Chapter 1: Introduction

Outline

1. what is the Internet?
2. network edge end systems, access networks, links
3. network core packet switching, circuit switching, network structure
4. delay, loss, throughput in networks
5. protocol layers, service models
6. networks under attack: security
7. history

What is the Internet?

互联网贡献组织

IETF (Internet Engineering Task Force): 互联网工程任务组，成立于 1985 年，主要定义互联网标准。网址: http://ietf.org (http://ietf.org)

我们平时提到的 RFC 文档都是他们的产物，例如常见的网络协议: IP、TCP、UDP、FTP、HTTP1.1 等。

RFC 文档也称请求注解文档（Requests for Comments），这是用于发布 Internet 标准和 Internet 其他正式出版物的一种网络文件或工作报告。RFC 文档初创于 1969 年，RFC 出版物由 RFC 编辑（RFC Editor）直接负责，并接受 IAB 的一般性指导。现在已经有 3000 多个 RFC 系列文件，并且这个数目还在不断增加，内容和 Internet (开始叫做为 ARPANET) 相关。草案讨论了计算机通讯的方方面面，重点在网络协议，过程，程序，以及一些会议注解，意见，风格方面的概念。文档下载: http://www.ietf.org/rfc.html (http://www.ietf.org/rfc.html)

网络七层协议

网络七层协议由上到下分为: 应用层、表示层、会话层、传输层、网络层、数据链路层、物理层。

- 物理层: 解决两个硬件之间怎么通信的问题，常见的物理媒介有光纤、电缆、中继器等。
- 数据链路层: 在计算机网络中由于各种干扰的存在，物理链路是不可靠的。功能上，都是通过各种控制协议，将有差错的物理信道变为无差错的、能可靠传输数据帧的数据链路。
- 网络层: 该层的主要任务就是: 通过路由选择算法，为报文（该层的数据单位，由上一层数据打包而来）通过通信子网选择最适当的路径。这一层定义的是 IP 地址，通过 IP 地址寻址，所以产生了 IP 协议。IP（Internet Protocol）: 互联网协议，是无连接的，不保证数据包的到达顺序和完整性，只负责将数据包发送给指定的电脑。
- 传输层: 用于监控数据传输服务的质量，保证报文的正确传输。
    - TCP（Transmission Control Protocol）: 传输控制协议，是面向连接的，保证数据包的到达顺序和完整性，通过握手、重传和排序等机制实现可靠的通信。
    - UDP（User Datagram Protocol）: 用户数据报协议，是无连接的，不保证数据包的到达顺序和完整性，只负责将数据包发送给指定的程序。UDP 的头部信息很短，额外开销小，适合实时性要求高的应用。

- 应用层：是计算机用户，以及各种应用程序和网络之间的接口，其功能是直接向用户提供服务，完成用户希望在网络上完成的各种工作。

互联网与万维网

- 互联网是一个全球性的网络网络，它由许多不同的设备和协议连接在一起，如电脑、手机、路由器、TCP/IP 等。
- 万维网是一个由许多互相链接的超文本组成的系统，它通过互联网访问，使用 HTTP 协议和 HTML 语言。
- 互联网是基础设施，而万维网是基础设施之上的服务。互联网还提供了其他服务，如电子邮件、即时通讯、文件传输等。

Network Protocol

网络协议（Network Protocol）是使设备能够通过网络相互通信的规则，包含三个主要类型的网络协议：communication protocols, management protocols and security protocols。

- Communication protocols 包括基本数据通信工具，如 TCP/IP 和 HTTP。
- Management protocols 维护和管理网络工具，如 ICMP 和 SNMP。
- Security protocols 包括 HTTPS、SFTP 和 SSL2。

Network Edge

设备

Network edge 是充当组织或服务提供商网络核心入口点的设备。它包括路由器、交换机、广域网 (WAN)、防火墙和集成接入设备 (IAD)。

Router 路由器

路由器在两个不同的网络之间传输数据包。此流量包括网站内容以及视频聊天、电子邮件和互联网协议语音 (VoIP) 传输等通信。路由器引导互联网上的流量，将其从一个点发送到另一个点，允许不同的边缘设备相互通信。

Switch 交换机

网络交换机通过数据包交换连接计算机网络中的设备，它接收数据，然后将其转发到目标设备。交换机允许边缘设备在不使用核心设备的情况下进行交互和共享资源。

Wide-area Network 广域网

WAN 由相互连接的局域网 (LAN) 组成。通过这种方式，WAN 边缘连接了 LAN 的边缘。例如，一个组织可以使用 WAN 或软件定义的 WAN (SD-WAN) 连接三个办公室，每个办公室都有自己的 LAN。

Firewall 防火墙

防火墙根据预定义的规则控制允许进入和退出网络基础设施的数据。防火墙检查数据包，寻找任何引起怀疑的东西，然后丢弃任何包含潜在威胁的数据包。防火墙是网络边缘的主要防线，可防止威胁进出。

Integrated Access Device 综合接入设备

IAD 转换不同类型的数据输入并将它们呈现为通用格式。例如，IAD 用于将模拟和数字电话信号转换为一个通用数字信号。 IAD 有助于简化通信并在边缘实现更高效的传输。

DSL

DSL（Digital Subscriber Line）意为数字用户线路，是指以电话线为传输介质的传输技术组合。

Host: Sends Packets of Data

包延时主要指将数据打包成一个个 Packages 需要的时间

$$packet\ transmission\ delay\ = time\ needed\ to\ transmit\ L(bit)\ packet\ into\ link = \frac{L (bits)}{R (bits/sec)}$$

网络延时是指一个数据包从源端发向目的端，然后再立即从目的端返回源端的时间。

网络连接方式

- Coaxial cable：由两个同轴铜导体构成。主要用于多频道有线电视和 HFC。
- Fiber optic cable：光纤电缆，特点是高速和低错误。
- Radio：无线电，电磁波谱信号进行。特点是没有“物理连接”，大多有双向能力。有以下几种应用：
    - Terrestrial microwave：地面微波
    - LAN：本地网络，如 WiFi
    - Wide-area：如移动网络 4G、5G 等
    - Satellite：卫星网络，如马斯克的星链

The Network Core

Packet Switching: 储存转发

- 发包/收包延迟：花费 L/R 秒，使用 R bps 的速率发收 L-bit 包
- store and forward：存储转发，整个数据包到达路由器之前必须传播/转发数据
- end-end delay：端对端延迟，等于 2L/R（假设传播延迟为 0）

延迟、丢包和抖动

> 延时、丢包、抖动是什么，该怎么办? - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/21968527)
>
> 延时、丢包、抖动是互联网这个信息公路网无法避免的三个特点。假设我们现在有一百辆车从北京鸟巢开往上海东方明珠，并且每隔一分钟出发一辆。

> 丢包：指的是有的车无法在有效时间内无法达到终点，甚至可能永远也到不了终点。有的车可能永远堵在北京的三环上了，有的车可能中途出了车祸。假如我们的一百辆车里有五辆车因为各种原因没能按时到达上海，我们这次车队传输的“丢包率”就是 5%。是的，互联网传输也一样，它并不是百分百可靠的，总有数据无法按时传输到目的地。
>
> 抖动：指的是车子到达的顺序、间隔和出发时的差异。虽然我们的一百辆车在北京是等间隔的一分钟一辆出发的，但是它们到达上海时却并不是按顺序一分钟一辆到达的，甚至可能有晚出发的车比早出发的车先到的情况。互联网传输也一样，如果简单地按照收到的音视频数据顺序直接播放出来，就会出现失真的现象。

两个关键的网络核心功能

- routing：确定源目的地路由数据包

- forwarding：将数据包从路由器的输入移动到相应的路由器输出

电路交换（**Circuit switching**）

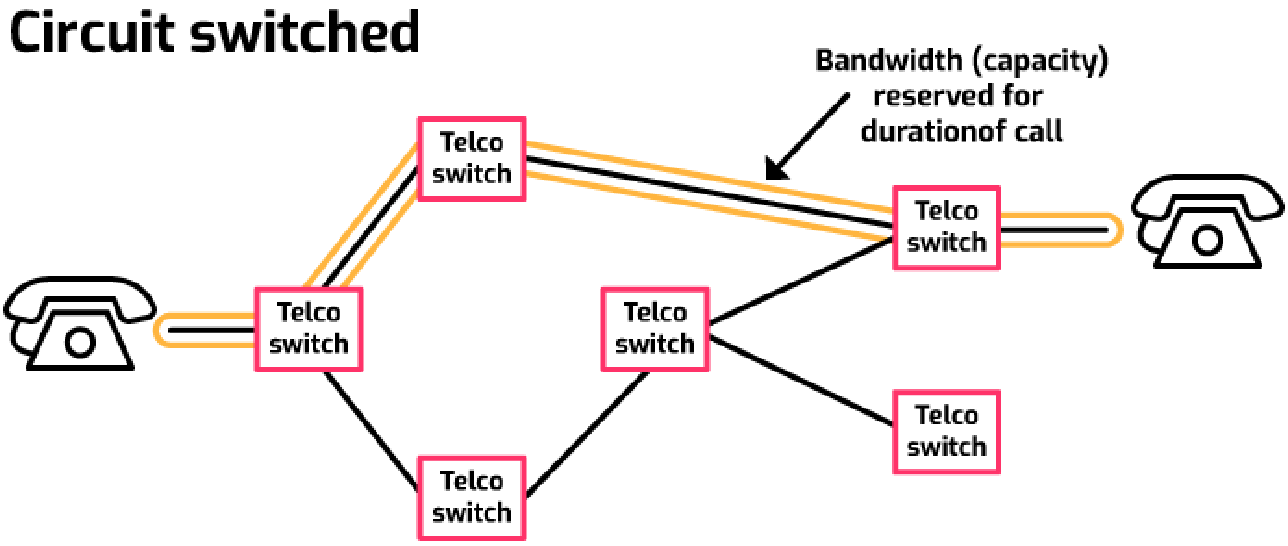
FDM (Frequency Division Multiplexing) 和 TDM (Time Division Multiplexing) 是两种复用技术，用于在一个信道上传输多路信号。

- FDM 是将信道划分为多个不重叠的频率带，每个频带携带一路信号；TDM 是将信道划分为多个固定长度的时间片，每个时间片携带一路信号。
- FDM 只适用于模拟信号，TDM 既适用于模拟信号也适用于数字信号。
- FDM 需要设置保护频带以防止相邻频带之间的干扰，这会导致频率资源的浪费；TDM 不需要设置保护时间片，因此更有效地利用了时间资源。
- FDM 具有较低的延迟，因为每路信号都可以同时传输；TDM 具有较高的延迟，因为每路信号都要等待自己的时间片到来。
- FDM 的电路或芯片比较复杂，需要进行调制和解调；TDM 的电路或芯片比较简单，只需要进行开关和同步。

Packet switching 与 circuit switching

- Packet switching 是将数据分割成小块的分组 (packet)，每个分组都带有目的地址和序号，然后通过共享的网络独立地发送到目的地，最后在接收端重新组合成完整的数据。

  > Structure of the internet — Isaac Computer Science (https://isaaccomputerscience.org/concepts/net_internet_structure?examBoard=all&stage=all)
  > Traffic on the internet is transported as **packets**. An internet packet is made up of the data that is being transported, which is called the **payload**, and a **header**.
  > Internet packets carry all sorts of payloads. For example, the data might be part of a web page, or an email, or a streamed audio track. The type of data is identified by the *protocol* field within the header. Internet packets have a maximum size to prevent anything 'hogging' the bandwidth. The packets are moved around the internet using a method called **packet switching**. The use of relatively small data packets allows a data path to be shared. Different packets from the same 'conversation' may be sent over different routes. The internet has an **end-to-end** principle（原则） where the end points (source and destination) are responsible for checking that everything that has been sent is received, as appropriate. This type of communication between sender and receiver is known as connectionless (rather than dedicated). Most traffic over the internet uses packet switching and the internet is basically a **connectionless network**.

> Structure of the internet — Isaac Computer Science (https://isaaccomputerscience.org/concepts/net_internet_structure?examBoard=all&stage=all)
> Before the internet came into being, the largest global network was the telephone network. In some ways it was similar to the internet in that it was a collection of interconnected（互联的） networks. These interconnections were telephone exchanges or telephone switches.
> In this telephone network, if someone wished to make a call, a signal was sent across the connection to request the line. Each telephone switch would then select the appropriate route and reserve（预先，预定） capacity（容量） on the line and send the request on.
>
> 
>

TIP

简单的说，概念上两者的区别：

- Packet switch：按需分配
- Circuit switch：短时间包场

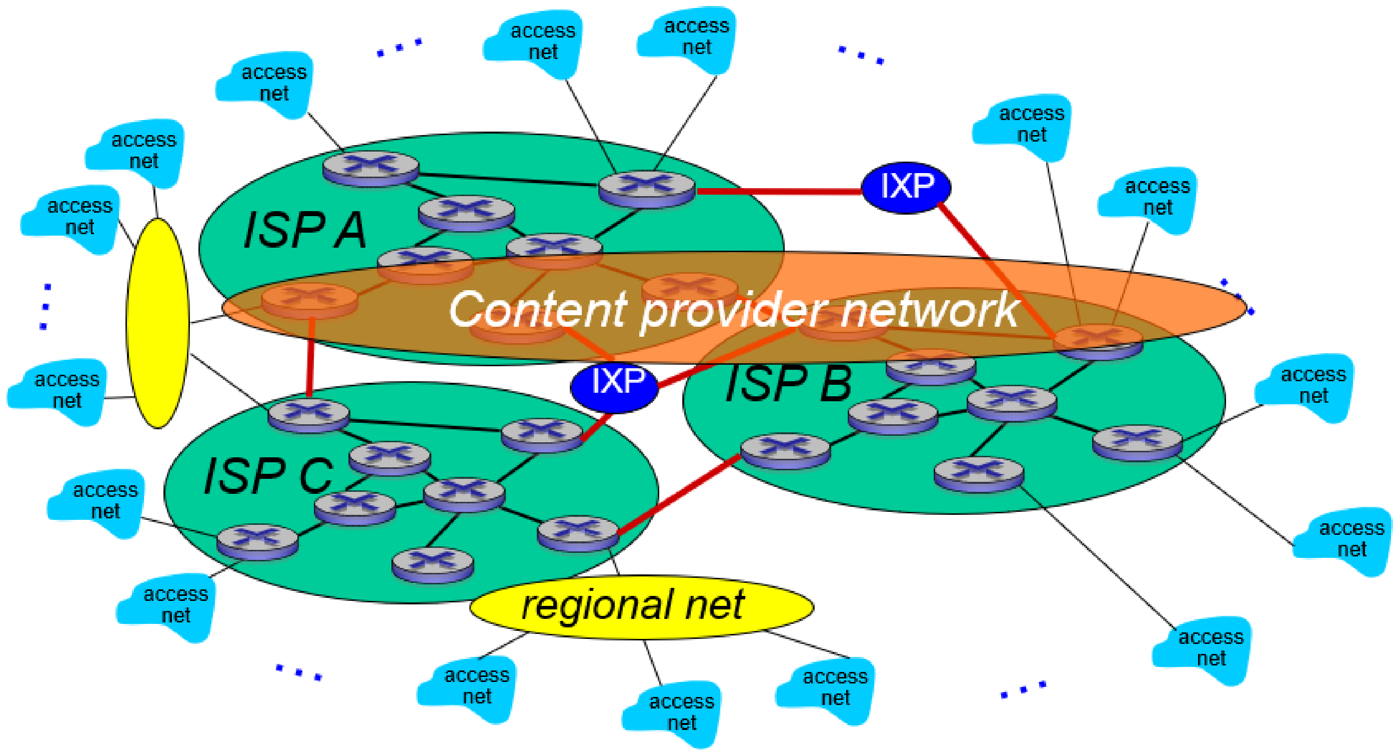
优缺点上：

- Packet switching 更有效地利用了网络资源，因为它可以动态地根据网络状况选择最佳路径，而不需要占用固定的带宽；circuit switching 则会造成带宽的浪费，因为它需要为每次通信保留一定量的带宽，即使没有数据传输也不能被其他用户使用。
- Packet switching 更适合于传输可变长度和突发性的数据，如互联网上的文件、邮件、视频等；circuit switching 更适合于传输固定长度和连续性的数据，如电话、传真等。
- Packet switching 由于分组可能经过不同的路径和顺序到达目的地，因此会导致较高的延迟和抖动；circuit switching 由于数据沿着固定的路径和顺序传输，因此会保证较低的延迟和抖动。
- Packet switching 由于分组可能在网络中丢失或损坏，因此需要更复杂的错误控制机制；circuit switching 由于数据在物理连接上传输，因此不需要错误控制机制。

互联网结构（Internet structure）: Network of networks

- 借助 ISP（Internet Service Providers，互联网服务提供商）通过访问端系统连接到互联网

- Internet exchange point□IXP□□IXP □□□□□□□□□□□ ISP □ CDN □□□□□□□□□ □□□□□□□□□□□□□□□□□
- □□□□□□□□□□□ regional net□
- Content provider network (e.g., Google, Microsoft, Akamai) may run their own network, to bring services, content close to end users.
- Network of networks is very complex.



Delay, loss, throughput in networks

□□□□□□□□□□□□□□□□□

□□ Delay

Transmission delay

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

> Delays in Computer Network - GeeksforGeeks (https://www.geeksforgeeks.org/delays-in-computer-network/)
> The time taken to transmit a packet from the host to the transmission medium is called Transmission delay.

> Let B bps is the bandwidth and L bit is the size of the data then transmission delay is,
> $$ Tt = L/B $$
> This delay depends upon the following factors:
>
> - If there are multiple active sessions, the delay will become significant(相当重要的).
> - Increasing bandwidth(带宽) decreases transmission delay.
> - MAC protocol largely influences the delay if the link is shared among multiple devices.
> - Sending and receiving a packet involves a context switch in the operating system, which takes a finite time.

Propagation delay

数据包传输到传输介质后，必须经过介质才能到达目的地。因此，数据包的最后一位到达目的地所花费的时间称为传播延迟。

> Delays in Computer Network - GeeksforGeeks (https://www.geeksforgeeks.org/delays-in-computer-network/)
> After the packet is transmitted to the transmission medium, it has to go through the medium to reach the destination. Hence the time taken by the last bit of the packet to reach the destination is called propagation delay.
>
> 
>
>
> Factors affecting propagation delay:
>
> 1. **Distance**: It takes more time to reach the destination if the distance of the medium is longer.
> 2. **Velocity**: If the velocity(speed) of the signal is higher, the packet will be received faster.
>
> $$ Tp = Distance / Velocity $$
> **Note:**
> $$ Velocity =3 \cdot 10^8 m/s \ (for air) \ Velocity= 2.1 \cdot 10^8 m/s \ (for optical fibre) $$

Queueing delay

让数据包被目的地接收，数据包不会立即被目的地处理。它必须在称为缓冲区的队列中等待。所以它在被处理之前在队列中等待的时间称为排队延迟。

> Let the packet is received by the destination, the packet will not be processed by the destination immediately. It has to wait in a queue in something called a buffer. So the amount of time it waits in queue before being processed is called queueing delay.
> In general, we can’t calculate queueing delay because we don’t have any formula for that.
> This delay depends upon the following factors:
>
> - If the size of the queue is large, the queuing delay will be huge. If the queue is empty there will be less or no delay.
> - If more packets are arriving in a short or no time interval, queuing delay will be large.
> - The less the number of servers/links, the greater is the queuing delay.

Processing delay

现在数据包将被用于处理，称为处理延迟。处理器处理数据包所花费的时间是中间路由器决定将数据包转发到何处、更新 TTL、执行报头校验和计算所需的时间。

> Delays in Computer Network - GeeksforGeeks (https://www.geeksforgeeks.org/delays-in-computer-network/)
> Now the packet will be taken for the processing which is called processing delay.
> Time is taken to process the data packet by the processor that is the time required by intermediate routers to decide where to forward the packet, update TTL, perform header checksum(校验) calculations.
> It also doesn’t have any formula since it depends upon the speed of the processor and the speed of the processor varies from computer to computer.
> This delay depends upon the following factors:
>
> - It depends on the speed of the processor.
>
> $$ Ttotal = Tt + Tp + Tq + Tpro $$

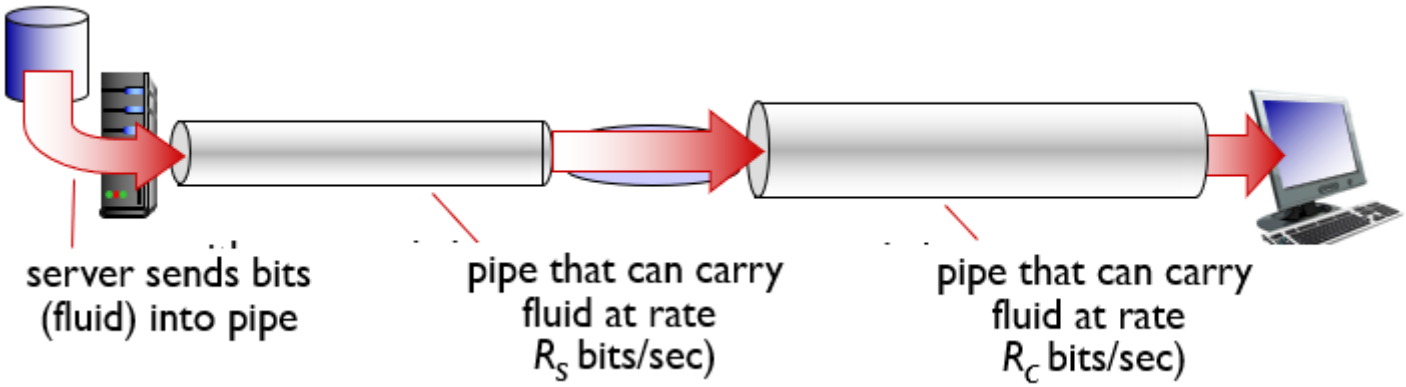
Packet loss 丢包

- queue (aka(别称) buffer) preceding(提前的) link in buffer has finite capacity(容量)
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all (压根没送)

Throughput 吞吐量

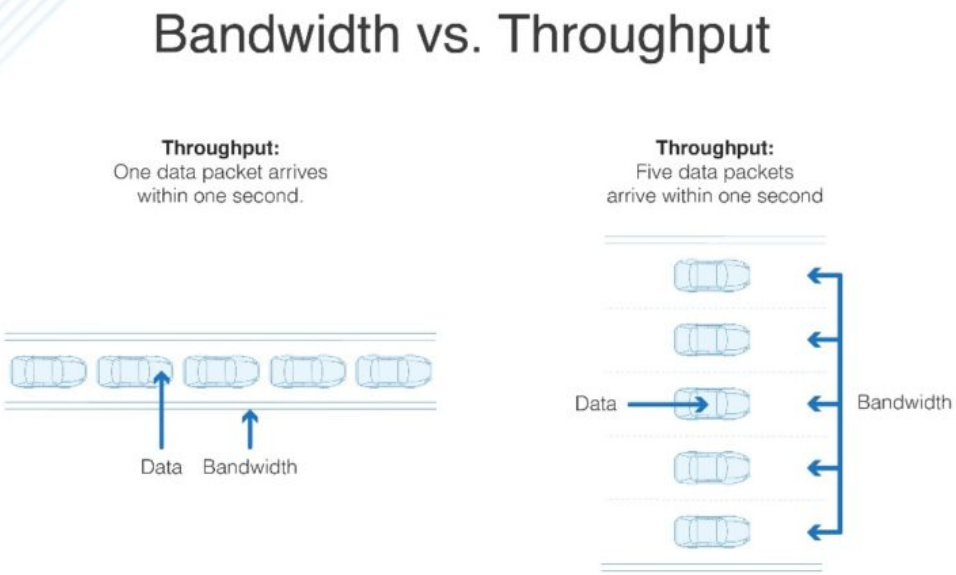
Rate (bits/time unit) at which bits transferred between sender/receiver

- instantaneous(□□): rate at given point in time
- average(□□): rate over longer period of time



Link on end-end path that constrains(□□) end-end throughput.

> Bandwidth and Throughput in Networking: Guide and Tools - DNSstuff (https://www.dnsstuff.com/network-throughput-bandwidth)
> So, how do we define throughput? Again, network throughput refers to how much data can be transferred from source to destination within a given timeframe. **Throughput measures how many packets arrive at their destinations successfully.** For the most part, throughput capacity **is measured in bits per second,** but it can also be measured in data per second.
> **Packet loss, latency(□□), and jitter(□□) are all related to slow throughput speed**. Latency is the amount of time it takes for a packet to make it from source to destination, and jitter refers to the difference in packet delay. Minimizing all these factors is critical to increasing throughput speed and data performance.



> through a small tube, it will go very slowly.

Protocol layers, service models

Layers: each layer implements(实现) a service, relies on services provided by layer below

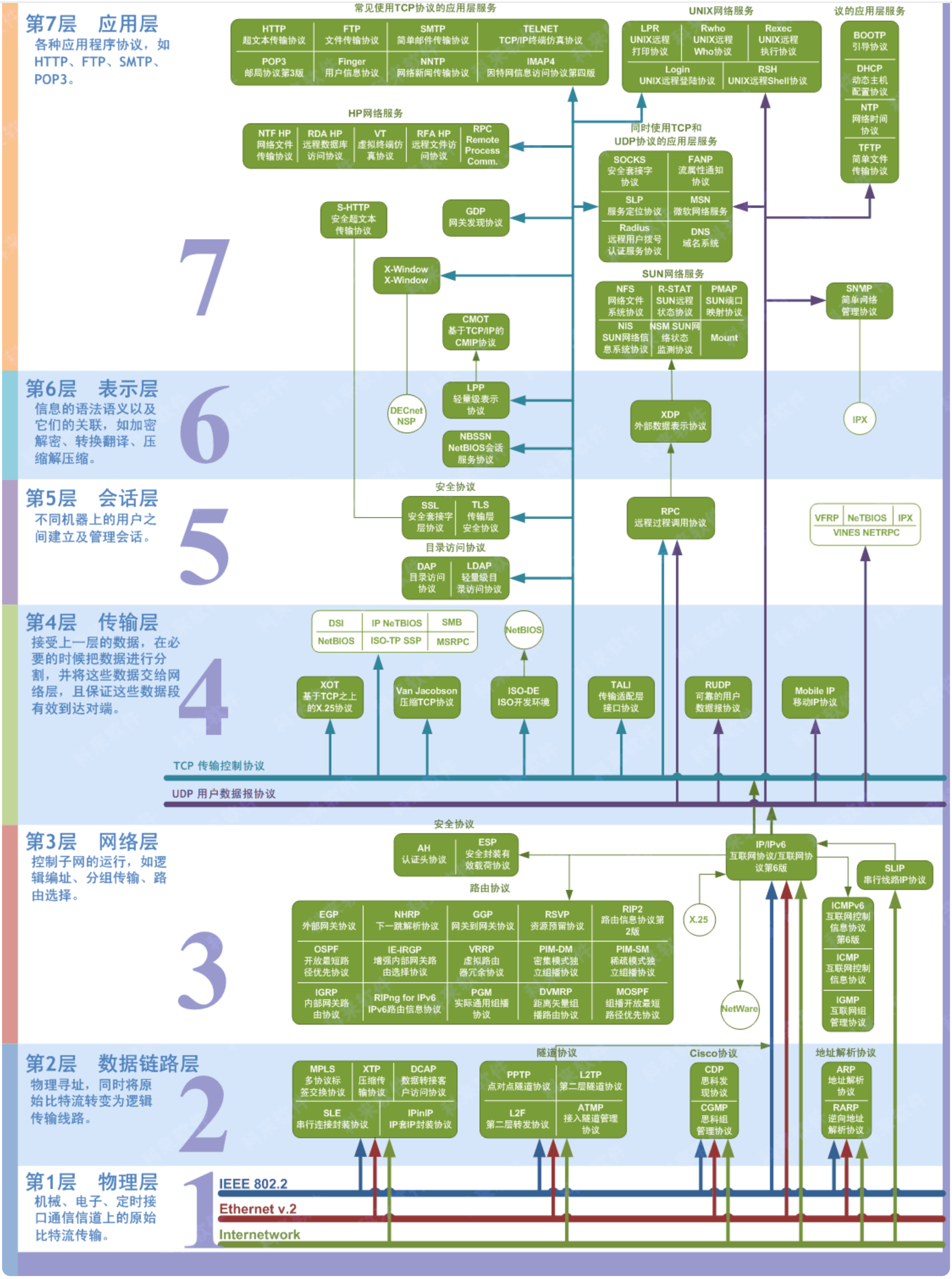
OSI

OSI是Open System Interconnect的缩写，意为开放式系统互联。

7层是指OSI七层协议模型，主要是：应用层（Application）、表示层（Presentation）、会话层（Session）、传输层（Transport）、网络层（Network）、数据链路层（Data Link）、物理层（Physical）。

最后3层也被统称为应用程序层。

> OSI七层模型详解 小鹏_加油的博客 (https://blog.csdn.net/yaopeng_2005/article/details/7064869)



物理层

在 OSI 参考模型中，物理层（Physical Layer）是参考模型的最低层，也是 OSI 模型的第一层。

物理层的主要功能是：利用传输介质为数据链路层提供物理连接，实现比特流的透明传输。

数据链路层

数据链路层（Data Link Layer）是 OSI 模型的第二层，负责建立和管理节点间的链路。该层的主要功能是：通过各种控制协议，将有差错的物理信道变为无差错的、能可靠传输数据帧的数据链路。

在计算机网络中由于各种干扰的存在，物理链路是不可靠的。因此，这一层的主要功能是在物理层提供的比特流的基础上，通过差错控制、流量控制方法，使有差错的物理线路变为无差错的数据链路，即提供可靠的通过物理介质传输数据的方法。

该层通常又被分为介质访问控制（MAC）和逻辑链路控制（LLC）两个子层：

- MAC 子层的主要任务是解决共享型网络中多用户对信道竞争的问题，完成网络介质的访问控制；
- LLC 子层的主要任务是建立和维护网络连接，执行差错校验、流量控制和链路控制。

数据链路层的具体工作是接收来自物理层的位流形式的数据，并封装成帧，传送到上一层；同样，也将来自上层的数据帧，拆装为位流形式的数据转发到物理层；并且，还负责处理接收端发回的确认帧的信息，以便提供可靠的数据传输。

网络层

网络层（Network Layer）是 OSI 模型的第三层，它是 OSI 参考模型中最复杂的一层，也是通信子网的最高一层。它在下两层的基础上向资源子网提供服务。

其主要任务是：通过路由选择算法，为报文或分组通过通信子网选择最适当的路径。该层控制数据链路层与传输层之间的信息转发，建立、维持和终止网络的连接。具体地说，数据链路层的数据在这一层被转换为数据包，然后通过路径选择、分段组合、顺序、进/出路由等控制，将信息从一个网络设备传送到另一个网络设备。

一般地，数据链路层是解决同一网络内节点之间的通信，而网络层主要解决不同子网间的通信。例如在广域网之间通信时，必然会遇到路由（即两节点间可能有多条路径）选择问题。

在实现网络层功能时，需要解决的主要问题如下：

- 寻址：数据链路层中使用的物理地址（如 MAC 地址）仅解决网络内部的寻址问题。在不同子网之间通信时，为了识别和找到网络中的设备，每一子网中的设备都会被分配一个唯一的地址。由于各子网使用的物理技术可能不同，因此这个地址应当是逻辑地址（如 IP 地址）。
- 交换：规定不同的信息交换方式。常见的交换技术有：线路交换技术和存储转发技术，后者又包括报文交换技术和分组交换技术。
- 路由算法：当源节点和目的节点之间存在多条路径时，本层可以根据路由算法，通过网络为数据分组选择最佳路径，并将信息从最合适的路径由发送端传送到接收端。
- 连接服务：与数据链路层流量控制不同的是，前者控制的是网络相邻节点间的流量，后者控制的是从源节点到目的节点间的流量。其目的在于防止阻塞，并进行差错检测。

传输层

OSI 下 3 层的主要任务是数据通信，上 3 层的任务是数据处理。而传输层（Transport Layer）是 OSI 模型的第 4 层。因此该层是通信子网和资源子网的接口和桥梁，起到承上启下的作用。

该层的主要任务是：向用户提供可靠的端到端的差错和流量控制，保证报文的正确传输。传输层的作用是向高层屏蔽下层数据通信的细节，即向用户透明地传送报文。该层常见的协议：TCP/IP 中的 TCP 协议、Novell 网络中的 SPX 协议和微软的 NetBIOS/NetBEUI 协议。

传输层提供会话层和网络层之间的传输服务，这种服务从会话层获得数据，并在必要时，对数据进行分割。然后，传输层将数据传递到网络层，并确保数据能正确无误地传送到网络层。因此，传输层负责提供两节点之间数据的可靠传送，当两节点的联系确定之后，传输层则负责监督工作。

综上，传输层的主要功能如下：

- 传输连接管理：提供建立、维护和拆除传输连接的功能。传输层在网络层的基础上为高层提供“面向连接”和“面向无接连”的两种服务。
- 处理传输差错：提供可靠的“面向连接”和不太可靠的“面向无连接”的数据传输服务、差错控制和流量控制。在提供“面向连接”服务时，通过这一层传输的数据将由目标设备确认，如果在指定的时间内未收到确认信息，数据将被重发。

会话层

会话层（Session Layer）是 OSI 模型的第 5 层，是用户应用程序和网络之间的接口，主要任务是：向两个实体的表示层提供建立和使用连接的方法。将不同实体之间的表示层的连接称为会话。因此会话层的任务就是组织和协调两个会话进程之间的通信，并对数据交换进行管理。

用户可以按照半双工、单工和全双工的方式建立会话。当建立会话时，用户必须提供他们想要连接的远程地址。而这些地址与 MAC（介质访问控制子层）地址或网络层的逻辑地址不同，它们是为用户专门设计的，更便于用户记忆。域名（DNS）就是一种网络上使用的远程地址，例如 `www.baidu.com` 就是一个域名。

- 会话流量控制：提供会话流量控制和交叉会话功能。
- 寻址：使用远程地址建立会话连接。
- 出错控制：从逻辑上讲会话层主要负责数据交换的建立、保持和终止，但实际的工作却是接收来自传输层的数据，并负责纠正错误。会话控制和远程过程调用均属于这一层的功能。但应注意，此层检查的错误不是通信介质的错误，而是磁盘空间、打印机缺纸等类型的高级错误。

表示层

表示层（Presentation Layer）是 OSI 模型的第六层，它对来自应用层的命令和数据进行解释，对各种语法赋予相应的含义，并按照一定的格式传送给会话层。其主要功能是“处理用户信息的表示问题，如编码、数据格式转换和加密解密”等。

表示层的具体功能如下：

- 数据格式处理：协商和建立数据交换的格式，解决各应用程序之间在数据格式表示上的差异。
- 数据的编码：处理字符集和数字的转换。例如由于用户程序中的数据类型（整型或实型、有符号或无符号等）、用户标识等都可以有不同的表示方式，因此，在设备之间需要具有在不同字符集或格式之间转换的功能。
- 压缩和解压缩：为了减少数据的传输量，这一层还负责数据的压缩与恢复。
- 数据的加密和解密：可以提高网络的安全性。

应用层

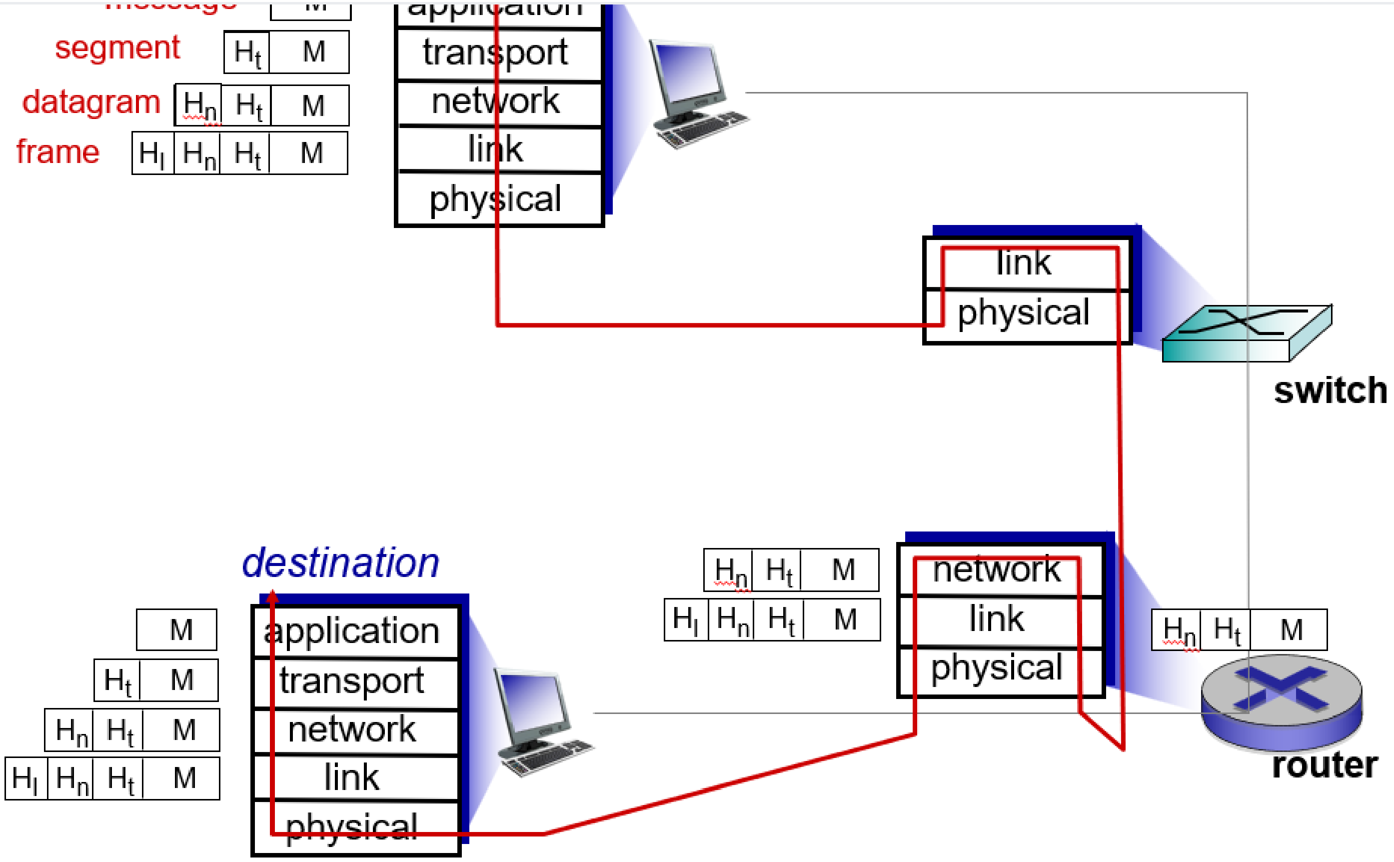
应用层（Application Layer）是 OSI 参考模型的最高层，它是计算机用户，以及各种应用程序和网络之间的接口，其功能是直接向用户提供服务，完成用户希望在网络上完成的各种工作。它在其他 6 层工作的基础上，负责完成网络中应用程序与网络操作系统之间的联系，建立与结束使用者之间的联系，并完成网络用户提出的各种网络服务及应用所需的监督、管理和服务等各种协议。此外，该层还负责协调各个应用程序间的工作。

应用层为用户提供的服务和协议有：文件服务、目录服务、文件传输服务（FTP）、远程登录服务（Telnet）、电子邮件服务（E-mail）、打印服务、安全服务、网络管理服务、数据库服务等。上述的各种网络服务由该层的不同应用协议和程序完成，不同的网络操作系统之间在功能、界面、实现技术、对硬件的支持、安全可靠性以及具有的各种应用程序接口等各个方面的差异是很大的。

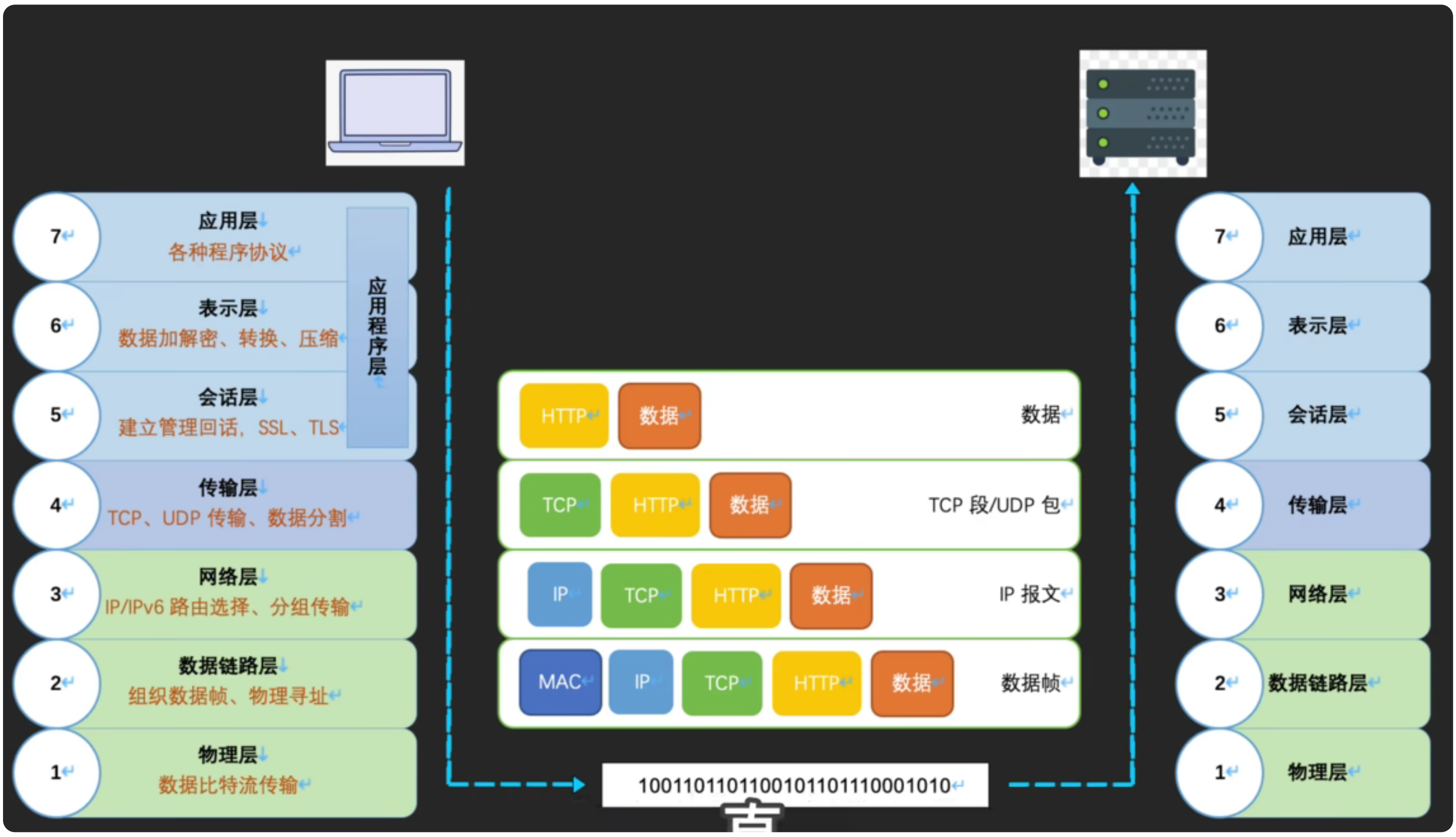
应用层的主要功能如下：

- 用户接口：应用层是用户与网络，以及应用程序与网络间的直接接口，使得用户能够与网络进行交互式联系。
- 实现各种服务：该层具有的各种应用程序可以完成和实现用户请求的各种服务。

Encapsulation



从上往下，每经过一层，协议就会在数据包包头上面做点手脚，加点东西，传送到接收端，再层层解 套出来，如下示意图：



Networks under attack: security

Put malware into hosts via Internet

malware：恶意软件，指电脑病毒等

- virus: self-replicating(自我复制) infection(感染) by receiving/executing(执行) object (e.g., e-mail attachment)
- worm: self-replicating infection by passively(被动地) receiving object that gets itself executed

Attack server, network infrastructure

Denial of Service (DoS，拒绝服务): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic.

造成 DoS 的攻击行为被称为 DoS 攻击，将大量的非法申请封包传送给指定的目标主机，其目的是完全消耗目标主机资源，使计算机或网络无法提供正常的服务。

History

计算机网络的发展阶段

第一代：远程终端连接

时间：20 世纪 60 年代早期

面向终端的计算机网络：主机是网络的中心和控制者，终端（键盘和显示器）分布在各处并与主机相连，用户通过本地的终端使用远程的主机。只提供终端和主机之间的通信，子网之间无法通信。

第二代：计算机网络阶段（局域网）

时间：20 世纪 60 年代中期

多个主机互联，实现计算机和计算机之间的通信。包括：通信子网、用户资源子网。终端用户可以访问本地主机和通信子网上所有主机的软硬件资源。电路交换和分组交换。

第三代：计算机网络互联阶段（广域网、**Internet**）

时间：1981 年国际标准化组织(ISO)制订：开放体系互联基本参考模型（OSI/RM），实现不同厂家生产的计算机之间实现互连。

TCP/IP 协议的诞生。

第四代：信息高速公路（高速，多业务，大数据量）

宽带综合业务数字网：信息高速公路 ATM 技术、ISDN、千兆以太网 交互性：网上电视点播、电视会议、可视电话、网上购物、网上银行、网络图书馆等高速、可视化。

Contributors: cworld1

Chapter 2: Application Layer

Outline

1. principles of network applications
2. Web and HTTP
3. Electronic mail: SMTP, POP3, IMAP
4. DNS
5. P2P applications
6. Video streaming and content distribution networks
7. Socket programming with UDP and TCP

Creating a network app

write programs that:

- run on (different) end systems
- communicate over network

> **TIP**
>
> e.g., web server software communicates with browser software

no need to write software for network-core devices:

- network-core devices(网络核心设备) do not run user applications
- applications on end systems(终端系统上的应用程序) allows for rapid app development, propagation(传播)

结构上:

- client-server 服务器与客户端
- peer-to-peer (P2P) 端对端

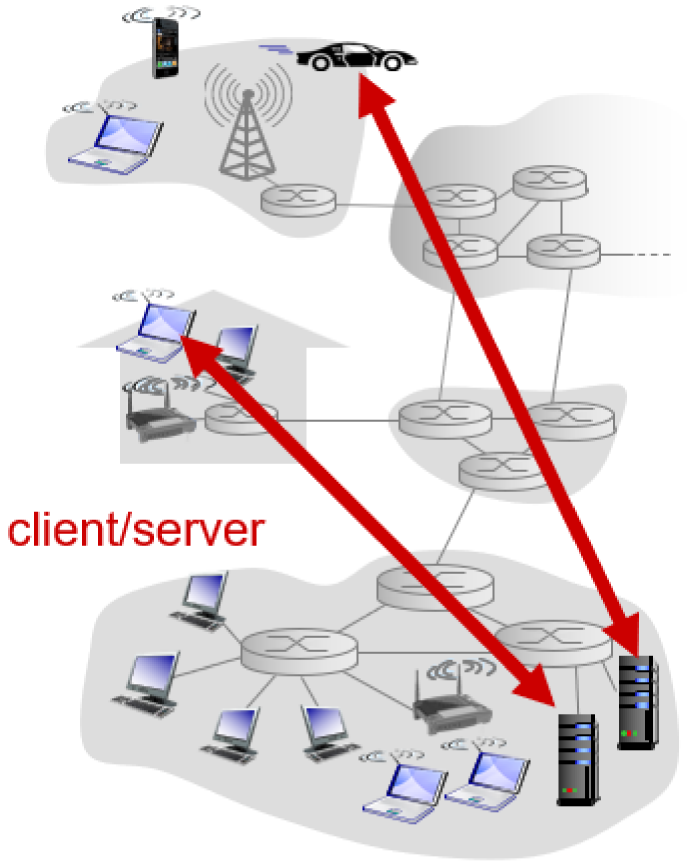
Client-server architecture

server:

- data centers for scaling(拓展)

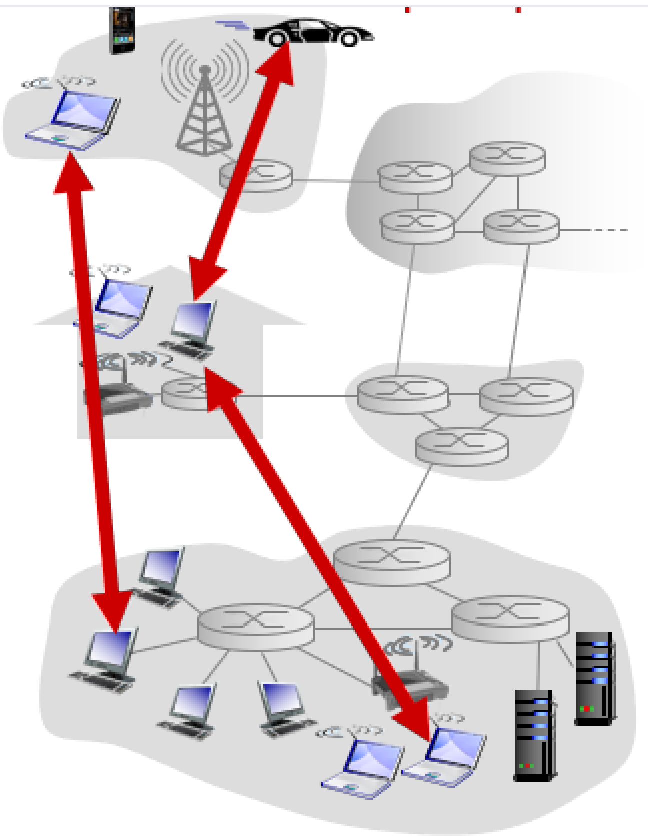
clients:

- communicate with server
- may be intermittently(间歇性的) connected
- may have dynamic IP addresses(动态 IP)
- do not communicate directly with each other



P2P architecture

- no always-on server
- arbitrary(任意的) end systems directly communicate
- peers(对等体) request service from other peers, provide service in return to other peers
- self scalability(自我弹性) – new peers bring new service capacity(容量), as well as new service demands(需求)
- peers are intermittently(间歇性) connected and change IP addresses
- complex(复杂的) management



Processes communicating

- client process: process that initiates communication
- server process: process that waits to be contacted

> **TIP**
>
> Applications with P2P architectures have client processes & server processes

Sockets

> 用大白话解释什么是 Socket - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/260139078)
>
> 套接字（socket）是一个抽象层，应用程序可以通过它发送或接收数据，可对其进行像对文件一样的打开、读写和关闭等操作。套接字允许应用程序将 I/O 插入到网络中，并与网络中的其他应用程序进行通信。网络套接字是 IP 地址与端口的组合。
>
> 我们将一个小区比作一台计算机，一台计算机里面跑了很多程序，怎么区分程序呢，用的是端口，就好像小区用门牌号区分每一户人家一样。手机送到小明家了，怎么进去呢？从大门进啊，怎么找到大门呢？门牌号呀。不就相当于从互联网来的数据找到接收端计算机后再根据端口判断应该给哪一个程序一样吗。小明家的入口就可以用小区地址+门牌号进行唯一表示，那么同样的道理，程序也可以用 IP+端口号进行唯一标识。那么这个程序的入口就被称作 Socket。

socket analogous(很相似的) to door:

- sending process shoves(推送) message out door

Addressing processes

To receive messages, process must have **identifier**.

identifier includes both IP address and port numbers associated with process on host.

What transport service does an app need?

data integrity(数据完整性)

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate(容忍，允许) some loss

timing(时效性，即时性)

- some apps (e.g., Internet telephony(网络电话), interactive games) require low delay to be “effective”

throughput(吞吐率)

- some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- other apps (“elastic apps”，弹性应用) make use of whatever throughput they get

security(安全性)

- encryption, data integrity(数据完整性)

Internet transport protocols(协议) services

TCP service:

- **reliable transport** between sending and receiving process
- **flow control(流量控制)**: sender won’t overwhelm(溢出) receiver
- **congestion control(拥塞控制)**: throttle(限制，掐死) sender when network overloaded
- does not provide: **timing, minimum throughput guarantee, security**
- **connection-oriented(面向连接)**: setup required between client and server processes

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

> TCP/IP 协议集包括应用层，传输层，网络层，网络访问层。
>
> **TCP（Transmission Control Protocol，传输控制协议）**
>
> TCP 是面向连接的协议，也就是说，在收发数据前，必须和对方建立可靠的连接。一个 TCP 连接必须要经过三次“对话”才能建立起来。
>
> **UDP（User Data Protocol，用户数据报协议）**
>
> 1、UDP 是一个非连接的协议，传输数据之前源端和终端不建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。在发送端，UDP 传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP 把每个消息段放在队列中，应用程序每次从队列中读一个消息段。
>
> 2、由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多个客户机传输相同的消息。
>
> 3、UDP 信息包的标题很短，只有 8 个字节，相对于 TCP 的 20 个字节信息包的额外开销很小。
>
> 4、吞吐量不受拥挤控制算法的调节，只受应用软件生成数据的速率、传输带宽、源端和终端主机性能的限制。
>
> 5、UDP 使用尽最大努力交付，即不保证可靠交付，因此主机不需要维持复杂的链接状态表（这里面有许多参数）。
>
> 6、UDP 是面向报文的。发送方的 UDP 对应用程序交下来的报文，在添加首部后就向下交付给 IP 层。既不拆分，也不合并，而是保留这些报文的边界，因此，应用程序需要选择合适的报文大小。

Securing TCP and SSL

TCP & UDP 存在的问题：

- no encryption
- cleartext passwds(明文密码) sent into socket traverse(通过) Internet in cleartext

SSL：

- provides encrypted TCP connection
- data integrity(数据完整性)
- end-point authentication(身份验证)

此外注意：

- SSL is at app layer(位于应用层): apps use SSL libraries, that “talk” to TCP

- SSL socket API: cleartext passwords sent into socket traverse Internet encrypted

Web and HTTP

> HTTP 的发展 - HTTP | MDN (mozilla.org) (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP)
>
> **HTTP**（HyperText Transfer Protocol）是万维网（World Wide Web）的基础协议。自 Tim Berners-Lee 博士和他的团队在 1989-1991 年间创造出它以来，HTTP 已经发生了太多的变化，在保持协议简单性的同时，不断扩展其灵活性。如今，HTTP 已经从一个只在实验室之间交换文件的早期协议进化到了可以传输图片，高分辨率视频和 3D 效果的现代复杂互联网协议。
>
> 万维网的发明

- [illegible] *(https://developer.mozilla.org/zh-CN/docs/Web/HTML)* [illegible]HTML[illegible]
- [illegible]HTTP[illegible]
- [illegible] *WorldWideWeb*[illegible]
- [illegible] *httpd* [illegible]

HTTP [illegible] HTTP/0.9[illegible]one-line[illegible]

HTTP/0.9——[illegible]

[illegible] HTTP [illegible] 0.9 [illegible]HTTP/0.9 [illegible] `GET` (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET) [illegible]

[illegible]

```text
GET /mypage.html
```

[illegible]

```text
<HTML>
[illegible] HTML [illegible]
</HTML>
```

[illegible]HTTP/0.9 [illegible] HTTP [illegible] HTML [illegible] HTML [illegible]

HTTP/1.0——[illegible]

[illegible] HTTP/0.9 [illegible]

- [illegible] `HTTP/1.0` [illegible] `GET` [illegible]
- [illegible]
- [illegible] HTTP [illegible]
- [illegible] HTTP [illegible] HTML [illegible] `Content-Type` (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type) [illegible]

[illegible]

```text
GET /mypage.html HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)

200 OK
Date: Tue, 15 Nov 1994 08:12:31 GMT
Server: CERN/3.0 libwww/2.17
```

```
□□□□□□□□□
  <IMG SRC="/myimage.gif">
</HTML>
```

接下来是第二个连接，请求获取图片（并具有相同的响应）：

text

```
GET /myimage.gif HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)

200 OK
Date: Tue, 15 Nov 1994 08:12:32 GMT
Server: CERN/3.0 libwww/2.17
Content-Type: text/gif
(这里是图片内容)
```

在 1991-1995 年，这些新扩展并没有被引入到标准中以促进协助工作，而仅仅作为一种尝试。服务器和浏览器添加这些新扩展功能，但出现了大量的互操作问题。直到 1996 年 11 月，为了解决这些问题，一份新文档（RFC 1945）被发表出来，用以描述如何操作实践这些新扩展功能。文档 RFC 1945 (https://datatracker.ietf.org/doc/html/rfc1945) 定义了 HTTP/1.0，但它是狭义的，并不是官方标准。

HTTP/1.1——标准化的协议

HTTP/1.0 多种不同的实现方式在实际运用中显得有些混乱。自 1995 年开始，即 HTTP/1.0 文档发布的下一年，就开始修订 HTTP 的第一个标准化版本。在 1997 年初，HTTP1.1 标准发布，就在 HTTP/1.0 发布的几个月后。

HTTP/1.1 消除了大量歧义内容并引入了多项改进：

- 连接可以复用，节省了多次打开 TCP 连接加载网页文档资源的时间。
- 增加管线化技术，允许在第一个应答被完全发送之前就发送第二个请求，以降低通信延迟。
- 支持响应分块。
- 引入额外的缓存控制机制。
- 引入内容协商机制，包括语言、编码、类型等。并允许客户端和服务器之间约定以最合适的内容进行交换。
- 凭借 `Host` (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Host) 标头，能够使不同域名配置在同一个 IP 地址的服务器上。

一个典型的请求流程，所有请求都通过一个连接实现，看起来就像这样：

text

```
GET /en-US/docs/Glossary/Simple_header HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/2010
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/en-US/docs/Glossary/Simple_header

200 OK
Connection: Keep-Alive
```

```
Etag: "547fa7e369ef56031dd3bff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 19 Jul 2016 00:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding

(content)


GET /static/img/header-background.png HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/2010
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/en-US/docs/Glossary/Simple_header

200 OK
Age: 9578461
Cache-Control: public, max-age=315360000
Connection: keep-alive
Content-Length: 3077
Content-Type: image/png
Date: Thu, 31 Mar 2016 13:34:46 GMT
Last-Modified: Wed, 21 Oct 2015 18:27:50 GMT
Server: Apache

(image content of 3077 bytes)
```

HTTP/1.1 在 1997 年 1 月以 RFC 2068 (https://datatracker.ietf.org/doc/html/rfc2068) 文件发布。由于 HTTP 协议的可扩展性使得创建新的头部和方法是很容易的。即使 HTTP/1.1 协议进行过两次修订，RFC 2616 (https://datatracker.ietf.org/doc/html/rfc2616) 发布于 1999 年 6 月，而另外两个文档 RFC 7230 (https://datatracker.ietf.org/doc/html/rfc7230) -RFC 7235 (https://datatracker.ietf.org/doc/html/rfc7235) 发布于 2014 年 6 月（在 HTTP/2 发布之前）。HTTP/1.1 协议已经稳定使用超过 15 年了。

HTTP overview

HTTP: hypertext(超文本) transfer protocol

- Web’s application layer protocol
- **client/server model**

HTTP request message: ASCII (human-readable format)

Uploading form input

POST method:

- web page often includes form input
- input is uploaded to server in entity(实体) body

URL method:

- uses GET method
- input is uploaded in URL field of request line: 如 `www.somesite.com/animalsearch?monkeys&banana`

Method types

HTTP/1.0:

- GET
- POST
- HEAD: asks server to leave requested object out of response

HTTP/1.1:

- GET
- POST
- HEAD
- PUT: uploads file in entity body to path specified in URL field
- DELETE: deletes file specified in the URL field

HTTP response status codes(HTTP 响应状态码)

status code appears in 1st line in server-to-client response message.

> HTTP 响应状态码 - HTTP | MDN (mozilla.org) (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status)
> HTTP 响应状态码用来表明特定 HTTP (https://developer.mozilla.org/zh-CN/docs/Web/HTTP) 请求是否成功完成。响应被归为以下五大类:
>
> 1. 信息响应 (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status#%E4%BF%A1%E6%81%AF%E5%93%8D%E5%BA%94) (`100` –

CN/docs/Web/HTTP/Status#%E6%88%90%E5%8A%9F%E5%93%8D%E5%BA%94) (`200` – `299`)

3. 重定向消息 (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status#%E9%87%8D%E5%AE%9A%E5%90%91%E6%B6%88%E6%81%AF) (`300` – `399`)
4. 客户端错误响应 (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status#%E5%AE%A2%E6%88%B7%E7%AB%AF%E9%94%99%E8%AF%AF%E5%93%8D%E5%BA%94) (`400` – `499`)
5. 服务端错误响应 (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status#%E6%9C%8D%E5%8A%A1%E7%AB%AF%E9%94%99%E8%AF%AF%E5%93%8D%E5%BA%94) (`500` – `599`)

备注：如果您收到的响应不在 此列表 (https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status#%E4%BF%A1%E6%81%AF%E5%93%8D%E5%BA%94) 中，则它为非标准响应，可能是服务器软件的自定义响应。

HTTP response status codes - PlayFab | Microsoft Learn (https://learn.microsoft.com/zh-cn/gaming/playfab/api-references/http-response-status-codes)

常见的相应状态码：

HTTP status code	General description
100	Continue: Returned on HEAD requests
200	OK: Returned for all successful requests. May indicate partial success for bulk APIs.
201	Created: Request was successful and resource was created.
202	Accepted: Request was successful, processing will continue asynchronously.
204	No Content: API successful, but there is no response to be returned from the API.
301	Moved Permanently: requested object moved, new location specified later in this msg (Location:)
400	Bad Request: Parameters in request where invalid or request payload structure was invalid. Do not retry.
401	Unauthorized: Caller is not authorized to either call the specific API or perform the action requested. Do not retry.
403	Forbidden: Caller is not allowed access. Do not retry.
404	Not Found: API does not exist. Do not retry.

408	Request Timeout: The request took too long to be sent to the server. Okay to retry using exponential backoff pattern.
409	Conflict: A concurrency error occurred between two API calls. Okay to retry using exponential backoff pattern.
413	Payload Too Large: The request is larger than the server is allowed to handle. Do not retry. If unexpected, contact support.
414	URI Too Long: The URI in the request is longer than the server is allowed to handle. Do not retry.
429	Too Many Requests: API calls are being rate limited. Pause and then retry request, check if API returned “Retry-After” header or retryAfter in JSON response for delay needed.
500	Internal Server Error: An error occurred on the PlayFab server. Okay to retry, contact support if problem persists.
501	Not Implemented: The API called has not been implemented yet. Do not retry.
502	Bad Gateway: PlayFab API servers are not available to process the request. Pause and then retry request using exponential backoff pattern.
503	Service Unavailable: PlayFab API servers are not available to process the request. Pause and then retry request using exponential backoff pattern.
504	Gateway Timeout: PlayFab API servers are not available to process the request. Pause and then retry request.
505	HTTP Version Not Supported

User-server state: cookies

many Web sites use cookies

four components:

1. cookie header line of HTTP *response* message
2. cookie header line in next HTTP *request* message
3. cookie file kept on user’s host, managed by user’s browser
4. back-end database at Web site

what cookies can be used for:

- authorization
- shopping carts(□□□□□□□□□□□□□□□□□□)

Web caches (proxy server)

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache

  object in cache: cache returns object else **cache** requests object from origin server, then returns object to client

Web 缓存（也称为代理服务器）是广泛用于提高 Web 页面加载速度和减轻网络流量压力的技术。它的工作方式是在用户和 Web 服务器之间放置一个位于网络边缘的服务器，该服务器存储先前获取的 Web 页面的副本，以便当后续用户请求相同页面时，可以直接返回本地存储的副本，而不必重新从源服务器下载。这样，用户就可以更快地加载页面，并且减轻了源服务器的压力。

Web 缓存可以根据不同的应用场景进行配置，例如在企业内部网络中设置私有 Web 缓存来提高内部网站的访问速度，或在 Internet 网络边缘设置公共 Web 缓存来减轻互联网上的流量压力。此外，Web 缓存还可以通过缓存一些常用的 JavaScript 和 CSS 文件等静态资源，来进一步提高页面加载速度。

需要注意的是，Web 缓存不适用于动态生成的 Web 页面或需要个性化处理的页面，因为这些页面的内容是根据用户请求动态生成的，所以不可能事先缓存下来。此外，一些受保护的 Web 页面（如登录页）也不能缓存，因为每个用户的页面内容都是不同的，不适合缓存下来供其他用户使用。

Conditional GET

Conditional GET 是一个 HTTP 协议的特性，可以在客户端像服务器发起请求时，通过发送 HTTP 头部中的条件标识，让服务器判断资源是否发生了变化，如果资源未发生变化，服务器可以选择不返回资源实体，而是返回 304 Not Modified 状态码，从而节省网络带宽和服务器资源。

使用条件 GET 时，客户端发送的请求中会包含 If-Modified-Since、If-Unmodified-Since、If-Match、If-None-Match 等条件标识，这些条件标识指定了客户端请求的资源应该满足的条件。当服务器对客户端的请求进行处理时，会根据这些条件是否满足来决定是否返回实际的资源，或者只返回 304 状态码。

条件 GET 可以提高网络传输效率和响应速度，因为服务器避免了重复发送已经有缓存的资源，从而降低了网络带宽的消耗。

Electronic Mail

> Introduction to Electronic Mail - GeeksforGeeks (https://www.geeksforgeeks.org/introduction-to-electronic-mail/)
> Electronic mail, commonly known as email, is a method of exchanging messages over the internet.
> Here are the basics of email:
>
> 1. An email address: This is a unique identifier for each user, typically in the format of `name@domain.com` .
> 2. An email client: This is a software program used to send, receive and manage emails, such as Gmail, Outlook, or Apple Mail.

SMTP (Simple Mail Transfer Protocol)、POP3 (Post Office Protocol Version 3) 和 IMAP (Internet Message Access Protocol) 是三種常用的電子郵件協議。

SMTP

SMTP (Simple Mail Transfer Protocol [RFC 2821]): 是一種用於從源地址到目的地址傳送郵件的規範，通過SMTP 協議來控制郵件的中轉方式。

1. 當用戶使用 Outlook 等郵件客戶端軟件
2. 當郵件服務器與另一台郵件服務器通過 SMTP 協議相互連接時
3. 當郵件服務器接收來自客戶端軟件的郵件時
4. 當郵件服務器向客戶端軟件投遞郵件時

SMTP 的運作流程：

1. 連線：SMTP connection opened，客戶端與伺服器建立 TCP 連線25 埠，伺服器回應 EHLO 指令，表示準備好接收郵件，回應 250 OK 訊息。

2. 傳送：Email data transferred，客戶端傳送郵件內容給伺服器。

3. **Mail Transfer Agent (MTA):** 伺服器收到郵件後，MTA 會檢查收件人 SMTP 伺服器是否存在。 SMTP 伺服器MTA 會查詢 DNS 伺服器，以找到收件人伺服器的 MTA 位址 Sendmail、Postfix 或 Qmail 等。

   > What is the Simple Mail Transfer Protocol (SMTP)? | Cloudflare (https://www.cloudflare.com/learning/email-security/what-is-smtp/)
   > The server runs a program called a Mail Transfer Agent (MTA). The MTA checks the domain of the recipient's email address, and if it differs from the sender's, it queries the Domain Name System (DNS) (https://www.cloudflare.com/learning/dns/what-is-dns/) to find the recipient's IP address (https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/) . This is like a post office looking up a mail recipient's zip code.

4. 連線：Connection closed，客戶端傳送 QUIT 指令，伺服器回應 221 訊息。

POP3 協議允許從伺服器上下載 POP3 郵件，這是客戶端訪問服務器和使用服務器上的電子郵件的最簡單協議。

IMAP 協議允許從伺服器上讀取 POP3 郵件，IMAP 是比 POP3 更新的協議，它支持客戶端訪問服務器和使用服務器上的電子郵件的更多功能。

> What is the Simple Mail Transfer Protocol (SMTP)? | Cloudflare (https://www.cloudflare.com/learning/email-security/what-is-smtp/)
> What are SMTP commands?

server to accept data correctly.

- `HELO/EHLO` : These commands say "Hello" and start off the SMTP connection between client and server. " `HELO` " is the basic version of this command; " `EHLO` " is for a specialized type of SMTP.
- `MAIL FROM` : This tells the server who is sending the email. If Alice were trying to email her friend Bob, a client might send "MAIL FROM:alice@example.com".
- `RCPT TO` : This command is for listing the email's recipients. A client can send this command multiple times if there are multiple recipients. In the example above, Alice's email client would send "RCPT TO:bob@example.com".
- `DATA` : This precedes the content of the email.

  It obeys RFC 822: standard for text message format. Like:

```text
1  DATA
2  Date: Mon, 4 April 2022
3  From: Alice alice@example.com
4  Subject: Eggs benedict casserole
5  To: Bob bob@example.com
6
7  Hi Bob,
8  I will bring the eggs benedict casserole recipe on Friday.
9  -Alice
10 .
```

Pay attention to line 6 and line 10.

- `RSET` : This command resets the connection, removing all previously transferred information **without closing the SMTP connection**. `RSET` is used if the client sent incorrect information.
- `QUIT` : This ends the connection.

- SMTP uses persistent(□□□□□□□□) connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses **CRLF.CRLF**(□□□□□□) to determine end of message

comparison with HTTP:

- HTTP: pull
- SMTP: push

POP

POP (Post Office Protocol [RFC 1939]): authorization, download.

而 POP3 是一种用于接收电子邮件的协议。当您使用 POP3 协议从邮件服务器下载邮件时，邮件服务器就会将邮件传送到您的邮件客户端中，并从服务器上删除邮件。

IMAP

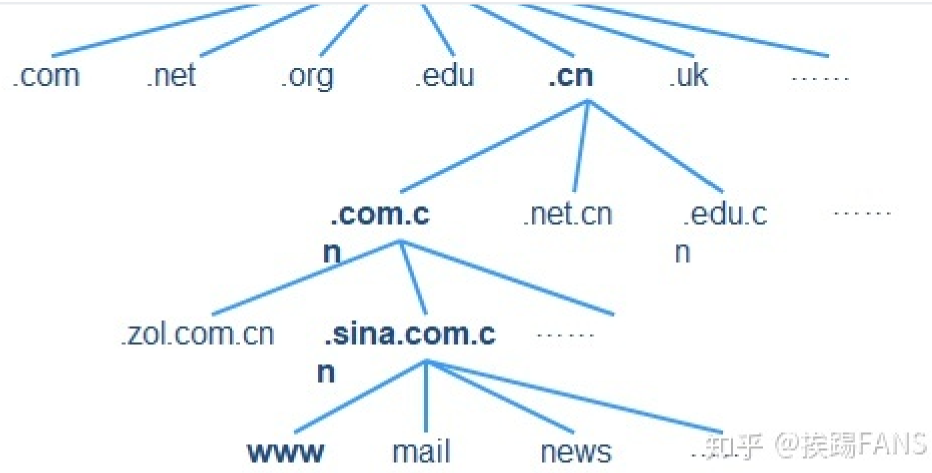
IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server.

IMAP 是另一种用于接收电子邮件的协议，它与 POP3 不同的是，它允许您在邮件服务器上保留邮件的副本（§keeps all messages in one place: at server），可以在多个设备之间同步邮件。这样，当您在一个设备上修改或删除邮件时，在其他设备上也可以看到这些修改。此外还有 allows user to organize messages in folders。

DNS

什么是 DNS

> 什么是 DNS？ - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/186028919)
>
> ### DNS 出现及演化
>
> 网络出现的早期 是使用 IP 地址通讯的，那时就几台主机通讯。但是随着接入网络主机的增多，这种数字标识的地址非常不便于记忆，UNIX 上就出现了建立一个叫做 hosts 的文件（Linux 和 Windows 也继承保留了这个文件）。这个文件中记录这主机名称和 IP 地址的对应表。这样只要输入主机名称，系统就会去加载 hosts 文件并查找对应关系，找到对应的 IP，就可以访问这个 IP 的主机了。
>
> 但是后来主机太多了，无法保证所有人都能拿到统一的最新的 hosts 文件，就出现了在文件服务器上集中存放 hosts 文件，以供下载使用。互联网规模进一步扩大，这种方式也不堪负重，而且把所有地址解析记录形成的文件都同步到所有的客户机似乎也不是一个好办法。这时 DNS 系统出现了，随着解析规模的继续扩大，DNS 系统也在不断的演化，直到现今的多层架构体系。
>
> ### DNS 概括
>
> DNS（Domain Name System，域名系统），因特网上作为域名和 IP 地址互相映射的一个分布式数据库，能够使用户更方便的访问互联网，而不用去记住能够被机器直接读取的 IP 数串。通过主机名，最终得到该主机对应的 IP 地址的过程叫做域名解析（或主机名解析）。DNS 协议运行在 UDP 协议之上，使用端口号 53。
>
> DNS 的分布数据库是以域名为索引的，每个域名实际上就是一棵很大的逆向树中路径，这棵逆向树称为域名空间（domain name space），如下图所示树的最大深度不得超过 127 层，树中每个节点都有一个可以长达 63 个字符的文本标号。
>
> ### DNS 的作用
>
> - 正向解析：根据主机名称（域名）查找对应的 IP 地址
> - 反向解析：根据 IP 地址查找对应的主机域名
>
> DNS 系统的分布式数据结构：



DNS 相关服务器及实现

什么是 DNS-DNS 如何工作-权威性 DNS 服务器 | Cloudflare 中国官网 | Cloudflare (https://www.cloudflare.com/zh-cn/learning/dns/what-is-dns/)

加载网页涉及 4 个 DNS 服务器

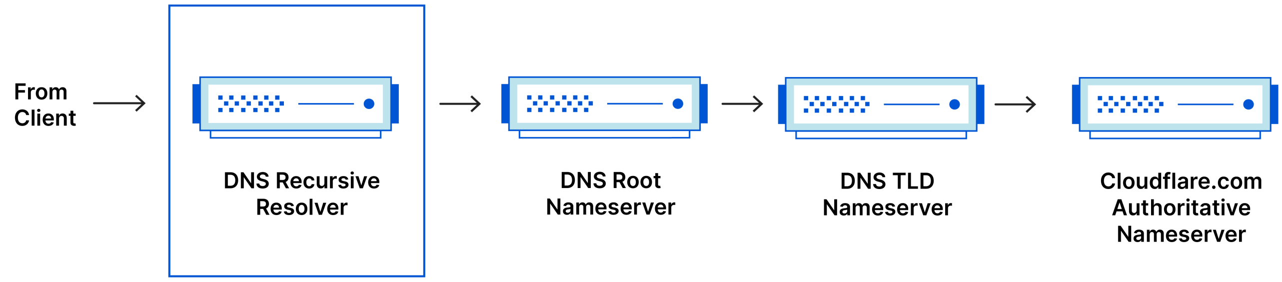
- DNS 解析器 (https://www.cloudflare.com/learning/dns/dns-server-types#recursive-resolver)：该解析器可被视为被要求去图书馆的某个地方查找特定图书的图书馆员。DNS 解析器是一种服务器，旨在通过 Web 浏览器等应用程序接收客户端计算机的查询。然后，解析器一般负责发出其他请求，以便满足客户端的 DNS 查询。
- 根域名服务器 (https://www.cloudflare.com/learning/dns/glossary/dns-root-server/)：根域名服务器是将人类可读的主机名转换（解析）为 IP 地址的第一步。可将其视为指向不同书架的图书馆中的索引 - 一般其作为对其他更具体位置的引用。
- TLD 名称服务器 (https://www.cloudflare.com/learning/dns/dns-server-types#tld-nameserver)：顶级域名服务器（TLD (https://www.cloudflare.com/learning/dns/top-level-domain/)，Top-level domain）可看做是图书馆中一个特殊的书架。这个域名服务器是搜索特定 IP 地址的下一步，其上托管了主机名的最后一部分（例如，在 example.com 中，TLD 服务器为 “com”）。
- 权威性域名服务器 (https://www.cloudflare.com/learning/dns/dns-server-types#authoritative-nameserver)：可将这个最终域名服务器视为书架上的字典，其中特定名称可被转换成其定义。权威性域名服务器是域名服务器查询中的最后一站。如果权威性域名服务器能够访问请求的记录，则其会将已请求主机名的 IP 地址返回到发出初始请求的 DNS 解析器（图书管理员）。

权威性 DNS 服务器与递归 DNS 解析器之间的区别是什么?

这两个概念都是指 DNS 基础设施不可或缺的服务器（服务器组），但各自担当不同的角色，并且位于 DNS 查询管道内的不同位置。考虑二者差异的一种方式是，递归 (https://www.cloudflare.com/learning/dns/what-is-recursive-dns/) 解析器位于 DNS 查询的开头，而权威性域名服务器位于末尾。

递归 DNS 解析器

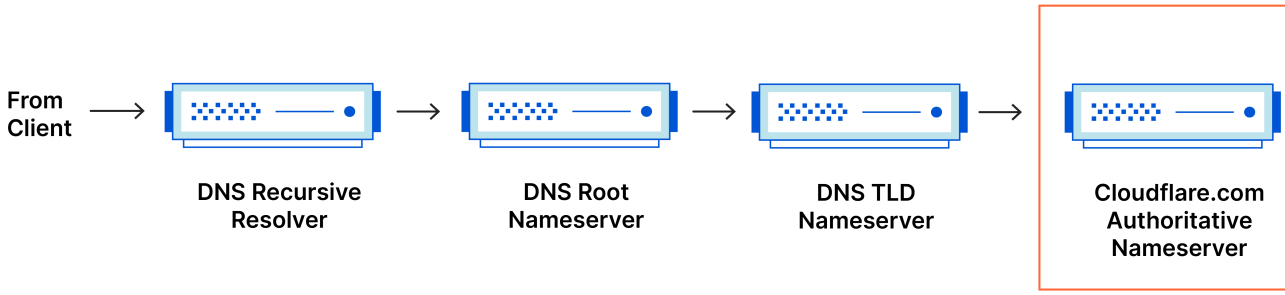
递归解析器是一种计算机，其响应来自客户端的递归请求并花时间追踪 DNS 记录 (https://www.cloudflare.com/learning/dns/dns-records/)。为执行此操作，其发出一系列请求，直至到达用于所请求的记录的权威性 DNS 域名服务器为止（或者超时，或者如果未找到记录，则返回错误）。幸运的是，递归 DNS 解析器并不总是需要发出多个请求才能追踪响应客户端所需的记录；缓存 (https://www.cloudflare.com/learning/cdn/what-is-caching/) 是一种数据持久性过程，可通过在 DNS 查找中更早地服务于所请求的资源记录来为所需的请求提供捷径。



权威性 DNS 服务器

简言之，权威性 DNS 服务器是实际持有并负责 DNS 资源记录的服务器。这是位于 DNS 查找链底部的服务器，其将使用所查询的资源记录进行响应，从而最终允许发出请求的 Web 浏览器达到访问网站或其他 Web 资源所需的 IP 地址。权威性域名服务器从自身数据满足查询需求，无需查询其他来源，因为这是某些 DNS 记录的最终真实来源。

DNS Record Request Sequence



DNS 查找的 8 个步骤

1. 用户在 Web 浏览器中键入 “example.com”，查询传输到 Internet 中，并被 DNS 递归解析器接收。

2. 接着，解析器查询 DNS 根域名服务器（.）。

3. 然后，根服务器使用存储其域信息的顶级域（TLD）DNS 服务器（例如 .com 或 .net）的地址响应该解析器。在搜索 example.com 时，我们的请求指向 .com TLD。

4. 然后，解析器向 .com TLD 发出请求。

5. TLD 服务器随后使用该域的域名服务器 example.com 的 IP 地址进行响应。

6. 最后，递归解析器将查询发送到域的域名服务器。

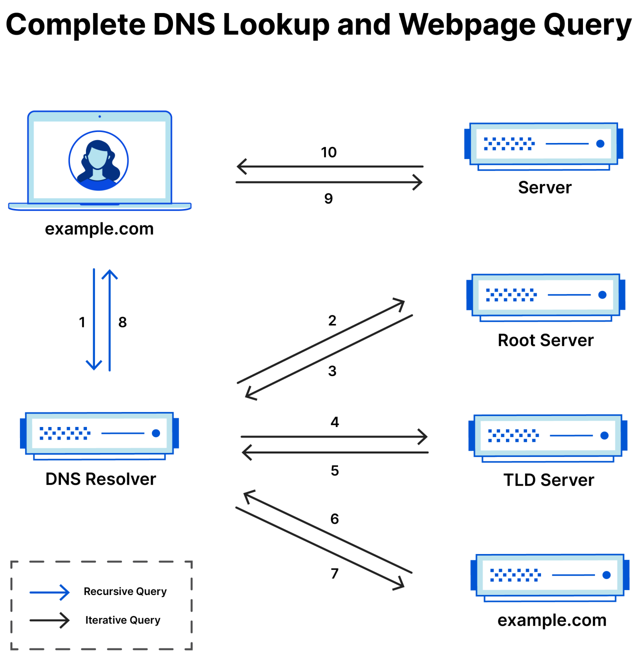
7. example.com 的 IP 地址而后从域名服务器返回解析器。

8. 然后 DNS 解析器使用最初请求的域的 IP 地址响应 Web 浏览器。

protocol-http/) 请求。

11. 位于该 IP 的服务器返回将在浏览器中呈现的网页（第 10 步）。

Complete DNS Lookup and Webpage Query



什么是 DNS 高速缓存？DNS 高速缓存发生在哪里？

缓存的目的是将数据临时存储在某个位置，从而提高数据请求的性能和可靠性。DNS 高速缓存涉及将数据存储在更靠近请求客户端的位置，以便能够更早地解析 DNS 查询，并且能够避免在 DNS 查找链中进一步向下的额外查询，从而缩短加载时间并减少带宽/CPU 消耗。DNS 数据可缓存到各种不同的位置上，每个位置均将存储 DNS 记录并保存由生存时间（TTL，time to live）(https://www.cloudflare.com/learning/cdn/glossary/time-to-live-ttl/) 决定的一段时间。

浏览器 DNS 缓存

现代 Web 浏览器设计为默认将 DNS 记录缓存一段时间。目的很明显；越靠近 Web 浏览器进行 DNS 缓存，为检查缓存并向 IP 地址发出正确请求而必须采取的处理步骤就越少。发出对 DNS 记录的请求时，浏览器缓存是针对所请求的记录而检查的第一个位置。

在 Chrome 浏览器中，您可以转到 chrome://net-internals/#dns 查看 DNS 缓存的状态。

操作系统（OS）级 DNS 缓存

操作系统级 DNS 解析器是 DNS 查询离开您计算机前的第二站，也是本地最后一站。操作系统内旨在处理此查询的过程通常称为“存根解析器”或 DNS 客户端。当存根解析器获取来自某个应用程序的请求时，其首先检查自己的缓存，以便查看是否有此记录。如果没有，则将本地网络外部的 DNS 查询（设置了递归标记）发送到 Internet 服务提供商（ISP）内部的 DNS 递归解析器。

与先前所有步骤一样，当 ISP 内的递归解析器收到 DNS 查询时，其还将查看所请求的主机到 IP 地址转换是否已经存储在其本地持久性层中。

根据其缓存中具有的记录类型，递归解析器还具有其他功能：

1. 如果解析器没有 A 记录 (https://www.cloudflare.com/learning/dns/dns-records/dns-a-record/) ，但确实有针对权威性域名服务器的 NS 记录 (https://www.cloudflare.com/learning/dns/dns-records/dns-ns-record/) ，则其将直接查询这些域名服务器，从而绕过 DNS 查询中的几个步骤。此快捷方式可防止从根和 .com 域名服务器（在我们对 example.com 的搜索中）进行查找，并且有助于更快地解析 DNS 查询。
2. 如果解析器没有 NS 记录，它会向 TLD 服务器（本例中为 .com）发送查询，从而跳过根服务器。
3. 万一解析器没有指向 TLD 服务器的记录，其将查询根服务器。这种情况通常在清除了 DNS 高速缓存后发生。

DNS Records

> DNS 记录 | Cloudflare (https://www.cloudflare.com/zh-cn/learning/dns/dns-records/)
> DNS (https://www.cloudflare.com/learning/dns/what-is-dns/) 记录（又名区域文件）是位于权威 DNS 服务器 (https://www.cloudflare.com/learning/dns/dns-server-types/) 中的指令，提供一个域的相关信息，包括哪些 IP 地址 (https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/) 与该域关联，以及如何处理对该域的请求。这些记录由一系列以所谓的 DNS 语法编写的文本文件组成。DNS 语法是用作命令的字符串，这些命令告诉 DNS 服务器执行什么操作。此外，所有 DNS 记录都有一个 “TTL (https://www.cloudflare.com/learning/cdn/glossary/time-to-live-ttl/) ”，其代表生存时间，指示 DNS 服务器多久刷新一次该记录。
> 最常见的 DNS 记录有：
>
> - **A** 记录：保存域的 IP 地址的记录。
> - **AAAA** 记录：包含域的 IPv6 地址的记录（与 A 记录相反，A 记录列出的是 IPv4 地址）。
> - **CNAME** 记录：将一个域或子域转发到另一个域，不提供 IP 地址。
> - **MX** 记录：将邮件定向到电子邮件服务器。
> - **TXT** 记录：可让管理员在记录中存储文本注释。这些记录通常用于电子邮件安全。
> - **NS** 记录：存储 DNS 条目的名称服务器。
> - **SOA** 记录：存储域的管理信息。
> - **SRV** 记录：指定用于特定服务的端口。
> - **PTR** 记录：在反向查询中提供域名。

P2P Applications（不重要）

Pure P2P architecture(体系结构)

- no always-on server
- arbitrary(任意的) end systems directly communicate(直接通信)
- peers(原叫同龄人，这里指对等端) are intermittently connected and change IP addresses

Video streaming and content distribution networks (CDNs)（不重要）

Video traffic

Video traffic: major consumer of Internet bandwidth

solution: distributed, application-level infrastructure(分布式应用程序级基础架构)

- CBR（恒定比特率）是指视频编码时采用固定的比特率来保证视频的一致性和稳定性，不论视频场景中的运动、细节、色彩变化等情况如何。因此，CBR 编码方式适用于要求视频画质和大小都相同的场合，比如直播、视频会议等。

如：

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG 2 (DVD) 3-6 Mbps
- MPEG 4 (often used in Internet, < 1 Mbps)

Streaming multimedia: DASH

DASH（Dynamic, Adaptive Streaming over HTTP）即动态自适应流式传输协议，是一种通过互联网以 HTTP 协议传输音视频内容的方法。DASH 协议允许在不同的网络环境下自动调整音视频的码率和分辨率，以保证用户可以获得最佳的观看体验。简而言之，DASH 协议可以根据用户的网络环境和设备性能自动选择最适合的视频质量，以确保视频的流畅播放和高质量观看。

CDN

CDN(Content Distribution Networks) 即内容分发网络，是一种在网络边缘部署节点的技术，通过靠近用户的部署位置，提高用户访问网站内容的速度和质量。以下是 CDN 的相关知识点：

1. CDN 的工作原理：CDN 通过将原始服务器上的内容缓存到分布在全球各地的服务器节点上，使用户请求能够从离用户最近的服务器节点获取内容，从而提高用户访问速度和性能。

2. CDN 的好处：CDN 可以帮助网站提高用户访问速度、减少带宽成本、提高网站的可用性、减轻服务器负载等。

3. CDN 的部署方式：CDN 可以采用两种部署方式，即自建 CDN 和使用第三方 CDN。自建 CDN 需要投入大量的资金和技术力量，而使用第三方 CDN 的成本较低，但需要考虑接入成本和服务质量等问题。

4. CDN 的缓存方式：CDN 缓存方式可以分为边缘缓存和中心缓存两种。边缘缓存是将内容缓存到离用户最近的服务器节点上，中心缓存是将内容缓存到核心节点上，然后由核心节点向各个边缘节点分发。

5. CDN 的工作流程：CDN 的工作流程可以分为 DNS 解析、请求路由、内容缓存和内容传输四个阶段。DNS 解析负责将用户请求路由到最佳的 CDN 节点，请求路由将请求分发到最近的边缘节点上，内容缓存负责将内容存储到缓存中，内容传输将缓存内容传输给用户。

Socket programming with UDP and TCP

网络协议由三个要素组成，分别是语义、语法和时序。

- 语义是解释控制信息每个部分的含义，它规定了需要发出何种控制信息，以及完成的动作与做出什么样的响应；
- 语法是用户数据与控制信息的结构与格式，以及数据出现的顺序；
- 时序是对事件发生顺序的详细说明。

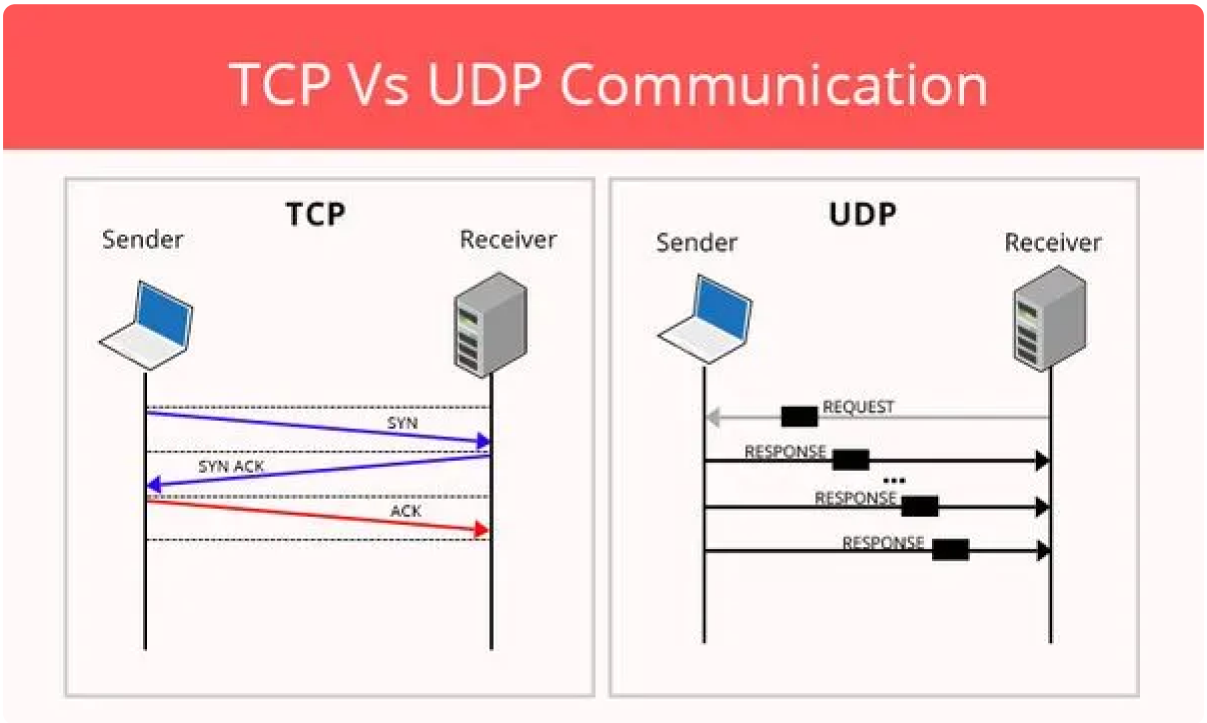
人们形象地将这三个要素描述为：语义表示要做什么，语法表示要怎么做，时序表示做的顺序。

Socket programming

Goal: learn how to build client/server applications that communicate using sockets.

Two socket types for two transport services:

Building TCP and UDP Client-Server Interactions | by Matthew MacFarquhar | Dev Genius (https://blog.devgenius.io/building-tcp-and-udp-client-server-interactions-eb8228644da2)



This photo is a great visual representation of what is going on between the server and the client. In a Connection-oriented system, the server and client send these SYN, SYN-ACK and ACK messages to ensure the packet is successfully received, if this pattern(模式) is broken (i.e. one of these is not sent) then the packet can be re-sent.

UDP 与 TCP 比较

Python 绝技 —— UDP 服务器与客户端 - i 春秋 - 博客园 (cnblogs.com) (https://www.cnblogs.com/ichunqiu/p/9200723.html)

为了更直观地比较 TCP 与 UDP 的异同，笔者将其整理成以下表格：

	TCP	UDP
连接模式	面向连接（单点通信）	无连接（多点通信）
传输可靠性	可靠	不可靠
通信模式	基于字节流	基于数据报
报头结构	复杂（至少 20 字节）	简单（8 字节）
传输速度	慢	快
资源需求	多	少
到达顺序	保证	不保证

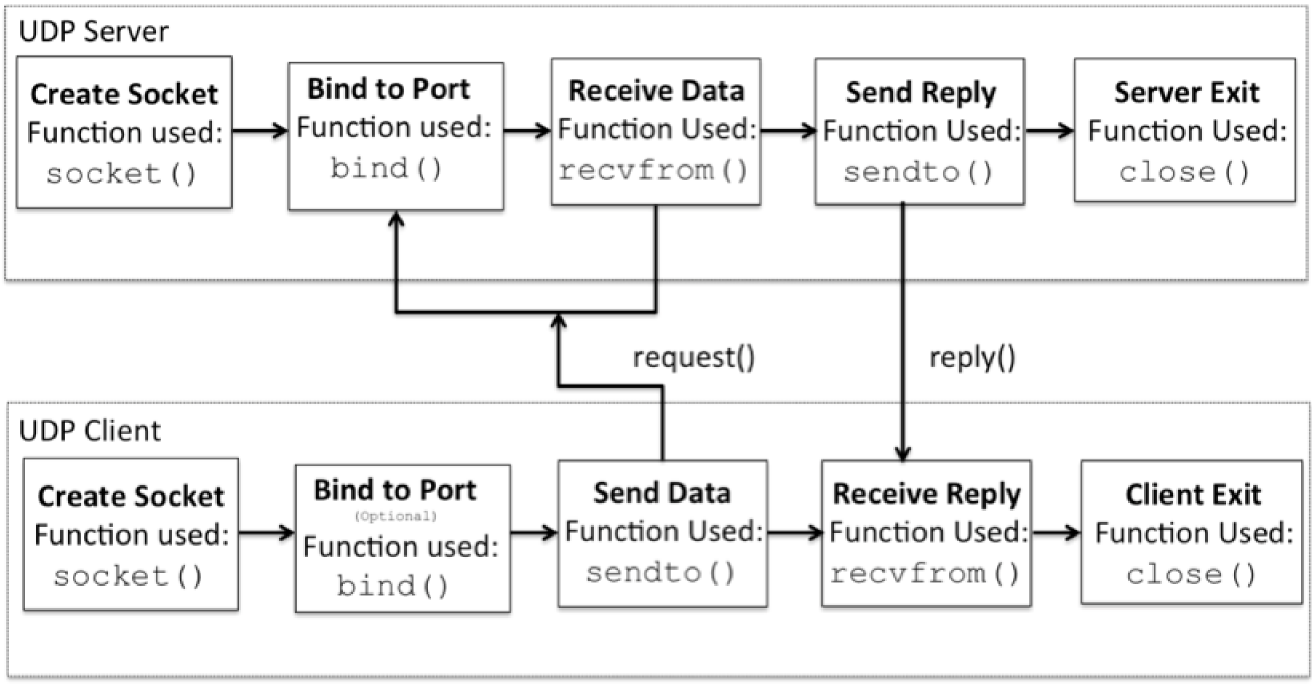
□□□□	□	□
拥塞控制	有	无
应用场合	大量数据传输	少量数据传输
支持的应用层协议	Telnet、FTP、SMTP、HTTP	DNS、DHCP、TFTP、SNMP

Client/server socket: UDP

Properties of UDP:

> UDP - Client and Server example programs in Python | Pythontic.com (https://pythontic.com/modules/socket/udp-client-server-example)

- The UDP does not provide guaranteed(放心的，可靠的) delivery of message packets. If for some issue in a network if a packet is lost it could be lost **forever**.
- Since there is no guarantee of assured(有把握的) delivery of messages, UDP is considered an unreliable protocol.
- The underlying mechanisms(底层机制) that implement(实现) UDP involve **no connection-based communication**. There is no streaming of data between a UDP server or and an UDP Client.
- An UDP client can send "n" number of distinct packets to an UDP server and it could also receive "n" number of distinct packets as replies from the UDP server.
- Since UDP is connectionless protocol the overhead(负载，开销) involved in UDP is **less** compared to a connection based protocol like TCP.



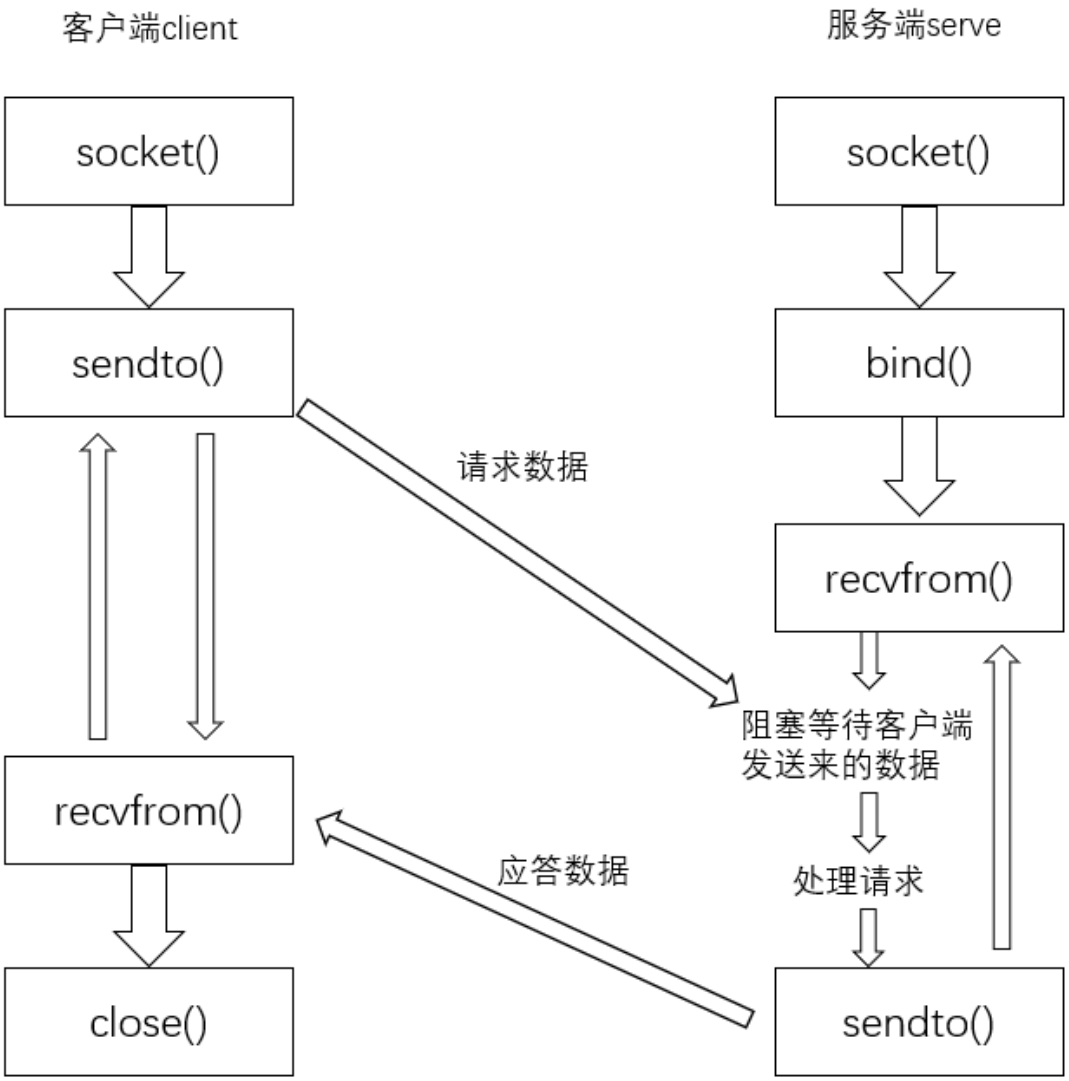
【Python】UDP/TCP_种花家 de 小红帽的博客-CSDN 博客 (https://blog.csdn.net/phoenixFlyzzz/article/details/129790340)

面向无连接型：无需对端是否存在，发送端可随时发送数据

特点：无连接，资源开销小，传输速度快，每个数据包最大是 64k，适用于广播应用

缺陷：传输数据不可靠，容易丢包；没有流量控制，需要接收方及时接收数据，否则会写满缓冲区

UDP 网络流程



1. 保证 UPD 服务端的正常启动，进入到 recvfrom() 模式，阻塞等到客户端发送数据
2. 开启 UDP 客户端，校准 IP 地址，通过 sendto() 模块进行数据发送
3. 当服务端接收到接收到客户端发送来的数据，进行数据处理，并将应答数据发送给客户端
4. 客户端接收到应答数据，可进行数据处理或重复发送数据，也可退出进程

UDP Serve 服务端

py

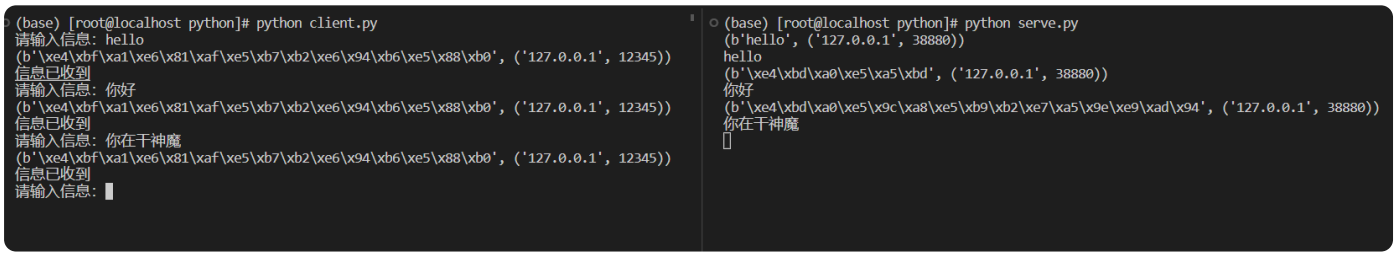
```
# coding=utf-8
from socket import *
```

```
udp_socket = socket(AF_INET, SOCK_DGRAM)

# 2. 绑定本地的相关信息，如果一个网络程序不绑定，系统会随机分配
#     ip地址和端口号，如果不指明ip，则表示本机的任何一个ip
#     如果不指明端口号，则每次启动都是随机生成端口号
local_addr = ('', 12345)
udp_socket.bind(local_addr)

while True:
    # 3. 阻塞等待接收对方发送的信息
    #     1024表示本次接收的最大字节数
    recv_data = udp_socket.recvfrom(1024)

    # 4. 显示接收到的数据，并解码为gbk
    print(recv_data)
    print(recv_data[0].decode('utf-8'))

    # 5. 发送应答信息
    # ip_addr = recv_data[1][0]
    # port = recv_data[1][1]
    addr = recv_data[1]
    data = '信息已收到'
    udp_socket.sendto(data.encode('utf-8'), addr)

udp_socket.close()
```

UDP Client 客户端

```py
import socket

# 1. 创建upd套接字
udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# 2. 准备服务端地址与端口号
#   127.0.0.1 代表自身ip地址，可向自身发送信息，也可指定ip地址发送信息
#   端口号随便填写一个未被占用的端口即可
#   Linux环境有65535个端口号，前1024个端口号是系统端口号，系统端口号不能直接使用
addr = ('127.0.0.1', 12345)

while True:
    # 3. 从键盘获取数据
    data = input('请输入信息：')

    # 4. 通过sendto()发送信息到指定进程中
    udp_socket.sendto(data.encode('utf-8'), addr)

    # 5. 通过recvfrom()阻塞等待获取应答数据
```

```
        print(recv_data)
        print(recv_data[0].decode('utf-8'))

udp_socket.close()
```

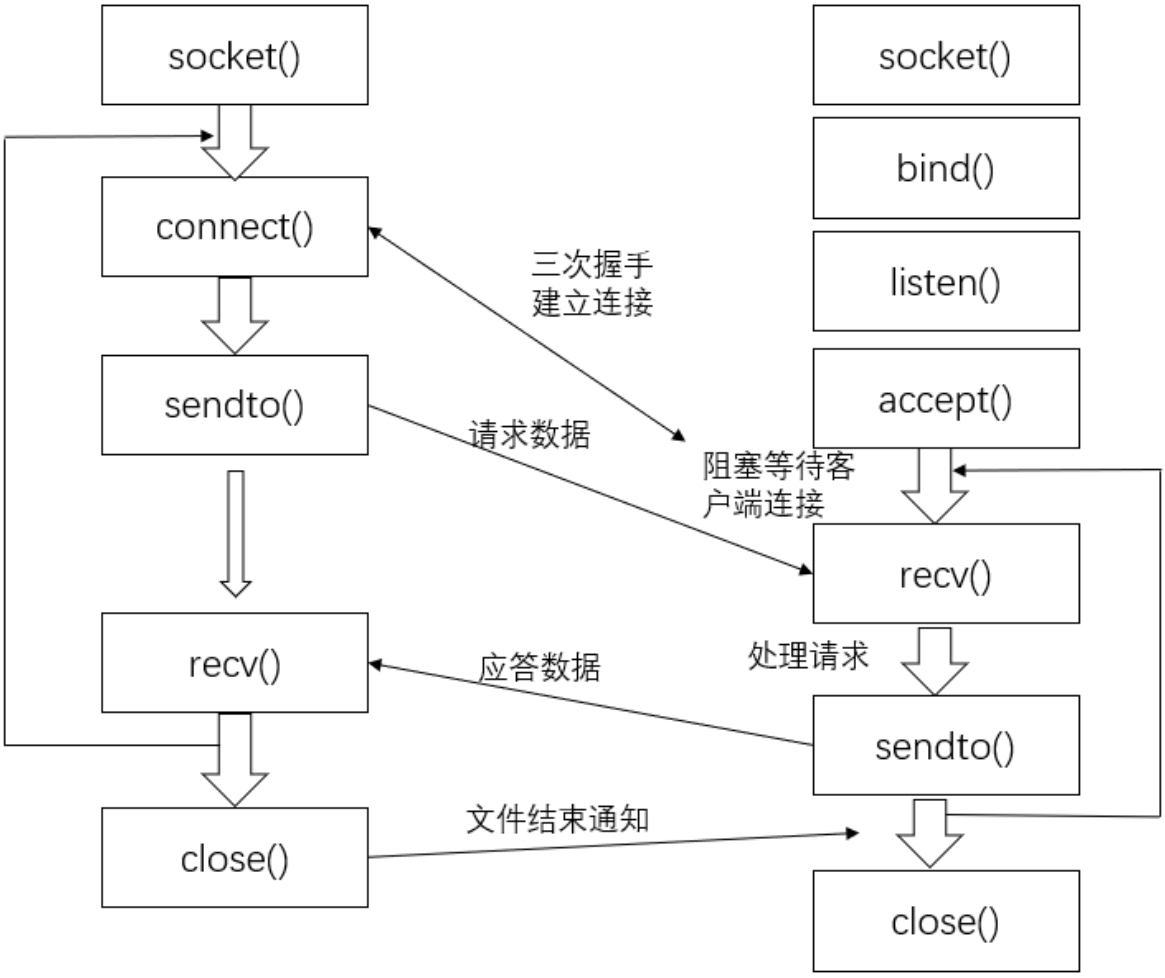


Client/server socket: TCP

[Python UDP/TCP_ de -CSDN]
(https://blog.csdn.net/phoenixFlyzzz/article/details/129790340)

-
-
-
- TCP
- UPD TCP
- TCP UDP
- TCP UPD

TCP



TCP Serve 服务端

```py
from socket import *

# 1. 创建tcp套接字
tcp_serve_socket = socket(AF_INET, SOCK_STREAM)

# 2. 设置socket选项，程序退出后，端口会自动释放
tcp_serve_socket.setsockopt(SOL_SOCKET, SO_REUSEADDR, True)

# 3. 本地信息，第二个为端口
addr = ('', 12345)

# 4. 绑定地址
tcp_serve_socket.bind(addr)

# 5. 设置监听
#     使用socket创建的套接字默认的属性是主动的，使用listen将其变为被动的
#     参数代表等待连接时间最多60秒
tcp_serve_socket.listen(60)

# 6. 如果有新的客户端来连接服务，就产生一个新的套接字，专门为这个客户端服务
```

```python
client_socket, client_addr = tcp_serve_socket.accept()

# 7. 阻塞等待客户端发送的信息
recv_data = client_socket.recv(1024)
print("接收到信息:", recv_data.decode('gbk'))

# 8. 发送应答信息
string = '已收到信息'
client_socket.send(string.encode('gbk'))

client_socket.close()
```

TCP Client 客户端

py

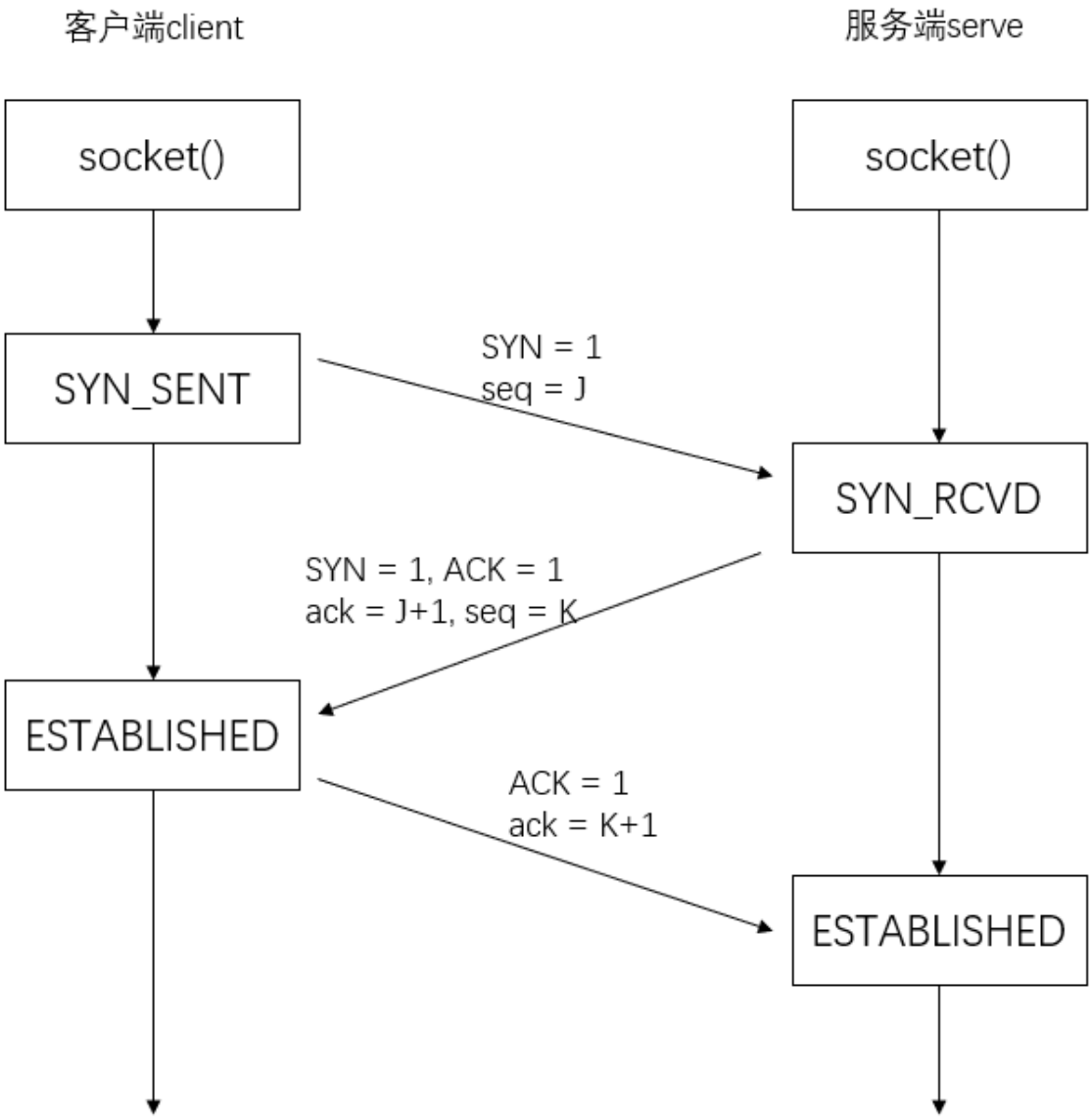
```python
import socket

# 1. 创建TCP的套接字
tcp_client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 2. 目标ip信息
ip = input('请输入服务端ip: ')
port = int(input('请输入服务端port: '))

# 3. 连接服务器
tcp_client_socket.connect((ip, port))

# 4. 提示用户输入数据
data = input('请输入要发送的信息: ')

# 5. 编码
tcp_client_socket.send(data.encode('gbk'))

# 6. 接收服务端的应答数据
recv_data = tcp_client_socket.recv(1024)
print('收到应答数据:', recv_data.decode('gbk'))

# 7. 关闭套接字
tcp_client_socket.close()
```

运行结果:

```
(base) [root@localhost python]# python client.py
请输入服务端ip: 127.0.0.1
请输入服务端port: 12345
请输入要发送的信息: hello
收到应答数据:  已收到信息
(base) [root@localhost python]#
```

```
(base) [root@localhost python]# python serve.py
接收到信息:  hello
(base) [root@localhost python]#
```

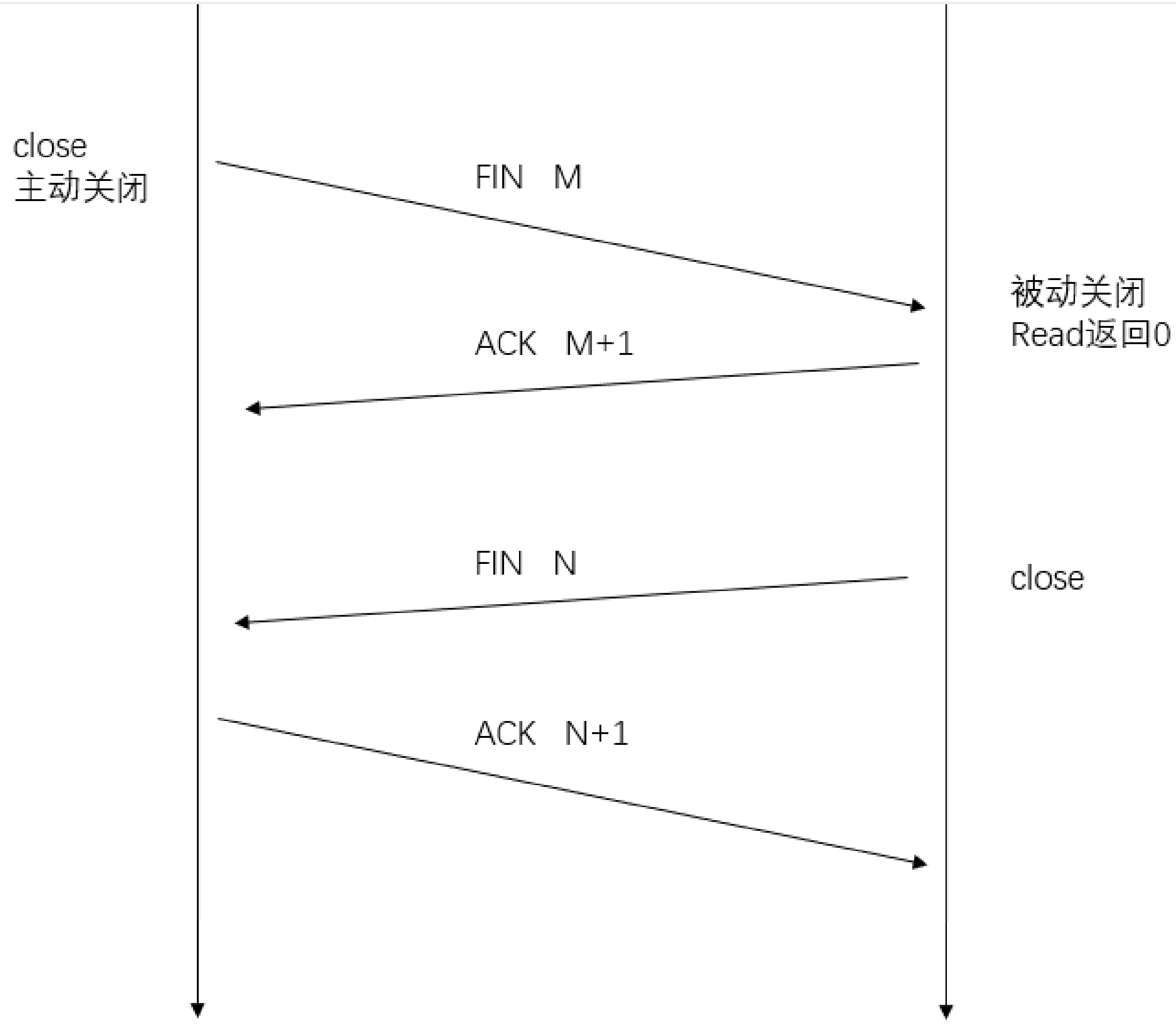
建立连接（三次握手）



SYN：连接请求 ACK：确认 FIN：关闭连接 seq：报文信号 ack：确认信号

1. 第一次握手：client标志位SYN置1，随机产生一个seq=J，并将该数据包发送给serve，client进入SYN_SENT状态，等待serve确认
2. 第二次握手：serve收到数据包后由标志位SYN=1知道client请求建立连接，serve将SYN和ACK都置1，ack（number）=J+1，+1是逻辑加一（加密），随机产生一个值seq=K，并将该数据包发送给client以确认连接请求，serve进入SYN_RECV状态
3. 第三次握手：client收到确认，检查ack是否为J+1（解密），如果正确则将标志位ACK置1，ack=K+1，并将该数据包发送给serve，serve检查ack是否为K+1，如果正确则建立连接成功，client和serve同时进入ESTABLISHED状态，完成三次握手，随后client和serve之间可以传输数据

断开连接（四次挥手）



1. 第一次挥手：client 发送一个 FIN，用来关闭 client 到 serve 的数据传送
2. 第二次挥手：serve 收到 FIN 后，发送一个 ACK 给 client，确认序号为收到序号 + 1，表示还有剩余数据未传送完
3. 第三次挥手：serve 发送一个 FIN，用来关闭 serve 到 client 的数据传送
4. 第四次挥手：client 收到 FIN 后，接着发送一个 ACK 给 serve，确认序号为收到信号 + 1

Q: TCP 传输中的 Segment 是什么?

A: 是该协议负责传输的数据单元的专用名词。在 TCP 协议中，分段（segment）是指将传输的数据分割成多个较小的部分，以便更有效地在网络上传输。每个分段包含在 TCP 首部中的控制信息，如端口号、序列号和确认号等，以及在数据字段中携带的有效负载部分。分段的大小可以根据不同的网络条件进行调整，以最大程度地减少重传或延迟。在接收方，TCP 会将收到的分段重新组合成完整的数据，以保证传输的准确性和完整性。

Last Updated: 6/15/2023, 1:37:06 PM

Outline

1. Transport-layer services
2. Multiplexing and demultiplexing
3. Connectionless transport: UDP
4. Principles of reliable data transfer
5. Connection-oriented transport: TCP
    - segment structure
    - reliable data transfer
    - flow control
    - connection management
6. Principles of congestion control
7. TCP congestion control

Transport-layer services

Transport vs. network layer

网络层是协议栈中的第三层，负责处理互联网上每个主机之间的路由。该层有两个主要的任务：

1. 选择最佳路径将数据包发送到目标地址；
2. 控制每个数据包在网络中的传输。网络层协议常用的是 IP（Internet Protocol），它定义了如何通过互联网把数据包从源主机传输到目标主机。

传输层是协议栈中的第四层，是端到端通信的主要协议。传输层实现了数据传输的可靠性，并采用流控制和拥塞控制等机制来协调系统中的数据流。该层的两个最常用的协议是 TCP（Transmission Control Protocol）和 UDP（User Datagram Protocol）。TCP 提供可靠的数据传输，而 UDP 则提供无连接、不可靠和基于报文的传输。

因此，网络层和传输层具有不同的目的和职责。网络层处理路由和数据包转发，而传输层负责端到端通信可靠性，数据流控制和拥塞控制等方面的任务。

Multiplexing and demultiplexing

> 计算机网络：多路复用（Multiplexing） VS 多路分解（Demultiplexing）_计算机 demultiplexing_SongXJ--的博客-CSDN 博客 (https://blog.csdn.net/SongXJ_01/article/details/106880461)

Multiplexing 多路复用

从源主机的不同套接字（socket）中收集数据块，并为每个数据块封装上首部信息（这将在多路分解时使用）从而生成报文段（segment），然后将报文段传递到网络层的工作称为多路复用。

- 在 TCP 或者 UDP 传输过程中，数据需要被不同程序识别，multiplexing 因此诞生
- 一般情况下 multiplexing 有以下几个 field 用于区分不同程序所需要的数据 source ip, source port, destination ip, destination port

Demultiplexing 多路分解

将传输层报文段中的数据交付到正确的套接字的工作称为多路分解。

- 通过检验上述 field, 传输协议（transport layer protocol）将不同数据包发送至正确的 socket port
- 0-1023 是被各种协议占用的端口，1024-65535 是一般程序可用的端口

Connectionless transport: UDP

UDP: User Datagram Protocol [RFC 768]

基础原理已经在 Chapter 2 @ Client/server socket: UDP 讲过，不再赘述。

UDP: segment header

Segmentation Explained with TCP and UDP Header (computernetworkingnotes.com) (https://www.computernetworkingnotes.com/ccna-study-guide/segmentation-explained-with-tcp-and-udp-header.html)

Segmentation

Segmentation is the process of dividing large data stream into smaller pieces. This functionality allows a host to send or receive a file of any size over the any size of network. For example, if network bandwidth is 1 Mbps and file size is 100 Mb, host can divide the file in 100 or more pieces. Once a piece becomes less or equal to the network bandwidth in size, it can be transferred easily. Destination host, upon(在…之后) receiving all pieces, joins them back to reproduce the original file.
If an application wants to use UDP to send its data, it can't give the data to UDP in actual size. It has to use its own mechanism(机制) to detect whether segmentation is required or not. And if segmentation is required, it has to do it on its own before giving data to UDP.

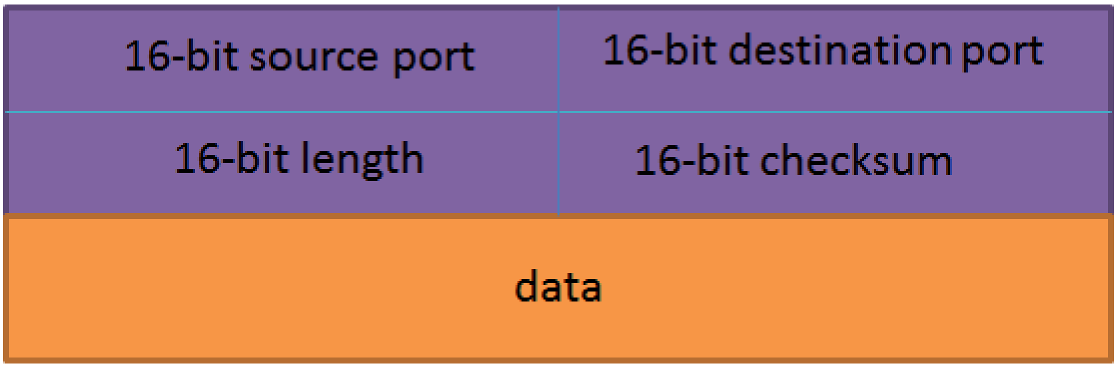
Packing data for transmission

1. The information that is required to send the segment at the correct destination.
2. The information that is required to support the protocol specific features.

How UDP Work with header

UDP neither provides any protocol specific service, nor adds any additional information in the header.

Following figure shows data with UDP header.



16-bit source port	16-bit destination port
16-bit length	16-bit checksum
data	

Field	Description
Source port	Port number of the application that is transmitting data from the source computer
Destination port	Port number of the application that will receive the data at destination.
Length	Denotes the length of the UDP header and the UDP data
Checksum	CRC of the complete segment
Data	Data which it received from the application

UDP checksum

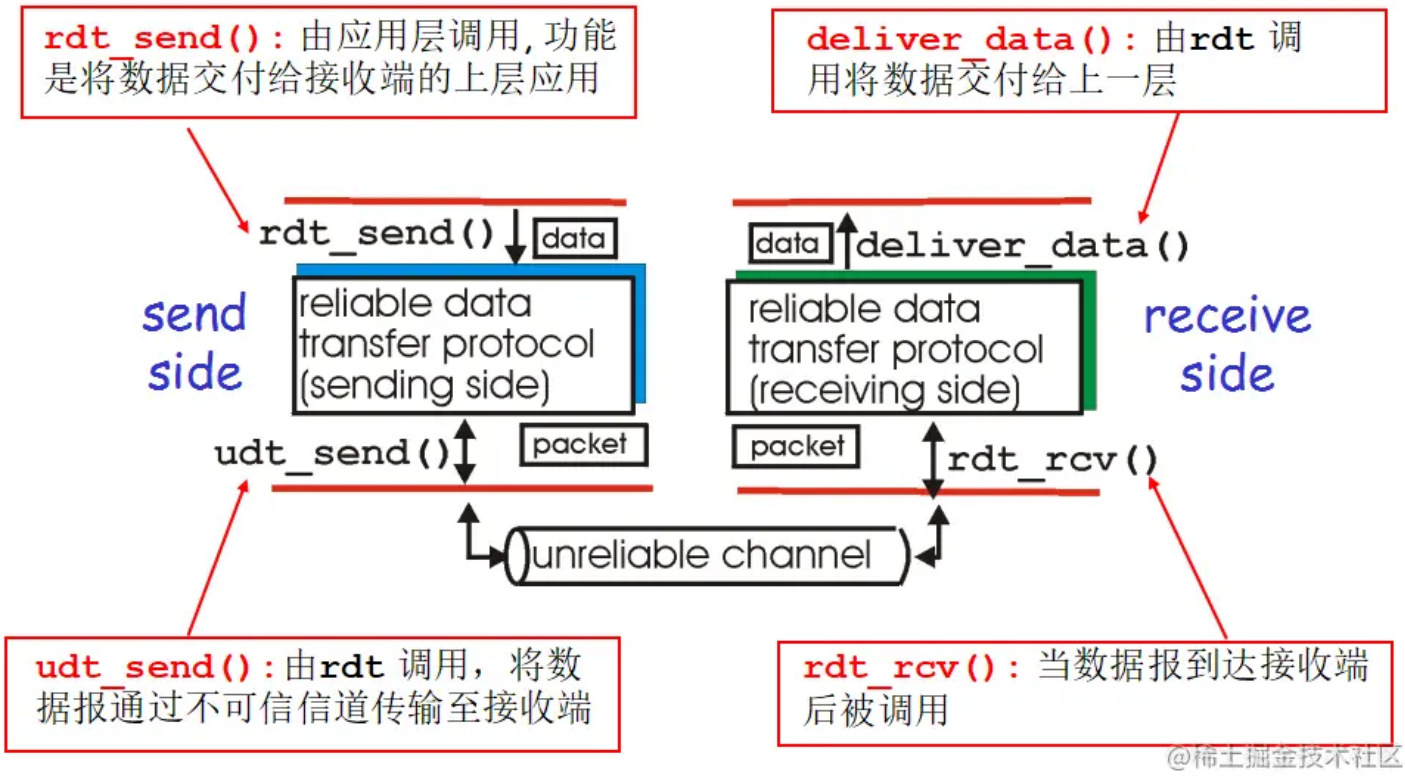
Goal: detect “errors” (e.g., flipped bits(比特位的翻转)) in transmitted segment

UDP（用户数据报协议）checksum是一种用于检测UDP数据包完整性的简单校验和。当UDP数据包被发送到网络时，发送端计算校验和并将其添加到UDP数据包中的checksum字段中。接收端通过再次计算校验和并将其与数据包的checksum字段进行比较来检测数据包是否在传输过程中出现任何错误或损坏。如果计算出的校验和不匹配，则数据包被认为是损坏的。UDP校验和是UDP协议的一项重要功能，可确保数据在传输过程中的完整性和正确性。

Principles of reliable data transfer

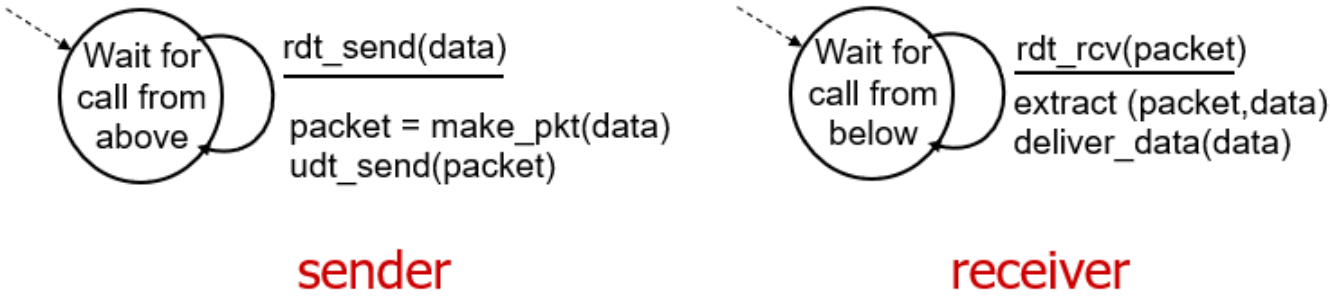
> **TIP**
>
> Reliable data transfer(rdt) 即可靠传输。

部分内容来自 网络协议 7:【传输层】可靠传输（rdt）的原理 - 掘金 (juejin.cn) (https://juejin.cn/post/7030066301062086670)，有删改



rdt 1.0

rdt 1.0 是基于理想情况下的协议，假设所有信道都是可靠的———没有比特位的翻转，没有数据包的丢失与超时。所以 rdt 1.0 的传输功能就是：发送方发送数据，接收方等着接受数据。



rdt 2.0

- rdt 2.0 在 rdt1.0 的基础上考虑了 **bit errors**，即，不可信信道中数据包中的 1 可能会变 0，0 可能会变成 1。rdt2.0 的任务是发现并修复这些 bit errors
- rdt 1.0 中接受者和发送者固定，rdt2.0 引入有限状态自动机 finite state machines (FSM) 来切换指定发送者和接受者

> 即只有有限种状态的，在特定条件下能切换状态的机器。比如刷卡就转的这种门禁机器。

rdt 2.0 增加了 3 种新机制来提升：

1. 通过 checksum 来错误校验

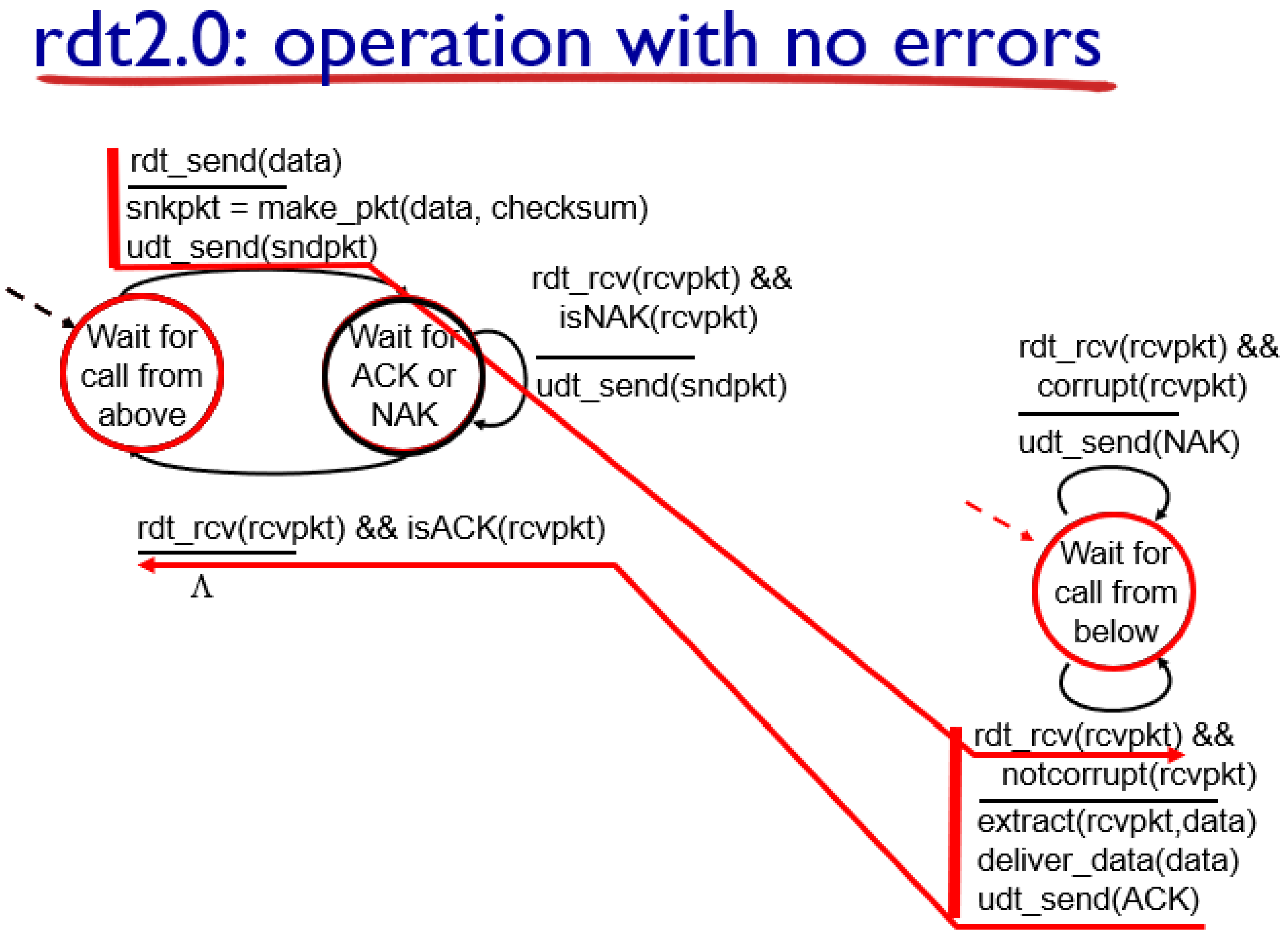
2. 接收者反馈接受正误信息（Acknowledgements）：

   - acknowledgements (ACKs): receiver explicitly(明确地) tells sender that pkt() received OK
   - negative acknowledgements (NAKs): receiver explicitly tells sender that pkt had errors

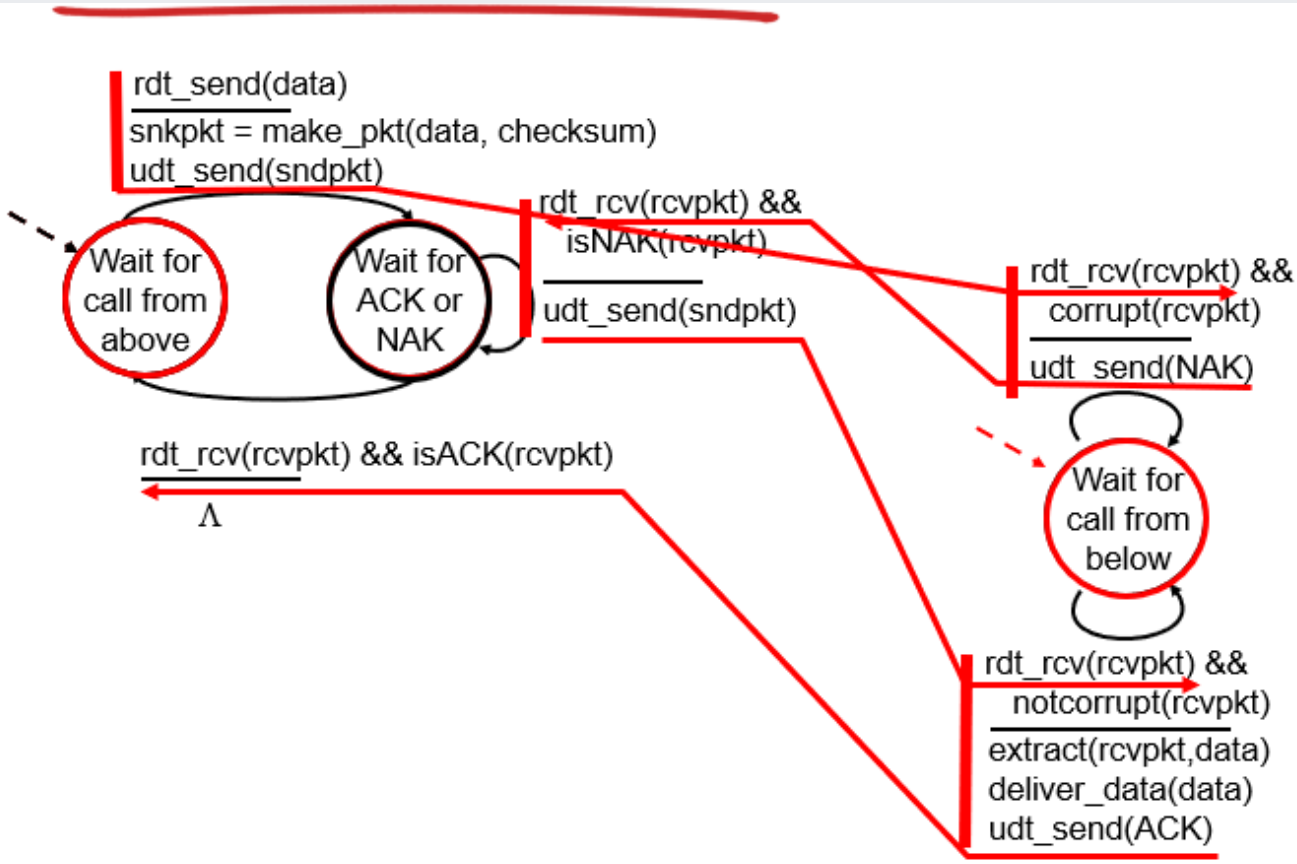
   > 这也叫做停等协议（**Stop and wait**）
   >
   > sender sends one packet, then waits for receiver response

3. 出错就重传

即，传输层对应用层的数据进行打包处理时，新增 checksum（校验和），从而接收端可以对其数据包进行检验，如果正确，返回 ACK，发送者继续发送下一个数据包；如果不正确，返回 NAK，发送者重传数据。



但如果客户端在接受后反馈信息 ACKs 在传输过程中出现问题，导致回答的内容在服务端看来并非 ACKs，服务端/发送端会重新回传内容。



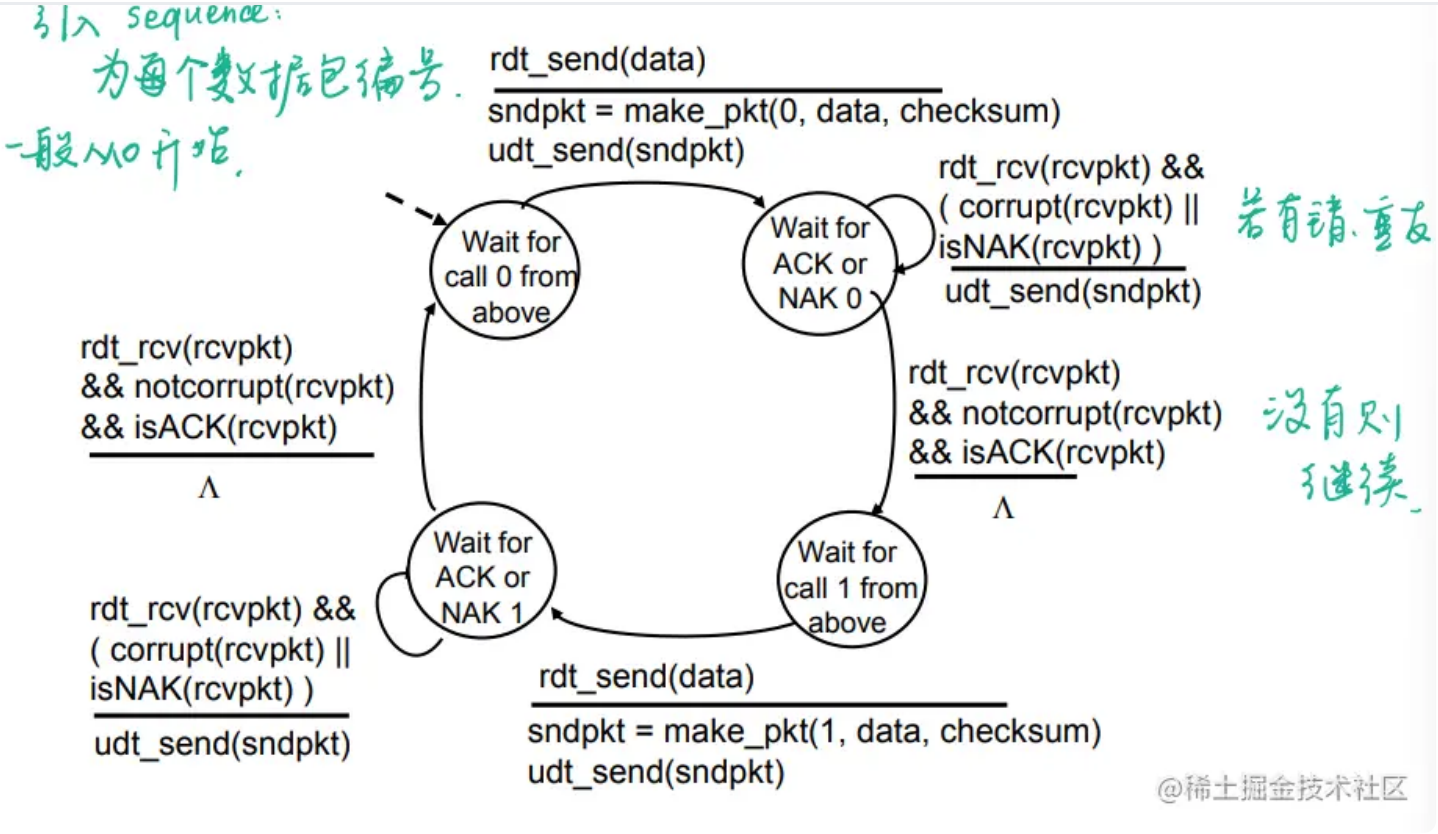
这也是 rdt 2.0 的一些大问题。不能只是简单地重发，可能重复了就麻烦了。因此 rdt 2.1 应运而生。

rdt 2.1

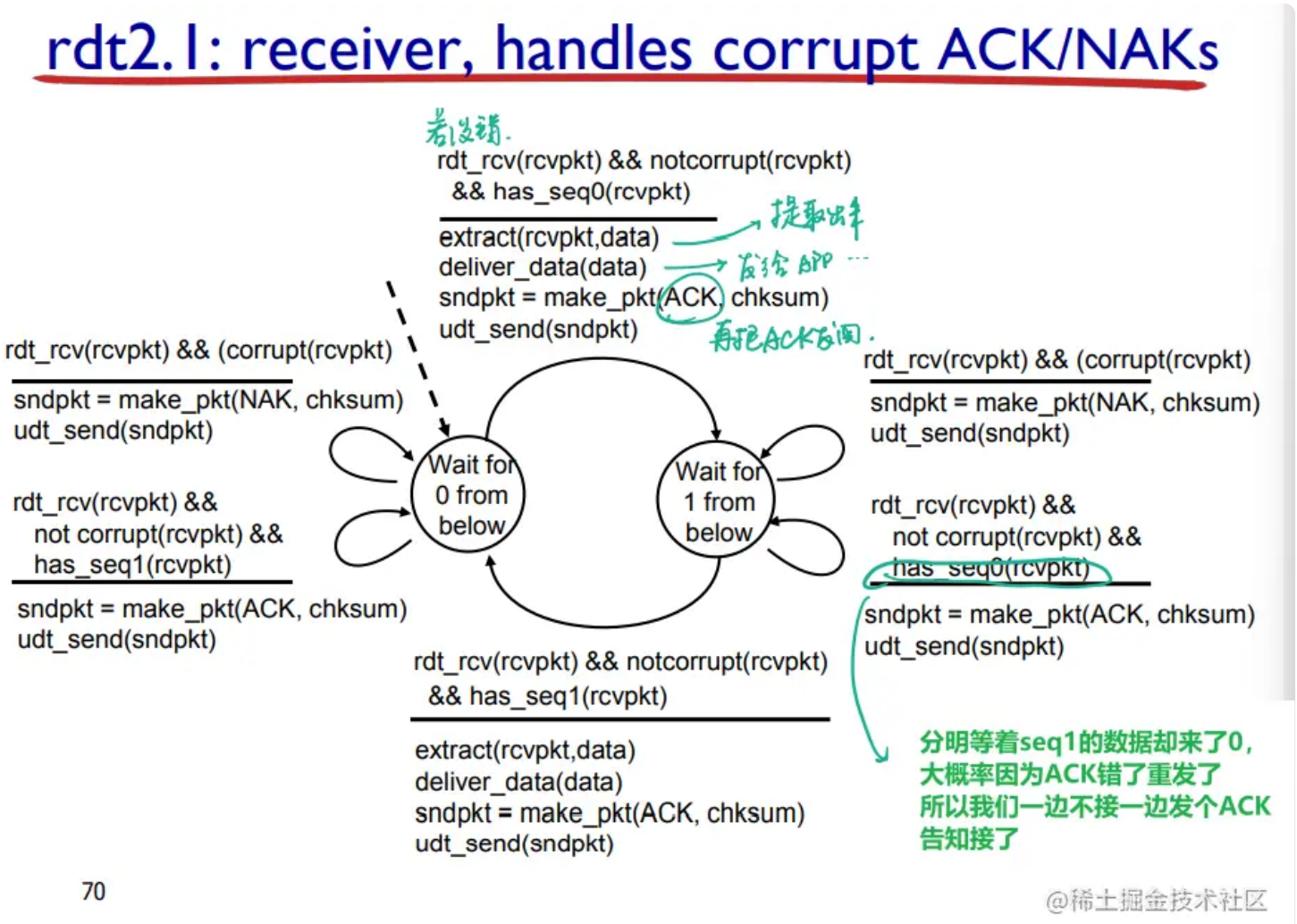
如何解决重复呢：

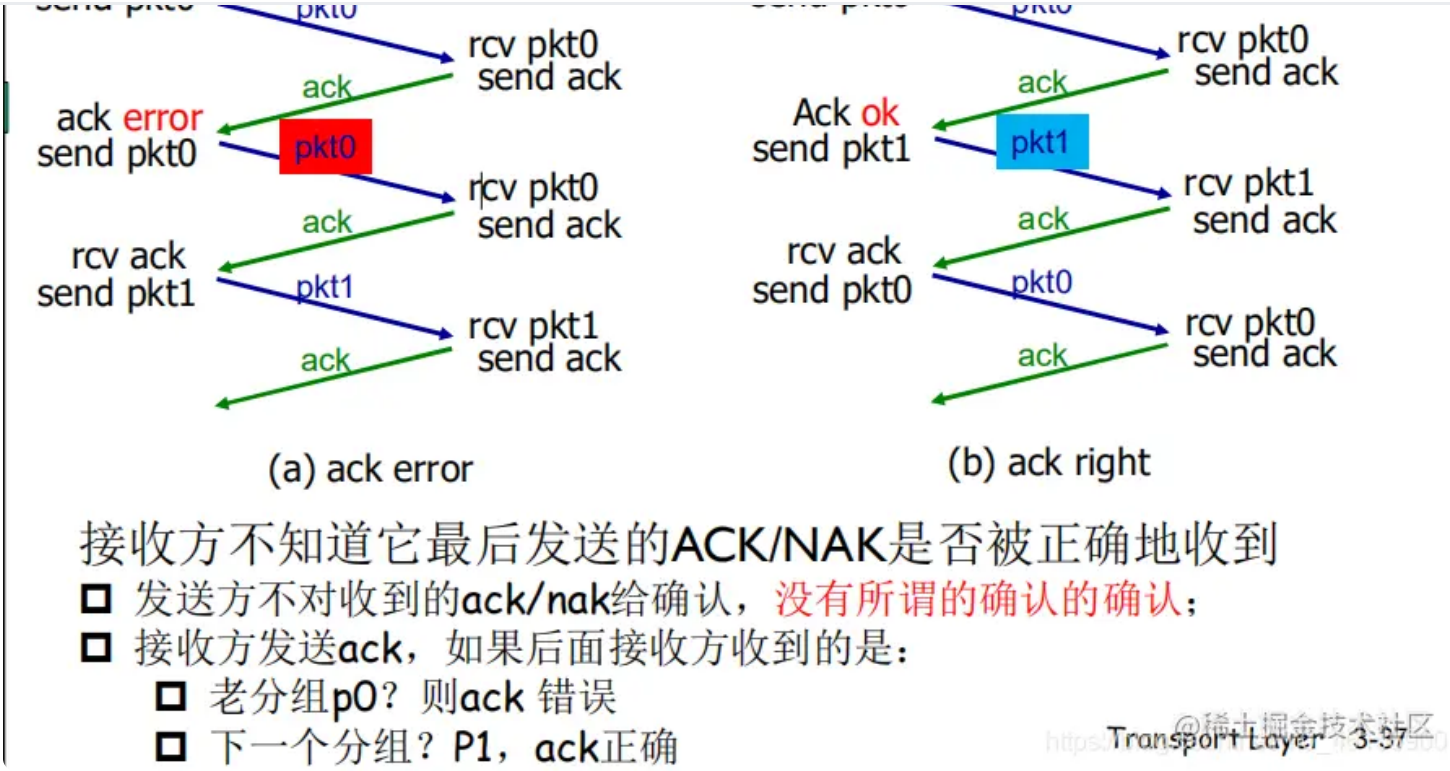
- 如果 ACK 或者 NAK 错了，corrupted(损坏) 了，还是重发
- 但是这回，发送方在打包数据包时添加了 0 或者 1 编号 sequence number (seq)（两个状态就够啦，一次只发送一个未经确认的分组）
- sender 就有 0 号数据包和 1 号 package 两种；receiver 也有了 2 种状态等待 0 号 package 和等待 1 号 package
- receiver 把序列重复的删了不接受即可（receiver 也不知道发送方是否正确收到了其 ACK/NAK）
- 需要“stop and wait”，发送方发送一个 package，然后等待接收方响应

对于 Sender：


引入 sequence:
为每个数据包编号.
一般从0开始.
rdt_send(data)
sndpkt = make_pkt(0, data, checksum)
udt_send(sndpkt)
Wait for call 0 from above
Wait for ACK or NAK 0
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isNAK(rcvpkt))
udt_send(sndpkt)
若有错,重发
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt)
Λ
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt)
Λ
没有则继续
Wait for ACK or NAK 1
Wait for call 1 from above
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isNAK(rcvpkt))
udt_send(sndpkt)
rdt_send(data)
sndpkt = make_pkt(1, data, checksum)
udt_send(sndpkt)
@稀土掘金技术社区

Receiver


rdt2.1: receiver, handles corrupt ACK/NAKs
若1没错.
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt)
extract(rcvpkt,data)
提取出来
deliver_data(data)
发给APP ...
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
再给ACK方向.
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)
sndpkt = make_pkt(NAK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)
sndpkt = make_pkt(NAK, chksum)
udt_send(sndpkt)
Wait for 0 from below
Wait for 1 from below
rdt_rcv(rcvpkt) && not corrupt(rcvpkt) && has_seq1(rcvpkt)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && not corrupt(rcvpkt) && has_seq0(rcvpkt)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq1(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
分明等着seq1的数据却来了0,
大概率因为ACK错了重发了
所以我们一边不接一边发个ACK
告知接了
70
@稀土掘金技术社区



(a) ack error

(b) ack right

接收方不知道它最后发送的ACK/NAK是否被正确地收到

- □ 发送方不对收到的ack/nak给确认，没有所谓的确认的确认；
- □ 接收方发送ack，如果后面接收方收到的是：
  - □ 老分组p0？则ack 错误
  - □ 下一个分组？P1，ack正确

Transport Layer 3-37

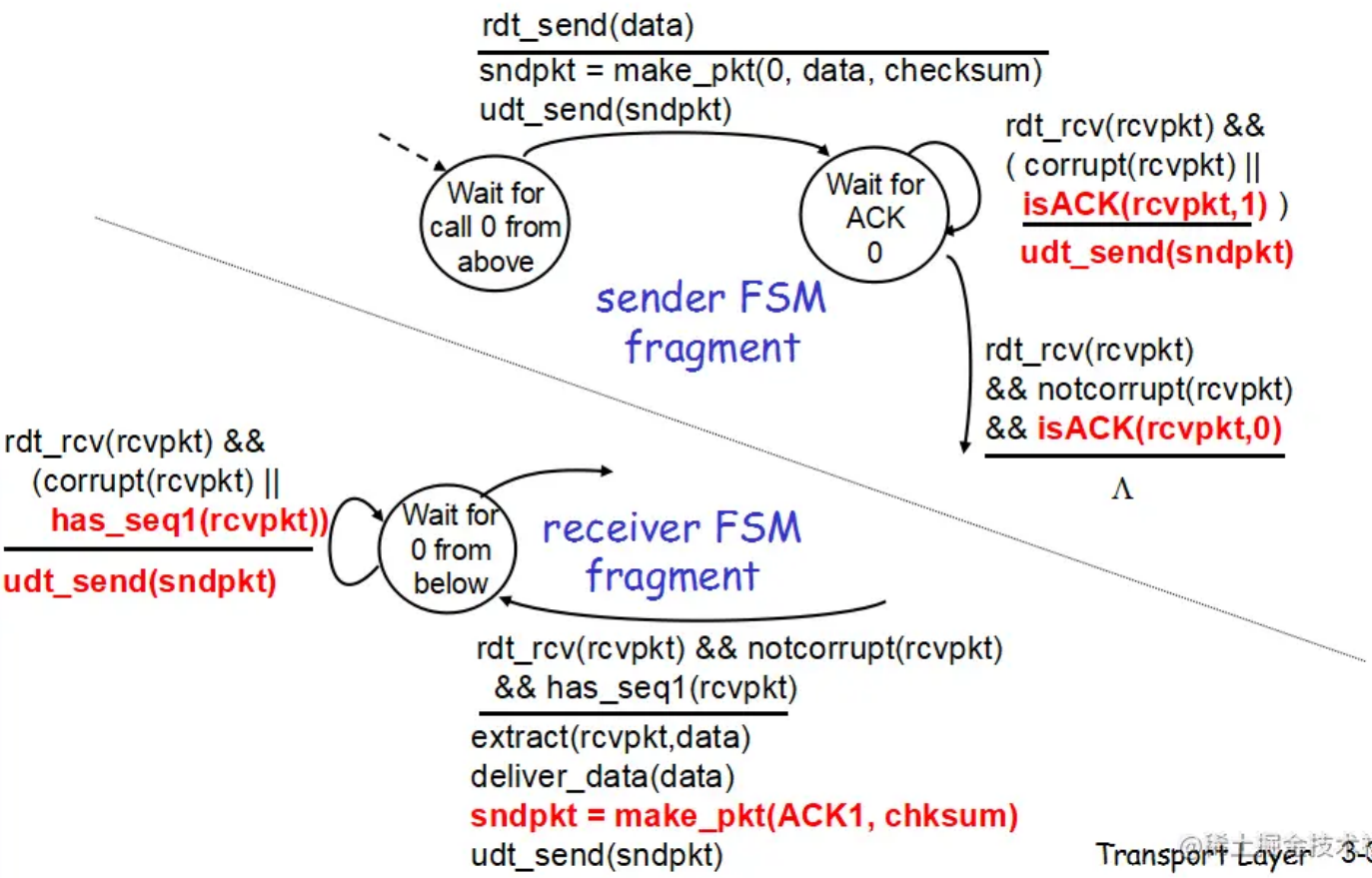
rdt 2.2

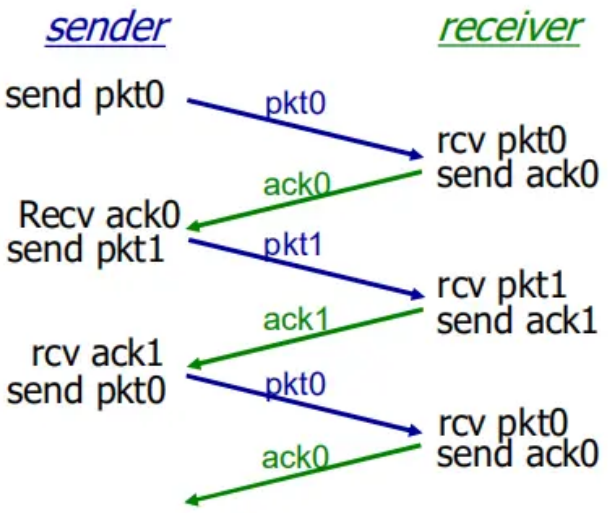
做了一点点微不足道的改进，再见啦 NAK，ACK 就够了

我们在 **ACK** 的信息上加上了顺序号（**sequence number**）：sender 发送 0 号数据包，如果接收方正确接收到 0 号，返回（ACK0），发送方接着发送 1 号数据包。如果接收方没有接收到 0 号数据包或出现错误，返回（ACK1），发送方重传 0 号数据包。

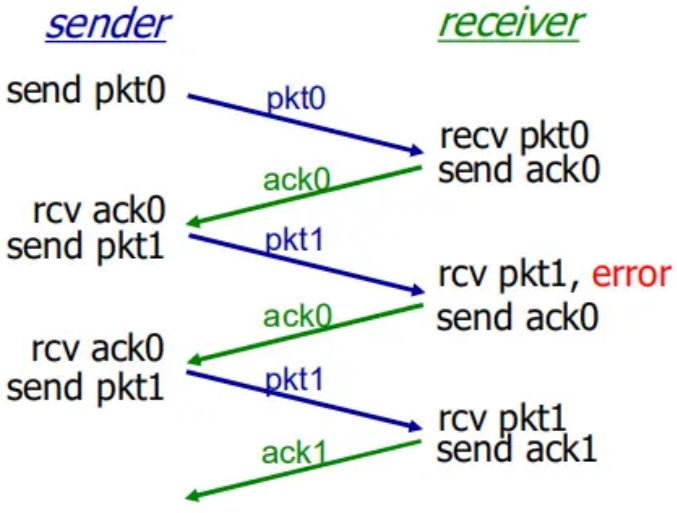
> **TIP**
>
> 相当于抽象成了“若不符合要求，重发”和“发新的”两个选项

rdt2.2: sender, receiver状态机片段

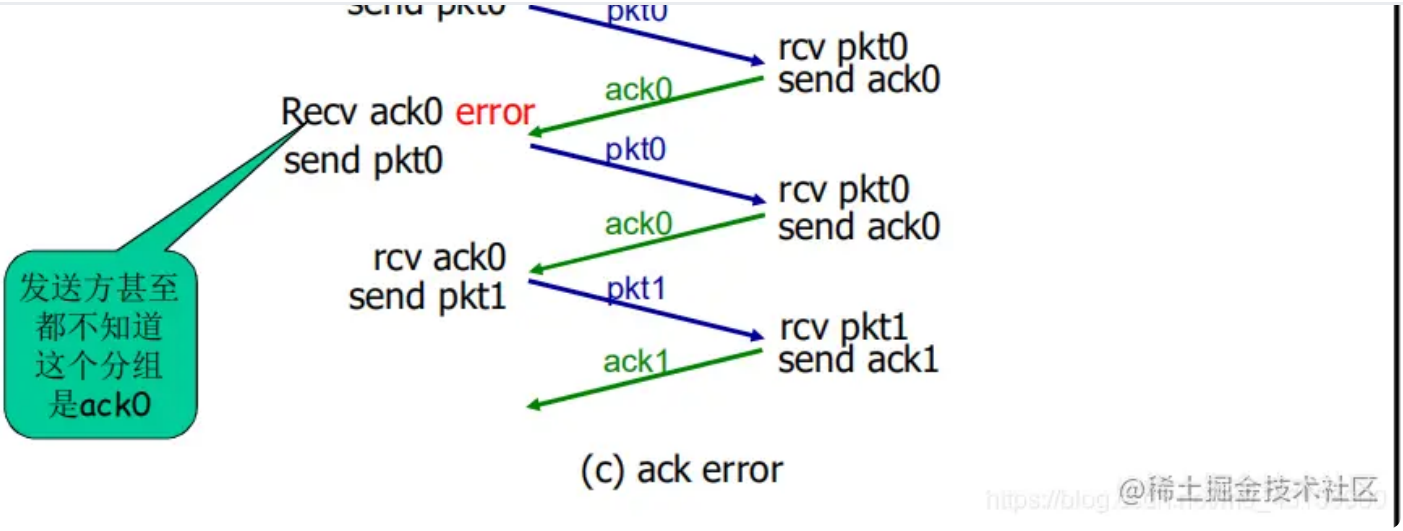




(a) No error（packet or ack error）



(b) packet error



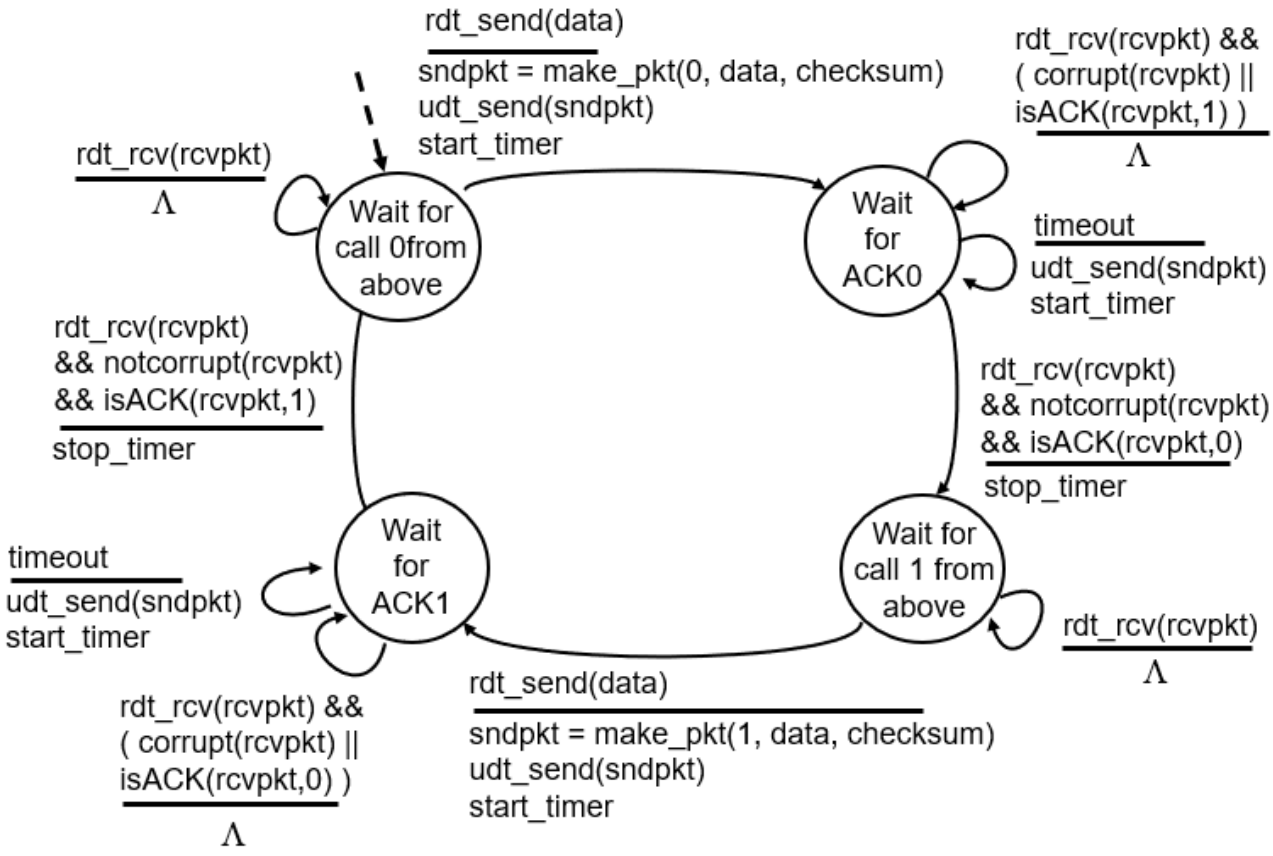
(c) ack error

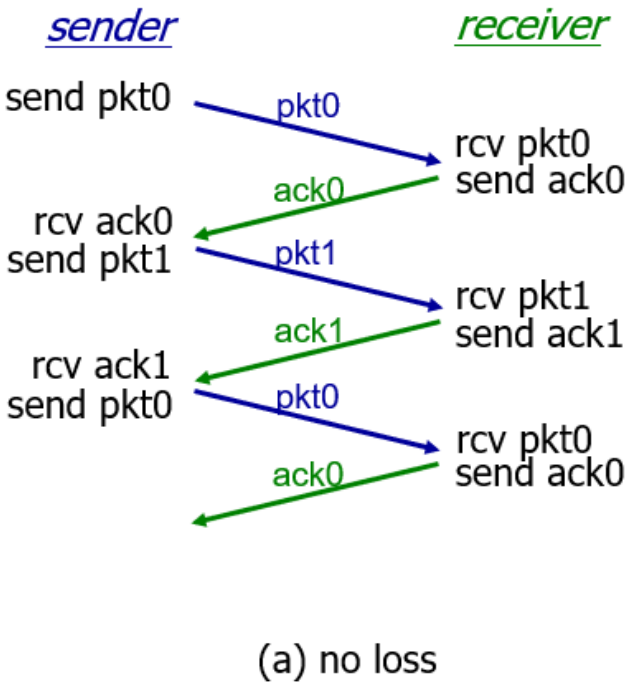
rdt 3.0

rdt2.2 之前的版本都重在处理数据包的 bit errors 情况，却没有考虑到数据包在传输过程中出现的数据包或者 **ACKs** 丢失问题，这样数据包丢失会使得网络处于拥塞状态

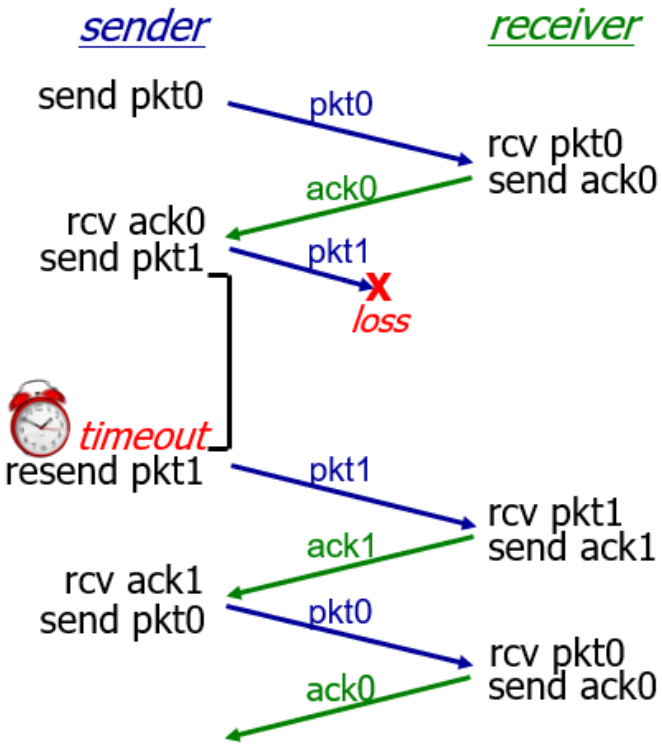
机制（解决方法）：在超过合理时间（reasonable amount of time，同 TCP）后重传

- 发送端超时重传：如果到时没有收到 ACK->重传
- 如果 package（或 ACK）只是被延迟了：
  - 重传将会导致数据重复，但利用序列号已经可以处理这个问题
  - 接收方必须指明被正确接收的序列号
- 需要一个倒计数定时器

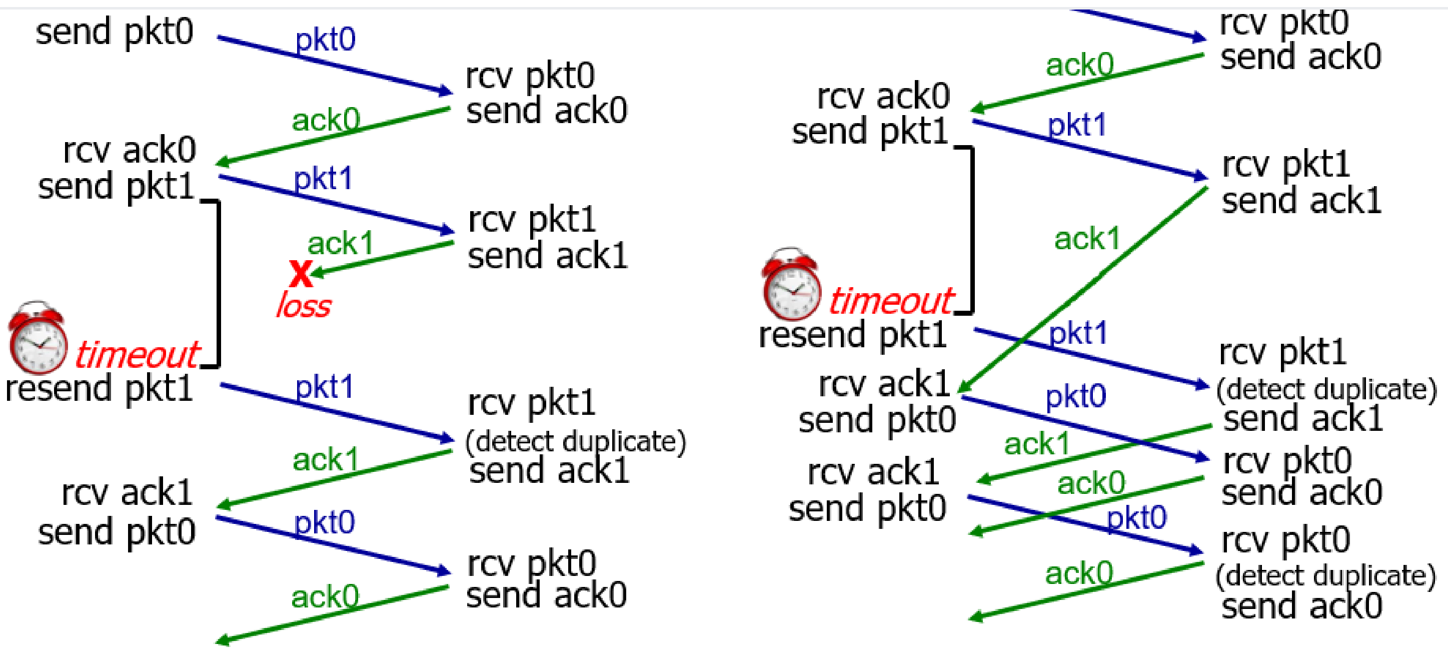


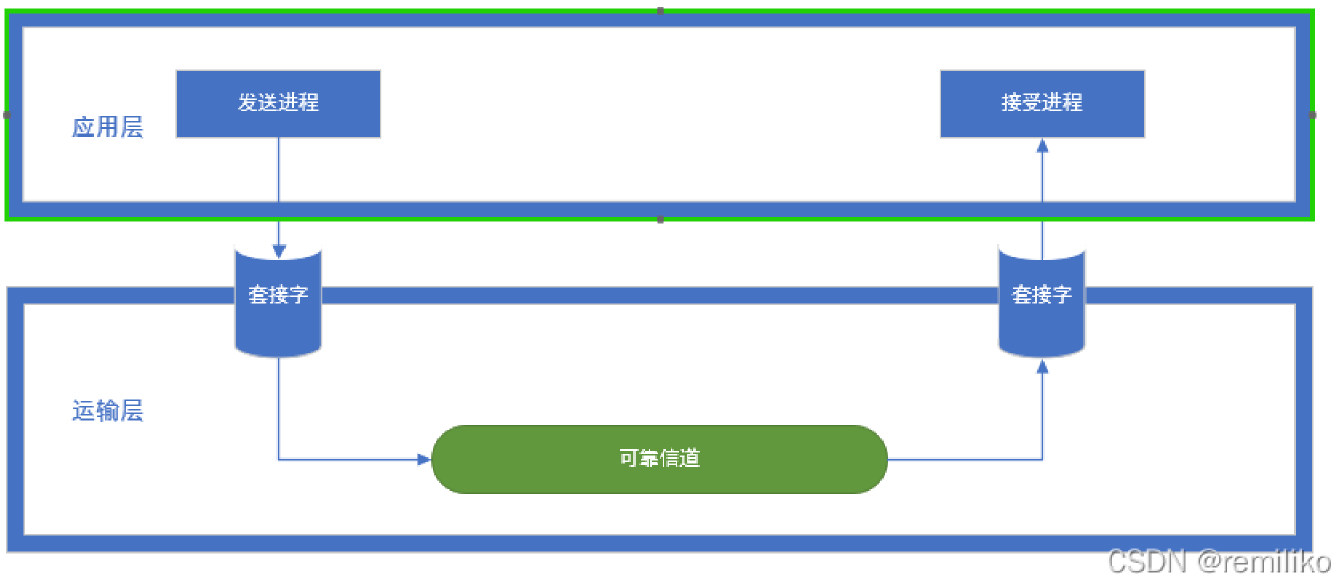


(a) no loss



(b) packet loss





(c) ACK loss

(d) premature timeout/ delayed ACK

过早超时(延迟的 ACK)也能够正常工作：但是效率较低，一半包和确认是重复的。

RDT 协议（可靠数据传输协议）_计算机网络 rdt_remiliko 的博客-CSDN 博客
(https://blog.csdn.net/m0_63657524/article/details/121916128)

计算机网络通过对网络进行分层设计，将一个庞大而复杂的系统，模块化层次化，（大致分层如图 2.1 所示）其中的每个层次为其上层提供特定的服务内容，并使用来自下层的特定功能，各个层次中明确了其需要实现的内容，但并不指明其中具体的实现方式。

运行在应用层中的客户端（服务端）应用程序进程通过套接字将数据推送到运输层。同样地，有服务端（客户端）进程通过套接字接受来自运输层的数据。对应用层而言，它所能看到的底层就是一条可靠的信道（如图 2.2 所示）。但是，对于现实中的数据传输，由于受到噪声干扰、网络拥堵等各种影响，难免会出现数据受损、丢包等事故，而 rdt 协议就是为了解决这样一个问题，而诞生的。

rdt 协议的实现

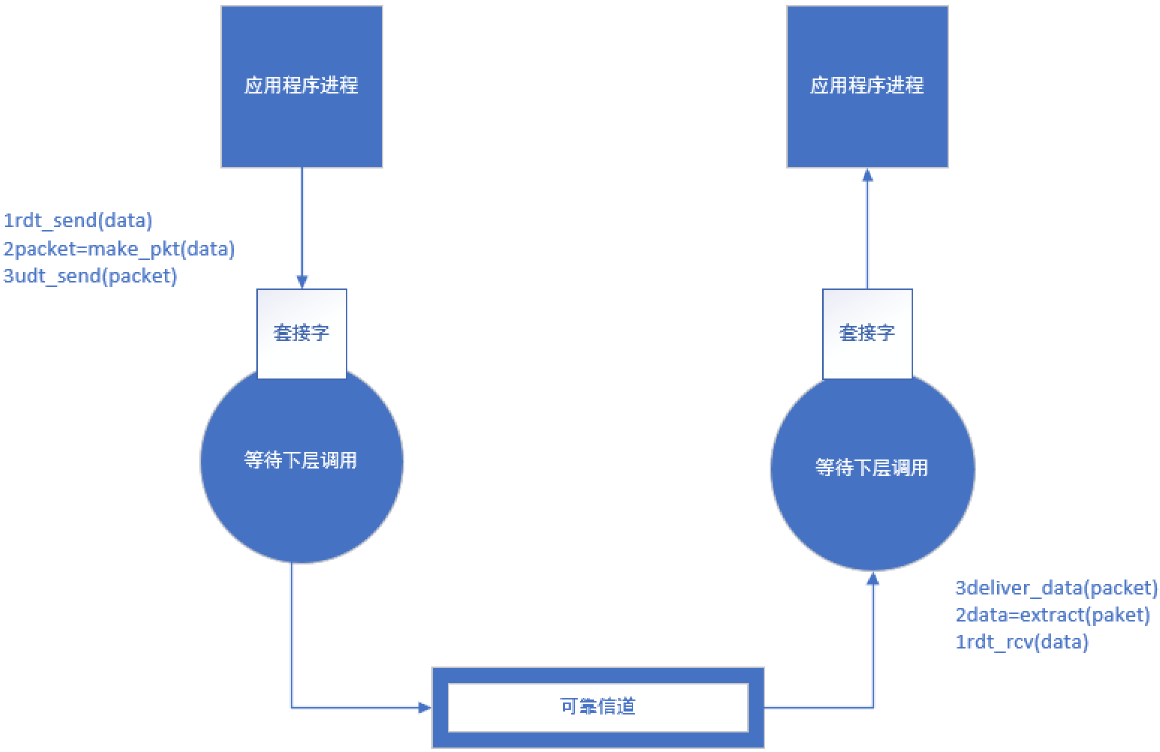
参考《计算机网络自顶向下》这本书，在这里我们从简单到复杂来探讨这个协议的实现过程（要注意的是，我们在此处探讨的 rdt 协议是建立在（stop-and-wait）停等协议上的。

首先，先将各个部分做好定义和命名初始化处理，为进一步探讨协议做好准备。

rdt 协议 1.0

在 rdt 1.0 中我们只专注于考虑如何实现核心功能，而不去考虑其他异常。因此我们在此处假定两个应用层之间存在着这么一条可靠信道，它可以保证从应用层的一侧到另一侧数据不丢失，因此此时的发送方和接收方只会有一个状态。

- 发送方：等待应用程序下发调用指令，发送数据
- 接收方：等待来自下层的调用指令，接受数据并缓存



数据流动情况分析

sender:

- 应用进程调用 rdt_send（data）方法，将数据推送至运输层
- 运输层调用 make_pkt 方法，将源自于应用程序的报文分组打包成报文段
- 运输层调用 udt_send 方法，将报文段推送至信道

receptor:

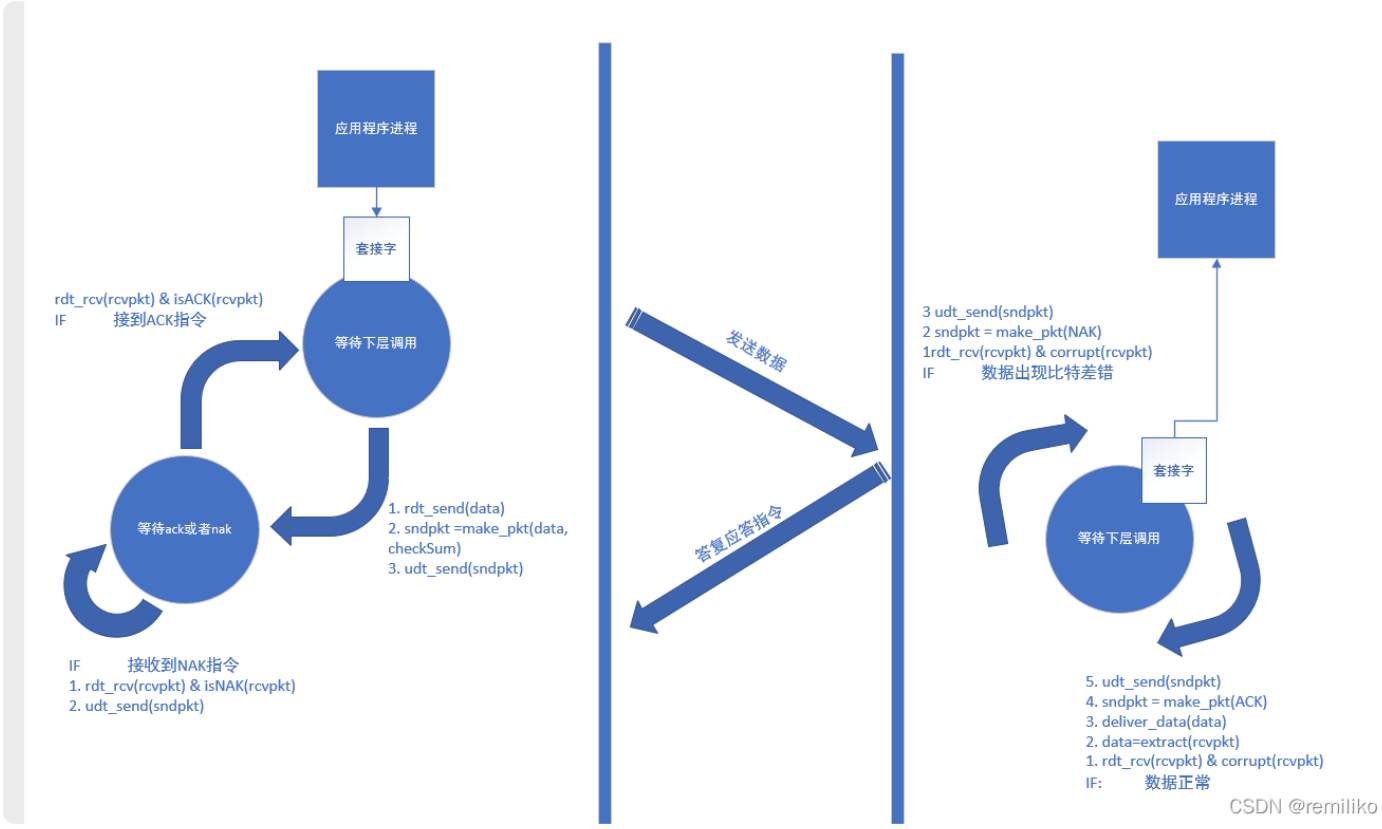
- 较低层（比运输层低的层次）执行 ret_rev 方法，将数据推送到运输层
- 运输层调用 extract，从报文段中提取出数据（可能涉及分组等细节实现，此处不作探讨）
- 运输层调用 deliver_data 方法将数据推送至应用层

此时，我们就可以简单地实现让数据从 sender 端传输到 receptor 端了，当然此时距离可靠数据传输还有一定的距离。因此，在 rdt2.0 及以后的版本我们开始考虑当异常事件发生的时候，如何保证数据的可靠性

rdt 协议 2.0

恢复）。基于这种重传机制的可靠数据传输协议称为 自动重传协议（Automatic Repeat reQuest，ARQ）

于是 Sender 方需要增加一个状态，等待 ACK 或 NAK，整个 rdt 2.0 执行流程如图所示：



数据流动情况分析

Sender 端：

- 应用层调用 rdt_send 方法，将数据推送至应用层
- 应用层调用 make_pdt 将数据打包成报文段，并在报文段中封装进一个校验码
- 应用层调用 udt_send 方法将打包完成的报文段推送至信道
- Sender 端此时状态迁移为等待 ACK 应答或者 NAK 应答状态

Receptor 端：

- 较低层通过 rdt_rcv 方法，将数据推送到运输层
- 运输层接收到报文段，对报文段数据进行校验处理，校验成功则执行第 4 步，校验失败则跳转到第 3 步
- 发送 NAK 指令，继续等待较低层的调用
- 发送 ACK 指令，继续等待较低层的调用

Sender 端：

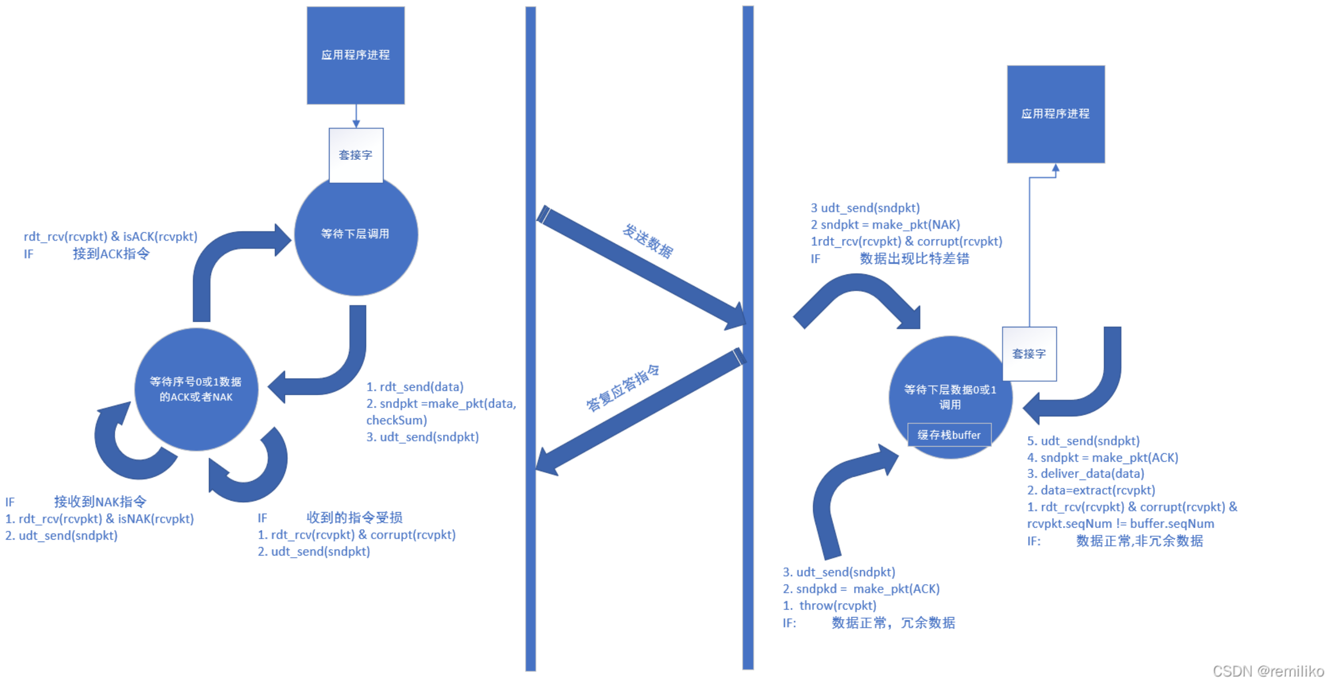
- 接收到 NAK 应答指令执行第 2 步，接收到 ACK 执行第 3 步
- 接收到 NAK，Sender 直接将打包好的数据再一次通过 udt_send 方法推送到信道，保持等待 ACK 或 NAK 指令状态
- 接受到 ACK，Sender 端不再阻塞，可以发送新的数据，状态迁移为等待上层调用状态

需要注意的是：由于在这里讨论的 rdt 协议采用了停等协议。因此，Sender 端在等待 ACK 和 NAK 答复指令的时候，处在阻塞的状态。

rdt 协议 2.1

（rdt 2.1 为什么不效仿 rdt 2.0 中，让 receptor 端去接受一个 ack 或 nak，这是因为可能出现一个比较有趣的问题，这个问题后续讨论）

数据流动如图所示：



数据流动情况分析

Sender 端：

- 应用层调用 rdt_send 方法，将数据推送至应用层
- 应用层调用 make_pdt 将数据打包成报文段，并在报文段中封装进一个校验码和一个值为 0 或 1 的序号
- 应用层调用 udt_send 方法将打包完成的报文段推送至信道
- Sender 端此时状态迁移为等待序号为 0 或 1 的报文段 ACK 应答或 NAK 应答状态

Receptor 端：

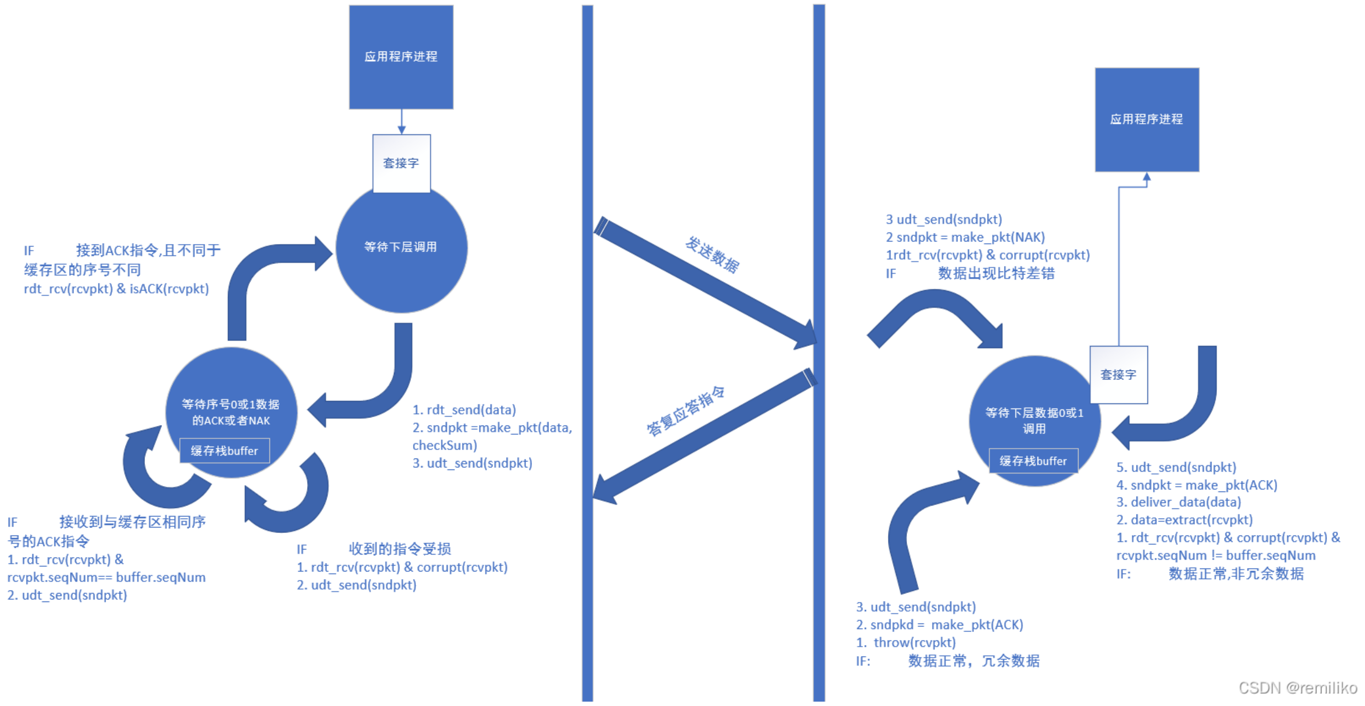
- 较低层通过 rdt_rcv 方法，将数据推送到运输层
- 运输层接收到报文段，对报文段数据进行校验处理，校验成功则执行第 4 步，校验失败则执行第 3 步
- 发送 NAK 指令，继续等待下层调用
- 检测数据序号，如果是冗余数据，直接丢弃数据，发送对缓存栈中数据的 ACK 指令，对于非冗余数据，则将数据置换到缓存栈之中，发送一个确认对本次数据的 ACK 指令，继续等待来自下层的调用

Sender 端：

- 接收到应答指令后，进行数据校验处理，如果数据校验错误，直接重传上次数据，如果数据校验正确，则执行第 2 步
- 判断接受到的应答指令，如果指令为 ACK 执行第 3 步，如果指令为 NAK 执行第 4 步
- 接收到 ACK 指令，Sender 端不再阻塞，可以发送新的数据 1 或 0，状态迁移到 等待来自上层调用
- 接收到 NAK 指令，重传上次数据，状态迁移到：等待 ACK 或 NAK 状态

rdt 协议 2.2

实际上 rdt 2.2 并没有做出更多变更，它只是在 rdt2.1 基础上进行了一次逻辑优化，因为有了序号的存在，receptor 端不在需要用 NAK 指令去表示收到的数据产生了比特错误。取而代之是，对于比特受损的数据，receptor 端直接丢弃，并发送一个对缓存区数据的确认 ACK（一开始缓存区为空也不要紧，由于序列是 010101 交替的序列，只要发送一个与当前 ACK 不同的序号即可），而此时，Sender 端也需要维护一个缓存，用于记录上一次发送的数据，当接受到的 ACK 与缓存序列号相同，那么就表示发送的数据发生了比特差错，此时重新发送一次缓存区中的数据即可。



数据流动情况分析

Sender 端:

- 应用层调用 rdt_send 方法，将数据推送至应用层
- 应用层调用 make_pdt 将数据打包成报文段，并在报文段中封装进一个 校验码，和一个值为 0 或 1 的序号，并将其存储至缓存区
- 应用层调用 udt_send 方法将打包完成的报文段推送至信道
- Sender 端 此时状态迁移为 等待序号为 0 或 1 的报文段 的 ACK 或 NAK 应答状态

Repetor 端:

- 较低层通过 rdt_rcv 方法，将数据推送到运输层
- 运输层接收到报文段，对报文段数据进行校验处理，校验成功则执行第 4 步，校验失败则执行第 3 步
- 发送缓存区数据序号的确认 ACK 指令，保持等待下层调用的状态
- 检测本次数据序号，如果是冗余数据，直接丢弃数据，发送缓存栈中数据序号的 ACK 指令，对于非冗余数据，将数据置换到缓存区之中，并发送一个对本次数据序号的确认 ACK 指令

Sender 端:

- 接收到应答指令后进行数据校验处理，如果数据校验错误，直接重新发送上次数据，如果正确则执行第 2 步
- 判断接受到的应答指令，如果指令为 ACK 中的序号等于缓存区的序号，执行第 3 步，如果 ACK 指令序号不等于缓存区序号则执行第 4 步
- 重新发送上次数据，状态迁移到 等待 ACK 应答指令
- Sender 端不再阻塞，可以发送新的序号为 1 或者 0 的数据，状态迁移到 等待来自上层调用

> ## rdt协议3.0
>
> 在处理好了比特差错的问题之后，需要考虑的就是来自底层信道传输的另一个问题，丢包异常（即发送方或者接受方由于网络阻塞等状况，并没有收到来自于对方的应答数据，在现实中丢包现象是非常常见的），因此，我们可以加入一个定时器来处理丢包现象，当发送一个报文段的时候，就开启一个定时器，在定时器结束期间，如果没有收到对应数据的应答报文，则重传数据。
>
> 数据流动分析
>
> ### Sender端：
>
> - 应用层调用rdt_send方法，将数据推送至应用层
> - 应用层调用make_pdt将数据打包成报文段，并在报文段中封装进一个校验码，和一个值为0或1的序号，并将其存储至缓存区
> - 应用层调用udt_send方法将打包完成的报文段推送至信道，并启动一个定时器事件
> - Sender端此时状态迁移为等待序号为0或1报文段的ACK应答状态
> - 倘若在定时器等待时间内，没有收到响应，则重新执行第3步
>
> ### Receptor端：
>
> - 较低层通过rdt_rcv方法，将数据推送到运输层
> - 运输层接收到报文段，对报文段数据进行校验处理，校验成功则执行第4步，校验失败则执行第3步
> - 发送缓存区数据序号的确认ACK指令，同时开启一个定时器，保持等待下层调用的状态
> - 检测数据序号，如果是冗余数据，直接丢弃数据，发送缓存区中数据序号的ACK指令，对于非冗余数据，将数据置换到缓存区中，发送一个对该数据序号的确认ACK指令，同时开启一个定时器
> - 倘若在定时器等待时间内，没有收到响应，则重新执行第3或第4步
>
> ### Sender端
>
> - 接收到应答指令后进行数据校验处理，如果数据校验错误，直接重新发送上次数据，如果正确则执行步第2步
> - 判断接受到的应答指令，如果指令为ACK中的序号等于缓存区的序号，执行第3步，如果ACK指令序号不等于缓存区序号则执行4
> - 重新发送上次数据，状态迁移到等待ACK，同时开启一个定时器
> - Sender端不再阻塞，可以发送新的序号值为1或0的数据，状态迁移到等待来自上层调用
> - 倘若在定时器等待时间内，没有收到响应，则重新执行第3或第4步
>
> 总结
>
> 至此，在rdt 3.0我们已经得到了一个可靠的数据传输协议，当然它只是一个抽象的雏形，建立于停等协议之上，还需要不少细致的优化，让其更好的切合现实，比如如何脱离停等协议实现更高的传输率，以及拥塞控制等等。文章内的方法为了保证准确性，表达式大多参照于《计算机网络自顶向下方法》书籍中的内容，但是为了简化一下表达，也做出了一点变更。

Pipelined protocols

Pipelined protocols(管道协议/窗口协议): sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

- range of sequence numbers must be increased
- buffering at sender and/or receiver

Two generic forms of pipelined protocols: go-Back-N(返回N个), selective repeat(选择重传)

Go-back-N(□□ N □)

- sender can have up to N unACKed packets in pipeline
- receiver only sends cumulative(□□□) ack: doesn't ack packet if there's a gap(□□□□)
- sender has timer for oldest unACKed packet: when timer expires, retransmit all unACKed packets

Go-Back-N Protocol | Baeldung on Computer Science (https://www.baeldung.com/cs/networking-go-back-n-protocol)

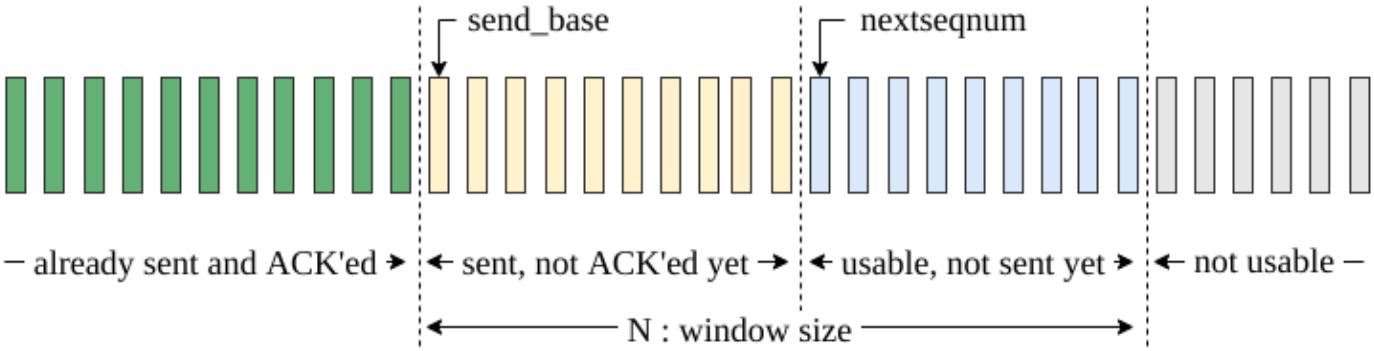
Go-Back-N and Selective(□□□□) Repeat protocols are fundamental(□□□) sliding window protocols(□□□□) that help us better understand the key idea behind reliable data transfer in the transport layer of computer networks.

The sliding window (pipelined) protocols achieve utilization(□□) of network bandwidth by not requiring the sender to wait for an acknowledgment before sending another frame.

In Go-Back-N, the sender controls the flow of packets, meaning we've got a simple and dummy receiver. Therefore, we'll start by discussing how the server handles(□□) data packets first.

Sender

The sender has a sequence of frames to send. We assume a window size of N. Furthermore, there exist two pointers to keep track of send base (send_base) and the next packet to send (nextseqnum).



First of all, the sender starts by sending the first frame. Initially, send_base = 0 and nextseqnum = 0. While there are more packets to send and the nextseqnum is smaller than the send_base + N; the sender sends the packet pointed by the nextseqnum pointer and then increments(□□) the nextseqnum.

Meanwhile, the send_base is incremented after receiving acknowledgment packets from the receiver. The reception of duplicate ACK messages does not trigger any mechanism.

There is a single timer for the whole sending window, which measures the timeout for the packet at the send_base. Therefore, if a timeout occurs, the sender restarts the timer and re-transmits all the packets in the sending window starting from send_base.

Receiver

There is no receiver buffer(□□□); out of order packets are simply discarded(□□□). Similarly, corrupted packets are also silently discarded.
It always sends the acknowledgment for the last in-order packet received upon reception of a new packet (successfully or unsuccessfully). As a result, it will generate duplicate acknowledgment messages if something goes wrong.
The Go-Back-N protocol adopts the use of cumulative acknowledgments. That is, receiving acknowledgment for frame n means the frames n-1, n-2, and so on are acknowledged as well. We denote such acknowledgments as ACK n.

Selective Repeat(□□□□)

- sender can have up to N unACKed packets in pipeline
- receiver sends **individual ack** for each packet
- sender maintains timer for each unACKed packet: when timer expires, retransmit only that unACKed packet

Selective Repeat Protocol | Baeldung on Computer Science (https://www.baeldung.com/cs/selective-repeat-protocol)
Selective Repeat Protocol (SRP) is a type of error control protocol (https://en.wikipedia.org/wiki/Error_detection_and_correction) we use in computer networks to ensure the reliable delivery of data packets. Additionally, **we use it in conjunction(□□) with the** Transmission Control Protocol (TCP) (https://www.baeldung.com/cs/udp-vs-tcp) **to ensure that the receiver receives data transmitted over the network without errors.**
In the SRP, the sender divides the data into packets and sends them to the receiver. Furthermore(□□), the receiver sends an acknowledgment (ACK) (https://www.baeldung.com/cs/tcp-protocol-syn-ack) for each packet received successfully. If the sender doesn't receive an ACK for a particular packet, it retransmits only that packet instead of the entire set of packets.
The SRP uses a window-based(□□□□□) flow control mechanism(□□□□□□) (https://www.baeldung.com/cs/tcp-flow-control-vs-congestion-control) to ensure the sender doesn't overwhelm(□□□□□) the receiver with too many packets. Additionally, **the sender and receiver maintain(□□) a window of packets.** Based on the window size, the sender sends packets and waits for a specific amount of time for acknowledgment from the receiver.
The receiver, in turn, maintains a window of packets that contains the frame number(□□□) it's receiving from the sender. If a frame is lost during transmission, the receiver sends the sender a negative acknowledgment attacking the frame number.

Steps
Now let's discuss the steps involved in the SRP.

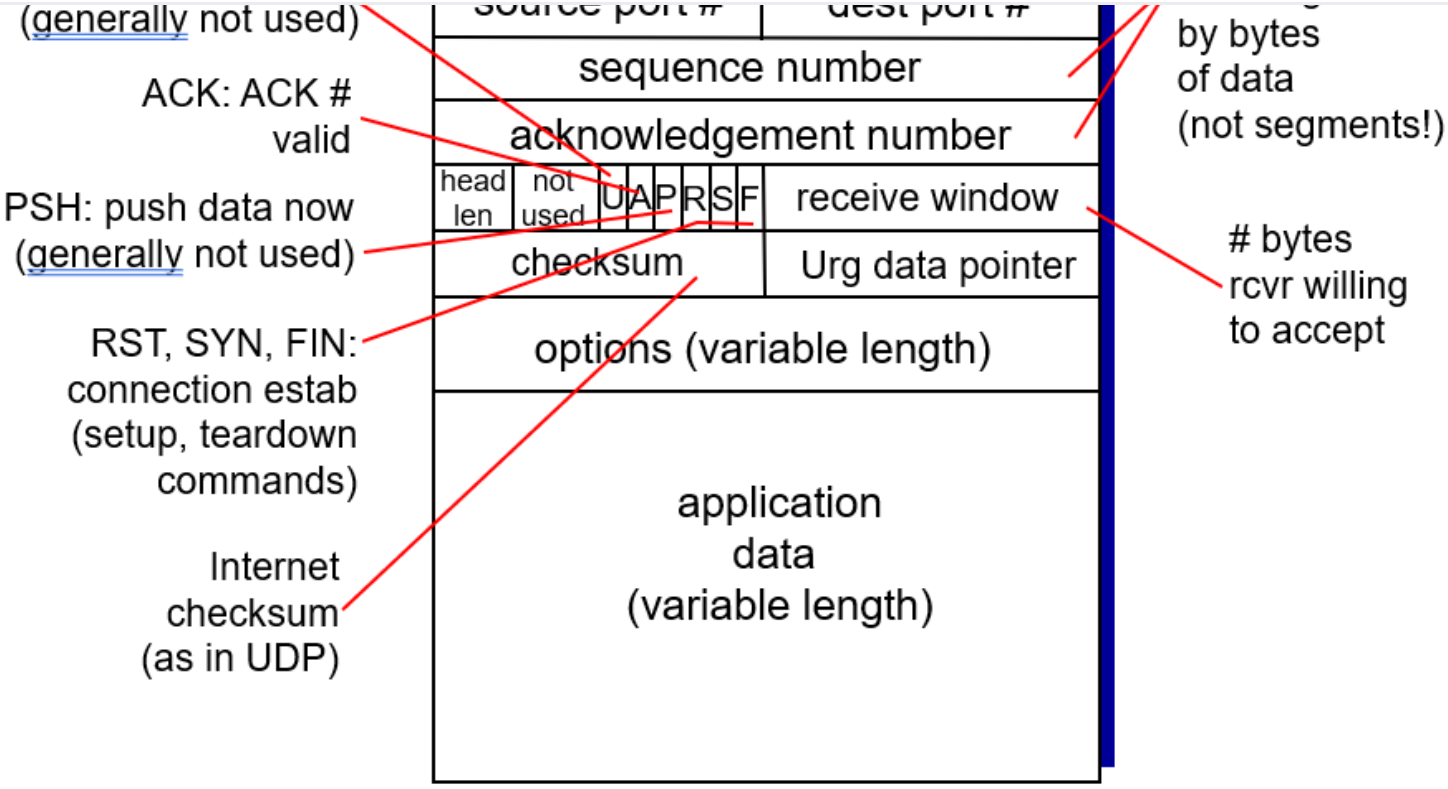
each packet. The numbering of packets plays a crucial role in the SRP.
The next step is to send the packets to the receiver. The receiver receives the packets and sends an acknowledgment(ACK) for each packet received successfully.
The sender and receiver maintain a window of packets indicating(指示) the number of frames we can transmit or receive at a given time. Additionally, we determine the size of the window based on the network conditions. As the sender sends packets, it updates its window to reflect(反映) the packets that have been transmitted, and the ACKs received.
However, if the sender doesn't receive an ACK for a particular(特指的) packet within a certain timeout period, it retransmits(重传) only that packet instead of the entire set of packets. The receiver only accepts packets that are within its window. If the receiver receives a packet outside the window, it discards(丢弃) the packet.
The receiver sends selective acknowledgments (SACKs) for packets received out of order or lost. The sender processes the SACKs to determine which packets need to be retransmitted.
Finally, we continue this process until we successfully send the data packets or the number of retransmissions exceeds(超过) a predetermined threshold(预定阈值).

Connection-oriented transport: TCP

TCP: Overview

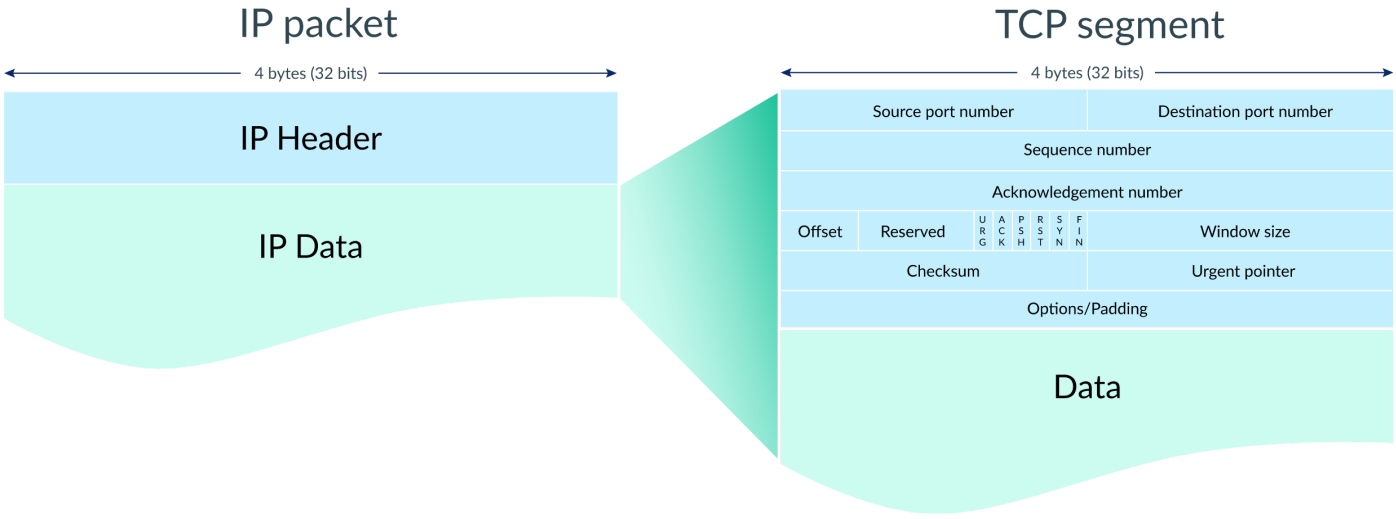
- **point-to-point(端对端)**: one sender, one receiver
- **reliable, in-order byte steam(可靠的传输)**: no "message "(消息边界)
- **pipelined(最大阈值控制)**: TCP congestion and flow control set window size
- **full duplex data(全双工数据)**:
  - bi-directional data flow(双向数据流) in same connection
  - MSS: maximum segment size(最大分段大小)
- **connection-oriented(以连接为导向)**: handshaking (exchange of control msgs) inits sender, receiver state before data exchange
- **flow controlled(流量控制)**: sender will not overwhelm receiver

Segment structure



Packet format

When sending packets using TCP/IP, the data portion of each IP packet (https://www.khanacademy.org/a/ip-packets) is formatted as a **TCP segment(TCP)**.



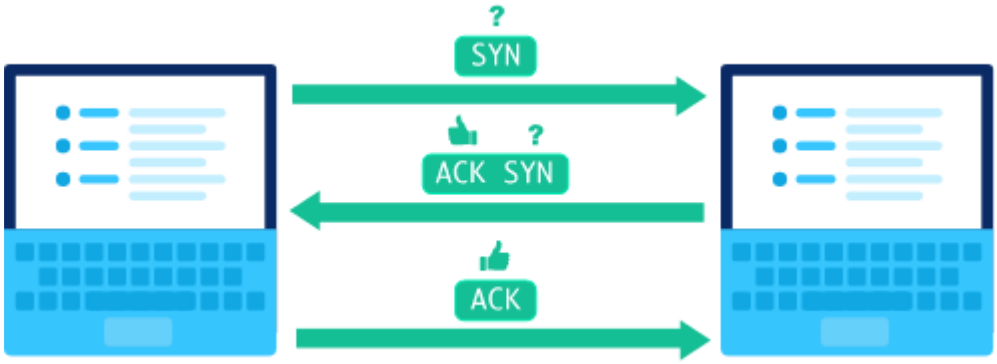
Each TCP segment contains a header and data. The TCP header contains many more fields than the UDP header and can range in size from 202020 to 606060 bytes, depending on the size of the options field().

The TCP header shares some fields with the UDP header: source port number, destination port number, and checksum. To remember how those are used, review the UDP article (https://www.khanacademy.org/a/user-datagram-protocol-udp) .

From start to finish

Step 1: Establish connection(□□□□)

When two computers want to send data to each other over TCP, they first need to establish(□□) a connection using a **three-way handshake(3 □□□)**.



The first computer sends a packet with the SYN bit set to 111 (SYN = "synchronize(□□□)?"). The second computer sends back a packet with the ACK bit set to 111 (ACK = "acknowledge!") plus the SYN bit set to 111. The first computer replies back with an ACK.

The SYN and ACK bits are both part of the TCP header:

![image (1)](./03-transport-layer.assets/image (1).png)

In fact, the three packets involved in the three-way handshake do not typically(□□) include any data. Once the computers are done with the handshake, they're ready to receive packets containing actual data(□□□□□□).

Step 2: Send packets of data

When a packet of data is sent over TCP, the recipient(□□□) must always acknowledge what they received.

![image (2)](./03-transport-layer.assets/image (2).png)

The first computer sends a packet with data and a sequence number. The second computer acknowledges it by setting the ACK bit and increasing the acknowledgement number by the length of the received data.

The sequence and acknowledgement numbers are part of the TCP header:

![image (3)](./03-transport-layer.assets/image (3).png)

Those two numbers help the computers to keep track of which data was successfully received, which data was lost, and which data was accidentally(□□□) sent twice.

Step 3: Close the connection

Either(□□□□) computer can close the connection when they no longer want to send or receive data.

![image (4)](./03-transport-layer.assets/image (4).png)

A computer initiates closing the connection by sending a packet with the FIN bit set to 1 (FIN = finish). The other computer replies with an ACK and another FIN. After one more ACK from the initiating computer, the connection is closed.

Detecting lost packets

After sending off a packet, the sender starts a timer and puts the packet in a retransmission queue(传输队列). If the timer runs out and the sender has not yet received an ACK from the recipient, it sends the packet again.
The retransmission may lead to the recipient receiving duplicate packets, if a packet was not actually lost but just very slow to arrive or be acknowledged. If so, the recipient can simply discard(丢弃) duplicate packets. It's better to have the data twice than not at all!

Handling out of order packets

TCP connections can detect out of order(乱序) packets by using the sequence(序列) and acknowledgement numbers.

When the recipient sees a higher sequence number than what they have acknowledged so far, they know that they are missing at least one packet in between. In the situation pictured above, the recipient sees a sequence number of #73 but expected a sequence number of #37. The recipient lets the sender know there's something amiss(错误，有什么不对劲) by sending a packet with an acknowledgement number set to the expected sequence number.
Sometimes the missing packet is simply taking a slower route through the Internet and it arrives soon after.

Other times, the missing packet may actually be a lost packet and the sender must retransmit the packet.

In both situations, the recipient has to deal with out of order packets. Fortunately, the recipient can use the sequence numbers to reassemble(重新组装) the packet data in the correct order.

TCP round trip time, timeout

How to set TCP timeout value?

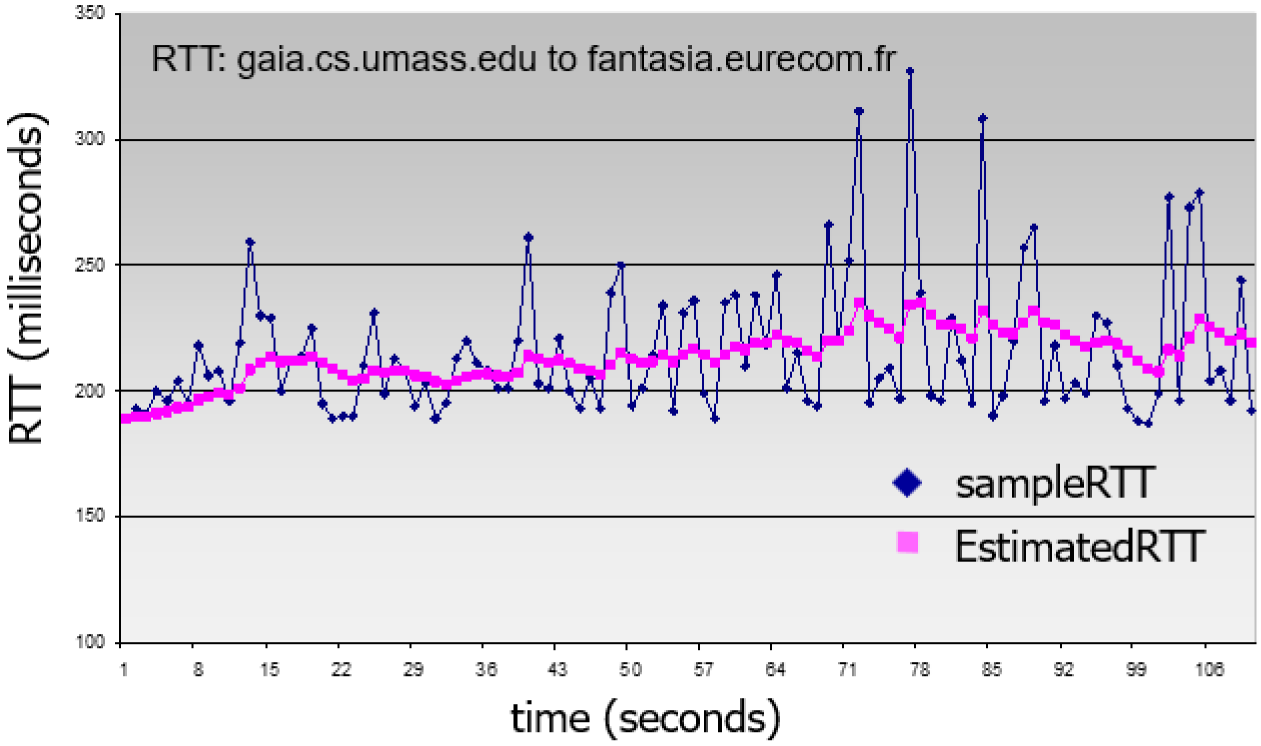
- longer than RTT(Round-Trip Time) (but RTT varies(是变化的))
- too short: premature(过早的) timeout, unnecessary retransmissions(不必要的重发)
- too long: slow reaction(反应缓慢) to segment loss(片段损失)

How to estimate RTT?

- SampleRTT: measured time(测量时间) from segment transmission until ACK receipt (ignore retransmissions)
- SampleRTT will vary, want estimated(估计的) RTT “smoother”

$$ EstimatedRTT = (1 - \alpha) * EstimatedRTT + \alpha * SampleRTT $$

typical value: α = 0.125



Reliable data transfer

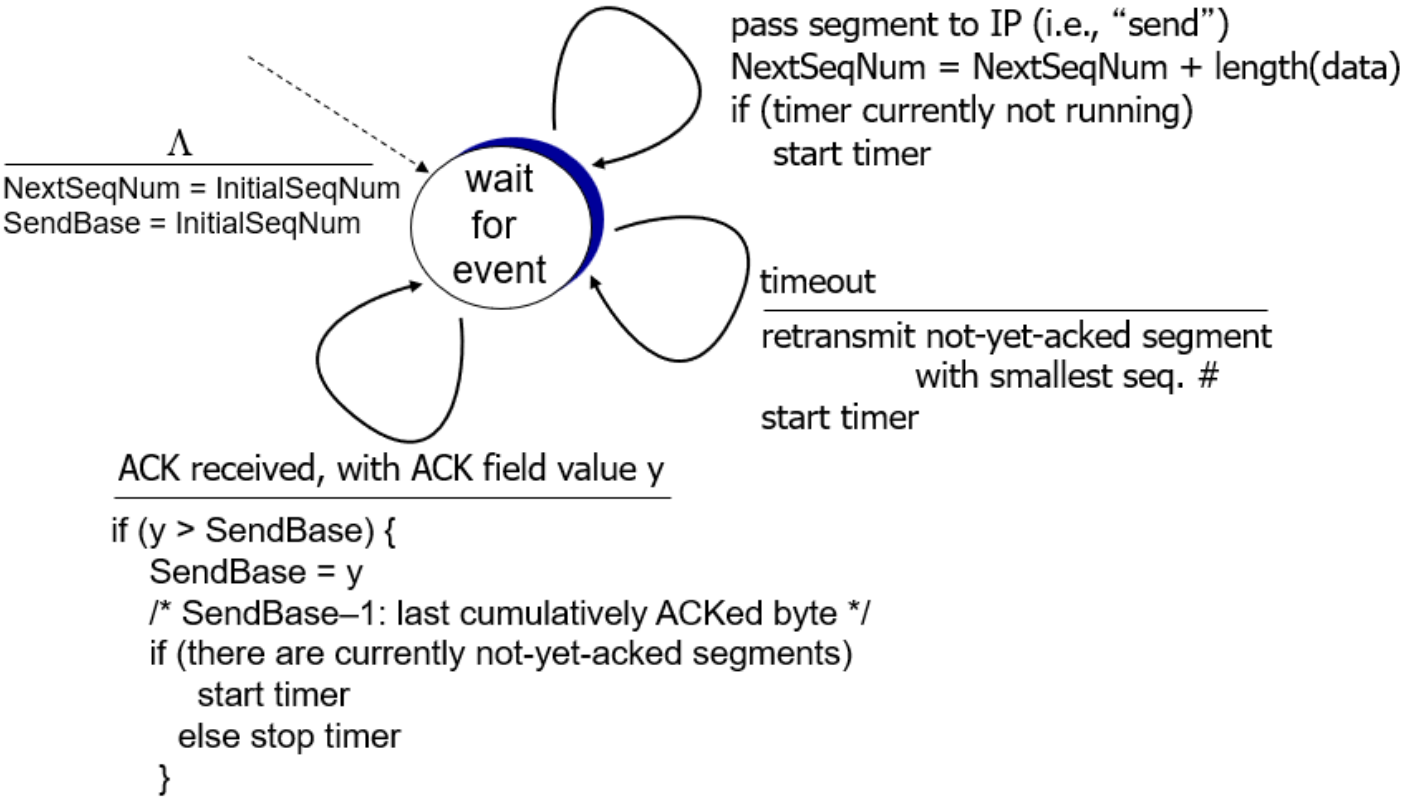
TCP creates rdt service on top of(在…之上) IP's unreliable service:

- pipelined segments(管道化分段传输)
- cumulative acks(累积式确认应答)
- single retransmission timer(单一重传计时器)

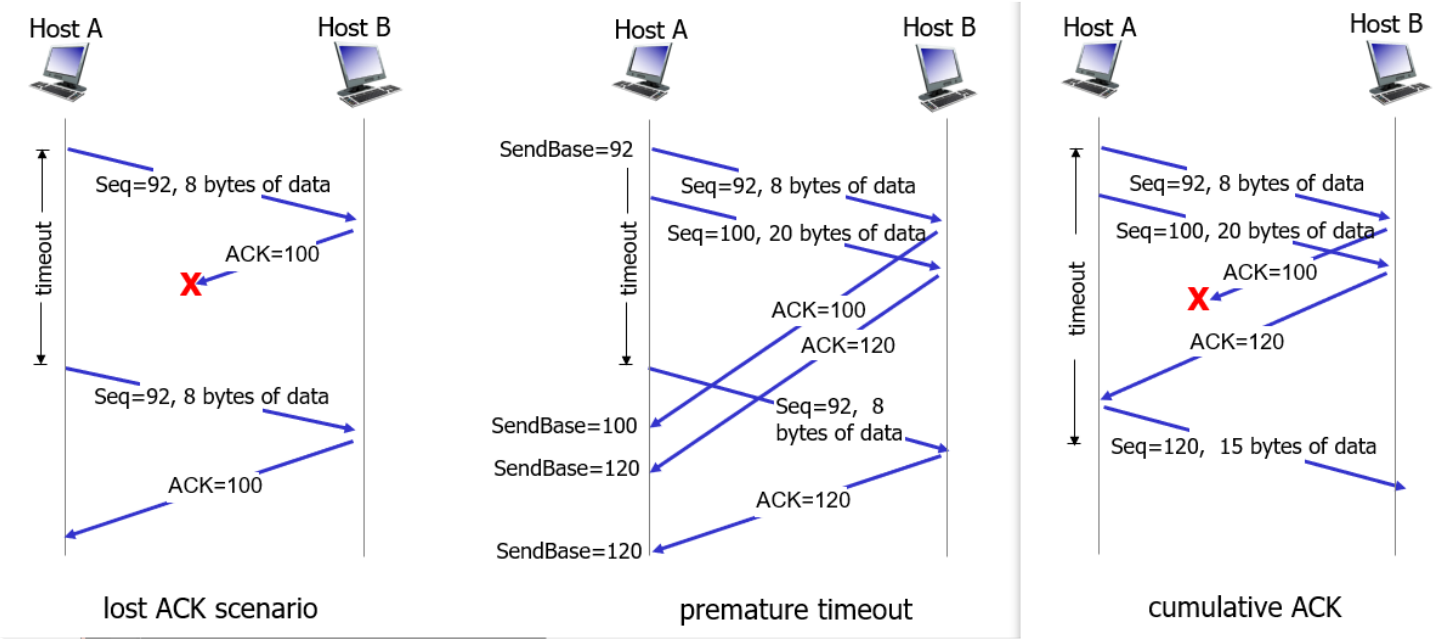
retransmissions triggered by(由…触发):

- timeout events
- duplicate acks(重复应答)

TCP sender (simplified)



TCP: retransmission scenarios



TCP ACK generation

event at receiver	TCP receiver action
arrival of in-order segment with expected seq # (□□□□□□#). All data up to expected seq # already ACKed	**delayed ACK(□□□□)**. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment(□□□□□□) with expected seq #. One other segment has ACK pending(□□□□□□	**immediately send single cumulative(□□□) ACK**, ACKing both in-order segments

arrival of out-of-order segment(无序段**)** higher-than-expect seq #(高于预期序号#). Gap detected(差异检测)	**immediately send duplicate ACK**, indicating seq # of next expected byte
arrival of segment that partially or completely(部分或完全) fills gap(填补缺口)	immediate send ACK, provided that segment starts at lower end of gap

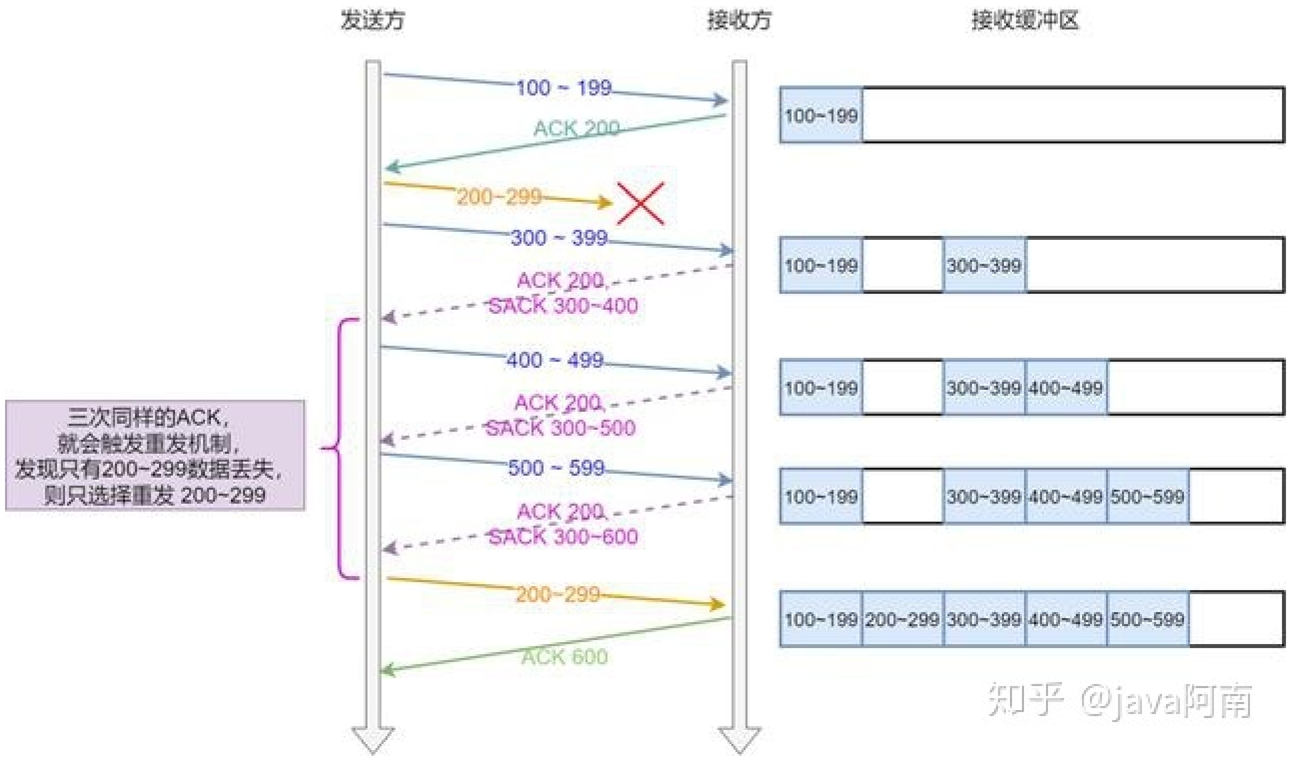
图解 TCP 重传、滑动窗口、流量控制、拥塞控制 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/135932018)

SACK 方法

还有一种实现重传机制的方式叫：SACK（ Selective Acknowledgment 选择性确认）。

这种方式需要在 TCP 头部「选项」字段里加一个 SACK 的东西，它可以将缓存的地图发送给发送方，这样发送方就可以知道哪些数据收到了，哪些数据没收到，知道了这些信息，就可以只重传丢失的数据。

如下图，发送方收到了三次同样的 ACK 确认报文，于是就会触发快速重发机制，通过 SACK 信息发现只有 200~299 这段数据丢失，则重发时，就只选择了这个 TCP 段进行重复。



Flow control

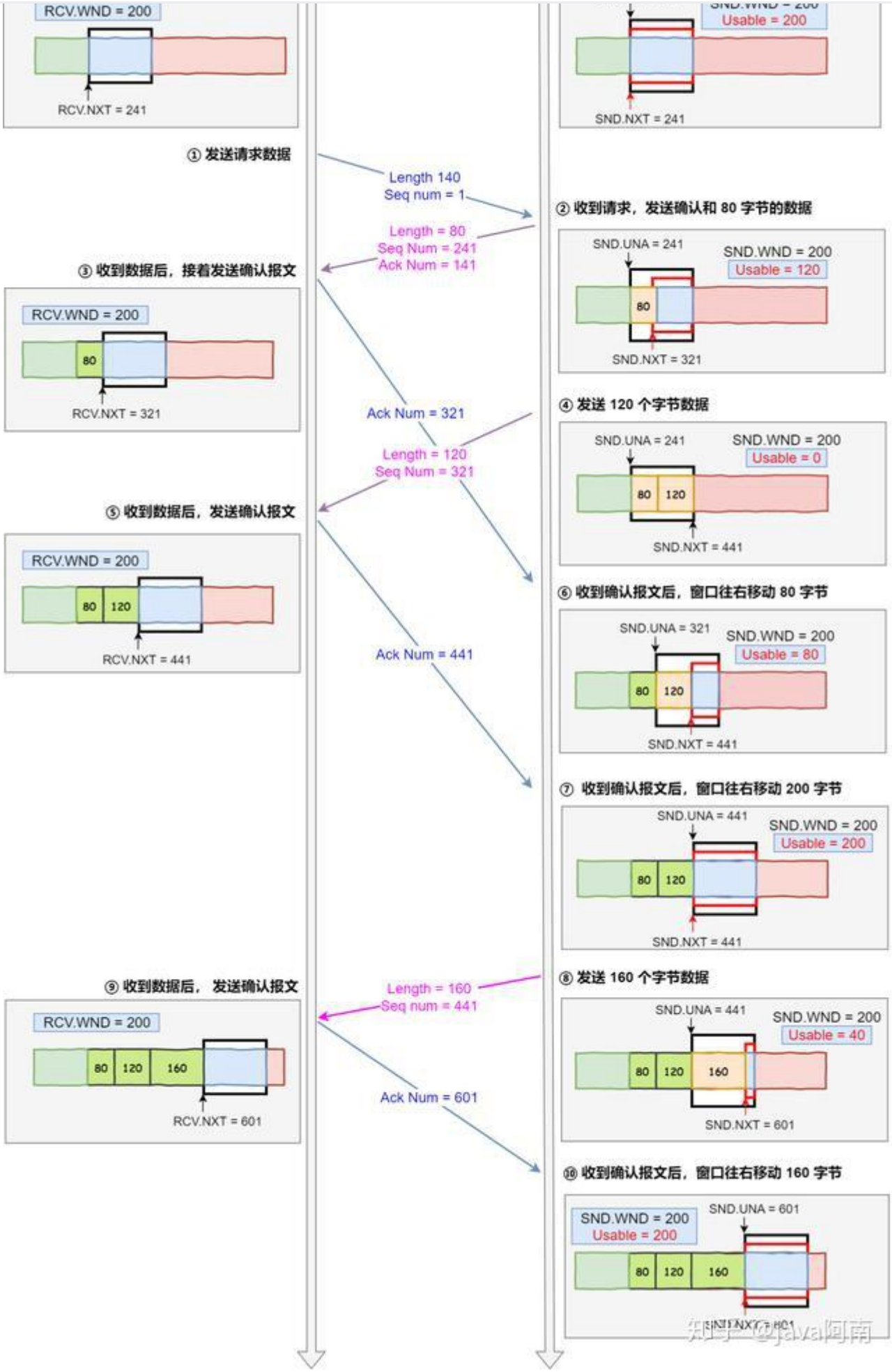
图解 TCP 重传、滑动窗口、流量控制、拥塞控制 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/135932018)

发送方不能无脑的发数据给接收方，要考虑接收方处理能力。如果一直无脑的发数据给对方，但对方处理不过来，那么就会导致触发重发机制，从而导致网络流量的无端的浪费。

为了解决这种现象发生，**TCP** 提供一种机制可以让「发送方」根据「接收方」的实际接收能力控制发送的数据量，这就是所谓的流量控制。

下面举个栗子，为了简单起见，假设以下场景：

- 客户端是接收方，服务端是发送方
- 假设接收窗口和发送窗口相同，都为 200
- 假设两个设备在整个传输过程中都保持相同的窗口大小，不受外界影响


RCV.WND = 200
RCV.NXT = 241
① 发送请求数据
Length 140
Seq num = 1
SND.WND = 200
Usable = 200
SND.NXT = 241
② 收到请求，发送确认和 80 字节的数据
Length = 80
Seq Num = 241
Ack Num = 141
SND.UNA = 241
SND.WND = 200
Usable = 120
80
SND.NXT = 321
③ 收到数据后，接着发送确认报文
RCV.WND = 200
80
RCV.NXT = 321
Ack Num = 321
④ 发送 120 个字节数据
Length = 120
Seq Num = 321
SND.UNA = 241
SND.WND = 200
Usable = 0
80
120
SND.NXT = 441
⑤ 收到数据后，发送确认报文
RCV.WND = 200
80
120
RCV.NXT = 441
⑥ 收到确认报文后，窗口往右移动 80 字节
Ack Num = 441
SND.UNA = 321
SND.WND = 200
Usable = 80
80
120
SND.NXT = 441
⑦ 收到确认报文后，窗口往右移动 200 字节
SND.UNA = 441
SND.WND = 200
Usable = 200
80
120
SND.NXT = 441
⑧ 发送 160 个字节数据
⑨ 收到数据后， 发送确认报文
Length = 160
Seq num = 441
RCV.WND = 200
80
120
160
RCV.NXT = 601
Ack Num = 601
SND.UNA = 441
SND.WND = 200
Usable = 40
80
120
160
SND.NXT = 601
⑩ 收到确认报文后，窗口往右移动 160 字节
SND.WND = 200
Usable = 200
SND.UNA = 601
80
120
160

□□□□□□□□□□□□□□□□□□□□□□

> **321**。
>
> 3. 客户端收到 80 字节数据后，于是接收窗口往右移动 80 字节，RCV.NXT 也就指向 321，这意味着客户端期望的下一个报文的序列号是 **321**，接着发送确认报文给服务端。
> 4. 服务端再次发送了 120 字节数据，于是可用窗口耗尽为 0，服务端无法在继续发送数据。
> 5. 客户端收到 120 字节的数据后，于是接收窗口往右移动 120 字节，RCV.NXT 也就指向 441，接着发送确认报文给服务端。
> 6. 服务端收到对 80 字节数据的确认报文后，SND.UNA 指针往右偏移后指向 321，于是可用窗口 Usable 增大到 80。
> 7. 服务端收到对 120 字节数据的确认报文后，SND.UNA 指针往右偏移后指向 441，于是可用窗口 Usable 增大到 200。
> 8. 服务端可以继续发送了，于是发送了 160 字节的数据后，SND.NXT 指向 601，于是可用窗口 Usable 减少到 40。
> 9. 客户端收到 160 字节后，接收窗口往右移动了 160 字节，RCV.NXT 也就是指向了 601，接着发送确认报文给服务端。
> 10. 服务端收到对 160 字节数据的确认报文后，发送窗口往右移动了 160 字节，于是 SND.UNA 指针偏移了 160 后指向 601，可用窗口 Usable 也就增大至了 200。
>
> 前面的流量控制例子，我们假定了发送窗口和接收窗口是不变的，但是实际上，发送窗口和接收窗口中所存放的字节数，都是放在操作系统内存缓冲区中的，而操作系统的缓冲区，会被操作系统调整。

Connection Management

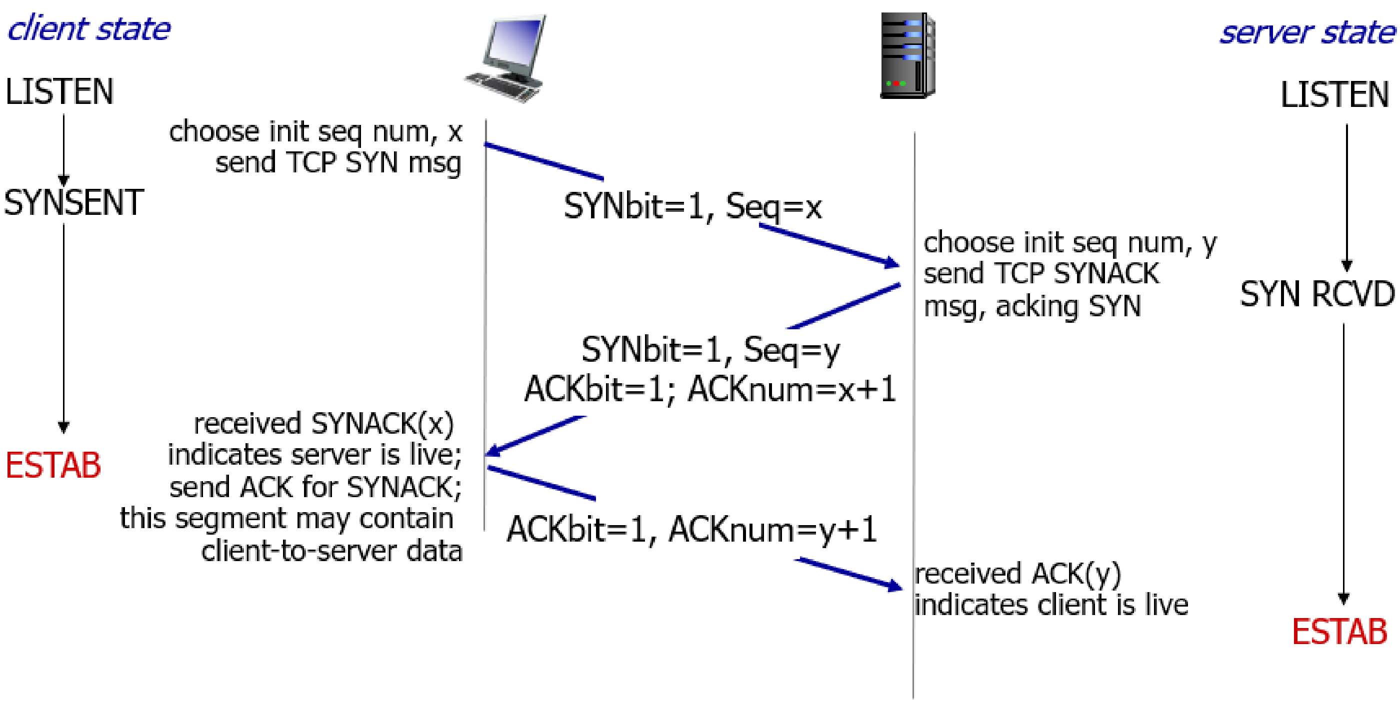
Before exchanging data, sender/receiver “handshake”:

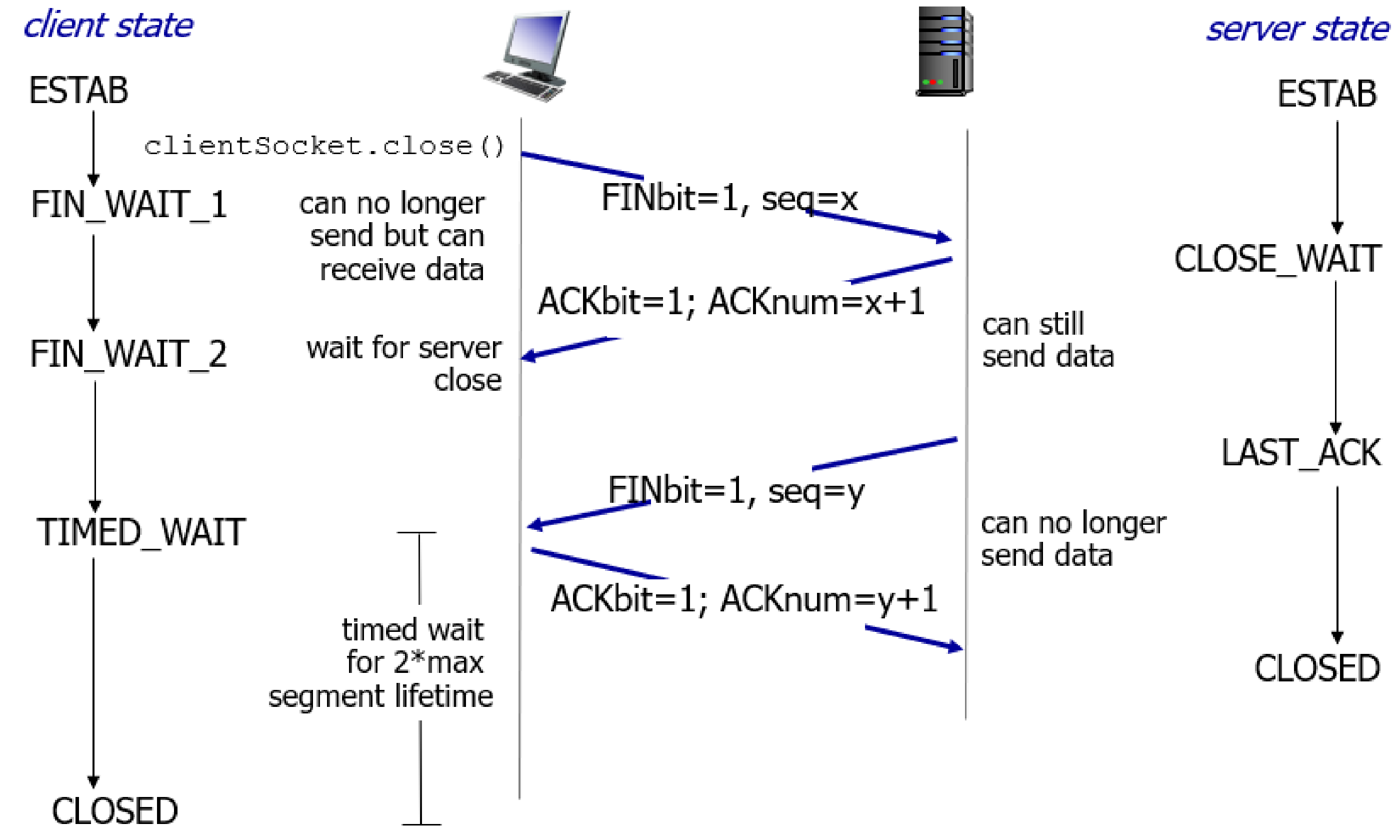
- Agree to establish(建立) connection (each knowing the other willing to establish connection)
- Agree on connection parameters

详见：

- Chapter2 @ 建立连接（三次握手）
- Chapter2 @ 断开连接（四次挥手）
- Chapter3 @ Packet format

三次握手：





Principles of congestion control

参考：一分钟搞懂流量控制与拥塞控制 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/102175027)，有删改。

流量控制与拥塞控制

流量控制（flow control）：接收端告诉发送端该发多少，解决发送方和接收方速率不匹配的问题，通过滑动窗口或者其他措施限制发送端的速率。

- 作用：控制发送方的窗口大小，让接收方来得及接收。
- 控制对象：流量控制是点对点的，只控制一个发送端。
- 实现原理：通过滑动窗口就可以实现，接收方告知发送方自己的窗口大小，发送方立刻更改即可实现流量控制。

拥塞控制（congestion control）：为了避免网络资源被耗尽的问题，通过拥塞窗口或者其他方法感知和调整网络的状态。

- 作用：避免给整体的网络造成较大的堵塞。
- 控制对象：拥塞控制是全局的，涉及到所有的主机和网络因素。
- 实现原理：四种算法（慢开始，拥塞避免、快重传、快恢复）

下面将详细讲解拥塞控制的四种算法。

慢开始

拥塞避免

该算法用来控制拥塞窗口的增长速率，每一次 RTT 往返之后，拥塞窗口 $+1$ 而不是翻倍，这样的话拥塞窗口以线性速率增长，流量可控。

快重传

当发送方没有在超时期限内收到确认信号的话就认为网络阻塞了，此时拥塞窗口变为 1 ，同时把慢开始门限值 $ssthresh$ 减半。

- 拥塞窗口大小 $< ssthresh$ 时，使用慢开始算法
- 拥塞窗口大小 $> ssthresh$ 时，使用拥塞避免算法
- 拥塞窗口大小 $= ssthresh$ 时，慢开始与拥塞避免算法均可

接收方收到一个失序的报文段后就立刻发出重复确认，如下图，$M3$ 丢失，会重复确认 $M2$

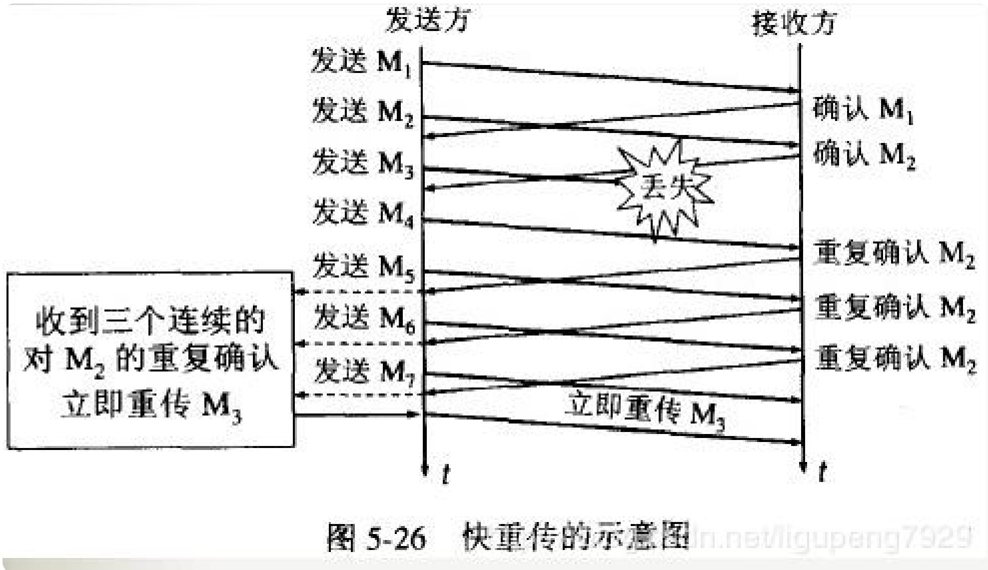


图 5-26 快重传的示意图

一旦接收方连续 3 次收到同一个重复确认就会立马启动快重传算法，即立马重传 $M3$，而不会等待 $M3$ 的超时时间到期。

快恢复

快恢复是配合快重传使用的。

上图所示的 $M3$ 丢失，可能是因为 $M3$ 在某个节点因为网络波动等非网络拥挤情况阻塞住了。

也就是网络其实并没有拥塞，快恢复就是在此情况下避免直接重传会真正导致网络阻塞，其原理就是先将拥塞窗口设置成 $ssthresh$ 的大小，然后执行拥塞避免算法

控制流程

下图展示了整个控制流程：

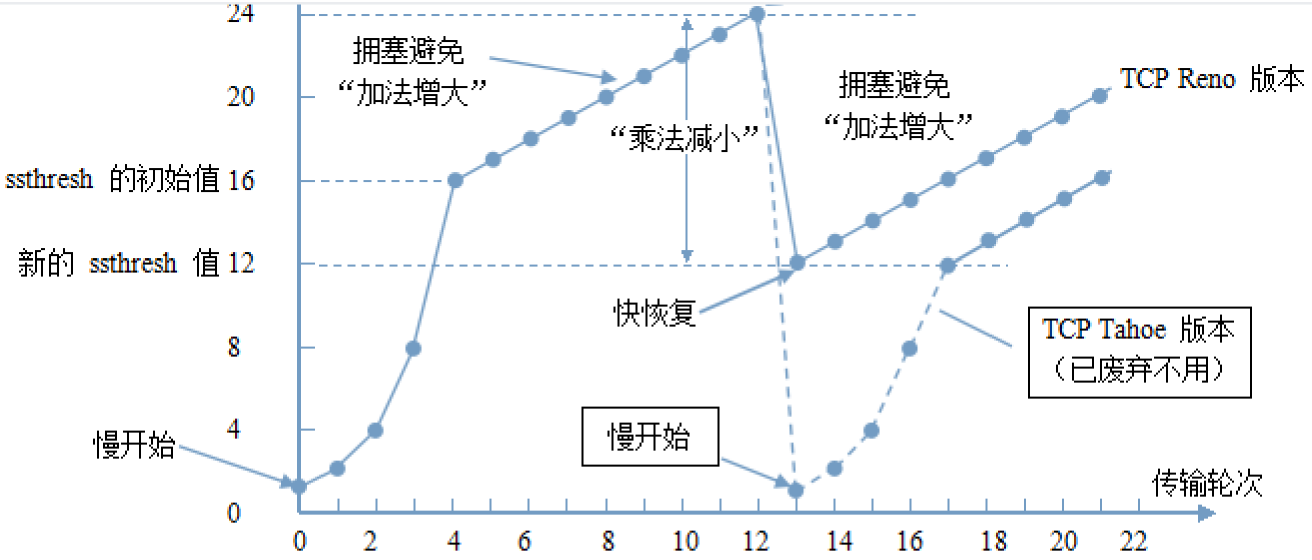


图 从连续收到三个重复的确认转入拥塞避免

TCP congestion control

TCP 的拥塞控制（详解） *tcp* 拥塞控制一颗程序媛 0915 的博客-CSDN 博客

(https://blog.csdn.net/qq_41431406/article/details/97926927)

待补充。

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

Outline

1. Overview of Network layer
    - data plane
    - control plane
2. What’s inside a router
3. IP: Internet Protocol
    - datagram format
    - fragmentation
    - IPv4 addressing
    - network address translation
    - IPv6
4. Generalized Forward and SDN
    - match
    - action
    - OpenFlow: examples of match-plus-action in action

Overview of Network layer

> 部分内容摘自【计算机网络-自顶向下】4—Network Layer: Data Plane 网络层：数据平面（概述、路由器工作原理、IPv4、DHCP、IPv6）_一棵____大树的博客-CSDN 博客 (https://blog.csdn.net/weixin_53580595/article/details/129480116)，有删改。

- transport segment(片段) from sending to receiving host
- on sending side encapsulates(封装) segments into datagrams(数据报；数据包)
- on receiving side, delivers(传送) segments to transport layer(传输层)
- network layer protocols(协议) in every host(主机), router
- router examines(检查) header fields in all IP datagrams passing through it

互联网采用的设计思路是这样的：网络层向上只提供简单灵活的、无连接的、尽最大努力交付的数据报服务。

Two key network-layer functions:

- forwarding 转发: move packets from router’s input to appropriate(适当的) router output
- routing 路由: determine(确定) route taken by packets from source to destination (using routing algorithms 路由算法)

> - 转发（**forwarding**）——数据平面（**data plane**）：将分组从一个输入链路接口转移到适当的输出链路接口的路由器的本地动作（主要利用硬件）；

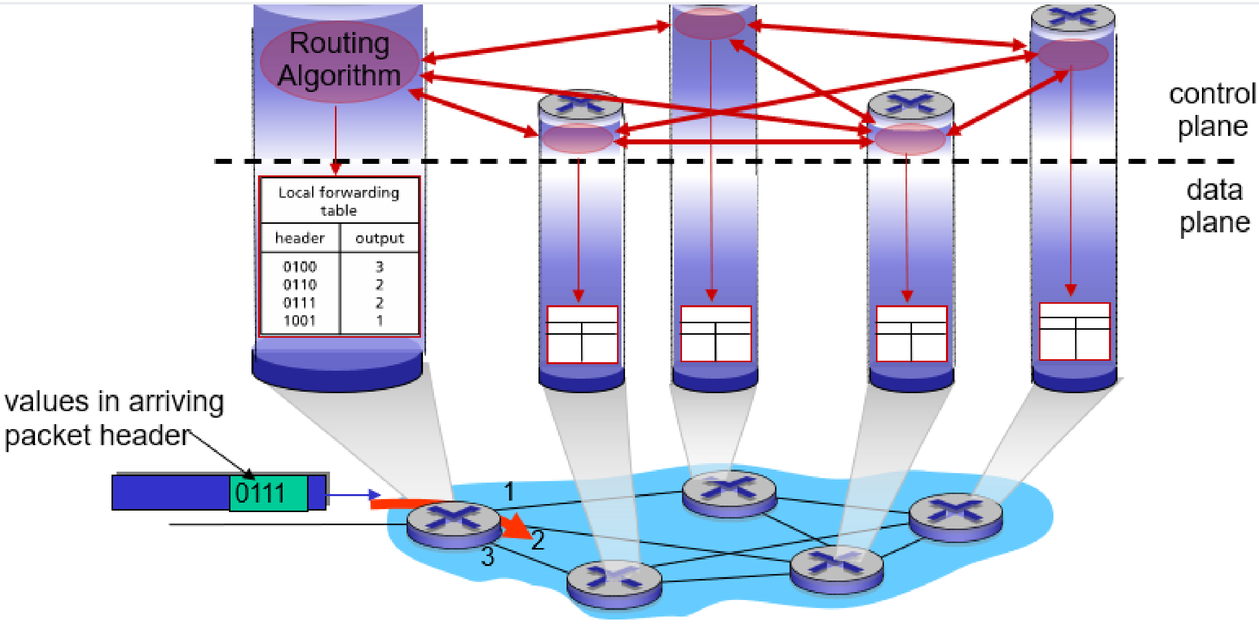
Data plane & control plane

> 什么是控制平面? | 控制平面与数据平面 | Cloudflare (https://www.cloudflare.com/zh-cn/learning/network-layer/what-is-the-control-plane/)
>
> 什么是网络中的“平面”?
>
> 在网络中，平面是特定过程发生位置的抽象概念。该术语是在“存在平面”的意义上使用的。网络中最常引用的两个平面是控制平面和数据平面（也称为转发平面）。
>
> 什么是控制平面?
>
> 控制平面（Control plane）是网络中控制数据包 (https://www.cloudflare.com/learning/network-layer/what-is-a-packet/) 转发方式的部分——即数据如何从一个地方发送到另一个地方。例如，创建路由 (https://www.cloudflare.com/learning/network-layer/what-is-routing/) 表的过程被认为是控制平面的一部分。路由器使用各种协议 (https://www.cloudflare.com/learning/network-layer/what-is-a-protocol/) 来识别网络路径，并将这些路径存储在路由表中。
>
> 什么是数据平面/转发平面?
>
> 与决定如何转发数据包的控制平面相反，数据平面（Data plane）会实际转发数据包。数据平面也称为转发平面。
>
> 打个比方，控制平面是在城市道路交叉口工作的交通信号灯。数据平面（或转发平面）则更像是在道路上行驶、在路口停下并遵守交通信号灯的汽车。

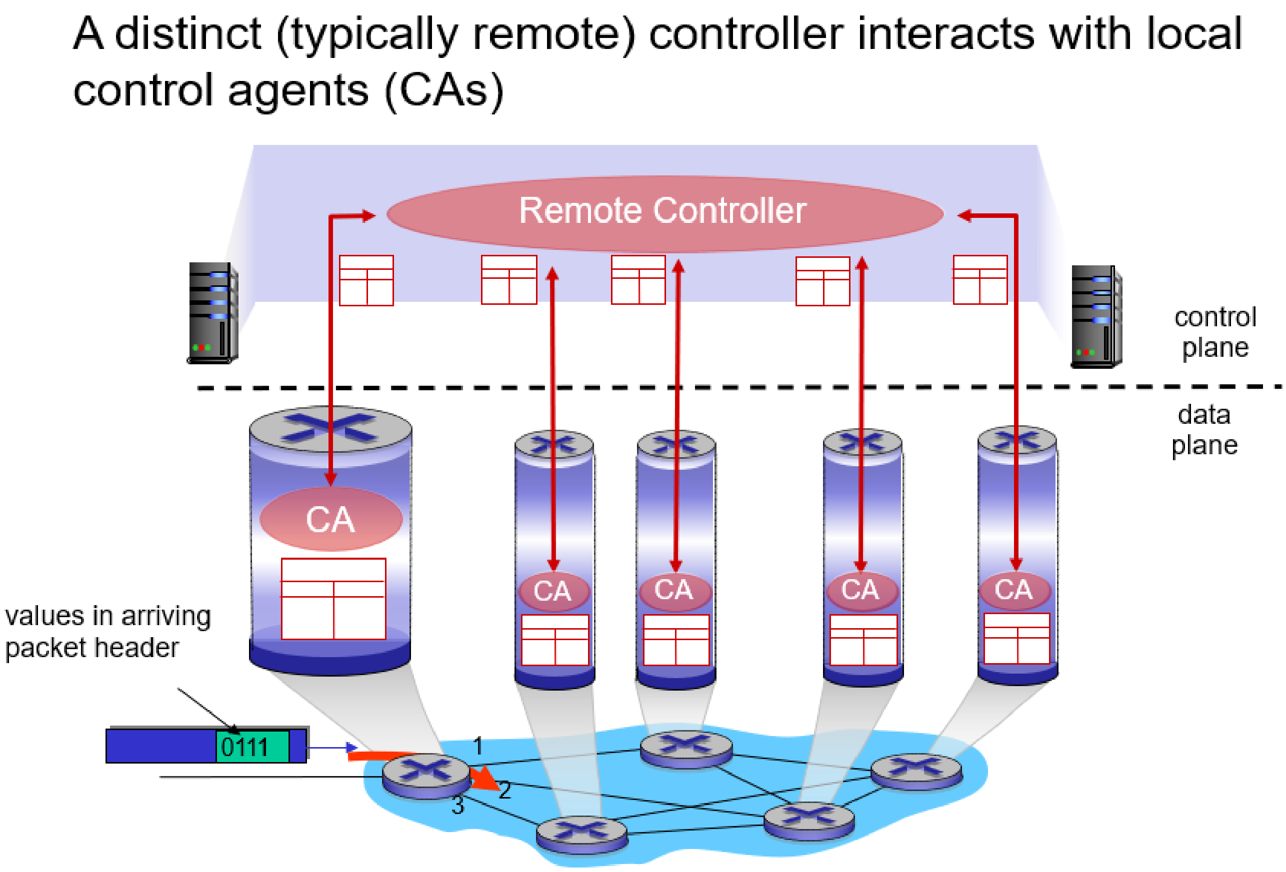
two control-plane approaches:

- traditional routing algorithms(传统路由算法): implemented(执行) in routers
- software-defined networking (SDN，软件定义网络): implemented in (remote) servers

> 每台网络路由器中有一个关键元素是它的转发表（forwarding table）。路由器检查到达分组头部的一个或多个字段值，进而使用这些头部值在其转发表中索引，通过这种方法来转发分组。
>
> 两种控制平面方法:
>
> - 传统方法是由路由器器中的路由选择算法决定转发表中的值:



- 软件定义网络（**Software-Defined Networking**，**SDN**）方法是从路由器物理上分离的另一种方法，远程控制器计算和分发转发表以供每台路由器所使用：


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

Network service model

ATM 是 "Asynchronous Transfer Mode" 的缩写，意为“异步传输模式”。它是一种网络通信技术，在计算机网络中被广泛使用。ATM 网络通过将数据划分为固定大小的单元（称为“单元”或“细胞”）来传输数据，这些单元具有相同的长度，并可以在网络中以任何顺序传输。

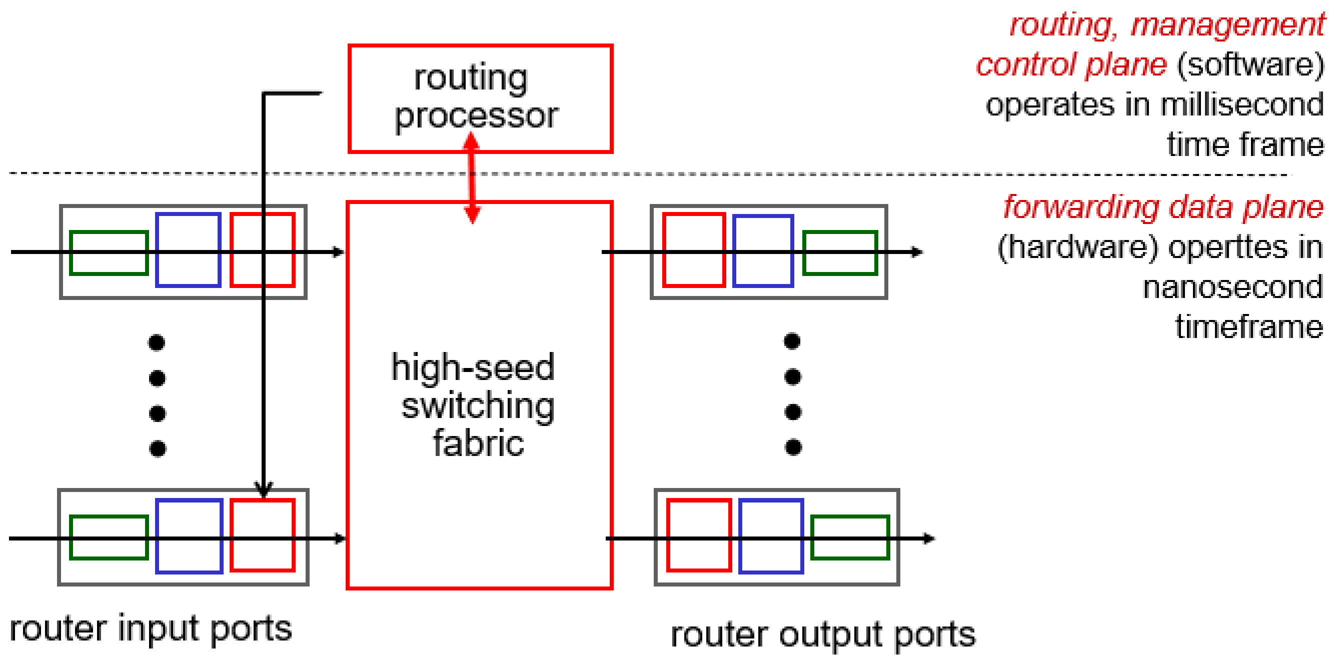
ATM 技术最初被设计用于广域网和局域网之间的高速数据传输，但由于其高效性和可靠性，也被应用于语音、视频和图像等多媒体数据的传输。

- 确保交付;
- 具有时延上界的确保交付;
- 有序分组交付;
- 确保最小带宽;
- 安全性。

What’s inside a router

Router architecture overview

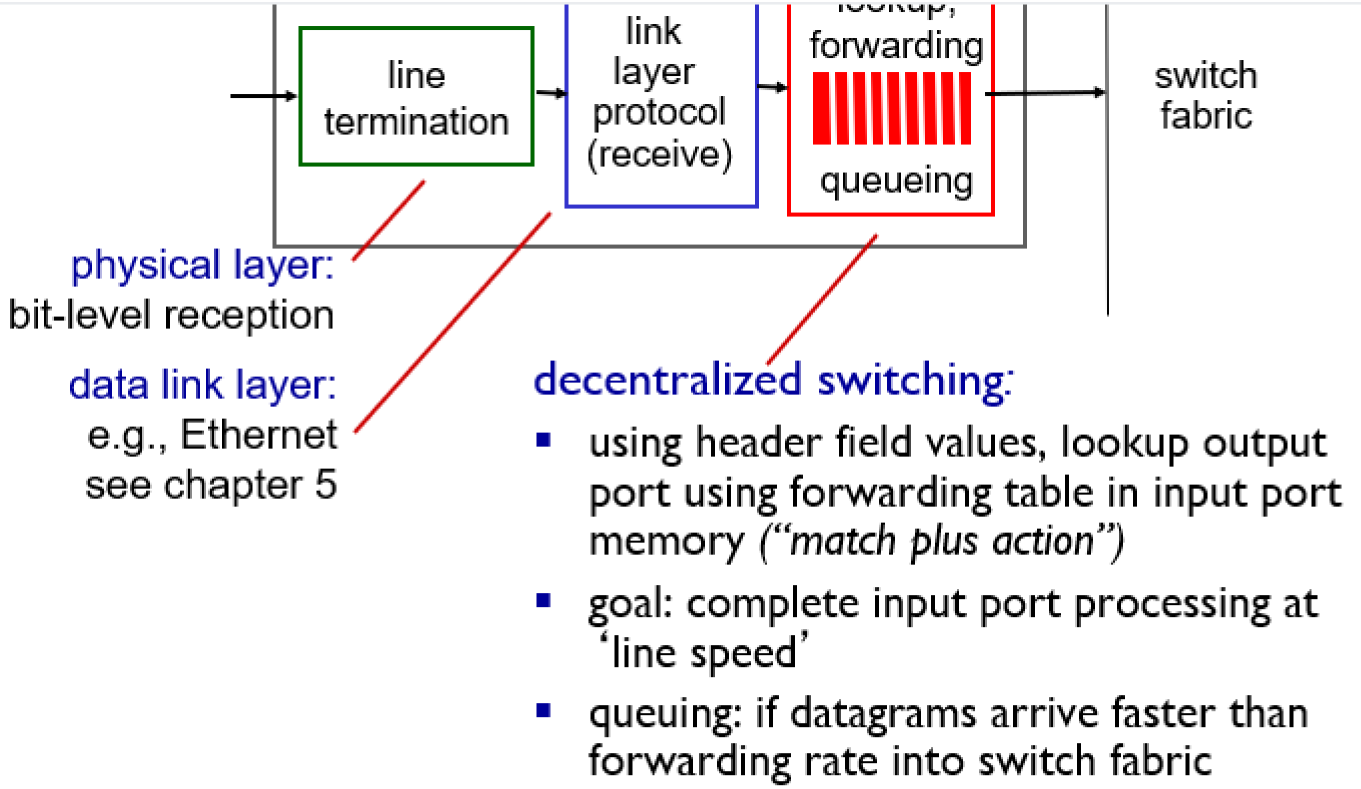
路由器体系结构概述



注：上半为路由、管理控制平面（软件），下半为在转发数据平面（硬件）

Input port functions

左侧输入端:



输入端口（input port）：

- 在路由器中终结进入物理链路的物理层；
- 与位于入链路远端的数据链路层交互；
- 查找转发表决定路由器的输出端口；
- 控制分组从输入端口转发到路由选择处理器

decentralized switching(分散式交换):

- *destination-based forwarding(基于目标的转发):* forward based only on destination IP address (traditional)
- *generalized forwarding(通用转发):* forward based on any set of header field values

Longest prefix matching

最长前缀匹配

When looking for forwarding table entry(转发表条目) for given destination address, use longest address prefix(前缀) that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1

11001000 00010111 00011	2
otherwise	3

examples:

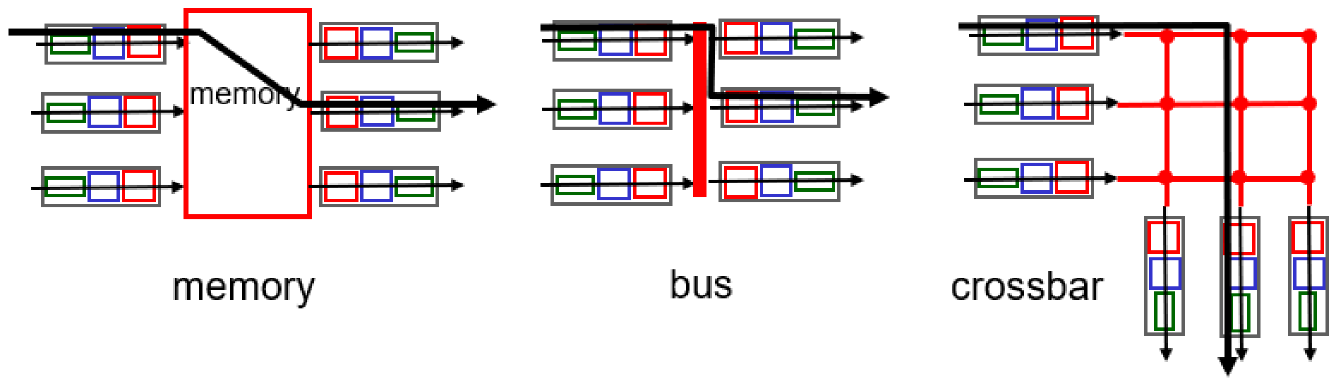
- DA: 11001000 00010111 00010110 10100001 means interface 0
- DA: 11001000 00010111 00011000 10101010 means interface 1

Switching fabrics

switch fabric(交换结构): transfer packet from input buffer to appropriate output buffer 将数据包从输入缓冲区传输到适当的输出缓冲区

switching rate(交换速率): rate at which packets can be transfer from inputs to outputs 数据包从输入传输到输出的速率

three types of switching fabrics(交换结构):



- Switching via memory(内存)

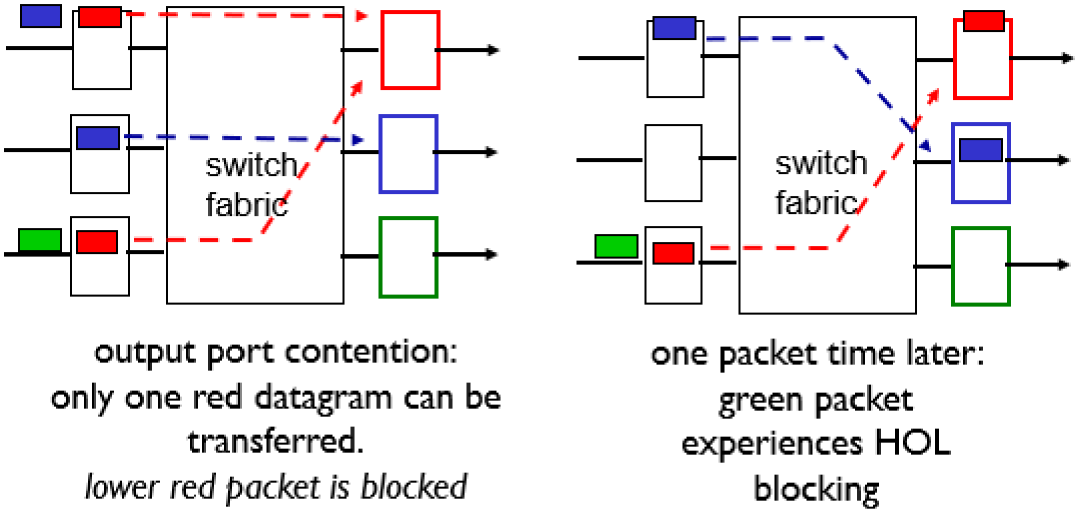
  first generation routers:

  - traditional computers with switching under direct control of CPU 在 CPU 直接控制下进行交换的传统计算机
  - packet copied to system's memory
  - speed limited by memory bandwidth
- Switching via a bus(总线)
  - *bus contention(总线争夺):* switching speed limited by bus bandwidth 交换速度受总线带宽限制
- Switching via interconnection network(互联网)

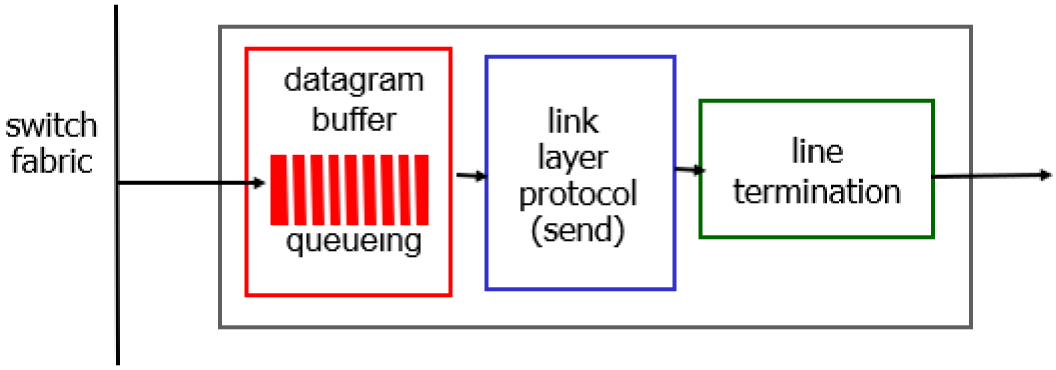
  是一种计算机网络中的通信方式，它使用一组相互连接的节点来传递数据。这些节点可以被看作是一个多端口的交换机，其中每个端口都与一个设备或者另一个节点相连。当一个节点想要发送数据时，它会将数据包发送到目标节点所在的端口，并且经过多次路由选择和转发，在互联网络中到达目的地。这种通信方式广泛应用于高性能计算、数据中心网络和分布式系统等领域。

Input port queuing 输入端口排队

- Head-of-the-Line(HOL) blocking(□□□□): queued datagram at front of queue prevents others in queue from moving forward 队列前面排队的 datagram 可防止队列中的其他人向前移动



Output ports



> 输出端口（output port）：
>
> - 存储从交换结构接收的分组；
> - 执行必要的链路层和物理层功能在输出链路上传输这些分组。

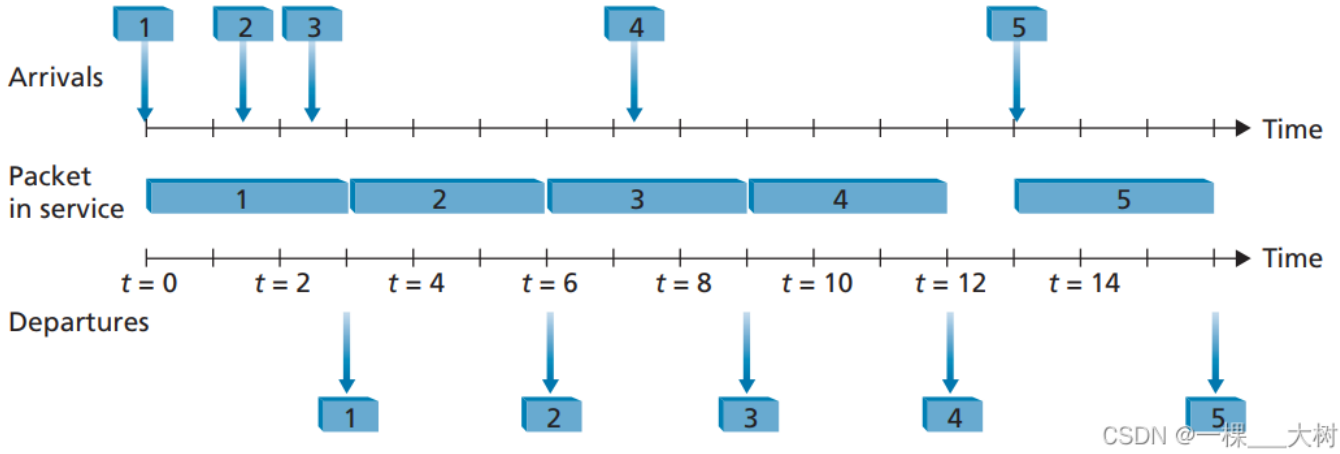
- *buffering* required when datagrams arrive from fabric faster than the transmission rate 当数据报从结构到达的速度快于传输速率时需要缓冲
- *scheduling discipline* chooses among queued datagrams for transmission 调度规则在排队的数据报中选择传输

Scheduling mechanisms

scheduling(调度): choose next packet to send on link

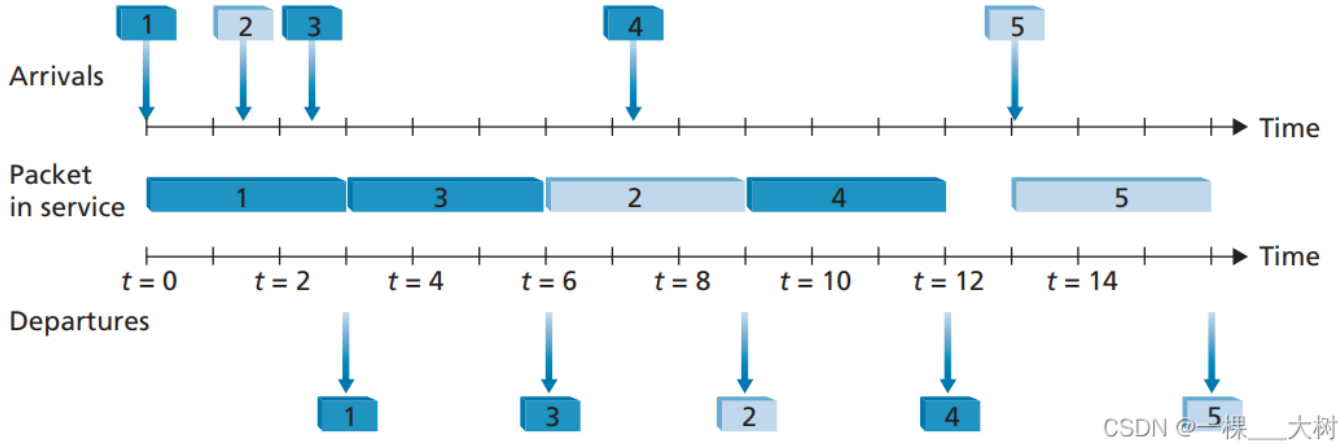
FIFO (first in first out, 先进先出) scheduling: send in order of arrival to queue

运行中的 FIFO 队列如下图所示：



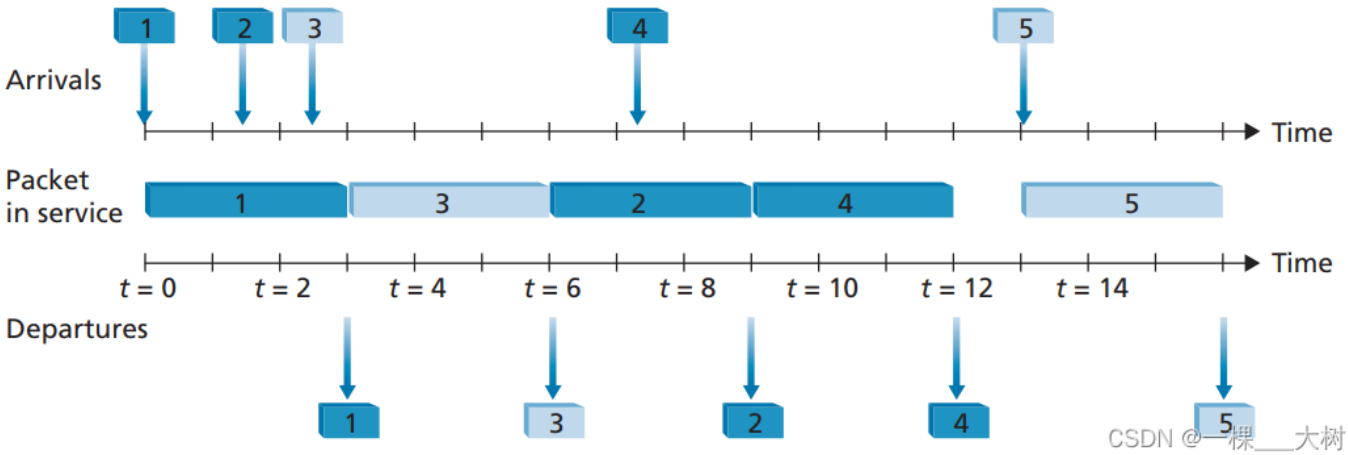
优先权排队（priority queuing）

假设分组 1、3 和 4 是高优先权，分组 2 和 5 是低优先权，在非抢占式优先权排队（non-preemptive priority queuing）中，运行如下图所示：



循环排队规则（round robin queuing discipline）

假设分组 1、2 和 4 属于第一类，分组 3 和 5 属于第二类，运行如下图所示：

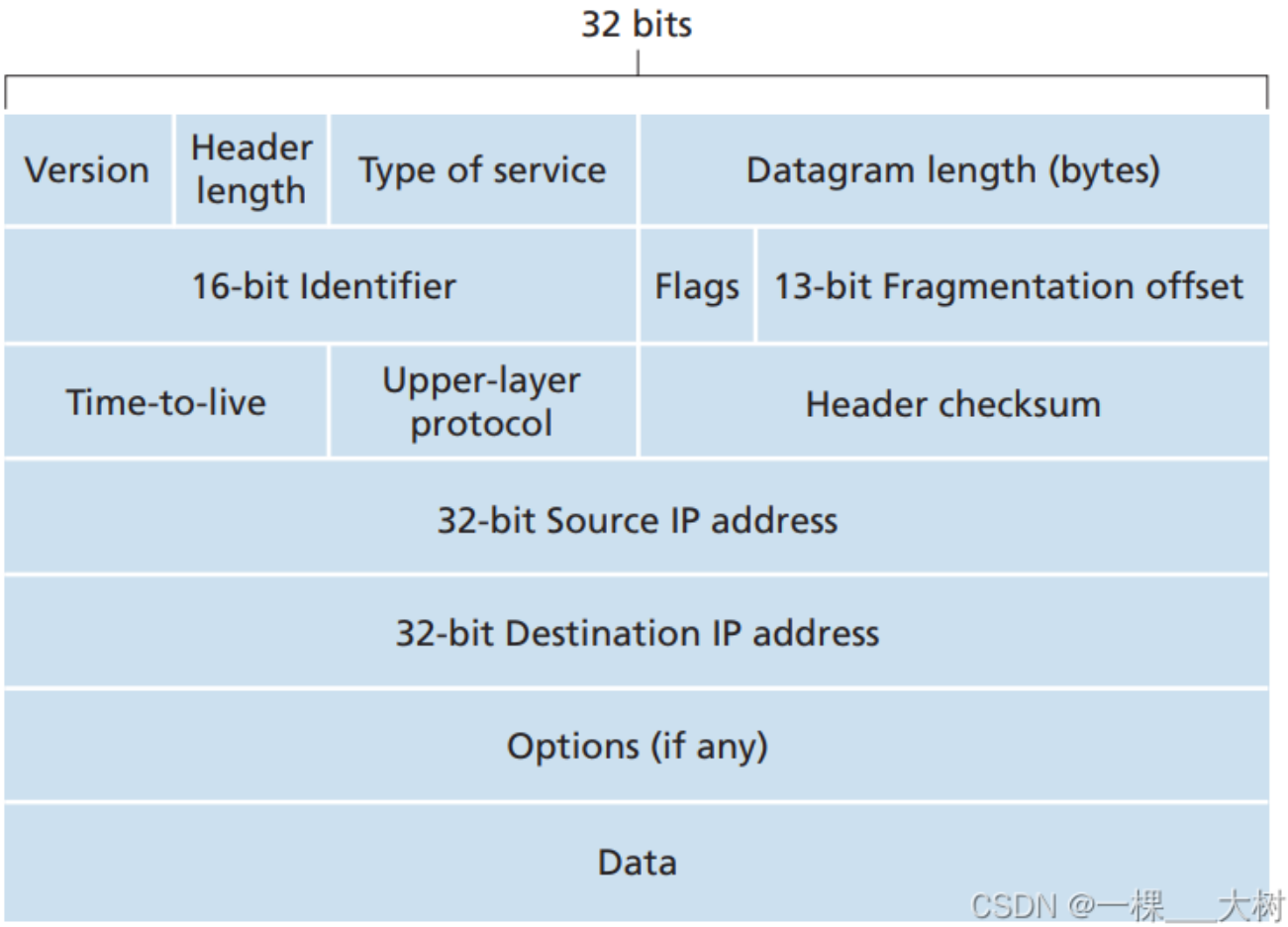


IP: Internet Protocol

> **TIP**
>
> 协议包括：语义、语法、时序。

Datagram format

IPv4 Datagram Format

> IPv4 数据报格式如下图所示：
>
> 
>
>
>
> - 版本号 (**Version**)：规定了数据报的 IP 协议版本；
> - 头部长度 (**Header length**)：在无选项 (Options) 头部占据 20 字节；
> - 服务类型 (**Type of service**)：不同类型的数据报可以相互区分；
> - 数据报长度 (**Datagram length**)：IP 数据报的总长度（头部加上数据），以字节计数；
> - 标识 (**Identifier**)、标志 (**Flags**)、片漂移 (**Fragmentation offset**)：用于 IP 分片；
> - 寿命 (**Time-to-live, TTL**)：确保数据段不会永远在网络中循环；
> - 上层协议 (**Upper-layer protocol**)：指示 IP 数据报应交付给哪个运输层协议；
> - 头部校验 (**Header checksum**)：帮助路由器检测收到的 IP 数据报中的比特错误；

> - 数据（Data）：一般为运输层报文段。

> **how much overhead?**
>
> $$ (\text{20 bytes of TCP + 20 bytes of IP} = \text{40 bytes}) + \text{app layer overhead} $$

Fragmentation

IP fragmentation, reassembly IP 分段与重组

一个链路层帧能承载的最大数据量叫作最大传送单元（Maximum Transmission Unit，MTU）。

large IP datagram divided (“fragmented”, 分段) within net:

- one datagram becomes several datagrams
- “reassembled”(重组) only at final destination
- IP header bits used to identify(识别), order related fragments(重排分段)

IPv4 addressing

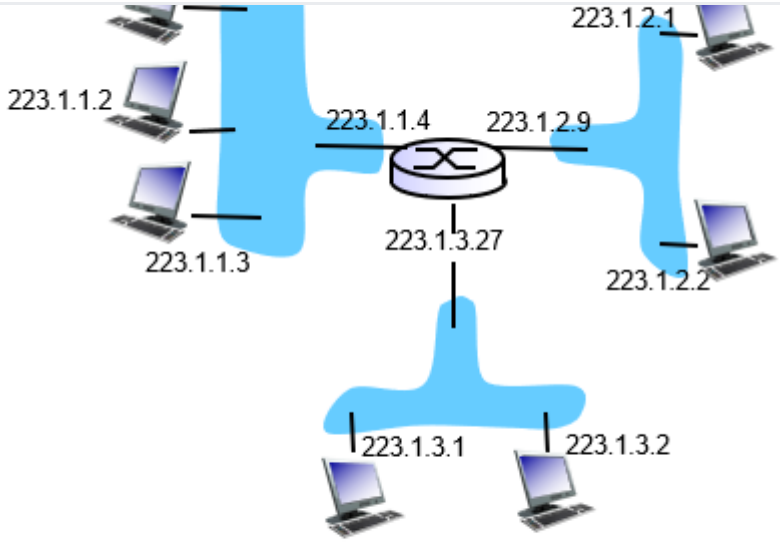
IPv4 编址：主机与物理链路之间的边界叫做接口（interface）。每个 IP 地址长度为 32 比特（4 字节），因此总共有 2^{32} 个可能的 IP 地址。这些地址通常按所谓点分十进制记法（dotted-decimal notation）书写，及地址中的每个字节用它的十进制形式书写，各字节以句点隔开。

如：地址 `192.32.216.9` 的二进制记法是 `11000001 00100000 11011000 00001001` 。

interface: connection between host/router and physical link(物理连接)

Subnets

下图中，一台路由器（具有三个接口）用于互联 7 台主机。

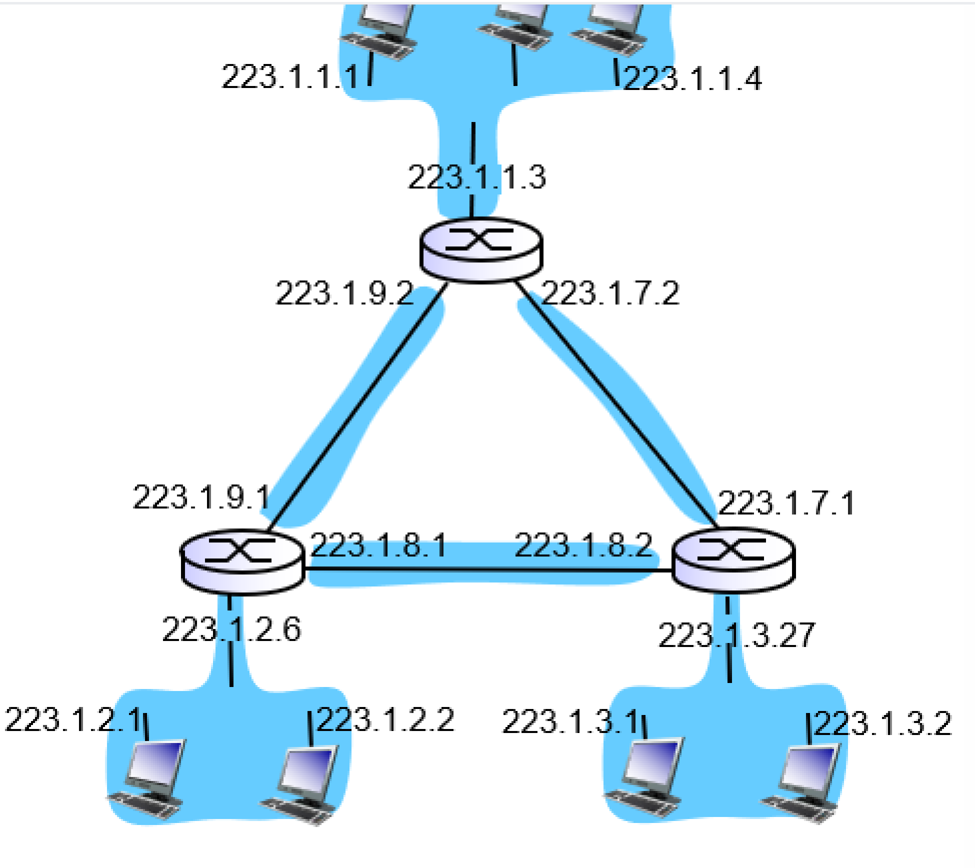


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

图中左上侧的 3 个主机和它们连接的路由器接口，都有一个形如 233.1.1.xxx 的 IP 地址。用 IP 的术语说，互联这 3 个主机接口与 1 个互联网接口的网络形成一个子网（subnet）。IP 编址为这个子网分配一个地址 233.1.1.0/24，其中/24 记法，有时称为子网掩码（network mask），指示 32 比特中的最左侧 24 比特定义了子网地址。

IP addressing: CIDR

下图是由 3 个路由器互联的 6 个子网:



因特网的地址分配策略被称为无类别域间路由选择（Classless Interdomain Routing, CIDR）。当使用子网寻址时，32 比特的 IP 地址被划分为两部分，并且也具有点分十进制形式 a.b.c.d/x，其中 x 指示了地址的第一部分中的比特数，并且该部分经常被称为该地址的前缀（prefix）（或网络前缀）。

> ### How does a _host_ get IP address?
>
> hard-coded by system admin in a file:
>
> - Windows: control-panel->network->configuration->tcp/ip->properties
> - UNIX: /etc/rc.config

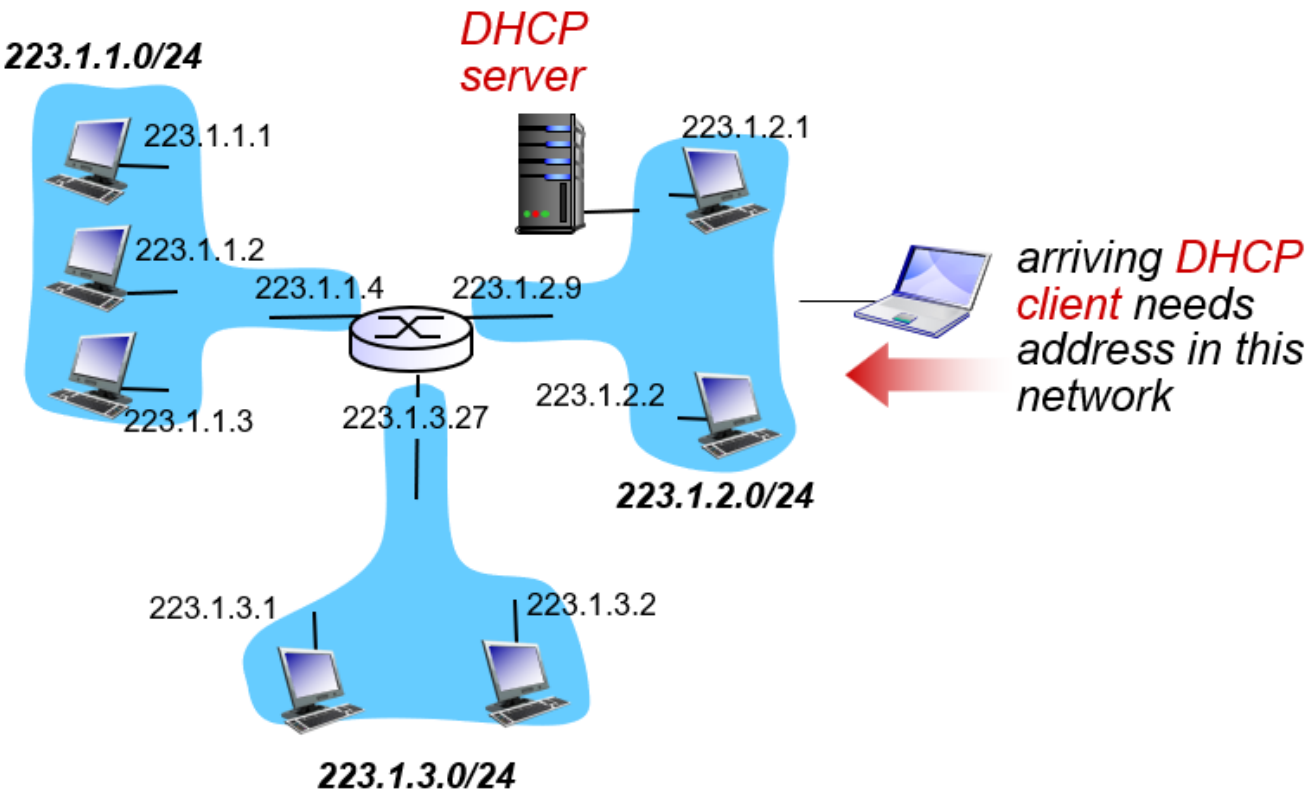
> ### TIP
>
> IP 广播地址 255.255.255.255：当主机发出以广播地址为目的地址的数据报时，该报文会交付给同一个网络中的所有主机。

获取主机地址（DHCP）

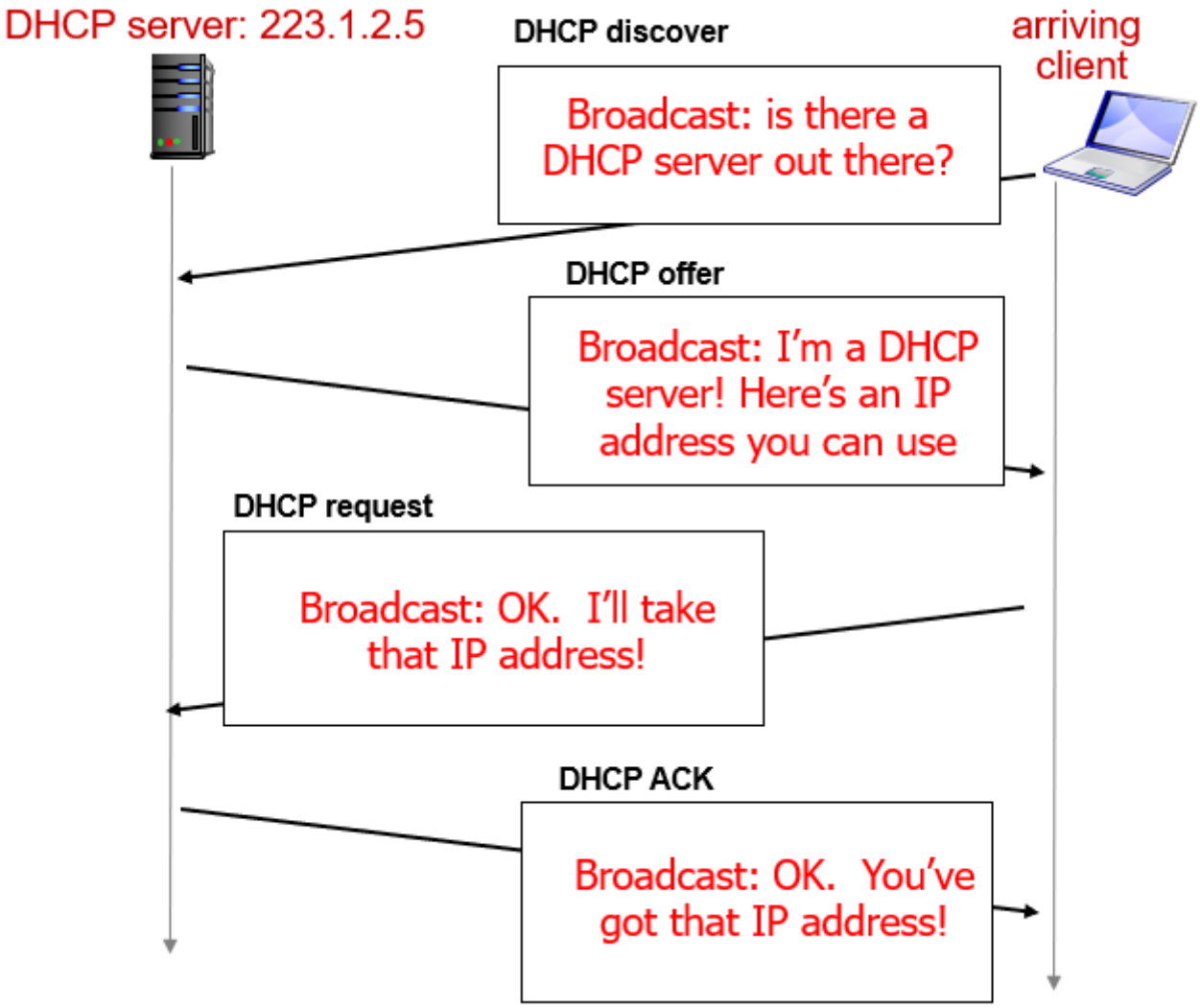
主机地址的配置更多的是使用动态主机配置协议（**Dynamic Host Configuration Protocol, DHCP**）。

goal: allow host to *dynamically(动态)* obtain its IP address from network server when it joins network

- 给定主机每次与网络连接时能得到一个相同的 IP 地址；
- 给某个主机分配一个临时的 IP 地址（temporary IP address），每次地址可能不同；
- 允许移动设备频繁的加入和离开网络。

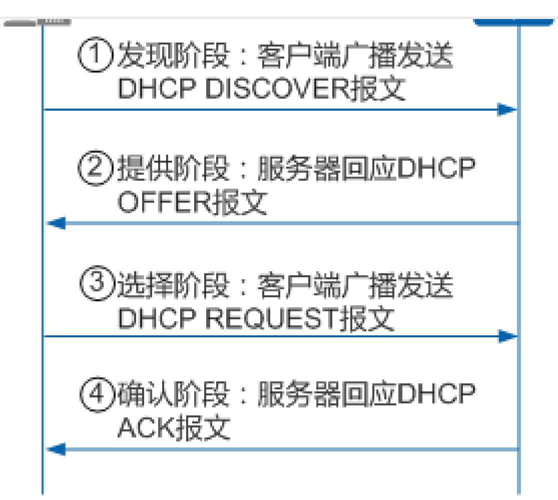


DHCP 四步交互 4 个报文都使用 `yiaddr` 字段来传递分配的地址



- address of first-hop router(□□□□□□) for client
- name and IP address of DNS sever
- network mask(网络掩码) (indicating network versus host portion of address 指示地址的网络与主机部分)

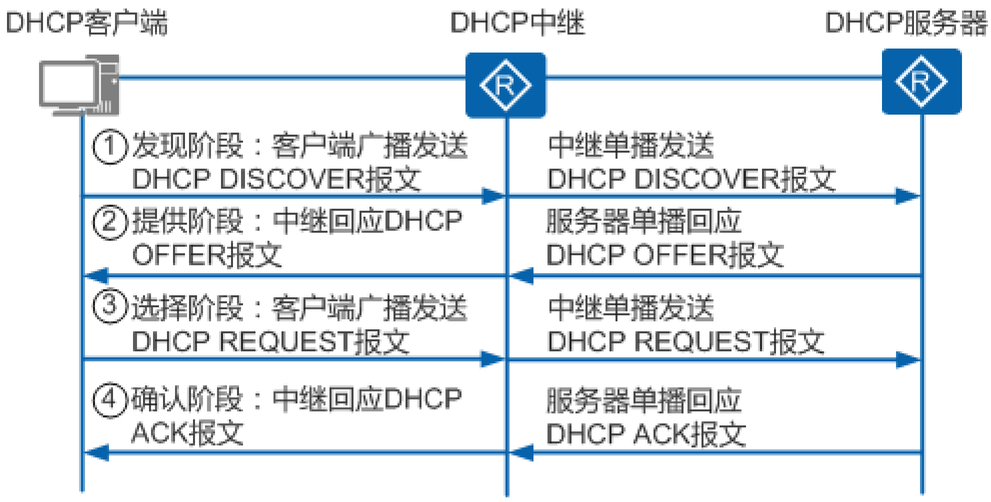
> 什么是DHCP？为什么要用DHCP？ - 华为 (huawei.com) (https://info.support.huawei.com/info-finder/encyclopedia/zh/DHCP.html)
>
> ## 什么是**DHCP**？
>
> 动态主机配置协议DHCP（Dynamic Host Configuration Protocol）是一种网络管理协议，用于集中对用户IP地址进行动态管理和配置。
>
> DHCP于1993年10月成为标准协议，其前身是BOOTP协议。DHCP协议由RFC 2131定义，采用客户端/服务器通信模式，由客户端（DHCP Client）向服务器（DHCP Server）提出配置申请，DHCP Server为网络上的每个设备动态分配IP地址、子网掩码、默认网关地址，域名服务器（DNS）地址和其他相关配置参数，以便可以与其他IP网络通信。
>
> ## 为什么要使用**DHCP**？
>
> 在IP网络中，每个连接Internet的设备都需要分配唯一的IP地址。DHCP使网络管理员能从中心结点监控和分配IP地址。当某台计算机移到网络中的其它位置时，能自动收到新的IP地址。DHCP实现的自动化分配IP地址不仅降低了配置和部署设备的时间，同时也降低了发生配置错误的可能性。另外DHCP服务器可以管理多个网段的配置信息，当某个网段的配置发生变化时，管理员只需要更新DHCP服务器上的相关配置即可，实现了集中化管理。
>
> 总体来看，DHCP带来了如下优势：
>
> - 准确的IP配置：IP地址配置参数必须准确，并且在处理“192.168.XXX.XXX”之类的输入时，很容易出错。另外印刷错误通常很难解决，使用DHCP服务器可以最大程度地降低这种风险。
> - 减少IP地址冲突：每个连接的设备都必须有一个IP地址。但是，每个地址只能使用一次，重复的地址将导致无法连接一个或两个设备的冲突。当手动分配地址时，尤其是在存在大量仅定期连接的端点（例如移动设备）时，可能会发生这种情况。DHCP的使用可确保每个地址仅使用一次。
> - IP地址管理的自动化：如果没有DHCP，网络管理员将需要手动分配和撤消地址。跟踪哪个设备具有什么地址可能是徒劳的，因为几乎无法理解设备何时需要访问网络以及何时需要离开网络。DHCP允许将其自动化和集中化，因此网络专业人员可以从一个位置管理所有位置。
> - 高效的变更管理：DHCP的使用使更改地址，范围或端点变得非常简单。例如，组织可能希望将其IP寻址方案从一个范围更改为另一个范围。DHCP服务器配置有新信息，该信息将传播到新端点。同样，如果升级并更换了网络设备，则不需要网络配置。
>
> ## **DHCP**是怎么工作的？
>
> DHCP协议采用UDP作为传输协议，DHCP客户端发送请求消息到DHCP服务器的68号端口，DHCP服务器回应应答消息给DHCP客户端的67号端口。
>
> 只有跟DHCP客户端在同一个网段的DHCP服务器才能收到DHCP客户端广播的DHCP DISCOVER报文。当DHCP客户端与DHCP服务器不在同一个网段时，必须部署DHCP中继来转发DHCP客户端和DHCP服务器之间的DHCP报文。在DHCP客户端看来，DHCP中继就像DHCP服务器；在DHCP服务器看来，DHCP中继就像DHCP客户端。
>
> ### 无中继场景时**DHCP**客户端首次接入网络的工作原理
>
> 如下图所示，在没有部署DHCP中继的场景下，首次接入网络DHCP客户端与DHCP服务器的报文交互过程，该过程称为DHCP报文四步交互。



有中继场景时DHCP客户端首次接入网络的工作原理

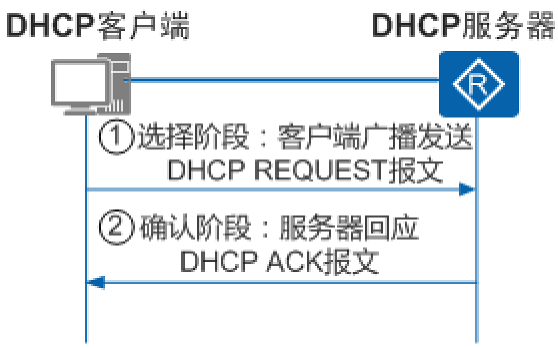
有DHCP中继的场景中，首次接入网络的DHCP客户端和DHCP服务器的工作原理与无中继场景时**DHCP**客户端首次接入网络的工作原理相同。主要差异是DHCP中继在DHCP服务器和DHCP客户端之间转发DHCP报文，以保证DHCP服务器和DHCP客户端可以正常交互。下面仅针对DHCP中继的工作原理进行介绍。

如下图所示，在部署DHCP中继的场景下，首次接入网络DHCP客户端与DHCP服务器的报文交互过程。



DHCP客户端重用曾经使用过的地址的工作原理

DHCP客户端非首次接入网络时，可以重用曾经使用过的地址。如下图所示，DHCP客户端与DHCP服务器交互DHCP报文，以重新获取之前使用的IP地址等网络参数，该过程称为两步交互。

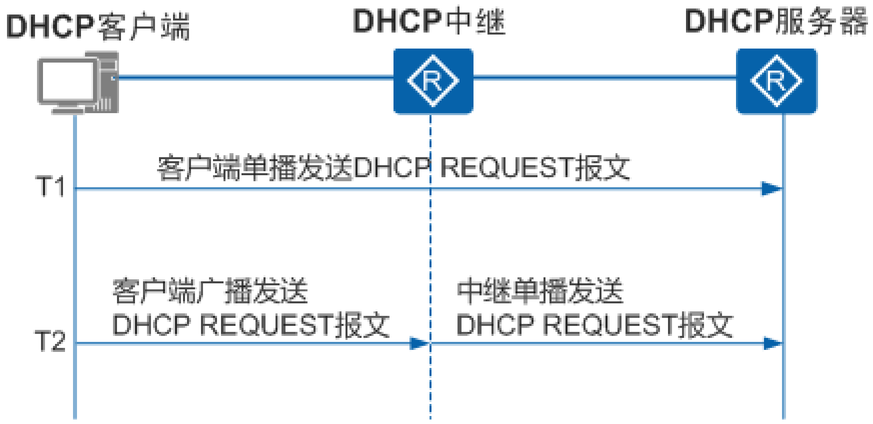


DHCP客户端更新租期的工作原理

DHCP服务器采用动态分配机制给客户端分配IP地址时，分配出去的IP地址有租期限制。DHCP客户端向服务器申请地址时可以携带期望租期，服务器在分配租期时把客户端期望租期和地址池中租期配置比较，分配其中一个较短的租期给客户端。租期到期或者客户端下线释放地址后，服务器会收回该IP地址，收回的IP地址可以继续分配给其他客户端使用。这种机制可以提高IP地址的利用率，避免客户端下线后IP地址继续被占用。如果DHCP客户端希望继续使用该地址，需要更新IP地址的租期（如延长IP地址租期）。

DHCP客户端更新租期的过程如下图所示。

如下图所示，部署 DHCP 中继时，更新租期的过程与上述过程相似。



DHCP 使用场景

DHCP 提供了两种地址分配机制，网络管理员可以根据网络需求为不同的主机选择不同的分配策略。

- 动态分配机制：通过 DHCP 为主机分配一个有使用期限的 IP 地址。

  DHCP 使用了租期的概念，或称为设备 IP 地址的有效期。租用时间是不定的，主要取决于用户在某地连接 Internet 需要多久，这种分配机制适用于主机需要临时接入网络或者空闲地址数小于网络主机总数且主机不需要永久连接网络的场景。

- 静态分配机制：网络管理员通过 DHCP 为指定的主机分配固定的 IP 地址。

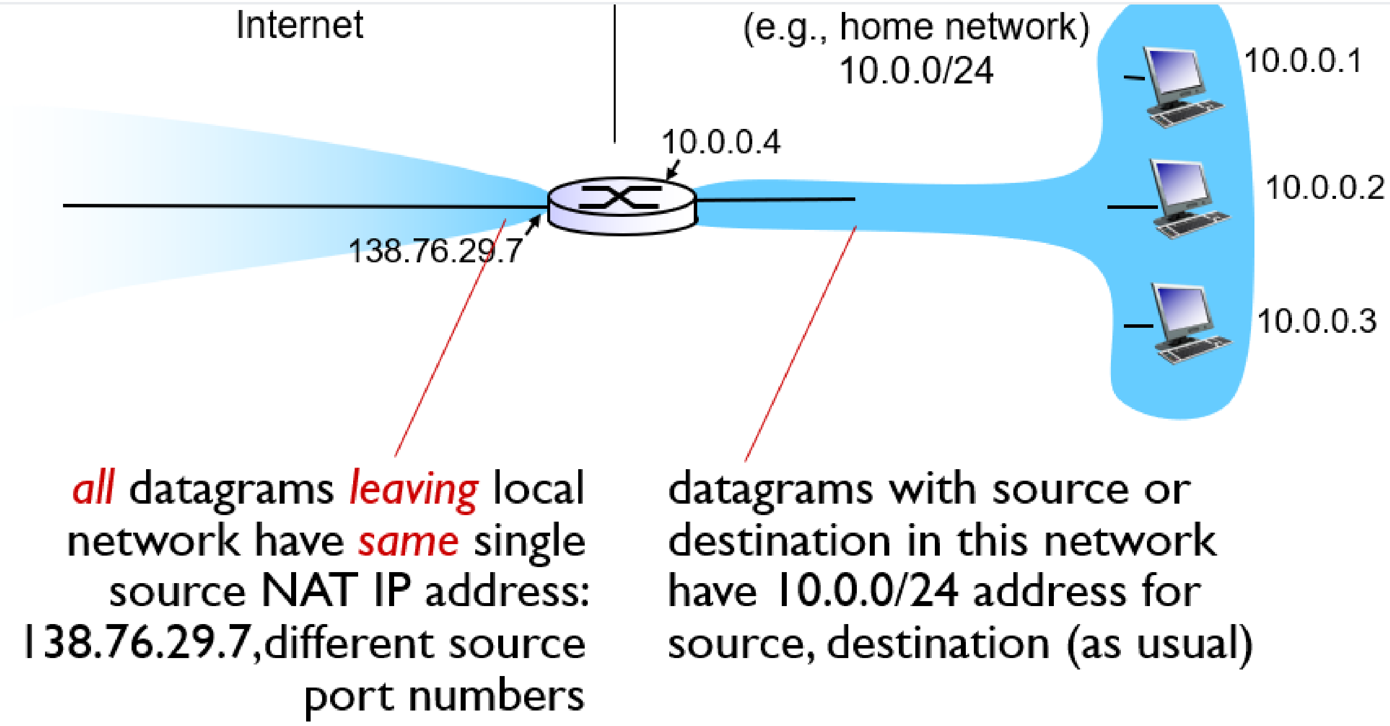
  相比手工静态配置 IP 地址，通过 DHCP 方式静态分配机制避免人工配置发生错误，方便管理员统一维护管理。

建议阅读：动态主机配置协议 (DHCP) | Microsoft Learn (https://learn.microsoft.com/zh-cn/windows-server/networking/technologies/dhcp/dhcp-top)

network address translation

网络地址转换（**Network Address Translation，NAT**）：就外界而言，本地网络中的所有设备只共享一个 IPv4 地址。

NAT 具体步骤如下图：



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

> 什么是 NAT? NAT 的类型有哪些? - 华为 (huawei.com) (https://info.support.huawei.com/info-finder/encyclopedia/zh/NAT.html)
>
> ### 什么是 NAT?
>
> NAT 是一种地址转换技术，它可以将 IP 数据报文头中的 IP 地址转换为另一个 IP 地址，并通过转换端口号达到地址重用的目的。NAT 作为一种缓解 IPv4 公网地址枯竭的过渡技术，由于实现简单，得到了广泛应用。
>
> ### NAT 解决了什么问题?
>
> 随着网络应用的增多，IPv4 地址枯竭的问题越来越严重。尽管IPv6 (https://info.support.huawei.com/info-finder/encyclopedia/zh/IPv6.html) 可以从根本上解决 IPv4 地址空间不足问题，但目前众多网络设备和网络应用大多是基于 IPv4 的，因此在 IPv6 广泛应用之前，使用一些过渡技术（如 CIDR、私网地址等）是解决这个问题的主要方式，NAT 就是这众多过渡技术中的一种。
>
> 当私网用户访问公网的报文到达网关设备后，如果网关设备上部署了 NAT 功能，设备会将收到的 IP 数据报文头中的 IP 地址转换为另一个 IP 地址，端口号转换为另一个端口号之后转发给公网。在这个过程中，设备可以用同一个公网地址来转换多个私网用户发过来的报文，并通过端口号来区分不同的私网用户，从而达到地址复用的目的。
>
> 早期的 NAT 是指 Basic NAT，Basic NAT 在技术上实现比较简单，只支持地址转换，不支持端口转换。因此，Basic NAT 只能解决私网主机访问公网问题，无法解决 IPv4 地址短缺问题。后期的 NAT 主要是指网络地址端口转换 NAPT（Network Address Port Translation），NAPT 既支持地址转换也支持端口转换，允许多台私网主机共享一个公网 IP 地址访问公网，因此 NAPT 才可以真正改善 IP 地址短缺问题。
>
> ### NAT 的类型
>
> 根据 NAT 转换是对报文中的源地址进行转换还是对目的地址进行转换，NAT 可以分为源 NAT、目的 NAT 和双向 NAT，下面我们分别介绍这三种 NAT 类型。
>
> #### 源 NAT
>
> 源 NAT 在 NAT 转换时，仅对报文中的源地址进行转换，主要应用于私网用户访问公网的场景。当私网用户主机访问 Internet 时，私网用户主机发送的报文到达 NAT 设备后，设备通过源 NAT 技术将报文中的私网 IPv4 地址转换成公网 IPv4 地址，从而使私网用户可以正常访问 Internet。
>
> #### 目的 NAT

双向 NAT

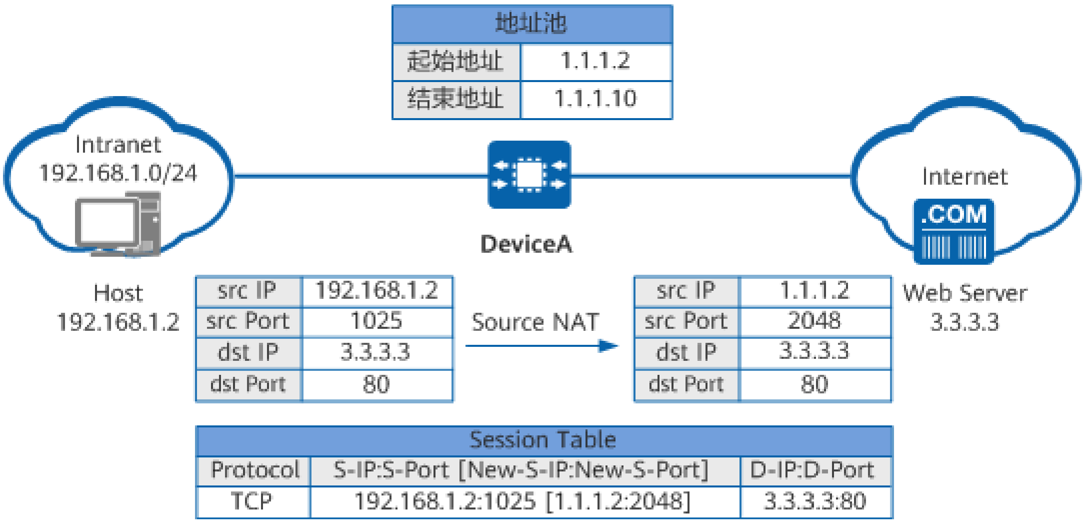
双向 NAT 指的是在转换过程中同时转换报文的源信息和目的信息。双向 NAT 不是一个单独的功能，而是源 NAT 和目的 NAT 的组合。双向 NAT 是针对同一条流，在其经过设备时同时转换报文的源地址和目的地址。双向 NAT 主要应用在同时有外网用户访问内部服务器和私网用户访问内部服务器的场景。

NAT 是如何工作的?

根据前面的分类，我们分别从源 NAT 和目的 NAT 中各选一种 NAT 为代表，介绍其工作原理。其他类型的 NAT 虽然在转换时，转换的内容有细微差别，但是工作原理都相似，不再重复介绍。此外，双向 NAT 是源 NAT 和目的 NAT 的组合，双向 NAT 的工作原理也不再重复介绍。

NAPT 工作原理

NAPT 在进行地址转换的同时还进行端口转换，可以实现多个私网用户共同使用一个公网 IP 地址上网。NAPT 根据端口来区分不同用户，真正做到了地址复用。



地址池	
起始地址	1.1.1.2
结束地址	1.1.1.10

src IP	192.168.1.2
src Port	1025
dst IP	3.3.3.3
dst Port	80

src IP	1.1.1.2
src Port	2048
dst IP	3.3.3.3
dst Port	80

Session Table		
Protocol	S-IP:S-Port [New-S-IP:New-S-Port]	D-IP:D-Port
TCP	192.168.1.2:1025 [1.1.1.2:2048]	3.3.3.3:80

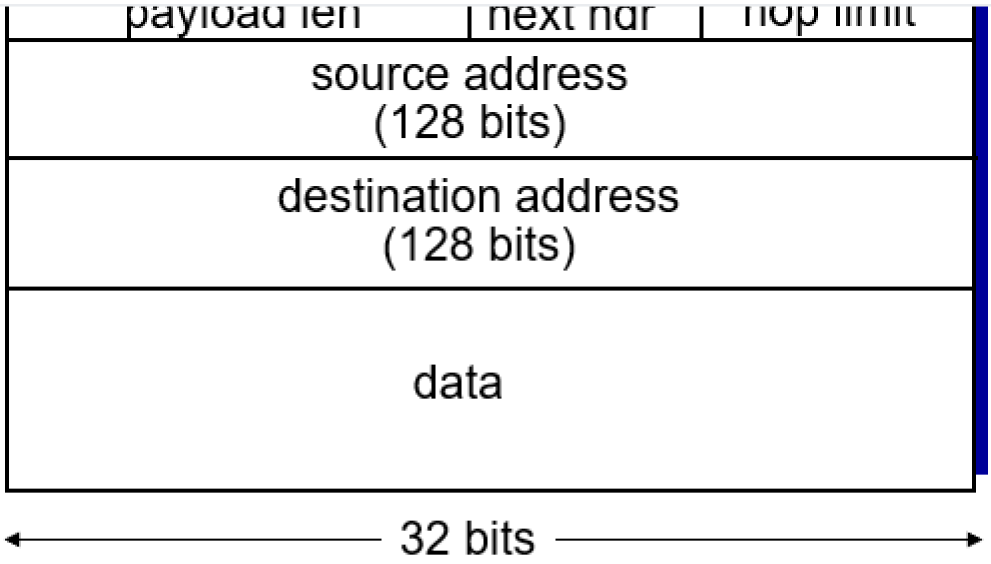
1. 设备收到 Host 发送的报文后查找 NAT 策略，发现需要对报文进行地址转换。
2. 设备根据源 IP Hash 算法从 NAT 地址池中选择一个公网 IP 地址，替换报文的源 IP 地址，同时使用新的端口号替换报文的源端口号，并建立会话表，然后将报文发送至 Internet。
3. 设备收到 Web Server 响应 Host 的报文后，通过查找会话表匹配到步骤 2 中建立的表项，将报文的目的地址替换为 Host 的 IP 地址，将报文的目的端口号替换为原始的端口号，然后将报文发送至 Intranet。

IPv6

产生动机：

- 将完全分配 32 位 IPv4 地址空间；
- 速度处理/转发：40 字节固定长度首部；
- 启用流标签的不同网络层处理。

报文格式：



- *priority(优先级):* identify priority among datagrams in flow
- *flow Label(流标签):* identify datagrams in same “flow.” (concept of“flow” not well defined “流”的概念没有很好地定义).
- *next header(下一个标头):* identify upper layer protocol(上层协议) for data

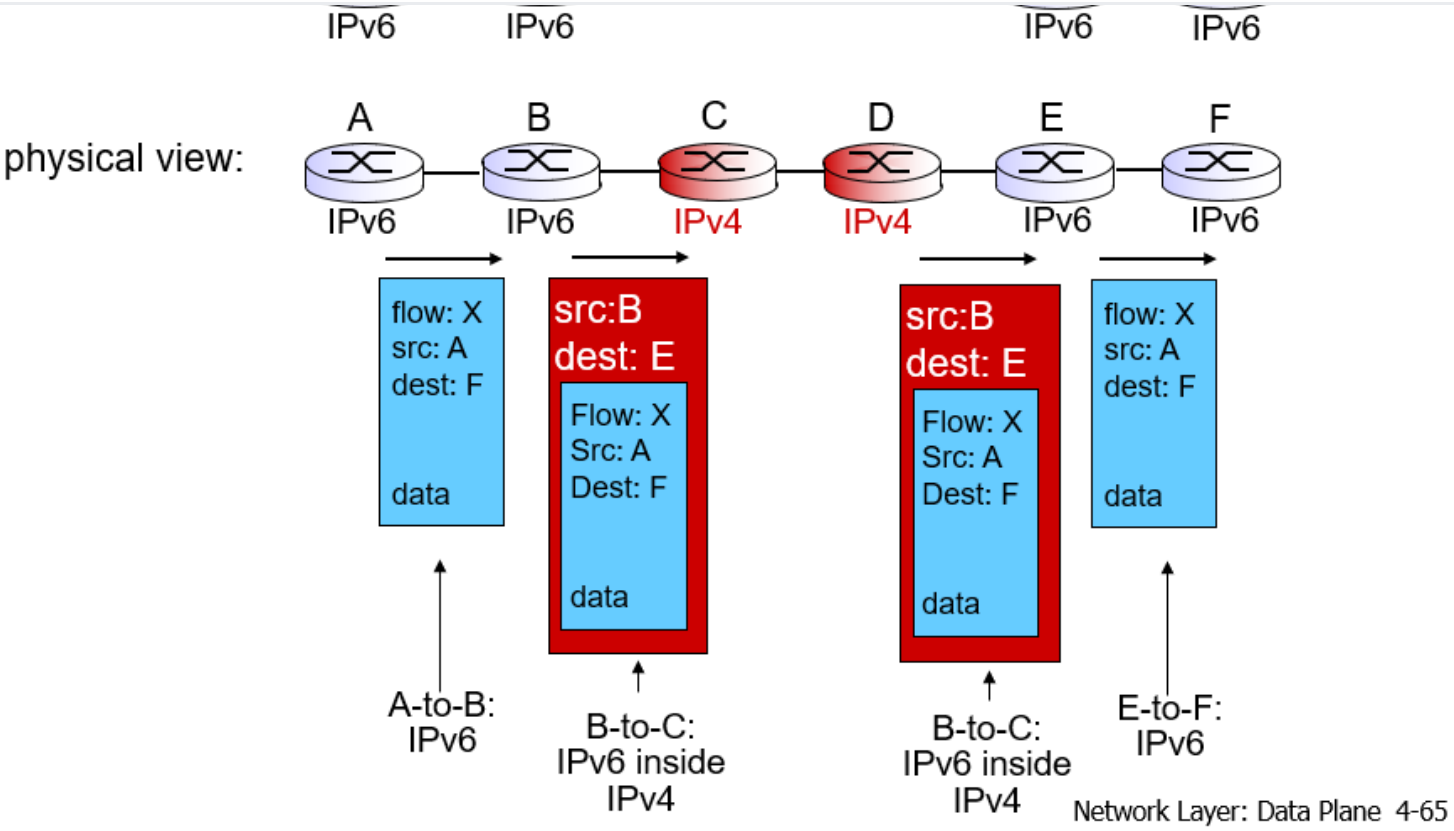
> - 删掉分片/重新组装;
> - 取消首部检验和;
> - 将变长选项字段由下一个首部指出。

Other changes from IPv4:

- checksum: removed entirely to reduce processing time at each hop(每个跃点)
- options: allowed, but outside of header(标头之外), indicated(指示) by “Next Header” field
- ICMPv6: new version of ICMP
- additional message types(其他消息类型), e.g. “Packet Too Big”
- multicast group management functions 多播组管理功能

IPv4 到 IPv6 的过渡

建隧道（**tunneling**）如下图:



Generalized Forward and SDN

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

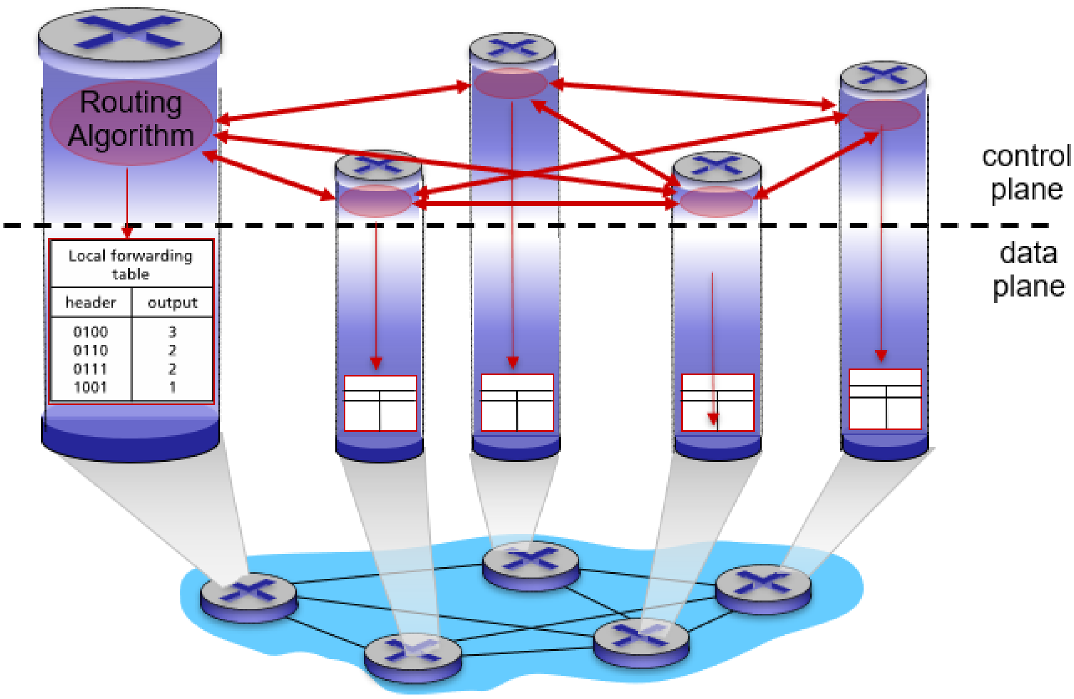
Outline

1. introduction
2. routing protocols
3. intra-AS routing in the Internet: OSPF
4. routing among the ISPs: BGP
5. The SDN control plane
    - link state
    - distance vector
6. ICMP: The Internet Control Message Protocol
7. Network management and SNMP

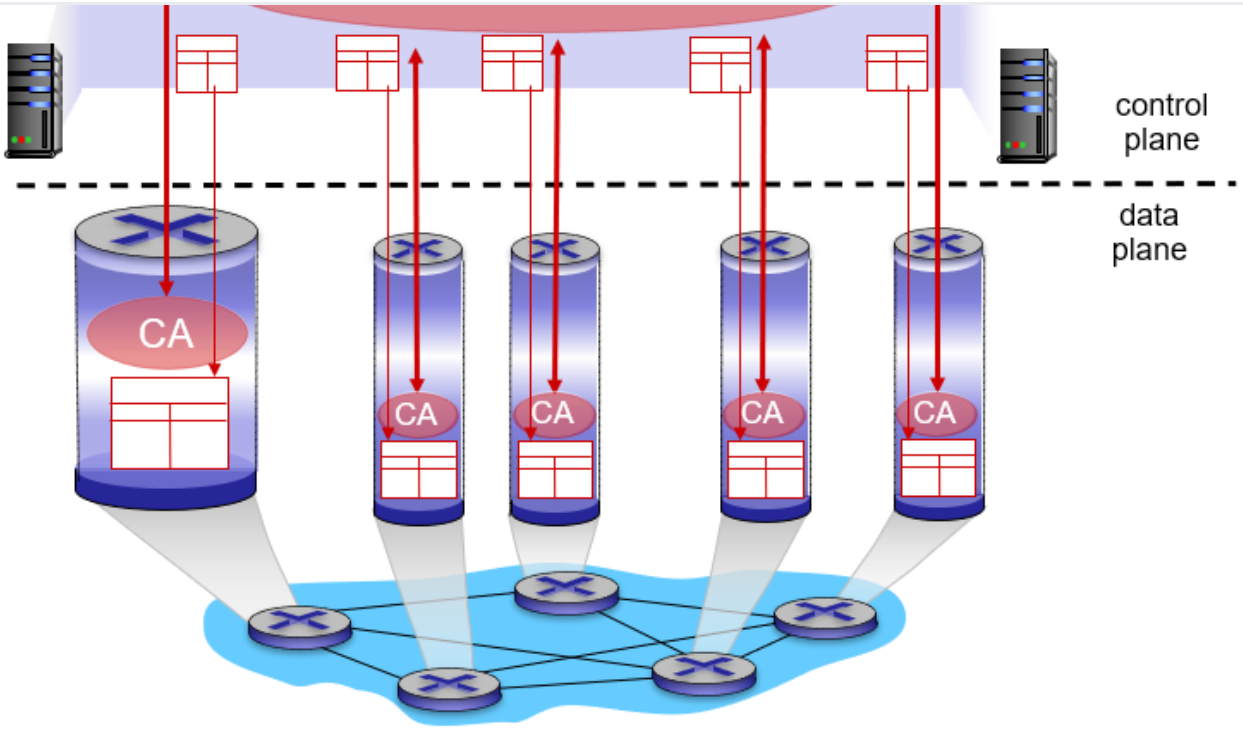
> 部分内容参考自【计算机网络-自顶向下】5—Network Layer:Control Plane 网络层：控制平面（概述、路由选择算法、OSPF、BGP、SDN、ICMP、SNMP）_一棵____大树的博客-CSDN博客 (https://blog.csdn.net/weixin_53580595/article/details/129482346)，有删改。

Introduction

- 每个路由器控制（**Per-router control**）：每台路由器有一个路由选择组件，用于与其他路由器中的路由选择组件通信，以计算其转发表的值。如下图所示：



- 逻辑集中式控制（**Logically centralized control**）：逻辑集中式控制器计算并分发转发表以供每台路由器使用，如下图所示：

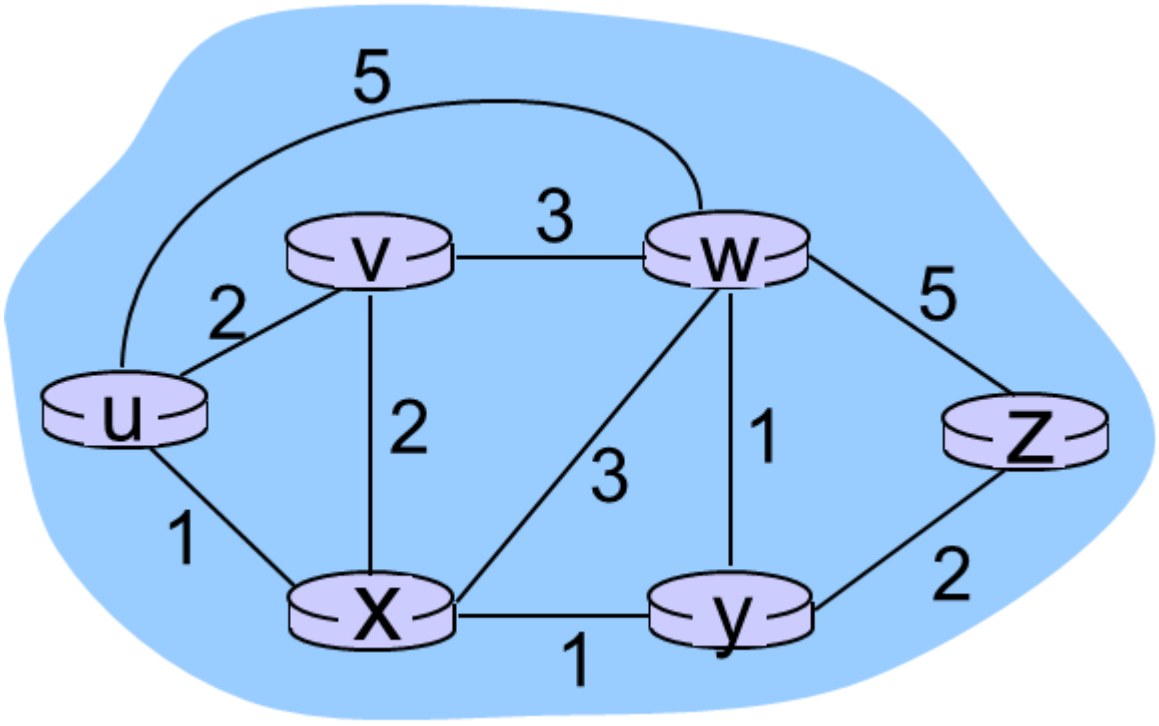


Routing protocols

goal: determine(□□) “good” paths (equivalently, routes □□□□□□), from sending hosts to receiving host, through network of routers

Introduction of Routing protocols

Graph abstraction of the network



- graph: $G = (N,E)$
- $N = \text{set of routers} = { u, v, w, x, y, z }$
- $E = \text{set of links} = { (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }$
- $c(x,x') = \text{cost of link}\ (x,x')$

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

Routing algorithm

即路由选择算法。

分类上，从 global or decentralized information:

- *global:* all routers have complete topology, link cost info 都有完整的拓扑结构，链路成本信息
- *decentralized:* router knows physically-connected neighbors, link costs to neighbors 路由器知道物理连接的邻居，与邻居的链路成本

  iterative process of computation, exchange of info with neighbors 计算的迭代过程，与邻居交换信息

从 static or dynamic:

- *static*: routes change slowly over time
- *dynamic*: routes change more quickly

  periodic update 定期更新 in response to link cost changes 响应链接成本总是保持变化的

算法上：

- 集中式路由选择算法（centralized routing algorithm）：用完整、全局性的网络知识计算出从源到目的地之间的最低开销路径。具有全局状态信息的算法常被称作链路状态（Link State，LS）算法，因为该算法必须知道网络中每条链路的开销。
- 分散式路由选择算法（decentralized routing algorithm）：路由器以迭代、分布式计算的方式计算出最低开销路径。没有节点拥有关于网络链路开销的完整信息。一个分散式路由选择算法为距离向量（Distance-Vector，DV）算法，每个节点维护到网络中所有其他节点的开销估计的向量。

Link state

在实践中，这经常由链路状态广播（link state broadcast）算法完成。下面给出的链路状态路由选择算法叫做 Dijkstra 算法，其计算从某节点（源节点 u）到网络中所有其他节点的最低开销路径。

基本思想：经算法的第 k 次迭代后，可知道到 k 个目的节点的最低开销路径，在到所有目的节点的最低开销路径之中，这 k 条路径具有 k 个最低开销。

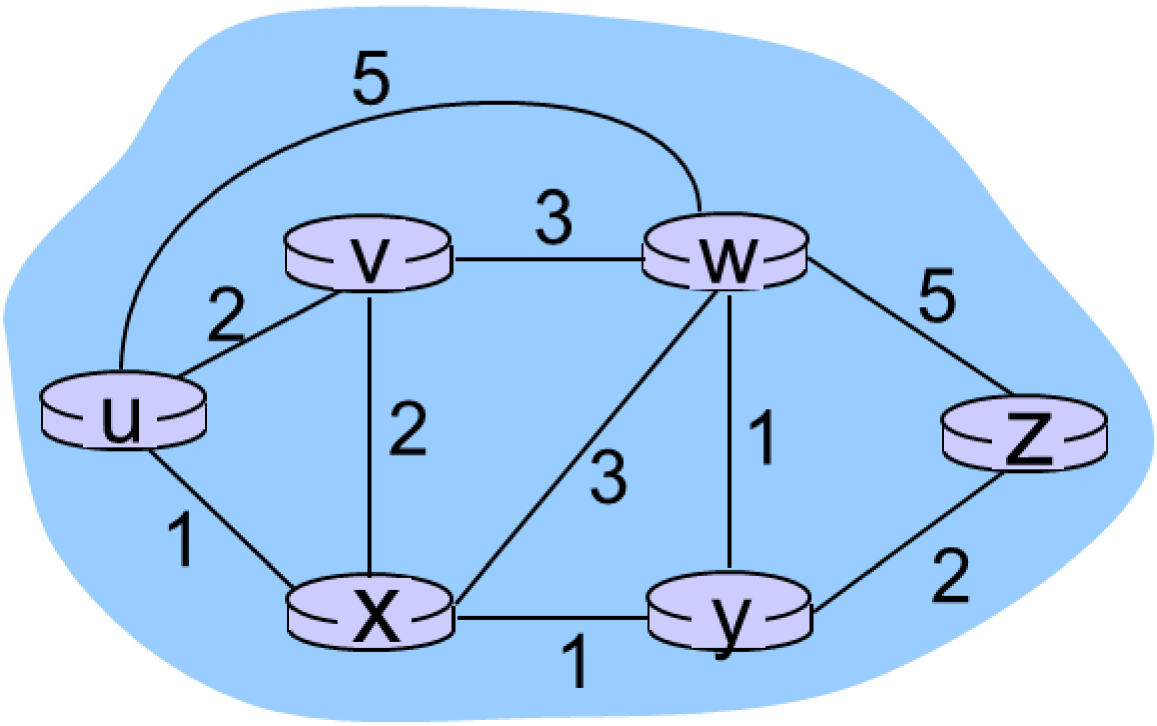
我们定义如下符号：

- N' ：节点子集；如果从源到 v 的最低开销路径已经确定，v 在 N' 中。

源节点 u 的链路状态 (LS) 算法如下：

```py
# Initialization:
N' = {u}
for all nodes v
if v adjacent to u
    then D(v) = c(u,v)
else D(v) = ∞

# Loop
find w not in N' such that D(w) is a minimum
add w to N'
update D(v) for all v adjacent to w and not in N' :
    D(v) = min( D(v), D(w) + c(w,v) )
/* new cost to v is either old cost to v or known
shortest path cost to w plus cost from w to v */
# until all nodes in N'
```

对于下图的网络，链路状态算法迭代运行如下：



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

3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

路线上的流量变化和拥塞会使LS算法产生路由震荡（**Routing Oscillations**）。

Distance vector

距离向量（Distance-Vector，DV）算法是一种迭代的（iterative）、异步的（asynchronous）、分布式的（distributed）和自我终止的（self-termination）。

> **TIP**
>
> 这种方法没有组建图，只通过方程不断迭代更新数据，收集了与邻居之间的信息。

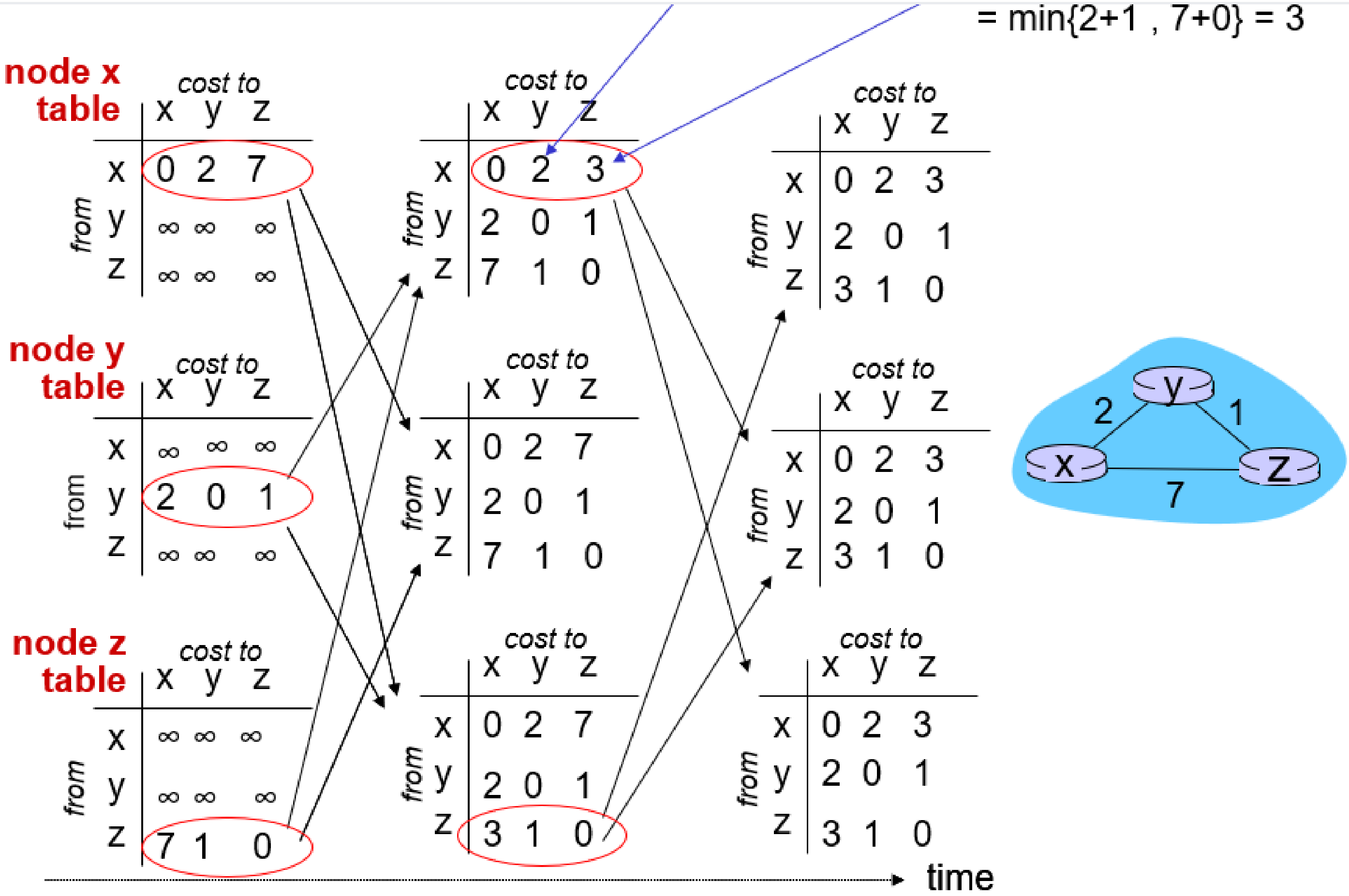
令$d_x(y)$是从节点x到节点y的最低开销路径。则该最低开销与著名的Bellman-Ford方程相关，即：

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

方程中的$\min_v$是对于x的所有邻居的。

算法基本思想：每个节点x以$D_x(y)$开始，对在N中的所有节点y，估计从x到y的最低开销路径的开销路径。

DV算法具体实例如下：



node x table

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

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

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

node y table

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

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

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

node z table

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

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

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

链路开销改变与链路故障

链路开销发生改变：

- node detects local link cost change 节点检测本地链路成本变化
- updates routing info, recalculates distance vector 重新计算距离矢量
- if DV changes, notify neighbors

有时当链路中有一条路的距离极长甚至接近于开路，这时我们会遇到路由选择环路（**routing loop**），这样的问题也被称为**无穷计数（count-to-infinity）**问题。

距离向量算法：增加毒性逆转

毒性逆转（poisoned reverse），思想：如果 z 通过 y 路由选择目的地 x，则 z 将通告 y，它到 x 的距离是无穷大，也就是 z 将通告：

$$ D_z(x)=\infin $$

但毒性逆转并不能解决一般的无穷计数问题。

Comparison of LS and DV algorithms

message complexity 消息复杂性：

- DV: exchange between neighbors only

  convergence time varies 收敛时间不同

speed of convergence 收敛速度:

- LS: $O(n^2)$ algorithm requires $O(nE)$ msgs

  may have oscillations 可能有震荡

- DV: convergence time varies

  may be routing loops 可能是路由环路

  count-to-infinity problem 计数无穷大问题

> **robustness: what happens if router malfunctions?**
>
> 稳健性：如果路由器出现故障会怎样?
>
> LS:
>
> - node can advertise incorrect link cost 节点可以通告不正确的链接成本
> - each node computes only its own table 每个节点仅计算自己的表
>
> DV:
>
> - DV node can advertise incorrect path cost DV 节点可以通告不正确的路径成本
> - each node’s table used by others 其他节点使用的每个节点的表
> - error propagate thru network 错误也会通过网络传播

Intra-AS routing in the Internet: OSPF

因特网中自治系统内部的路由选择：OSPF（开放最短路优先 Open Shortest Path First）

随着路由器规模增大和管理自治的要求，可以通过将路由器组织进自治系统（Autonomous System，AS）来解决。在一个自治系统内运行的路由算法叫做自治系统内部路由选择协议（intra-autonomous system routing protocol）。

> 市面上的常见路由协议
>
> 基于 DV:
>
> - RIP
>
> - IGRP

> - OSPF（主流）
> - IS-IS

OSPF 是一种链路状态协议，它使用洪泛链路状态信息和 Dijkstra 最低开销路径算法。使用 OSPF，一台路由器构建了一幅关于整个自治系统的完整拓扑图。于是，每台路由器在本地运行 Dijkstra 的最短路径算法，以确定一个自身为根节点到所有子网的最短路径树。

使用 OSPF 时，路由器向自治系统内所有其他路由器广播路由选择信息，而不仅仅是向其相邻路由器广播。每当一条链路的状态发生变化时，路由器就会广播链路状态信息。

OSPF 的优点：

- 安全（Security）：能够鉴别 OSPF 路由器之间的交换；
- 多条相同开销的路径（Multiple same-cost paths）：允许使用多条路径；
- 对单播与多播路由选择的综合支持（Integrated support for unicast and multicast routing）；
- 支持在单个 AS 中的层次结构（Support for hierarchy within a single AS）。

routing among the ISPs: BGP

当分组跨越多个 AS 进行路由时，我们需要一个自治系统间路由协议（**inter-autonomous system routing protocol**）。在因特网中，所有的 AS 运行相同的 AS 间路由选择协议，称为边界网关协议（**Broder Gateway Protocol**，**BGP**）。

BGP provides each AS a means to:

- **eBGP**: obtain subnet reachability information from neighboring ASes. 从相邻 AS 获取子网可访问性信息
- **iBGP**: propagate reachability information to all AS-internal routers. 将可达性信息传播到所有 AS 内部路由器

and it determines “good” routes to other networks based on reachability information and *policy*(可达性信息和策略)

BGP 的作用

在 BGP 中，分组并不是路由到一个特定的目的地址，相反是路由到 CIDR 化的前缀，其中每个前缀表示一个子网或者一个子网集合。

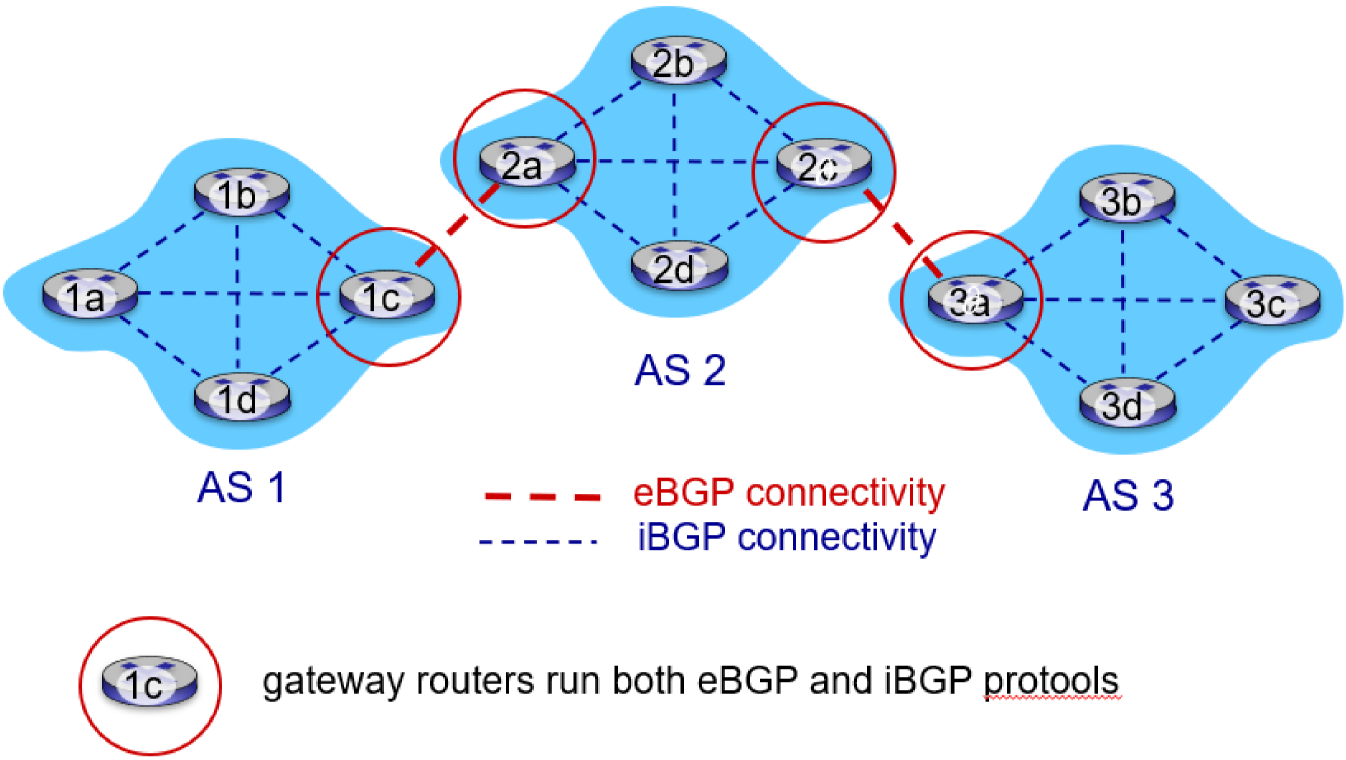
BGP 为每台服务器提供完成以下任务的手段：

- 从邻居 AS 获得前缀的可达性信息 Obtain prefix reachability information from neighboring ASs；
- 确定到该前缀的“最好”的路由 Determine the “best” routes to the prefixes.

eBGP, iBGP connections

通告 BGP 路由信息

两种连接如下图所示：



BGP 会话：

两个 BGP 路由器（“对等方”）通过半永久 TCP 连接交换 BGP 消息：通告到不同目标网络前缀的路径（BGP 是一种“path vector”协议）

BGP messages

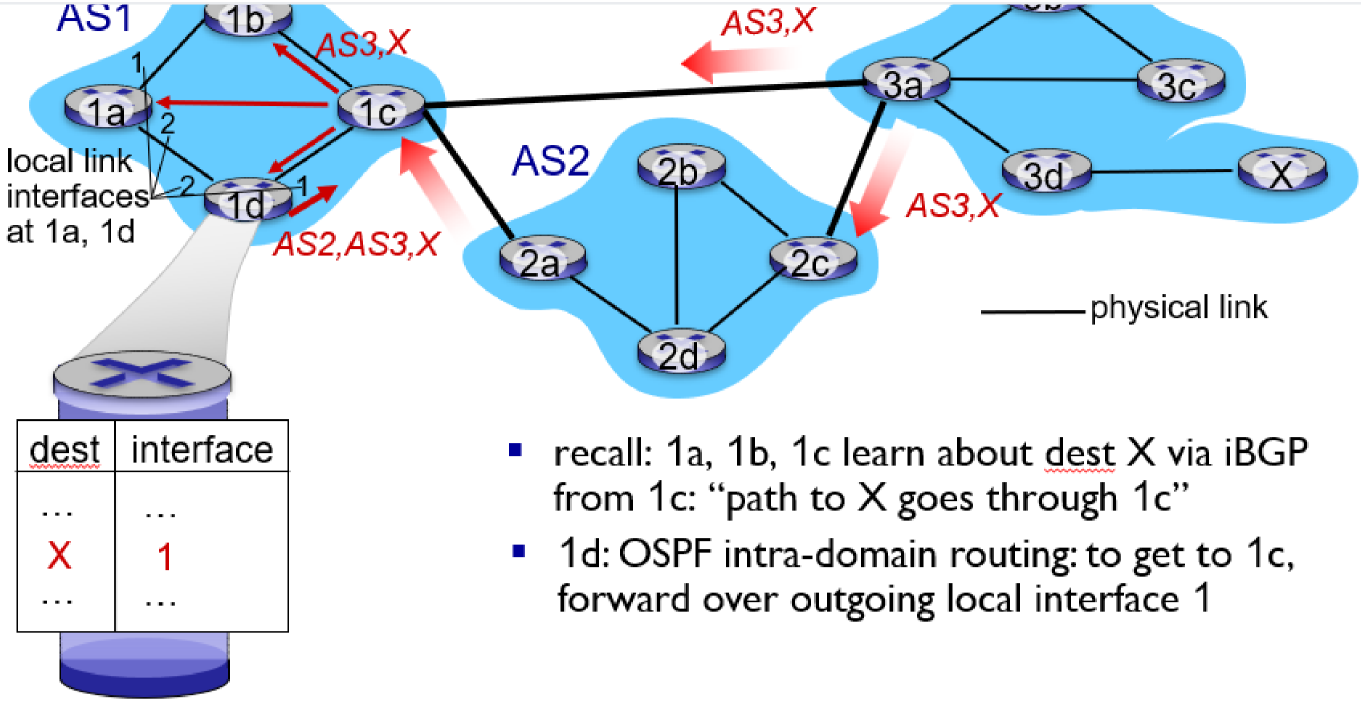
BGP messages exchanged between peers over TCP connection.

BGP messages:

- `OPEN` : opens TCP connection to remote BGP peer(与远程 BGP 对等体) and authenticates sending BGP peer
- `UPDATE` : advertises new path (or withdraws old 撤回旧路径)
- `KEEPALIVE` : keeps connection alive in absence of `UPDATES` (没有更新的情况下); also ACKs `OPEN` request(确认打开请求)
- `NOTIFICATION` : reports errors in previous msg; also used to close connection

BGP, OSPF, forwarding table entries

How does router set forwarding table entry to distant prefix?



Path attributes and BGP routes

advertised prefix includes BGP attributes 播发的前缀其实包括 BGP 属性:

$$ \text{prefix} + \text{attributes} = \text{“route”} $$

其中 two important attributes:

- `AS-PATH:` list of ASes through which prefix advertisement has passed
- `NEXT-HOP` : indicates(指示) specific internal-AS router to next-hop(下一跃点) AS

Determining the Best Routes

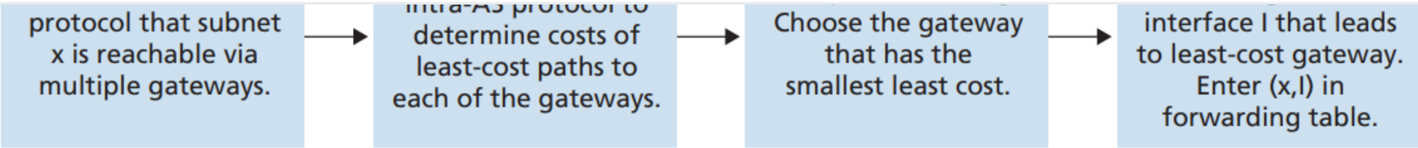
通告前缀包括一些 BGP 属性（BGP attribute），前缀及其属性称为路由（route）。

热土豆路由选择

hot potato routing

热土豆路由选择的基本思想是：对于一个路由，尽可能快地将分组送出其 **AS**，而不必担心其 **AS** 外部到目的地的余下部分的开销。

在路由器转发表中增加 AS 外部目的地的步骤：



路由选择算法

Route-Selection Algorithm

如果到相同前缀有两条或多条路由，则顺序地调用下列消除规则知道余下一条路由：

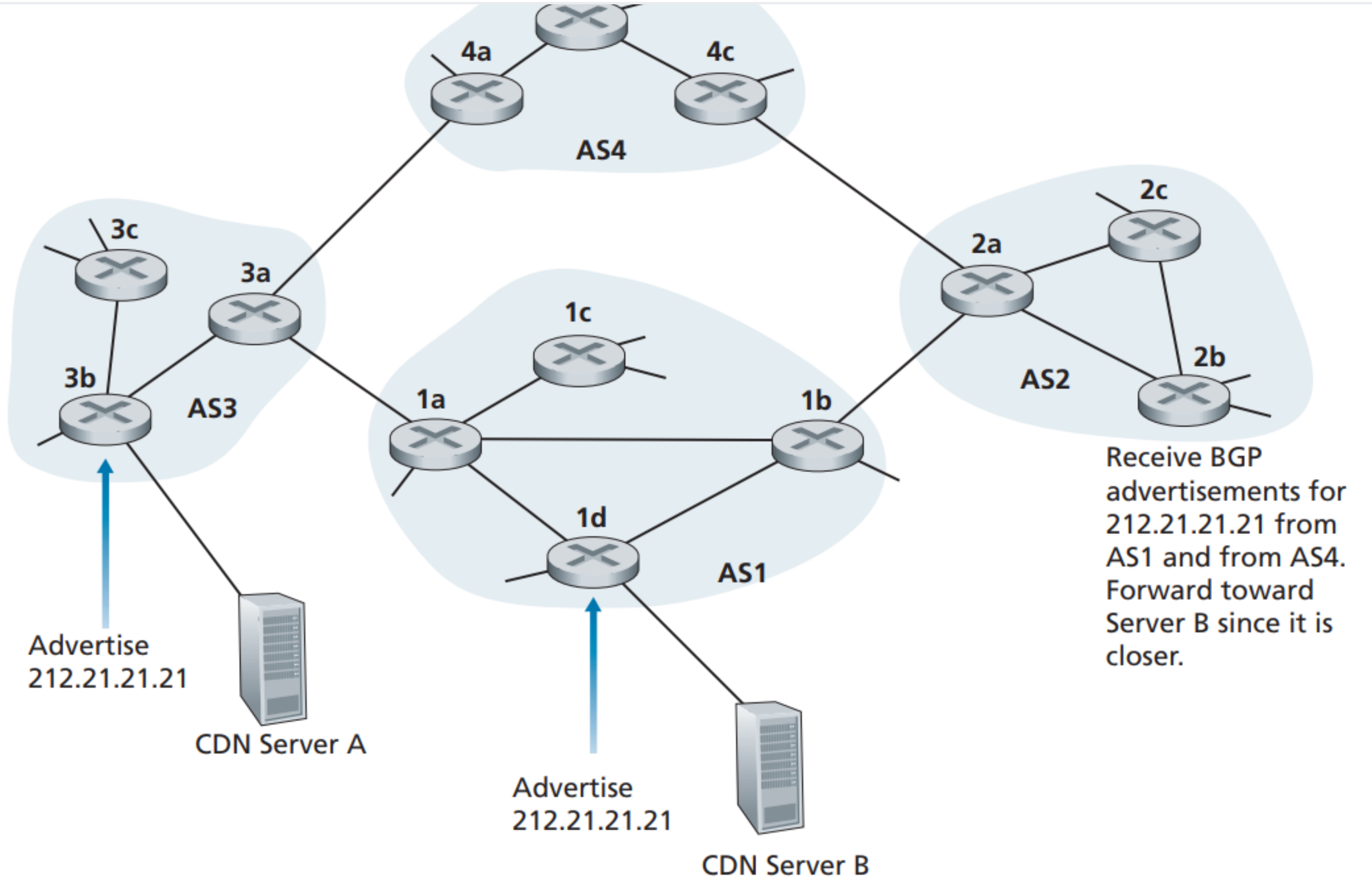
1. 路由被指派一个本地偏好（local preference）值作为其属性值之一，具有最高本地偏好值的将被选择；
2. 从余下的路由中，将选择具有最短 `AS-PATH` 的路由；
3. 从余下的路由中，使用热土豆路由选择，即选择具有最靠近 `NEXT-HOP` 路由器的路由；
4. 如果仍留下多条路由，该路由器使用 BGP 标识符来选择路由。

IP 任播

IP 任播的动机：

1. 在许多分散的不同地理位置，替换不同服务器上的相同内容；
2. 让每个用户从最靠近的服务器访问内容。

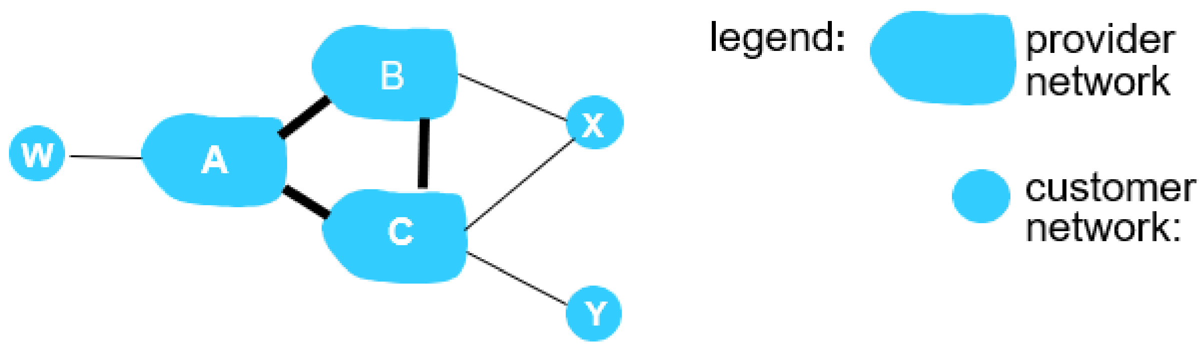
使用 IP 任播将用户引向最近的 CDN 服务器如下图：



路由选择策略

Routing Policy

如下图所示，X 是一个多宿主接入 **ISP（multi-homed stub network）**，因为它是经由两个不同的提供商连到网络的其余部分。



- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C

ISP 遵循的法则：任何穿越某 ISP 主干网的流量必须是其源或者目的位于该 ISP 的某个客户网络中；不然的话这些流量将会免费搭车通过该 ISP 的网络。

区分 AS 间和 AS 内部路由选择的原因：

- 策略（Policy）；

The SDN control plane

Software defined networking (SDN)

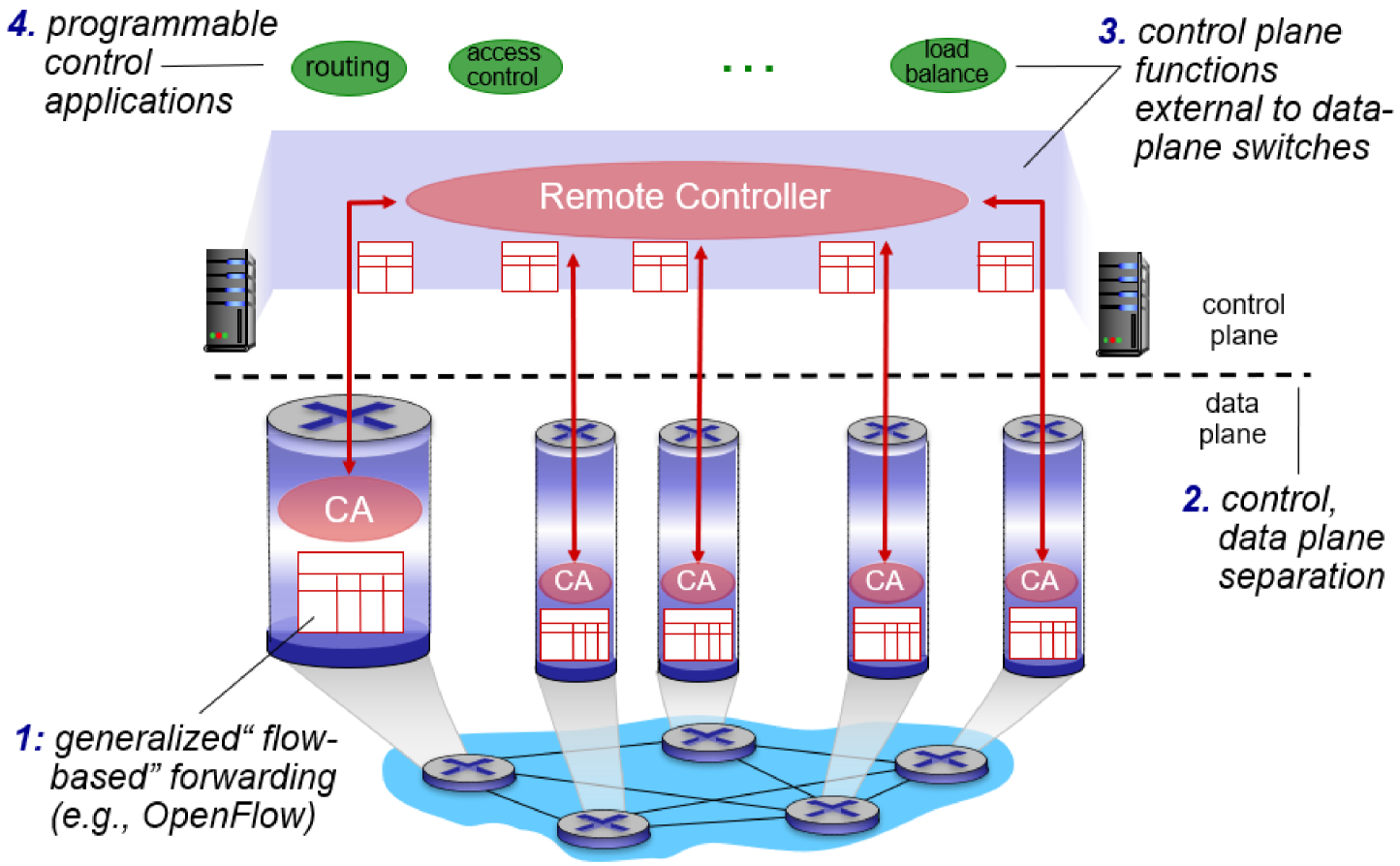
Internet network layer: historically has been implemented via distributed, per-router approach(历史上一直通过分布式、每路由器方法实现), and till 2005: renewed interest in rethinking network control plane(重新审视网络控制平面的兴趣)

Why a logically centralized control plane? 为什么选择逻辑集中的控制平面?

- 更轻松的网络管理：避免路由器配置错误，提高流量灵活性
- 基于表的转发（回想一下 OpenFlow API）允许"编程"路由器
- 控制平面（control plane）的开放（非专有）实现

> What is Software-Defined Networking? (ibm.com) (https://www.ibm.com/topics/sdn)
>
> ### What is SDN?
>
> SDN is an approach to networking that uses software controllers that can be driven by application programming interfaces (APIs) to communicate with hardware infrastructure (https://www.ibm.com/topics/infrastructure) to direct network traffic. Using software, it creates and operates a series of virtual overlay networks that work in conjunction with a physical underlay network. SDNs offer the potential to deliver application environments as code and minimize the hands-on time needed for managing the network. SDN 是一种网络方法，它使用可由应用程序编程接口 (API) 驱动的软件控制器与硬件基础设施通信以引导网络流量。它使用软件创建和运行一系列与物理底层网络协同工作的虚拟覆盖网络。 SDN 提供了将应用程序环境作为代码交付的潜力，并最大限度地减少了管理网络所需的动手时间。
>
> ### Why use SDN?
>
> Companies today are looking to SDN to bring the benefits of the cloud to network deployment and management. With network virtualization, organizations can open the door to greater efficiency through new tools and technology, such as Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS (https://www.ibm.com/cloud/learn/iaas)) and other cloud computing services, as well as integrate via APIs with their software-defined network. 当今的公司希望通过 SDN 将云的优势带入网络部署和管理中。借助网络虚拟化，组织可以通过软件即服务 (SaaS)、基础设施即服务 (IaaS) 和其他云计算服务等新工具和技术打开提高效率的大门，以及通过 API 与其软件定义网络集成。
>
> SDN also increases visibility and flexibility. In a traditional environment, a router or switch—whether in the cloud or physically in the data center—is only aware of the status of network devices next to it. SDN centralizes this information so that organizations can view and control the entire network and devices. Organizations can also segment different virtual networks within a single physical network or connect different physical networks to create a single virtual network, offering a high degree of flexibility. SDN 还提高了可见性和灵活性。在传统环境中，路由器或交换机———无论是在云端还是物理上在数据中心———只知道它旁边的网络设备的状态。SDN 集中了这些信息，以便组织可以查看和控制整个网络和设备。组织还可以在单 个物理网络内分割不同的虚拟网络，或连接不同的物理网络以创建单个虚拟网络，从而提供高度的灵活性。

SDN 体系结构如下图所示:



SDN 体系结构具有四个关键特征:

- 基于流的转发（Flow-based forwarding）;
- 数据平面与控制平面分离（Separation of data plane and control plane）;
- 网络控制功能（Network control functions）;
- 可编程网络（A programmable network）。

SDN Controller

- maintain network state information 维护网络状态信息
- interacts with network control applications(控制应用程序进行交互) “above” via northbound API
- interacts with network switches(与网络交换机交互) “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness 作为分布式系统实现，以实现性能、可扩展性、容错性和健壮性

OpenFlow protocol

three classes of OpenFlow messages:

- controller-to-switch
- asynchronous (switch to controller)
- symmetric (misc)

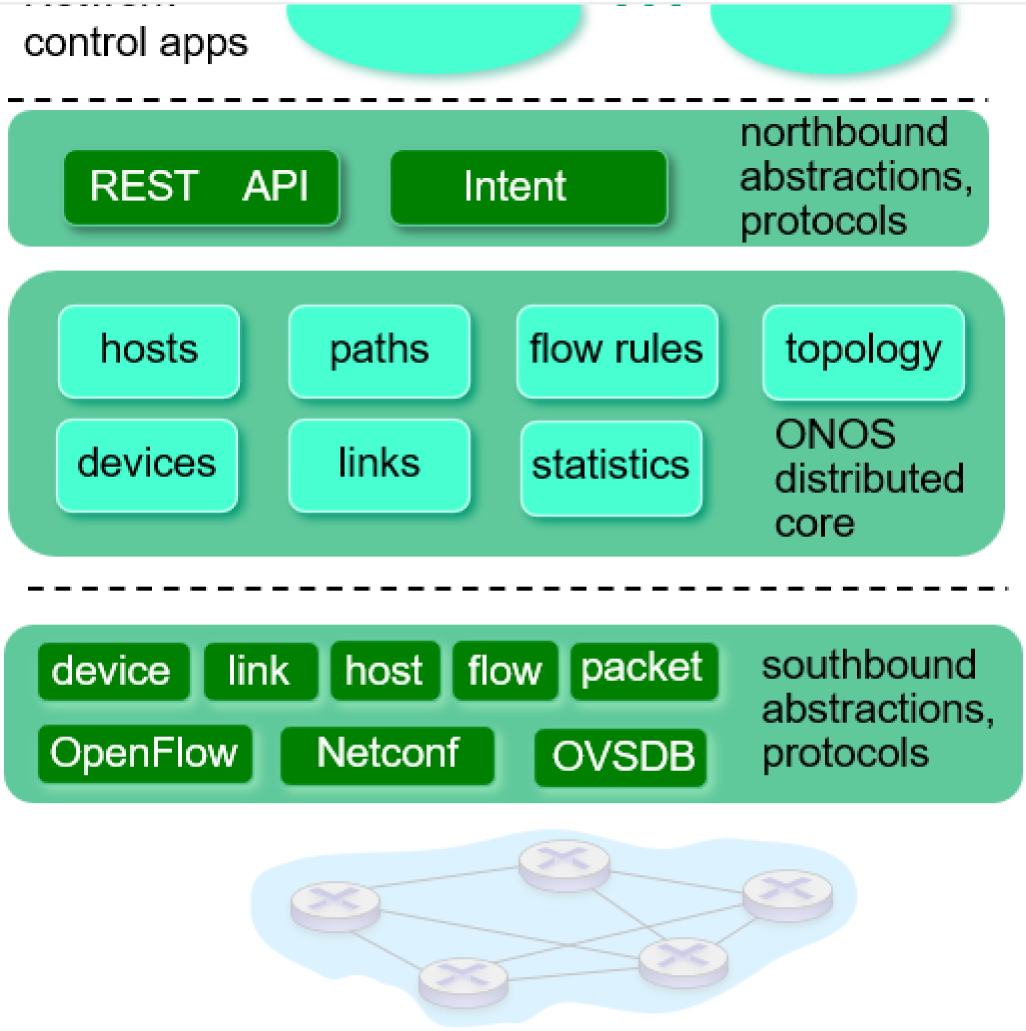
ONOS controller

> What is NFV? (redhat.com) (https://www.redhat.com/en/topics/virtualization/what-is-nfv)
>
> ### What is NFV?
>
> Network functions virtualization (网络功能虚拟化，NFV) is a way to virtualize network services (https://www.redhat.com/en/topics/virtualization/what-is-virtualization) , such as routers, firewalls, and load balancers(路由器、防火墙和负载平衡器), that have traditionally been run on proprietary hardware(专有硬件上). These services are packaged as virtual machines (VMs) (https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine) on commodity hardware, which allows service providers to run their network on standard servers instead of proprietary ones. It is one of the primary components of a telco cloud (https://www.redhat.com/en/topics/cloud-computing/what-is-telco-cloud) , which is reshaping the telecommunications industry. 这些服务被打包为商品硬件上的虚拟机 (VM)，这允许服务提供商在标准服务器而不是专有服务器上运行他们的网络。它是电信云的主要组成部分之一，正在重塑电信行业。 With NFV, you don’t need to have dedicated hardware for each network function. NFV improves scalability and agility by allowing service providers to deliver new network services and applications on demand, without requiring additional hardware resources. 使用 NFV，您无需为每个网络功能配备专用硬件。 NFV 允许服务提供商按需提供新的网络服务和应用程序，而无需额外的硬件资源，从而提高了可扩展性和敏捷性。

自己搭建一个 SDN：

- 上层：Floodlight / Opendaylight
- 中间沟通：Openflow
- 下层：Mininet (https://github.com/mininet/mininet)

> Mininet 能实现在一台机器上模拟一个完整的主机、链路和交换机网络



ICMP: The Internet Control Message Protocol

因特网控制报文协议（**the Internet Control Message Protocol**, **ICMP**），被主机和路由器用来彼此沟通网络层的信息。ICMP 最典型的用途是差错报告。

> **TTL**
>
> TTL 指寻找路由器的跳数，如 TTL = 1 代表只有一跳。

Traceroute and ICMP

- source sends series of UDP segments to destination 源将一系列 UDP 段发送到目标
- when datagram in *n*th set arrives to nth router 当第 n 个集中的数据报到达第 n 个路由器时
- when ICMP message arrives, source records RTTs 当 ICMP 消息到达时，源记录 RTT

> college_assignment/计算机网络.md at main · A-BigTree/college_assignment · GitHub (https://github.com/A-BigTree/college_assignment/blob/main/learning_Notes/%E8%AE%A1%E7%AE%97%E6%9C%B

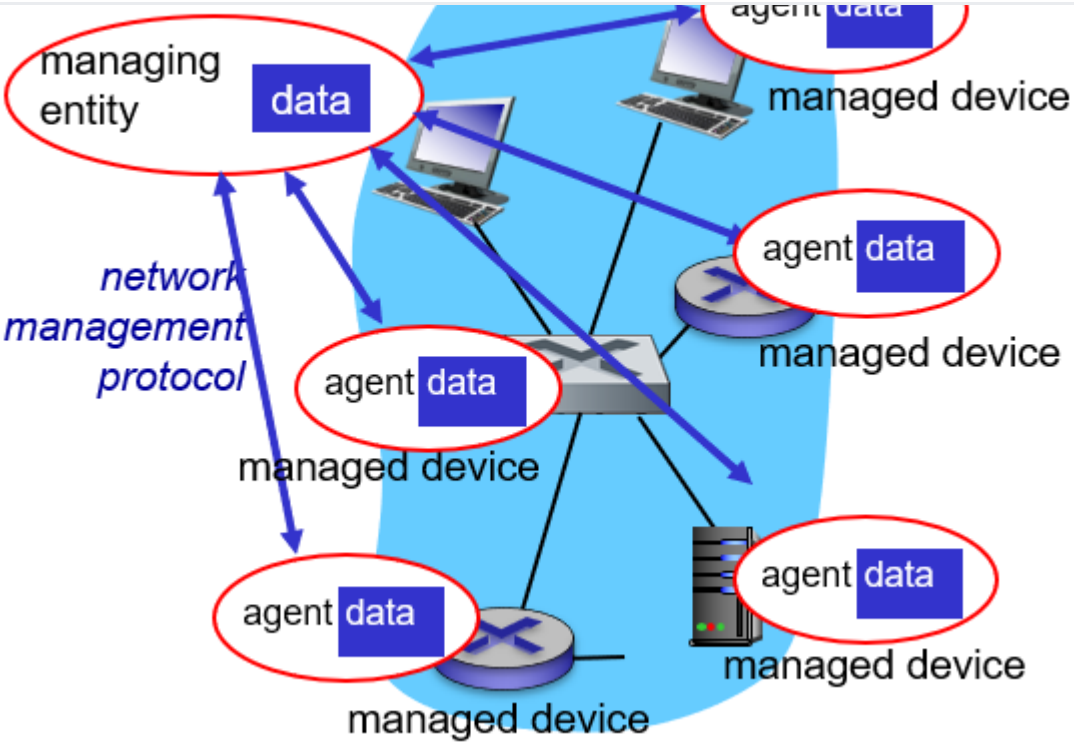
> 96%87%E5%8D%8F%E8%AE%AE)
>
> ICMP 报文类型如下图所示:
>
>
>
> | ICMP Type | Code | Description |
> |---|---|---|
> | 0 | 0 | echo reply (to ping) |
> | 3 | 0 | destination network unreachable |
> | 3 | 1 | destination host unreachable |
> | 3 | 2 | destination protocol unreachable |
> | 3 | 3 | destination port unreachable |
> | 3 | 6 | destination network unknown |
> | 3 | 7 | destination host unknown |
> | 4 | 0 | source quench (congestion control) |
> | 8 | 0 | echo request |
> | 9 | 0 | router advertisement |
> | 10 | 0 | router discovery |
> | 11 | 0 | TTL expired |
> | 12 | 0 | IP header bad |

Network management and SNMP

网络管理包括了硬件、软件和人类元素的设置、整合和协调，以监视、测试、轮询、配置、分析、评价和控制网络及网元资源，用合理的成本满足实时性、运营性能和服务质量的要求。Network management includes the deployment, integration, and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost.

Infrastructure for network management

网络管理框架。网络管理的关键组件如下图所示:



> What Is Network Management? - Cisco (https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-network-management.html)
>
> ### What Is Network Management?
>
> Network management refers to two related concepts. First is the process of configuring, monitoring, and managing the performance of a network. Second is the platform that IT and NetOps teams use to complete these ongoing tasks. [illegible] IT [illegible]

- [illegible]**managing server**[illegible]NOC[illegible]
- [illegible]**managed device**[illegible]managed object[illegible]
- [illegible]**Management Information Base**[illegible]**MIB**[illegible]
- [illegible]**network management agent**[illegible]
- [illegible]**network management protocol**[illegible]

SNMP protocol: message types

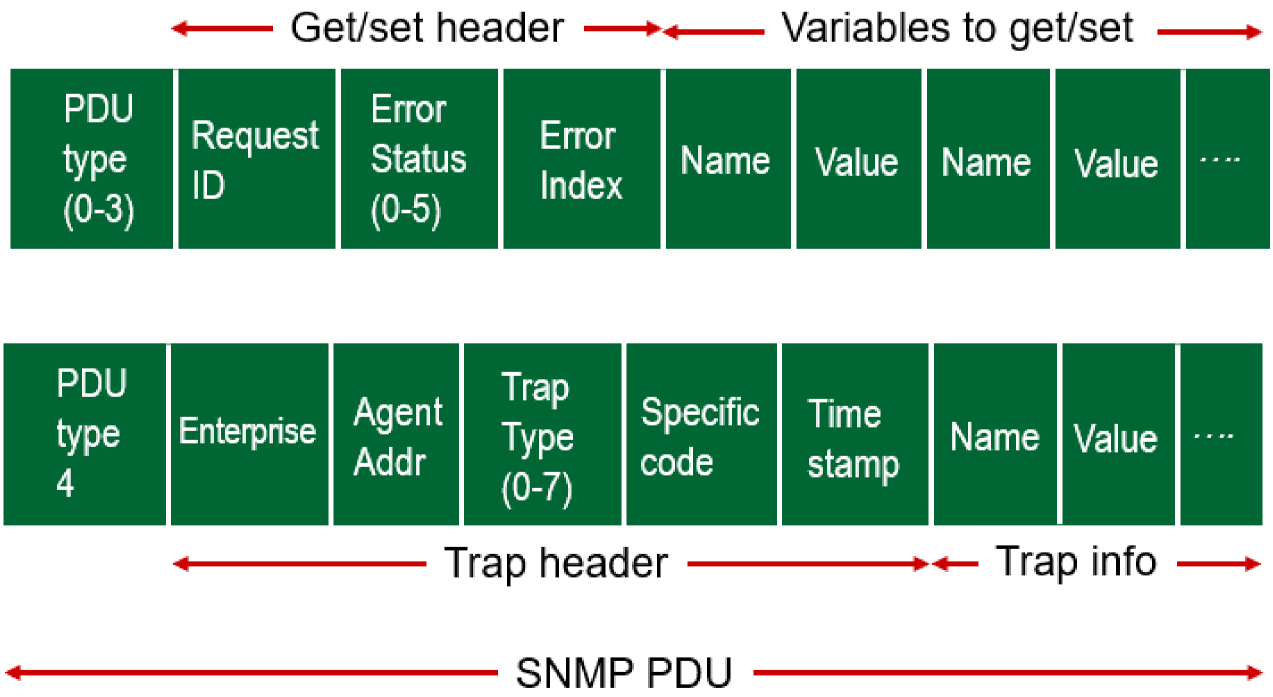
[illegible]

[illegible]Simple Network Management Protocol[illegible]SNMP[illegible][illegible]

SNMPv2 [illegible] 7 [illegible]PDU[illegible]PDU [illegible]

GetRequest GetNextRequest GetBulkRequest	manager-to-agent: “get me data” (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here’s MIB value
SetRequest	manager-to-agent: set MIB value
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event

SNMP protocol: message formats



发送接收都是上半这样的格式。下半是 Trap。

其中 PDU 是 Protocol Data Unit，合起来叫做“简单网络管理协议数据单元”。

Summary

- approaches to network control plane
  - per-router control (traditional)
  - logically centralized control (software defined networking)
- traditional routing algorithms
  - implementation in Internet: OSPF, BGP
- SDN controllers
  - implementation in practice: ODL, ONOS
- Internet Control Message Protocol
- network management

第五章主要讲了路由表怎么做出来的，展示了网络链路上怎么指路标。最新的“指路标”技术是SDN。

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

即链路层和局域网

Outline

1. introduction, services
2. error detection, correction
3. multiple access protocols *
4. LANs *
    - addressing, ARP
    - Ethernet
    - switches
    - VLANS
5. link virtualization: MPLS (少考)
6. data center networking (少考)
7. a day in the life of a web request

Link layer: introduction

> 笔记-计算机网络-自顶向下 | FEZ 的博客 (toby-fish.github.io) (https://toby-fish.github.io/2021/11/22/%E7%AC%94%E8%AE%B0-%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C-%E8%87%AA%E9%A1%B6%E5%90%91%E4%B8%8B/)
>
> 为了透彻理解链路层以及它是如何与网络层关联的，我们考虑一个交通运输的类比例子。
>
> 假设一个旅行计划为游客开辟从美国新泽西州的普林斯顿到瑞士洛桑的旅游路线。假定该旅行社认为对于游客而言最为便利的方案是：从普林斯顿乘豪华大轿车到 JFK 机场，然后乘飞机从 JFK 机场去日内瓦机场，最后乘火车从日内瓦机场到洛桑火车站。一旦该旅行社作了这 3 项预定，普林斯顿豪华大轿车公司将负责将游客从普林斯顿带到 JFK，航空公司将负责将游客从 JFK 带到日内瓦，瑞士火车服务将负责将游客从日内瓦带到洛桑。该旅途中 3 段中的每一段都在两个“相邻”地点之间是“直达的”。注意到这 3 段运输是由不同的公司管理，使用了完全不同的运输方式（豪华大轿车、飞机和火车）。尽管运输方式不同，但它们都提供了将旅客从一个地点运输到相邻地点的基本服务。
>
> 在这个运输类比中，一个游客好比一个数据报，每个运输区段好比一条链路，每种运输方式好比一种链路层协议，而该旅行社好比一个路由选择协议。

一些概念：

- 节点（node）：运行链路层协议的任何设备，如：主机、路由器、网桥、交换机等；
- 链路（link）：沿着通信路径连接相邻节点的通信信道，包括：
    - 有线链路
    - 无线链路
    - 局域网，共享性链路
- 数据单元帧（frame）：封装数据报，位于第二层协议

链路层提供的服务

数据链路层负责从一个节点通过链路将（帧中的）数据报发送到相邻的物理节点（一个子网内部的 2 节点）。

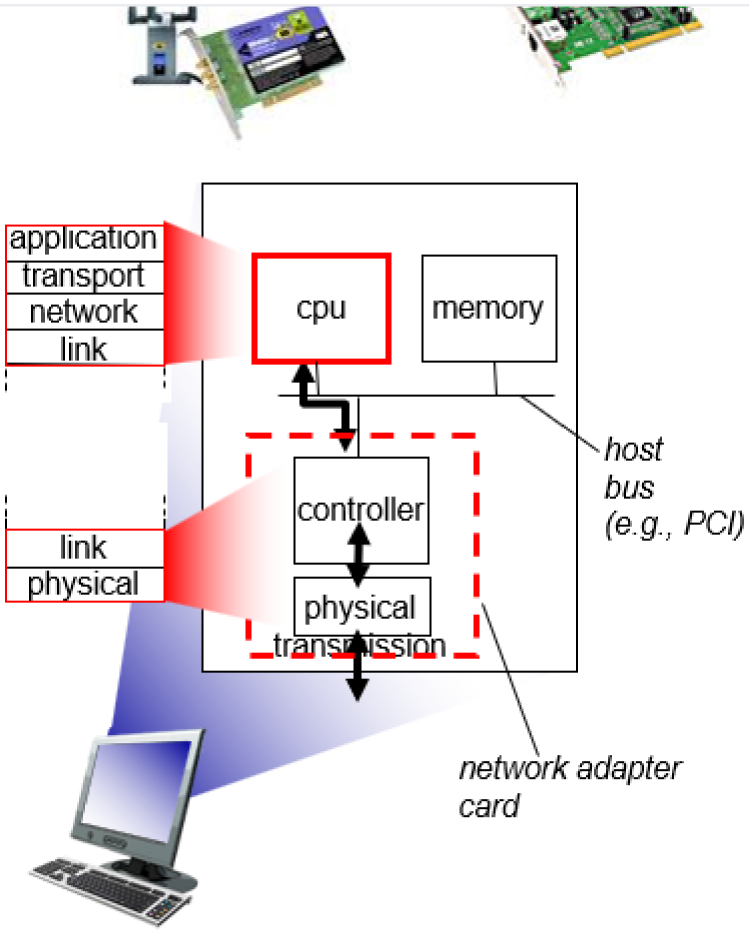
- 成帧（Framing）、链路接入（Link access）：
    - 将数据报封装在帧中，加上帧头、帧尾部
    - 如果采用的是共享性介质，信道接入获得信道访问权；
    - **媒体访问控制（Medium Access Control，MAC）**协议规定了帧在链路上传输规则；
- 可靠交付（Reliable deliver）、差错检测和纠正（Error detection and correction）：
    - 第三章传输层的可靠数据传输
    - 在低出错率的链路上（光纤和双绞线电缆）很少使用
    - 在无线链路经常使用：出错率高

> 为什么在链路层和传输层都实现了可靠性？
>
> 一般化的链路层服务，不是所有的链路层都提供这些服务。一个特定的链路层只是提供其中一部分的服务。

链路在何处实现

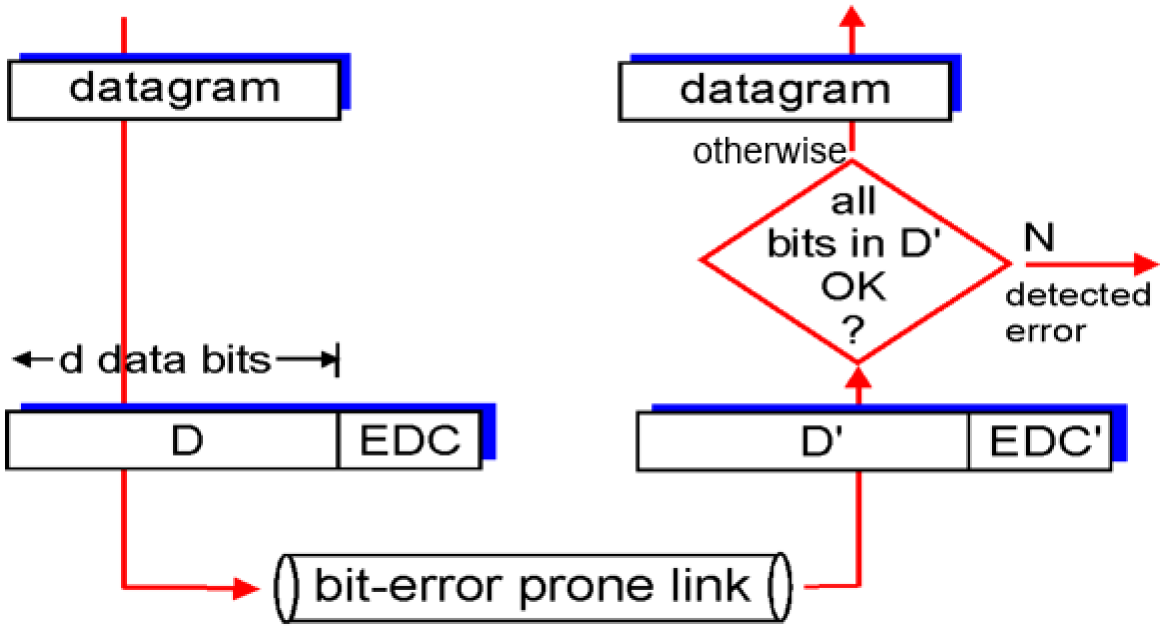
链路层的主体部分在网络适配器（**network adapter**）中实现，网络适配器有时也称为网络接口卡（**Network Interface Card，NIC**）。位于网络适配器核心的是链路层控制器，该控制器通常是一个实现了许多链路层服务（成帧、链路接入、差错检测等）的专用芯片。

一个典型的主机体系结构如下图所示：



Error detection, correction

差错检测与纠正的场景如下图所示：



差错检测和纠正技术有：

1. 奇偶校验，包括使用单个奇偶检验位（parity bit）和二维奇偶校验（two-dimensional parity）；

EDC

- EDC=差错检测和纠正位（冗余位）
- D=数据由差错检测保护，可以包含头部字段错误检测。

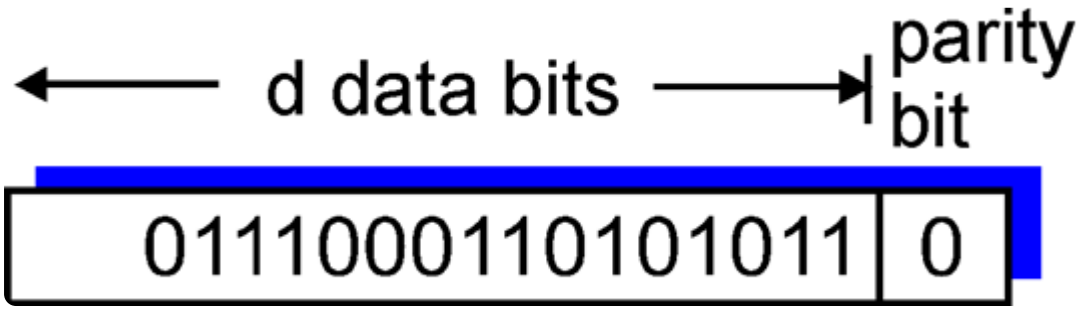
错误检测不是 100%可靠的！

- 协议会漏检一些错误，但是很少
- 更长的 EDC 字段可以得到更好的检测和纠正效果

Parity Checks

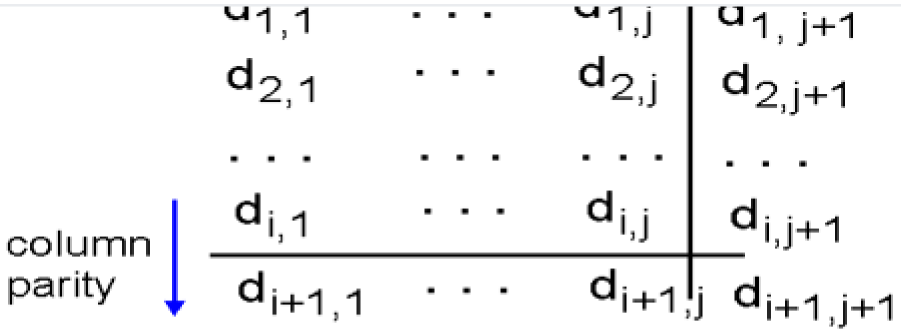
单个奇偶校验位（**single bit parity**）：detect single bit errors

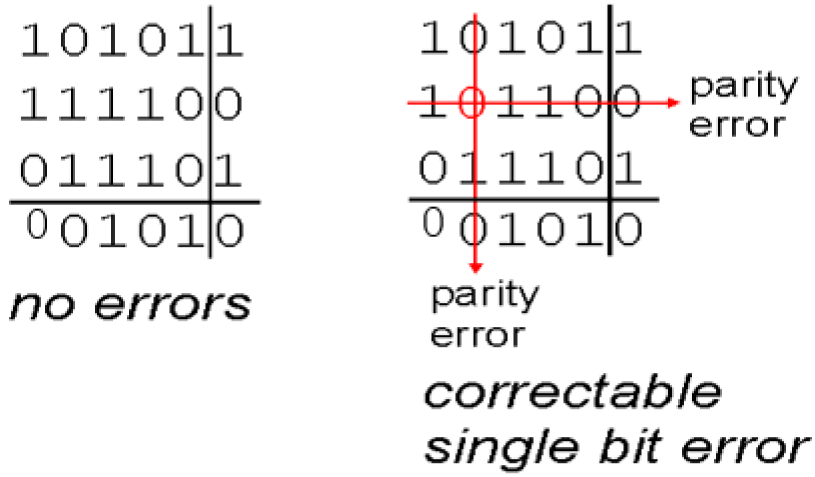
发送 d 比特信息附加一个比特使 d+1 比特中 1 的总数是偶数（偶校验）或奇数（奇校验），偶校验如下图所示：



但注意此种方法只能检测不能纠错，并且也不能检测超过 1 比特位的错误。

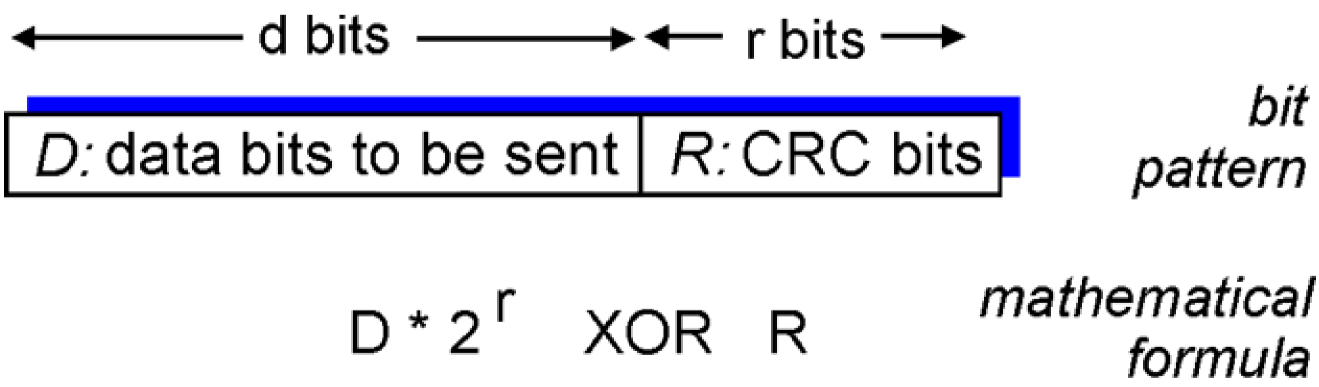
二维奇偶校验（**two-dimension parity**）：detect and correct single bit errors

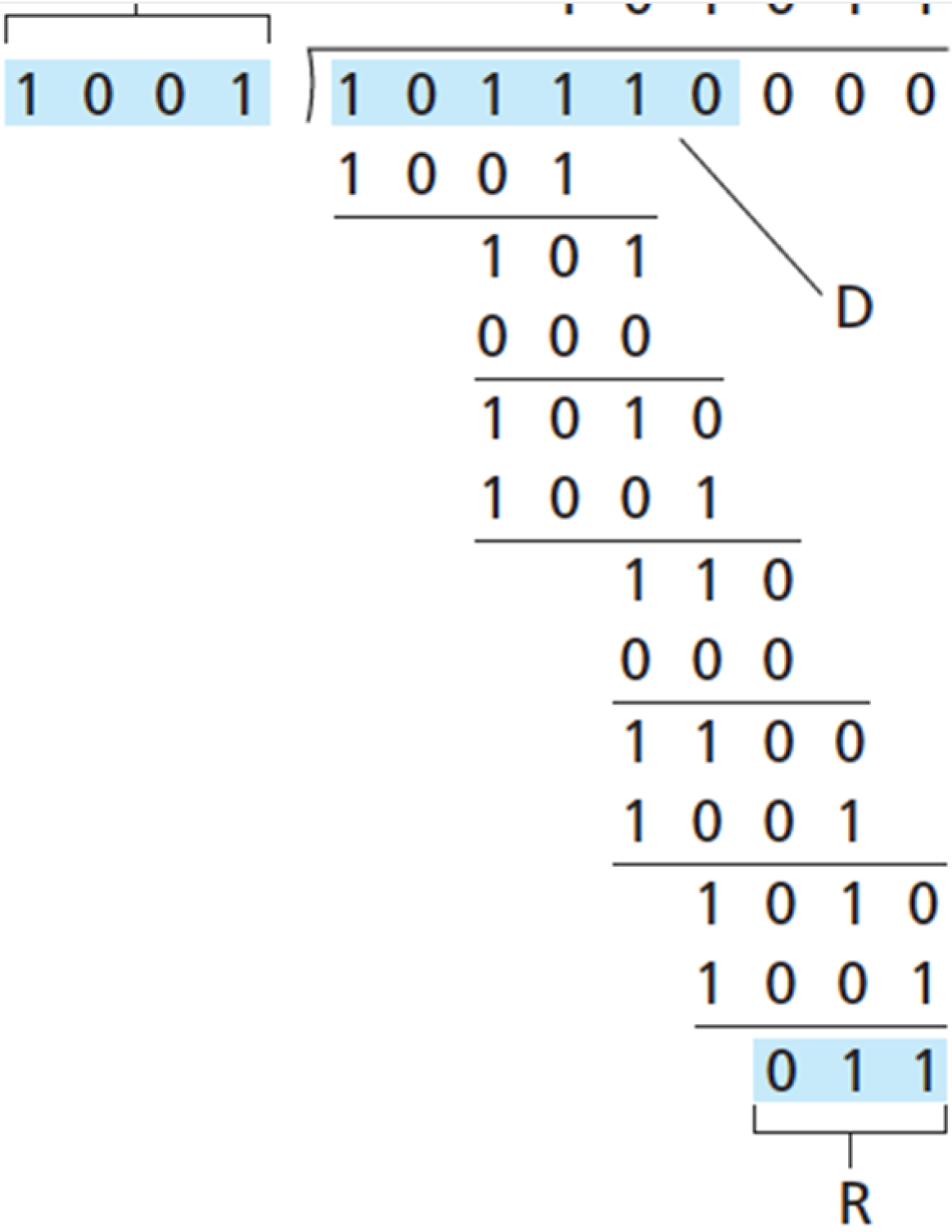




Cyclic Redundancy Check(CRC)

计算机网络中广泛应用的差错检测技术基于循环冗余检测（**Cyclic Redundancy Check**，**CRC**）编码，也称为多项式编码（**polynomial code**），CRC 如下图所示:





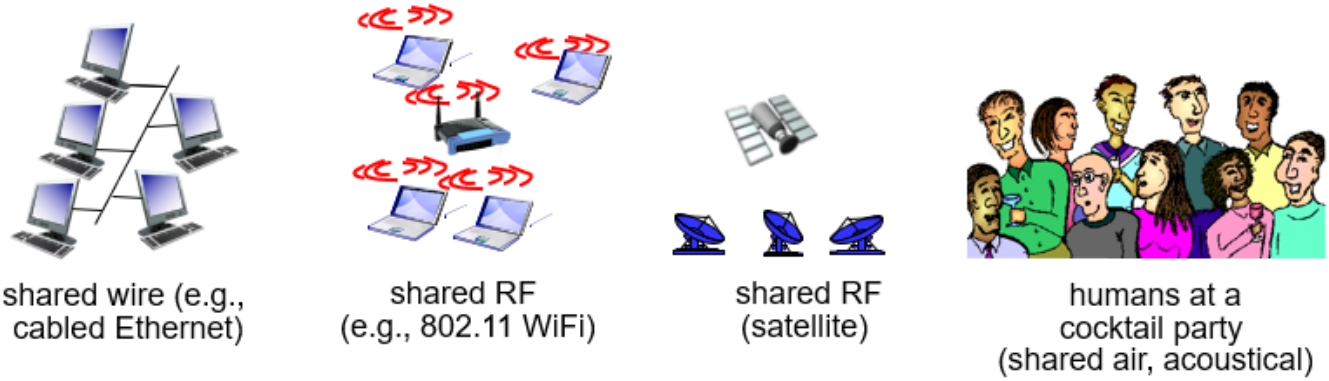
其中 R 计算:

$$ R=remainder\frac{D\cdot2^r}{G} $$

Multiple access protocols

多路访问链路和协议，或者说全称 Multiple Acess Links and Protocol。

两种类型的网络链路:



  - 以太网交换机和主机之间的点对点链路
- 广播链路（**broadcast link**）：共享线路或媒体
  - 传统以太网
  - HFC 上行链路
  - 802.11 无线局域网

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)

shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

> 多路访问协议
>
> - 单个共享的广播型链路
> - 传输的帧在接收方可能在接收方处碰撞/冲突（collide）。
> - 分布式算法-决定节点如何使用共享信道，即：决定节点什么时候可以发送？

在理想情况下，对于速率为 R bps 的广播信道，多路访问协议应该具有以下所希望的特性：

1. 当且仅当一个节点发送数据时，该节点具有 R bps 的吞吐量；
2. 当有 M 个节点发送数据时，每个节点吞吐量为 R/M bps。这不必要求 M 个节点中的每一个节点总是有 R/M 的瞬时速率，而是每个节点在一些适当定义的时间间隔内应该有 R/M 的平均传输速率；
3. 协议是分散的，这就是说不会因为某主节点故障而使整个系统崩溃

   （高级的说法：没有特殊节点协调发送、没有时钟和时隙的同步）

4. 协议是简单的，使实现不昂贵

现在主流的 3 种类型多路访问协议（介质访问控制协议，MAC）：

- 信道划分协议（**channel partitioning protocol**）
  - 把信道划分成小片（时间、频率、编码）
  - 分配片给每个节点专用
- 随机接入协议（**random access protocol**）
  - 信道不划分，允许冲突
  - 冲突后恢复
- 轮流协议（**taking-turns protocol**）

信道划分协议

Channel Partitioning Protocols

时分多路复用（time division multiple access，简称 TDMA）：

- 轮流（“rounds”）使用信道（channel），信道的时间分为周期
- 每个站点使用每周期中固定的时隙（长度＝帧传输时间）传输帧
- 时隙空闲（浪费）主要出现在站点无帧传输

频分多路复用（frequency division multiple access，简称 FDMA）：

- 信道的有效频率范围被分成一个个小的频段
- 每个站点被分配一个固定的频段
- 浪费主要出现在分配给站点的频段没有被使用

码分多址（Code Division Multiple Access，简称 CDMA）

- 所有站点在整个频段上同时进行传输，采用编码原理加以区分
- 完全没有冲突（假定信号同步很好，线性叠加）

随机接入协议

> Random Access Protocols

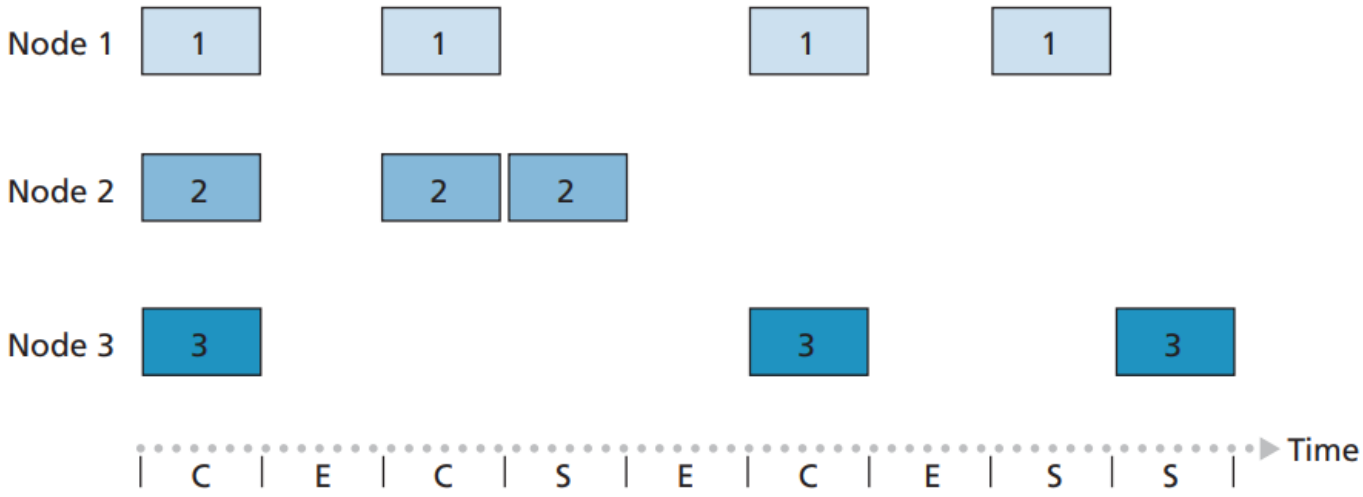
Slotted ALOHA

Slotted 时隙，ALOHA 是夏威夷土著语，意为“你好”、“谢谢”。

假设：

- all frames same size
- time divided into equal size slots(时间划分为大小相等的插槽) (time to transmit 1 frame)
- nodes start to transmit only slot beginning(插槽开头)
- nodes are synchronized（同步过）
- if 2 or more nodes transmit in slot, all nodes detect collision(所有节点都检测到冲突)

在每个节点中，时隙 ALOHA 的操作如下：



- 如果有碰撞，该节点在时隙结束之前检测到这次碰撞，该节点以概率 p 在后续的每个时隙中重传它的帧，直到该帧被无碰撞的传输出去；

Slotted ALOHA: efficiency

效率：当有很多节点，每个节点有很多帧要发送时，x%的时隙是成功传输帧的时隙

- 假设 N 个节点，每个节点都有很多帧要发送，在每个时隙中的传输概率是 p
- 一个节点成功传输概率是 $p(1-p)^{N-1}$
- 任何一个节点的成功概率是 $Np(1-p)^{N-1}$
- N 个节点的最大效率：求出使 $Np(1-p)^{N-1}$ 最大的 p*
- 代入 P*得到最大 $f(p^)=Np^(1-p^*)^{N-1}$
- N 为无穷大时的极限为 $1/e=0.37$

即最好情况：信道利用率 37%

Pure (unslotted) ALOHA

在纯 ALOHA 中，当一帧首次到达，节点立刻将该帧完整地传输进广播信道。

效率上：比时隙 ALOHA 更差了！

载波侦听多路访问

CSMA (carrier sense multiple access)

CSMA 规则

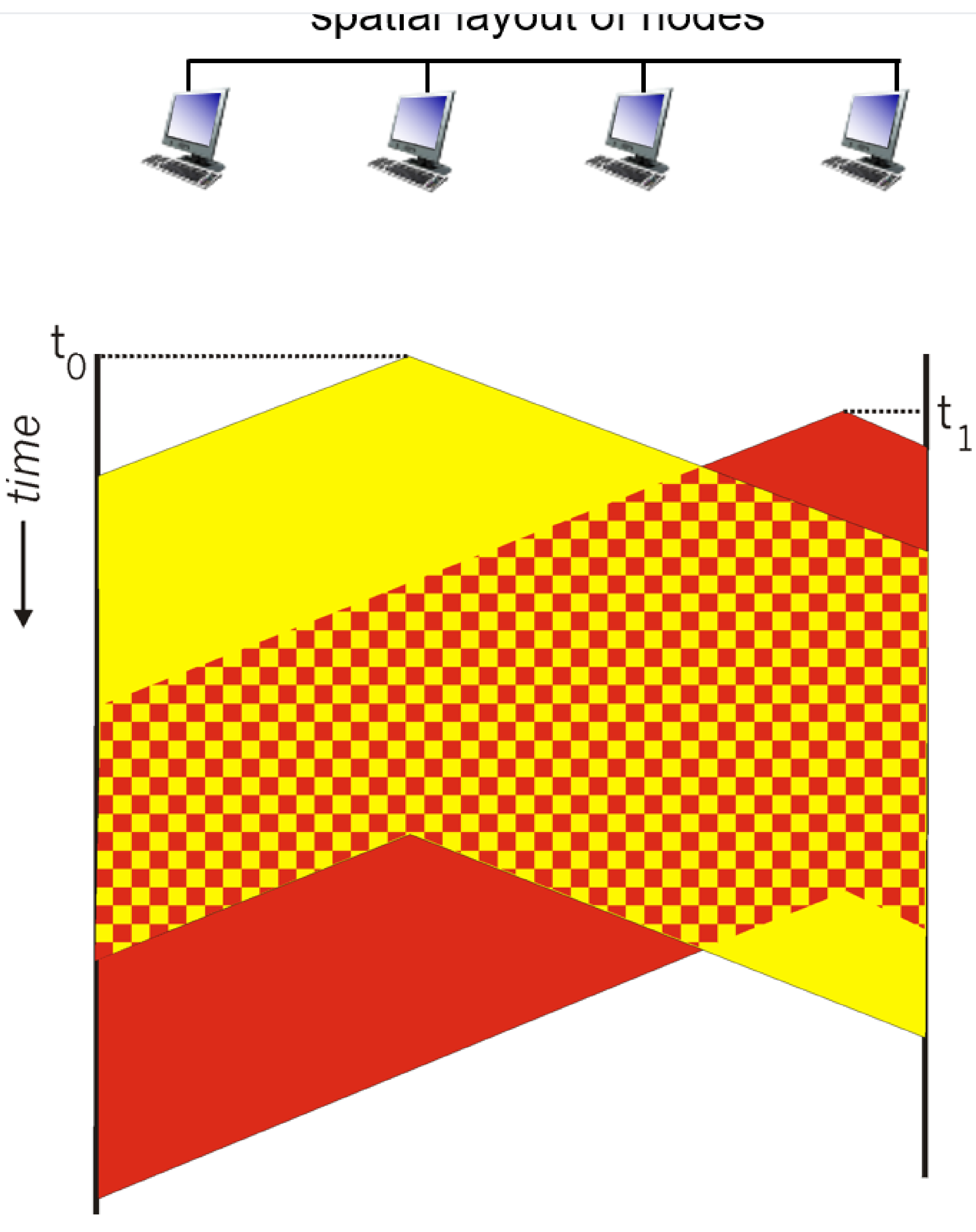
两个重要的规则：

1. 说话之前先听。如果其他人正在说话，等到他们说完话为止。在网络领域中，这被称为载波侦听（carrier sensing），即一个节点在传输前先听信道。如果来自另一个节点的帧正向信道上发送，节点则等待直到检测到一小段时间没有传输，然后开始传输。
2. 如果与他人同时开始说话，停止说话。在网络领域中，这被称为碰撞检测（collision detection），即当一个传输节点在传输时一直在侦听此信道。如果它检测到另一个节点正在传输干扰帧，它就停止传输，在重复“侦听–当空闲时传输”循环之前等待一段随机时间。

> 这两个规则包含在载波侦听多路访问（Carrier Sense Multiple Access, CSMA）和具有碰撞检测的 **CSMA**（CSMA with Collision Detection, CSMA/CD）协议族中。

CSMA 冲突

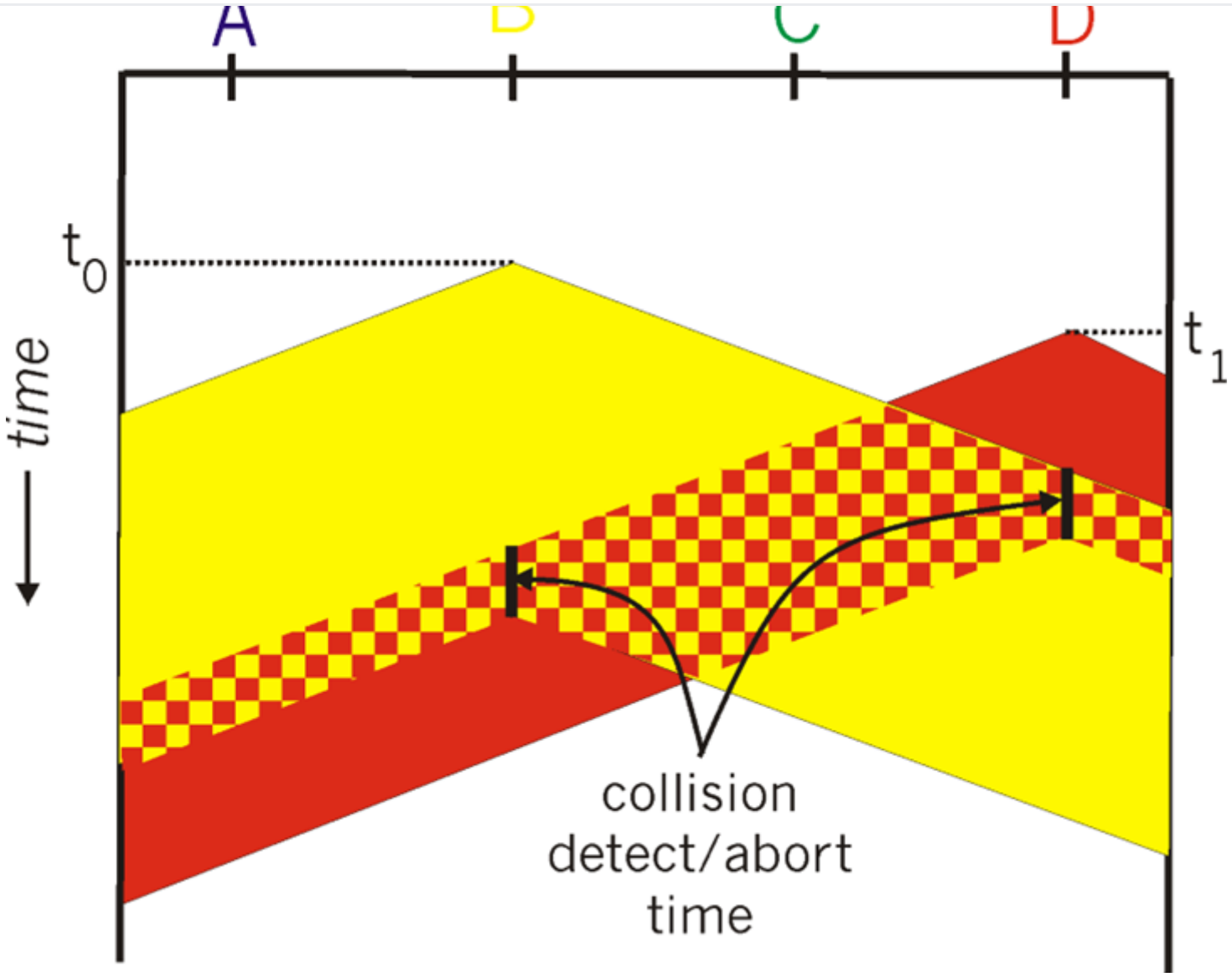
collisions can still occur: propagation delay(传播延迟) means two nodes may not hear each other’s transmission(无法听到彼此的传输).



影响: entire packet transmission time waste(浪费整个数据包传输时间), and distance & propagation delay play role in in determining collision probability 距离和传播延迟在确定碰撞概率方面发挥作用

CSMA/CD(冲突检测)

CSMA 传播时延越长，载波侦听节点不能侦听到网络中另一个节点已经开始传输的机会就越大。如果一个节点开始传播后发送碰撞，其实此次传播已经失败，将剩余数据传完没有意义，即使中止才是上策，这便是具有 CSMA/CD 所作改进。



算法：

- 适配器获取数据报，创建帧
- 发送前：侦听信道 CS
    - 闲：开始传送帧
    - 忙：一直等到闲再发送
- 发送过程中，冲突检测 CD
    - 没有冲突：成功
    - 检测到冲突：放弃，之后尝试重发
- 发送方适配器检测到冲突，除放弃外，还发送一个 Jam 信号，所有听到冲突的适配器也是如此

    > 强化冲突：让所有站点都知道冲突

- 如果放弃，适配器进入指数退避状态。

轮流协议

channel partitioning(信道划分) MAC protocols:

- share channel efficiently and fairly at high load 共享信道在高负载时是有效和公平的
- inefficient at low load: delay in channel acces 在低负载时效率低下

random access(随机访问) MAC protocols:

- efficient at low load: single node can fully utilize channel 在低负载时效率高：单个节点可以完全利用信道全部带宽
- high load: collision overhead 高负载时：冲突开销较大，效率极低，时间很多浪费在冲突中

“taking turns” protocols(轮流协议): look for best of both worlds!

轮流协议包含：

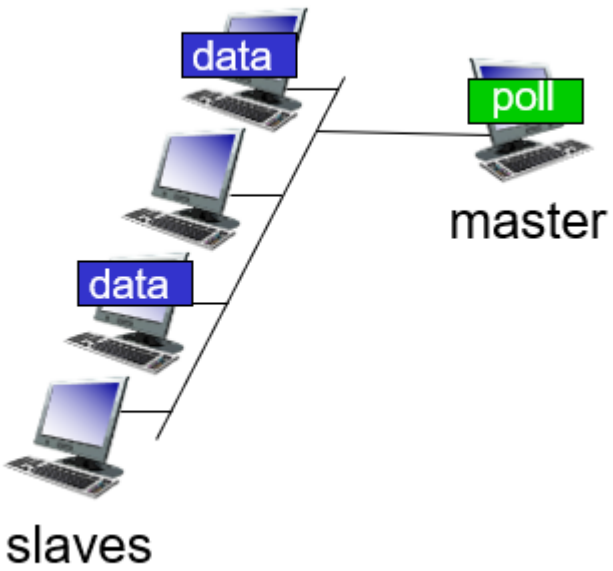
- 轮询协议（polling protocol）
- 令牌传递协议（token-passing protocol）

轮询协议

主节点邀请从节点依次传送；从节点一般比较“dumb”。

缺点：

- 轮询开销：轮询本身消耗信道带宽
- 等待时间：每个节点需等到主节点轮询后开始传输，即使只有一个节点，也需要等到轮询一周后才能够发送
- 单点故障：主节点失效时造成整个系统无法工作



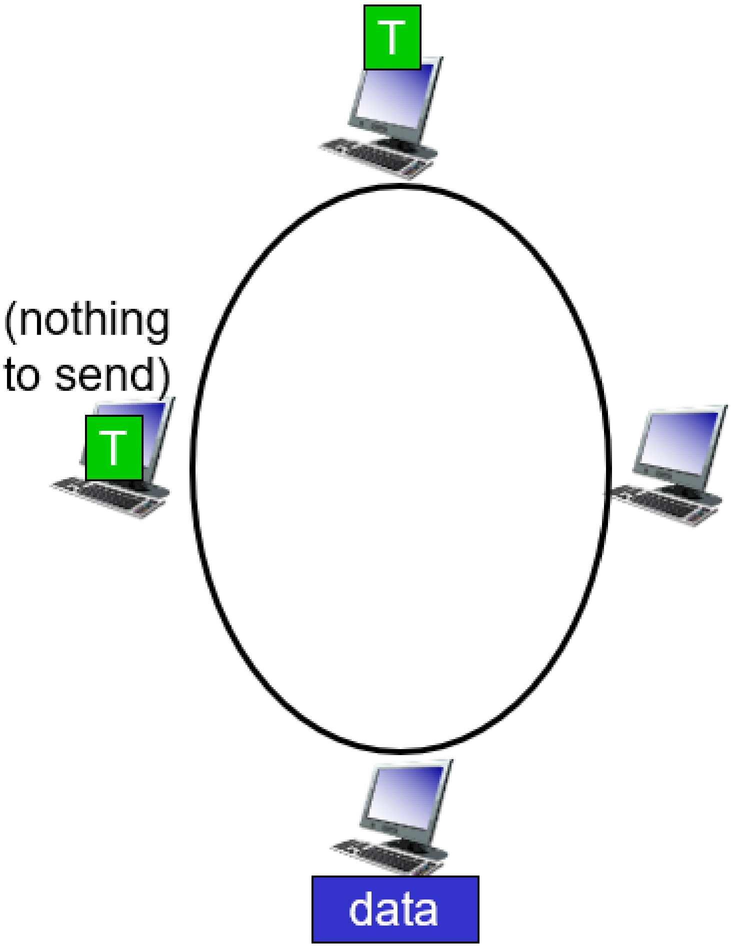
令牌传递协议

控制令牌(token)循环从一个节点到下一个节点传递。

> 令牌报文本质上是一种特殊的帧。

缺点:

- 令牌开销：本身消耗带宽
- 延迟：只有等到抓住令牌，才可传输
- 单点故障 (token) (令牌丢失系统级故障，整个系统无法传输；复杂机制重新生成令牌)

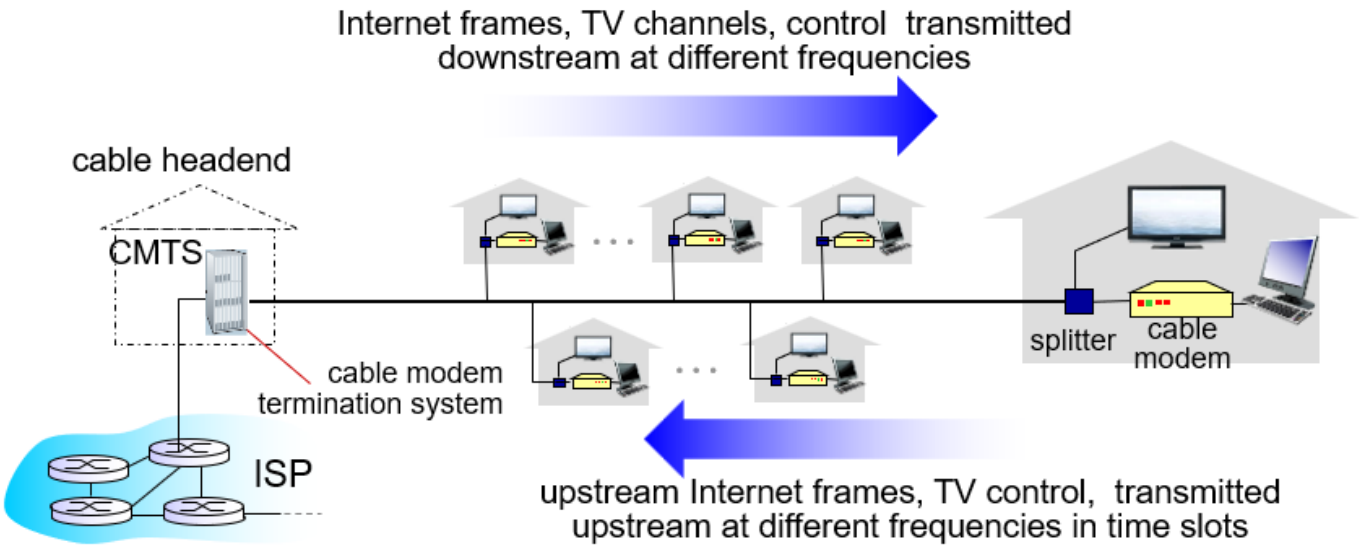


线缆接入网络

多个 40Mbps 下行(广播)信道，FDM:

- 下行：通过 FDM 分成若干信道，互联网、数字电视等
- 互联网信道：只有 1 个 CMTS 在其上传输

多个 30 Mbps 上行的信道，FDM:



MAC 协议总结

多点接入问题：对于一个共享型介质，各个节点如何协调对它的访问和使用？

- 信道划分：按时间、频率或者编码
    - TDMA、FDMA、CDMA
- 随机访问 (动态)
    - ALOHA, S-ALOHA, CSMA, CSMA/CD
    - 载波侦听：在有些介质上很容易 (wire：有线介质)，但在有些介质上比较困难 (wireless：无线)
    - CSMA/CD：802.3 Ethernet 网中使用
    - CSMA/CA：802.11WLAN 中使用
- 依次轮流协议
- 集中：由一个中心节点轮询；

  分布：通过令牌控制
- 蓝牙、FDDI、令牌环

LANs

Addressing, ARP

32bit 的 IP 地址包含：

- 最后一跳：到达子网中的目标节点

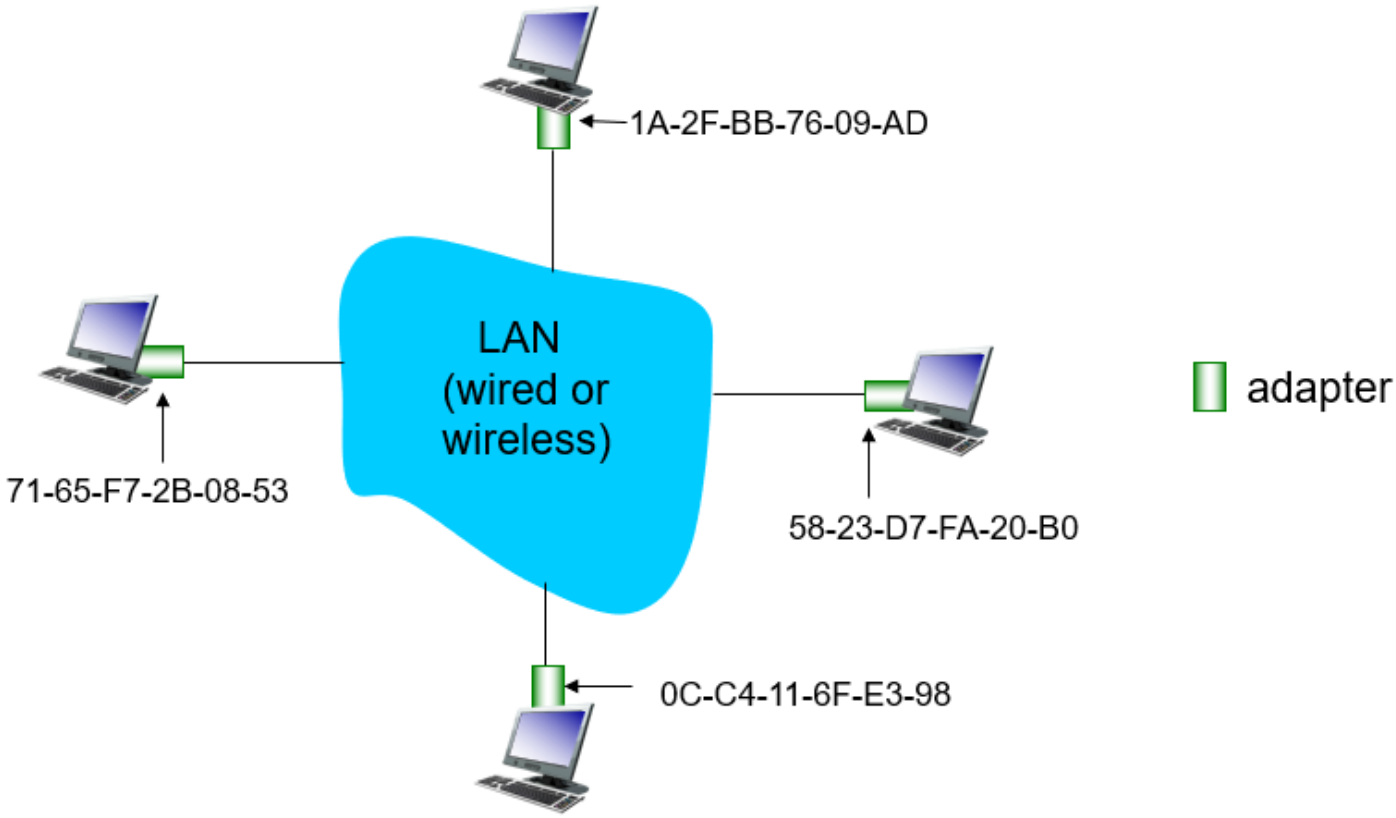
LAN（MAC/物理/以太网）地址：

- 用于使帧从一个网卡传递到与其物理连接的另一个网卡（在同一个物理网络中）
- 48bit MAC 地址固化在适配器的 ROM，有时也可以通过软件设定
- 理论上全球任何 2 个网卡的 MAC 地址都不相同

LAN 地址形如： `1A-2F-BB-76-09-AD`（16 进制表示，每一位代表 4 个 bits）

LAN 地址和 ARP

局域网上每个适配器都有一个唯一的 LAN 地址。



注：

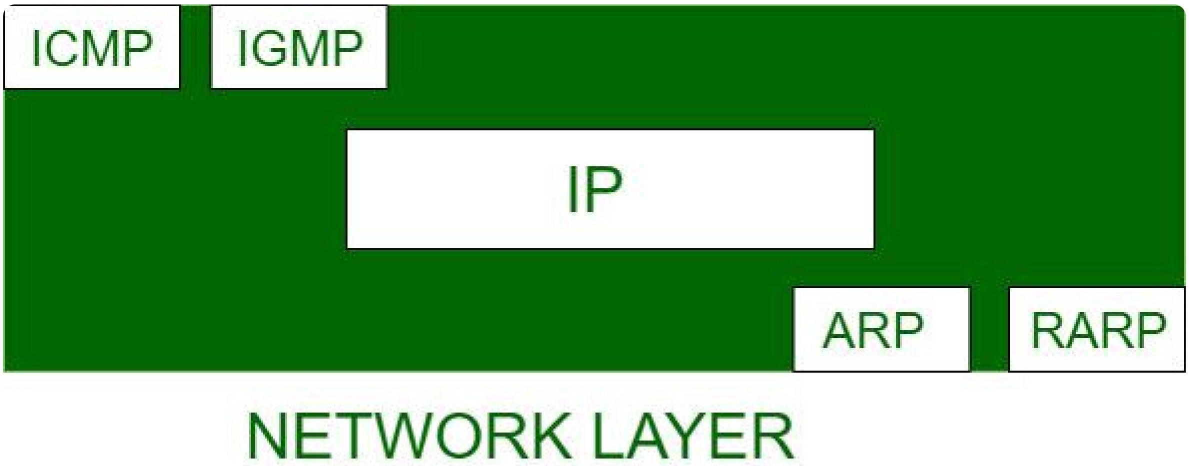
- MAC 地址由 IEEE 管理和分配
- 制造商购入 MAC 地址空间（保证你的 MAC 是唯一的，全球都没有重复）

类比：MAC 地址就好比社会安全号，IP 地址就好比通讯地址。

但由于 MAC 平面地址，并且是固定的，所以支持移动。

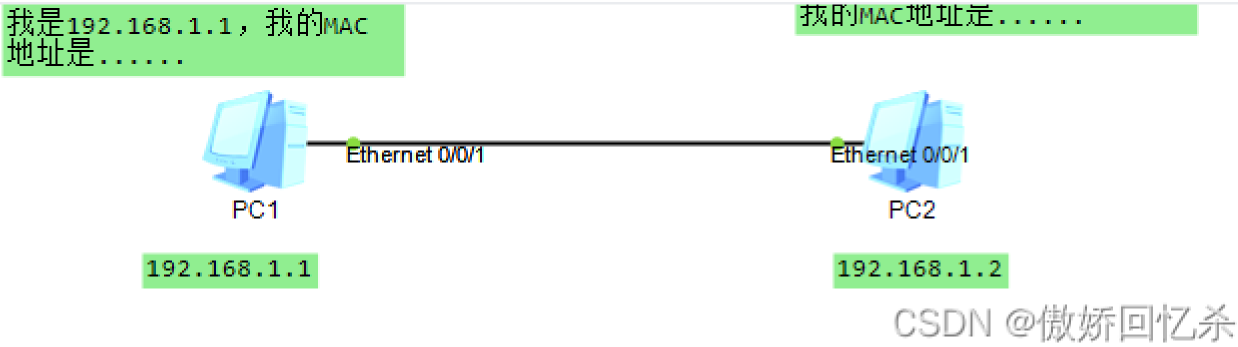
ARP: Address Resolution Protocol

> 在LAN上的每个IP节点都有一个ARP表。
>
> ARP表包括一些LAN节点IP/MAC地址的映射：`< IP address; MAC address; TTL>`。其中TTL时间是指地址映射失效的时间。典型是20min。

> How Address Resolution Protocol (ARP) works? - GeeksforGeeks (https://www.geeksforgeeks.org/how-address-resolution-protocol-arp-works/)
> The acronym ARP stands for **Address Resolution Protocol** which is one of the most important protocols of the Network layer in the OSI model. It is responsible to find the hardware address of a host from a known IP address. There are three basic ARP terms. ARP代表地址解析协议，它是OSI模型中网络层最重要的协议之一。它负责从已知IP地址中查找主机的硬件地址。
>
> 
>

ARP概念及其分类：

> ARP(地址解析协议)_傲娇回忆杀的博客-CSDN博客 (https://blog.csdn.net/weixin_62594100/article/details/123992695)
>
> 我们以以太网的工作环境作为背景来探讨这一协议（串行链路由于是点到点链路，故而不需要ARP）。在以太网的工作环境中，当主机需要向一个IP地址发送数据时，它需要将目标的物理地址（也就是MAC地址，也有文献称其为硬件地址）写在数据帧的目的MAC地址字段位置上，而这一动作的前提是，网络层已经知道了这一地址并且将其与逻辑目的地址建立了一个映射关系。这就好比在手机上存了一个电话号码并备注上了一个联系人一样，当需要打电话时，只需要查找该联系人的姓名即可，手机会帮我们自动选择他的电话号码拨过去。
>
> 当我们并不知道一台主机的IP地址与物理地址的映射关系的时候，就需要用到ARP。
>
> ## ARP分类：普通ARP
>
> 我们用一个简单的比喻来形容ARP的工作过程。当你只知道你跟张三是同班同学却不知道他的具体座位时，你站起来大喊了一声：“我是王二，谁是张三？”于是张三说：”我是张三。“这样，你就知道了张三的具体位置，同时张三也知道了你的位置和姓名。
>
> 如下图所示为一个ARP请求报文的示例。



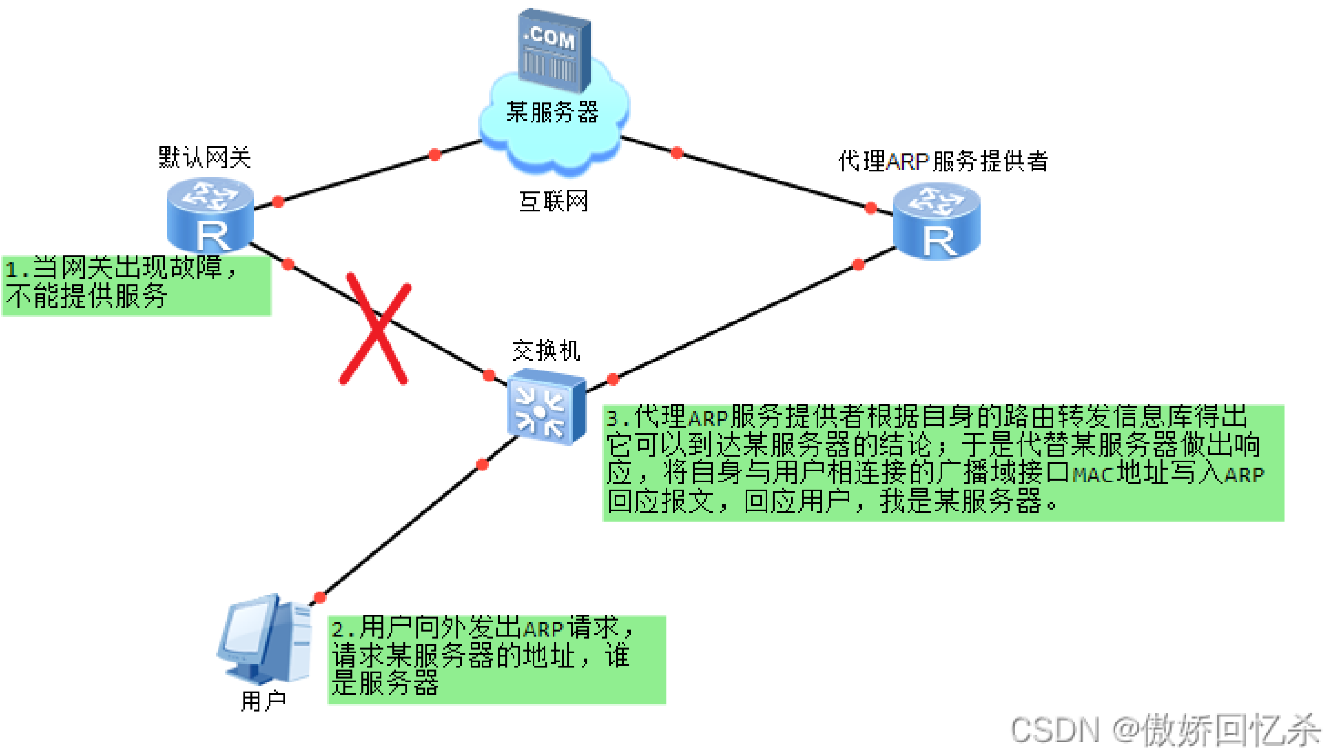
当一台主机需要访问一个与自己在同一个网络的 IP 地址但不知道目的主机的物理地址时，它就会发送一个 ARP 请求报文。由于我们并不知道目标物理地址是什么，该报文的目标物理地址（即 MAC 地址）在数据帧的头部用二层广播地址 FFFF.FFFF.FFFF 来填充。

一个二层目的地址为广播地址的数据帧是会被发送给广播域内所有的成员的，如果网络规划正确，那么这其中一定会包含真正的接收者。当真正的接收者收到该数据帧之后，会转交给自身的 ARP 程序，经过比对，发现其中的目的 IP 地址正是其所拥有的，就会对发送者做出回应，在回应报文中会将自身的物理地址写在发送者 MAC 地址的位置。

一次美妙的陌生人之间的互相介绍就这样完成了。看起来是不是很简单呢?

ARP 分类：代理 ARP

在一般情况下，只能为主机分配一个默认网关。如果需要互通的主机处在相同的网段却不在同一物理网络，并且连接主机的网关设备具有不同的网关地址，在这种场景中，如果发生网络故障，我们该如何防止业务中断呢？在这种场景中，需要代理 ARP，其工作过程如下图所示。



可以看出，实际上，代理 ARP 只是一种服务，它并不是一种协议。并且，服务提供者对用户进行了“欺骗”，只是将自身的 MAC 地址回应给了用户，以此来达到代替用户转发数据的目的。

ARP 报文格式

ARP 协议包（ARP 报文）主要分为 ARP 请求包和 ARP 响应包。ARP 协议是通过报文进行工作的，ARP 报文格式如图所示。

> 
>
>
> ARP 报文总长度为 28 字节，MAC 地址长度为 6 字节，IP 地址长度为 4 字节。
>
> 其中，每个字段的含义如下。
>
> - 硬件类型：指明了发送方想知道的硬件接口类型，以太网的值为 1。
> - 协议类型：表示要映射的协议地址类型。它的值为 0x0800，表示 IP 地址。
> - 硬件地址长度和协议长度：分别指出硬件地址和协议的长度，以字节为单位。对于以太网上 IP 地址的 ARP 请求或应答来说，它们的值分别为 6 和 4。
> - 操作类型：用来表示这个报文的类型，ARP 请求为 1，ARP 响应为 2，RARP 请求为 3，RARP 响应为 4。
> - 发送方 MAC 地址：发送方设备的硬件地址。
> - 发送方 IP 地址：发送方设备的 IP 地址。
> - 目标 MAC 地址：接收方设备的硬件地址。
> - 目标 IP 地址：接收方设备的 IP 地址。

ARP 协议：在同一个 LAN（网络）

需求：已知 B 的 IP 地址，想要确定同一网络内 B 的 MAC 地址。

流程：

- 首先，A 要发送帧给 B（其中 B 的 IP 地址已知），但 B 的 MAC 地址并不在 A 的 ARP 表中。

  > **TIP**
  >
  > 发送帧只能在同一个网关下进行，不能穿透路由器前往别的网络。

  所以，A 广播了包含 B 的 IP 地址的 ARP 查询包：Dest MAC address = FF-FF-FF-FF-FF-FF

  LAN 上的所有节点都会收到该查询包。

- A [illegible] ARP [illegible] IP-to-MAC [illegible]

> **TIP**
> - 软状态：靠定期刷新维持的系统状态
> - 定期刷新周期之间维护的状态信息可能和原有系统不一致

ARP 是即插即用的：

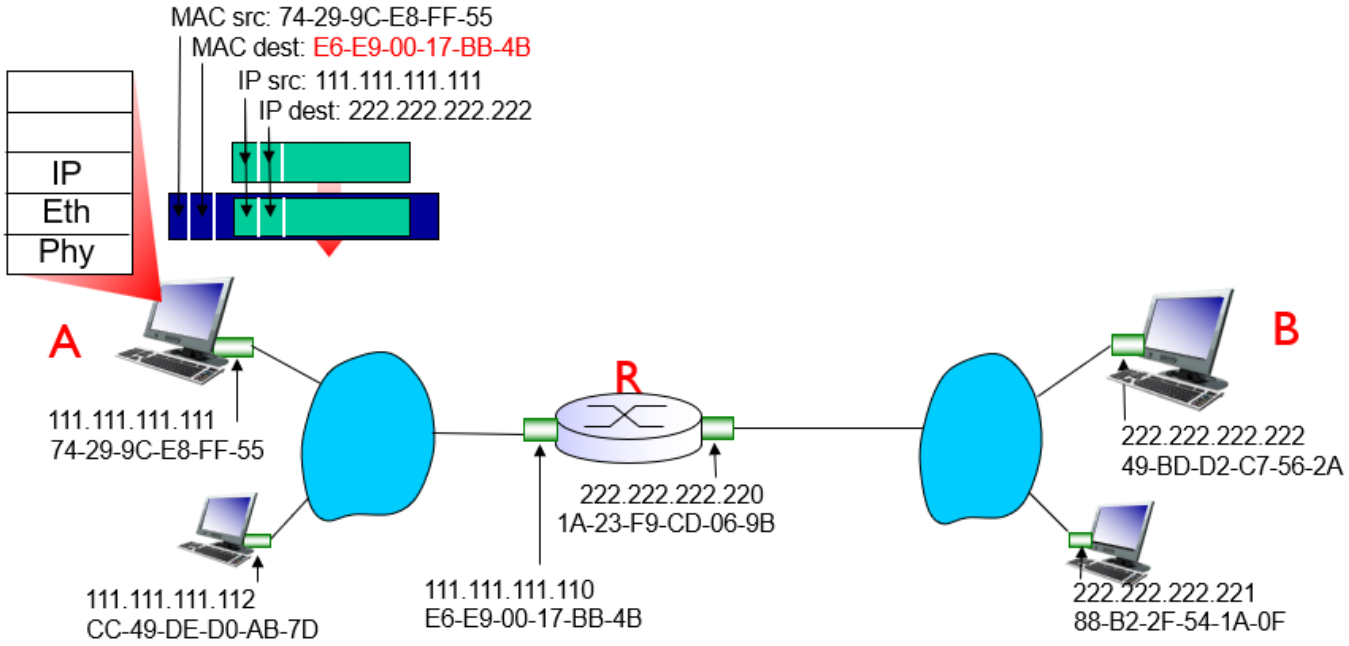
- 节点自己创建 ARP 的表项
- 无需网络管理员的干预

ARP 协议：路由到其他 LAN

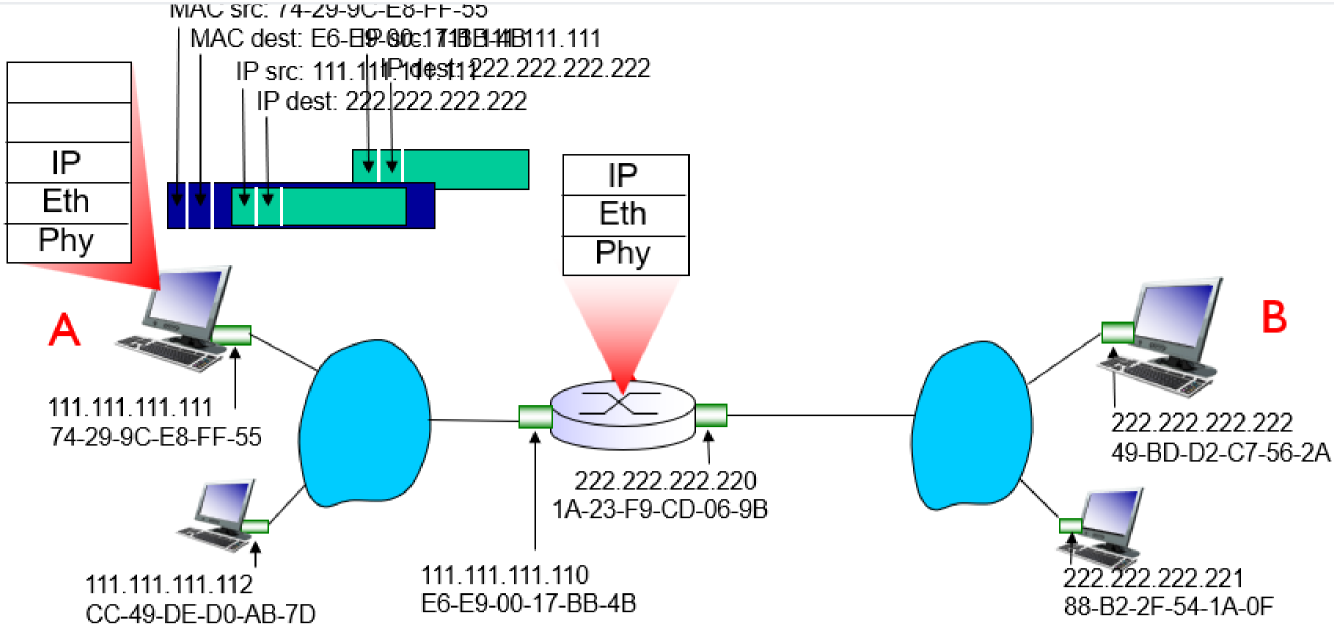
需求：A 想要通过 R 给 B 发送数据报，假设 A 知道 B 的 IP 地址

流程：

- A 创建数据报：源 IP 地址为 A，目标 IP 地址为 B 并封装一层
- A 创建一个链路层的帧：目标 MAC 地址是 R，该帧包含 A 到 B 的 IP 数据报

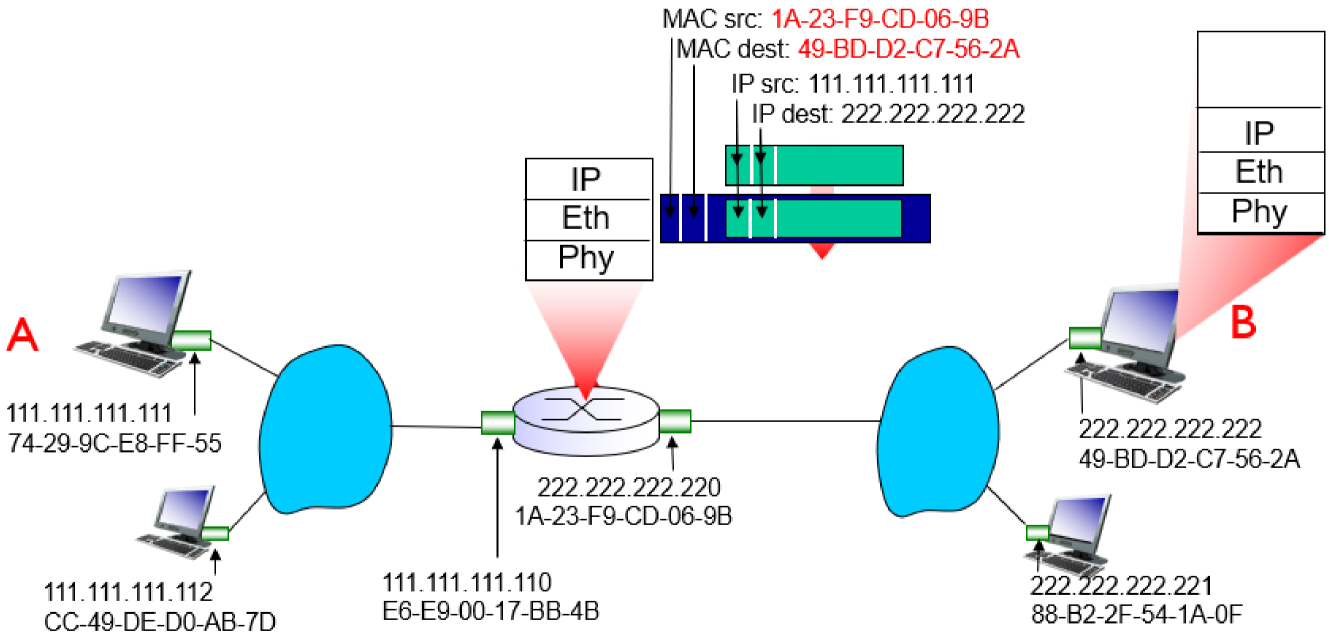


- A 将这个链路层的帧发送给 R

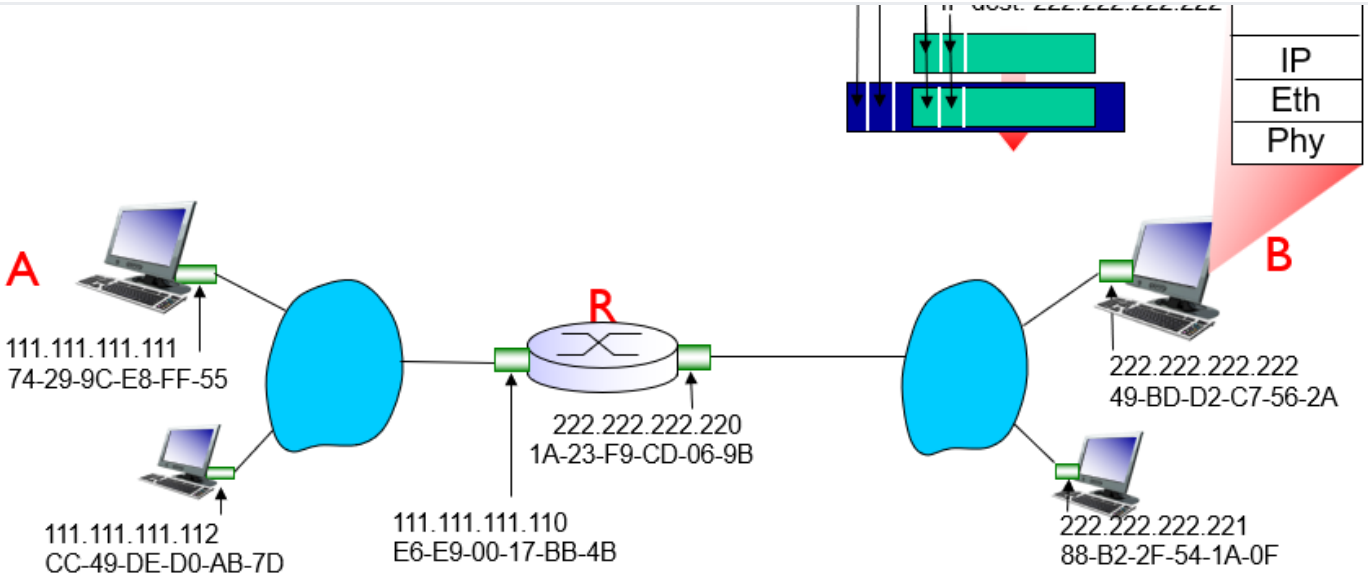


- R 转发数据报，数据报源 IP 地址为 A，目标 IP 地址为 B

> **TIP**
>
> 为了跨网络，只能使用数据报格式。



- R 创建一个链路层的帧，目标 MAC 地址为 B，帧中包含 A 到 B 的 IP 数据报

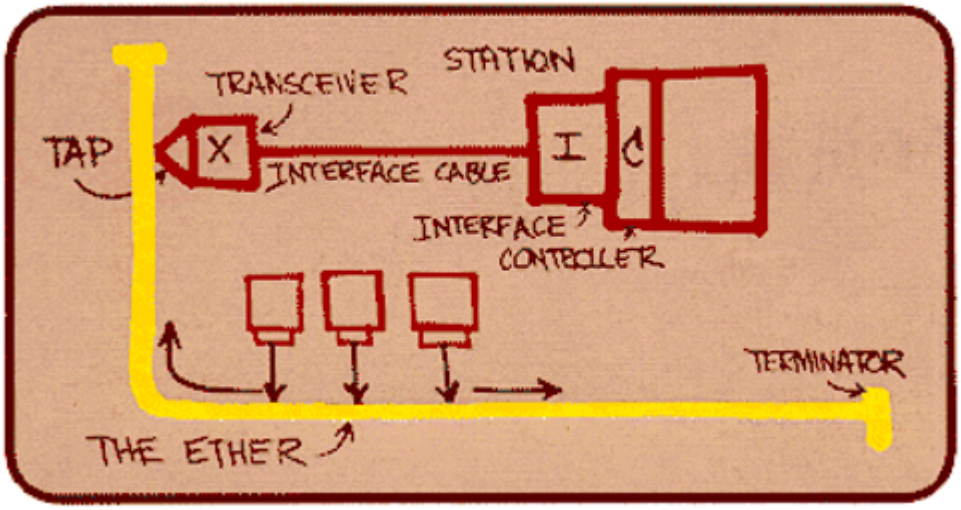


Ethernet

以太网“主导”着有线局域网技术:

- 第一个被广泛使用的局域网技术
- 占有率很高: 目前最主流的 LAN 技术拥有大概 98%占有率
- 简单、廉价
- 带宽随着发展不断提升: 10M, 100M, 1G, 10G

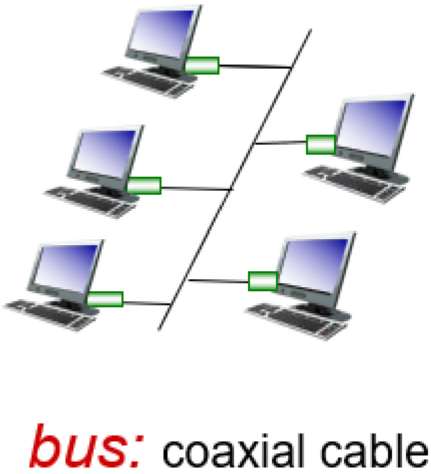
最初的以太网设计稿（仅供参考）:



Metcalfe’s Ethernet sketch

Ethernet: physical topology

- 总线结构（bus）: 在上个世纪 90 年代中期很流行



- 星型结构 (star)：目前最主流

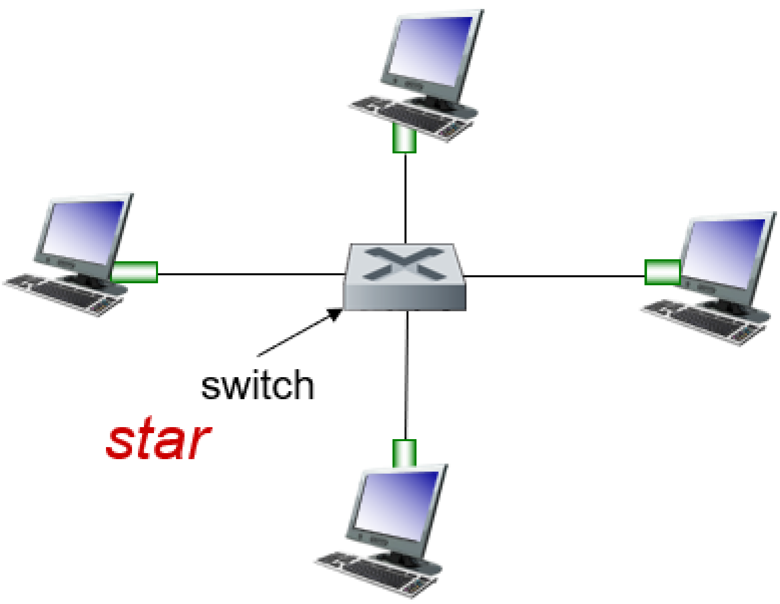
  连接选择（二选一）：

  - hub: 无法并行，一发全收
  - switch: 可以并行。switch 也叫网桥。

  > **TIP**
  >
  > 我们把:
  >
  > - hub 这种“无法并行，一发全收”叫做冲突域
  > - switch 这种的叫做广播域 (broadcast domain)

  现在一般是交换机 (switch) 在中心。

  每个节点以及相连的交换机端口使用（独立的）以太网协议（不会和其他节点的发送产生碰撞）。



以太帧结构

发送方适配器在以太网帧中封装IP数据报或其他网络层协议数据单元：

- 前导码（preamble）：（7 byte）10101010 + （1 byte）10101011

  用来同步接收方和发送方的时钟速率，使得接收方将自己的时钟调到发送端的时钟，从而可以按照发送端的时钟来接收所发送的帧。

- 地址（address）：6字节源MAC地址，目标MAC地址

  如果帧目标地址=本站MAC地址或广播地址，则接收并递交帧中的数据到网络层；否则，适配器忽略该帧。

- CRC：在接收方校验

  如果没有通过校验，丢弃错误帧。

Ethernet: unreliable, connectionless

以太网：无连接、不可靠的服务

- 无连接：帧传输前，发送方和接收方之间没有握手
- 不可靠：接收方适配器不发送ACKs或NAKs给发送方
  - 递交给网络层的数据报流可能有gap
  - 如上层使用像传输层TCP协议这样的rdt，gap会被补上（源主机，TCP实体）
  - 否则，应用层就会看到gap

以太网的MAC协议：采用二进制退避的CSMA/CD介质访问控制形式

以太网标准

IEEE的一些常见标准：

1. IEEE 802.11：无线局域网（Wireless Local Area Network，简称WLAN）
2. IEEE 802.3：有线局域网（Local Area Network，简称LAN）
3. IEEE 802.15：无线个人局域网（Wireless Personal Area Network，简称WPAN）

以太网就是第 2 个(802.3)。

以太网使用 CSMA/CD:

- 没有时隙
- NIC 如果侦听到其它 NIC 在发送就不发送：载波侦听 (carrier sense)
- 发送时，适配器当侦听到其它适配器在发送就放弃对当前帧的发送，冲突检测 (collision detection)
- 冲突后尝试重传，重传前适配器等待一个随机时间，随机访问 (random access)

Switches

这个 Switches 是链路层设备：扮演主动角色（端口执行以太网协议），主要工作：

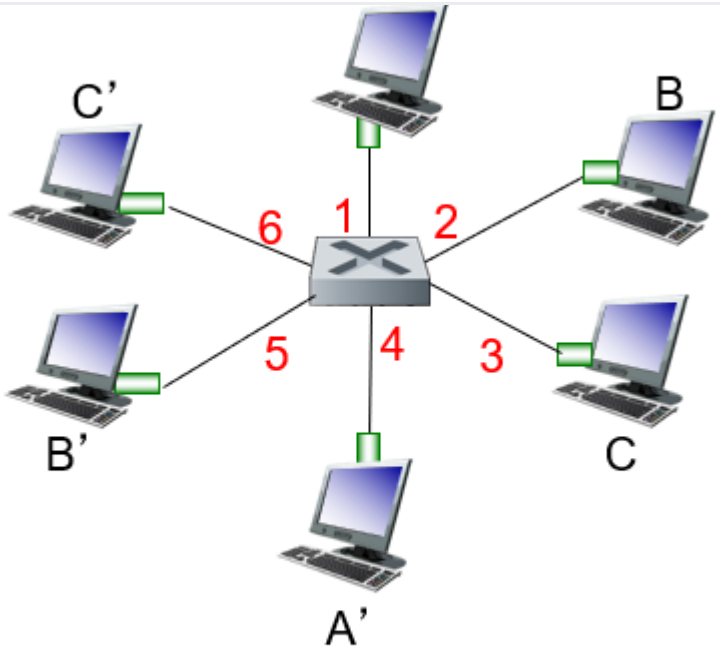
- 对帧进行存储和转发
- 对于到来的帧，检查帧头，根据目标 MAC 地址进行选择性转发
- 当帧需要向某个（些）网段进行转发，需要使用 CSMA/CD 进行接入控制通常一个交换机端口一个独立网段

交换机的特点：

- 交换机是透明的：主机对交换机的存在可以不关心，或者说，路由层面（主机）是看不到交换机的
- 交换机即插即用，自学习：交换机无需配置

交换机：多路同时传输

- 主机有一个专用和直接到交换机的连接
- 交换机缓存到来的帧；对每个帧进入的链路使用以太网协议，没有碰撞；全双工
    - 每条链路都是一个独立的碰撞域
    - MAC 协议在其中的作用弱化了
- 交换：A-to-A’ 和 B-to-B’ 可以同时传输，没有碰撞



switch with six interfaces
(1,2,3,4,5,6)

交换机：转发表

每个交换机都有一个交换表 switch table，每个表项都包含了：

- 主机的 MAC 地址
- 到达该 MAC 经过的接口
- 时间戳（time stamp，即 ttl）

比较像路由表！

交换机：自学习

交换机通过学习得到哪些主机（mac 地址）可以通过哪些端口到达：

- 当接收到帧，交换机学习到发送站点所在的端口（网段）
- 记录发送方 MAC 地址/进入端口映射关系，在交换表中

交换机：过滤/转发

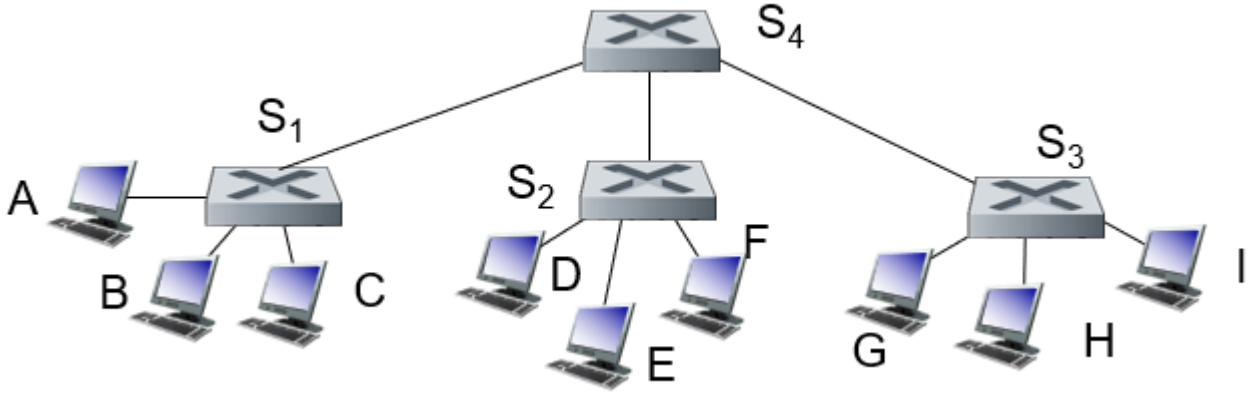
当交换机收到一个帧：

1. 记录进入链路，发送主机的 MAC 地址
2. 使用目标 MAC 地址对交换表进行索引

```py
if entry found for destination # 有目标地址
then{
    if dest on segment from which frame arrived
    then drop the frame # 过滤掉
    else forward the frame on interface indicated # 转发对应的端口
}
else flood # 泛洪：除了帧到达的网段，向所有网络接口
```

交换机级联

交换机可被级联到一起：



交换机 vs. 路由器

- 都是存储转发设备，但层次不同
    - 交换机：链路层设备（检查链路层头部）
    - 路由器：网络层设备（检查网络层的头部）
- 都有转发表：
    - 交换机：维护交换表，按照 MAC 地址转发
    - 路由器：路由器维护路由表，执行路由算法

VLANS

全称 Virtual Local Area Network，虚拟局域网。

带有 VLAN 功能的交换机（们）可以被配置成：一个物理 LAN 基础设施，虚拟成多个 LANs。

基于端口的 VLAN：

- 在 VLANs 间转发：通过路由器进行转发 (就像他们通过各自的交换机相联一样)

Link virtualization: MPLS

Link virtualization 即链路虚拟化。

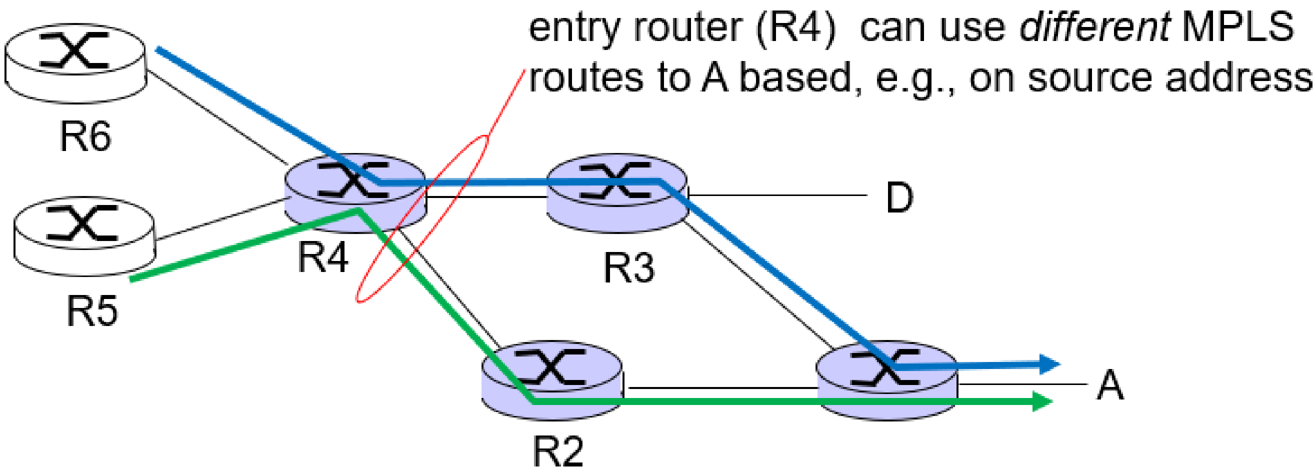
MPLS 概念：建立基于标签的转发表-信令协议：支持逐跳和显式路由：路由信息传播，路由计算(基于 Qos，基于策略的)，标签分发

MPLS 网络按照标签 label 进行分组的转发（纯 IP 网络是按照 IP 地址对分组进行转发的）。

标签交换过程

- 入口路由器：LER 对进入的分组按照 EFC 的定义打上标签
- 在 MPLS 网络中（虚拟成了链路）对分组按照标签进行交换
- 到了出口路由器，再将标签摘除
- 支持 MPLS 的路由器组构成的网络，从 IP 网络的角度来看虚拟 成了链路

MPLS vs IP 路径



- IP 路由：到达目标的路径仅仅取决于目标地址
- MPLS 路由：到达目标的路由，可以基于源和目标地址

> MPLS 路由特有的快速重新路由：在链路失效时，采用预先计算好的路径

Data center networking

数据中心网络：一般是数万-数十万台主机构成 DC 网络，密集耦合、距离临近。

- 管理/负载均衡，避免处理、网络和数据的瓶颈

负载均衡器：应用层路由

- 接受外部的客户端请求
- 将请求导入到数据中心内部
- 返回结果给外部客户端（对于客户端隐藏数据中心的内部结构）

互联

在交换机之间，机器阵列之间有丰富的互连措施：

- 在阵列之间增加吞吐 (多个可能的路由路径)
- 通过冗余度增加可靠性

A day in the life of web request

回顾：页面请求的历程

Top-down的协议栈旅程结束了！

我们主要学了应用层、运输层、网络层和链路层。

A day in the life… connecting to the Internet

- 笔记本需要一个IP地址，第一跳路由器的IP地址，DNS的地址：采用DHCP
- DHCP请求被封装在UDP中，封装在IP，封装在802.3以太网帧中
- 以太网的帧在LAN上广播（dest: FFFFFFFFFFFF），被运行中的DHCP服务器接收到
- 以太网帧中解封装IP分组，解封装UDP，解封装DHCP
- DHCP服务器生成DHCPACK包括客户端IP地址，第一跳路由器P地址和DNS名字服务器地址
- 在DHCP服务器封装，帧通过LAN转发(交换机学习)在客户端段解封装
- 客户端接收DHCP ACK应答

至此，客户端有了IP地址，知道了DNS域名服务器的名字和IP地址、第一跳路由器的IP地址

A day in the life… ARP (before DNS, before HTTP)

- 在发送 HTTP request 请求之前，需要知道www.google.com的IP地址：DNS
- DNS 查询被创建，封装在 UDP 段中，封装在 IP 数据报中，封装在以太网的帧中. 将帧传递给路由器，但是需要知道路由器的接口： MAC 地址：ARP
- ARP 查询广播，被路由器接收，路由器用 ARP 应答，给出其 IP 地 址某个端口的 MAC 地址

至此，客户端现在知道第一跳路由器 MAC 地址，所以可以发送 DNS 查询帧了

A day in the life… using DNS

- 包含了 DNS 查询的 IP 数据报通过 LAN 交换机转发，从客户端到第一跳路由器
- IP 数据报被转发，从校园到达 comcast 网络，路由（路由表被 RIP，OSPF，IS-IS 和/或 BGP 协议创建）到 DNS 服务器
- 被 DNS 服务器解封装
- DNS 服务器回复给客户端：www.google.com 的 IP 地址

A day in the life…TCP connection carrying HTTP

- 为了发送 HTTP 请求，客户端打开到达 web 服务器的 TCP socket
- TCP SYN 段（3 次握手的第 1 次握手）域间路由到 web 服务器
- web 服务器用 TCP SYNACK 应答（3 次握手的第 2 次握手）

至此，TCP 连接建立了!

A day in the life… HTTP request/reply

- HTTP 请求发送到 TCPsocket 中
- IP 数据报包含 HTTP 请求，最终路由到 www.google.com
- web 服务器用 HTTP 应答回应(包括请求的页面)
- IP 数据报包含 HTTP 应答最后被路由到客户端

Last Updated: 6/15/2023, 1:37:06 PM

Outline

1. What is network security?
2. Principles of cryptography
3. Message integrity, authentication
4. Securing e-mail
5. Securing TCP connections: SSL
6. Network layer security: IPsec
7. Securing wireless LANs
8. Operational security: firewalls and IDS

What is network security?

Network security 包含：

- 机密性：只有发送方和指定的接收方能否理解传输的报文内容，并且遵从：
    - 发送方加密报文
    - 接收方解密报文
- 认证：发送方和接收方需要确认对方的身份
- 报文完整性：发送方、接受方需要确认报文在传输的过程中或者事后没有被改变
- 访问控制和服务的可用性：服务可以接入以及对用户而言是可用的

网络交流的双方与行恶的第三方

双方可能是：

- 现实世界中的两个人
- 电子交易中的 Web browser/server
- 在线银行的 client/server
- DNS servers
- 路由信息的交换等

第三方可以做：

- 窃听：截获报文

- [illegible]

- 劫持：将发送方或者接收方踢出，接管连接

- 拒绝服务：阻止服务被其他正常用户使用（e.g.,通过对资源的过载使用）

Principles of cryptography

Principles of cryptography 即加密原理

一些专业词汇：

- plain text 纯文本
- cipher text 密文

密码学上我们把加密方式分为：

- 对称密钥密码学：发送方和接收方的密钥相同

- 公开密钥密码学：发送方使用接收方的公钥进行加密，接收方使用自己的私钥进行解密

对称密钥加密

Symmetric key cryptography 即对称密钥加密

双方共享一个对称式的密钥。

替换密码

如单码替换密码，将一个字母替换成另外一个字母（密码强度非常低）

DES: Data Encryption Standard

- 块式加密，也叫块密码（block cipher）
- US 加密标准[NIST 1993]
- 56-bit 对称密钥，64-bit 明文输入

DES 本身也很安全，破解需要非常长的时间。

> 什么是 DES 算法，详解 DES 算法的基本原理 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/575214691)
>
> DES 是一个分组加密算法，就是将明文分组进行加密，每次按顺序取明文一部分，一个典型的 DES 以 64 位为分组，加密解密用算法相同。它的密钥长度为 56 位，因为每组第 8 位是用来做奇偶校验，密钥可以是任意 56 位的数，保密性依赖于密钥。DES 算法的核心主要包括两部分，第一部分是 Feistel 结构，第二部分是 16 个子密钥的生成。

- □□ 3 □ key□ 3 □ DES □□
- 密文分组成串技术

但安全性还是在日渐下降，所以我们推出了新的加密方法。

AES: Advanced Encryption Standard

- 也是块密码
- 新的对称密钥 NIST 标准(Nov. 2001) 用于替换 DES
- 数据 128bit 成组加密
- 支持 128, 192, or 256 bit 三种长度的密钥

> 什么是 AES 加密？详解 AES 加密算法原理流程 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/562256846)
>
> AES 是高级加密标准，在密码学中又称 Rijndael 加密法，是美国联邦政府采用的一种区块加密标准。这个标准用来替代原先的 DES，目前已经被全世界广泛使用，同时 AES 已经成为对称密钥加密中最流行的算法之一。
>
> AES 支持三种长度的密钥：128 位，192 位，256 位。平时大家所说的 AES128，AES192，AES256，实际上就是指的 AES 算法对不同长度密钥的使用。

AES 加密使用多轮加密算法，每轮包括四个步骤：SubBytes、ShiftRows、MixColumns 和 AddRoundKey。这些步骤通过对明文和密钥进行一系列的替换、移位和异或操作，以实现加密。解密时，需要将加密过程中的操作逆序执行。

公开密钥密码学

对称加密模式有一个最大弱点：甲方必须把加密规则告诉乙方，否则无法解密。保存和传递密钥，就成了最头疼的问题。公开密钥密码学就是与对称密钥密码学完全不同的方法 [Diffie-Hellman76, RSA78]。

这种密码学一般都有这样的特性：

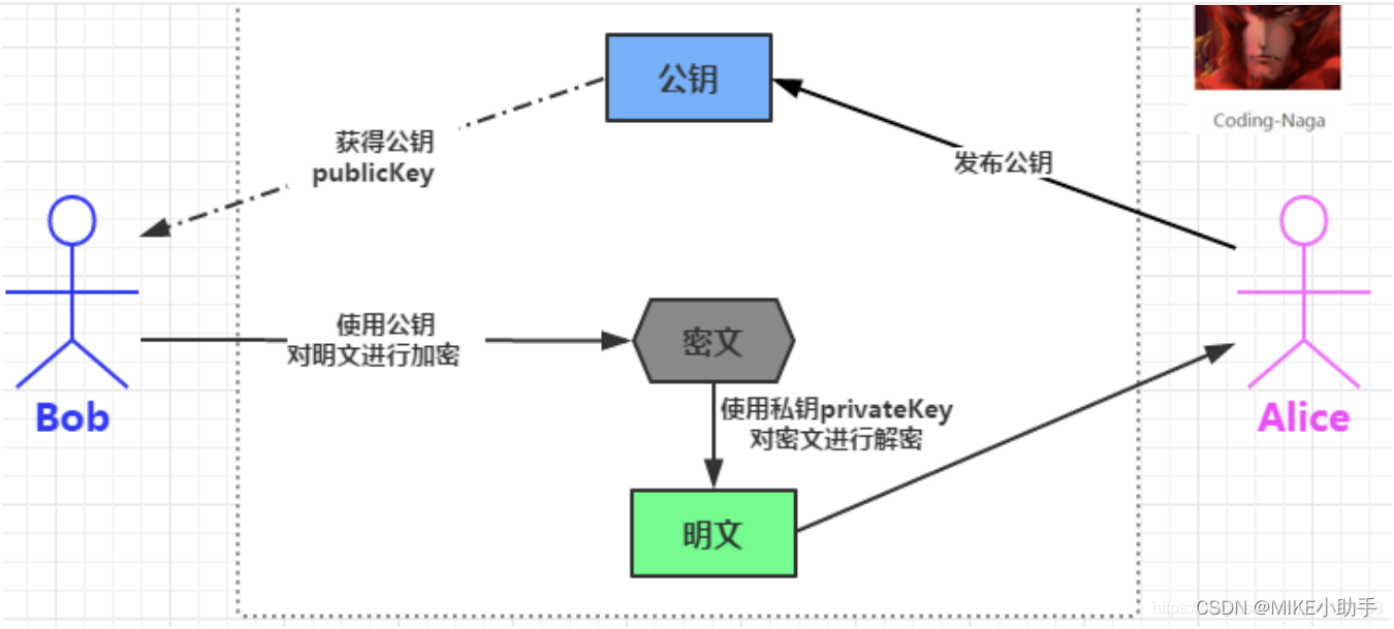
- 发送方和接收方无需共享密钥
- 一个实体的公钥公诸于众
- 私钥只有他自己知道

> RSA 算法原理（一） - 阮一峰的网络日志 (ruanyifeng.com) (https://ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html)
>
> 1976 年，两位美国计算机学家 Whitfield Diffie 和 Martin Hellman，提出了一种崭新构思，可以在不直接传递密钥的情况下，完成解密。这被称为"Diffie-Hellman 密钥交换算法" (https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange)。这个算法启发了其他科学家。人们认识到，加密和解密可以使用不同的规则，只要这两种规则之间存在某种对应关系即可，这样就避免了直接传递密钥。
>
> 这种新的加密模式被称为"非对称加密算法"。
>
> 1. 乙方生成两把密钥（公钥和私钥）。公钥是公开的，任何人都可以获得，私钥则是保密的。
> 2. 甲方获取乙方的公钥，然后用它对信息加密。
> 3. 乙方得到加密后的信息，用私钥解密。
>
> 如果公钥加密的信息只有私钥解得开，那么只要私钥不泄漏，通信就是安全的。

> 1977 年，三位数学家 Rivest、Shamir 和 Adleman 设计了一种算法，可以实现非对称加密。这种算法用他们三个人的名字命名，叫做RSA 算法(https://zh.wikipedia.org/zh-cn/RSA%E5%8A%A0%E5%AF%86%E7%AE%97%E6%B3%95) 。从那时直到现在，RSA 算法一直是最广为使用的"非对称加密算法"。毫不夸张地说，只要有计算机网络的地方，就有 RSA 算法。

RSA

> 公开密钥加密之 RSA 算法【概念+计算+代码实现】_rsa 算法代码_MIKE 笔记的博客-CSDN 博客(https://blog.csdn.net/m0_51607907/article/details/123884953)
>
> 密钥计算方法
>
> 1. 选择两个大素数 p 和 q(典型值为 1024 位)
> 2. 计算 `n=p×q` 和 `z=(p-1)×(q-1)` (其中 n 表示欧拉函数)
> 3. 选择一个与 z 互质的数，令其为 d
> 4. 找到一个 e 使满足 `exd= 1 (mod z)`
> 5. 公开密钥为 `(e, m)` ，私有密钥为 `(d, m)`
>
> 加密方法
>
> 1. 将明文看成比特串，将明文划分成 k 位的块 P 即可，这里 k 是满足 $2*k<n$ 的最大整数。
> 2. 对每个数据块 P，计算 $C= P^e(mod\ n)$，而 C 即为 P 的密文。
>
> 解密方法
>
> 对每个密文块 C，计算 $P=C^d(mod\ n)$，P 即为明文。
>
> ### RSA 算法流程图



> 代码实现: 2️⃣ 代码实现 (csdn.net) (https://blog.csdn.net/m0_51607907/article/details/123884953#t14) 有时间可以看看学一学。

RSA 的一个重要的特性:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{先用公钥，然后用私钥}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{先用私钥，然后用公钥}}$$

Exponentiation in RSA is computationally intensive. RSA 中的求幂运算是计算密集型的。

Message integrity, authentication

Authentication

即信息完整性与认证

- Protocol ap1.0: 直接发送对方信息
- Protocol ap2.0: 对方信息中加入了 IP 地址
- Protocol ap3.0: 对方信息中加入了自己的密码证明
- Protocol ap3.1: 对方信息中加入了自己的加密之后的密码来证明

- Protocol ap4.0：为了证明 Alice 的活跃性，Bob 发送给 Alice 一个 nonce, R。Alice 必须返回加密之后的 R，使用双方约定好的 key。

> Nonce：一生只用一次（*once-in-a-lifetime*）的整数 R

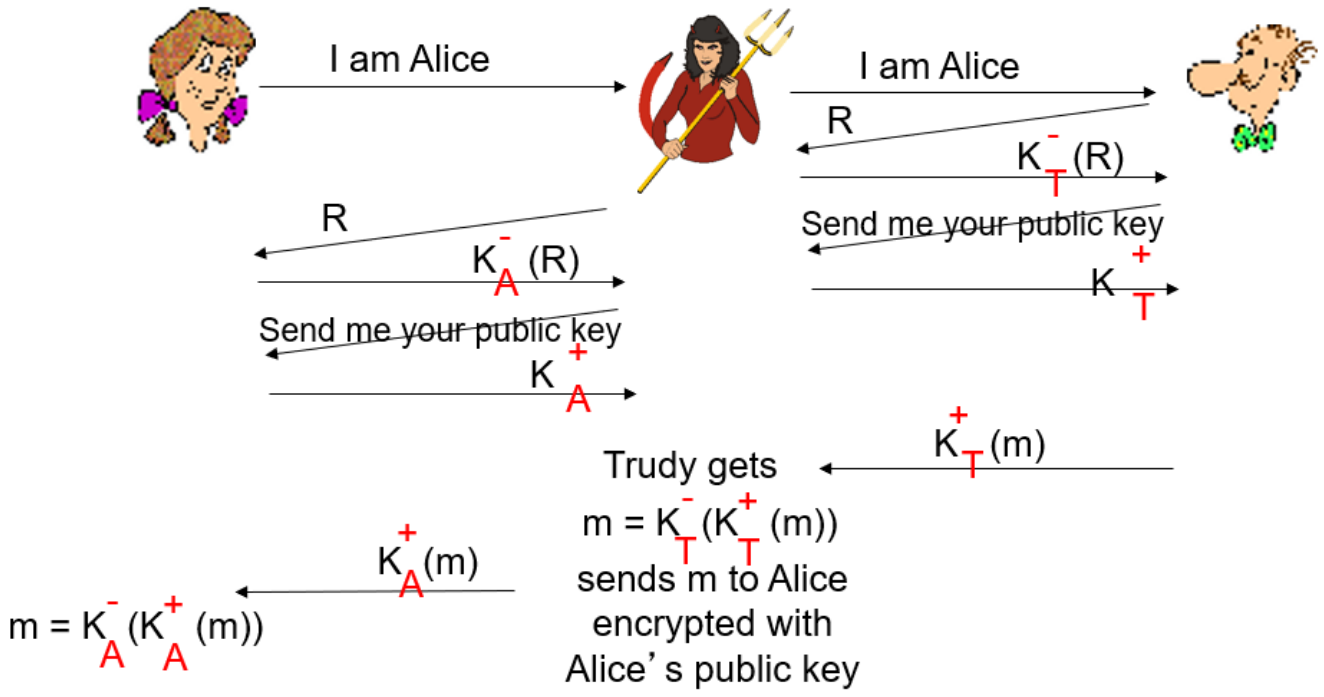
问题是需要双方共享一个对称式的密钥。这个协议的缺点是需要共享秘钥！

- Protocol ap5.0：使用 nonce，公开密钥加密技术

（安全漏洞 —— 中间攻击）仍然存在问题：

  - Trudy 用自己的私钥加密 R，传给 Bob 公钥
  - Trudy 截获 Alice 的信息和公钥，用 Alice 的公钥解密她的私钥

至此 Bob 与 Alice 之间的信息全被 Trudy 截获



Message integrity

Digital signatures

即数字签名。我们类比于手写签名。

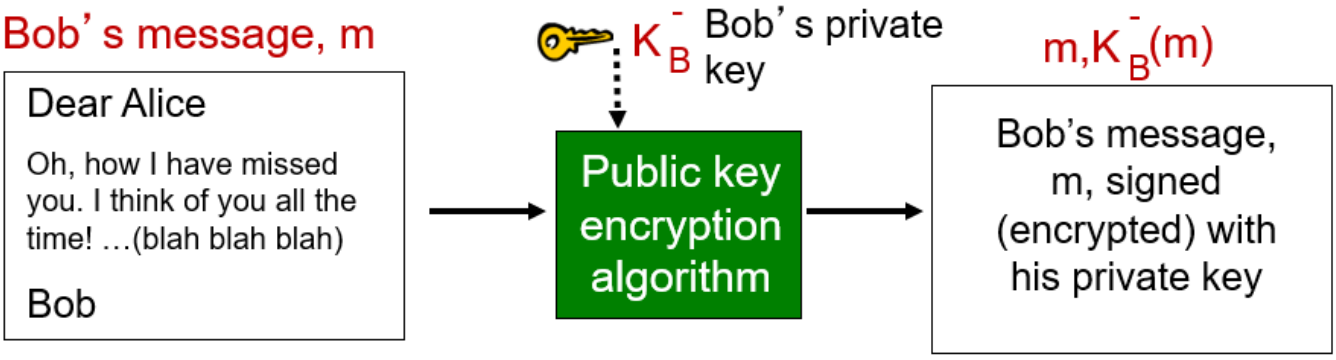
特性：

- 可验证性 *verifiable*
- 不可伪造性 *nonforgeable*
- 不可抵赖性 *non-repudiation*

- [illegible]Alice[illegible] Bob[illegible] Alice[illegible]
- [illegible]

[illegible]

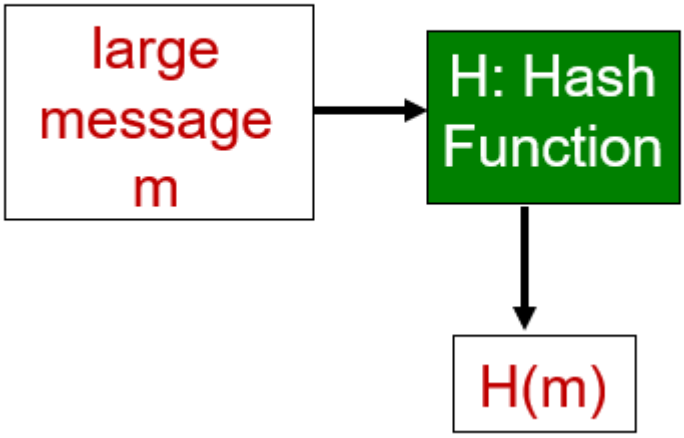
- [illegible] [illegible]Bob [illegible] m [illegible] [illegible] $K^-_B (m)$



Message digests

[illegible]Message digests[illegible]

[illegible] m [illegible] H[illegible] H(m)[illegible]



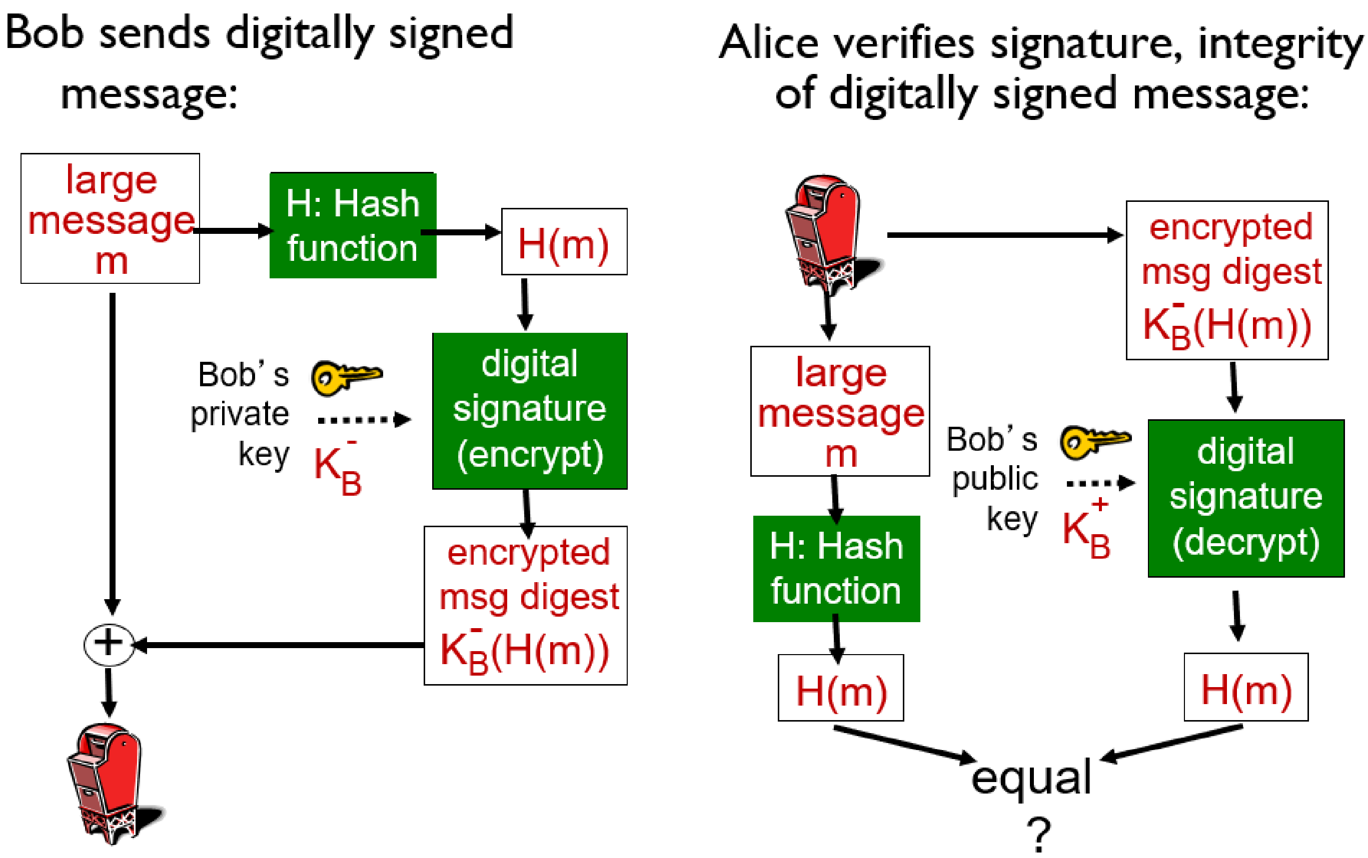
[illegible]

- [illegible] 1
- [illegible]
- [illegible] x[illegible] $x = H(m)$

- Alice □□ Bob □□□ K_B □ $K^-_B (m)$ □□□□□ □□ $K^+_B(K^-_B(m)) = m$ □□□□□

  如果成立，那么签署这个文件的人一定拥有 Bob 的私钥。

数字签名 = 对报文摘要进行数字签署：



Hash function algorithms

> 摘自 密码学家王小云：十年破解 MD5 和 SHA-1 两大国际密码-新华网 (xinhuanet.com) (http://www.xinhuanet.com/politics/2019-12/27/c_1125394020.htm)，有删改。
>
> 时间回到 2004 年，对于国际密码学界来说，这注定是不同寻常的一年。
>
> 这年的 8 月，在美国加州圣巴巴拉召开的国际密码大会上，王小云宣读了自己和研究团队对于 MD4、MD5、HAVAL-128 和 RIPEMD 四个国际著名密码算法的破译结果。
>
> 这被认为是 2004 年密码学界最具突破性的结果，堪称学术界的一场强烈地震。当年国际密码大会总结报告上写道：我们该怎么办？MD5 被重创了，它即将从应用中淘汰。SHA-1 仍然活着……
>
> 多年来，由美国国家标准技术研究院（NIST）颁布的基于哈希函数的 MD5 和 SHA-1 算法，是国际上公认最先进、应用范围最广的两大重要算法，后者更被视为计算安全系统的基石，有着“白宫密码”之称。
>
> 没多久，SHA-1 的末日降临。2005 年 2 月，在美国召开的国家信息安全研讨会上，5 名著名密码学家公布了哈希函数发展史上的重要研究进展———他们收到了来自中国的王小云等 3 位女研究者对 SHA-1 全算法的攻击。
>
> 2005 年，美国《新科学家》杂志在一篇文章中，用了颇具震撼力的标题———《崩溃！密码学的危机》，报道了王小云团队花 10 年时间取得的学术成果。
>
> 2006 年，NIST 颁布了美国联邦机构 2010 年之前必须停止使用 SHA-1 的新政策，并于次年向全球密码学者征集新的国际标准密码算法。
>
> “天书”哈希函数到底是什么
>
> 今天，计算机网络、移动网络、物联网、卫星网络还有大数据、云计算，这些人们已经熟知的科技场景，都离不开密码技术的支撑，需要密码来解决安全问题。
>
> 王小云曾将密码比作钥匙：“没有密码的保障，就相当于有人偷了家里的钥匙，可以随时自由进出你家，而你却浑然不知。”
>
> 密码学重要到何种地步？不得不从一个密码学中的基本工具说起，它就是王小云打了多年交道的哈希函数。

> 不过，数字签名算法必须和哈希函数一起才能保证可认证性和不可抵赖性。因此，五大安全属性里有三个，都离不开哈希函数。
>
> 密码上的哈希函数，可以将任意长度的消息压缩成固定长度的哈希值，而哈希值就像每个人都拥有唯一的“指纹”一样，哈希函数的重要之处就是能够赋予每个消息唯一的“数字指纹”，即使更改该消息的一个比特，对应的哈希值也会变为截然不同的“指纹”。
>
> 清华大学高等研究院数学博士吴彦冰打了个比方，就像把一本书里的某一页或一个字更改了，但看书的人很难判断更改的地方，即便全书通读一遍也未必能发现，“但通过哈希函数，输入稍有不同，输出结果就会完全不同”。
>
> ### 解密的惊心动魄更多在内心
>
> 吴彦冰曾听老师王小云讲起破解 MD5 的经历，“那时候王老师还没学过编程，就用手写推导的方式，写了 400 多页纸，几百个方程，推导了两三个月才得到结果。”
>
> 王小云沉浸在密码的美妙世界中，享受着外人无法体会的乐趣。生活中她喜欢在家里和实验室养花，有时候思考一个数学问题，却找不到答案时，就会起来打扫打扫卫生，或者是给花浇浇水，干点别的事情，但实际上脑子里一直没有放下科学问题。
>
> 科研过程中，王小云也把这种乐趣传递给学生们。他们既扮演着“设谜者”的角色，设计一套巧妙的密码算法，弥补前人的不足，希冀让攻击者无法突破；同时还扮演着“猜谜人”的角色，把别人设计精妙的体系一举攻破。“参与者会有很大的喜悦感和成就感。”吴彦冰说。
>
> 就如王小云在未来科学大奖的获奖致辞中所说，虽然目前密码学只被少数人所熟悉，但未来会有更多的年轻力量为密码学的发展助力，愿意尽自己的全力去帮助年轻的科学家们开拓密码学这门神秘而又充满力量的学科。（记者完颜文豪、李牧鸣）

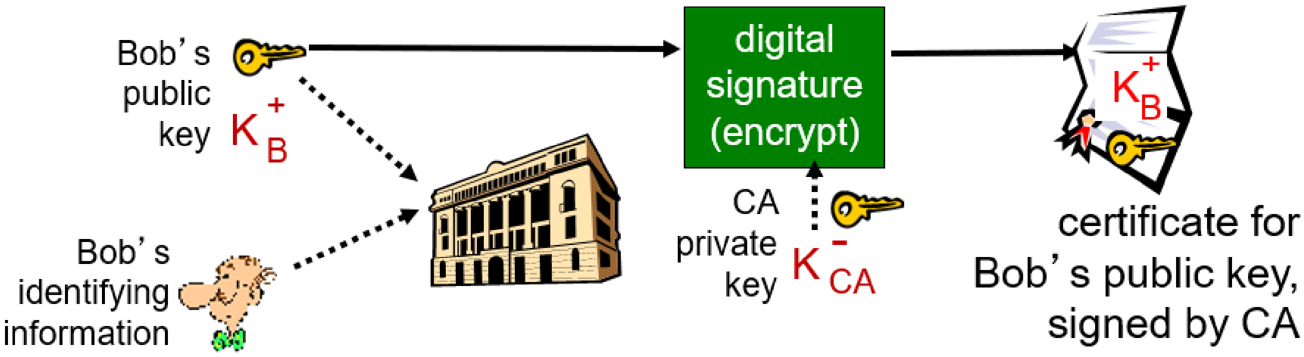
计算机中常见的摘要算法（散列函数算法）：

- MD5(RFC 1321 标准，不再完全安全，但是算法简单快速、已经被广泛传播使用，常用于下载文件校验)
- SHA-1(US 标准，不再完全安全，同上)
- SHA-128
- SHA-256(国际上最为常用的算法)
- SM3(由国内王小云开发的算法，尽管目前为止绝对安全，但国际认可度不高)

Public-key certification

主要由可信赖中介完成。

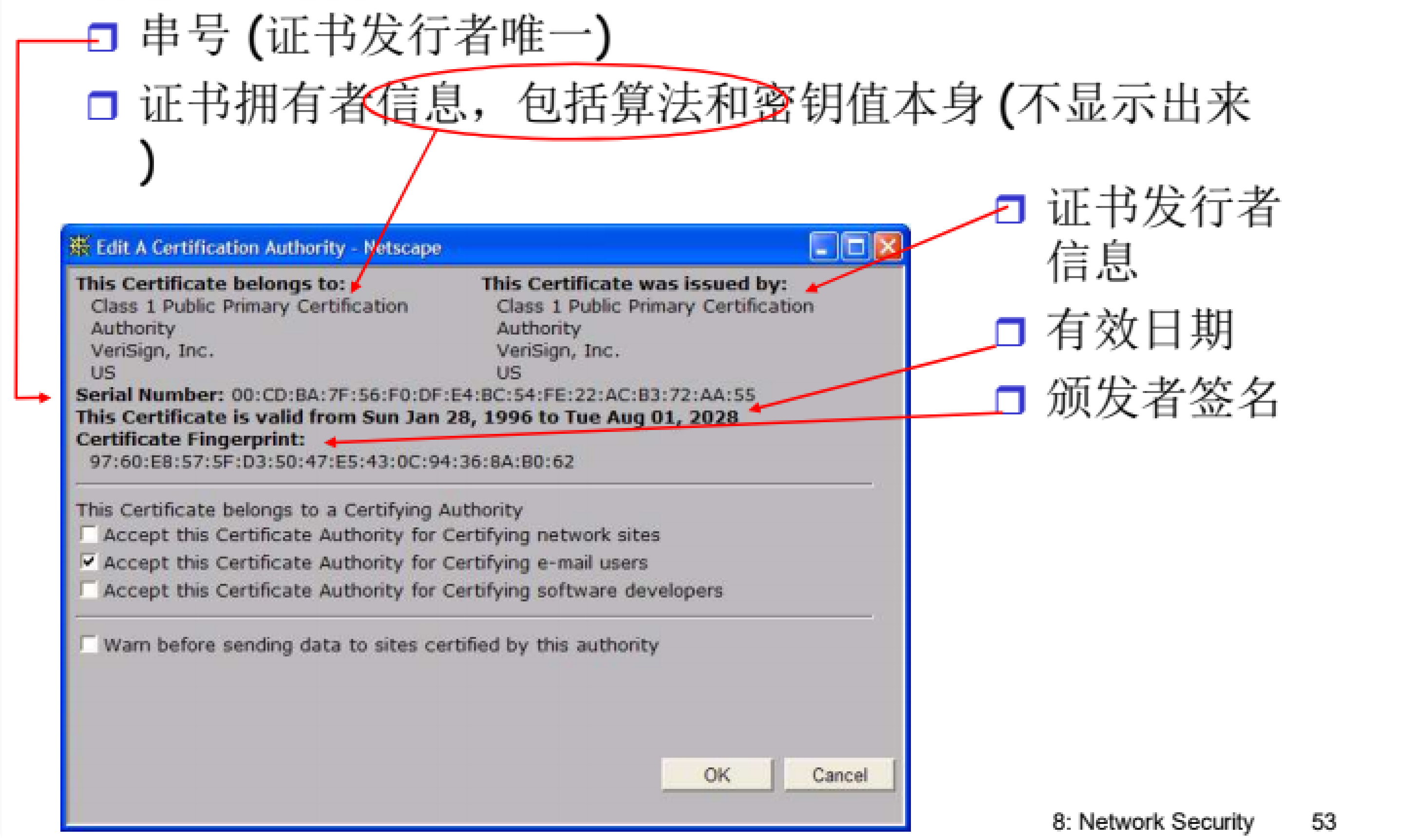
- Trusted key distribution center (KDC) 在实体之间扮演可信赖中介的角色；
- 服务器和每一个注册用户都分享一个对称式的密钥(many users)



- 该可信赖中介的可信赖的 certification authority (CA) 专门颁布可信赖的证书：

  CA 创建一个证书，捆绑了实体信息和他的公钥，而且是被 CA 签署的** (被 CA 用自己的私钥加了密的) **

- CA 说 “this is E’s public key”。



- 串号 (证书发行者唯一)
- 证书拥有者信息，包括算法和密钥值本身 (不显示出来)

Edit A Certification Authority - Netscape

This Certificate belongs to:
Class 1 Public Primary Certification
Authority
VeriSign, Inc.
US

This Certificate was issued by:
Class 1 Public Primary Certification
Authority
VeriSign, Inc.
US

Serial Number: 00:CD:BA:7F:56:F0:DF:E4:BC:54:FE:22:AC:B3:72:AA:55
This Certificate is valid from Sun Jan 28, 1996 to Tue Aug 01, 2028
Certificate Fingerprint:
97:60:E8:57:5F:D3:50:47:E5:43:0C:94:36:8A:B0:62

This Certificate belongs to a Certifying Authority
☐ Accept this Certificate Authority for Certifying network sites
☑ Accept this Certificate Authority for Certifying e-mail users
☐ Accept this Certificate Authority for Certifying software developers

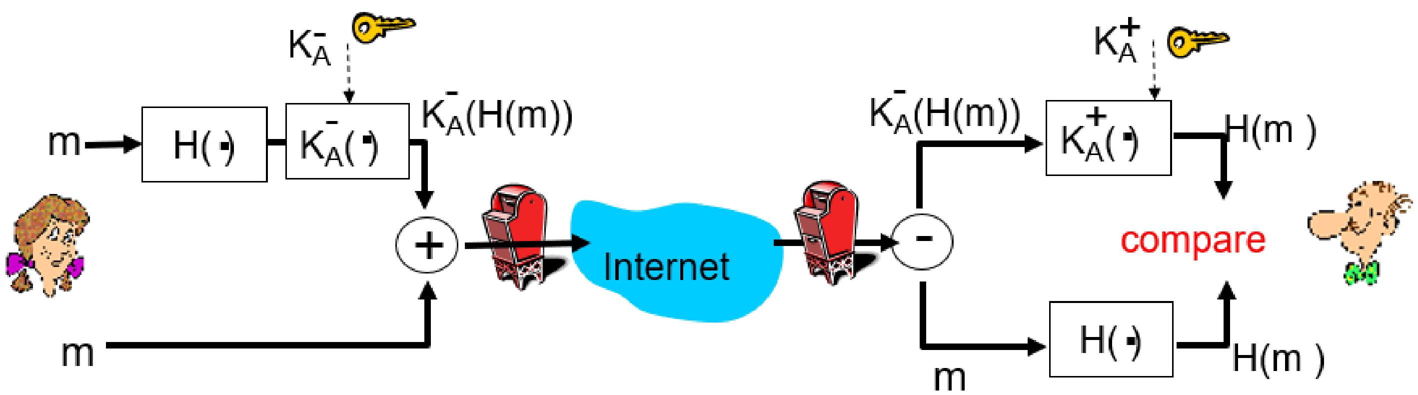
☐ Warn before sending data to sites certified by this authority

- 证书发行者信息
- 有效日期
- 颁发者签名

8: Network Security 53

Securing e-mail

这是一个例子，这里我放几张截图得了，反正就是之前讲的这一套安全系统。

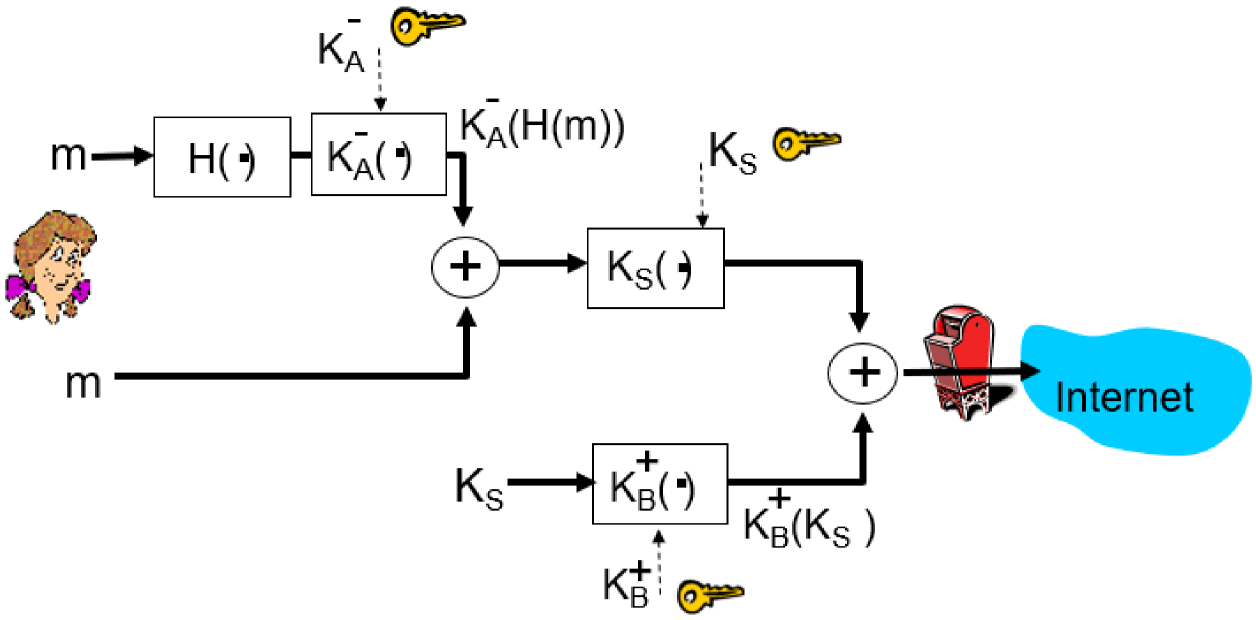


Alice:

- 产生随机的对称密钥 K_S
- 使用 KS 对报文加密(为了效率)
- 对 K_S 使用 Bob 的公钥进行加密
- 发送 $K_S(m)$ 和 $K_B(K_S)$ 给 Bob

Bob:

- 使用自己的私钥解密 K_S

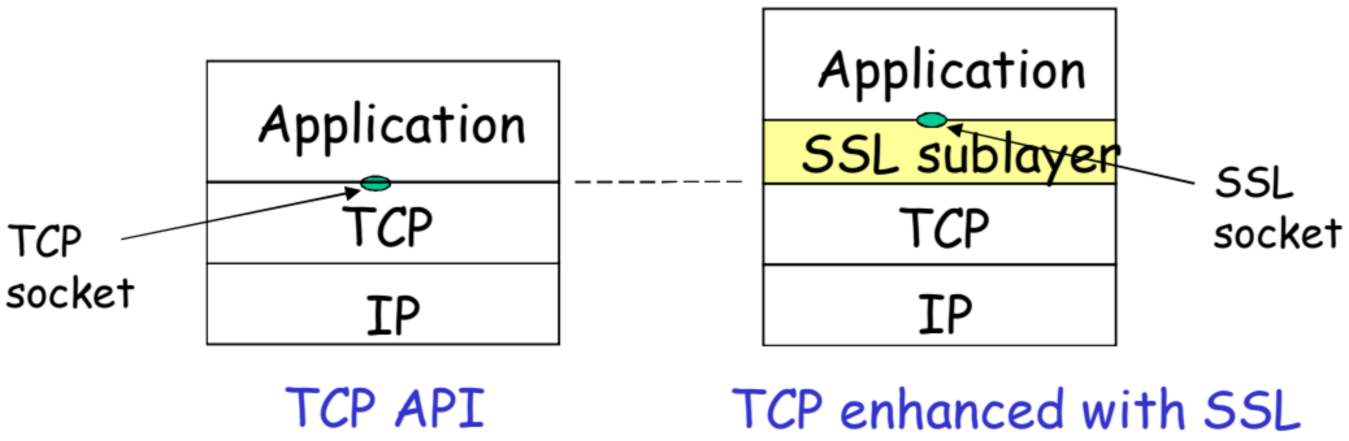


- Alice 使用了 3 个 keys: 自己的私钥，Bob 的公钥，新产生出的对称式密钥

Securing TCP connections: SSL

SSL: Secure Sockets Layer

为使用 SSL 服务的、基于 TCP 的应用提供传输层次的安全性。



实现了:

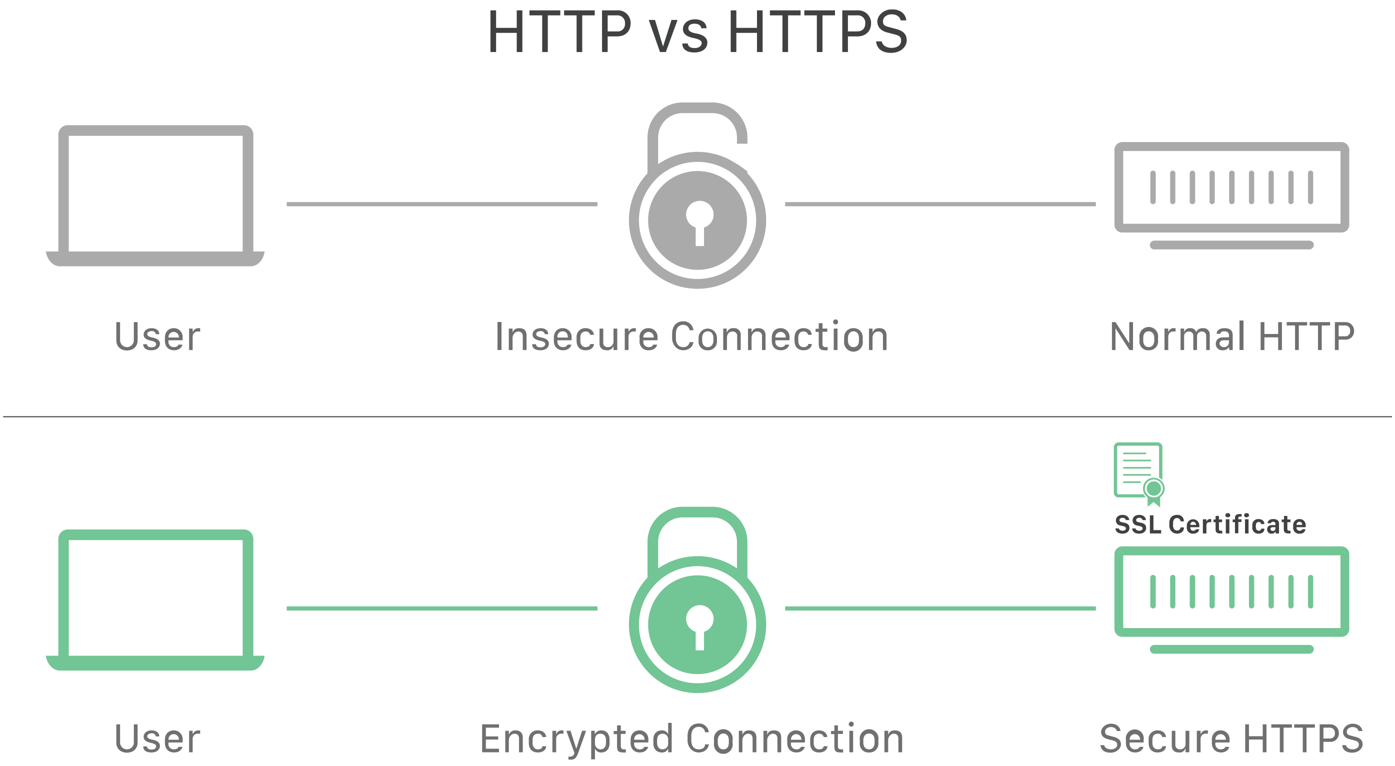
- *confidentiality*
- *integrity*
- *authentication*

> ssl 是什么-ssl 和 tls 的区别-SSL 证书 | Cloudflare (https://www.cloudflare.com/zh-cn/learning/ssl/what-is-ssl/)

什么是SSL？

安全套接字层（SSL）是一种加密安全 (https://www.cloudflare.com/learning/ssl/what-is-encryption/) 协议 (https://www.cloudflare.com/learning/network-layer/what-is-a-protocol/) 。它最初由Netscape于1995年开发，旨在确保Internet通信中的隐私、身份验证和数据完整性。SSL是如今使用的现代TLS (https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/) 加密的前身。

实施SSL/TLS的网站的URL中带有“HTTPS (https://www.cloudflare.com/learning/ssl/what-is-https/) ”，而不是“HTTP” (https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/) 。



SSL/TLS如何工作？

- 为了提供高度隐私 (https://www.cloudflare.com/learning/privacy/what-is-data-privacy/) ，SSL会对通过Web传输的数据进行加密。这意味着，任何试图截取此数据的人都只会看到几乎无法解密的乱码字符。
- SSL在两个通信设备之间启动称为握手 (https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/) 的身份验证过程，以确保两个设备确实是它们声称的真实身份。
- SSL还对数据进行数字签名，以提供数据完整性，验证数据是否在到达目标接收者之前被篡改过。

SSL已经过多次迭代，安全性逐代增强。SSL在1999年更新为TLS。

SSL/TLS为何重要？

最初，Web上的数据是以明文形式传输的，任何人只要截获消息都可以读取。例如，如果消费者访问了购物网站，下了订单并在网站上输入了他们的信用卡号，那么该信用卡号将不加隐藏地在Internet上传播。

创建SSL就是为了纠正此问题并保护用户隐私。通过对用户和Web服务器之间传输的所有数据进行加密，SSL可确保截获数据的人只能看到混乱的字符。消费者的信用卡号现在可以确保安全，仅在他们输入卡号的购物网站上可见。

SSL还可以阻止某些类型的网络攻击：它对Web服务器进行身份验证，这非常重要，因为攻击者通常会尝试建立伪造网站来欺骗用户并窃取数据。它还可以防止攻击者篡改传输中的数据，就像药品容器上的防篡改封条一样。

SSL和TLS是同一回事吗？

> 由于它们紧密地联系在一起，这两个术语经常互换使用并混为一谈。有些人仍然使用 SSL 来指代 TLS，其他人则使用术语“SSL/TLS 加密”，因为 SSL 仍然具有很大的知名度。
>
> ### SSL 仍然没有落伍吗?
>
> SSL 自 1996 年推出 SSL 3.0 以来未曾更新过，现已弃用。SSL 协议中存在多个已知漏洞，安全专家建议停止使用。实际上，大多数现代 Web 浏览器已彻底不再支持 SSL。
>
> TLS 是依然在网络上实施的最新加密协议，尽管有许多人仍将其称为“SSL 加密”。这可能会使购买安全解决方案的消费者感到困惑。事实上，如今提供“SSL”的任何供应商提供的几乎肯定都是 TLS 保护，这已成为二十多年来的行业标准。但是，由于许多人仍在搜寻“SSL 保护”，因此这个术语在许多产品页面上仍然处于醒目位置。
>
> ### 什么是 SSL 证书?
>
> SSL 只能由具有 SSL 证书 (https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/) （技术上称为“TLS 证书”）的网站来实现。SSL 证书就像身份证或徽章一样，证明某人就是他们所说的真实身份。SSL 证书由网站或应用程序的服务器存储并显示在 Web 上。
>
> SSL 证书中最重要的信息之一是网站的公共密钥 (https://www.cloudflare.com/learning/ssl/what-is-a-cryptographic-key/) 。公钥 (https://www.cloudflare.com/learning/ssl/how-does-public-key-encryption-work/) 使得加密和身份验证成为可能。用户的设备查看公钥，并使用它与 Web 服务器建立安全的加密密钥。同时，Web 服务器还具有一个保密的私有密钥。私钥解密使用公钥加密的数据。
>
> 证书颁发机构（CA）负责颁发 SSL 证书。
>
> ### SSL 证书有哪些不同类型?
>
> 有几种不同类型的 SSL 证书 (https://www.cloudflare.com/learning/ssl/types-of-ssl-certificates/) 。一个证书可以应用于一个或多个网站，具体取决于类型：
>
> - **单域：**单域 SSL 证书仅适用于一个域（“域”是网站的名称，例如 www.cloudflare.com）。
> - **通配符：**与单域证书一样，通配符 SSL 证书仅适用于一个域。但是，它也包括该域的子域。例如，通配符证书可以覆盖 www.cloudflare.com、blog.cloudflare.com，和 developers.cloudflare.com，而单域证书只能覆盖第一个。
> - **多域：**顾名思义，多域 SSL 证书可以应用于多个不相关的域。
>
> SSL 证书还具有不同的验证级别。验证级别就像背景检查一样，并且级别会根据检查的彻底性而变化。
>
> - **域验证：**这是最严格的验证级别，也是最便宜的级别。企业只需要证明他们控制着域。
> - **组织验证：**这是一个需要亲力亲为的过程：证书机构直接联系请求证书的人员或企业。这些证书更受用户信赖。
> - **扩展验证：**在发出 SSL 证书之前，需要对组织进行全面的背景检查。

Pretty good privacy (PGP)

Internet e-mail 加密方案，事实上的标准。该方案是一个商业方案，后来也有了开源的、非商业化的 GPG 方案，与 PGP 基本完全相同。

```
---BEGIN PGP SIGNED MESSAGE--
   -
Hash: SHA1

Bob:My husband is out of town
   tonight.Passionately
   yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3
   mqJhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

Hello

Welcome to your VuePress site

✨ Get Started

Introduction

Simplicity First

从最简单最基础的开始。

Powerful

也许你不信，这或许是非常优秀的笔记项目。

Neatly formatted

认真、仔细排练文档格式，让你的阅读体验更加舒适。

Computer Network Learning

Some notes and code about CWorld learning Computer Network.

Get Started □□□□ → (https://cn.cworld.top/)

Local Development

Environment requirements:

- Node.js (https://nodejs.org) 16.14.0+

1. Enable corepack & pnpm

> If your Node.js version is lower than 16.13.0□Please install corepack (https://nodejs.org/api/corepack.html) first.
>
> ```sh
> npm install -g corepack
> ```

```sh
corepack enable
corepack prepare pnpm@latest --activate
```

2. Clone the repository

```sh
git clone https://github.com/cworld1/cn-learning.git
cd cn-learning
```

2. Install dependencies

```sh
pnpm install
```

3. Start the development server

```sh
pnpm dev
```

This command starts a local development server and opens up a browser window. Most changes are reflected live without having to restart the server.

4. Some useful commands

`pnpm build` Bundles your website into static files for production.

Contributions

As the author is only a beginner in learning Computer Network, there are obvious mistakes in his notes. Readers are also invited to make a lot of mistakes. In addition, you are welcome to use PR or Issues to improve them.

Thanks

Some of the electronic textbooks have helped the author a lot in his studies, and without them, this notebook would not have been possible. I would like to express my gratitude to the original authors of these materials. If you have any doubts about this project, you can also read the following textbooks carefully to remedy them.

- STATS 201 : Computer Network (https://courseoutline.auckland.ac.nz/dco/course/STATS/201/1215)
- college_assignment/□□□□□ · GitHub (https://github.com/A-BigTree/college_assignment/blob/main/learning_Notes/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C.md#56-icmp%E5%9B%A0%E7%89%B9%E7%BD%91%E6%8E%A7%E5%88%B6%E6%8A%A5%E6%96%87%E5%8D%8F%E8%AE%AE)
- □□-□□□□□-□□□□ | FEZ□□□ (https://toby-fish.github.io/2021/11/22/%E7%AC%94%E8%AE%B0-%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C-

- Jim Kurose (author’s website of Computer Networking: a Top Down Approach (P) (http://gaia.cs.umass.edu/kurose_ross/index.php)

License

> **Note** This website is built using Vuepress Next (https://github.com/vuepress/vuepress-next) , a Vuejs (https://vuejs.org) static website generator.

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

At the last

至此已经全部结束了！感谢您看到最后。

下面是考试相关。

考试分数分布

1. Single choice and Ferminlogy ($102 + 51$ score)
2. Multiple choice fill blanks ($52 + 151$ score)
3. Problem and analysis (25 score)
4. Configuration (25 score)

重要考点

1. 单选题：重点在 3456 章，部分在 2 章。如：

    - 数据链路层常见的设备，如 Hub、Switch、冲突域、广播域等知识点。
    - 或可靠或不可靠协议的语义，如 IP、TCP、UDP 等。语法涉及少。FTP、HTTP、Email 等常见的协议（Socket）端口号要考
    - 怎么使用 IP 地址
    - TCP 三次握手、四次挥手
    - 什么协议属于什么层的

2. 多选题：尽量选满

    填空题：

    - FSM 有限状态机（可靠的数据传递）
    - TCP 问题
    - 主要在传输层、数据链路层

3. 理论题

    - 帧和 IP 包（ARP、帧的转换、路由表等技术）
    - IP 地址的规划（子网划分等技术）

    如：把 IP 地址分成 9 个子网应该怎么分，尽量采用聚合地址或余间路由技术（Cider）

4. 实验题：考实验 3（以实验报告的形式，尽量写的详尽，完整的如何配置这个网络的），并且队访问控制部分没有太多要求，静态路由、端口设置、IP 地址设置等是重点考察。

  - Configuring the static route, where you can see all the routes are configured in the router.

例子

现在有资源：202.202.96.0/21

现在分配：

- 500*1
- 250*2
- 120*3
- 60*2
- 30*3

1. 21位为网络号，剩下11位为主机号，所以可以分配2^11-2=2046个主机。
2. 我们拿两位来分配给子网，所以可以分配2^2=4个子网（00、01、10、11）。
3. 分配：
   - 202.202.96.0/23，500大小，即子网掩码为255.255.250.0
   - 202.202.98.0/24，250大小
   - 202.202.99.0/24，250大小
   - 202.202.100.

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

Getting Started

课程目标

1. 理解网络结构及其基本原理
2. 独立构建部门级小型交换局域网
3. 进行网络规划，构建企业级局域网，及其与广域网的连接

实验环境

机房采用 19 英寸标准网络机柜。

专业词汇：

- Access Controller（AC）：控制器
- Walfare：防火墙
- Router：路由器
- Switch：交换机
- Distribution Frame：配线架

机柜包含：

- **PC** 网线接口：直接连接其支配的所有电脑
- **PCIe** 卡（计算机网卡）： 2 张，一张用于接入交换机统一管理，另一个用于直接接入线架
- **COM** 口：多用户系统时使用，用于短程通讯，相当于终端口，用于终端管理
- 机柜主机接口：也是终端口，用于控制箱内主机/路由器，也是短程通讯，大概不能超过 15 米
- 交换机：一般用于连接不同的设备。一般在配线架上有多个接口，如 S1L1 表示 Server 1 Link 1
- 路由：主要用于连接网络，或者与不同路由器形成错综复杂的局域网。路由同时也是互联网的核心，不同路由相互连接形成互联网。一般在配线架上有多个端口，如 R1L1 表示 Route 1 Link 1
- 防火墙：分为两类。代理服务器防火墙：一般在应用层；电路网关防火墙：在网络层、传输层。

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

1. 网络基础知识及双绞线的制作

实验目的

- 熟悉和了解实验室的网络设备和环境，认识常见的网络设备外观，基本构造；
- 了解和学习 TCP/IP 协议的基本原理，学习如何进行基本的网络地址规划。
- 掌握双绞线的制作方法和制作工艺；
- 掌握双绞线的接线标准 T568A、T568B；
- 掌握双绞线的检测方法，了解常见的连接方式；
- 认识常见的网络设备外观，基本构造；
- 了解小型局域网的组建方式。
- 掌握 TCP/IP 协议的设置方法。

实验原理

IP 地址介绍

- IP 地址唯一标识一台网络设备
- IP 地址以点分十进制的方式进行表示
- IP 地址通常分为网络位和主机位两部分

IP 地址分四个字段，不同字段之间用点隔开。其中每个字段支持 8 位数的二进制编码，也就是说，在十进制层面上最高支持 255（即二进制的 11111111）。

私有地址范围：

1. 10.0.0.1 ~ 10.255.255.254
2. 172.16.0.1 ~ 172.31.255.254
3. 192.168.0.1 ~ 192.168.255.254

子网掩码：用来指明一个 IP 地址的哪些位标识的是主机所在的子网，以及哪些位标识的是主机的位掩码。子网掩码不能单独存在，它必须结合 IP 地址一起使用。

默认子网掩码：

类别	子网掩码的二进制数值	子网掩码的十进制数值
A	11111111 00000000 00000000 00000000	255.0.0.0
B	11111111 11111111 00000000 00000000	255.255.0.0

C	11111111 11111111 11111111 00000000	255.255.255.0

我们一般遇到的子网掩码就是类别 C，意思是前三个字段代表网络部分，最后一个字段代表主机部分。

协议

- 物理协议：不同设备接线、机械通讯协议，是最底层的协议
- 网络协议：如 IP（Internet Proxy）
- 软件协议：软件自己的协议如 Twitter 通讯协议、Tiktok 软件通讯协议

常见网络设备的拓扑标志

双绞线

双绞线是由两条相互绝缘的导线按照一定的规格互相缠绕在一起儿子做成的一种通用配线，属于信息通信网络传输介质。

1. 传输速率大致在 100Mb/s 左右，最好不要超过 100 米。
2. 不要超过一定的角度扭曲，会影响传输速率（电路干扰）。
3. 有两个水晶头连接到其他设备：新的连接互联网的水晶头能满足 `RJ45` 物理层协议；旧的家用固话一般使用协议 `RJ11` 。
4. 检测双绞线是否有问题，可通过“专用测试电表”测试，测试正常且成功的情况下，会有绿灯成排向上移动“闪烁”。
5. 使用通断仪可以测量双绞线的实际速率。当然也可以直接连接电脑等设备测试速率。

双绞线分类

绞线分为屏蔽双绞线 STP（Shield Twisted-Pair）和非屏蔽双绞线 UTP（Unshielded Twisted-Pair）。UTP 由多对双绞线和一个绝缘外皮构成，是目前使用最为广泛的传输介质。

按照 EIA/TIA 568A 标准，UTP 可分为 1~6 类，其中计算机使用的是 3、5、6 类，分别提供 16、100、250MHz 的带宽。

构建局域网常采用 CAT-5 或 CAT-5e 类线。双绞线制作有两种接线标准，即 T568A 和 T568B 标准。其中 T568B 标准是 100MHz 局域网中常用的接线方式。两者物理层面的差异是在 1、3 脚位和 2、6 脚位交叉。

T568A 标准：

脚位	1	2	3	4	5	6	7	8
颜色	绿白	绿	橙白	蓝	蓝白	橙	棕白	棕

T568B 标准：

□□	□□	□	□□	□	□□	□	□□	□

按双绞线的两端线序分可以分为直连线（Straight Line）和交叉线（Cross-over Line）。直连双绞线：双绞线的两端线序相同；交叉双绞线：一端为 T568A 标准，另一端为 T568B 标准，当两个计算机互相连接为非常规方式，有时可能会用到这种。

	计算机	路由器	交换机 **MDIX**	交换机 **MDI**
计算机	交叉	交叉	直连	N/A
路由器	交叉	交叉	直连	N/A
交换机 MDIX	直连	直连	交叉	直连
交换机 MDI	N/A	N/A	直连	交叉

实验步骤

制作双绞线

1. 使用夹线器夹断线，大约 20~30cm
2. 使用夹线器的孔位使内部双绞线裸露，大约裸露 2~3cm
3. 简单理清双绞线顺序，尽可能贴近 [[T568B 标准]]
4. 使用夹线器的平板夹位使双绞线长短一致，大约裸露 1.5cm
5. 将双绞线顺序再次整理，对着水晶头金属片方向插入水晶头内
6. 检查双绞线顺序，如无误，将水晶头插入夹线器的水晶头孔，注意插入方向、双绞线松动情况
7. 压下夹线器，制作完成

测试双绞线连通

1. 将专业双绞线测试器的顶部的左右两个端口对接上测试双绞线的两头
2. 打开测试器，调节到合适挡位
3. 观察指示灯变化，正常情况下会有 ”1-1“、”2-2“ ... 灯光逐步跳动，表示两个端口的对应双绞线连接正确

测试双绞线的电脑连接能力

1. 使用两条双绞线分别将两台 PC 的端口 `PC*` 连接至同一台交换机的不同接口 `S*L*`
2. 修改电脑网络设置：
    1. 前往系统设置 → 网络和 Internet → 高级适配器设置
    2. 禁用默认的本地连接，启用额外通道（如本地连接 1），此时如果正常会提示缆线已接入，否则会看到红色的“❌”图标提示
    3. 设置额外通道属性，并修改其 IPv4 协议配置，IP 地址设置为 `192.168.0.*`（注意不同 PC 的设置），子网掩码为 `255.255.255.0`
3. 打开终端工具，如 CMD、Powershell

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

2. 交换机配置及 VLAN 建立

> VLAN：Local Virtal Area Network，即本地虚拟局域网络

实验目的

- 了解交换机功能，学会使用 Windows 操作系统上的超级终端程序，通过交换机的控制口简单配置交换机。
- 学习一些交换机的基本配置命令。
- 学习查看配置过程中的出错信息和各种配置信息。
- 学习查看相关操作手册和帮助文档。
- 学习 VLAN 的建立、配置和调试命令。
- 学习如何实现 VLAN 间通讯，掌握交换机端口模式的使用。

实验原理

连接方式

交换机的终端控制一般使用串专用串口或 Telnet。其中专用串口方式的连接一般为：

交换机 Console 口 ↔ 交换机 Console 线 ↔ PC 机串口（COM 口）

> 注：
>
> 1. 串口，也叫 COM 口（Cluster Communication Port），即串行通讯端口。
> 2. 注意连接不能太远，最好控制在 15 米以内（实验用的线为 RS232）。

以太网端口链路类型

以太网端口有三种链路类型有三种：Access、Hybrid 和 Trunk。

- Access 类型的端口只能属于一个 VLAN，一般用于连接计算机的端口；
- Trunk 类型的端口可以属于多个 VLAN，可以接收和发送多个 VLAN 的报文，一般用于路由器之间连接的端口；
- Hybrid 类型的端口可以属于多个 VLAN，可以接收和发送多个 VLAN 的报文，可以用于路由器之间连接，也可以用于连接用户的计算机。

> Hybrid 端口和 Trunk 端口的不同之处在于 Hybrid 端口可以允许多个 VLAN 的报文发送时不打标签，而 Trunk 端口只允许缺省 VLAN 的报文发送时不打标签。

Access	□□□
Trunk	主机通讯
Hybrid	计算机和主机通讯均可

终端常用命令

> 三种视图:
>
> 1. 用户视图: `<switch>`
> 2. 系统视图: [`switch`]
> 3. 接口视图: [`switch-Ethernet**`]
> 4. VLAN 视图: [`switch-vlan**`]

通用命令:

- `quit | return` :返回用户视图/退出当前端口
- `undo xxx` :取消执行某个命令
- `reboot` :重启当前交换机
- `display history-command` :显示历史命令,命令行接口为每个用户缺省保存 10 条历史命令。临时调用命令可以尝试:
    - Ctrl+P 或 ↑: 上一条历史命令
    - Ctrl+N 或 ↓: 下一条历史命令
- TAB: 快速补全命令

用户视图下:

- `system-view` :进入系统视图

系统视图下:

- `reset saved-configuration` :重置配置 (需要重启应用配置)
- `display saved-configuration` :查看当前配置 其中部分显示信息:
    - `telnet server` :通过网络 (Telnet) 配置交换机
    - `interface Vlan-interface*` :连接上游主机,其 ip 可以填写某个指定的 ip,也可以填写 dhcp-alloc (动态主机分配协议) ,使用上游主机自动分配 ip (ip 协议一般用来连接互联网)
- `sysname <switch-name>` :给交换机重新命名
- `vlan <vlan_id>` :创建指定名称的 VLAN (VLAN 最多能定义 4094 个)
- `display|displ <ethernet>` :查看某个端口的状态
- `(interface) <ethernet>` 进入某个端口 (如 ethernet0/1) 的视图
- `save` :保存当前配置到本地,需要手动确认并命文件名 (文件名可以默认) 。

- `display interface`
- `speed 10|100|1000|auto`：设置端口的速度（如 10 代表 10M）
- `duplex auto|full|half`：设置端口的模式（自动、双工、半双工）
- `port link-type access|hybrid|trunk`：设置以太网端口的链路类型（Access、Hybrid 或 Trunk）
- `port access vlan <vlan_id> (to vlan_id)`：将该端口加入到指定 VLAN 下
- `port hybrid vlan <vlan_id_list> tagged|untagged`：将 Hybrid 端口加入到指定 VLAN
- `port trunk permit vlan <vlan_id_list>|all`：将当前以太网端口加入到指定 VLAN
- `shutdown`：关闭当前端口

VLAN 视图下：

- `display vlan`：查看当前 VLAN 信息
- `port <interface_list>`：将指定端口（或端口列表）添加到该 VLAN 下。

实验环境

- A PC with Windows XP, Super Shell（超级终端）
- A Switch with 相关操作手册
- A Console 线缆（RS232 型号）

实验步骤

基础实验内容

1. 使用 Console 线将交换机的 Console 口与电脑 COM 口相连。
2. 在 PC 机上运行超级终端程序进行连接：
    1. 新建配置并为之命名，指定 COM 口。
    2. 设置终端通信参数为波特率（位/秒）为 9600bit/s，数据位 8 位，无校验，停止位 1 位和无流控制。
3. 连接终端：交换机上电，终端上开始显示以太网交换机的自检信息，自检结束后提示用户键入回车，之后将出现命令行提示符，如`<Quidway>`并可能伴随显示 `login from Console` 等提示，即表示交换机正确连接成功。
4. 重置配置文件：
    1. `reset saved-configuration`：还原初始配置（用户视图下使用）
    2. `reboot`：重启交换机（用户视图下使用）
    3. `display  current-configuration`：显示当前配置信息
5. `sysname SwitchA`：修改交换机名称

实验 1：在同一台交换机上划分 VLAN

STEP 2：在6台计算机完全连通之后，将同在一侧的三台计算机所在的端口划分为一个VLAN，另一侧的三台计算机所在的端口划分为另一个VLAN（VLAN编号自由设定）。利用ping命令测试其连通性，可以发现同属一个VLAN计算机可以连通，而不同VLAN计算机不能连通。删除VLAN，六台计算机恢复全连通状态。

1. `vlan 10`：创建VLAN
2. `vlan 10`：进入VLAN 10
3. `port ethernet 1/0/1 to ethernet 1/0/4`：将1~4端口接入到VLAN10内
4. `quit`：退出VLAN 10
5. 各台电脑均接入交换机，并设置好IP地址，互相使用 `ping` 指令测试连通性。
6. `vlan 11`：创建VLAN 11
7. `vlan 11`：进入VLAN 11
8. `port ethernet 1/0/3 to ethernet 1/0/4`：将3~4端口接入到VLAN11内
9. `quit`：退出VLAN 11
10. 3、4电脑互相 `ping` 进行测试，并对1、2电脑尝试 `ping` 测试。

实验2：同一个VLAN的跨交换机通讯

1. `interface ethernet 1/0/1`：进入端口 `ethernet 1/0/1`
2. `port trunk`：设置当前以太网端口的链路类型为Trunk
3. `port trunk permit vlan 11`：将当前以太网端口加入到VLAN 11
4. 使用相同操作设置另外一台交换机
5. 将两台交换机的端口1相互连接
6. 对两台交换机对应的3、4电脑互相 `ping` 进行测试，并对2电脑尝试 `ping` 测试。

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

3. 三层交换及 VLAN 间路由

网关(Gateway)又称网间连接器、协议转换器。网关在传输层上以实现网络互连，是最复杂的网络互连设备，仅用于两个高层协议不同的网络互连。网关的结构也和路由器类似，不同的是互连层。网关既可以用于广域网互连，也可以用于局域网互连。

顾名思义，网关(Gateway)就是一个网络连接到另一个网络的“关口”。如三层交换机或路由器等。

实验目的

- 进一步理解三层交换机配置的基本原理。
- 掌握三层交换机的基本命令配置。
- 掌握不同组网通信含义。

实验原理

系统视图下：

- `interface vlan-interface <vlan_id>`：创建 VLAN 接口或进入 VLAN 接口视图。（Undo 删除）

> - VLAN 接口视图和 VLAN 视图是不同的；
> - VLAN 路由接口（或简称 VLAN 接口）与 VLAN 具有对应关系，是整个 VLAN 的逻辑接口，只有在已经建立了某个 VLAN 的情况下，才能为该 VLAN 建立路由接口。
> - 在 2 层交换机上，由于只有一个默认 VLAN（VLAN1），只能为 vlan 1 建立路由接口，而在三层交换机下，可以为每个 VLAN 建立一个逻辑接口。

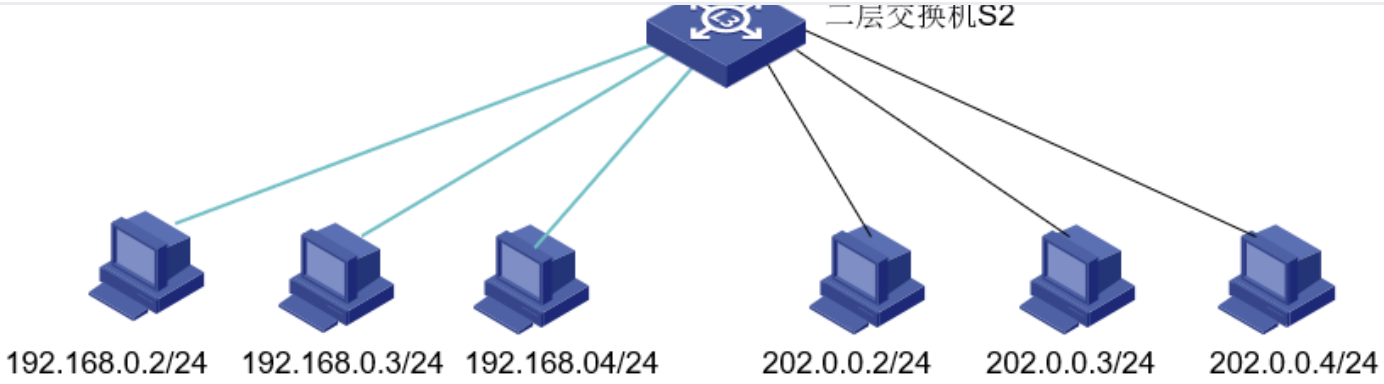
接口视图下：

- `ip address ip-address net-mask (sub)`：给 VLAN 接口指定 IP 地址和掩码。

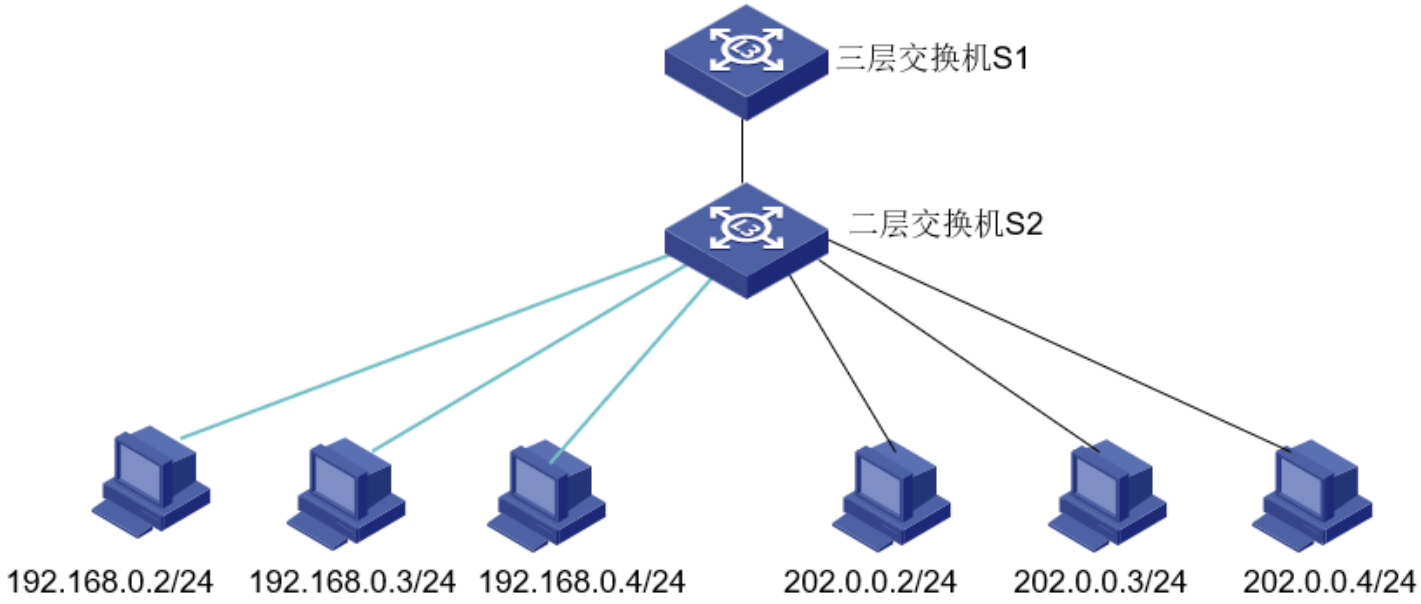
实验步骤

实验 1：在三层交换机上实现不同网段相同 VLAN 计算机互联

STEP 1：按图示搭建网络环境，每 6 人为一组，均通过网线连接在 2 层交换机 S2 的 1-6 号口。为各计算配置 IP 地址，同侧的三台计算机在同一网段，然后用 ping 命令测试其连通性，六台计算机属于同一 VLAN 的情况下属于同一网段的计算机应该都能连通。



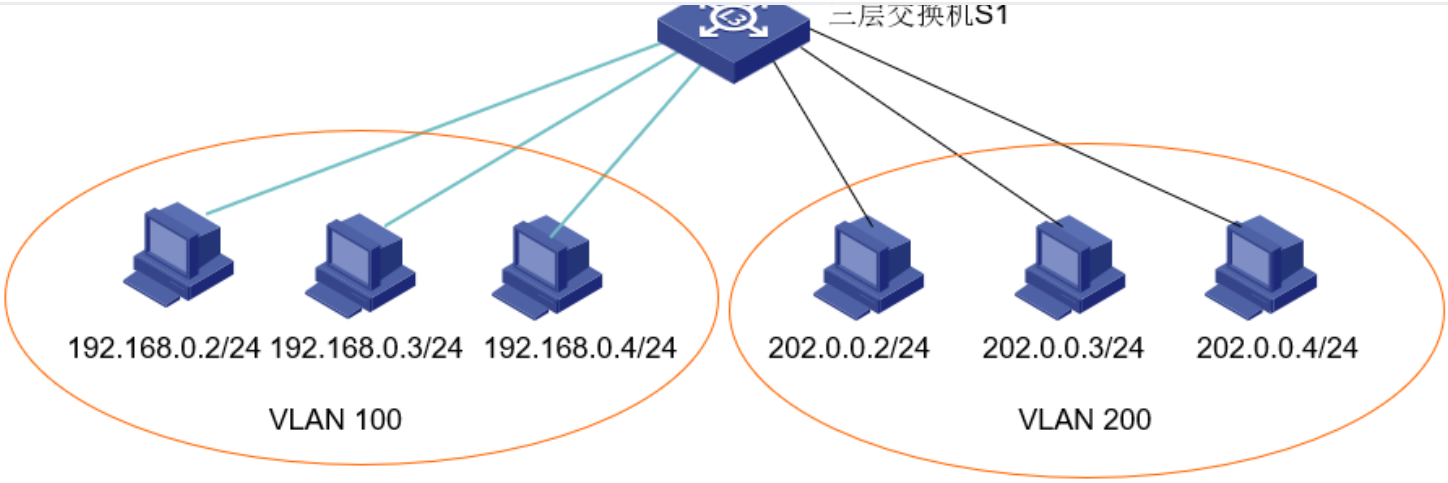
STEP 2: 将二层交换机 S2 级联到三层交换机 S1，在 S1 上创建 VLAN 路由接口，并在 VLAN 接口视图下配置路由接口的主从 IP 地址，然后测试不同网段间计算的连通性。



1. `system-view` ：进入系统视图
2. 链接三层交换机 S1，建立 vlan 1 的接口视图，并设置其 ip 地址：
    1. `vlan 1` ：进入 vlan 1 视图
    2. `interface vlan 1` ：为 vlan 1 创建路由接口
    3. `ip address 192.168.0.1 255.255.255.0` ：为 vlan 1 的路由接口配置 IP 地址
    4. `ip address 202.0.0.1 255.255.255.0 sub` ：为 vlan 1 的路由器接口配置子 IP 地址
3. 使用 `ping` 命令测试连通性。

实验 2：不同网段不同 VLAN 的计算机互联

STEP 1：按图示搭建网络环境，每 6 人为一组，均通过网线连接在 2 层交换机 S2 的 1-6 号口。为各计算配置 IP 地址，同侧的三台计算机在同一网段同一 VLAN。然后用 ping 命令测试其连通性，同侧三台计算机应该能连通。



STEP 2：在交换机 S1 上，建立 VLAN 100 和 VLAN 200 的路由接口，并分别设置 ip 地址，该 ip 应和 VLAN 内的计算机 ip 地址处于同一网段内。将计算机网段地址设置为对应的 VLAN 接口地址，通过 ping 命令测试其连通性。

1. `vlan 100` ：创建 vlan 100 并进入其视图
2. `port e1/0/1 to e1/0/3` ：将 1-3 号以太口加入 VLAN100
3. `vlan 200` ：创建 vlan 200 并进入其视图
4. `port e1/0/4 to e1/0/6` ：将 4-6 号以太口加入 VLAN200
5. `int vlan 100` ：创建 vlan100 的路由接口
6. `ip address 192.168.0.1 255.255.255.0` ：配置 IP 地址
7. `int vlan 200` ：创建 vlan200 的路由接口
8. `ip address 202.0.0.1 255.255.255.0` ：配置 IP 地址

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

4. 路由器基本配置及静态路由配置

实验目的

了解路由器的功能和组成结构。掌握路由器的基本配置方法。掌握静态路由和动态路由协议 RIP 的配置。

实验原理

初步认识路由器

路由器 - 工作在 OSI 参考模型第三层的网络设备：用于网络互连的设备。

作为路由器，必须具备：

- 两个或两个以上的接口
- 协议至少向上实现到网络层
- 具有存储、转发、寻径功能

路由器基本组成：

1. 转发模块：通过查找路由表记录，判断下一个转发目标
2. 端口模块：有支持 ADSL 路由器，支持专线线路路由器，通过端口模块，对接收到的包进行接收进来，这一步取决于端口所对应的通信技术。如果是以太网，则按照以太网规则进行接收，如果是无线局域网，则按照无线局域网进行接收。总之，目的就是委托端口的硬件对网络包进行接收。

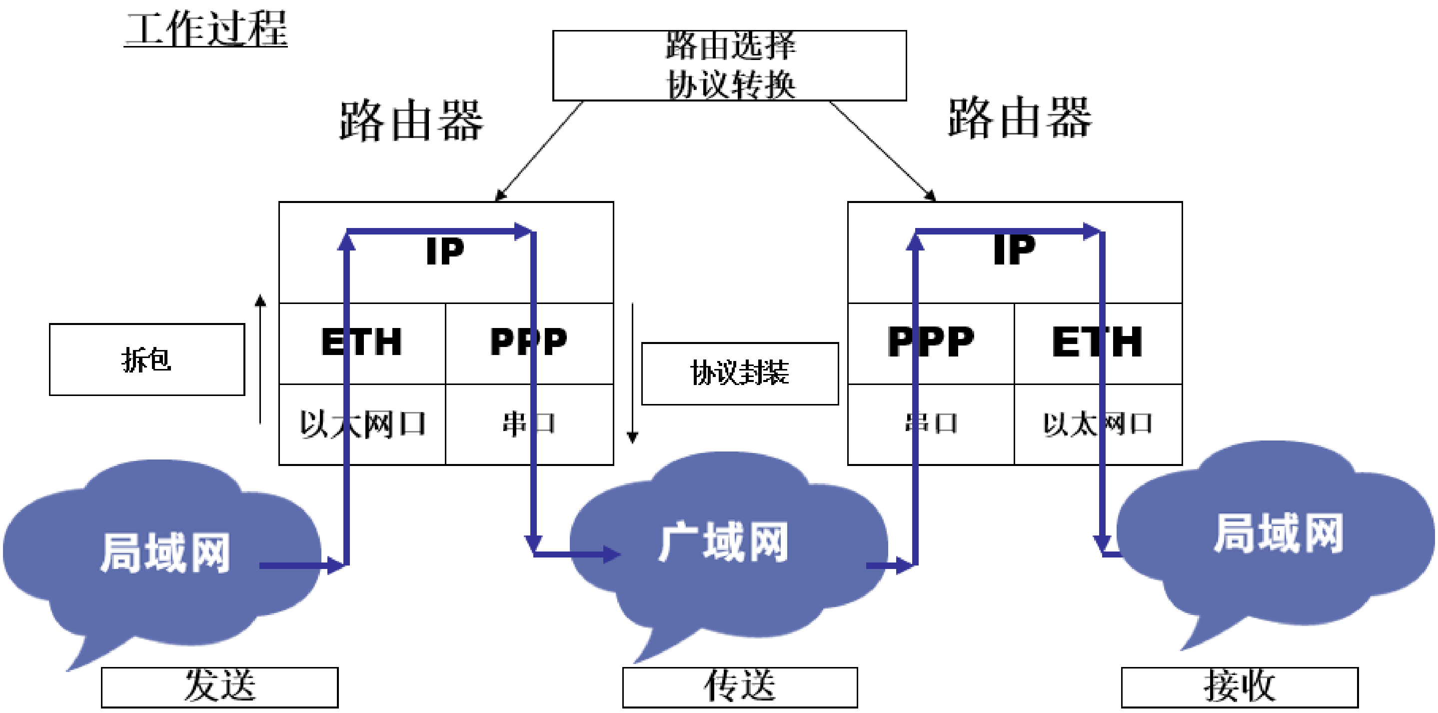
路由器的核心作用：实现网络互连

- 分组数据转发
- 路由（寻径）：路由表建立、刷新、查找
- 子网间的速率适配
- 隔离网络，防止网络风暴，制定访问规则（防火墙）
- 异种网络互连

路由器的接口

在路由器的背板上，有以太网口（Ethernet，用于连接局域网）和串口（Serial，用于连接广域网）。在本实验中，路由器的串口是通过 V.35 电缆进行连接的，而以太口连接在机柜的面板上，如 R1L0 即是指路由器 1 的 0 号以太口。

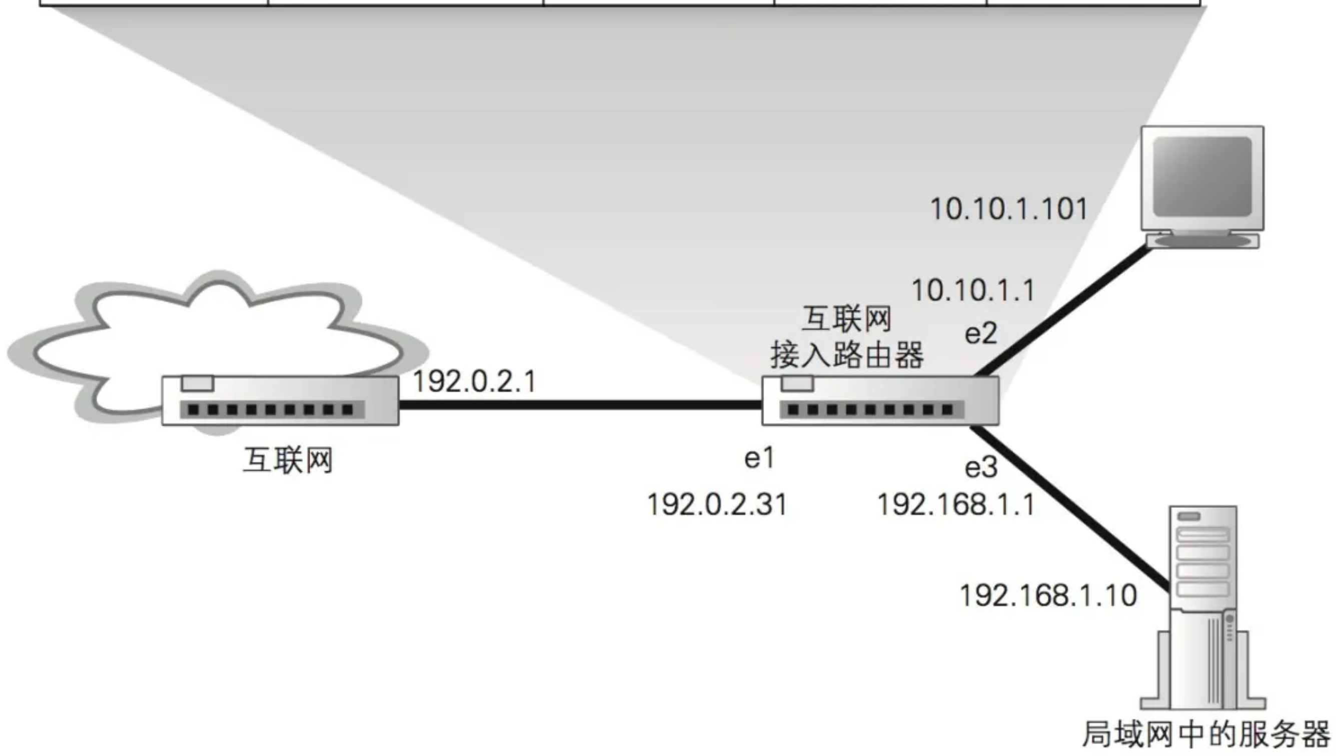
路由器工作流程



路由表信息

路由表记录了接收方 IP 地址与转发目标之间的关系：

- 第一列记录了接收方 IP 地址（只表示子网的网络号比特值，忽略主机号）。
- 第二列表示子网掩码，记录需要匹配网络号的比特数，路由器在匹配接收方 IP 过程中，通过子网掩码就知道需要匹配的网络号的比特数是多少，相当于我们生活中快递在转发过程中所经过的省，市，区，县，区域范围逐层减小。
- 第三列表示网关，网络包的转发目标。
- 第四列表示接口，在匹配到具体记录后，路由器会将网络包转发到网关（具体 IP 地址）的指定接口上（端口）。
- 第五列表示跃点数，表示距离目标 IP 地址的距离是远还是近，数字越小，表示距离目的地越近，数字越大，表示距离目的地越远。



目标地址（Destination）	子网掩码（Netmask）	网关（Gateway）	接口（Interface）	跃点数（Metric）
10.10.1.0	255.255.255.0	——	e2	1
10.10.1.101	255.255.255.255	——	e2	1
192.168.1.0	255.255.255.0	——	e3	1
192.168.1.10	255.255.255.255	——	e3	1
0.0.0.0	0.0.0.0	192.0.2.1	e1	1

有的路由表也可能包含：

- Destination/Mask：目标地址/子网掩码
- Proto：路由协议
- Pre：优先级
- Cost：开销
- Nexthop：下一跳
- Interface：接口

DCE 与 DTE

DCE（Data Circuit-terminating Equipment，数据通信设备或者数据电路终端设备）：该设备和其与通信网络的连接构成了网络终端的用户网络接口。它提供了到网络的一条物理连接、转发业务量，并且提供了一个用于同步 DCE 设备和 DTE 设备之间数据传输的时钟信号。调制解调器和接口卡都是 DCE 设备的例子。

DTE（Data Terminal Equipment，数据终端设备）：指的是位于用户网络接口用户端的设备，它能够作为信源、信宿或同时为二者。数据终端设备通过数据通信设备（例如，调制解调器）连接到一个数据网络上，并且通常使用数据通信设备产生的时钟信号。数据终端设备包括计算机、协议翻译器以及多路分解器等设备。

以拨号上网为例，计算机作为 DTE 设备，将调制解调器作为 DCE 设备。当计算机需要通过电话线连接到互联网时，它会将数据传输到调制解调器。该调制解调器会将数字信号转换为模拟信号以通过电话线传输。当数据到达目的地时，另一个调制解调器将模拟信号转换为数字信号，并将其传输给目标计算机的 DTE 设备。

> ### TIP
>
> 调制解调器是一种设备，它可以将数字信号转换为模拟信号并将其发送到电话线或数字信道中，并且可以将这些模拟信号接收并转换回数字信号以进行处理。在互联网的早期发展阶段，调制解调器是计算机连接到互联网的唯一方式。该设备支持计算机与网络之间的数据传输，以用于传输电子邮件、文本文件、图像、音频、视频等信息。现在，随着宽带网络的普及，调制解调器已经被更快、更便捷的网络连接方式所取代，如光纤、DSL（数字用户线，利用电话线传输数字信号，详见Chapter 2 @ DSL）、电缆调制解调器（有线电缆）和无线网络（Wi-Fi）等。

静态路由

静态路由（Static Routing）是在路由器中设置固定的路由表。由网络管理员管理路由表。由于静态路由不能对网络的改变作出反映，一般用于网络规模不大、拓扑结构固定的网络中。优点是简单、高效、可靠。在所有的路由中，静态路由优先级最高。当动态路由与静态路由发生冲突时，以静态路由为准。

命令：

```sh
ip route-static ip-address {mask|mask-length} [interface-type interface-number]
```

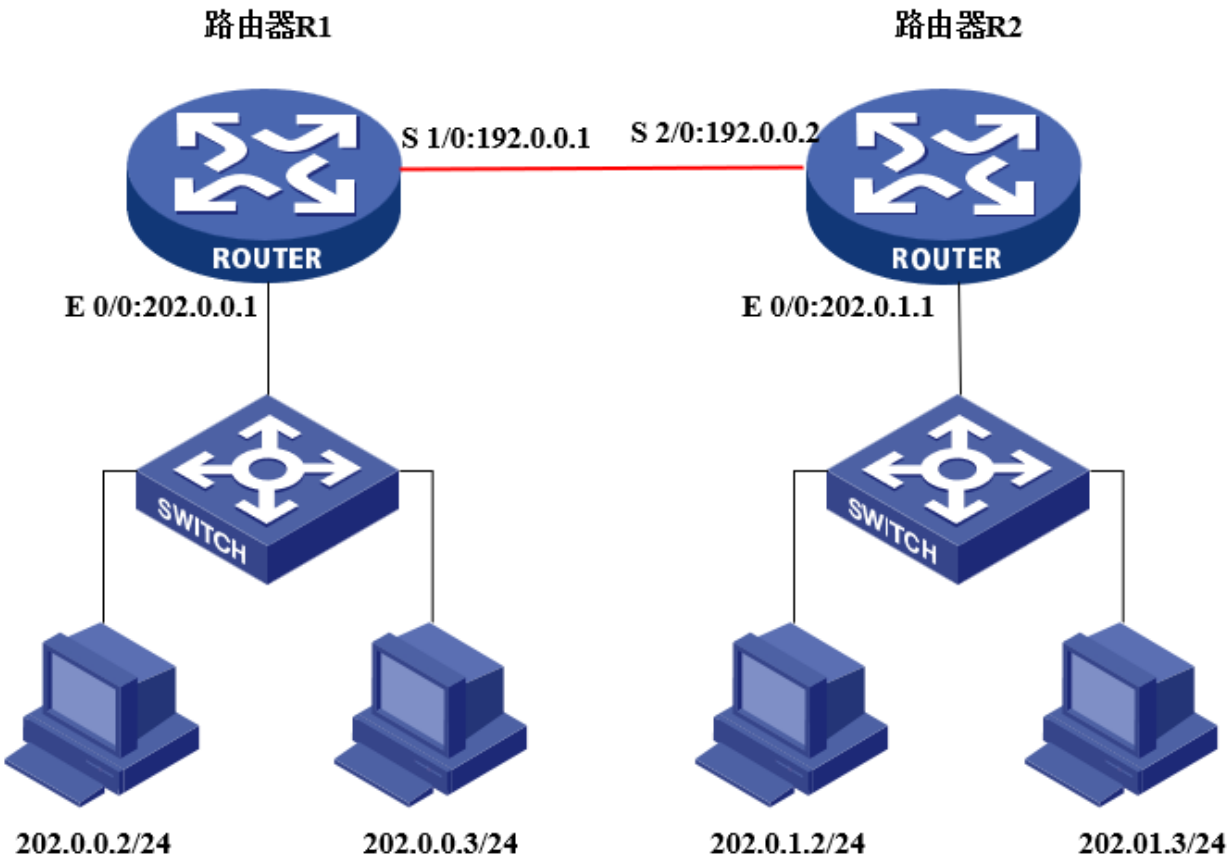
实验步骤

路由器接口与机柜上接口标签的对应关系：

路由器的以太网口连接上机柜上方的 RJ-45 接口，1 号路由器的 Ethernet 0/0 对应 R1L0，Ethernet 1/0 对应 R1L1，以此类推；

p 路由器的串口连接在机柜顶部的 V.35 线缆，1 号路由器的 Serial 1/0 口对应顶部的标签为 R1S1 的 DCE（或 DTE）接头，Serial 2/0 口对应顶部的标签为 R1S2 的 DCE（或 DTE）接头，以此类推。

1. 按图示搭建网络环境：



2. 设置 IP 地址

   路由器 R1 的设置：

   - 通过超级终端，进入路由器控制口。
   - `system-view` ：进入系统视图。
   - `sysname R1` ：重命名路由。
   - `interface Ethernet 0/0` ：进入以太网 0 口的接口视图
   - `ip address 202.0.0.1 24` ：设置 ip 地址
   - `interface serial 1/0` ：再进入串口的 1/0 ▯
   - `ip address 192.0.0.1 24` ：设置 ip 地址

   路由器 R2 的设置：

   进入系统视图后，进入以太网 0 口的接口视图，设置 ip 地址：

   - `system-view`
   - `sysname R2`
   - `interface Ethernet 0/0`

- `interface serial 2/0`
- `ip address 192.0.0.2 24`

3. 设置 PC 的 IP 地址，并将网关指向所连接的路由器的以太口。

设置完成后，可以在路由器上看到以太口和串口的状态转为 Up，出现类似下面的提示：

```text
%Oct 22 17:19:16:602 2007 RouteR1 IFNET/5/UPDOWN:PPP IPCP protocol on the inte
```

此时在 PC 上 ping 网关或本地路由器的串口，应该 ping 通（请将多余的本地连接禁用），如果 ping 不通，请检查交换机的配置，是否有 vlan 阻止了数据包的通过。

但在 pc 上 ping 对端路由器的 ip 地址或其以太网上的 pc，不能 ping 通，这是因为本地路由器上还没有对方网段的路由信息，数据包无法获知到达对端的路径。

4. 设置静态路由

R1 上的配置：`ip route-static 202.0.1.0 255.255.255.0 192.0.0.2`

R2 上的配置：`ip route-static 202.0.0.0 255.255.255.0 192.0.0.1`

5. 使用默认路由设置静态路由

> **TIP**
>
> 默认路由是静态路由的一种，当目标网络为 `0.0.0.0` 时，表示任何发往外网的数据包都将发往指定的下一跳地址。

R1 上的配置：`ip route-static 0.0.0.0 0.0.0.0 192.0.0.2`

R2 上的配置：`ip route-static 0.0.0.0 0.0.0.0 192.0.0.1`

建立了静态路由后，可以在路由器上用命令 `display ip routing-table` 查看路由表，检查路由器上是否有到达对端的路由项。也可以用 `display ip routing-table protocol static` 命令查看是否正确配置。

```text
[R1] display ip routing-table
Routing Table: public net
Destination/Mask Proto Pre Cost Nexthop Interface
1.1.1.0/24 DIRECT 0 0 1.1.1.1 Interface Serial1/0/0
1.1.1.1/32 DIRECT 0 0 127.0.0.1 InLoopBack0
2.2.2.0/24 DIRECT 0 0 2.2.2.1 Interface serial2/0/0
2.2.2.1/32 DIRECT 0 0 127.0.0.1 InLoopBack0
3.3.3.0/24 DIRECT 0 0 3.3.3.1 Interface ethernet1/0/0
3.3.3.1/32 DIRECT 0 0 127.0.0.1 InLoopBack0
4.4.4.0/24 DIRECT 0 0 4.4.4.1 Interface ethernet2/0/0
4.4.4.1/32 DIRECT 0 0 127.0.0.1 InLoopBack0
```

如果配置正确，应该有到达对方网段的路由信息。

6. 删除全部静态路由： `delete static-routes all`

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

5. 动态路由协议的配置

实验目的

掌握动态路由协议 RIP 和 OSPF 的配置方法。

实验原理

动态路由（Dynamic Routing）是网络中的路由器之间相互通信，传递路由信息，利用收到的路由信息动态更新路由器表的过程。它能实时地适应网络结构的变化。如果路由更新信息表明发生了网络变化，路由选择软件就会重新计算路由，并发出新的路由更新信息。这些信息通过各个网络，引起各路由器重新启动其路由算法，并更新各自的路由表以动态地反映网络拓扑变化。动态路由适用于网络规模大、网络拓扑复杂的网络。当然，动态路由协议会不同程度地占用网络带宽和 CPU 资源。

在设置动态路由之前，先用命令 `delete static-routes all` 将所有静态路由删除。

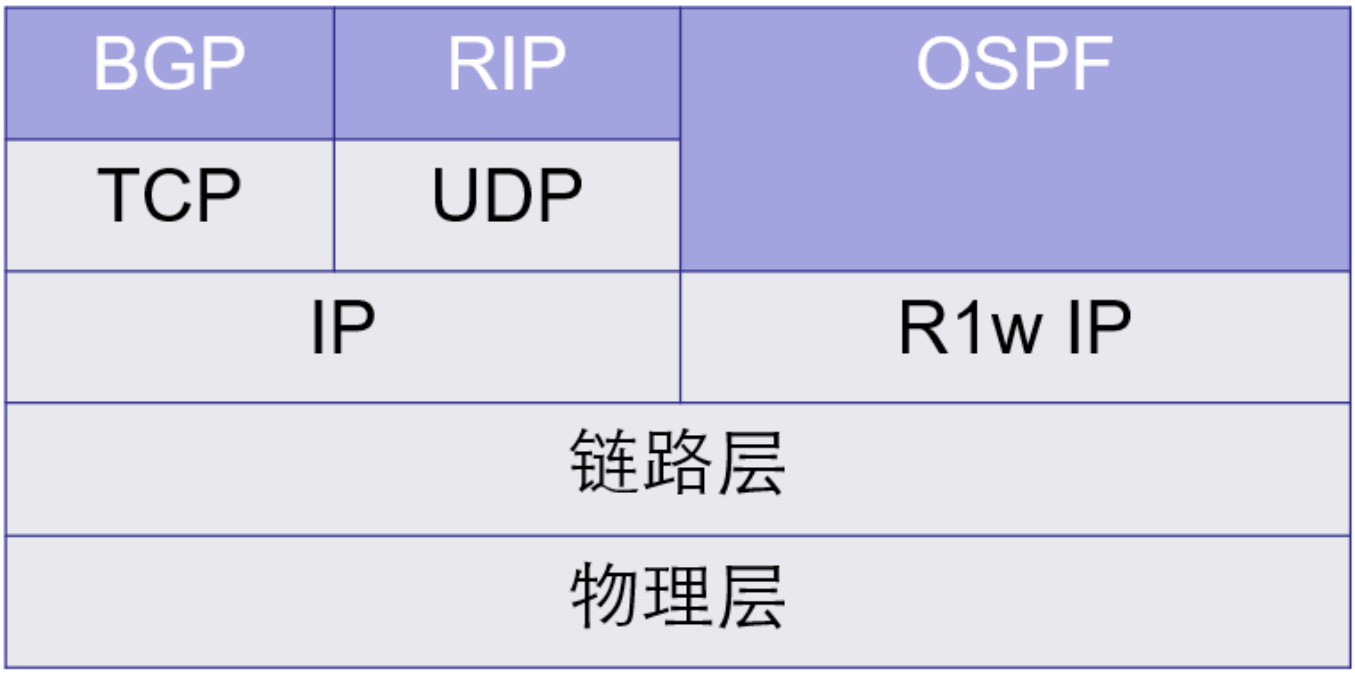
动态路由协议简述

> 动态路由协议概述 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/164747890)
>
> 动态路由协议，就是路由器能够自己建立路由表，不需要管理员手动一条一条将路由加表，这样就节省了许多人力。同时当网络中出现故障时，路由器可以自行检测链路，并选择最优的路径进行数据转发。
>
> 所有的动态路由协议在 **TCP/IP** 协议栈中都属于应用层的协议。但是不同的路由协议使用的底层协议不同。
>
> OSPF 将协议报文直接封装在 IP 报文中，协议号 89，由于 IP 协议本身是不可靠传输协议，所以**OSPF** 传输的可靠性需要协议本身来保证。
>
> **BGP 使用 TCP 作为传输协议，提高了协议的可靠性，TCP 的端口号是 179。RIP 使用 UDP 作为传输协议，端口号 520。**
>
> IS-IS 协议是开放系统互联（OSI）协议中的网络层协议，IS-IS 协议基础是 CLNP（Connectionless Network Protocol，无连接网络协议）。
>
> 所以，常见的动态路由有 RIP、OSPF、ISIS、BGP 等，其中 BGP 是唯一一个用于 AS 与 AS 之间的路由协议，其他三个都是用于 AS 内部的路由协议。
>
> 动态路由共同具有的特点：
>
> 1. 无需管理员手工维护，减轻了管理员的工作量。
> 2. 占用了网络带宽。
> 3. 在路由器上运行路由协议，使路由器可以自动根据网络拓扑结构的变化调整路由条目。
>
> 路由器协议性能指标：
>
> - 正确性：能够正确找出最优的路由，且无自环。
> - 快收敛：当网络的拓扑结构发生变换之后，能够迅速在自治系统中作相应的路由改变。
> - 低开销：协议自身的开销（内存、CPU、网络带宽）最小。
> - 安全性：协议自身不易受攻击，有安全机制。

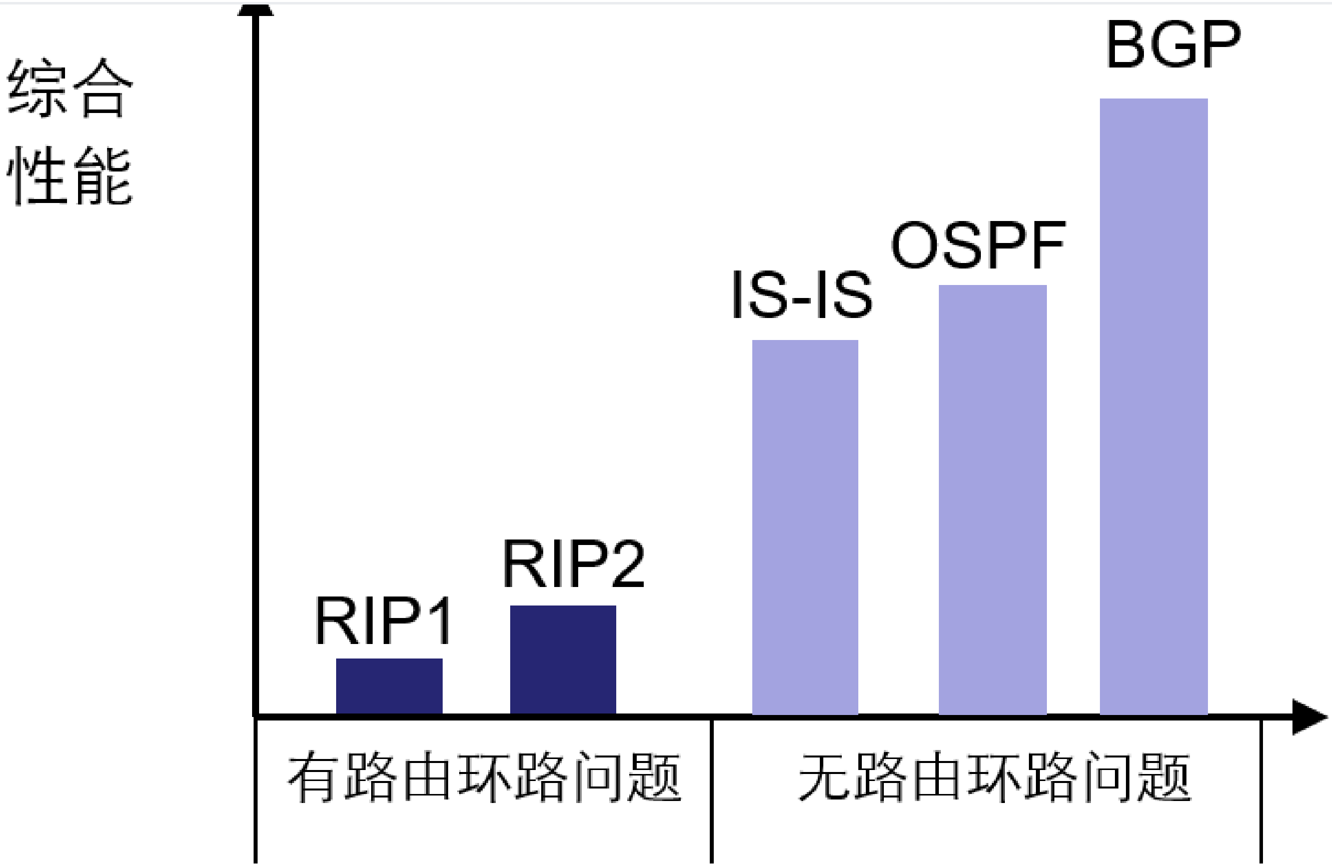
IS-IS、OSPF 和 BGP 的区别

- IS-IS（Intermediate System to Intermediate System）是一种链路状态型IGP协议，专为ISO的CLNS网络设计，IS-IS 协议最初是为了与 OSPF 竞争而设计的 IS-IS 用于 CLNS 网络，后来经过扩展支持 OSPF 用于 IP 网络。
- OSPF（Open Shortest Path First）是一种链路状态型IGP协议，专为IP网络设计，OSPF 使用 SPF 算法计算最短路径，并维护一个链路状态数据库。
- 边界网关协议（EGP）是一种用于在不同自治系统之间交换路由信息的协议，BGP 是一种用于在大型网络中交换路由信息的协议，BGP 通常用于互联网服务提供商。

协议栈的区别



协议的区别



RIP协议

RIP是Routing Information Protocol（路由信息协议）的简称，是距离矢量路由协议的一个具体实现。（如今RIP已经不怎么用了）

RIP协议适用于中小型网络，有RIP-1和RIP-2。其中RIP-2使用组播（224.0.0.9）发送，支持验证和VLSM。

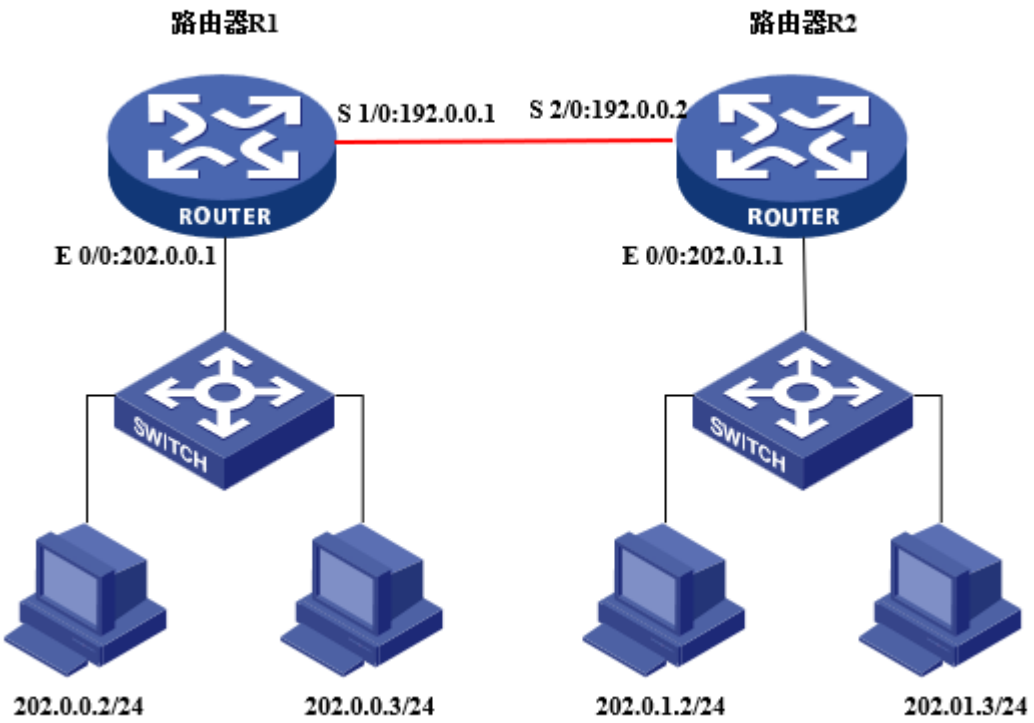
RIP支持：水平分割、路由中毒和触发更新。

RIP协议配置命令：

- `rip`：启动RIP协议，进入RIP协议配置视图
- `network network-number`：在指定的网络上使能RIP
- `peer IP-address`：配置报文的定点传送（不支持广播时）
- `rip version 1`：指定接口版本1（接口视图下）
- `rip version 2 [broadcast|multicast]`：指定接口版本2

实验步骤

STEP 1：搭建基本路由器网络



1、通过超级终端，进入路由器 R1 的控制口并进行设置：

- `system-view` ：进入系统视图
- `sysname R1` ：重命名路由器
- `interface Ethernet 0/0` ：进入以太网 0 口的接口视图
- `ip address 202.0.0.1 24` ：设置 ip 地址
- `interface serial 1/0` ：进入串口的 1/0
- `ip address 192.0.0.1 24` ：设置 ip 地址

2、路由器 R2 的设置：

进入系统视图后，进入以太网 0 口的接口视图，设置 ip 地址：

- `system-view` ：进入系统视图
- `sysname R2` ：重命名路由器
- `interface Ethernet 0/0` ：进入以太网 0 口的接口视图
- `ip address 202.0.1.1 24` ：设置 ip 地址
- `interface serial 2/0` ：进入串口的 2/0
- `ip address 192.0.0.2 24` ：设置 ip 地址

3、设置 PC 的 IP 地址，并将网关指向所连接的路由器的以太口。

设置完成后，可以在路由器上看到以太口和串口的状态转为 Up，出现类似下面的提示：

此时在PC上ping网关或本地路由器的串口，应该ping通（请将多余的本地连接禁用），如果ping不通，请检查交换机的配置，是否有vlan阻止了数据包的通过。

但在pc上ping对端路由器的ip地址或其以太网上的pc，不能ping通，这是因为本地路由器上还没有对方网段的路由信息，数据包无法获知到达对端的路径。

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

6. 访问控制列表及防火墙使用

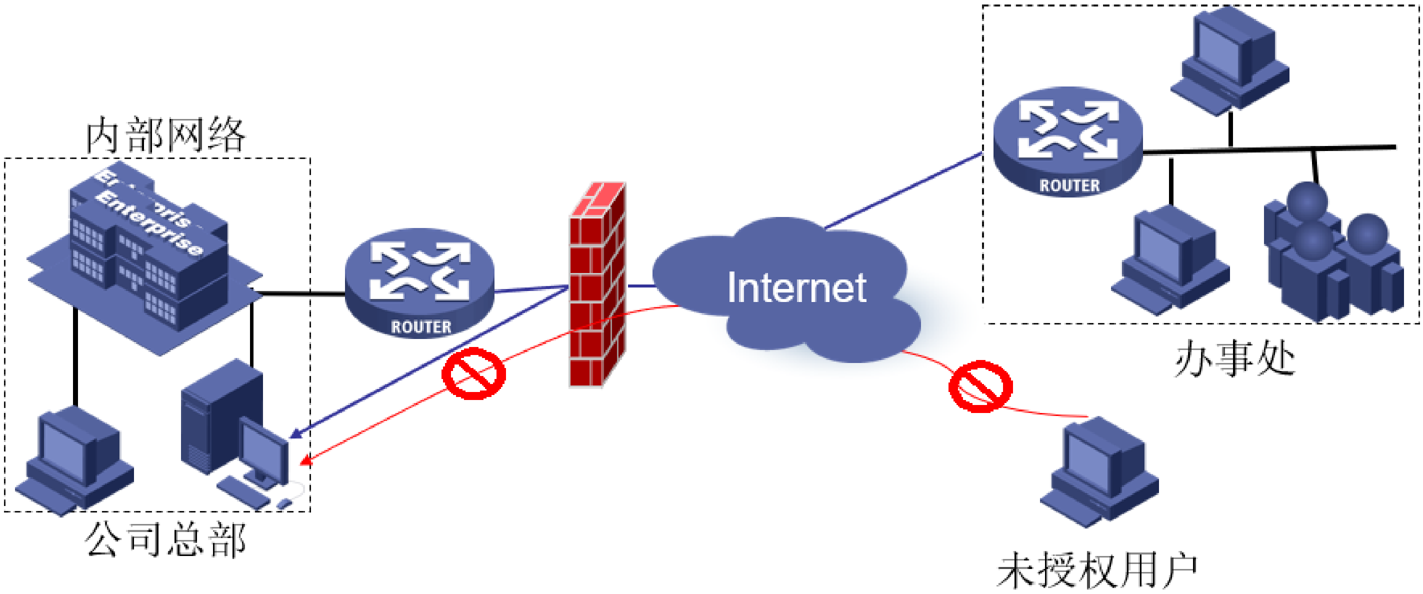
实验目的

学习访问控制列表 ACL 的配置方法，学习防火墙的基本使用方法。

实验原理

IP 包过滤技术

对路由器需要转发的数据包，先获取包头信息，然后和设定的规则进行比较，根据比较的结果对数据包进行转发或者丢弃。而实现包过滤的核心技术是访问控制列表。

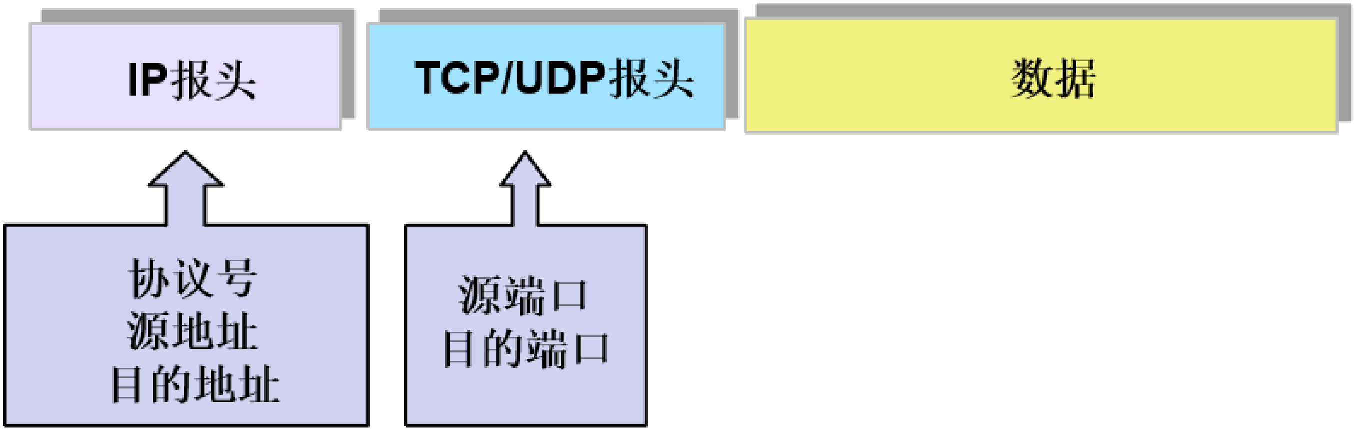


访问控制列表

访问控制列表的作用：

- 访问控制列表可以用于防火墙；
- 访问控制列表可以用于 QoS（Quality of Service），对数据流量进行控制；
- 在 DCC 中，访问控制列表还可用来规定触发拨号的条件；
- 访问控制列表还可以用于地址转换；
- 在配置路由策略时，可以利用访问控制列表来作路由信息的过滤。

访问控制列表的定义：



TIP

对于 TCP/UDP 来说，这 5 个元素组成了一个 TCP/UDP 相关，访问控制列表就是利用这些元素定义的规则

访问控制列表的分类：

按照访问控制列表的用途可以分为四类：

- 基本的访问控制列表（basic acl）
- 高级的访问控制列表（advanced acl）
- 基于接口的访问控制列表（interface-based acl）
- 基于 MAC 的访问控制列表（mac-based acl）

访问控制列表的标识：

- 利用数字标识访问控制列表
- 利用数字范围标识访问控制列表的种类

列表的种类	数字标识的范围
基于接口的访问控制列表	1000 ~ 1999
基本的访问控制列表	2000 ~ 2999
高级的访问控制列表	3000 ~ 3999
基于 MAC 地址访问控制列表	4000 ~ 4999

基本访问控制列表

基本访问控制列表只使用源地址描述数据，表明是允许还是拒绝。

配置基本访问列表的命令格式如下：

```
name] [logging] [fragment] [vpn-instance vpn-instanc-name]
```

反掩码和子网掩码相似，但写法不同：

- 0表示需要比较
- 1表示忽略比较

反掩码和IP地址结合使用，可以描述一个地址范围：

First quarter	Second quarter	Third quarter	Last quarter	Comment
0	0	0	255	只比较前24位
0	0	3	255	只比较前22位
0	255	255	255	只比较前8位

反掩码的实质：根据掩码，按位取反

如：

- 匹配192.168.0.0 /16网段，反掩码应该为0.0.255.255
- 匹配192.168.1.0 /26网段，反掩码应该为0.0.0.(255-192)=0.0.0.63
- 0.0.0.0 0.0.0.0匹配所有地址，等同于any
- 192.168.1.5 0.0.0.0匹配单独的ip地址
- 192.168.1.5 255.255.255.255根据反掩码，都不匹配，也等同于any

实例如：

```sh
acl number 2000
rule 1 deny source 192.110.10.0 0.0.0.255
rule 2 permit source 202.110.10.0 0.0.0.255
rule 3 permit source any
```

高级访问控制列表

高级访问控制列表使用除源地址外更多的信息描述数据包，表明是允许还是拒绝。

高级访问控制列表规则的配置命令：

sh

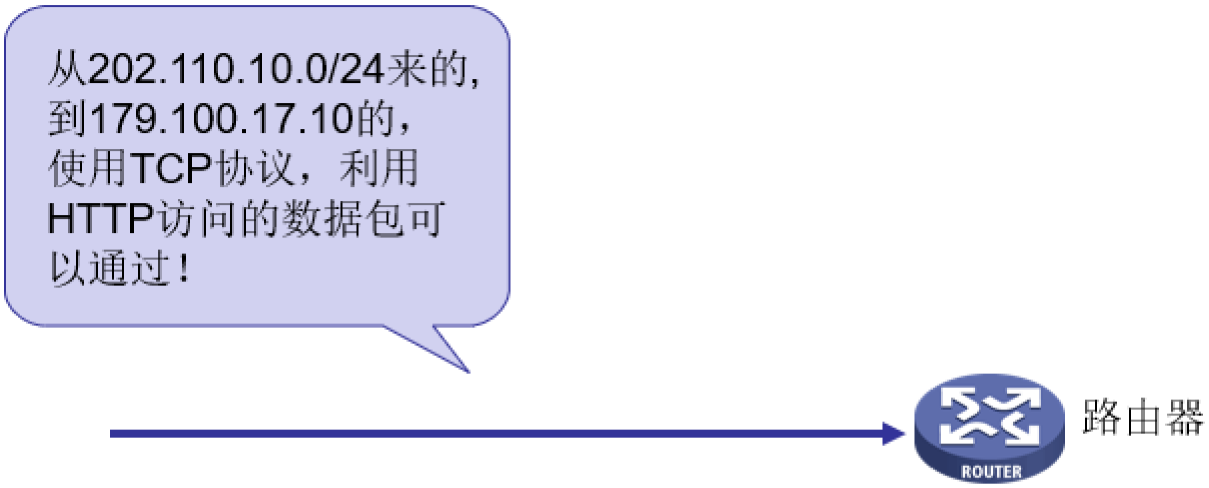
```sh
rule [rule-id] {permit|deny} protocol
   [source sour-addr sour-wildcard|any]
   [destination dest-addr dest-mask|any]
   [soucre-port operator port1 (port2)]
   [destination-port operator port1(port2)]
   [icmp-type {icmp-message|icmp-type icmp-code}]
   [precedence precedence] [tos tos] [time-range time-name]
   [logging] [fragment] [vpn-instance vpn-instanc-name]
```

其中：

操作符及语法	意义
eq portnumber	等于端口号 portnumber
gt portnumber	大于端口号 portnumber
lt portnumber	小于端口号 portnumber
neq portnumber	不等于端口号 portnumber
range portnumber1 portnumber2	介于端口号 portnumber1 和 portnumber2 之间

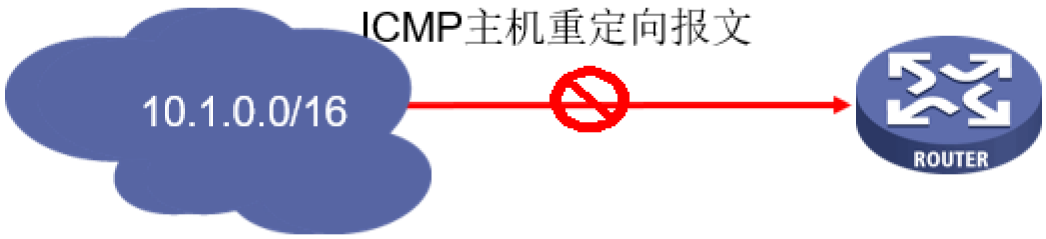
如：

```
3   rule 2 deny ip source any destionation any
```

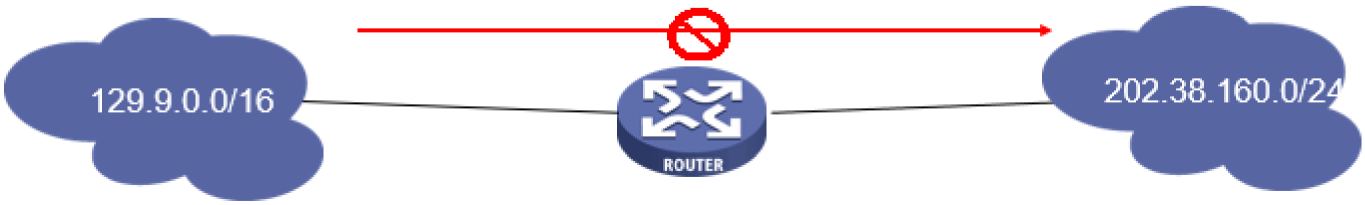


或是:

- ```sh
  1   rule deny icmp source 10.1.0.0 0.0.255.255 destination any icmp-type host-redi
  ```




- ```sh
  1   rule deny tcp source 129.9.0.0 0.0.255.255 destination 202.38.160.0 0.0.0.255
  ```



访问控制列表的显示

访问控制列表的显示与调试:

- display acl {all|acl-number}

访问控制列表的匹配规则

一条访问列表可以由多条规则组成，对于这些规则，有两种匹配顺序：auto 和 config。规则冲突时，若匹配顺序为 auto（深度优先），描述的地址范围越小的规则，将会优先考虑。

> 深度的判断要依靠通配比较位和 IP 地址结合比较：
>
> ```sh
> rule deny 202.38.0.0  0.0.255.255
> rule permit 202.38.160.0  0.0.0.255
> ```
>
> 两条规则结合则表示禁止一个大网段（202.38.0.0）上的主机但允许其中的一小部分主机（202.38.160.0）的访问。

规则冲突时，若匹配顺序为 config，先配置的规则会被优先考虑。

> **TIP**
>
> “深度优先”原则的具体标准如下：
>
> - 对于基本访问控制列表的规则，直接比较源地址通配符，通配符相同的按配置顺序；
> - 对基于接口过滤的访问控制规则，配置了 any 的规则排在后面，其他的按配置顺序；
> - 对于高级访问控制规则，首先比较源地址通配符，相同的再比较目的地址通配符，仍相同比较端口号范围，范围小的排在前面，如果都相同就按配置顺序。

访问控制列表的使用

防火墙配置常见步骤：

- 启用防火墙
- 定义访问控制列表
- 将访问控制列表应用到接口上

对应命令：

- 打开或者关闭防火墙：`firewall {enable|disable}`
- 设置防火墙的缺省过滤模式：`firewall default {permit|deny}`
- 显示防火墙的统计信息：

  `display firewall-statistics {all|interface interface-name|fragments-inspect}`
- 打开防火墙包过滤调试信息开关：

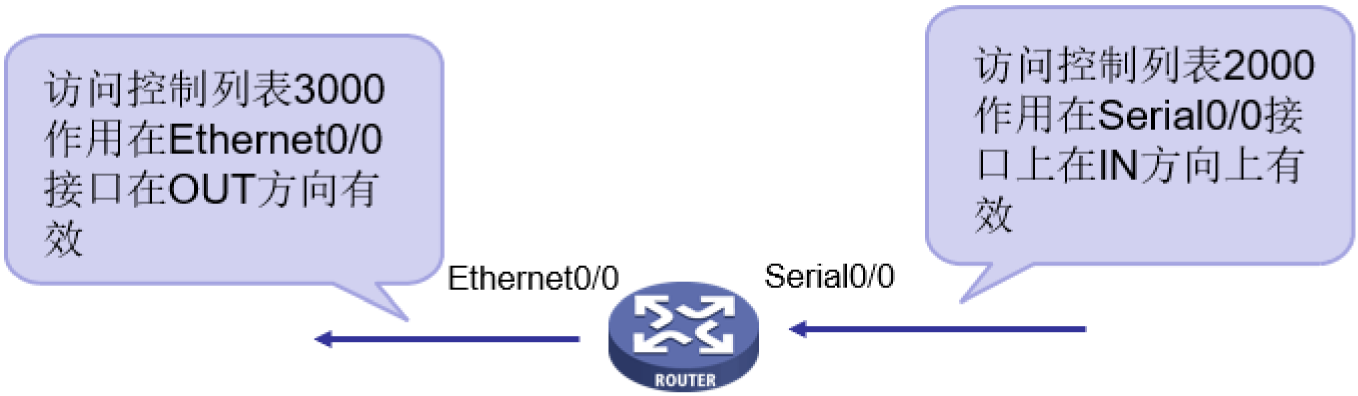
  `debugging firewall {all|icmp|tcp|udp|others} [interface interface-name]`

- □□□□□□□□□□□□□□

- 指明在接口上是 OUT 还是 IN 方向

- 在接口视图下配置:

```
 firewall packet-filter acl-number {inbound|outbound} [match-fragments
{normally|exactly}]
```

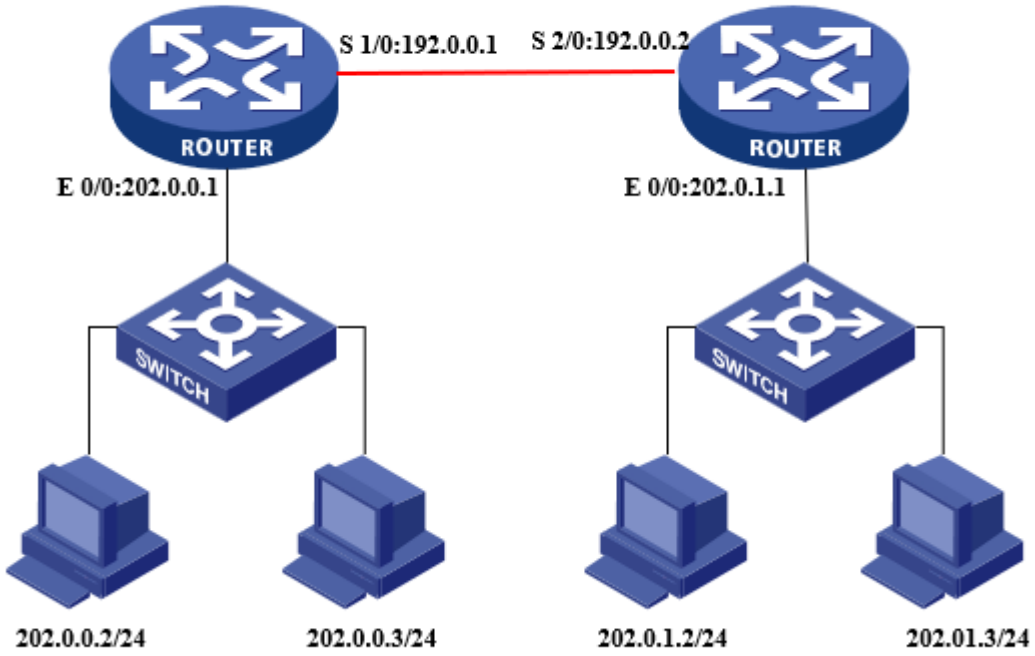


如:

```sh
acl num 3000
rule deny icmp source 202.110.10.1 0.0.0.255 destination 179.100.17.10 0.0.0.0 i
firewall enable
interface e 0
firewall packet-filter 3000 inbound
interface s 0
firewall packet-filter 3000 outbound
```

实验步骤

按图示搭建网络环境:



 R1

sh

```
acl num 3000
rule deny ip source 202.0.0.0 0.0.0.255 destination 202.0.1.0 0.0.0.255
firewall enable
interface e 0
firewall packet-filter 3000 inbound
interface s 0
firewall packet-filter 3000 outbound
```

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1

7. 应用服务器的配置

实验目的

- 学习和了解 WWW 的工作原理;
- 学习和了解域名服务的工作原理;
- 掌握配置和应用 Web 服务方法和过程;
- 掌握配置和应用 DNS 的方法和过程。

实验内容

- 配置 Web 服务，建立一测试网页，用浏览器测试是否能打开测试网页;
- 安装配置 DNS 服务器，并在其他 PC 上测试所配置的 DNS 是否可用。

WEB 服务的安装和配置

1. 启动 Windows Server 2008 虚拟机

- 点击桌面“Oracle VM VirtualBox”图标，启动虚拟系统。

AR_Base (AR_Base_Link)
已关闭
WLAN_AC_Base (WLAN_AC_Base...)
已关闭
WLAN_AP_Base (WLAN_AP_Base...)
已关闭
CentOS65
已关闭
Window 2008
已关闭
Ubuntu16
已关闭

常规

名称: Ubuntu16
操作系统: Ubuntu (64-bit)

系统

内存大小: 1324 MB
启动顺序: 软驱, 光驱, 硬盘
硬件加速: VT-x/AMD-V, 嵌套分页

显示

显存大小: 16 MB
远程桌面服务器端口: 3389
录像: 已禁用

存储

控制器: IDE
第二IDE控制器主通道: [光驱] 没有盘片
控制器: SATA
SATA 端口 0: Ubuntu16.vdi (普通, 10.00 GB)

声音

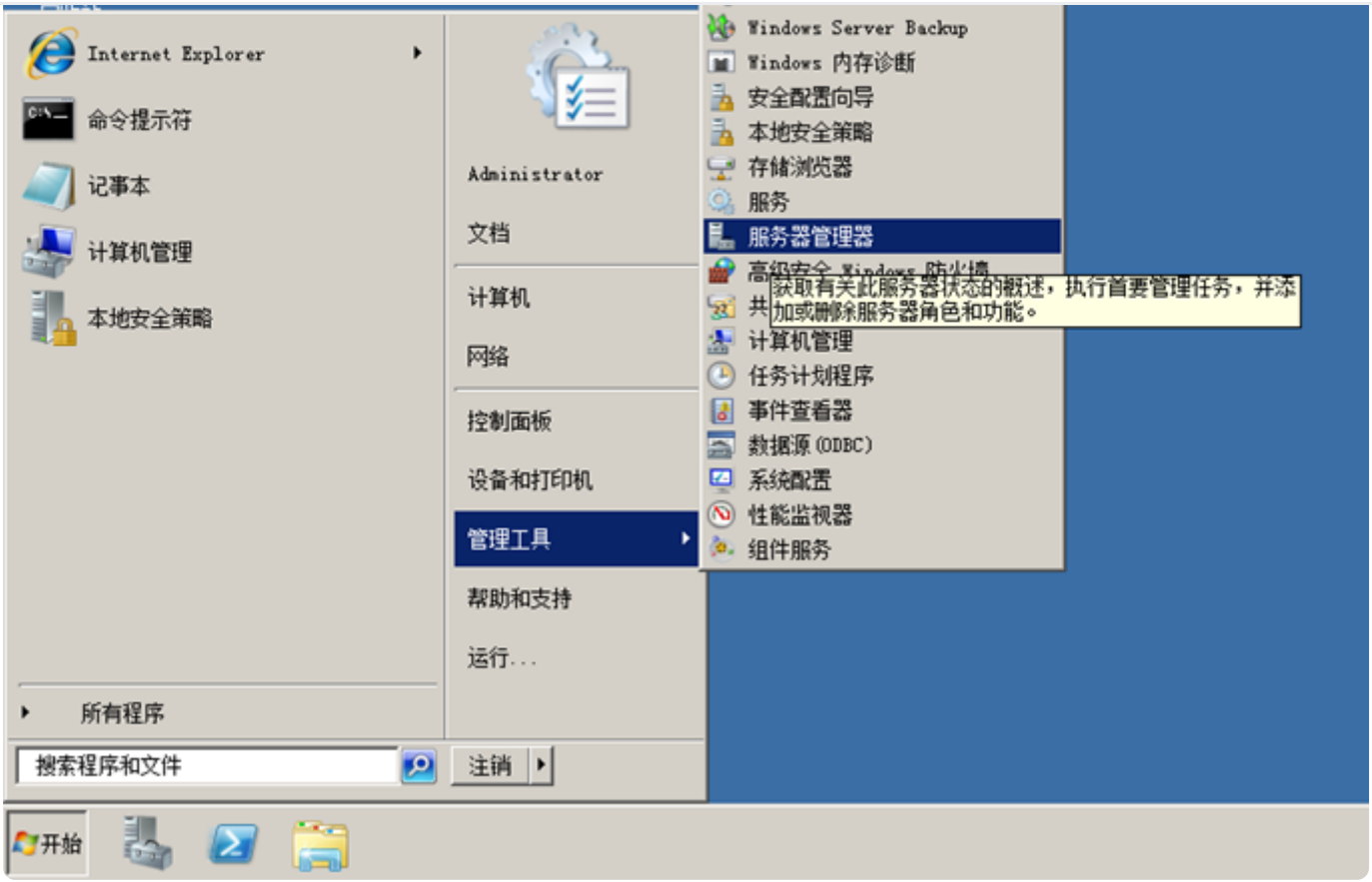
主机音频驱动: Windows DirectSound
控制芯片: ICH AC97

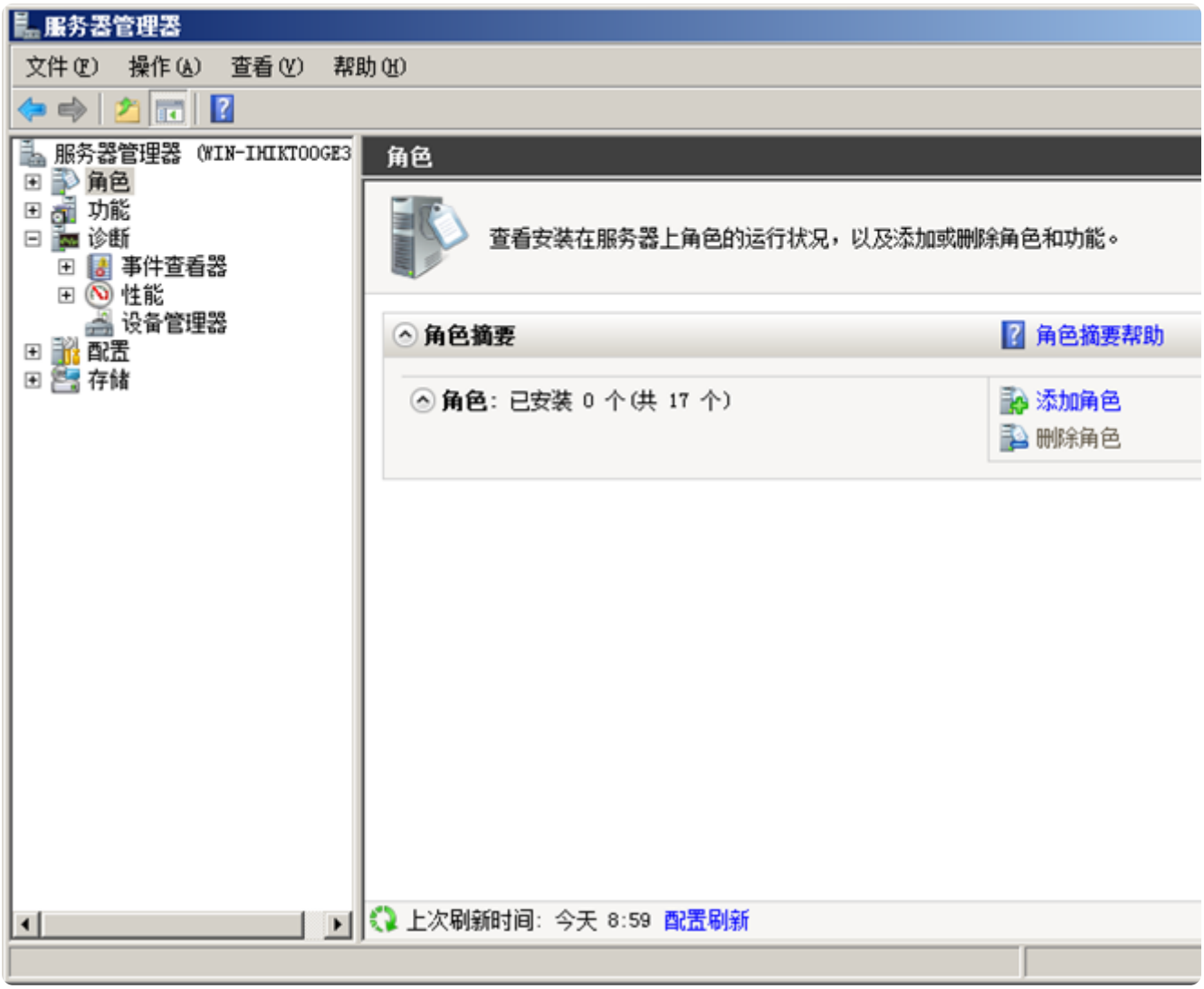
- 选择“windows 2008”，并点击绿色“启动”按钮，进入 winodws 2008 登录界面。
- 按右侧 `Ctrl + Alt + Delete` ，登录密码: network

2. 安装WEB服务器（IIS）

> **TIP**
>
> 在Windows Server 2008中，WEB服务被捆绑在IIS 7组件中。默认情况下，IIS 7组件并没有被安装，相应的WEB服务组件也没有被安装，因此需要手动安装WEB服务组件。

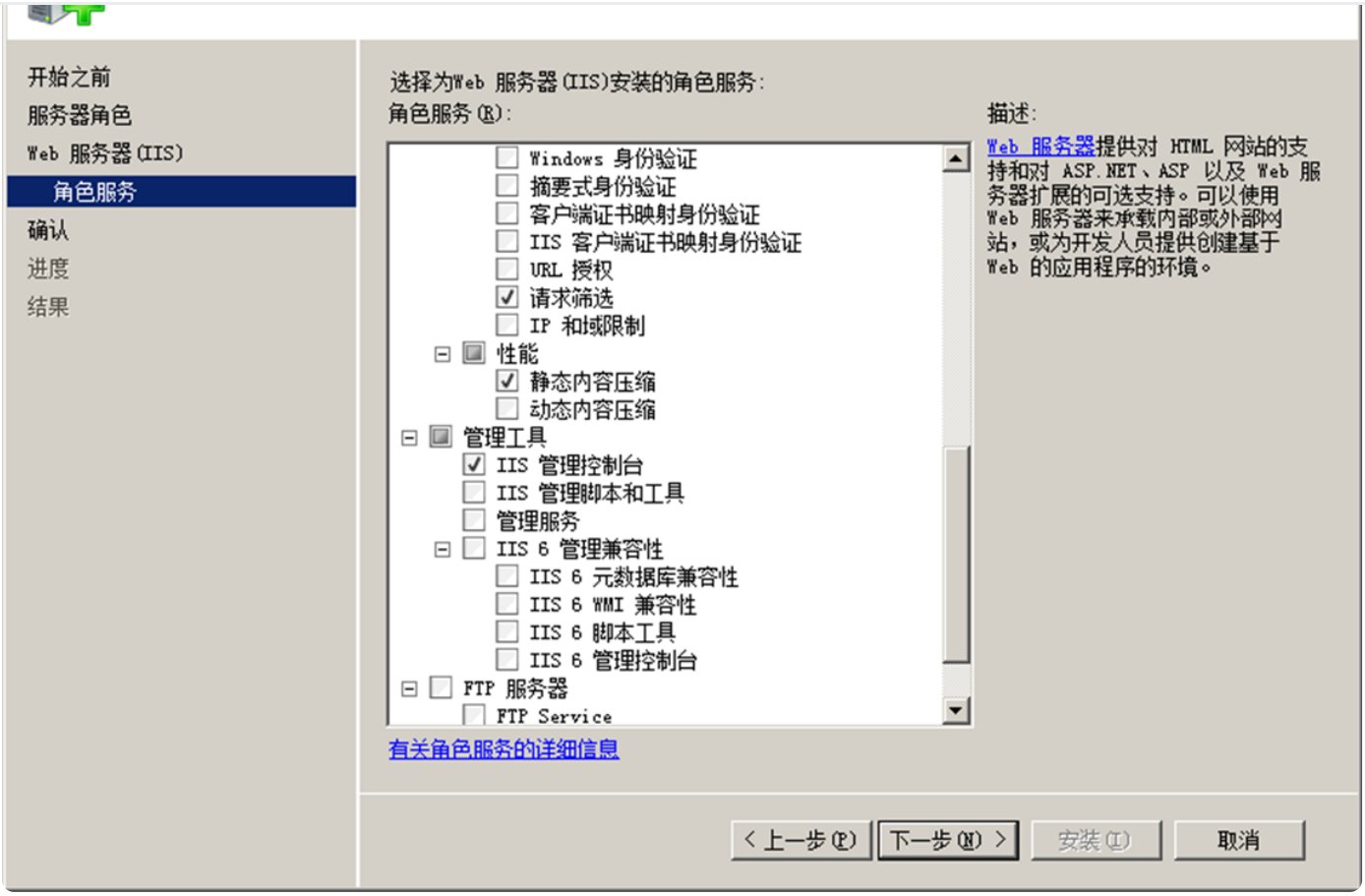
- 前往 开始 → 管理工具 → 服务器管理器；
- 在服务器管理器内选择 角色 → 添加角色 → Web服务器（IIS）进行安装。


Internet Explorer
命令提示符
记事本
计算机管理
本地安全策略
所有程序
搜索程序和文件
注销
Administrator
文档
计算机
网络
控制面板
设备和打印机
管理工具
帮助和支持
运行...
Windows Server Backup
Windows 内存诊断
安全配置向导
本地安全策略
存储浏览器
服务
服务器管理器
计算机管理
任务计划程序
事件查看器
数据源 (ODBC)
系统配置
性能监视器
组件服务
获取有关此服务器状态的概述，执行首要管理任务，并添加或删除服务器角色和功能。
开始

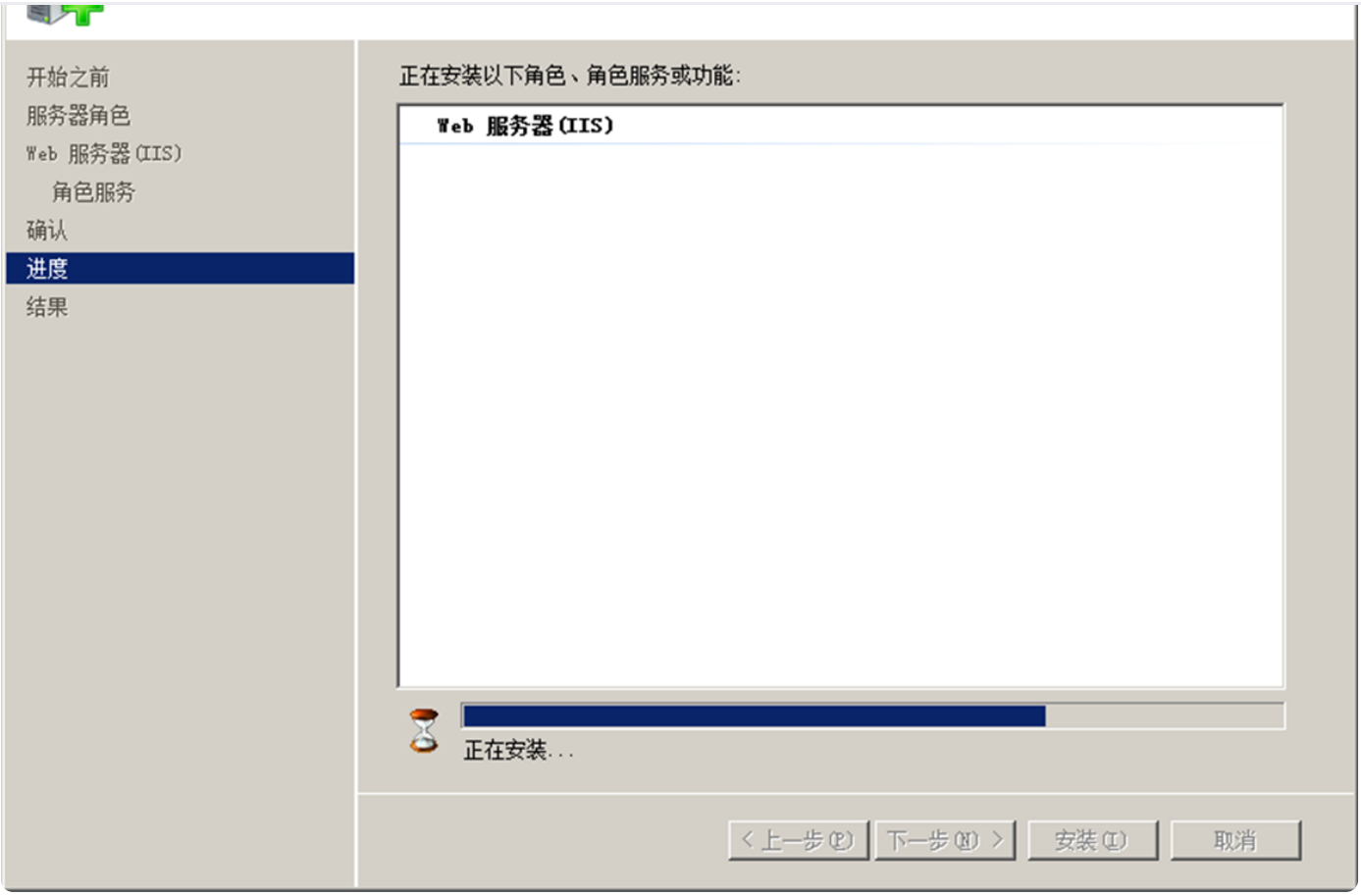

服务器管理器
文件(F) 操作(A) 查看(V) 帮助(H)
服务器管理器 (WIN-IHIKTOOGE3
角色
功能
诊断
事件查看器
性能
设备管理器
配置
存储
角色
查看安装在服务器上角色的运行状况，以及添加或删除角色和功能。
角色摘要
角色摘要帮助
角色：已安装 0 个(共 17 个)
添加角色
删除角色
上次刷新时间：今天 8:59 配置刷新


开始之前
服务器角色
Web 服务器(IIS)
角色服务
确认
进度
结果
选择要安装在此服务器上的一个或多个角色。
角色(R):
Active Directory Rights Management Services
Active Directory 联合身份验证服务
Active Directory 轻型目录服务
Active Directory 域服务
Active Directory 证书服务
DHCP 服务器
DNS 服务器
Hyper-V
Web 服务器(IIS)
Windows Server Update Services
Windows 部署服务
传真服务器
打印和文件服务
网络策略和访问服务
文件服务
应用程序服务器
远程桌面服务
描述:
Web 服务器(IIS)提供可靠、可管理并且可扩展的 Web 应用程序基础结构。
有关服务器角色的详细信息
< 上一步(P)
下一步(N) >
安装(I)
取消


添加角色向导
Web 服务器(IIS)
开始之前
服务器角色
Web 服务器(IIS)
角色服务
确认
进度
结果
Web 服务器 (IIS) 简介
Web 服务器是具有特定软件的计算机，这些软件使服务器可以从客户端计算机接受请求并对这些请求返回响应。Web 服务器支持在 Internet、Intranet 或 Extranet 上共享信息。Web 服务器角色包括 Internet 信息服务(IIS) 7.0，是一个集成了 IIS 7.0、ASP.NET 和 Windows Communication Foundation 的统一 Web 平台。IIS 7.0 还具有安全性增强、诊断简单和委派管理的特点。
注意事项
使用 Windows 系统资源管理器(WSRM)有助于确保等同对待 Web 服务器通信，特别是此计算机上存在多个角色时。
Web 服务器(IIS)角色的默认安装包括角色服务的安装，该安装使您能够使用静态内容、进行最小自定义(如默认的文档和 HTTP 错误)、监视和记录服务器活动以及配置静态内容压缩。
其他信息
Web 服务器(IIS)概述
IIS 7.0 中的可用角色服务概述
IIS 清单
IIS 中的常见管理任务
WSRM 概述
< 上一步(P)
下一步(N) >
安装(I)
取消

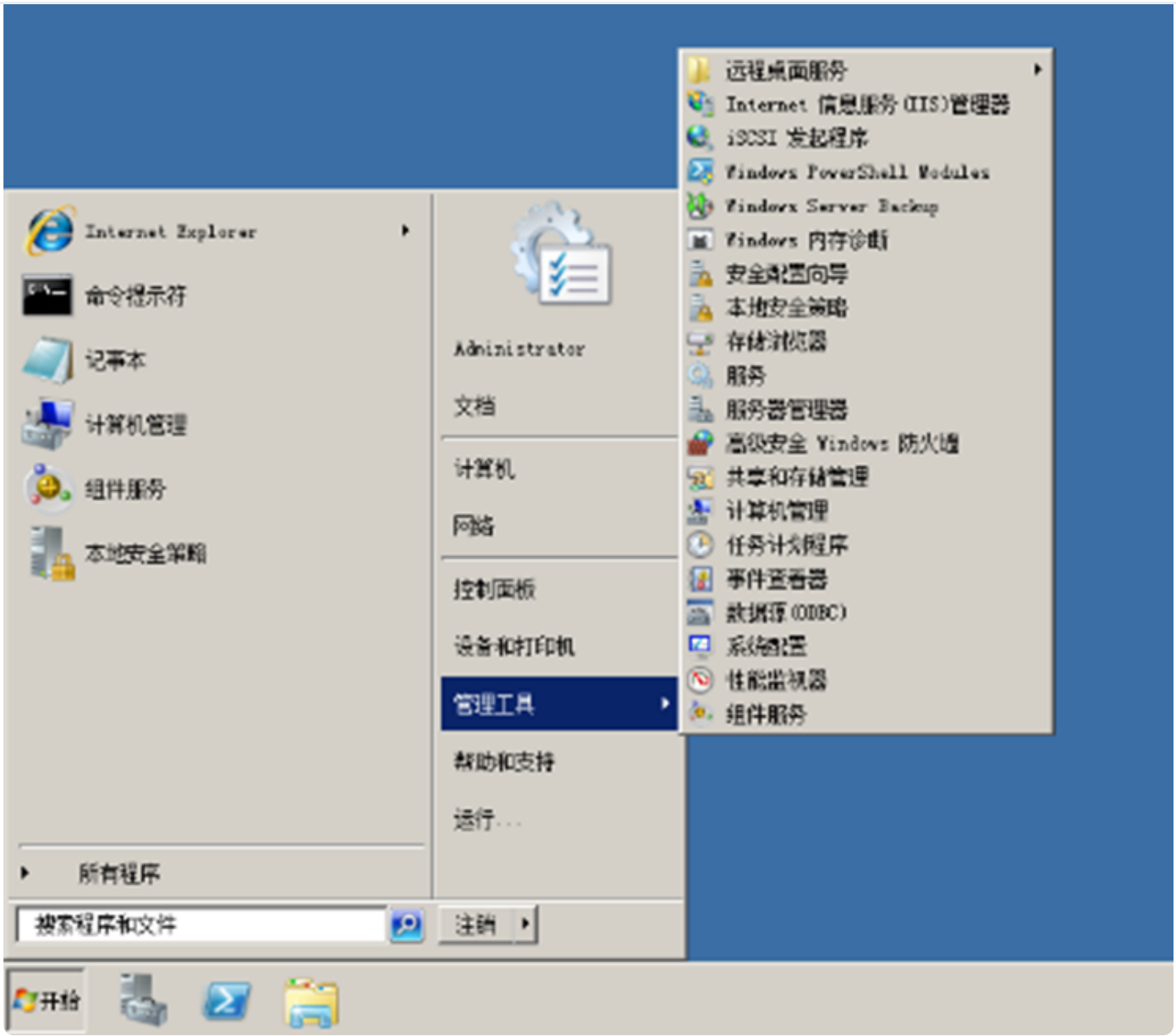

开始之前
服务器角色
Web 服务器(IIS)
角色服务
确认
进度
结果
选择为Web 服务器(IIS)安装的角色服务:
角色服务(R):
Windows 身份验证
摘要式身份验证
客户端证书映射身份验证
IIS 客户端证书映射身份验证
URL 授权
请求筛选
IP 和域限制
性能
静态内容压缩
动态内容压缩
管理工具
IIS 管理控制台
IIS 管理脚本和工具
管理服务
IIS 6 管理兼容性
IIS 6 元数据库兼容性
IIS 6 WMI 兼容性
IIS 6 脚本工具
IIS 6 管理控制台
FTP 服务器
FTP Service
有关角色服务的详细信息
描述:
Web 服务器提供对 HTML 网站的支持和对 ASP.NET、ASP 以及 Web 服务器扩展的可选支持。可以使用 Web 服务器来承载内部或外部网站，或为开发人员提供创建基于 Web 的应用程序的环境。
< 上一步(P)
下一步(N) >
安装(I)
取消


添加角色向导
确认安装选择
开始之前
服务器角色
Web 服务器(IIS)
角色服务
确认
进度
结果
若要安装以下角色、角色服务或功能，请单击“安装”。
2 条提示性消息显示如下
安装完成之后，可能需要重新启动该服务器。
Web 服务器(IIS)
查找有关 Windows 系统资源管理器(WSRM)以及它如何帮助优化 CPU 使用情况的详细信息
Web 服务器
常见 HTTP 功能
静态内容
默认文档
目录浏览
HTTP 错误
健康和诊断
HTTP 日志记录
请求监视
安全性
请求筛选
性能
静态内容压缩
管理工具
IIS 管理控制台
打印、保存或通过电子邮件发送此信息
< 上一步(P)
下一步(N) >
安装(I)
取消


开始之前
服务器角色
Web 服务器(IIS)
角色服务
确认
进度
结果
正在安装以下角色、角色服务或功能:
Web 服务器(IIS)
正在安装...
< 上一步(P)
下一步(N) >
安装(I)
取消


添加角色向导
安装结果
开始之前
服务器角色
Web 服务器(IIS)
角色服务
确认
进度
结果
已成功安装以下角色、角色服务或功能:
1 条警告消息显示如下
未启用 Windows 自动更新。为确保自动更新最新安装的角色或功能，请启用“控制面板”的 Windows Update。
Web 服务器(IIS)
安装成功
已安装以下角色服务:
Web 服务器
常见 HTTP 功能
静态内容
默认文档
目录浏览
HTTP 错误
健康和诊断
HTTP 日志记录
请求监视
安全性
请求筛选
性能
静态内容压缩
管理工具
IIS 管理控制台
打印、保存或通过电子邮件发送安装报告
< 上一步(P)
下一步(N) >
关闭(O)
取消

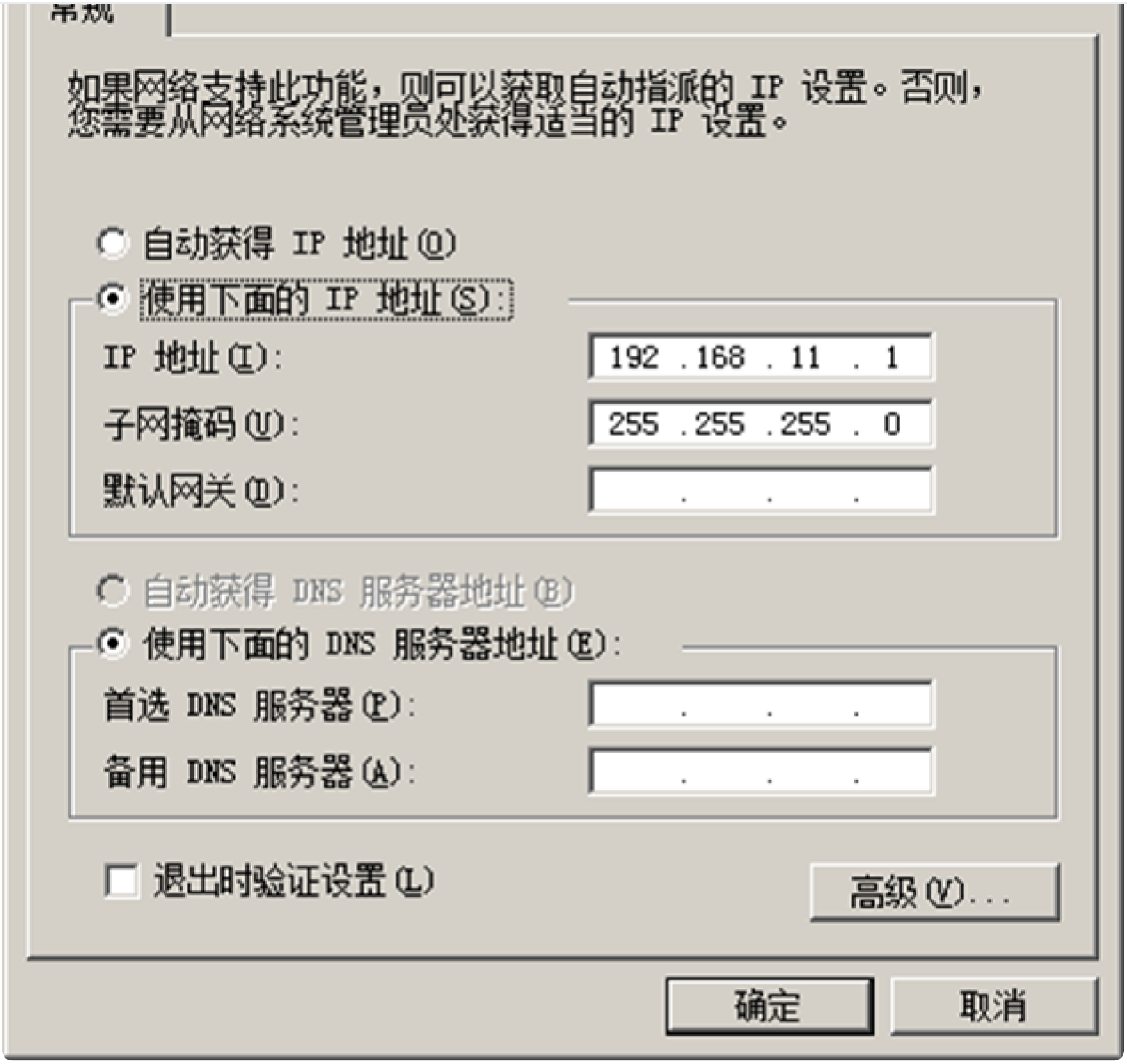
3. 在WEB中


Internet Explorer
命令提示符
记事本
计算机管理
组件服务
本地安全策略
所有程序
搜索程序和文件
Administrator
文档
计算机
网络
控制面板
设备和打印机
管理工具
帮助和支持
运行...
注销
开始
远程桌面服务
Internet 信息服务(IIS)管理器
iSCSI 发起程序
Windows PowerShell Modules
Windows Server Backup
Windows 内存诊断
安全配置向导
本地安全策略
存储浏览器
服务
服务器管理器
高级安全 Windows 防火墙
共享和存储管理
计算机管理
任务计划程序
事件查看器
数据源(ODBC)
系统配置
性能监视器
组件服务


Internet 信息服务(IIS)管理器
WIN-IMIKTOOGE3N ▸ 网站 ▸ Default Web Site ▸
文件(F) 视图(V) 帮助(H)
连接
起始页
WIN-IMIKTOOGE3N (WIN-IMIKT
应用程序池
网站
Default Web Site
Default Web Site 主页
筛选:
开始(G) 全部显示(A)
IIS
HTTP 响应标头
MIME 类型
SSL 设置
处理程序映射
错误页
模块
默认文档
目录浏览
请求筛选
日志
身份验证
输出缓存
压缩
管理
配置编辑器
功能视图
内容视图
操作
浏览
编辑权限...
编辑网站
绑定...
基本设置...
查看应用程序
查看虚拟目录
管理网站
重新启动
启动
停止
浏览网站
浏览 *:80 (http)
高级设置...
配置
限制...
帮助
联机帮助
就绪

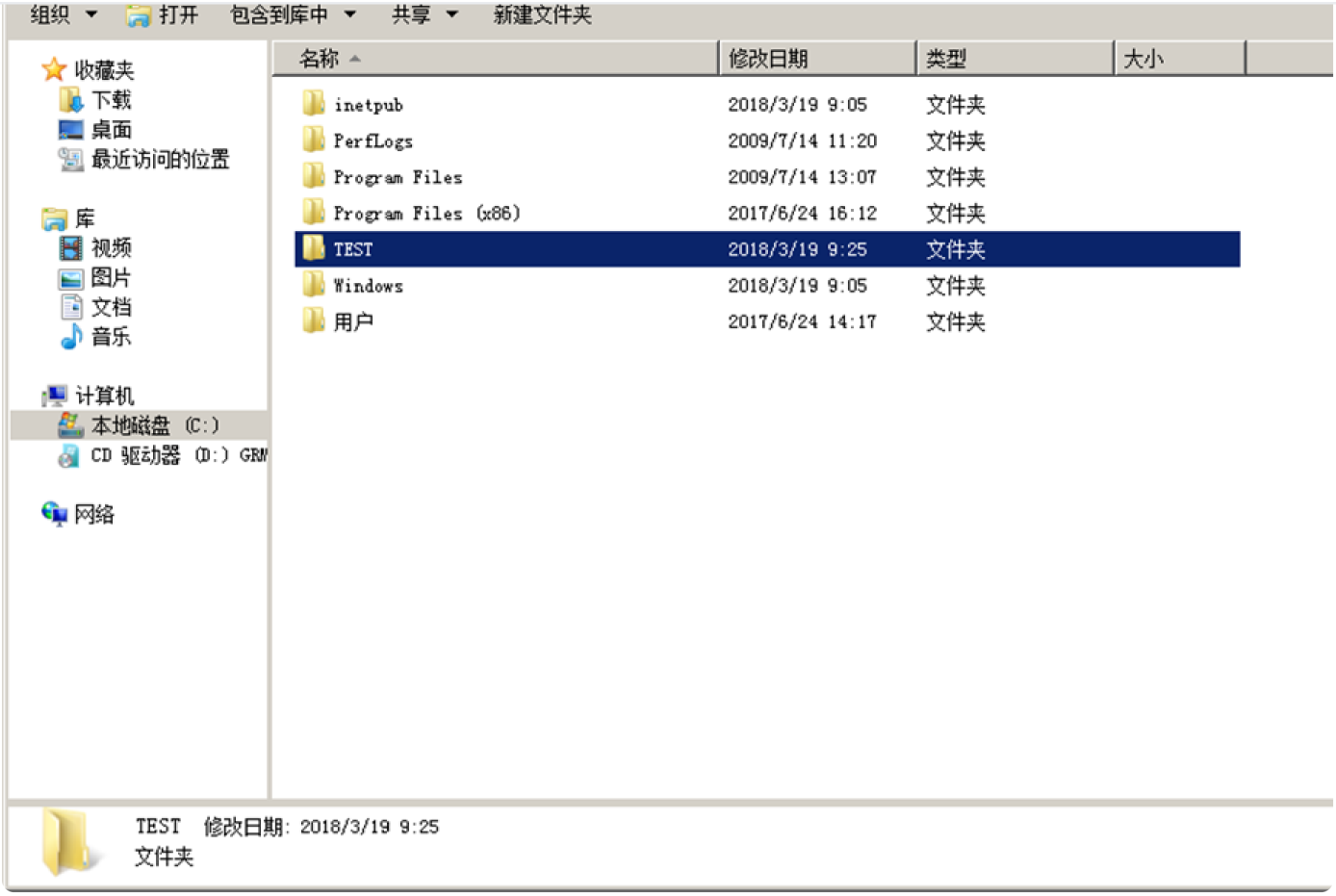

http://localhost/
Bing
收藏夹
建议网站
网页快讯库
IIS7
页面(P)
安全(S)
工具(O)
Welcome
ようこそ
Bienvenido
Willkommen
Bem-vindo
Vítejte
Bienvenue
Tervetuloa
歡迎
ברוכים הבאים
Velkommen
VELKOMEN
Benvenuto
欢迎
Welkom
Witamy
Välkommen
مرحباً
Hoş Geldiniz
환영합니다
Üdvözöljük
Καλώς ορίσατε
Добро пожаловать
IIS7
internet information services
完成
本地 Intranet | 保护模式: 禁用
100%

- □□□□□“IP□□□□□□□”

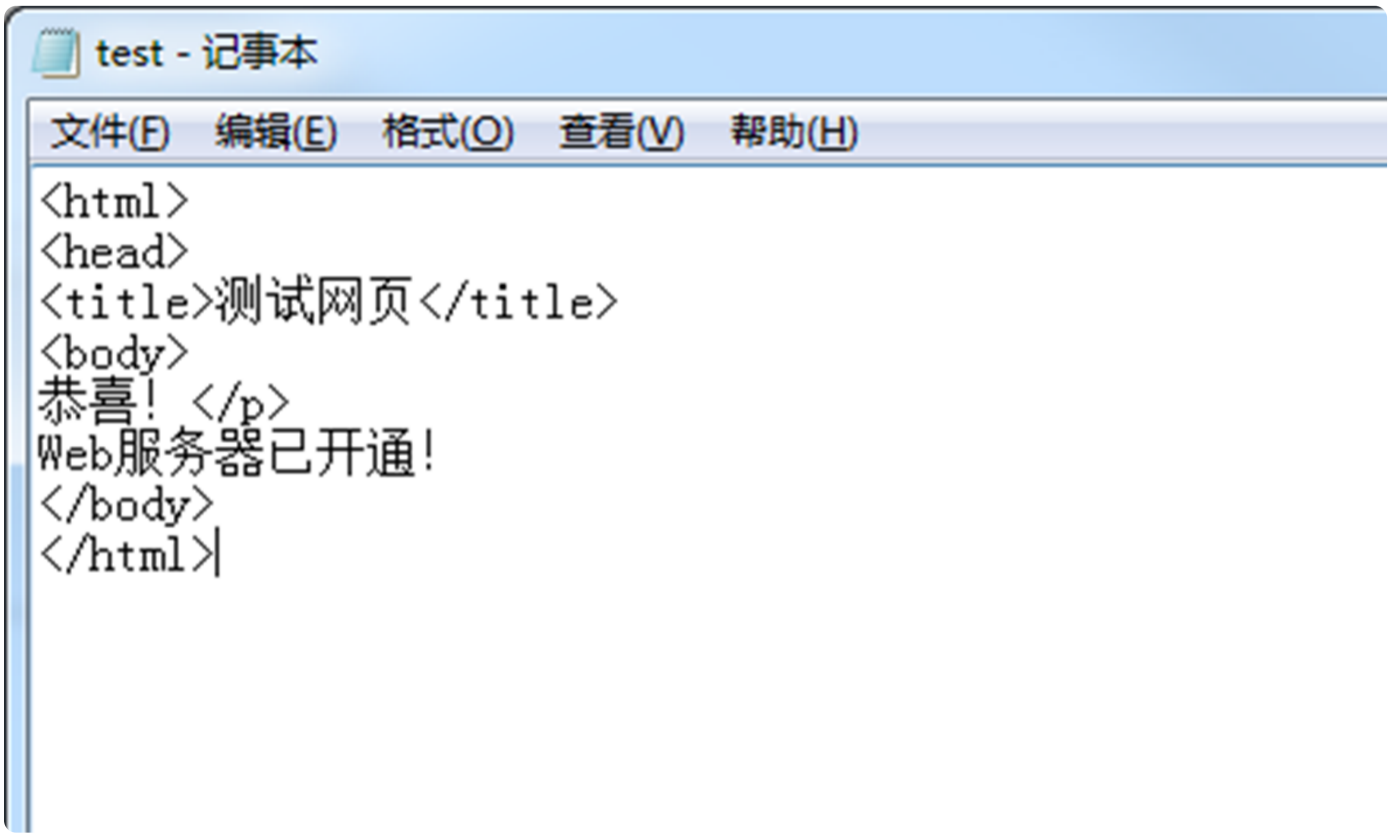


- 新建（或事先准备）一个web网页的文件夹: TEST

  建议就建在inetpub/wwwroot 目录下面。



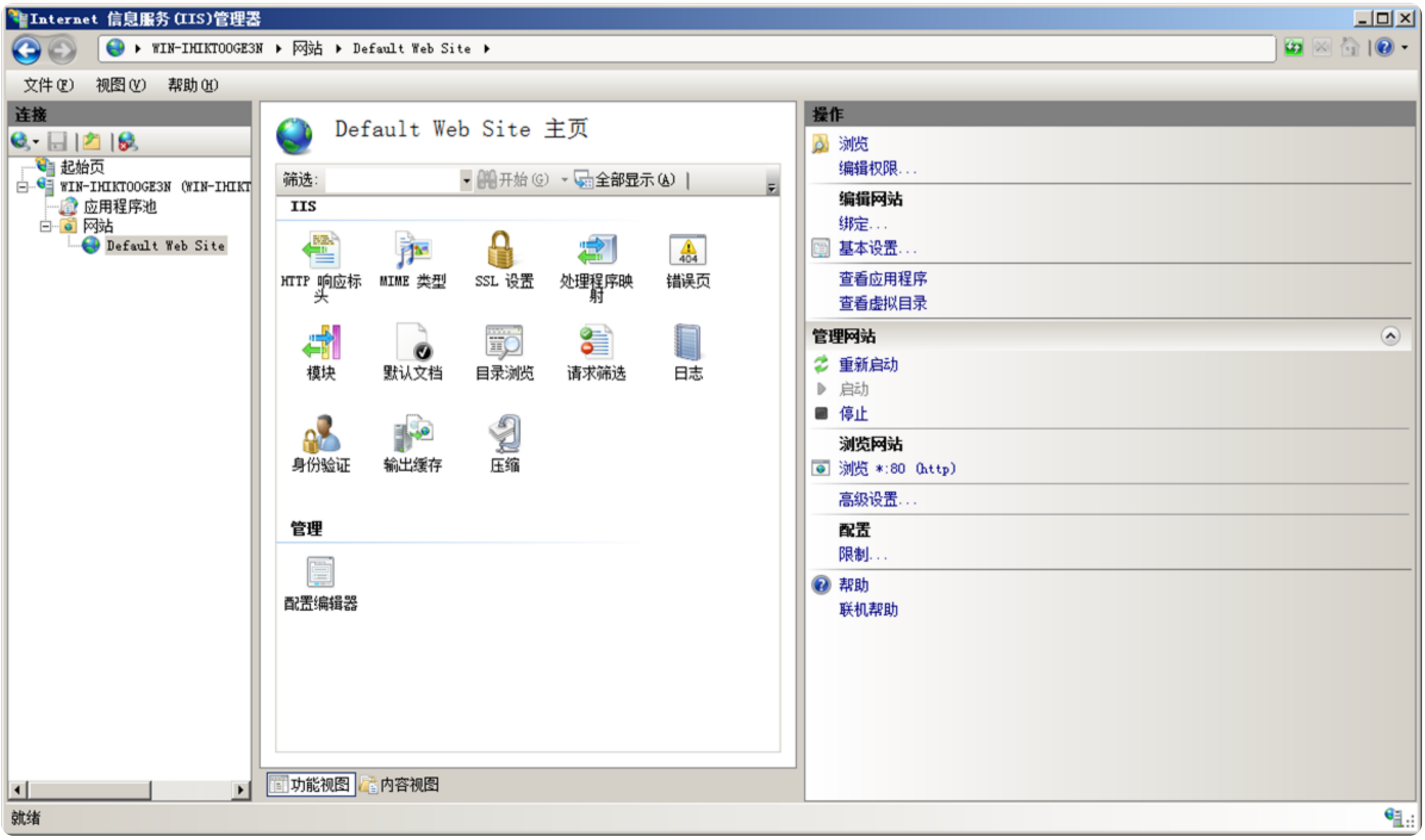
- 编辑 Web网页文件: test.htm

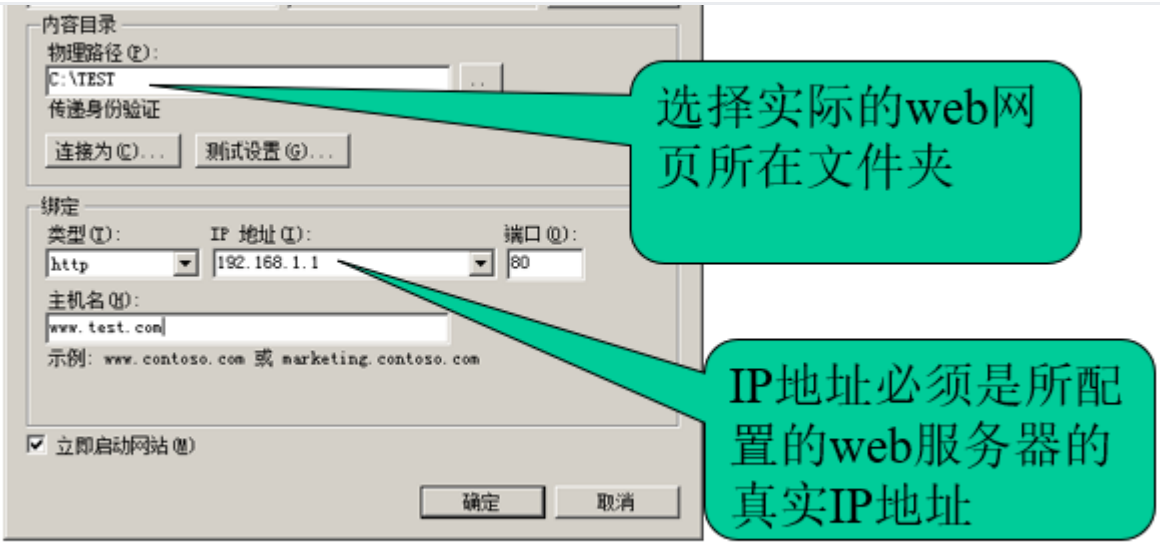


- 将编辑好的Web网页文件放在TEST中

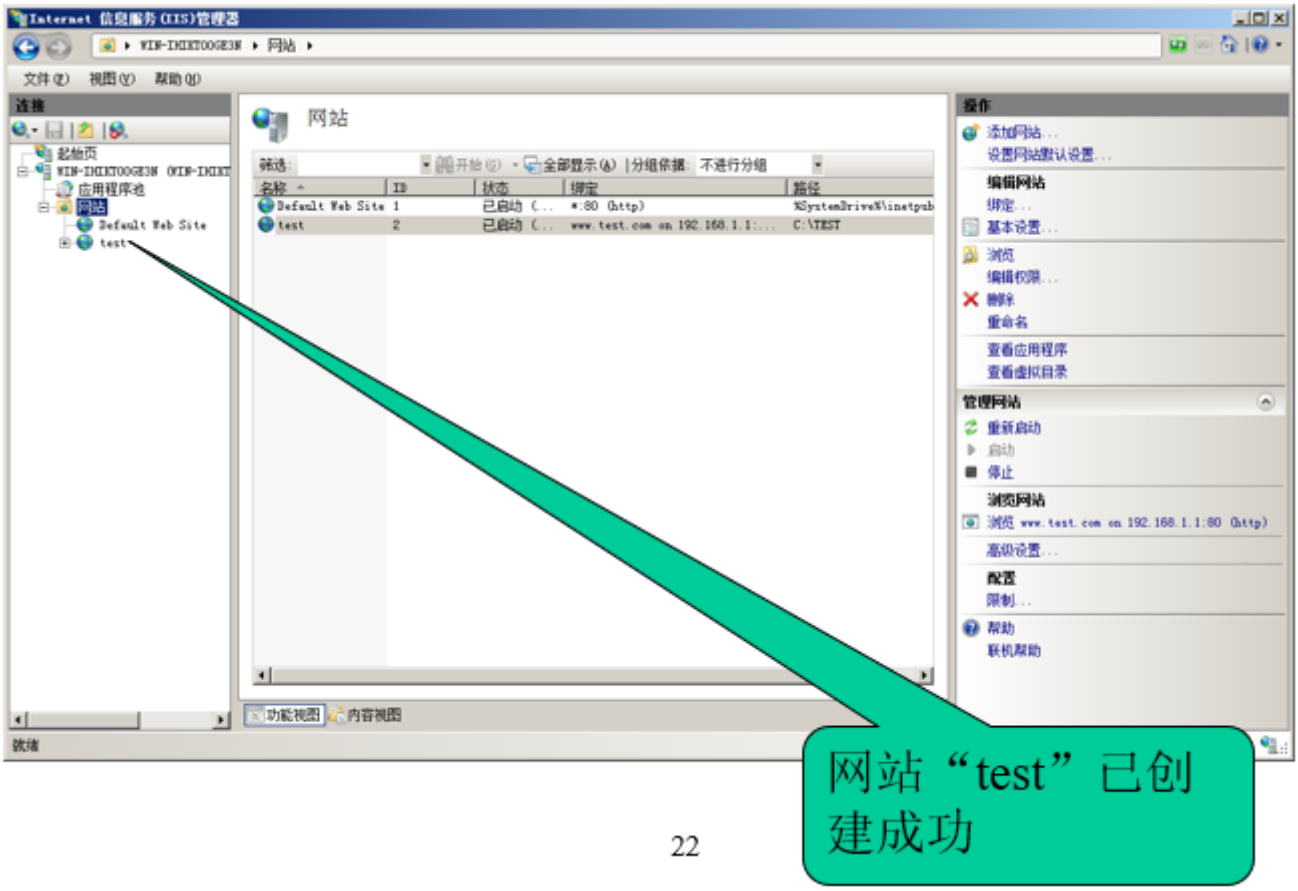
4. 创建Web站点

- 选择 管理工具 → Internet信息服务(IIS) 管理器 → 网站 (右击) → 添加网站

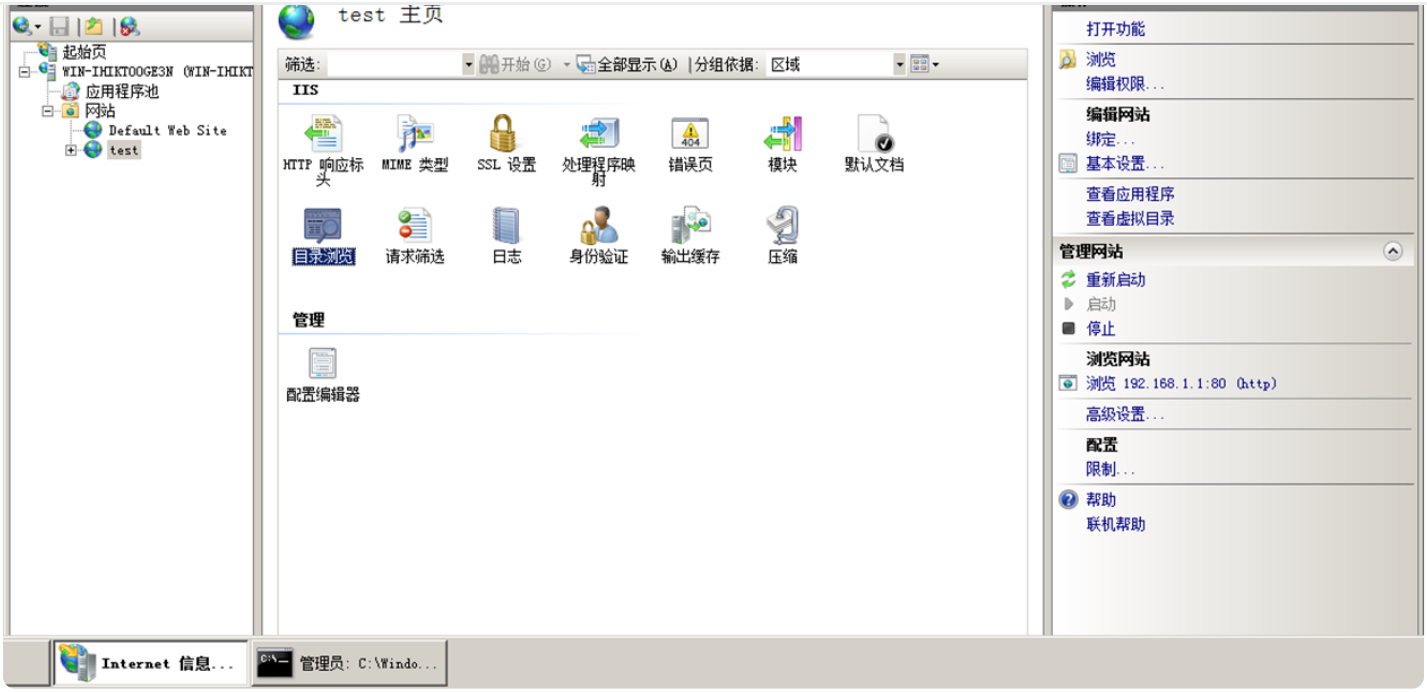




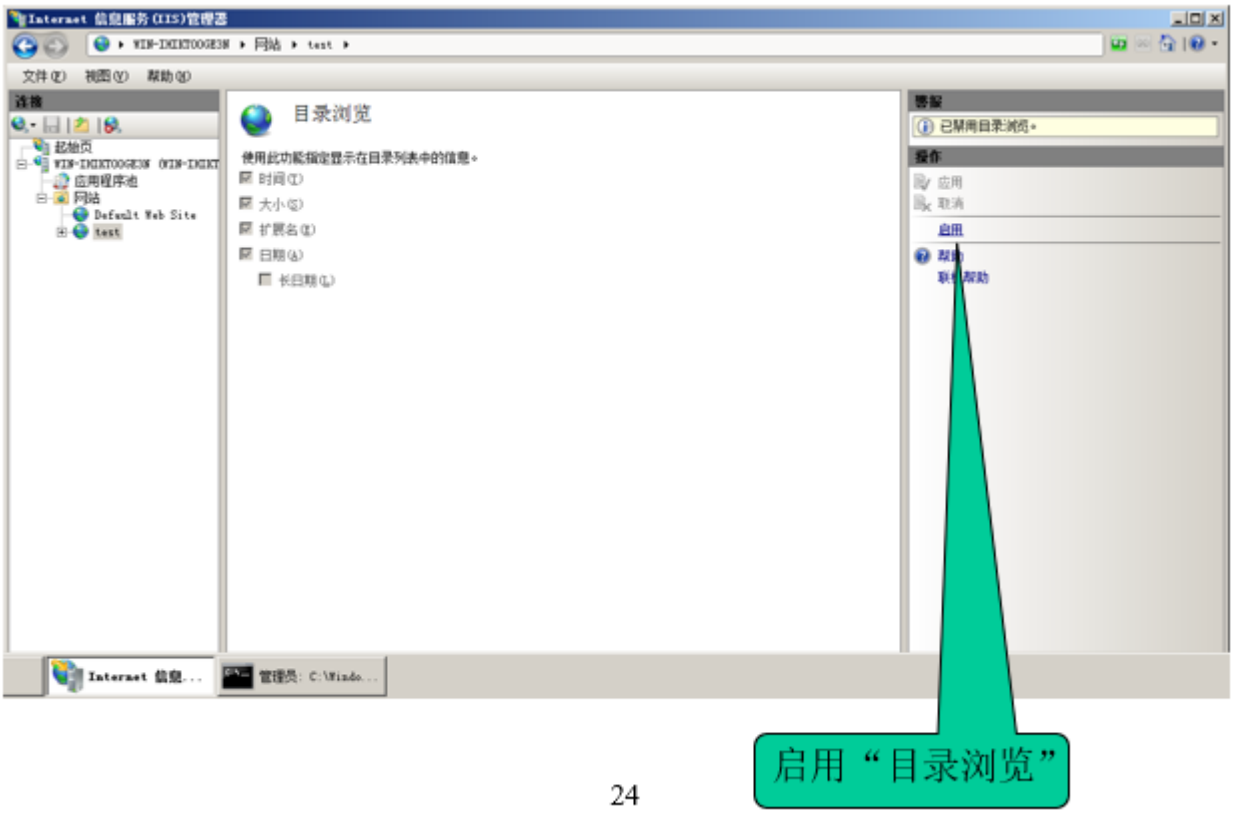
！暂时不输入域名，还不能用域名访问



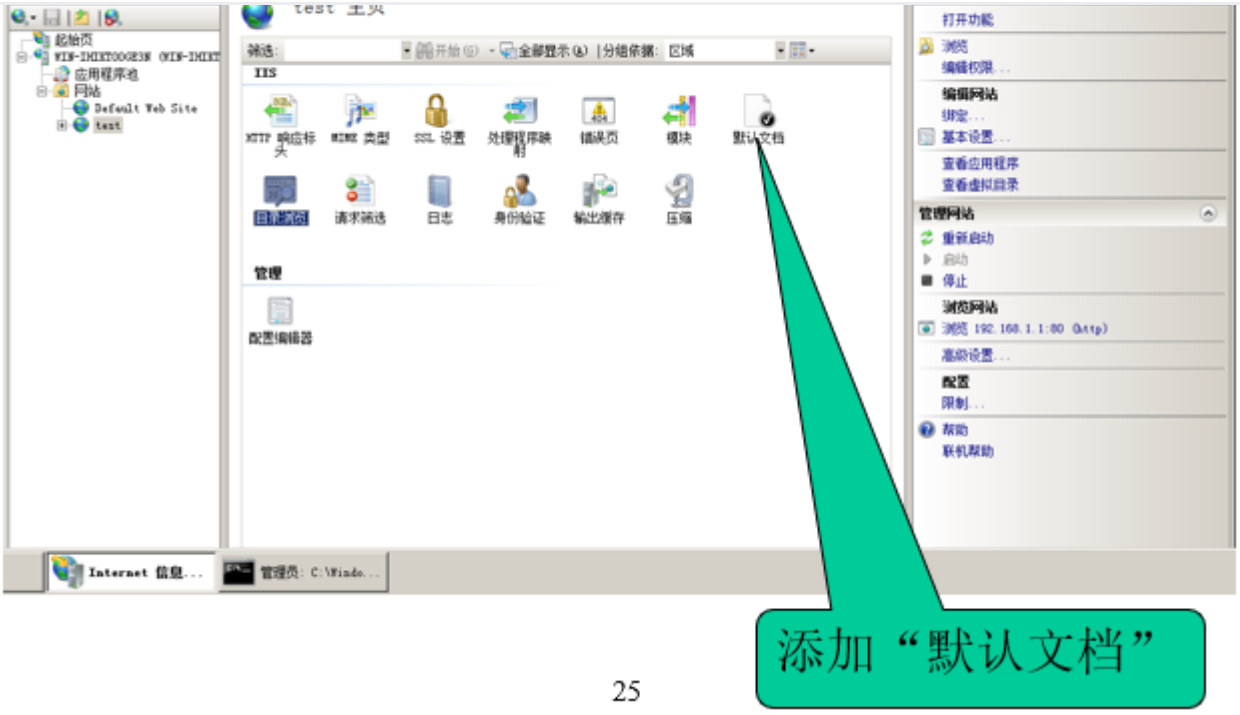
- □□“□□□□”□□



- 点击“目录浏览”图标

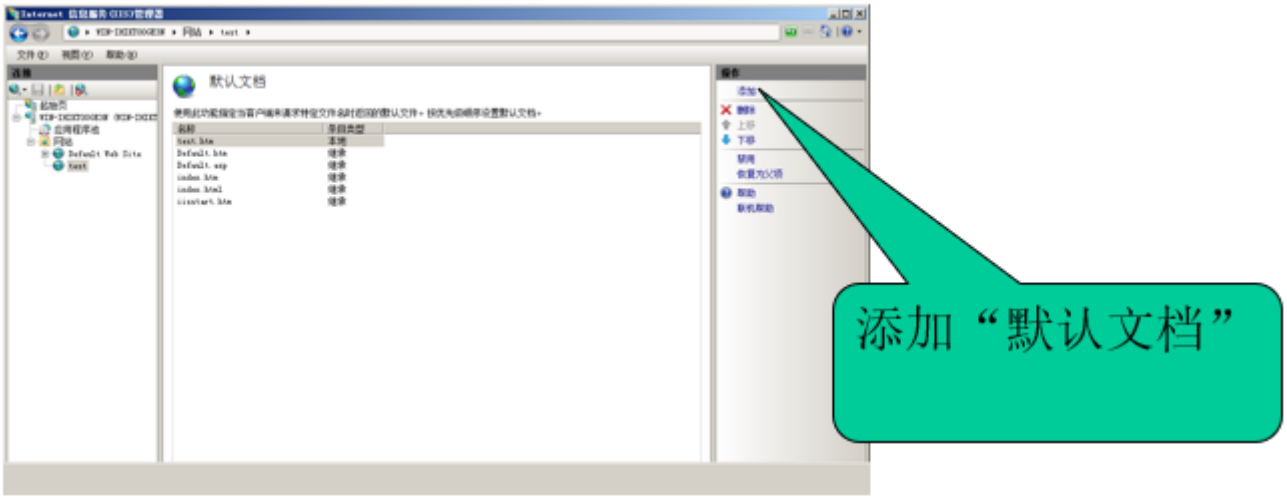


- 点击“启用”



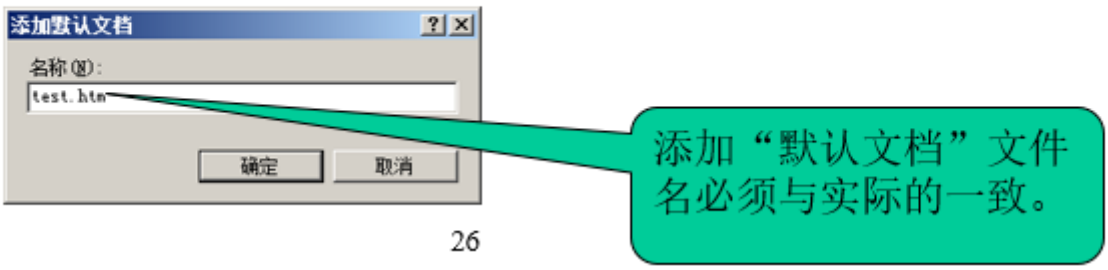
添加"默认文档"

25



名称	条目类型
test.htm	本地
Default.htm	继承
Default.asp	继承
index.htm	继承
index.html	继承
iisstart.htm	继承

添加"默认文档"



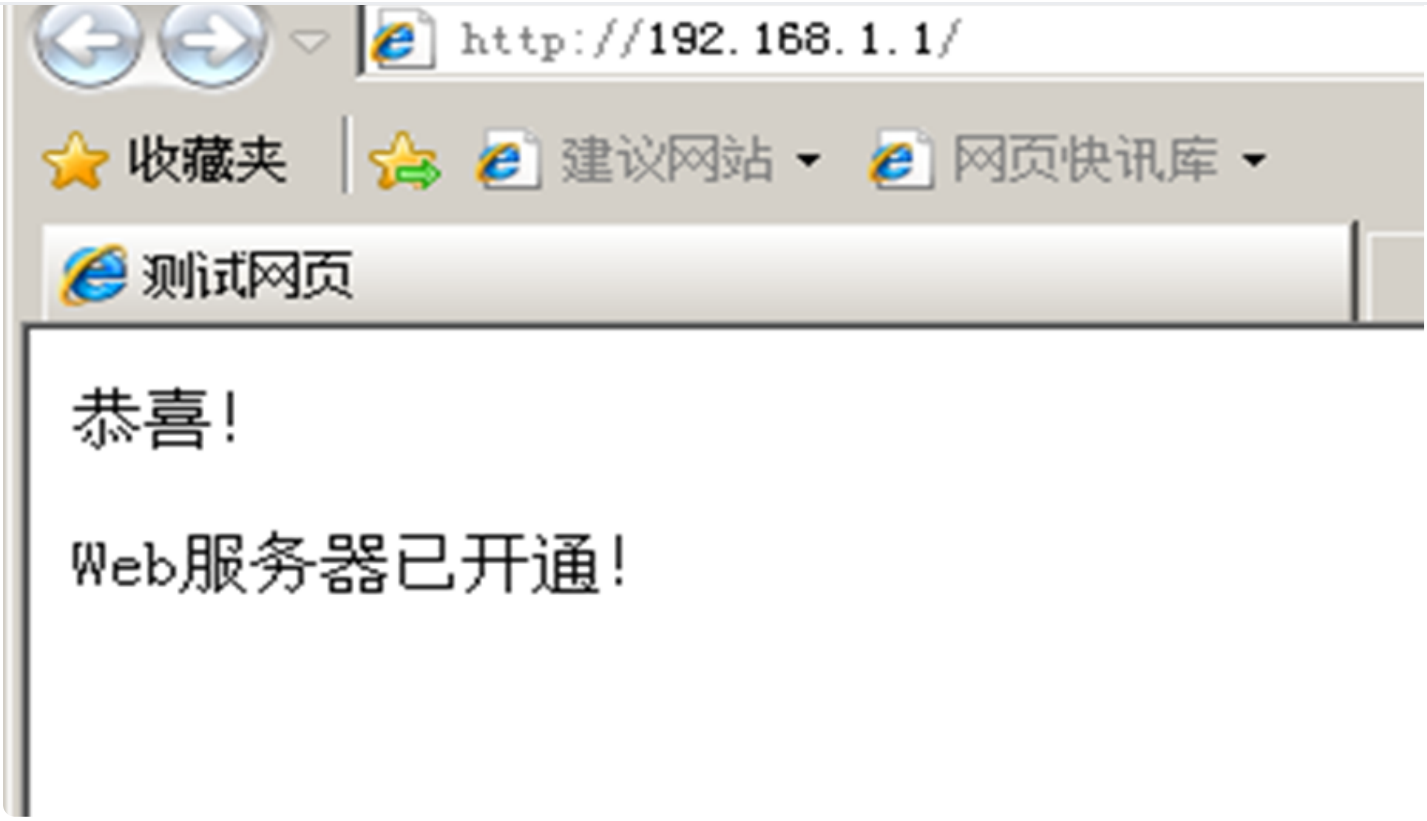
名称(N): test.htm

添加"默认文档"文件名必须与实际的一致。

26

5. 验证Web站点

在浏览器里输入WEB站点指向的网址，就可以浏览你自己制作的网页了。



恭喜！

Web服务器已开通！

DNS服务的安装和配置

> DNS：Domain Name System，域名系统。
>
> 作用：域名和IP地址的相互解析。
>
> 在DNS的数据库中，域名和IP地址的映射关系都被放置在资源记录中。DNS服务器所在的计算机的TCP/IP必须是静态配置的。

1. 安装DNS 服务

   本次实验中，DNS服务器安装在Window Server 2008上。负责主域控制器和客户机的域名服务。

   查看该计算机的TCP/IP是否是静态配置（并记录该DNS服务器的IP地址）的。

Last Updated: 6/15/2023, 1:37:06 PM

Contributors: cworld1