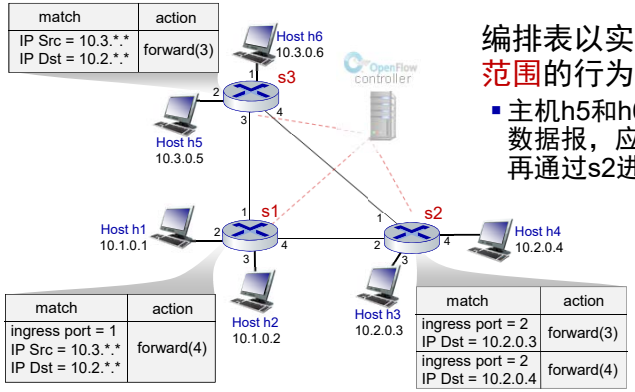
NAT
- 匹配: IP地址和端口号
- 动作: 重写地址和端口号

OpenFlow 例子



编排表以实现整个网络范围的行为，例如：
- 主机h5和h6发往h3或h4的数据报，应首先通过s1，再通过s2进行

OpenFlow 例子



编排表以实现整个网络范围的行为，例如：
- 主机h5和h6发往h3或h4的数据报，应首先通过s1，再通过s2进行

通用转发: 总结

- “匹配+动作”抽象： 匹配到达的分组的任何层的首部的比特，并采取动作
  - 匹配多个字段 (链路层，网络层，传输层)
  - 本地动作: 丢弃，转发，修改或将匹配的分组发送给控制器
  - 网络范围的“编程”行为
- “网络可编程性”的简单形式
  - 可编程的，对每个分组进行“处理”
  - 历史根源：主动网络
  - 今天：更通用的编程：
    P4（请参阅P4.org）。

网络层：“数据平面”路线图

- 网络层：概述
- 路由器内部工作原理
- 网际协议IP(Internet Protocol)
- 通用转发和SDN
- 中间盒子(Middleboxes)
  - 中间盒子的功能
  - 互联网的演化

中间盒子

中间盒子(Middlebox) (RFC 3234)

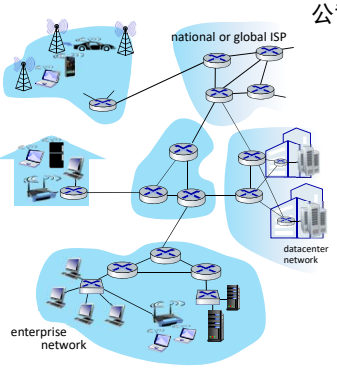
“any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host”

在源主机和目标主机之间的数据路径上、执行IP路由器正常和标准功能之外的功能的任何中间盒子

中间盒子无处不在

NAT:
家庭网、蜂窝网、机构网

特定应用:
服务提供商、机构的应用、CDN应用



防火墙, IDS:
公司、机构、服务提供商、ISP

负载均衡器:
企业、服务提供商、数据中心、移动网络

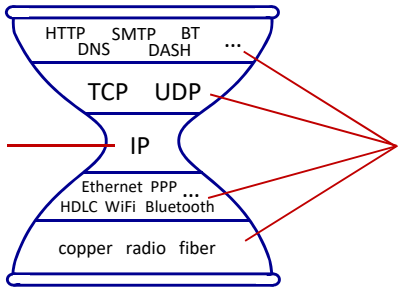
缓存:
服务提供商、移动设备、CDN

中间盒子

- 最初: 专有（封闭）硬件解决方案
- 转向实现开放API的“白盒”硬件
  - 摆脱专有硬件解决方案
  - 通过匹配+动作实现的可编程本地操作
  - 迈向软件创新/差异化
- SDN:（逻辑上）通常在私有/公共云中进行集中控制和配置管理
- 网络功能虚拟化（NFV）: 白盒网络、计算、存储上的可编程服务

IP 沙漏

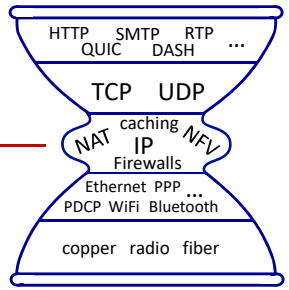
互联网的“细腰”：
- 一个网络层协议：IP
- 每（数十亿）个互联网连接设备必须实现



物理、链路、传输和应用层中的许多协议

IP沙漏

中间盒子，在网络内部运行着



问题是: *是否互联网喜欢这些中间盒子的这种设计？*

互联网的架构是怎样设计的？

RFC 1958

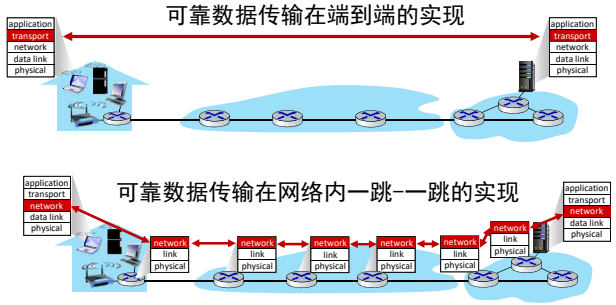
“Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network.”

“互联网社区的许多成员会争辩说，没有架构设计，只有传统而已，架构是如何设计的在最初的25年里没有被记录下来（或者至少没有被IAB记录下来）。然而，一般来说，互联网社区认为目标是连通性，工具就是IP协议，智能应该由端到端完成，而不是隐藏在网络中。”

三个基本理念：
- 简单的连接
- IP协议：窄腰
- 智能、复杂性在网络边缘实现

端到端的论点

- 一些网络功能（如可靠数据传输、拥塞控制）可以在网络内实现，也可以在网络边缘实现

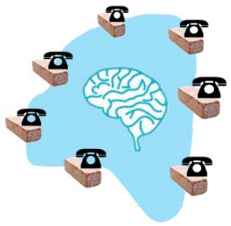


端到端的论点

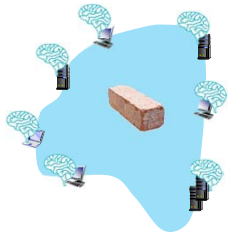
- 一些网络功能（如可靠数据传输、拥塞控制）可以在网络内实现，也可以在网络边缘实现？- 到底互联网如何呢？

> "The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)
>
> We call this line of reasoning against low-level function implementation the "end-to-end argument."
>
> Saltzer, Reed, Clark 1981

智能在哪里?



20th 世纪电话网：
- 智能/计算在网络交换机中实现



Internet (2005年前)
- 智能，计算在网络边缘实现

Internet (2005年后)
- 可编程网络设备
- 智能，计算，大量的应用层体系架构在网络边缘实现

第4章：总结

- 网络层：概述
  - 数据平面（ data plane ）
  - 控制平面（ control plane ）
- 路由器内部工作原理
  - 输入端口处理、交换、输出端口处理
  - 缓冲区管理、分组调度
- 网际协议IP(Internet Protocol)
  - 数据报格式
  - 编址
  - 网络地址转换
  - IPv6

- 通用转发和SDN
  - 匹配和动作
  - OpenFlow：匹配加动作
- 中间盒子(Middleboxes)

问: 如何计算得到路由器中的转发表（基于目的地的转发） 或流表（通用转发）？

答: 通过网络层的控制平面（下一章）