1 计算机网络总结 2

1.1 计算机网络 2

1.1.1 计算机网络概述总结 2

1.1.1.1 定义与核心目标 2

1.1.1.2 发展历程 3

1.1.1.3 网络构成 3

1.1.1.4 网络分类 3

1.1.1.5 网络的核心指标 4

1.1.1.6 网络分层模型 4

1.1.2 物理层 6

1.1.2.1 物理介质 6

1.1.2.2 数据交换方式 7

1.1.2.3 信道复用 9

1.1.3 链路层（Link Layer） 10

1.1.3.1 链路层概述 10

1.1.3.2 物理层与数据链路层的关系 11

1.1.3.3 数据链路层的封装与帧结构 12

1.1.3.4 错误检测与校验 13

1.1.3.5 流量控制 14

1.1.3.6 MAC 15

1.1.3.7 链路层协议 18

1.1.4 网络层（Internet Layer） 19

1.1.4.1 网络层概述 19

1.1.4.2 IP（v4）协议 20

1.1.4.3 IP（v4）地址 23

1.1.4.4 DHCP 协议 25

1.1.4.5 ARP 协议 27

1.1.4.6 NAT（Network Address Translation） 27

1.1.4.7 ICMP 协议 29

1.1.4.8 路由协议 31

1.1.4.9 VPN（虚拟专用网络） 38

1.1.5 传输层（Transport Layer） 39

1.1.5.1 传输层概述 39

1.1.5.2 端口与套接字（Socket） 40

1.1.5.3 UDP 协议 40

1.1.5.4 TCP 协议 41

1.1.5.5 滑动窗口 45

1.1.5.6 TCP 流量控制 47

1.1.5.7 TCP 拥塞控制 48

1.1.6 应用层（Application Layer） 50

1.1.6.1 应用层概述 50

1.1.6.2 DNS 协议（Domain Name System） 53

1.1.6.3 电子邮件 56

1.1.6.4 万维网体系结构 58
1.1.6.5 HTTP 协议（HyperText Transfer Protocol） 60
1.1.6.6 CDN 63
1.1.6.7 其它应用层协议 64

1 计算机网络总结

1.1 计算机网络

1.1.1 计算机网络概述总结

1.1.1.1 定义与核心目标

计算机网络通过**物理连接**与**逻辑协议**的结合，构建了一个高效、透明的资源共享与信息交互平台。

1.1.1.1.1 计算机网络的定义

计算机网络是将**地理位置分散**、具有**独立功能**的计算机系统（包括主机、外部设备等），通过**通信线路和通信设备**互联，并在**网络软件（如操作系统、协议）**的协调管理下，实现资源共享和信息传递的系统。

其核心特征包括：

1. **自主性**：网络中的计算机是独立的，可脱离网络单独运行。
2. **互联性**：通过物理链路（如光纤、电缆）或无线技术实现设备连接。
3. **协议支持**：依赖标准化的通信协议（如 TCP/IP）确保数据传输的可靠性。
4. **透明性**：用户无需了解网络内部结构即可便捷调用资源。

✧ 典型示例：因特网（Internet）是最著名的计算机网络。

1.1.1.1.2 核心目标与功能

计算机网络的核心目标可概括为以下两点：

- **资源共享**
  包括硬件（如打印机、存储设备）、软件（如应用程序）和数据资源（如数据库）。通过共享，用户可降低设备成本、提升资源利用率，并支持协同工作。
  ✧ 示例：企业内部的文件服务器集中管理文档，供多部门调用。
- **信息传递**
  实现高效、可靠的数据传输，支持电子邮件、即时通信、网页访问等多种服务。这一目标依赖通信协议和网络架构设计，确保信息在复杂环境中准确送达

1.1.1.2 发展历程

第一阶段（1950s）：面向终端的远程联机系统（如 SAGE 系统）。
第二阶段（1960s）：分组交换网络（ARPANET）奠定现代网络基础。
第三阶段（1970s-1980s）：OSI 与 TCP/IP 模型推动标准化。
第四阶段（1990s 至今）：互联网普及（IPv4/IPv6）、高速网络（5G）与智能化（SDN）。

1.1.1.3 网络构成

1.1.1.3.1 网络边缘:

位于互联网边缘与互联网相连的计算机和其他设备。
如：桌面计算机、移动计算机、服务器、其他智能终端设备

1.1.1.3.2 网络核心:

由互联端系统的分组交换设备和通信链路构成的网状网络。
如：分组交换路由器、链路层交换机、通信链路(光纤、铜缆、无线电、激光链路)

1.1.1.4 网络分类

计算机网络可以按照地理范围、传输介质、拓扑结构进行多种划分，下面主要介绍按地理范围和传输介质。

1.1.1.4.1 按地理范围划分

- **个人区域网络（PAN -Personal Area Network）**
  - ✧ 定义：覆盖范围 5-10 米的短距离无线网络，支持蓝牙（BT）、NFC 等技术，用于设备间数据传输（如文件共享）。
  - ✧ 扩展功能：与局域网（LAN）协议兼容，支持 ICMP、TCP/IP 等，常用于智能家居和可穿戴设备
- **局域网（LAN - Local Area Network）**
  - ✧ 覆盖范围：通常在 10 公里以内，如办公室、校园或建筑群。
  - ✧ 特点：高传输速率（可达千兆/万兆）、低延迟、设备集中管理，采用以太网（Ethernet）、Wi-Fi 等技术。
  - ✧ 应用场景：企业内部资源共享（如文件服务器、打印机）、智能家居网络。
- **城域网（MAN - Metropolitan Area Network）**
  - ✧ 覆盖范围：延伸至整个城市，连接多个局域网。
  - ✧ 特点：通过光纤骨干网实现跨区域互联，支持高带宽需求（如视频监控、城市级数据中心）。
  - ✧ 示例：城市教育网整合各校资源，实现远程教学资源共享。
- **广域网（WAN - Wide Area Network）**
  - ✧ 覆盖范围：跨越国家或全球，如互联网（Internet）。
  - ✧ 特点：依赖电信运营商基础设施（如光纤、卫星），传输速率较低但扩展性强，采用 IP

协议、MPLS 等技术。

- ✧ 应用：跨国公司跨洲际通信、云计算服务互联。

1.1.1.4.2 按传输介质划分

- **有线网络**

- ✧ 双绞线：成本低、易部署，适用于短距离局域网（如 Cat6 网线支持 10Gbps）。
- ✧ 同轴电缆：抗干扰强，早期用于电视信号和局域网（如 10BASE-2 以太网）。
- ✧ 光纤：长距离传输（达数百公里）、高带宽（Tbps 级别），用于骨干网和数据中心互联。

- **无线网络**

- ✧ 短距离：Wi-Fi（IEEE 802.11 标准）、蓝牙，适用于移动设备接入。
- ✧ 长距离：4G/5G 蜂窝网络、卫星通信，支持广域移动互联。

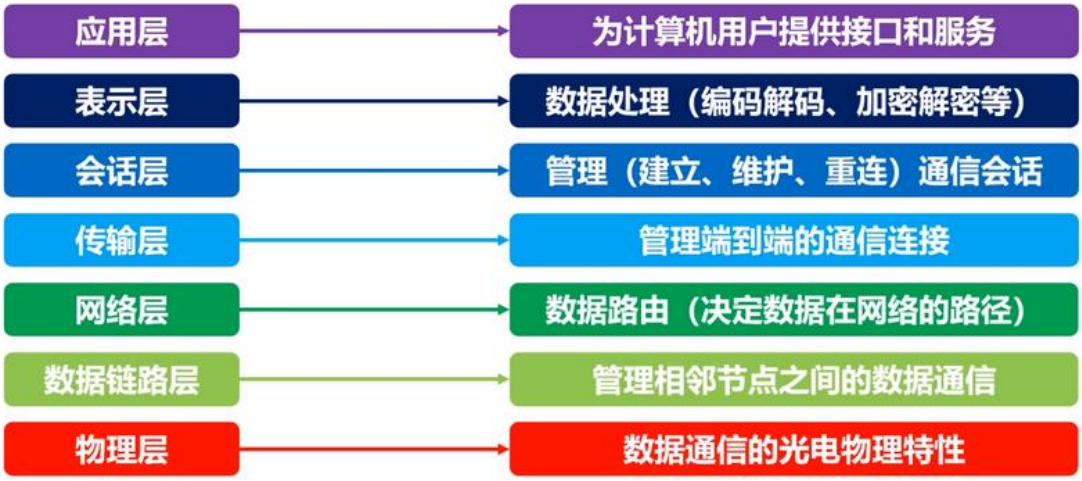
1.1.1.5 网络的核心指标

1. **带宽 (Bandwidth)**：网络在单位时间内能传输的数据量，决定网络的容量和传输速度。
2. **时延 (Latency)**：数据从源到目标的传输时间，影响响应速度，通常包括传播延迟、传输延迟等。
3. **吞吐量 (Throughput)**：实际的传输速率，表示网络的实际数据传输能力。
4. 丢包率 (Packet Loss Rate)：传输过程中丢失的数据包比例，影响网络可靠性和性能。
5. 可靠性 (Reliability)：网络保持数据正确传输的能力，包括抗干扰能力和故障恢复能力。
6. 拥塞 (Congestion)：当网络流量超过承载能力时产生的延迟和丢包现象，影响网络的稳定性。
7. 服务质量 (QoS, Quality of Service)：为不同类型的流量分配优先级，保证实时应用的性能，如语音、视频等。
8. 抖动 (Jitter)：时延波动，特别影响实时通信应用，导致数据传输不均衡。
9. 连接建立时间 (Connection Establishment Time)：从发起连接到建立完成所需的时间，直接影响应用的响应速度。
10. 重传率 (Retransmission Rate)：丢失数据包的重传比例，较高的重传率会增加网络负载和延迟。

1.1.1.6 网络分层模型

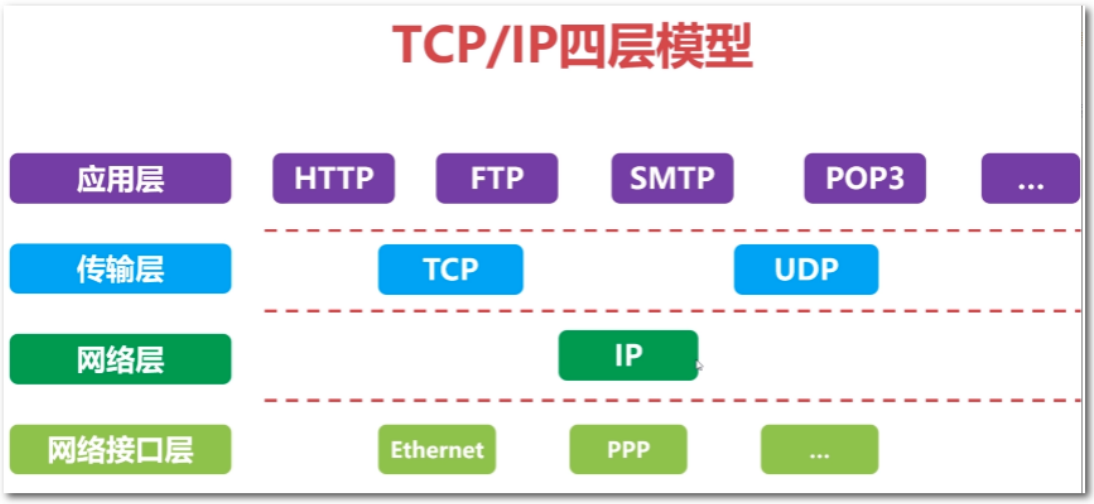
1.1.1.6.1 OSI 七层模型

OSI（开放系统互联，Open Systems Interconnection）是国际标准化组织（ISO）制定的一种网络通信模型，用于描述计算机网络中不同设备间的通信过程。OSI 模型将网络通信划分为七个层次，每个层次负责不同的功能，从物理传输到用户应用的服务，旨在促进不同系统和设备间的互联和兼容。



1.1.1.6.2 TCP/IP 四层模型

TCP/IP 四层模型是用于描述互联网协议栈的模型，基于 TCP/IP 协议族，主要用于实际的网络通信。它简化了 OSI 七层模型，分为四个层次，每个层次完成特定的功能。因其开放性和易用性在实践中得到了广泛的应用，TCP/IP 协议栈也成为互联网的主流协议。



有时候网络接口层同样被叫做链路层，且也有人会把这一层次对应划分成链路层和物理层由此得到五层结构，便于教学理解。

1.1.1.6.3 OSI 和 TCP/IP 对应关系

- OSI 引入了 服务、接口、协议、分层 的概念，TCP/IP 借鉴 OSI 的概念建立 TCP/IP 模型。
- OSI 先有模型，后有协议，先有标准，后进行实践；而 TCP/IP 则相反，先有协议和应用再提出了模型，且是参照的 OSI 模型。
- OSI 是一种理论下的模型，而 TCP/IP 已被广泛使用，成为网络互联事实上的标准。

OSI 七层 模型	TCP/IP 四层模型	对应网络协议

应用层（Application）	应用层	HTTP、TFTP, FTP, NFS, WAIS、SMTP，SMTP 等
表示层（Presentation）		Telnet, Rlogin, SNMP, Gopher 等
会话层（Session）		SMTP, DNS 等
传输层（Transport）	传输层	TCP, UDP 等
网络层（Network）	网络层	IP, ICMP, ARP, RARP, AKP, UUCP, OSPF, EIGRP 等
数据链路层（Data Link）	链路层	FDDI, Ethernet, Arpanet, PDN, SLIP, PPP 等
物理层（Physical）		IEEE 802.1A, IEEE 802.2 到 IEEE 802.11 等

1.1.2 物理层

- 物理层考虑的是**怎样才能在连接各种计算机的传输媒体上传输数据比特流**，而不是指具体的传输媒体。
- 物理层的作用是要尽可能地屏**蔽掉不同传输媒体和通信手段的差异**。
- 用于物理层的协议也常称为**物理层规程**（procedure)。

1.1.2.1 物理介质

1.1.2.1.1 引导型介质

信号在固体介质中传播，例如铜、光纤、同轴电缆

- 光纤
  - 高速运行：高速点对点传输（10-100 Gbps）
  - 低错误率：中继器相距很远，对电磁噪声免疫
- 双绞线
  - 两根绝缘铜线互相缠绕为一对
  - 电话线为 1 对双绞线，网线为 4 对双绞线，广泛用于计算机网络（以太网）双向传输
  - 第 5 类：100 Mbps~1 Gbps；第 6 类：10Gbps
  - 传输距离一般为为 100 米
- 同轴电缆
  - 两根同心铜导线，双向传输
  - 电缆上的多个频率通道
  - 带宽可达 100Mbps
  - 传输距离一般为 200 米

1.1.2.1.2 非引导型介质

信号自由传播，例如无线电（陆地无线电、卫星无线电信道)

- 无线电
  - 电磁频谱中各种“波段”携带的信号
  - 没有物理“电线”
  - 不依赖介质的广播

- 半双工（发送方到接收方）

- 无线链路类型
  - 无线局域网（WiFi）
  - 10-100 Mbps；10 米
  - 广域（如 3/4/5G 蜂窝）
  - 在~10 公里范围内
  - 蓝牙：短距离，有限速率
  - 地面微波：点对点；45 Mbps
  - 同步卫星：36000km 高空， 280 毫秒的往返时延
  - 低轨卫星：近地，但围绕地球高速运动，需要大量卫星才能覆盖地球

1.1.2.2 数据交换方式

1.1.2.2.1 分组交换

- **定义：**

分组交换是一种将数据分成多个较小的“分组”（Packet），并通过网络中的不同路径发送，每个分组在传输过程中可能会经过不同的路径，最终在目的地重组为完整数据的交换方式。

- **工作原理：**
  1. 数据分割：将发送的数据分成多个小数据包（每个包包含数据、源地址、目的地址、顺序号等信息）。
  2. 独立路由：每个数据包可以通过不同的路径在网络中独立传输，路由器根据目的地址选择最佳路径。
  3. 到达后重组：数据包在目的地按顺序重新组合成原始数据。
  4. 动态路径：由于不同数据包可以通过不同的路由路径，因此分组交换具有较高的灵活性和动态调整能力。

- **特点：**
  - 高效的资源利用：不同的数据包共享网络资源，当网络不繁忙时，带宽和路由资源得到了充分的利用。
  - 无须预先建立连接：与电路交换不同，分组交换不需要在通信开始前建立固定的连接，通信双方可以随时开始数据传输。
  - 灵活性强：数据可以通过不同的路径传输，可以有效避免单一路径发生故障的情况。

- **缺点：**
  - 延迟和抖动：由于数据包可能经过不同的路径，可能会出现延迟和网络抖动。不同的路径可能导致传输时间的不均匀，影响实时性较强的应用（如语音和视频通话）。
  - 数据包丢失：在网络拥塞或路径故障的情况下，部分数据包可能丢失，需要重传。

- **应用场景：**
  - 互联网：现代互联网就是基于分组交换技术建立的。网页浏览、电子邮件、文件传输等都依赖分组交换。
  - 数据通信：分组交换非常适合大量短时间的通信数据交换，如文件传输、即时消息、社交网络等。

1.1.2.2.2 电路交换

- **定义：**

电路交换是一种在通信双方之间建立固定的通信路径（电路），并在整个通信过程中保持该路径的交换技术。通信的每一方都可以通过一条专用的物理路径进行连续的、稳定的通信，直到通信结束。

- **工作原理：**

1. 建立连接：在通信开始前，网络会为通信双方在交换机之间建立一条专用的通信路径（例如通过多个交换机）。
2. 持续通信：在通话过程中，整个通信链路会被独占，并持续传输数据。
3. 断开连接：通信结束后，电路被断开，资源被释放。

- **特点：**
  - ✧ 预先分配带宽：电路交换为通信双方预先分配了一条专用通道，因此在整个通信过程中带宽不会被共享。
  - ✧ 稳定性好：通信过程中的数据传输稳定，不受其他通信影响。
  - ✧ 实时性强：非常适合实时通信应用，如电话系统，因为数据传输的延迟和波动都比较小。

- **缺点：**
  - ✧ 低效利用资源：即使通信双方没有发送数据，整个通信路径也会占用网络资源，造成资源浪费。
  - ✧ 建立连接时间长：为了建立一条固定的通信电路，需要一些时间，因此在通信开始之前有较长的等待时间。
  - ✧ 灵活性差：电路交换适用于固定的、长时间的数据传输，无法适应动态变化的需求。

- **应用场景：**

传统电话网络：这是典型的电路交换应用，电话系统通过专用的电路为每个通话提供稳定的连接。

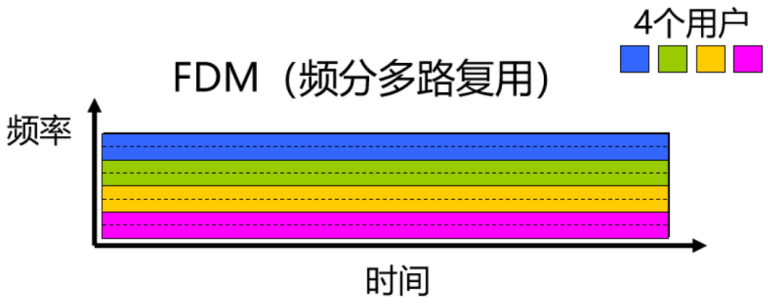
视频会议系统：需要稳定、高质量、低延迟的实时通信，适合使用电路交换。。

1.1.2.2.3 电路交换的多路复用

电路交换的多路复用（Multiplexing）是指在同一物理传输介质上，通过技术手段将多个独立的通信信号进行组合和复用，从而实现多个通信会话共享同一条物理链路。这项技术提高了通信线路的利用效率，避免了网络资源的浪费。

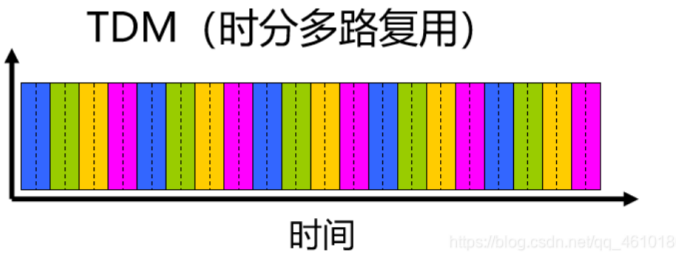
- **频分多路复用 FDM**

频分多路复用通过将频带划分为多个子频带，每个通信信号占用一个不同的频段进行传输。这些信号共享同一物理传输介质（例如同轴电缆或无线电频谱），但使用不同的频率。



- **时分多路复用 TDM**

时分多路复用是通过在时间上划分传输信道，每个通信会话被分配一个固定的时间片（时隙）来传输数据。这些时隙按照固定的顺序循环使用，多个通信会话可以共享同一条物理链路。



1.1.2.3 信道复用

- **基本概念：**

  信道复用（Channel Multiplexing）是一种在有限的传输介质（如电缆、光纤、无线频谱等）上同时传输多个信号的技术。信道复用通过有效利用可用的带宽，将多个通信信号合并在同一信道上传输，以提高资源利用率和通信效率。

- **基本原理：**

  信道复用的核心思想是将多个信号或数据流以某种方式组合在一起，然后通过同一物理信道进行传输。接收端将这些复用的信号拆分或解复用，恢复出各个独立的信号。复用技术通常依赖于时间、频率、代码或光波等维度的划分。

- **优点：**
  - ✧ 提高带宽利用率：通过信道复用技术，多个信号可以共享同一物理传输介质，从而大大提高了带宽的使用效率。
  - ✧ 提升网络容量：在有限的资源下，复用技术能够同时支持更多的通信流量，提高网络的吞吐量。
  - ✧ 降低通信成本：通过共享同一物理信道，可以减少网络基础设施的建设和维护成本。
  - ✧ 增强灵活性：信道复用技术允许不同类型的数据流共享网络资源，适应不同的通信需求。

1.1.2.3.1 时分复用（TDM, Time Division Multiplexing）

- ✧ **原理**：将时间划分为多个时间片，每个信号在固定的时间片内传输。各个信号轮流

在不同的时间片上传输，确保每个信号都能在时间上独立进行传输。

- **应用场景**：常用于电话通信、卫星通信等需要稳定传输的场景。
- **示例**：在一个时分复用系统中，多个电话通话共享同一条线路，每个电话通话在不同的时隙内传输数据。

1.1.2.3.2 频分复用（FDM, Frequency Division Multiplexing）

- **原理**：将可用的频率带宽划分成多个子频带，每个信号使用不同的频带进行传输。每个信号被分配一个固定的频率范围，这些频率范围互不干扰，可以同时传输多个信号。
- **应用场景**：广泛应用于广播电视、无线通信等场合。
- **示例**：无线电广播中的每个电台都使用不同的频率进行广播，保证信号互不干扰。

1.1.2.3.3 波分复用（WDM, Wavelength Division Multiplexing）

- **原理**：在光纤通信中，波分复用将不同波长（光的颜色）分配给不同的信号。每个信号使用不同的光波长进行传输，允许多个信号在同一根光纤上传输。
- **应用场景**：广泛应用于高速的光纤通信系统，特别是在大容量数据传输中。
- **示例**：现代光纤通信系统通过多个波长的光信号传输多个数据流。

1.1.2.3.4 码分复用（CDM, Code Division Multiplexing）

- **原理**：每个信号使用一个唯一的代码（码字）进行标识，所有信号共享相同的频率带宽和时间资源，但通过不同的码字进行区分。接收端使用相同的代码对信号进行解码，从而提取出独立的信号。
- **应用场景**：常用于无线通信系统，如 CDMA（码分多址）系统。
- **示例**：在 CDMA 中，多个用户通过不同的码字在同一频率频道上进行通信，接收端通过对应的码字将每个用户的信号提取出来。

1.1.2.3.5 空分复用（SDM, Space Division Multiplexing）

- **原理**：利用空间不同的物理路径或天线进行信号传输，使得每个信号都在不同的空间位置传输。通过多个物理通道进行并行传输。
- **应用场景**：广泛应用于无线通信中的多天线技术，如 MIMO（多输入多输出）技术。
- **示例**：MIMO 技术通过多个天线同时传输不同的数据流，实现空间上的信道复用。

1.1.3 链路层（Link Layer）

1.1.3.1 链路层概述

链路层是 TCP/IP 协议栈中的最底层，它负责在物理网络上直接传输数据帧。它的主要任务是处理网络设备之间的数据传输和确保数据在物理链路中的可靠性。链路层既包括数据封装，也包括错误检测、流量控制、拥塞控制等功能。

1.1.3.1.1 数据链路层的作用

链路层的作用主要包括以下几个方面：

- ✧ **数据封装**：将网络层传输的数据（即包）封装成帧，添加头部信息和尾部校验。
- ✧ **帧的传输**：负责物理介质上帧的传输，确保数据从源主机到目标主机。
- ✧ **错误检测与纠正**：通过校验和（CRC 等技术）对数据进行错误检测，并通过重传机制（如 ARQ）进行错误纠正。
- ✧ **流量控制**：调节数据传输速率，避免接收方的缓冲区溢出。
- ✧ **链路层协议支持**：如以太网、PPP 等协议，为上层协议提供支持。

1.1.3.1.2 数据链路层提供的服务

链路层根据不同的传输要求，可以提供不同类型的服务：无确认的无连接服务、带确认的无连接服务，以及带确认和连接的服务，这些服务的选择依据网络环境、信道的可靠性以及通信的实时性需求。

- **无确认 无连接 服务（Unacknowledged connectionless service）**
  - ✧ **特性**：在这种服务模式下，链路层的接收方不会对收到的帧进行确认，也不会保证帧的可靠交付。发送方在发送数据时不需要建立连接，也不会等待接收方的确认。
  - ✧ **适用场景**：适用于误码率低的可靠信道，或实时通信场景，如音视频流传输等，因为实时性要求较高，并且稍微丢失数据不会影响整体质量。
  - ✧ **网络实例**：以太网（Ethernet）是常见的应用实例，通常基于该服务进行数据传输。

- **有确认 无连接 服务（Acknowledged connectionless service）**
  - ✧ **特性**：每一帧的数据都会得到单独的确认。接收方需要发送确认帧（ACK），确认收到数据。这种服务模式通常没有建立持久的连接，数据传输仍然是面向无连接的，但每次数据传输需要确认。
  - ✧ **适用场景**：适用于不可靠的信道，比如无线信道。在无线通信中，由于信道的不稳定性，丢包较为常见，因此需要通过确认机制来提高可靠性。
  - ✧ **网络实例**：IEEE 802.11（Wi-Fi）协议就是一个典型的例子，使用这种确认机制确保数据的可靠传输。

- **有确认 有连接 服务（Acknowledged connection-oriented service）**
  - ✧ **特性**：在这种服务模式下，链路层不仅有确认机制，而且还需要建立连接。连接的建立、数据传输和断开是有序的，通常有一定的控制协议来管理连接状态。每次数据传输都需要经过确认，并且保证传输的顺序和完整性。
  - ✧ **适用场景**：适用于长延迟的不可靠信道，比如卫星通信或其他需要确保高可靠性的通信链路。
  - ✧ **网络实例**：例如，PPP（Point-to-Point Protocol）协议就是一种有确认、有连接的协议，它用于建立点对点的连接并确保数据传输的可靠性。

1.1.3.2 物理层与数据链路层的关系

物理层传输比特，链路层通过这些比特封装成数据帧，并为数据传输提供可靠性。

1.1.3.2.1 物理层与链路层的关系

- ✧ 数据链路层依赖于物理层提供的通信媒介来传输数据。
- ✧ 物理层负责将数据以物理信号的形式传送，数据链路层则将这些信号封装成帧并进行管理和控制，确保数据的有效和可靠传输。
- ✧ 在发送数据时，数据链路层将数据封装成帧，物理层则负责将帧转换为信号并通过物理介质传送出去。

1.1.3.2.2 物理层与链路层的区别

- ✧ **物理层**：物理层主要负责在物理介质上传输比特流，定义了物理信号的电气特性、传输介质和连接方式。它关心的是比特的传输，而不关心数据的结构和内容。
- ✧ **数据链路层**：数据链路层则是负责在物理层上以数据帧的形式传输数据，依靠协议控制，解决了如何利用物理介质进行可靠传输的问题。

1.1.3.3 数据链路层的封装与帧结构

1.1.3.3.1 数据链路层的封装

数据链路层的封装涉及将来自网络层的数据封装成一个**数据帧**，这个帧在物理介质上传输。数据链路层的封装包括以下几个步骤：

1. **接收数据**：网络层提供一个数据包（Packet），即 IP 包或其他层协议的数据单元。
2. **添加帧头和帧尾**：数据链路层在数据包的前后添加控制信息，构成帧
   - ✧ **帧头**：用于标识数据帧的开始，包含目标地址、源地址、类型字段等信息。
   - ✧ **数据部分**：即网络层传递下来的数据包，这部分数据会被封装在帧中。
   - ✧ **帧尾**：通常包括校验信息（如 CRC 校验）以确保数据在传输过程中没有发生错误。
3. **发送帧**：将封装后的帧发送到物理层，准备通过物理媒介进行传输。

1.1.3.3.2 帧结构

数据链路层的帧结构主要由以下几个部分组成：

1. **帧起始标志（Frame Start Flag）**：
   这是帧的起始部分，用于标识数据帧的开始，通常是一个特定的比特模式（如 HDLC 协议中的 01111110）。起始标志帮助接收方识别数据帧的边界。
2. **目标地址（Destination Address）**：
   用于标识数据帧的目的地址，即接收方的物理地址。在以太网中，这通常是 MAC 地址。
3. **源地址（Source Address）**：
   表示发送方的物理地址，也通常是 MAC 地址。
4. **长度或类型字段（Length/Type）**：
   长度字段：指示数据部分的长度。
   类型字段：指示上层协议的类型，例如，IPv4 或 ARP 协议。
5. **数据部分（Data）**：
   这是从网络层传递下来的实际数据，可能是 IP 数据包或其他上层协议的数据单元。
6. **帧校验序列（Frame Check Sequence, FCS）**：

这是用于数据验证的校验信息。接收方使用它来验证接收到的数据是否完整和正确，通常使用 CRC（循环冗余校验）来生成校验码。

7. **帧结束标志（Frame End Flag）**：

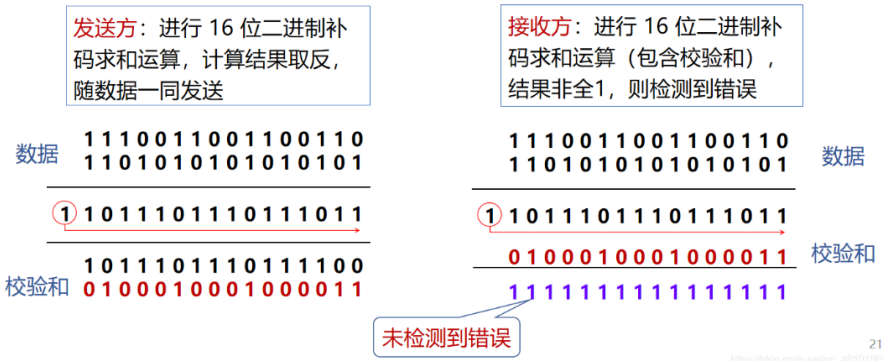
   标志帧的结束，与起始标志一样，帮助接收方识别数据帧的边界。

1.1.3.4 错误检测与校验

1.1.3.4.1 校验和与奇偶校验

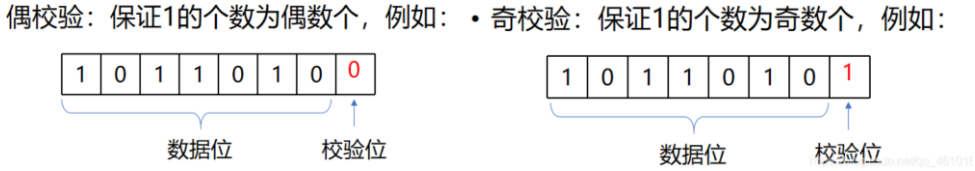
1.1.3.4.1.1 校验和

校验和是一种简单的错误检测方法，通过将数据包中的所有字节相加得到一个和，并将结果放入数据包中，接收方进行验证。



1.1.3.4.1.2 奇偶校验

数据传输中可以在数据位中增加一位用于检测位，确保接收的数据中的 1 和 0 的数量满足奇数或偶数规则。



1.1.3.4.2 循环冗余校验（CRC）

CRC（Cyclic Redundancy Check）是一种更强大的错误检测方法。它基于多项式运算，在数据帧中计算出一个校验值，接收方通过相同的计算方式检查数据是否发生错误。

CRC 校验码计算方法

- ✧ 设原始数据 D 为 k 位二进制位模式
- ✧ 如果要产生 n 位 CRC 校验码，事先选定一个 n+1 位二进制位模式 G (称为生成多项式，收发双方提前商定)，G 的最高位为 1
- ✧ 将原始数据 D 乘以 2^n （相当于在 D 后面添加 n 个 0），产生 k+n 位二进制位模式，用 G 对该位模式做模 2 除，得到余数 R（n 位，不足 n 位前面用 0 补齐）即为 CRC 校验码

1.1.3.4.3 错误处理机制

1.1.3.4.3.1 错误检测

错误检测是通过添加校验信息来识别数据是否在传输过程中出现了错误。
常见的错误检测方法包括：

- **循环冗余校验（CRC，Cyclic Redundancy Check）**
- **奇偶校验（Parity Check）**
- **检验和（Checksum）**

1.1.3.4.3.2 错误纠正

错误纠正是指在数据传输过程中，能够自动修复错误数据的机制。常见的错误纠正方法有：

- **自动重传请求（ARQ，Automatic Repeat reQuest）**：
  - **原理**：接收方在发现错误后，向发送方请求重新传输数据。常见的 ARQ 类型有：
    - **停止等待 ARQ（Stop-and-Wait ARQ）**：发送方每发送一个数据帧后，必须等待接收方的确认。如果没有收到确认，则重新发送数据。
    - **连续 ARQ（Continuous ARQ）**：发送方可以连续发送多个数据帧，在等待确认的同时发送后续帧。接收方会返回每个帧的确认或重传请求。
  - **优点**：简单、可靠，适用于错误率较高的环境。
  - **缺点**：增加了延迟，特别是在长距离或高延迟网络中。
- **前向纠错（FEC，Forward Error Correction）**：
  - **原理**：发送方在数据中添加冗余信息，使接收方可以在错误发生时自行修复数据，而无需重传。这些冗余信息通常是通过特殊的编码（如汉明码、卷积编码等）实现的。
  - **应用**：FEC 常用于实时通信系统（如卫星通信、视频会议等），因为它可以减少重传的需求。
  - **优点**：无需重传，实时性较好。
  - **缺点**：增加了数据的冗余，导致带宽的浪费。

1.1.3.5 流量控制

链路层的流量控制通常分为两种机制：基于反馈（feedback-based）和基于速率（rate-based）的控制。

1.1.3.5.1 链路层流量控制的背景

在链路层的通信中，接收方的缓冲区（Buffer）有大小限制。如果发送方发送数据过快，超过了接收方的处理能力，可能会导致接收方缓冲区溢出，进而丢失数据。因此，链路层流量控制的目标就是避免这种情况，通过有效地控制发送速率，确保接收方能够按时处理收到的数据。

1.1.3.5.2 基于反馈（Feedback-based）流量控制

基于反馈的流量控制方法依赖于接收方定期向发送方发送反馈信息。通过这些反馈信息，接收方告知发送方自己当前的缓冲区状态或处理能力，从而调整发送方的发送速率。

- **工作原理：**
  - ✧ 接收方反馈：接收方在收到数据帧后，会向发送方发送一个确认（ACK）帧，其中包括其当前的缓冲区状况或剩余容量。
  - ✧ 发送方调整：发送方根据接收到的反馈信息来决定是否继续发送数据或减缓发送速率。如果接收方的缓冲区满了，发送方就需要暂停发送数据，直到接收到接收方的继续请求。
- **优点：**
  - ✧ 简单有效，能够根据接收方的状态动态调整发送速率。
  - ✧ 对缓冲区溢出问题有较好的控制，防止丢失数据。
- **缺点：**
  - ✧ 反馈机制可能引入延迟，尤其是在高延迟或长距离的网络中，可能影响流量控制的实时性。
  - ✧ 需要频繁交换控制信息，增加了协议的复杂性。

1.1.3.5.3 基于速率（Rate-based）流量控制

基于速率的流量控制是另一种方式，发送方根据内建机制限制自己的发送速率，而不依赖于接收方的反馈。发送方会自行决定何时和以多快的速率发送数据，通常是基于固定的速率控制。

- **工作原理：**
  - ✧ 发送方自我调节：发送方根据网络或链路的当前状况（如链路带宽、延迟等）自我调整发送速率，避免发送过快导致拥塞或丢包。
  - ✧ 不依赖反馈：与基于反馈的流量控制不同，基于速率的流量控制并不依赖于接收方的缓冲区状况，而是基于发送方自身对网络状况的判断。
- **优点：**
  - ✧ 实现简单，不需要频繁的反馈机制。
  - ✧ 在网络条件稳定的情况下可以提供较为稳定的数据传输。
- **缺点：**
  - ✧ 无法及时响应接收方的缓冲区状态变化，因此在网络或接收方条件突然变化时可能导致数据丢失。
  - ✧ 可能无法充分利用链路资源，特别是在接收方缓冲区有空余空间时，发送方的速率可能受限。

1.1.3.6 MAC

MAC（Media Access Control，媒体访问控制）是数据链路层的一部分，主要负责如何在共享媒介上访问和传输数据。在局域网中，多个设备通常共享同一个物理介质，因此需要一种机制来控制设备如何在媒介上发送和接收数据。MAC 层通过管理介质访问来确保数据传输的顺利进行。

1.1.3.6.1 MAC 地址

- **定义**：MAC 地址是硬件地址，用于唯一标识网络中的每个设备。它通常由设备厂商在

生产时赋予，每个设备在网络上的 MAC 地址都是唯一的。

- **格式**：MAC 地址通常是 48 位（6 字节），以 16 进制表示，如 00:1A:2B:3C:4D:5E。
- **作用**：MAC 地址用于局域网内的设备标识和数据帧的传输。数据链路层使用 MAC 地址来确定数据包的目的地和来源设备。
- **广播与组播**：
  - ✧ 广播是指数据包同时发送给网络中所有的设备。在以太网中，广播 MAC 地址是 FF:FF:FF:FF:FF:FF，它意味着数据包发送给所有设备。
  - ✧ 组播是指数据包发送给特定一组设备。组播 MAC 地址是某一范围的 MAC 地址，用于指定一组设备，而不是全体设备。

1.1.3.6.2 MAC 协议

MAC 协议是控制和协调多个设备在同一通信媒介上如何访问和传输数据的规则集合。它决定了何时以及如何允许设备发送数据，以避免冲突。

- ✧ **CSMA/CD（Carrier Sense Multiple Access with Collision Detection）：**
  以太网中常用的 MAC 协议，适用于共享介质。设备在发送数据之前会监听信道是否空闲，若空闲则发送；若发生冲突，则退避一段时间后重试。
- ✧ **CSMA/CA（Carrier Sense Multiple Access with Collision Avoidance）：**
  Wi-Fi 中常用的 MAC 协议，避免冲突发生。设备发送数据前先发送请求信号，确认信道空闲后再发送数据。
- ✧ **TDMA（Time Division Multiple Access）：**
  时分多址，在每个时隙分配给不同设备传输数据，避免冲突。适用于有固定时间窗口的数据传输。
- ✧ **FDMA（Frequency Division Multiple Access）：**
  频分多址，设备根据频率分配不同的信道进行通信，避免冲突。

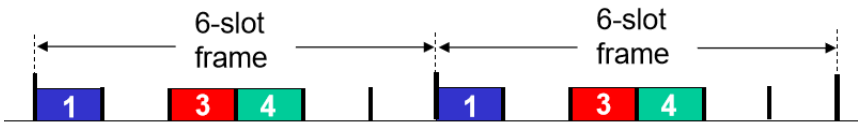
1.1.3.6.3 MAC 信道分配

1.1.3.6.3.1 时分多址接入-TDMA: time division multiple access

时分多址，在每个时隙分配给不同设备传输数据，避免冲突。适用于有固定时间窗口的数据传输。

- ✧ 按顺序依次接入并使用信道
- ✧ 每个用户使用固定且相同长度的时隙
- ✧ 某时隙轮到某用户使用时，该用户没有数据要发送，则该时隙被闲置

例子: 6-user LAN, 1,3,4 时隙有数据发送, 2,5,6 时隙被闲置

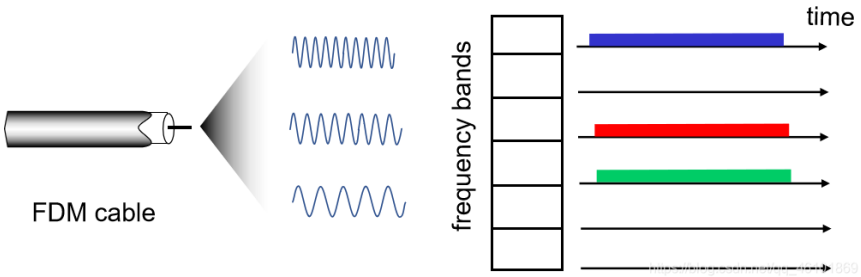


1.1.3.6.3.2 频分多址接入-FDMA: frequency division multiple access

频分多址，设备根据频率分配不同的信道进行通信，避免冲突。

- ✧ 信道总频带被划分为多个相同宽度的子频带
- ✧ 每个用户占用一个子频带，不管用户是否有数据发送

例子: 6-user LAN, 1,3,4 频带有数据发送, 2,5,6 频带被闲置



1.1.3.6.4 MAC 多路访问（随机访问）

MAC 多路访问（Multiplexing）是指在共享的传输介质（如无线信道或有线网络）上，多个设备如何通过不同的方式协调和管理对介质的访问，以避免冲突并有效利用带宽。

1.1.3.6.4.1 ALOHA 自动分配通道

- **ALOHA 自动分配通道**是一种简单的通信协议，允许设备随机在共享信道上发送数据。若发生碰撞，设备会等待随机时间后重新发送。

- **两种 ALOHA**
  - ✧ **纯 ALOHA 协议**设备在任何时刻可以开始发送数据，一旦发生碰撞，它会在一个随机的时间后重发。
  - ✧ **分隙 ALOHA** 信道被分成固定时隙，每个设备仅能在时隙开始时发送数据。这种方式减少了碰撞的概率，因为设备仅在时隙的边界上发送。

- **ALOHA 协议的特点**
  - ✧ **简单性**：ALOHA 协议非常简单，没有复杂的信道管理机制。
  - ✧ **低效率**：由于设备在发送时没有协调，很容易发生碰撞，导致大量的带宽浪费。特别是在信道负载较高时，效率会显著下降。
  - ✧ **适用环境**：ALOHA 协议适合低负载和较为简单的网络环境，如卫星通信中的早期实现。

1.1.3.6.4.2 CSMA 载波监听多路访问

CSMA (Carrier Sense Multiple Access) 是一种允许多个设备共享信道的协议。它要求设备在发送数据之前先监听信道，确保信道空闲。若信道忙，设备会等待，直到信道变空再发送。CSMA 协议通过避免和检测碰撞来提高信道的利用率。

- **CSMA 的基本工作原理：**
  1. **监听信道**：设备在准备发送数据前，首先监听信道是否空闲。
  2. **发送数据**：
     - ✧ 如果信道空闲，设备发送数据。
     - ✧ 如果信道忙，设备等待，直到信道变空。
  3. **碰撞**：
     - ✧ 如果多个设备同时发送数据，会发生碰撞，导致信号干扰。此时设备需要重试。

- **CSMA 变体：**
  1. **CSMA/CD (Collision Detection)**：
     - **原理**：设备发送数据后继续监听信道，若发生碰撞，立即停止发送并发送冲突信号。
     - **适用场景**：有线网络（如以太网）。
     - **优点**：实时检测和停止碰撞，降低数据丢失。
     - **缺点**：无法在无线网络中使用，因为碰撞检测困难。
  2. **CSMA/CA (Collision Avoidance)**：
     - **原理**：设备通过发送 RTS/CTS 消息来避免碰撞，确保信道空闲后再发送数据。
     - **适用场景**：无线局域网（如 Wi-Fi）。
     - **优点**：减少碰撞概率，提高效率。
     - **缺点**：增加 RTS/CTS 的开销，适用于流量较低的网络。
  3. **持续与非持续监听**：
     - **持续监听**：设备在发送数据时持续监听信道，实时停止发送。
       - **优点**：实时性好，避免资源浪费。
       - **缺点**：增加硬件复杂性和能耗。
     - **非持续监听**：设备仅在准备发送数据前检查信道，发送后不再监听。
       - **优点**：实现简单，减少硬件需求。
       - **缺点**：无法实时检测碰撞，增加碰撞概率。

1.1.3.6.5 VLAN

VLAN（Virtual Local Area Network，虚拟局域网） 是一种网络分割技术，用于将一个物理局域网（LAN）划分为多个逻辑上的子网，以提高网络的灵活性、安全性和管理性。VLAN 的工作原理是在数据链路层（尤其是在 MAC 层）通过在以太网帧中嵌入 VLAN 标签来标识帧的所属虚拟网络。

- **VLAN 的工作原理**
  - **VLAN ID**：每个 VLAN 有一个唯一的 ID（1-4095），用于标识帧所属的 VLAN。
  - **VLAN 标签**：在以太网帧头部插入 4 字节的 VLAN 标签，标识该帧的 VLAN ID，交换机根据标签转发数据。
  - **交换机处理**：交换机根据端口配置的 VLAN ID 或通过标签来处理帧的转发。
- **VLAN 类型**
  - **静态 VLAN**：手动配置交换机端口为特定 VLAN，简单易管理，但设备移动时需重新配置。
  - **动态 VLAN**：根据设备的 MAC 地址或其他特征自动分配 VLAN，灵活性高，但配置复杂。

1.1.3.7 链路层协议

1.1.3.7.1 以太网协议

- **作用：**以太网协议是广泛使用的局域网协议，主要用于数据链路层，通过 MAC 地址进

行设备间的通信。

- **工作原理**：数据以帧的形式传输，每个帧包括目标 MAC 地址、源 MAC 地址、数据和 CRC 校验。

- **特点：**
  - ✧ 使用 CSMA/CD（碰撞检测）机制，允许设备共享同一信道。
  - ✧ 广泛应用于局域网（LAN）和以太网交换机中。

1.1.3.7.2 ARP 协议（Address Resolution Protocol）

- **作用：**ARP 用于将网络层的 IP 地址转换为数据链路层的 MAC 地址。

- **工作原理：**发送 ARP 请求广播到网络中，询问目标 IP 地址对应的 MAC 地址，接收方返回 ARP 响应，提供 MAC 地址。

- **应用场景：**用于以太网等基于 IP 的网络通信，使设备能够找到彼此的物理地址。

1.1.3.7.3 PPP 协议（Point-to-Point Protocol）

- **作用：**PPP 协议用于点对点连接，通常用于拨号、DSL 等通信中，提供数据帧传输。

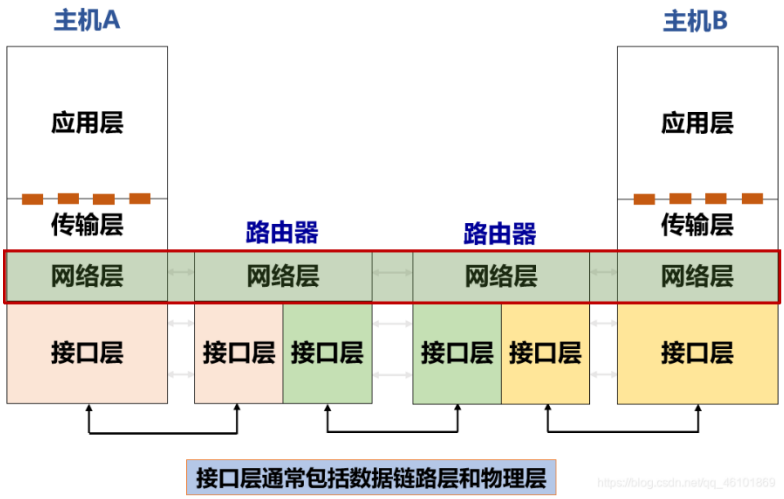
- **工作原理：**PPP 通过封装数据并增加帧头部和尾部进行传输，支持链路质量监测、身份验证和多协议支持。

- **特点：**
  - ✧ 支持 LCP（链路控制协议）进行链路建立和管理。
  - ✧ 支持 NCP（网络控制协议）用于不同网络层协议的支持。

1.1.4 网络层（Internet Layer）

1.1.4.1 网络层概述

网络层（第三层）是 TCP/IP 模型中的一层，主要负责数据包在不同网络间的传输与路由选择。它的核心任务是通过 IP 地址实现数据包的寻址与转发，使得不同网络中的设备能够相互通信。



1.1.4.1.1 网络层的职责

网络层的职责主要是确保数据能够在不同网络之间进行传输和寻址。
最关键的职责是路由和转发：

1. **路由选择**：决定数据包从源到目的地的最佳路径。
2. **数据包转发**：将数据包从一个网络传输到另一个网络。
3. 逻辑地址分配：使用 IP 地址标识设备，进行寻址。
4. 分段和重组：将大数据包分段，传输后再重组为原始数据。

1.1.4.1.2 网络层协议

网络层的主要协议包括：

1. **IP（Internet Protocol）**：负责数据包的寻址和路由，分为 IPv4 和 IPv6。
2. **ICMP（Internet Control Message Protocol）**：用于错误报告和网络诊断，如 ping 命令。
3. **ARP（Address Resolution Protocol）**：通过 IP 地址获取设备的物理 MAC 地址。
4. **DHCP（Dynamic Host Configuration Protocol）**：动态分配 IP 地址和其他网络配置信息。

1.1.4.2 IP（v4）协议

IPv4（Internet Protocol version 4） 是目前互联网最常用的网络层协议，主要用于数据在不同主机之间的寻址和传输。它定义了如何将数据包从源主机传送到目标主机，并确保在多个网络之间路由数据。

1.1.4.2.1 基本概念

- **IP 协议**：IPv4 是一种无连接的、不可靠的协议，主要负责数据包的寻址和路由。
- **地址长度**：IPv4 地址由 32 位二进制数构成，通常以点分十进制表示（例如：192.168.1.1）。
- **IP 地址**：IPv4 使用 32 位 IP 地址，能够支持大约 42 亿个唯一地址。
- **无连接**：IPv4 在数据传输过程中不提供建立和终止连接的过程，它只是简单地将数据包从源发送到目的地。
- **不可靠性**：IP 协议不保证数据包的交付，它并不提供重传机制。如果数据包丢失，应用

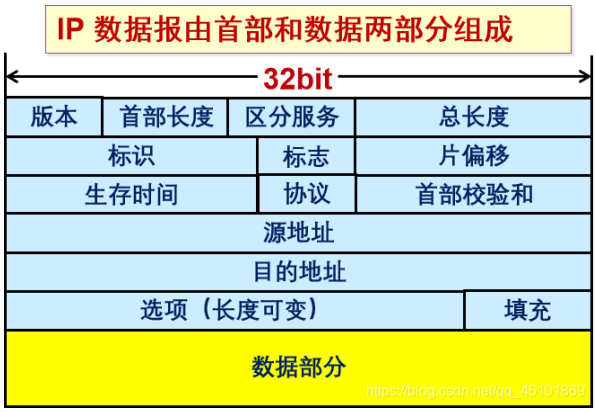
层需要依赖其他协议（如 TCP）来提供可靠传输。

IP 协议的主要作用

- ✧ 寻址和分片
- ✧ 提供设备间的寻址机制，并在网络中传输数据包，确保数据能够从源端正确地路由到目标端。

1.1.4.2.2 IP 数据报结构

IPv4 数据报是数据包在网络上传输的基本单位，包含了需要传送的源和目标地址以及其他一些控制信息。



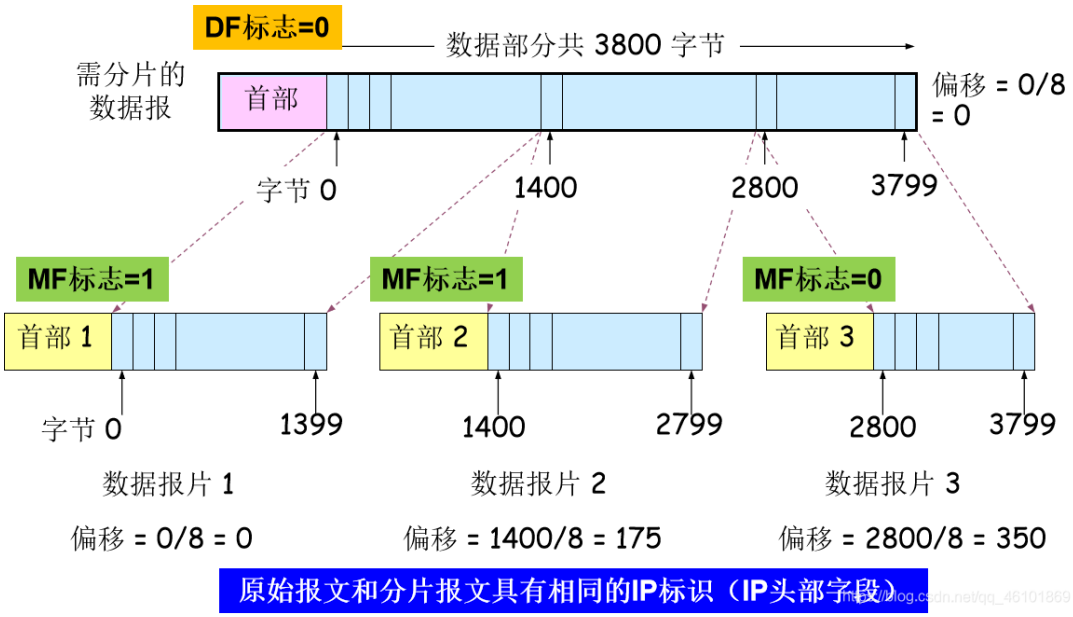
1.1.4.2.3 数据报分片

1.1.4.2.3.1 分片

IP 协议对数据报进行分片处理，以适应不同网络的最大传输单元（MTU）。如果数据报的大小超过了网络链路的 MTU，IP 层会将数据报分割成多个较小的片段。

- **分片策略**
  - ✧ 允许途中分片：根据下一跳链路的 MTU 实施分片
  - ✧ 不允许途中分片：发出的数据报长度小于路径 MTU（路径 MTU 发现机制）

- 分片的工作原理:
  - ✧ 每个分片会带上原始数据报的标识符，并指明其在数据报中的位置。
  - ✧ 每个分片都包含头部和数据部分，头部字段与原数据报一致，数据部分包含数据的一部分。
  - ✧ 在目标主机接收到所有分片后，IP 层会根据标识符和片偏移重新组合成原始数据报。



1.1.4.2.3.2 重组

分片后的数据包会在目标主机进行重组，通过标识符和片偏移字段来将不同的片段恢复成完整的数据报。

- **重组策略**
  - ✧ 途中重组，实施难度太大
  - ✧ 目的端重组（互联网采用的策略）
  - ✧ 重组所需信息：原始数据报编号、分片偏移量、是否收集所有分片

1.1.4.2.4 IPv4 与 IPv6 的区别

IPv6 是对 IPv4 的升级和扩展，旨在解决 IPv4 的不足，尤其是 **地址枯竭**、**路由效率**、**安全性**等问题。IPv6 通过引入新的特性和改进，提供了更大的地址空间、简化的报文头、更强的安全性等优势。

特性	IPv4	IPv6
地址长度	32位，约42亿个地址	128位，约340万亿万亿个地址
地址表示	点分十进制（如 `192.168.1.1`）	十六进制冒号分隔（如 `2001:0db8::`）
地址分配	需使用NAT，地址共享	每个设备拥有唯一全球地址
路由效率	路由表大，路由效率低	路由表小，路由效率高
分片	路由器分片	源主机分片，不通过路由器分片
安全性	无内建安全性，需要外部协议（如IPSec）	内建IPSec，提供加密与认证功能
广播	支持广播	不支持广播，使用多播与任播代替广播
NAT	需要使用NAT技术	无需NAT，每个设备唯一地址
自动配置	通过DHCP配置	支持无状态自动配置（SLAAC）

1.1.4.3 IP（v4）地址

IP 地址（Internet Protocol Address） 是用来标识网络中设备的唯一地址，确保数据在互联网上可以正确地从源设备传输到目的设备。IP 地址是网络通信中的基础要素，用于设备之间的定位与寻址。

1.1.4.3.1 基本概念

- IP 地址，网络上的每一台主机（或路由器）的每一个接口都会分配一个全球唯一的 32 位的标识符
- 将 IP 地址划分为固定的类，每一类都由两个字段组成
- 网络号相同的这块连续 IP 地址空间称为地址的前缀，或网络前缀
- IP 地址共分为 A、B、C、D、E 五类，A 类、B 类、C 类为单播地址
- IP 地址的书写采用点分十进制记法，其中每一段取值范围为 0 到 255

1.1.4.3.2 地址类型

A、B、C、D、E 类地址 是 IPv4 地址的分类方法，用于根据地址范围和使用场景区分不同类型的 IP 地址。每个类的地址在 **0.0.0.0 到 255.255.255.255** 之间的 IPv4 地址空间内有不同的分配规则。下面是各个类的简介：

- **A 类地址**
  - **地址范围**：0.0.0.0 到 127.255.255.255
  - **网络部分**：最高 8 位（第一个字节）
  - **主机部分**：剩余 24 位（后 3 个字节）
  - **用途**：A 类地址主要用于大型网络，提供最多的主机数量（约 1677 万个主机地址）。它通常分配给大型组织或互联网服务提供商。
  - **示例**：10.0.0.1（私有地址）
- **B 类地址**
  - **地址范围**：128.0.0.0 到 191.255.255.255
  - **网络部分**：前 16 位（前两个字节）
  - **主机部分**：剩余 16 位（后两个字节）
  - **用途**：B 类地址适用于中等规模的网络，提供较少的主机地址（约 65,000 个）。通常分配给中型企业。
  - **示例**：172.16.0.1（私有地址）
- **C 类地址**
  - **地址范围**：192.0.0.0 到 223.255.255.255
  - **网络部分**：前 24 位（前三个字节）
  - **主机部分**：剩余 8 位（最后一个字节）
  - **用途**：C 类地址用于小型网络，提供较少的主机地址（最多 254 个）。常见于小型局域网（LAN）。
  - **示例**：192.168.1.1（私有地址）
- **D 类地址**
  - **地址范围**：224.0.0.0 到 239.255.255.255
  - **用途**：D 类地址用于 **多播**（Multicast），即将数据发送到多个特定的设备（而非广播给所有设备）。该类地址并不用于常规的点对点通信。

- ✧ **示例**：224.0.0.1（用于多播组）

- **E 类地址**
  - ✧ **地址范围**：240.0.0.0 到 255.255.255.255
  - ✧ **用途**：E 类地址保留供未来使用或进行实验，通常不用于公共网络通信。
  - ✧ **示例**：240.0.0.1（实验或预留地址）

IP 特殊地址

地址	用途
全0网络地址	只在系统启动时有效，用于启动时临时通信，又叫主机地址
网络127.0.0.0	指本地节点(一般为127.0.0.1)，用于测试网卡及TCP/IP软件，这样浪费了1700万个地址
全0主机地址	用于指定网络本身，称之为网络地址或者网络号
全1主机地址	用于广播，也称定向广播，需要指定目标网络
0.0.0.0	指任意地址
255.255.255.255	用于本地广播，也称有限/受限广播，无须知道本地网络地址

1.1.4.3.3 子网划分

IP 子网划分（Subnetting）是将一个大的网络分割成多个较小的子网的过程，它有助于提高网络的管理效率、资源的有效利用和安全性。子网划分通常基于子网掩码（Subnet Mask）来实现，目的是为网络设备分配合理的 IP 地址，避免浪费。

1.1.4.3.3.1 子网掩码（Subnet Mask）

子网掩码是一种帮助区分网络地址和主机地址的工具。它与 IP 地址一起工作，告诉路由器如何在 IP 地址中分离出网络部分和主机部分。

- ✧ 格式：子网掩码的格式与 IP 地址相同，通常表示为四个十进制数（如：255.255.255.0）。
- ✧ 原理：子网掩码中的 1 位表示网络部分，0 位表示主机部分。例如，255.255.255.0 表示前 24 位是网络部分，后 8 位是主机部分。

1.1.4.3.3.2 子网划分的基本步骤

子网划分的主要目标是为网络设备提供合适的 IP 地址，并且避免浪费。

1. **步骤一：确定子网掩码**
   - ✧ 确定需要的子网数量：首先确定你需要多少个子网。比如你想把一个 A 类网络划分为 10 个子网，那么你就需要从原网络的地址中借用一定数量的主机位来形成新的子网。
   - ✧ 确定每个子网中的主机数量：每个子网中需要容纳多少设备，确定可以为每个子网分配的主机地址数。
2. **步骤二：选择子网掩码**
   - ✧ 子网掩码的选择依据是子网数目和每个子网的主机数目。在 IPv4 中，子网掩码是一个 32 位的二进制数，通过增加网络部分的位数（即借用主机位）来划分子网。

3. **步骤三：计算可用的子网地址**
   - ✧ 在确定了子网掩码后，你可以计算出每个子网的网络地址和广播地址。
4. **步骤四：分配 IP 地址**
   - ✧ 根据划分的子网地址，分配给每个子网内的设备适当的 IP 地址。

1.1.4.3.3.3 子网划分的常见应用

- A、B、C 类地址的划分：在传统的 IP 地址分类中，A、B、C 类的地址有不同的默认子网掩码：
  - ✧ A 类：255.0.0.0（默认子网掩码）
  - ✧ B 类：255.255.0.0（默认子网掩码）
  - ✧ C 类：255.255.255.0（默认子网掩码）

根据实际需求，可以进一步调整子网掩码，划分更多或更少的子网。

- CIDR（无类域间路由选择）：
  - ◆ CIDR 的基本概念
    - ✧ CIDR 表示法：IP 地址后跟**斜杠和前缀长度**，例如 192.168.1.0/24，其中 /24 表示网络部分占 24 位。
    - ✧ 灵活的网络划分：CIDR 不依赖固定的 A/B/C 类划分，允许精确分配所需的 IP 地址数量，避免了传统方法中的地址浪费。
  - ◆ CIDR 的优势
    - ✧ 更高效的地址分配：允许任意大小的网络划分，避免浪费。
    - ✧ 减少路由表规模：通过**路由聚合**（将多个地址块合并），减少路由器需要处理的路由条目，提高网络性能
  - ◆ CIDR 与传统 IP 分类的区别
    - ✧ 传统的 A/B/C 类划分固定，导致 IP 地址分配不灵活。
    - ✧ CIDR 通过前缀长度（例如 /24）来表示网络规模，灵活分配地址。

1.1.4.4 DHCP 协议

DHCP（动态主机配置协议，Dynamic Host Configuration Protocol） 是一种网络协议，主要用于自动为网络中的设备（如计算机、手机、打印机等）分配 IP 地址和其他网络配置信息。通过 DHCP，设备无需手动配置 IP 地址、子网掩码、默认网关等信息，极大地简化了网络的管理和配置过程。

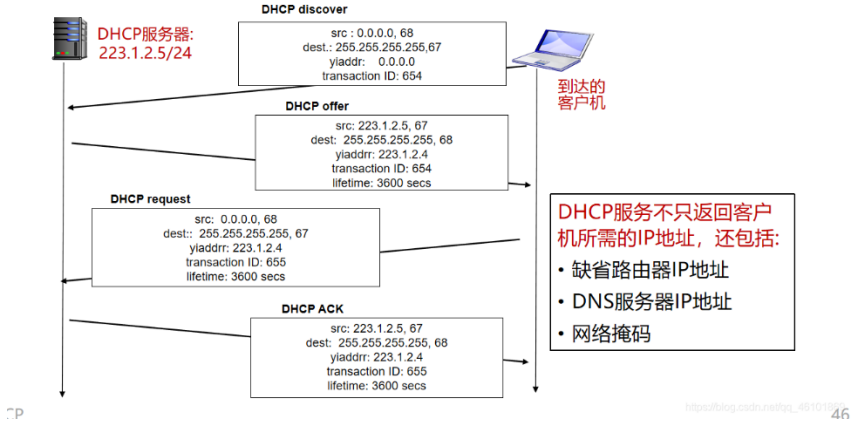
1.1.4.4.1 DHCP 的主要功能

- ✧ **动态 IP 地址分配**：DHCP 可以根据网络需求动态分配 IP 地址。每次设备连接到网络时，DHCP 服务器都会自动分配一个可用的 IP 地址。
- ✧ **分配网络配置信息**：除了 IP 地址，DHCP 还可以为设备提供其他配置信息，如子网掩码、默认网关、DNS 服务器等。
- ✧ **自动管理 IP 地址池**：DHCP 服务器会维护一个 IP 地址池，确保所有设备都能分配到唯一且有效的 IP 地址，避免 IP 地址冲突。

1.1.4.4.2 DHCP 工作流程

DHCP 的工作流程通常分为四个阶段，简称 **DORA**：

1. **Discover（发现）**：设备（客户端）向网络广播 DHCP 发现请求，寻找 DHCP 服务器。请求包含设备的 MAC 地址等信息。
2. **Offer（提供）**：DHCP 服务器接收到发现请求后，会回复一个 DHCP 提供的可用 IP 地址及其他配置信息（如子网掩码、默认网关等）。
3. **Request（请求）**：客户端从多个 DHCP 服务器的响应中选择一个，并向其发送请求，确认接受该 IP 地址及配置信息。
4. **Acknowledge（确认）**：DHCP 服务器接收到请求后，确认并分配 IP 地址，客户端则可以使用这个 IP 地址进行网络通信。



1.1.4.4.3 DHCP 的报文格式

DHCP 报文采用 **UDP 协议**，**C/S 架构**；客户端和服务器之间通过 UDP 端口 **67** 和 **68** 进行通信，使用特定格式的报文进行交换，常见的有以下几种：

- **DHCP Discover**：客户端向网络广播，寻找 DHCP 服务器。
- **DHCP Offer**：DHCP 服务器响应客户端的 Discover 请求，提供 IP 地址等配置信息。
- **DHCP Request**：客户端从多个 Offer 中选择一个，向服务器确认请求。
- **DHCP Acknowledge**：DHCP 服务器确认客户端请求并分配 IP 地址。

1.1.4.4.4 DHCP 的 IP 地址分配方式

DHCP 协议有不同的 IP 地址分配方式，常见的有：

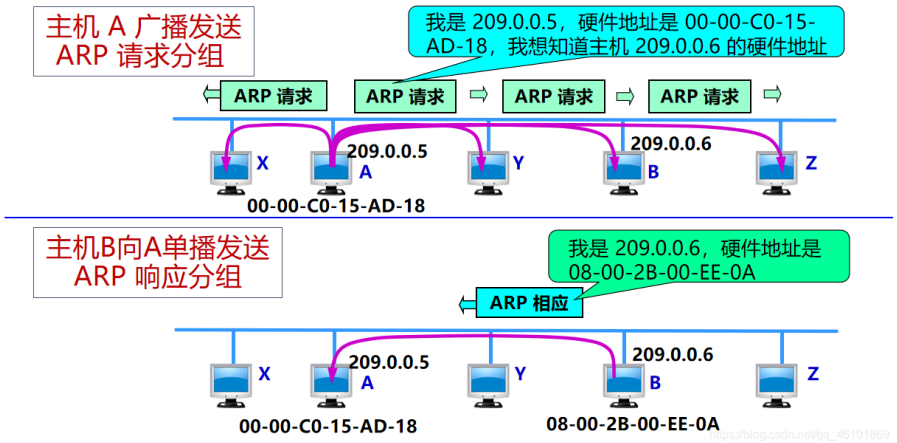
1. **自动分配（Automatic Allocation）**：DHCP 服务器为客户端永久分配一个 IP 地址，直到客户端离开网络或服务器关闭。适用于需要固定 IP 地址的设备（如服务器）。
2. **动态分配（Dynamic Allocation）**：DHCP 服务器从一个 IP 地址池中动态分配一个 IP 地址。分配的 IP 地址有一个租期（Lease Time），租期结束后，客户端必须重新申请一个新的 IP 地址。适用于大多数普通终端设备（如计算机、手机等）。
3. **手动分配（Manual Allocation）**：网络管理员为特定设备手动指定 IP 地址，设备会始终使用这个 IP 地址。通常用于需要固定 IP 的设备。

1.1.4.5 ARP 协议

ARP（地址解析协议）用于将 IP 地址映射到 **MAC 地址**。在局域网内，数据帧的传输依赖于 MAC 地址，而路由器和主机通常通过 IP 地址进行通信。因此，ARP 协议帮助设备找到目标设备的物理地址，确保数据能在网络内正确传输。

1.1.4.5.1 工作原理

1. **ARP 请求**：
   - ✧ 设备 A 想和设备 B 通信，它知道 B 的 **IP 地址**，但不知道 **MAC 地址**。
   - ✧ 设备 A 向局域网广播 ARP 请求："谁拥有 IP 地址 X.X.X.X？请回复你的 MAC 地址。"这个请求是广播的，会发给局域网内的所有设备。
2. **ARP 响应**：
   - ✧ 局域网中每个设备都会收到 ARP 请求，只有目标设备 B 会检查 IP 地址是否匹配自己的。如果匹配，B 会向 A 发送 ARP 响应，包含 B 的 **MAC 地址**。
3. **ARP 缓存**：
   - ✧ 设备 A 收到 B 的 MAC 地址后，会将这个 **IP-MAC 映射关系**缓存下来，存储在 ARP 缓存表中。这样，下次 A 再需要与 B 通信时，就可以直接使用缓存的 MAC 地址，避免再次发送 ARP 请求。



1.1.4.5.2 主要功能：

- ✧ **IP 到 MAC 地址转换：**ARP 确保设备能够找到目标设备的 MAC 地址，从而保证数据能在局域网内正确传输。
- ✧ **自动化地址解析：**ARP 自动执行 IP 和 MAC 地址的转换，无需手动配置，简化了设备间的通信过程。
- ✧ **缓存机制：**设备将 IP-MAC 映射存入 ARP 缓存（ARP 表），避免重复的 ARP 请求，提高效率。

1.1.4.6 NAT（Network Address Translation）

NAT（网络地址转换）是一种网络协议，它用于在局域网（LAN）和广域网（WAN）之间转换 IP 地址。NAT 最常见的应用是在家庭或公司网络中，通过一个公共的公网 IP 地址来共享

多个设备的**私有 IP 地址**，从而节省 IP 地址并提供一定程度的安全性

1.1.4.6.1 工作原理机制

NAT 的工作原理可以分为两个主要部分：**地址转换**和**端口映射（PAT）**。

1. **地址转换（Address Translation）**
   NAT 的核心功能是将私有 IP 地址转换为公网 IP 地址，或反向转换。
   - ✧ 内网设备发起请求时，NAT 设备将私有 IP 地址转换为公网 IP 地址，并记录转换信息。
   - ✧ 返回时，NAT 通过映射表将外部数据包转发到对应的内网设备。
2. **端口映射（PAT，Port Address Translation）**
   为每个内网设备的请求分配不同的端口号，多个内网设备共享同一个公网 IP 地址。NAT 设备根据端口号来区分各个连接。
   - ✧ 内网设备**发起连接**时，NAT 会记录下内网设备的私有 IP 地址及其源端口，并为该请求分配一个唯一的端口号。
   - ✧ 在**返回数据**时，NAT 通过端口号来区分不同内网设备的请求，确保数据能被正确转发到相应的内网设备。
3. **NAT 表（Translation Table）**
   NAT 维护一个转换表，记录内网设备的私有 IP、端口及其对应的公网 IP 和端口，从而确保数据准确转发。

1.1.4.6.2 常见的 NAT 类型

- **静态 NAT（Static NAT）**
  静态 NAT 是指每个内网设备的私有 IP 地址都会与一个固定的公网 IP 地址进行一一映射。
  - ✧ **优点**：固定的映射关系方便外部设备访问。
  - ✧ **缺点**：公网 IP 资源消耗较大，适用场景有限。

- **动态 NAT（Dynamic NAT）**
  动态 NAT 将内网设备的私有 IP 地址映射到一组公网 IP 地址中的一个。这些公网 IP 地址是有限的，并且是动态分配的。内网设备发起连接时，NAT 设备从这组公网 IP 中选择一个可用 IP 进行映射。
  - ✧ **优点**：比静态 NAT 更节省公网 IP 地址。
  - ✧ **缺点**：公网 IP 的可用性受限，可能导致内网设备无法访问外部。

- **端口地址转换（PAT，Port Address Translation）**
  端口地址转换是最常见的 NAT 形式，它允许多个内网设备共享同一个公网 IP 地址。通过为每个连接分配不同的端口号，NAT 设备可以区分不同的连接请求。
  - ✧ **优点**：最大程度地节省公网 IP 地址，每个内网设备可以通过不同的端口共享一个公网 IP。
  - ✧ **缺点**：端口号有限，可能会遇到端口耗尽的情况。

1.1.4.6.3 NAT 的功能作用

1. **解决 IP 地址短缺**
   NAT 的最大作用之一就是缓解 IPv4 地址的短缺问题。通过 NAT，多个内网设备可以共享一个公网 IP，从而极大地节省了公网 IP 的使用。
2. **增强网络安全性**
   NAT 通过将内网的私有 IP 地址隐藏在公网 IP 之后，外部设备无法直接访问内网设备。这种地址隐藏技术可以有效提高内网的安全性，防止外部的攻击和未经授权的访问。
3. **网络地址隔离**
   NAT 将内网与外网分隔开来，提供了一个天然的安全屏障。内网中的设备无法直接与外网设备通信，所有通信必须通过 NAT 设备，从而确保了网络的隔离性。
4. **简化网络管理**
   NAT 使得内网的 IP 地址管理更加灵活，管理员可以使用私有 IP 地址进行内部网络规划，而无需考虑公网 IP 的分配和管理。此外，NAT 还可以简化 IP 地址变更的操作，内网设备的 IP 地址变更时不影响外部的访问。
5. **支持 IPv4 向 IPv6 过渡**
   NAT 还在 IPv4 和 IPv6 的过渡过程中发挥了作用。通过 NAT64 等技术，可以实现 IPv6 和 IPv4 网络的互通，支持 IPv6 过渡到 IPv4 的过程中。

1.1.4.7 ICMP 协议

ICMP 是网络层协议，用于在网络设备（如路由器、主机）之间传输控制消息。其主要功能是帮助网络设备报告错误、诊断网络问题、并进行通信状态反馈，其与 IP 协议紧密合作，IP 负责数据包的路由和传输，而 ICMP 则用于反馈路由过程中遇到的错误和问题。

1.1.4.7.1 ICMP 的核心功能（工作原理）

- 错误报告
  - ✧ 检测并反馈网络问题，如目的不可达（主机/端口/协议不可达）、数据包超时（TTL 耗尽）、分片需求（需分片但禁止分片）等。
  - ✧ 错误报文仅返回至源端，帮助定位问题（如路由失效或防火墙拦截）。

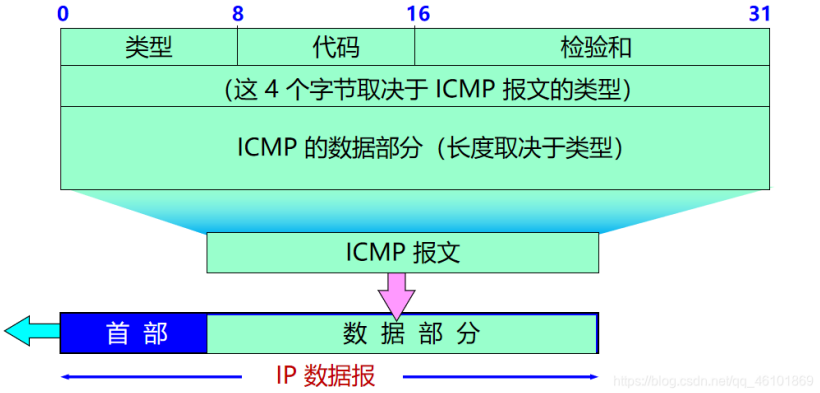
- 网络诊断
  - ✧ Ping 命令：基于 Echo Request/Reply（Type 8/0），测试主机可达性与延迟。
  - ✧ Traceroute：利用 TTL 递减触发 Time Exceeded（Type 11）报文，追踪路径并分析跳数。

- 路由优化与维护
  - ✧ 路由重定向（Type 5）：通知源端存在更优路径。
  - ✧ 路径 MTU 发现：通过 Fragmentation Needed（Type 3, Code 4）确定最大传输单元，避免分片

1.1.4.7.2 ICMP 报文结构类型

- **关键字段：**

Type（类型）：标识报文大类（如 0 为回显应答，3 为目的不可达）。
Code（代码）：细化错误原因（如 Type 3 下 Code 3 表示端口不可达）。
Checksum：校验数据完整性。



- **常见类型：**

Type	名称	用途
0	回显应答 (Echo Reply)	响应回显请求，测试主机是否可达（`ping` 的应答）。
3	目标不可达 (Destination Unreachable)	通知源主机目的地址不可达，如网络或主机不可达。
8	回显请求 (Echo Request)	请求目标主机响应，测试主机是否可达（`ping` 请求）。
11	超时 (Time Exceeded)	数据包的 TTL 超时，通常用于 `traceroute` 工具追踪路径。
5	重定向 (Redirect)	告知源主机使用不同的路由。
4	源抑制 (Source Quench)	通知源主机减少发送数据包的速度（很少使用）。

1.1.4.7.3 Ping 命令与 Traceroute 命令

基于 ICMP 协议，Ping 主要测试连通性，Traceroute 用于追踪路由路径。

1.1.4.7.3.1 Ping 命令

- **功能**：Ping 命令用于测试网络主机的连通性。它通过发送 ICMP 回显请求（Type 8）到目标主机，并等待 ICMP 回显应答（Type 0）来确认目标主机是否在线。
- **工作原理**：源主机发送回显请求，目标主机收到后返回回显应答。Ping 会计算请求和应答之间的往返时间（RTT）来评估延迟。
- **用途**：主要用于检查网络延迟、主机是否可达以及网络是否正常。

1.1.4.7.3.2 2. Traceroute 命令

- **功能**：Traceroute 命令用于追踪数据包经过的路由路径。它帮助分析数据包从源主机到目标主机经过的每一跳路由器。
- **工作原理**：Traceroute 通过逐步增加数据包的 TTL（Time to Live）值，发送 ICMP 超时消息（Type 11）。每经过一个路由器，TTL 会减 1，当 TTL 为 0 时，路由器发送 ICMP 超时响应。通过此过程，Traceroute 显示出每一跳的路由器及其响应时间。
- **用途**：帮助诊断网络路径、识别网络瓶颈或故障点。

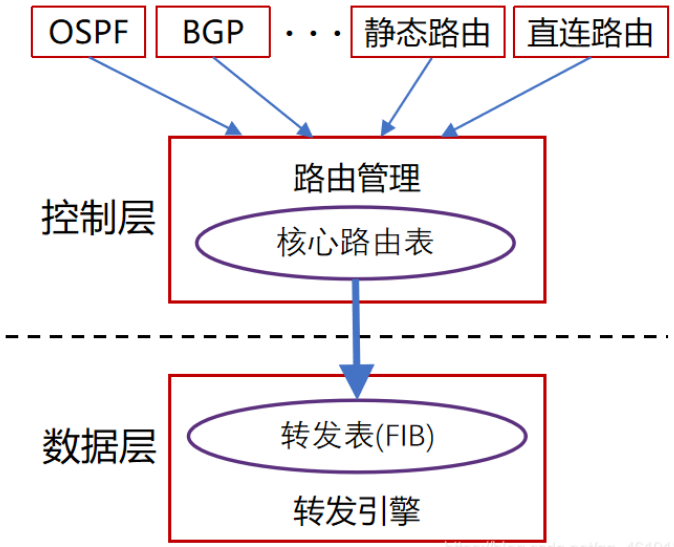
1.1.4.8 路由协议

路由协议是网络中确保数据能够从源头顺利到达目的地的关键技术。它们定义了网络设备之间如何交换信息，以选择最优路径来转发数据包。

1.1.4.8.1 路由器原理

路由器是互联网最主要的网络设备，包含 2 个核心功能

- ✧ 控制层：运行各种路由协议：BGP、OSPF、RIP，学习去往不同目的的转发路径：路由表
- ✧ 数据层：根据上述路由表，将收到的 IP 分组转发到正确的下一跳链路



1.1.4.8.1.1 路由表

路由表是网络设备用来存储路由信息的地方，通常包含：

- ✧ **目标网络**：目的地网络或子网。
- ✧ **下一跳**：到达目标网络的下一个路由器或设备的 IP 地址。
- ✧ **接口信息**：该路由条目对应的物理或虚拟接口。
- ✧ **度量值**：评估路由的优先级或“成本”，例如跳数、带宽、延迟等。 路由表在路由器内是动态更新的，当网络拓扑发生变化时，控制层会更新路由表。

1.1.4.8.1.2 控制层

控制层负责路由决策和路由表管理。它执行以下任务：

- ✧ **运行路由协议**：控制层通过 BGP、OSPF、RIP 等协议与其他路由器交换信息，决定最优路径。
- ✧ **生成路由表**：基于路由协议的计算，生成并更新路由表，记录数据包的最佳转发路径。
- ✧ **路径优化**：根据路由协议选择最短或最优的路径来提高网络效率。

1.1.4.8.1.3 数据层

数据层负责实际的数据转发工作。它的主要功能包括：

- ✧ **数据包转发**：根据目标 IP 地址查找路由表，确定下一跳路由器或目标设备。
- ✧ **高效转发**：数据层通常依赖硬件加速进行高效转发，减少延迟。

✧ **网络地址转换（NAT)**：执行 NAT 功能，允许多个设备共享一个公共 IP 地址。

1.1.4.8.1.4 SDN 数据层与控制层

在软件定义网络（SDN）中，数据层与控制层分离，带来更大的灵活性和可编程性：

✧ **集中控制**：控制层集中管理网络的路由决策，通过控制器发送转发规则给数据层设备。
✧ **简化数据层**：数据层专注于按控制层的指令转发数据包，避免复杂的路由计算。
✧ **灵活性**：SDN 允许动态调整网络配置和优化流量管理

1.1.4.8.1.5 路由类型

路由类型指的是网络中如何选择和配置路由路径的方式。根据路由路径的配置和更新方式，主要分为**静态路由**和**动态路由**两大类。

- **静态路由**
  - ✧ **定义**：由管理员手动配置，固定路径，不会自动更新。
  - ✧ **优点**：简单、资源消耗少、安全性高。
  - ✧ **缺点**：需要手动更新，无法适应网络拓扑变化。
  - ✧ **适用场景**：小型网络或网络拓扑不常变化的环境。
- **动态路由**
  - ✧ **定义**：使用路由协议（如 BGP、OSPF、RIP）自动更新路径，适应网络拓扑变化。
  - ✧ **优点**：自动调整路由，适合大型复杂网络。
  - ✧ **缺点**：配置复杂、资源消耗较大。
  - ✧ **适用场景**：中大型企业或互联网服务提供商网络。

1.1.4.8.2 路由算法

路由算法是用于确定网络中数据包传输路径的算法。根据不同的网络需求、规模和特点，路由算法可以采取不同的策略。路由算法的选择影响到网络的性能、稳定性和效率。

1.1.4.8.2.1 距离向量路由

距离向量路由算法是一种基于跳数的动态路由协议。每个路由器通过与邻居交换路由信息来更新自己的路由表，最终确定到达各目的地的最佳路径。

- **算法核心思想**
  - ✧ **核心思想是每个路由器依赖于其他路由器的路由信息来决定路径。**
  - ✧ 每个路由器维护一张路由表，表中记录了到达各个目的地的距离（通常是跳数）以及下一跳路由器。当路由器与邻居交换路由信息时，它根据邻居的路由表更新自己的路由表。路由器之间通过定期交换路由信息来实现路径更新。

- **算法过程**
  1. **初始化**：每个路由器维护一个路由表，包含到达每个目的地的距离（初始为无穷大）和下一跳。
  2. **信息交换**：每个路由器将自己的路由表广播给所有的邻居路由器。
  3. **更新路由表：**收到邻居的路由信息后，路由器将根据距离更新自己的路由表。如果新路径更短，就更新路由表。

4. **重复交换**：不断地交换路由信息，直到网络稳定，所有路由器的路由表收敛。

- **特殊情况**
  - ✧ **计数到无穷（Count to Infinity）**：当网络中出现环路时，距离向量算法可能陷入无限循环，导致路由表中的某些路径的跳数不断增大，最终无法收敛。常见的解决方法是**毒性逆转（Split Horizon）**和**水平分割**。

- **优点**
  - ✧ 简单易懂，便于实现。
  - ✧ 配置和维护工作相对较少。
  - ✧ 对小型网络有效，开销较小。

- **应用**
  - ✧ **RIP（Routing Information Protocol）**：RIP 是最典型的使用距离向量算法的协议，它通过定期交换路由信息来更新路由表，并且根据跳数（最大 15 跳）来选择最短路径。

1.1.4.8.2.2 链路状态路由

链路状态路由算法是一种基于拓扑的动态路由协议。每个路由器与网络中的所有其他路由器交换链路状态信息，通过计算网络拓扑图来选择最佳路径。该算法的

- **算法核心思想**
  - ✧ **关键思想是每个路由器知道整个网络的拓扑结构，并利用此信息计算最短路径。**
  - ✧ 每个路由器通过向所有路由器广播其链路状态信息（例如连接的网络、链路的开销等），从而让每个路由器了解整个网络的拓扑结构。路由器使用 Dijkstra 算法基于拓扑图计算到达各目的地的最短路径。

- **算法过程**
  1. **广播链路状态：**每个路由器将其链路状态（包括自身的链路开销和状态）广播给网络中的所有其他路由器。
  2. **建立拓扑图：**每个路由器通过接收到的链路状态信息构建完整的网络拓扑图。
  3. **计算最短路径**：每个路由器使用 Dijkstra 算法（最短路径算法）计算到其他路由器的最短路径，并更新自己的路由表。
  4. **链路状态更新**：当网络拓扑发生变化时，路由器会重新广播链路状态信息并重新计算路径。

- **特殊情况**
  - ✧ **路由环路：**由于链路状态路由算法每个路由器都有完整的拓扑信息，所以可以避免传统的距离向量路由算法中常见的路由环路问题。
  - ✧ **网络拓扑变化：**每当网络拓扑发生变化时，路由器会更新链路状态，确保每个路由器都能重新计算最优路径。

- **优点**
  - ✧ 收敛速度快，能够迅速适应网络变化。

- ✧ 避免了“计数到无穷”的问题。
- ✧ 提供了更精确的路径选择，适用于大规模复杂网络。

- **应用**
  - ✧ **OSPF（Open Shortest Path First）**：OSPF 是广泛使用的链路状态路由协议，支持大规模网络，能够高效地计算最短路径。
  - ✧ **IS-IS（Intermediate System to Intermediate System）**：IS-IS 协议也基于链路状态算法，常用于运营商网络。

1.1.4.8.2.3 路径向量路由

路径向量路由算法是距离向量算法的扩展，适用于跨自治系统（AS）进行路由。每个路由器维护路径向量，其中包含到目的地的路径信息和所经过的自治系统（AS）。这种方法避免了传统距离向量路由中的路由环路问题。

- **算法核心思想**
  - ✧ 路由器维护一张路径向量表，其中记录到达各目的地的路径信息。每个路径向量不仅记录跳数，还包含路径中经过的自治系统（AS）。路径信息被逐步交换，以确保路由表的正确性和避免环路。

- **算法过程**
  1. **维护路径向量**：每个路由器将目的地的路径信息和自治系统 ID 添加到路径向量中。
  2. **交换路径信息**：路由器向邻居路由器发送路径向量信息。
  3. **更新路由表**：每个路由器根据收到的路径向量更新自己的路由表，选择最优路径，并避免环路。
  4. **避免环路**：通过路径中包含的自治系统信息，路径向量能够避免形成环路。

- **优点**
  - ✧ 避免了传统距离向量路由中的环路问题。
  - ✧ 能够有效处理跨自治系统的路由。
  - ✧ 路由稳定性高，适用于跨域的复杂网络。

- **应用**
  - ✧ BGP（Border Gateway Protocol）：BGP 是最典型的路径向量协议，广泛用于互联网中的自治系统之间的路由选择。

1.1.4.8.2.4 层次路由

层次路由算法将大规模网络划分为多个层次或区域，以减小路由表的大小，并提高路由效率。每个区域内的路由器使用内部路由协议，区域之间的路由则由区域边界路由器（ABR）处理。

- **算法核心思想**
  - ✧ 通过将网络划分为多个区域或层次，减少每个路由器需要维护的路由表的大小。区域之间的路由信息通过区域边界路由器传递，避免了大规模网络中每个路由器都必须了解完整网络拓扑的问题。

- **算法过程**
  1. **网络划分**：将网络划分为多个区域，每个区域内使用内部路由协议（如 OSPF）。
  2. **区域边界路由器（ABR）**：区域边界路由器负责在区域之间转发路由信息。
  3. **跨区域路由**：不同区域之间的路由信息通过区域边界路由器传播。
  4. **区域内部路由**：每个区域内的路由由区域内的路由器处理。

- **特殊情况**
  - ✧ 区域划分不当：如果区域划分不合理，可能导致区域内的路由表过大，影响路由效率。
  - ✧ 区域边界路由器的负担：区域边界路由器需要处理跨区域的路由信息，可能成为网络的瓶颈。

- **优点**
  - ✧ 减小路由表的规模，提高路由效率。
  - ✧ 增强网络的可扩展性，适用于大型网络。
  - ✧ 通过区域化管理，简化了网络的管理和维护。

- 与链路状态路由关系
  - ✧ **链路状态路由在层次路由中的应用**：链路状态路由需要每个路由器拥有整个网络拓扑，随着规模扩大会增加路由表和计算复杂度。通过将网络划分为区域，每个区域内部使用链路状态协议（如 OSPF），区域间通过区域边界路由器交换信息，缓解这一问题。
  - ✧ **层次路由提高可扩展性：**层次路由减少了每个路由器的计算和存储负担。通过划分区域，链路状态协议仍在区域内运行，跨区域的路由由区域边界路由器处理，从而提高了大规模网络的效率。
  - ✧ **层次路由与链路状态路由的协作**：在 OSPF 中，网络划分为区域，每个区域内部使用链路状态路由，区域间通过区域边界路由器交换信息，减轻路由器负担。

- **应用**
  - ✧ OSPF（Open Shortest Path First）：OSPF 支持层次化路由，网络可被划分为多个区域，通过区域边界路由器进行管理。
  - ✧ IS-IS（Intermediate System to Intermediate System）：IS-IS 协议也采用层次化结构，通过区域划分来优化网络的路由。

1.1.4.8.2.5 广播路由

广播路由算法通过广播消息来发现网络中所有节点并建立路由。每个路由器向所有网络节点广播路由信息，直到所有路由器都获取到网络拓扑信息。

- **算法核心思想**
  - ✧ 路由器通过广播信息到所有节点来发现网络中的所有设备，并建立路由表。广播消息通常包含路由请求或网络拓扑信息。

- **算法过程**

1. **广播消息**：每个路由器将路由信息广播给网络中的所有节点。

2. **响应和更新**：网络中的所有节点接收到广播消息后，会更新自己的路由信息，并可能通过广播响应给发起请求的路由器。
3. **网络收敛**：通过多次广播，网络中所有的路由器都获取到路由信息，最终形成完整的路由表。

- **特殊情况**
  - ✧ **广播风暴**：在大型网络中，频繁的广播可能会导致广播风暴，严重影响网络性能。
  - ✧ **网络规模限制**：广播路由仅适用于小规模网络或局域网，因广播量过大会导致性能瓶颈。

- **优点**
  - ✧ 实现简单，适用于小型网络或局域网。
  - ✧ 不需要复杂的配置，适合动态环境。

- **应用**
  - ✧ ARP（Address Resolution Protocol）：ARP 使用广播消息来查找目标设备的 MAC 地址。
  - ✧ RARP（Reverse Address Resolution Protocol）：RARP 也通过广播来查询设备的 IP 地址。

1.1.4.8.3 RIP

RIP (Routing Information Protocol，路由信息协议）是一种基于距离向量的**内部网关协议（IGP)**，广泛用于小型和中型网络中。其主要特点是通过交换路由表来传递网络中的路由信息。RIP 使用跳数作为衡量路径距离的标准，跳数上限为 15 跳，超过 15 跳的网络被认为是不可达的。

- **RIP 的工作原理：**

1. **距离向量算法**：RIP 基于**距离向量算法**，每个路由器维护一张包含目标网络和跳数（距离）的路由表。每隔一定时间，路由器会将其路由表发送到相邻的路由器。
2. **更新周期**：RIP 默认的路由更新周期是 30 秒。路由器定期广播其路由表，这种定时更新可能会导致网络收敛时间较长。
3. **路由度量**：RIP 的度量是跳数，跳数的最大值为 15 跳，因此它不适用于大规模网络。
4. **路由收敛**：RIP 的收敛时间较长，意味着当网络拓扑发生变化时，可能会在网络中出现不一致的路由信息，直到网络收敛。

- **版本**：
  - ✧ RIP v1：基于类的 IP 地址，它不支持 CIDR（无类域间路由），只能传递默认的子网掩码。
  - ✧ RIP v2：支持 CIDR，并引入了身份验证机制来增强安全性。
  - ✧ RIPng：是 RIP v2 的扩展，支持 IPv6 地址。

- **优缺点：**
  - ✧ 优点：实现简单，适用于小型网络。
  - ✧ 缺点：收敛速度慢，最大跳数为 15，无法支持大规模网络。

1.1.4.8.4 OSPF

OSPF（开放最短路径优先）是一种基于链路状态的**内部网关协议（IGP）**，它与 RIP 不同，使用更先进的算法（Dijkstra 算法）计算路由，能够提供更快的收敛和更高效的路由管理。OSPF 广泛用于中型和大型企业网络中，特别是需要大规模、复杂路由的场景。

- **OSPF 的工作原理：**
  - **链路状态算法**：OSPF 每个路由器会创建链路状态数据库（LSDB），其中包含了网络中各个链路的信息。通过交换 LSA（链路状态广告）来实现路由信息的传递。
  - **区域划分**：OSPF 支持将网络分为多个区域（Area），每个区域内部的路由信息由区域内的路由器共享，跨区域的路由信息由区域间的路由器共享。这种分区设计使得 OSPF 能够更好地扩展到大规模网络。
  - **Dijkstra 算法**：OSPF 使用 Dijkstra 算法计算最短路径树（SPT），从而选择出最佳路径。
  - **状态类型**：OSPF 使用多种状态类型，例如，邻接状态、交换状态、同步状态等，以确保链路状态的同步和一致性。

- **优缺点：**
  - **优点**：
    - 高效，收敛速度快，适用于大规模网络。
    - 支持复杂的路由设计（如区域划分）。
    - 支持无类路由（CIDR）。
  - **缺点**：
    - 配置较复杂，适用于中大型网络。
    - 占用更多的网络带宽和路由器资源进行链路状态信息的广播和同步。

1.1.4.8.5 BGP

BGP（边界网关协议）是一种外部网关协议（EGP），用于不同自治系统（AS）之间的路由信息交换。它是互联网上最常用的路由协议，特别用于大规模、跨域的网络环境，如互联网中的路由选择。

- **BGP 的工作原理：**
  - **路径向量协议**：BGP 是路径向量协议，它通过交换包含 AS 路径（AS Path）的路由信息来选择最佳路由。每个 BGP 路由器会向邻居广播路由信息，包括该路径经过的 AS 序列。
  - **自治系统**：BGP 用于不同自治系统之间的路由选择，每个自治系统由一个唯一的 AS 编号标识。BGP 的主要任务是确定通过哪个 AS 达到目标网络。
  - **路由选择标准**：BGP 不单单使用跳数来选择路径，它使用一系列标准（例如 AS 路径长度、路由优先级、路由的可达性等）来选择最优路径。
  - **BGP 类型**：
    - **EBGP（外部 BGP）**：用于不同 AS 之间的路由交换。
    - **IBGP（内部 BGP）**：用于同一 AS 内部的路由信息交换。
- **优缺点：**

- ✧ **优点**：
  - 能够处理大规模网络路由，特别适合互联网环境。
  - 提供灵活的路由控制机制，如路由过滤、策略路由等。
  - 具有高扩展性，能够支持复杂的路由策略。
- ✧ **缺点**：
  - 配置复杂，需要专业知识和经验。
  - 收敛速度较慢，特别是在大规模网络中。

1.1.4.9 VPN（虚拟专用网络）

1.1.4.9.1 VPN 的基本概念

- VPN（虚拟专用网络）是一种通过公共网络（如互联网）建立专用网络的技术，使得用户能够在不安全的公共网络上进行安全的数据传输。
- VPN 的目的是提供数据的隐私性、完整性、身份认证和加密保护，常用于远程办公、跨地区分支机构的安全通信等场景。

1.1.4.9.2 核心技术

1.1.4.9.2.1 隧道技术（Tunneling）

隧道技术是 VPN 的基础，它通过将数据包封装在另一个数据包中，使得数据能够通过公网安全地传输。VPN 隧道可以在以下两种模式下运行：

- ✧ **点对点隧道（P2P）**：点对点之间通过 VPN 隧道进行通信，通常适用于两台设备之间的连接。
- ✧ **站点到站点隧道**：通过 VPN 连接多个不同地点的网络，通常用于企业内部的不同分支机构之间的通信。

1.1.4.9.2.2 加密与认证

为了确保数据的安全性，VPN 使用加密技术来保护数据的机密性，并通过身份认证来确保只有合法用户才能访问 VPN。

- ✧ **加密算法**：常用的加密算法包括 AES（高级加密标准）、3DES 等，用于对数据进行加密。
- ✧ **身份认证协议**：常见的身份认证协议包括 PAP、CHAP、EAP 等，用于验证用户身份。

1.1.4.9.2.3 IPSec 协议

IPSec 是网络层 VPN 中最常用的协议，提供四个主要的安全功能：

- ✧ **数据加密**：加密数据以防止数据被窃听。
- ✧ **身份认证**：验证通信双方的身份，确保数据的发送方和接收方是合法的。
- ✧ **数据完整性**：确保数据在传输过程中未被篡改。
- ✧ **重放攻击防护**：防止旧的数据包被截获并重新发送，从而引发重放攻击。

IPSec 的应用广泛，特别是在远程访问 VPN 和站点到站点的连接中。

1.1.4.9.2.4 路由和地址转换

网络层 VPN 通常会使用 NAT（Network Address Translation）和路由协议（如 RIP、OSPF 等）来管理不同网络间的地址转换和路由。

- ✧ **NAT**：在 VPN 网关上使用 NAT 技术来转换源地址或目的地址，以确保数据包能正确传输。
- ✧ **路由协议**：如 OSPF 或 BGP，通常用来在 VPN 设备之间交换路由信息，确保正确的数据包转发。

1.1.4.9.3 VPN 的工作原理

VPN 的核心思想是通过在公用网络上建立一个“虚拟的私有隧道”，使得数据能够以加密的形式通过不安全的网络进行传输。VPN 的工作过程通常包含以下几个步骤：

1. **客户端发起连接**：用户通过 VPN 客户端软件发起连接请求，选择 VPN 服务器。
2. **认证过程**：客户端与 VPN 服务器进行身份验证，确保用户是合法的。
3. **建立隧道**：一旦认证通过，客户端与 VPN 服务器之间会建立一个加密的“隧道”，所有数据包都会在这个隧道中传输。这个隧道通常使用某种隧道协议来实现。
4. **数据加密与解密**：数据在传输过程中被加密，确保数据在公共网络中不会被窃取或篡改。当数据到达 VPN 服务器时，数据会解密，并转发到目标网络。

1.1.5 传输层（Transport Layer）

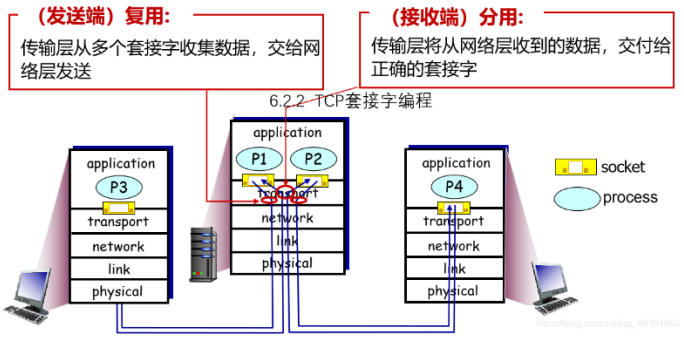
传输层是 TCP/IP 协议栈的第二层，位于网络层之上，应用层之下，主要任务是为端到端通信提供可靠的传输服务。它确保了数据的完整性、顺序以及传输的有效性。传输层通过提供不同的协议来支持不同的应用需求，常见的协议有 UDP 和 TCP。

1.1.5.1 传输层概述

1.1.5.1.1 传输层的作用

传输层主要负责以下功能：

- ✧ **数据传输**：实现不同主机之间的数据传输，确保数据从源主机传送到目标主机。
- ✧ **端到端的可靠性保证**：传输层通过错误检测与修正、流量控制、重传等机制确保数据的完整性和可靠性。
- ✧ **分用与复用**：传输层的一个重要特性是能够对应用进行分用与复用。**分用**是指将来自不同应用的数据传送到正确的应用程序，**复用**则是指将来自应用的数据复用到同一个网络连接中。



1.1.5.1.2 传输层协议

传输层协议的主要任务包括数据分段、传输、错误检测和控制等。

- **TCP（传输控制协议）**：提供可靠、面向连接的服务，确保数据按顺序正确传送，并实现数据的错误校验、重传和流量控制。
- **UDP（用户数据报协议）**：提供不可靠、无连接的服务，速度较快，但不保证数据传输的可靠性，适合实时应用（如视频、语音传输）

1.1.5.2 端口与套接字（Socket）

1.1.5.2.1 端口

端口是网络协议中的一个数字标识符，用于区分不同的服务或应用程序。每个计算机在网络中有一个 IP 地址，而端口则用来区分同一台计算机上运行的不同程序。端口号是一个 16 位的数字，范围从 0 到 65535。根据功能，端口号可以分为三类：

- **知名端口（Well-Known Ports）**：0–1023。由系统保留，通常用于常见的协议（如 HTTP 使用 80 端口，FTP 使用 21 端口）。
- **注册端口（Registered Ports）：**1024–49151。应用程序使用，不像知名端口那样固定。
- **动态端口（Dynamic Ports）：**49152–65535。由操作系统动态分配给用户进程

1.1.5.2.2 套接字

套接字是应用程序与网络进行通信的接口，它是**网络编程**中的基本概念。套接字是由 IP 地址、端口号和协议类型（如 TCP/UDP）组成的。通过套接字，应用程序能够发送和接收网络数据。

在 TCP/IP 协议中，套接字通常由两个主要部分构成：

- **IP 地址**：指定通信的目标计算机。
- **端口号**：指定该计算机上具体的服务或应用。

1.1.5.3 UDP 协议

1.1.5.3.1 UDP 的基本概念

UDP（用户数据报协议，User Datagram Protocol）是 OSI 模型传输层的一个协议。与 TCP（传输控制协议）不同，UDP 提供一种**无连接、不可靠**的数据传输服务。它的设计目标是减少协议开销和通信延迟，适用于对实时性要求较高且可以容忍数据丢失的应用场景。

UDP 的主要特点是：

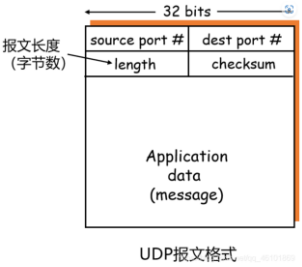
- **无连接**：发送数据前不需要建立连接。
- **不可靠**：不提供重传机制，也不保证数据到达顺序。
- **轻量级**：由于没有复杂的控制机制，UDP 具有更小的协议开销，适合高效、低延迟的数据传输。

1.1.5.3.2 UDP 报文段

UDP 数据包（或称报文段）由 **UDP 头部**和**数据部分**组成，头部格式非常简单，只有 4 个字段，整体大小固定为 8 个字节。

- **源端口和目的端口**：用来标识发送和接收数据的应用程序。
- **长度**：表示整个 UDP 报文的长度（包括头部和数据部分）。
- **校验和**：用于检测数据传输过程中是否发生了错误，虽然可选，但通常使用它来增强数据的可靠性。

字段	长度 (字节)	描述
源端口	2	发送方的端口号
目的端口	2	接收方的端口号
长度	2	UDP头部和数据部分的总长度（单位：字节）
校验和	2	用于检验数据完整性的校验和，若未使用则可置为0
数据	可变	实际传输的应用数据



UDP报文格式

1.1.5.3.3 UDP 的意义

- UDP 的核心优势在于低延迟和简洁的设计，适用于对实时性要求高的应用，如流媒体、在线游戏、语音通话和 DNS 查询等。它不进行重传和排序控制，减少了协议开销，因此在带宽有限的情况下，能够快速传输数据。
- UDP 特别适用于对速度要求高的应用，尽管它不保证可靠性，但对于一些容忍数据丢失的场景（如流媒体、实时通信和在线游戏），UDP 可以提供更高效的传输例如：
  - **实时通信**：如 VoIP，低延迟比数据丢失更重要。
  - **流媒体传输**：如在线视频和音乐，少量数据丢失不会影响体验。
  - **在线游戏**：低延迟和快速反馈是游戏体验的关键。
  - **DNS 查询**：快速响应，无需建立连接，丢失的数据可以简单重发。

1.1.5.4 TCP 协议

1.1.5.4.1 TCP 的基本概念

TCP（传输控制协议，Transmission Control Protocol）是传输层的核心协议之一，它提供可靠、面向连接的通信服务，确保数据按顺序、完整且无误地传输。相比于 UDP，TCP 通过各种机制确保数据的可靠性，但因此会带来更高的开销和延迟。

TCP 的关键特性

- **面向连接**：TCP 通信前需要建立连接，双方通过三次握手（Three-Way Handshake）过程进行连接确认。
- **可靠性保证**：TCP 确保数据传输的可靠性，通过数据确认、重传和排序等机制，保证数据的正确到达。
- **流量控制**：通过滑动窗口机制，TCP 能够调节数据发送速度，避免发送方过快导致接收方处理不过来。

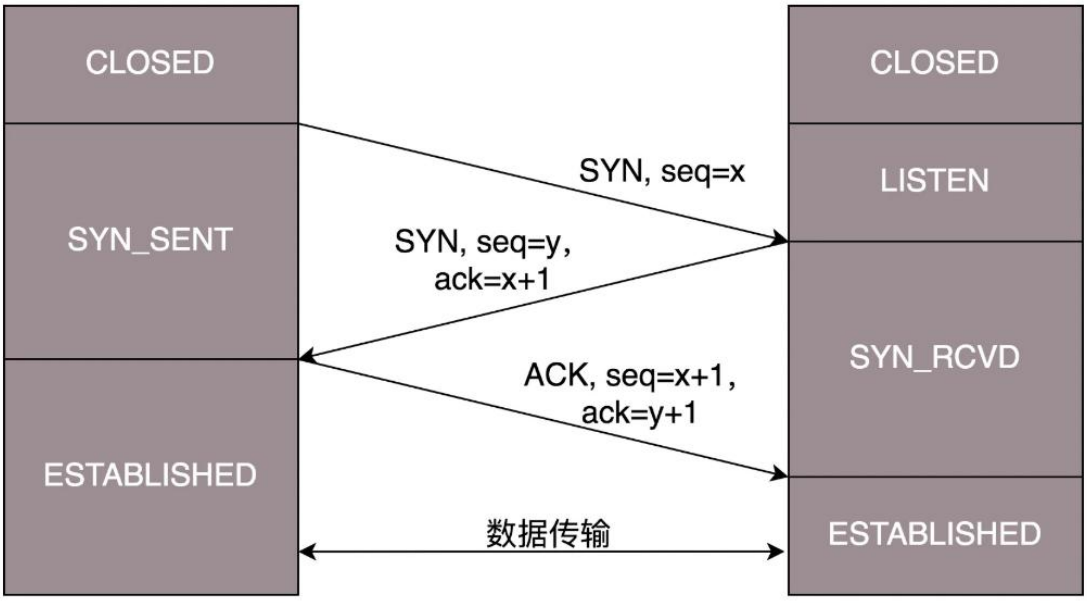
- ✧ **拥塞控制**：TCP 通过调整发送窗口大小来避免网络拥塞，确保网络不会过载。
- ✧ **数据顺序保证**：TCP 通过序列号保证数据包按正确的顺序到达接收方。
- ✧ **全双工通信**：TCP 支持双向同时通信，客户端和服务器可以同时发送和接收数据。

1.1.5.4.2 TCP 工作流程

1.1.5.4.2.1 三次握手（Three-Way Handshake）

三次握手是 TCP 连接建立过程中的关键步骤，确保双方可以正常通信。

1. **第一次握手：客户端发送 SYN 包**
   - ✧ 客户端向服务器发送一个 SYN（同步）报文段，表示客户端希望建立连接，并且客户端生成一个初始的序列号（ISN，Initial Sequence Number）。
   - ✧ 客户端发送 SYN，初始序列号 seq=x
   - ✧ 此时，客户端处于 **SYN_SENT** 状态，等待服务器确认。
2. **第二次握手：服务器回应 SYN-ACK**
   - ✧ 服务器收到客户端的 SYN 包后，表示愿意建立连接，便会回复一个 SYN-ACK（同步-确认）报文段，确认客户端的 SYN 请求。
   - ✧ 服务器选择自己的初始序列号（ISN）并向客户端确认，报文中的确认号（ACK）是客户端 ISN+1。
   - ✧ 服务器发送 SYN-ACK，seq 为自己序列号 y，ack 为 x+1
   - ✧ 此时，服务器处于 **SYN_RECEIVED** 状态，等待客户端的确认。
3. **第三次握手：客户端回应 ACK**
   - ✧ 客户端收到服务器的 SYN-ACK 报文后，向服务器发送一个 ACK（确认）报文段，确认服务器的 SYN 包，并将自己的序列号加 1 作为确认号。
   - ✧ 客户端也回应发送 ACK，seq 为 x+1，ack 为接受的 seq+1=y+1
   - ✧ 客户端进入 **ESTABLISHED** 状态，表示连接已建立。
   - ✧ 服务器收到 ACK 后，也进入 **ESTABLISHED** 状态，表示连接已经完全建立。



1.1.5.4.2.2 数据传输

一旦连接建立，数据传输可以开始。TCP 协议的传输过程有以下几个关键点：

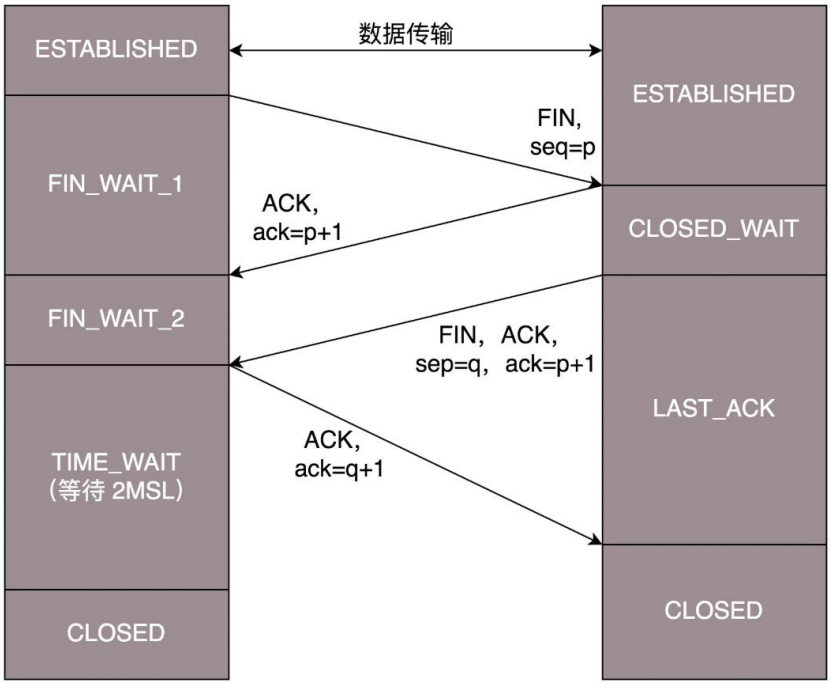
- **数据分段与序列号**
  - ✧ TCP 会将大的数据流分割成小的报文段，并为每个报文段分配一个序列号（Sequence Number），确保接收方能够按照顺序重组数据。
  - ✧ 序列号是单调递增的，接收方通过序列号来识别数据包的顺序，确保数据的顺序性。
- **确认应答（ACK）**
  - ✧ 每收到一个数据包，接收方都会向发送方发送一个确认（ACK）报文，告知发送方数据已成功接收。
  - ✧ 确认号是下一个期望接收的字节序列号，即接收方期望的下一个数据包的序列号。
- **流量控制（滑动窗口）**
  - ✧ TCP 使用滑动窗口（Sliding Window）机制来控制数据传输的速率。接收方会告知发送方自己可以接收的窗口大小（即缓冲区大小），发送方依据这个信息控制发送的数据量。
  - ✧ 窗口大小动态变化，发送方会根据接收方的能力调整发送速率，以避免接收方的缓冲区溢出。
- **拥塞控制**
  - ✧ TCP 通过拥塞控制算法（如慢启动、拥塞避免、快速重传等）来避免网络拥塞。
    - ○ **慢启动（Slow Start）**：开始时，发送方的拥塞窗口较小，随着确认的接收，窗口大小逐渐增大。
    - ○ **拥塞避免（Congestion Avoidance）**：当拥塞窗口增长到一定程度，进入线性增长阶段，避免过快地增加窗口，导致网络拥塞。
    - ○ **快速重传与恢复**：当发送方连续收到三个重复的 ACK 时，认为数据包丢失，立刻进行重传，并且进入快速恢复状态。
- **超时与重传**
  - ✧ 如果发送方在等待确认时超过了一个指定的超时时间（RTT，即往返时延），则认为数据包丢失，发送方会重传该数据包。

1.1.5.4.2.3 连接关闭（四次挥手）

TCP 连接关闭是通过四次挥手（Four-Way Handshake）来实现的。与建立连接的三次握手不同，关闭连接需要双方都参与，确保双方的数据都传输完成，避免数据丢失。

1. **第一次挥手：客户端发起 FIN**
   - ✧ 客户端发送一个 FIN（Finish）报文段，表示客户端没有数据要发送了，要求关闭连接。此时客户端进入 **FIN_WAIT_1** 状态。
   - ✧ 发送 FIN，seq 为 **p**
2. **第二次挥手：服务器回应 ACK**
   - ✧ 服务器收到客户端的 FIN 包后，发送一个 ACK 包，确认客户端关闭连接请求。此时，服务器进入 **CLOSE_WAIT** 状态，等待关闭其自己的连接。
   - ✧ 回应发送 ACK，ack 为 **p+1**
   - ✧ 客户端收到这个 ACK 后，进入 **FIN_WAIT_2** 状态，等待服务器关闭连接。
3. **第三次挥手：服务器发起 FIN**
   - ✧ 服务器准备关闭连接时，发送一个 FIN 报文段，表示服务器也没有数据要发送了，要求关闭连接。
   - ✧ 服务器发送 FIN，ack 还是 **p+1**，seq 为自己序列号 **q**

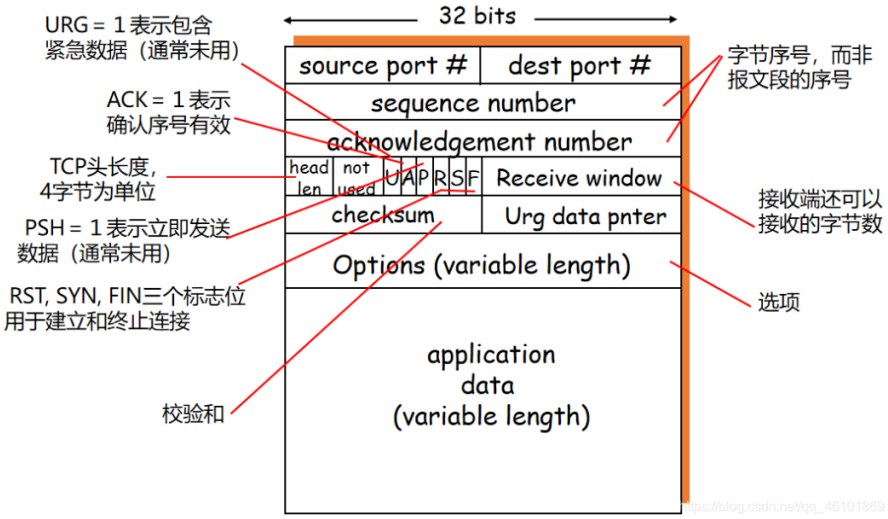
4. **第四次挥手：客户端回应 ACK**
   - ✧ 客户端收到服务器的 FIN 包后，向服务器发送一个 ACK 报文段，确认服务器关闭连接的请求。
   - ✧ 客户端进入 **TIME_WAIT** 状态，等待足够的时间（通常是 2 倍的最大报文段生存时间，即 2MSL）以确保服务器收到 ACK，避免丢包的情况。
   - ✧ 客户端发送 ACK，ack 为前面服务器 FIN 的序列号加一 q+1
   - ✧ 服务器收到 ACK 后，进入 **CLOSED** 状态，连接完全关闭。



1.1.5.4.3 TCP 报文段格式

TCP 头部较为复杂，由固定的 20 字节组成（如果没有选项字段），主要包含以下字段：

字段名称	长度 (字节)	描述
源端口	2	发送端的端口号
目的端口	2	接收端的端口号
序列号	4	数据流中第一个字节的编号，用于确保数据按顺序传输
确认号	4	表示接收到的数据的下一个字节的编号，用于数据确认
数据偏移	1	头部长度，单位为4字节（包含TCP选项）
保留	1	保留为0，确保TCP头部对齐
标志位	1	控制位，如SYN、ACK、FIN、RST等，用于控制连接状态
窗口大小	2	指定接收方的缓冲区大小，用于流量控制
校验和	2	用于检测数据传输中的错误
紧急指针	2	如果URG标志位为1，则指示紧急数据的最后字节的位置
选项	可变	可选字段，提供额外的控制功能，如最大段大小（MSS）等
数据	可变	实际传输的应用数据



1.1.5.5 滑动窗口

滑动窗口协议是用于数据传输中控制流量和确保可靠性的机制，主要通过**发送方窗口**和**接收方窗口**来控制数据的传输。在传输层，滑动窗口被用于 **TCP 协议**中，它确保了数据按顺序传输，且不会造成网络的拥塞或接收方的缓冲区溢出。

基本概念：

- ✧ **发送窗口**：发送方能够发送的最大数据量，它的大小受到接收方和网络条件的限制。
- ✧ **接收窗口**：接收方能够接收的最大数据量，它控制着发送方的流量。
- ✧ **滑动**：随着数据包的确认或超时重传，窗口在时间轴上“滑动”，使得发送方可以继续发送新的数据。

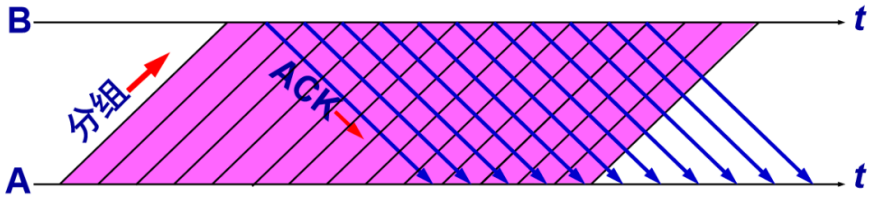
滑动窗口协议允许发送方在接收到确认之前连续发送多个数据包，这样可以有效利用带宽，减少等待确认的时间，从而提高网络的传输效率。

1.1.5.5.1 流水线

流水线是指在不等待每个数据包确认的情况下，发送方可以连续发送多个数据包的机制。通过使用滑动窗口，流水线机制使得数据传输更加高效，特别是在高延迟或带宽较大的网络中。

工作原理：

- ✧ **并行传输**：发送方可以连续发送多个数据包，而无需等待每个数据包的确认。这样，多个数据包可以同时在网络中流动。
- ✧ **滑动窗口**：发送方和接收方的窗口滑动决定了可以并行发送的数据量。发送方的窗口决定了能发送的最大数据包数，接收方窗口控制接收的最大数据量。
- ✧ **传输效率**：流水线能够提高网络的带宽利用率，减少等待时间，特别是对于高延迟的网络，能显著提高传输效率。



流水线传输可提高信道利用率

流水线的关键在于**滑动窗口协议的支持**，它使得数据包可以按顺序接收和确认，避免了发送方在等待确认时的空闲时间，从而使得整个传输过程更高效。

1.1.5.5.2 GBN

Go-Back-N（GBN）是一种简单的可靠数据传输协议，发送方最多可以发送 N 个数据包，但必须等待确认。在该协议中，如果一个数据包丢失或发生错误，接收方会丢弃该数据包并等待它的重新传输。

- **工作原理：**
  - ✧ **发送窗口：**发送方可以在没有接收到确认的情况下，发送最多 N 个数据包（窗口大小为 N）。发送窗口是一个滑动窗口，随着数据的发送，发送窗口向前滑动。
  - ✧ **接收窗口**：接收方的窗口通常为 1，因为接收方要求数据按顺序到达。如果接收方接收到乱序的数据包，它会丢弃该数据包并不发送确认。
  - ✧ **数据包编号**：每个数据包都会有一个唯一的序列号，序列号是循环递增的。发送方和接收方都需要跟踪这些序列号。
  - ✧ **确认机制**：接收方对按顺序接收到的数据包进行确认，并将确认号反馈给发送方。确认号表示接收到的最后一个按顺序的包的序列号。
  - ✧ **重传机制**：如果发送方没有在一定时间内收到确认，它会重传从丢失数据包开始的所有数据包。也就是说，丢失的一个数据包及其后续的数据包都会被重新发送。

- **具体步骤：**
  - ✧ **发送方操作：**
    - 发送方维护一个大小为 N 的发送窗口，窗口中的数据包可以被发送出去。
    - 每发送一个数据包，窗口向前滑动一个位置。
    - 发送方继续发送数据包，直到窗口满或数据包都发送完。
  - ✧ **接收方操作：累计确认**

    •接收方按照顺序接收数据包，如果接收到的数据包是按顺序的，则将其交给上层应用，并向发送方发送一个确认。

    •如果接收到的数据包是乱序的，接收方会丢弃该数据包，并不会发送确认。
  - ✧ **确认与重传：**

    •如果发送方在超时时间内没有收到某个数据包的确认，它会重传丢失的数据包及其后续所有的数据包。

    •如果发送方收到一个确认号为 k 的数据包确认，则表示序列号小于等于 k 的数据包已经成功到达。

- **优缺点：**

- ✧ 优点：简单易实现，适用于低延迟和低丢包率的网络。
- ✧ 缺点：效率较低，丢失一个数据包需要重传整个窗口的数据包，浪费带宽。

1.1.5.5.3 SR

Selective Repeat（SR）协议比 **GBN** 更加高效，它允许接收方接收乱序到达的数据包，并且仅重传丢失的数据包，而不是整个窗口中的所有数据包。

- **工作原理：**
  - ✧ **发送窗口**：发送方可以在没有接收到确认的情况下，发送最多 **N** 个数据包。与 GBN 不同的是，SR 允许发送方连续发送多个数据包，并且每个数据包的确认是独立的。
  - ✧ **接收窗口**：接收方可以接收乱序的数据包。接收方根据自己的缓冲区大小（通常也是 N）来管理接收的数据包。接收到的数据包被缓存，直到丢失的数据包到达，才会交给上层应用。
  - ✧ **数据包编号**：每个数据包都有一个唯一的序列号，发送方和接收方都维护一个编号表，来跟踪已发送和已接收的包。
  - ✧ **确认机制**：接收方会为每个成功接收到的数据包发送独立的确认。如果接收方收到了一个数据包，它会对这个数据包发送确认，而不必等待整个窗口的数据包。
  - ✧ **重传机制**：如果发送方没有收到某个数据包的确认，它只会重传丢失的数据包，而不会重传整个窗口中的所有数据包。

- **具体步骤：**
  - ✧ **发送方操作**：
    - 发送方发送多个数据包到接收方，最多可以发送 **N** 个数据包。
    - 发送方为每个数据包分配一个序列号，并维护一个发送窗口。发送方在窗口中发送数据包，并等待接收方的确认。
  - ✧ **接收方操作**：
    - 接收方接收到数据包后，若数据包序列号与期望的序列号一致，接收方将数据包交给上层应用并发送确认。
    - 如果接收到的数据包是乱序的，接收方会将其缓存，直到缺失的前一个数据包到达。
  - ✧ **确认与重传**：
    - 发送方会等待每个数据包的独立确认。如果某个数据包的确认没有在超时内收到，发送方只会重传该数据包，而不是整个窗口。
    - 每个数据包的确认是独立的，接收方在收到数据包后会发送确认，无论数据包是否按顺序到达。

- **优缺点：**
  - **优点**：相比 GBN，SR 协议更高效，减少了重传的次数，尤其在网络丢包时，只有丢失的数据包需要重传，节省带宽。
  - **缺点**：实现较为复杂，需要管理缓存和乱序数据包的接收。需要为每个包维护一个计时器。

1.1.5.6 TCP 流量控制

TCP 流量控制是确保发送方不会过快地向接收方发送数据，以避免接收方的缓冲区溢出的机

制。它是 TCP 协议在传输层的核心功能之一，主要通过滑动窗口机制来实现。

1.1.5.6.1 滑动窗口机制

- ✧ **定义**：滑动窗口机制是 TCP 流量控制的核心，允许发送方在未收到接收方确认的情况下发送一定量的数据。接收方通过“窗口大小”来告知发送方它能够接收的最大数据量。
- ✧ **窗口大小**：TCP 接收方在 TCP 报文头中的窗口大小字段中，告诉发送方它的缓冲区剩余容量。发送方根据接收方的窗口大小来控制数据的发送速率。
- ✧ **动态调整**：接收方可以根据其缓冲区的使用情况动态调整窗口大小。如果接收方的缓冲区空间足够，它会增大窗口；如果缓冲区满了，窗口会减小。

1.1.5.6.2 流量控制的工作原理

- **发送方行为**

发送方将数据分为多个数据段，并根据接收方提供的窗口大小发送数据。每个数据段都包含一个序列号和一个确认号，发送方会根据接收到的确认消息决定发送下一批数据。

- 使用显式的窗口通告，告知发送方可用的缓存空间大小
- 在接收窗口较小时，推迟发送确认
- 仅当接收窗口显著增加时，通告新的窗口大小

- **接收方行为**

接收方负责缓冲接收到的数据并通过 TCP ACK 确认收到的数据。接收方通过调整窗口大小来控制流量。

- 使用 Nagle 算法确定发送时机
- 使用接收窗口限制发送的数据量，已发送未确认的字节数不超过接收窗口的大小

1.1.5.6.3 窗口大小的计算

- ✧ 窗口大小通常是由接收方根据其缓冲区的剩余容量来动态计算的。它通过 window size 字段告诉发送方可以接收的数据量。
- ✧ 计算公式：接收窗口大小 = 接收方的接收缓冲区容量 - 已接收但尚未确认的数据量。

1.1.5.6.4 流量控制与拥塞控制的区别

- ✧ **流量控制**：流量控制侧重于确保接收方不会因数据处理过慢而被淹没，单纯为接收方的接收能力而设计的。
- ✧ **拥塞控制**：拥塞控制则侧重于避免网络中的过载，防止网络的拥堵和数据包的丢失。TCP 使用不同的算法（如慢启动、拥塞避免、快重传等）来控制发送方的发送速率。
- ✧ **相同点**：流量控制和拥塞控制都旨在优化数据传输，避免不必要的数据丢失和提高传输效率。

1.1.5.7 TCP 拥塞控制

TCP 拥塞控制是 TCP 协议的一部分，旨在避免网络中的拥堵。拥塞指的是网络中数据包的

过度积累，导致网络资源（如带宽、路由器缓冲区等）耗尽，进而导致数据包丢失和延迟增加。TCP 拥塞控制通过动态调整发送方的传输速率，以适应当前的网络状况，确保网络不会发生过度拥堵。

1.1.5.7.1 核心思想

TCP 拥塞控制的核心思想是**动态调整发送速率**以避免过度拥塞和数据丢失。具体来说，TCP 试图以“自适应的方式”控制数据流量，使得发送方的发送速率不会超过网络的承载能力。

1.1.5.7.2 重点概念

- **拥塞窗口（cwnd）**：表示 TCP 连接中，发送方可以发送的最大数据量，单位是字节。cwnd 根据网络的拥塞状况动态变化。
- **慢启动阈值（ssthresh）**：这是拥塞窗口在慢启动和拥塞避免阶段之间的转换点。当 cwnd 达到 ssthresh 后，进入拥塞避免阶段。
- **网络拥塞**：指网络中数据流量过大，导致路由器的缓冲区溢出，数据包丢失，进而影响传输性能。
- **RTT（Round Trip Time）**：表示从发送一个数据包到收到该数据包的确认所需的时间，它是衡量网络延迟的重要参数。

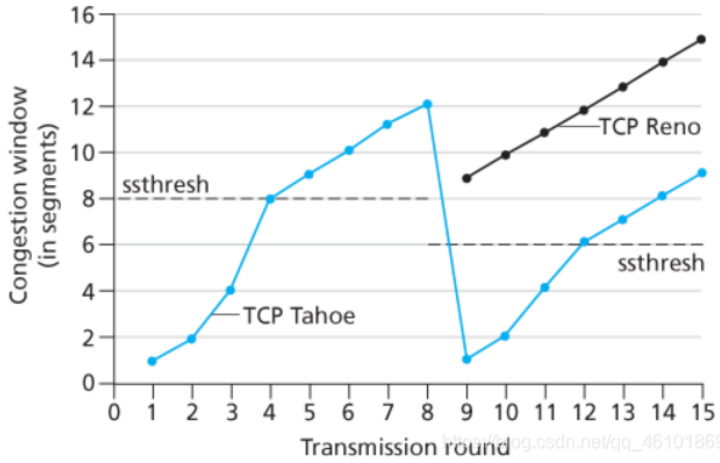
1.1.5.7.3 算法具体实现

TCP 使用以下几种算法来控制网络中的拥塞：

- **慢启动（Slow Start）**
  - **慢启动阶段**从拥塞窗口的初始值（通常是 1 个 MSS，最大报文段大小）开始，并且每收到一个确认（ACK）报文，cwnd 就增加 1 个 MSS。这是指数增长过程，使得在网络空闲时能够迅速占用带宽。
  - 在每个往返时延（RTT）结束时，cwnd 增加 1 个 MSS，直到达到阈值 ssthresh 为止。
  - **核心机制**：初始阶段快速增加窗口大小，快速利用可用带宽。
- **拥塞避免（Congestion Avoidance）**
  - 当 cwnd 达到 ssthresh 时，TCP 进入拥塞避免阶段。在这个阶段，cwnd 的增长速度变慢。每经过一个 RTT，cwnd 增加的量为 **MSS^2 / cwnd**，这意味着增长是线性的而非指数的。
  - **核心机制**：窗口大小逐步增加，避免网络过度拥塞。
- **快速重传（Fast Retransmit）**
  - 当接收方收到一个丢失的报文段后，会发送重复的 ACK，以告知发送方该数据包丢失。发送方收到 3 个相同的重复 ACK 时，认为该数据包已经丢失，并立即重传该数据包。
  - **核心机制**：快速检测和重传丢失的数据包，无需等待超时。
- **快速恢复（Fast Recovery）**
  - 当快速重传发生时，拥塞窗口（cwnd）会减半，并将 ssthresh 设置为 cwnd 的一半。然后进入快速恢复阶段，不会回到慢启动阶段，而是逐步增加 cwnd，直到恢复到正常的拥塞控制状态。
  - **核心机制**：减小窗口大小并快速恢复，避免回到慢启动阶段。

1.1.5.7.4 TCP 拥塞控制的关键过程

1. **初始阶段（慢启动）**：
   从 1 个 MSS 开始，cwnd 不断增加，直到达到 ssthresh。
2. **拥塞避免阶段**：
   cwnd 增速减缓，逐步增加数据传输速率。
3. **丢包发生（通过重复 ACK 或超时检测）**：
   通过快速重传（重复 ACK）进行丢包重传，并通过快速恢复减少 cwnd。
4. **调整窗口大小**：
   发生丢包时，cwnd 被减小，恢复过程通过增加 ssthresh 和逐步增大 cwnd 实现。



State	Event	TCP Sender Action	Commentary
慢启动 (SS)	收到新的确认	cwnd = cwnd + MSS, If (cwnd > ssthresh) set state to "Congestion Avoidance"	每经过一个RTT，cwnd 加倍
拥塞避免 (CA)	收到新的确认	cwnd = cwnd+MSS * (MSS/cwnd)	每经过一个RTT，cwnd 增加一个MSS
SS or CA	收到3个重复的确认	ssthresh = cwnd/2, cwnd = ssthresh+3, Set state to "Congestion Avoidance"	cwnd减半，然后线性增长
SS or CA	超时	ssthresh = cwnd/2, cwnd = 1 MSS, Set state to "Slow Start"	cwnd降为一个MSS，进入慢启动
SS or CA	收到一个重复的确认	统计收到的重复确认数	cwnd 和 ssthresh 都不变

1.1.6 应用层（Application Layer）

1.1.6.1 应用层概述

应用层是四层模型的最上层，负责为用户和应用提供网络服务。它通过直接与用户的应用程序交互，确保数据能够在网络上有效传输。应用层处理所有最终用户的通信，并通过应用协议定义了通信规则。

1.1.6.1.1 应用层的职责

应用层的主要职责是为用户提供直接的网络服务，并管理应用程序之间的通信。具体来说，应用层的职责包括：

- ✧ **数据格式化与表示**：定义数据的格式、语法和语义，确保不同应用程序之间的数据能够正确理解和处理。
- ✧ **提供网络服务**：通过各种协议（如 HTTP、FTP、SMTP、DNS 等），为用户提供网页浏览、文件传输、电子邮件等网络服务。
- ✧ **建立和管理会话**：应用层协议负责管理会话的建立、维持和终止，确保通信双方能持续交换数据。
- ✧ **端到端数据传输**：应用层通过传输层（如 TCP、UDP）来确保数据从源端到目的端的传递，涉及错误检测和数据完整性。
- ✧ **支持应用程序**：为具体的应用程序提供接口和支持，使其能够利用网络进行数据交换和服务访问。

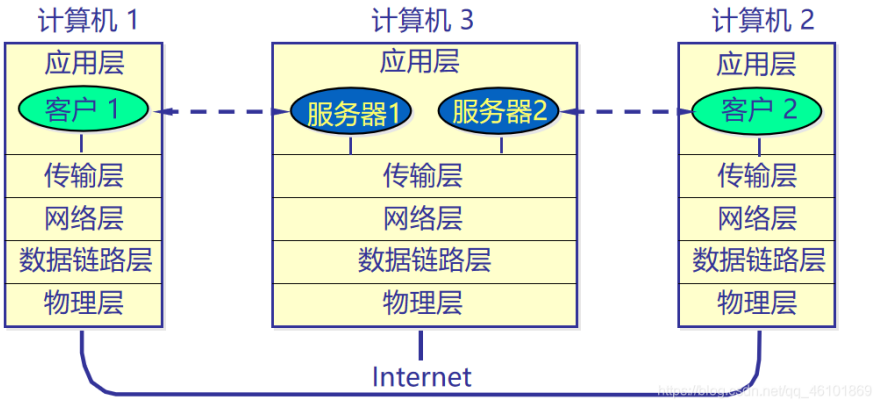
1.1.6.1.2 应用进程通信

应用进程通信通过不同的通信模型实现，主要有 C/S 模型和 P2P 模型。

1.1.6.1.2.1 C/S

在 C/S 模型中，客户端发起请求，服务器端提供服务。客户端和服务器之间通过网络进行通信，服务器通常负责管理资源和提供服务。例如，网页浏览器和网站之间的互动便是基于 C/S 架构。

- ✧ C/S 方式可以是面向连接的，也可以是无连接的
- ✧ 面向连接时，C/S 通信关系一旦建立，通信就是双向的，双方地位平等，都可发送和接收数据



- **特点：**
  - ✧ 集中式管理：服务器集中管理资源和服务，客户端通过网络向服务器发送请求并接收响应。
  - ✧ 请求-响应模型：客户端发起请求，服务器处理请求并返回结果。
  - ✧ 常见应用：Web 浏览器（客户端）与 Web 服务器（服务器）之间的 HTTP 通信、邮件客户端与邮件服务器之间的 SMTP/POP3 通信等。

- **优点：**
  - ✧ 易于管理和维护：服务器集中管理所有服务和数据，便于统一控制和维护。

- ✧ 扩展性好：可以根据需要增加多个客户端，且服务器可以进行扩展以处理更多的请求。

- **缺点：**
  - ✧ 单点故障：如果服务器发生故障，所有客户端的服务将受到影响。
  - ✧ 负载集中：所有请求都集中到服务器，可能会导致服务器负载过重。

1.1.6.1.2.2 P2P

P2P 模型是对等网络模型，网络中的每个节点都可以作为客户端和服务器。不同于 C/S 模型，P2P 模型中每个节点都可以直接与其他节点进行通信，不需要集中式的服务器。常见的 P2P 应用有文件共享和视频通话等。

✧ P2P 方式从本质上看仍然是使用了 C/S 方式，但强调的是通信过程中的对等，这时每一个 P2P 进程既是客户同时也是服务器

- **特点：**
  - ✧ 去中心化：每个节点都可以充当服务提供者和请求者，没有中心服务器。
  - ✧ 资源共享：各节点之间直接交换资源、文件或信息，通常用于文件共享、实时通讯等应用。
  - ✧ 常见应用：文件共享软件（如 BitTorrent）、即时消息应用（如 Skype）、区块链网络等。

- **优点：**
  - ✧ 高容错性：没有单点故障，节点的加入和退出不会影响整体网络的运行。
  - ✧ 高效利用带宽：通过多个节点之间的资源共享，可以提高带宽利用率，减少对单一服务器的依赖。
  - ✧ 扩展性强：可以根据需求随时增加或减少节点，灵活应对网络负载变化。

- **缺点：**
  - ✧ 安全性问题：由于没有中心控制，P2P 网络可能存在更多的安全隐患，如恶意节点、数据泄露等。
  - ✧ 管理复杂：去中心化的结构使得整个网络的管理和维护变得更复杂，尤其在资源共享和协同工作时可能出现冲突。

1.1.6.1.3 常见应用层协议

应用层包括许多标准协议，如：

✧ **HTTP（HyperText Transfer Protocol）**：用于网页浏览。
✧ **SMTP（Simple Mail Transfer Protocol）**：用于电子邮件发送。
✧ **FTP（File Transfer Protocol）**：用于文件传输。
✧ **DNS（Domain Name System）**：用于将域名解析为 IP 地址。

这些协议构成了应用层的核心，它们定义了应用程序如何通过网络传输数据。

1.1.6.2 DNS 协议（Domain Name System）

1.1.6.2.1 DNS 的概述

DNS（Domain Name System，域名系统）是用于将域名解析为 IP 地址的协议。由于计算机网络中传输数据需要使用 IP 地址，而人类更容易记住域名，因此 DNS 作为一种分布式的命名系统，将人类易记的域名转换为计算机理解的 IP 地址。DNS 是互联网中最重要的基础设施之一，它使得互联网的访问更加简便和高效。

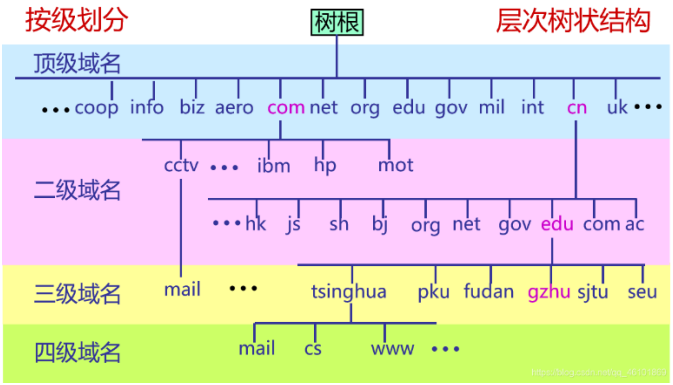
1.1.6.2.2 DNS 相关概念

- **域名（Domain Name）**：是为 Internet 上某一特定主机、服务、资源等指定的一个易于识别的标识符，通常包括多个层级，例如 www.example.com。
- **IP 地址（IP Address）**：是每台主机在网络上的唯一标识符，是由数字组成的地址，计算机通过 IP 地址相互通信。
- **DNS 解析（DNS Resolution）**：指将域名转换为 IP 地址的过程，这个过程可能通过多级查询来实现。
- **DNS 服务器（DNS Server）**：负责管理和解析域名信息的服务器，分为不同的类型（如根 DNS 服务器、权威 DNS 服务器、递归 DNS 服务器等）

1.1.6.2.3 域名系统名字空间和层次结构

DNS 采用分布式的层次结构来管理域名。域名从右到左的顺序代表了不同的层级，最右侧是最基本的层级，逐渐向左层级递增。

- **根域（Root Domain）**：DNS 结构的顶层，通常表示为一个点（.）。它是整个域名系统的起点。
- **顶级域（Top-Level Domain, TLD）**：位于根域下的第一层，如.com、.org、.net 等，也有国家代码顶级域（如.cn、.jp）。
- **二级域（Second-Level Domain）**：位于顶级域下的一层，通常由用户注册，如 example.com 中的 example。
- **三级域及更低层级的域（Subdomain）**：更详细的域名层次结构，例如 www.example.com 中的 www。



1.1.6.2.4 DNS 服务器

DNS 服务器的核心分类及其层级结构是支撑互联网域名解析的关键，以下从核心分类包括：根、顶级、权威、本地

1.1.6.2.4.1 根域名服务器（Root DNS Server）

- 功能：
  - 作为 DNS 解析的起点，存储所有顶级域（TLD）的权威服务器地址（如 .com、.cn 对应的 TLD 服务器 IP）。
- 特点：
  - 全球仅有 13 组逻辑根服务器集群（A-M 编号），通过任播技术扩展为上千台物理节点。
  - 不直接解析具体域名，仅返回下一级（TLD）的服务器地址。

1.1.6.2.4.2 顶级域名服务器（TLD DNS Server）

- 功能：
  - 管理特定顶级域（如 .com、.edu、.cn）下的权威服务器信息。
- 特点：
  - 由 ICANN 授权的机构运营（如 Verisign 管理 .com，CNNIC 管理 .cn）。
  - 根据根服务器的指引，返回目标域名的权威服务器地址。

1.1.6.2.4.3 权威域名服务器（Authoritative DNS Server）

- 功能：
  - 存储并管理具体域名的 DNS 记录（如 A 记录、MX 记录），直接提供最终解析结果。
- 分类：
  - 主服务器（Primary）：直接管理域名记录，可修改配置。
  - 辅助服务器（Secondary）：同步主服务器数据，提供冗余。

1.1.6.2.4.4 本地域名服务器（Local DNS Resolver）

- 功能：
  - 用户设备直接连接的 DNS 服务器，负责发起递归查询并缓存结果。
- 特点：
  - 通常由 ISP 或公共 DNS 服务商（如 Google DNS、Cloudflare）提供。
  - 若本地无缓存，则向根、TLD、权威服务器逐级查询。

1.1.6.2.4.5 DNS 服务器层级

1.1.6.2.5 DNS 解析过程

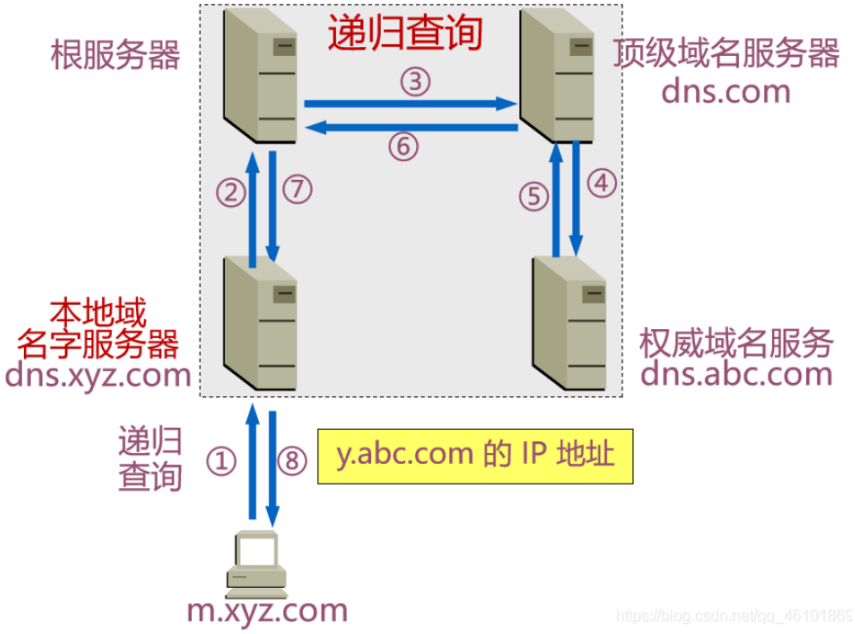
DNS 解析过程是域名转化为 IP 地址的过程，通常分为两种查询方式：**递归查询**和**迭代查询**。

1.1.6.2.5.1 递归查询

在递归查询中，客户端请求 DNS 解析时会向 DNS 服务器发送查询请求。如果该服务器无法解析请求，它会代为向其他 DNS 服务器进行查询，直到获取到结果为止。客户端收到最终结果后，再将其返回给用户。

- **过程**：
  1. 客户端向本地 DNS 服务器发起域名查询请求。

2. 如果本地 DNS 服务器无法解析，便递归地向其他 DNS 服务器查询（通常从根 DNS 服务器开始）。
3. 递归查询过程中，DNS 服务器会返回每个查询结果，直到最终解析出 IP 地址。
4. 结果最终返回给客户端。

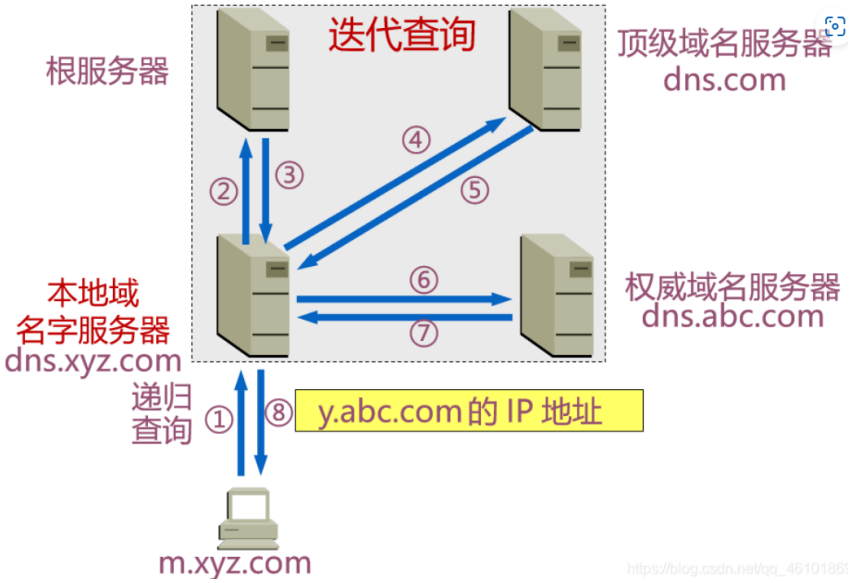


- **优点**：客户端只需要向本地 DNS 服务器请求一次，其他所有查询操作由 DNS 服务器负责处理。
- **缺点**：对 DNS 服务器的负担较重，需要 DNS 服务器在整个查询过程中保持状态。

1.1.6.2.5.2 迭代查询

在迭代查询中，客户端请求 DNS 解析时，DNS 服务器不会代为查询其他 DNS 服务器，而是返回指向下一个 DNS 服务器的地址，客户端需要继续向返回的 DNS 服务器发起查询，直到获得最终结果。

- **过程**：
  1. 客户端向本地 DNS 服务器发起域名查询请求。
  2. 如果本地 DNS 服务器无法解析，它会返回根 DNS 服务器或顶级 DNS 服务器的地址。
  3. 客户端根据返回的 DNS 服务器地址，向该服务器发起新的查询请求。
  4. 直到得到最终的 IP 地址，查询过程完成。



- **优点**：每个 DNS 服务器的负载较轻，因为它们只需要返回指向下一个服务器的地址，而不是代为完成查询。
- **缺点**：客户端需要发起多次查询请求，直到获得结果。

1.1.6.2.6 DNS 缓存

DNS 缓存是提高 DNS 解析效率的重要机制，它用于存储已解析的域名和 IP 地址的映射信息。通过缓存，DNS 服务器或客户端可以避免重复查询，提高解析速度，减少网络流量。

- **DNS 缓存的类型**：
  - **本地 DNS 缓存**：客户端计算机（如浏览器）会缓存最近解析的域名和对应的 IP 地址，减少频繁的 DNS 查询。
  - **DNS 服务器缓存**：DNS 服务器（如递归 DNS 服务器）也会缓存已解析的域名信息，供后续的查询使用。
- **缓存过期时间**：
  - 缓存中的数据具有有效期（TTL，Time to Live），TTL 值由权威 DNS 服务器设置，表示缓存条目的过期时间。TTL 过期后，缓存数据会被删除，客户端或服务器需要重新查询。
- **缓存的优点**：
  - **提高效率**：避免重复查询，减少解析延迟。
  - **减少网络负载**：缓存可以减少向 DNS 服务器发送的请求，减轻服务器压力。
- **缓存的缺点**：
  - **缓存污染**：如果缓存中存储了过时或错误的 IP 地址，可能导致无法正确访问网站。
  - **同步问题**：当 DNS 记录发生变化时，缓存可能会导致客户端或服务器继续使用过时的 IP 地址。

1.1.6.3 电子邮件

电子邮件系统的主要功能是提供一种可靠的方式来传送文本信息、文件和其他数据。它的核心包含两部分：发送邮件的机制和接收邮件的机制。

1.1.6.3.1 SMTP（发送）

SMTP（Simple Mail Transfer Protocol）是用于电子邮件传输的协议，负责将邮件从客户端发送到邮件服务器，或在邮件服务器之间传递。SMTP 主要用于**发送邮件**，处理发送端到接收端的邮件投递过程。

工作流程：

1. 用户在客户端（例如 Outlook 或 Webmail）编写邮件并点击发送。
2. 客户端通过 SMTP 协议将邮件发送到发件人的邮件服务器。
3. 邮件服务器通过 SMTP 协议将邮件传递到接收方的邮件服务器。
4. 邮件被存储在接收方的邮件服务器上，等待被接收方下载或读取。

1.1.6.3.2 POP3 与 IMAP 协议（接收）

POP3（邮局协议 3）和 IMAP（互联网邮件访问协议）也是用于从邮件服务器**接收**电子邮件的协议。

1.1.6.3.2.1 POP3

POP3 是较早的邮件接收协议，它允许用户从邮件服务器上下载电子邮件并存储在本地计算机上。默认情况下，POP3 会将邮件从服务器上删除，除非进行配置保持副本。

工作流程：

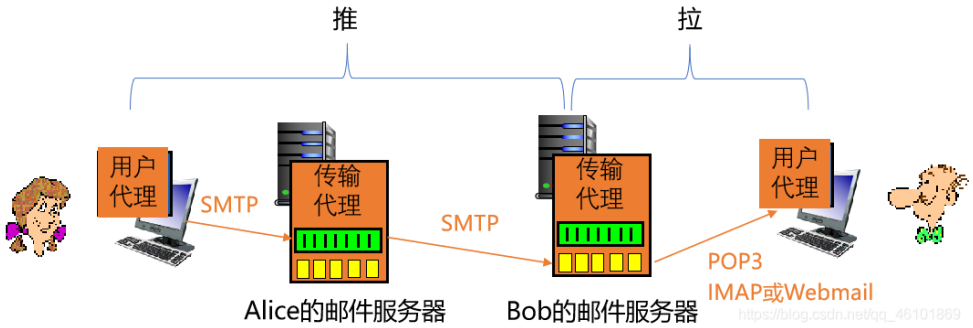
1. 用户连接到邮件服务器。
2. 下载邮件到本地计算机。
3. 邮件会从服务器上删除（除非选择保留副本）。

1.1.6.3.2.2 IMAP

IMAP 是一个更加先进和灵活的协议，允许用户在邮件服务器上保留邮件并进行管理。用户可以在多个设备上访问同一封邮件，IMAP 通过同步操作保持邮件的状态一致，支持对邮件进行标记、删除、归档等操作，用户可以选择在不同文件夹之间移动邮件。

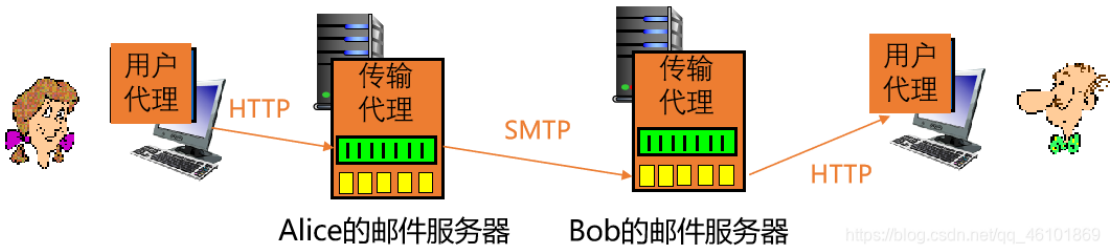
工作流程：

1. 用户连接到邮件服务器并同步邮件。
2. 邮件保留在服务器上，用户可以在多个设备间访问和管理邮件。



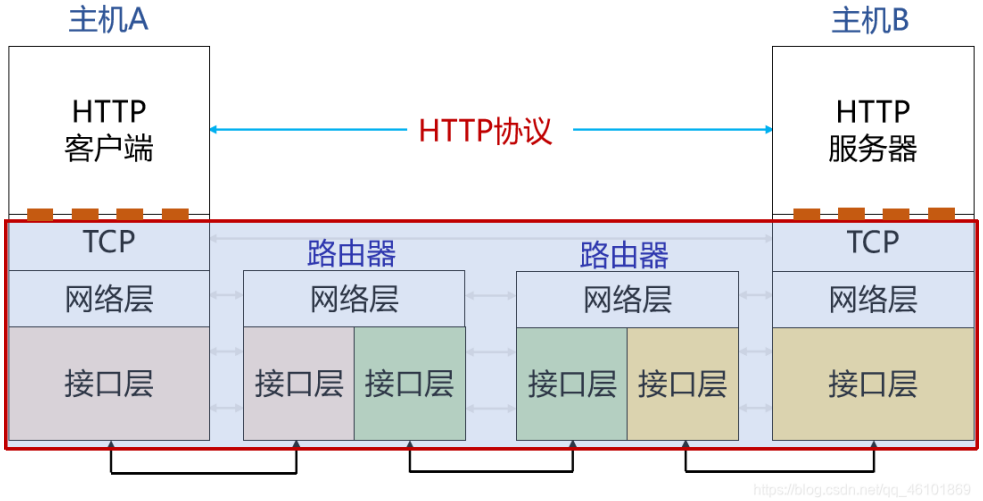
1.1.6.3.3 web 电子邮件

- 提供电子邮件服务的 IMAP 和 SMTP 替代方案
- 使用 Web 作为界面，用户代理就是普通的浏览器
- 用户及其远程邮箱之间的通信通过 HTTP 进行



1.1.6.4 万维网体系结构

万维网（World Wide Web, WWW）体系结构是一个分布式的信息系统，旨在通过互联网提供全球范围的信息访问。它建立在客户端-服务器模型上，利用标准协议和技术，如 HTTP、HTML、URL 等，使得用户能够通过浏览器访问各种信息资源。



1.1.6.4.1 服务器

Web 服务器是万维网的核心，它负责响应来自客户端的请求，提供所需的网页资源或服务。

- **作用：**

服务器通过 HTTP 协议接收客户端的请求并返回响应，通常包括网页、图片、视频等。Web 服务器可以是单一的，也可以是分布式的，能够处理多个客户端的请求。

- **关键技术：**
  - ✧ HTTP 协议：Web 服务器和客户端之间的通信协议。Web 服务器处理客户端的 HTTP 请求并返回 HTTP 响应。
  - ✧ Web 服务：Web 服务器提供的不仅限于静态资源，还可以通过动态 Web 应用（如 PHP、ASP.NET、Java Servlets 等）提供交互式功能。

1.1.6.4.2 客户端（浏览器）

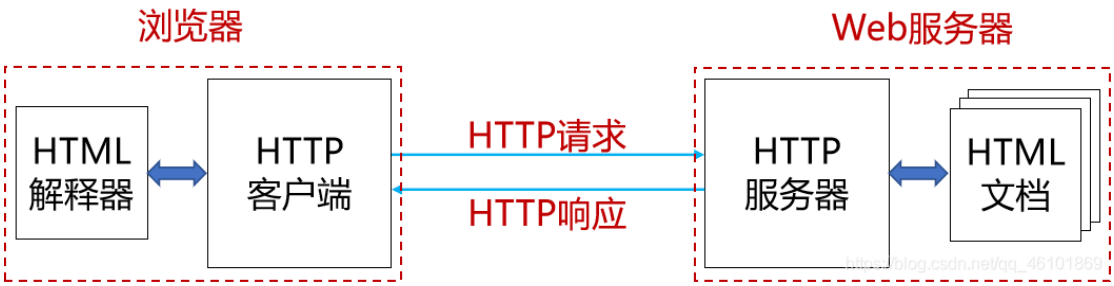
客户端层是用户与万维网交互的界面，主要通过 Web 浏览器（如 Chrome、Firefox、Safari 等）来访问资源。

- **作用：**

浏览器向 Web 服务器发送请求并解析返回的内容，展示给用户。客户端通过 HTTP 协议向服务器发起请求，接收返回的网页内容、图片、视频等。

- **关键技术：**

  - ✧ HTML（超文本标记语言）：定义网页的结构。
  - ✧ CSS（层叠样式表）：定义网页的样式和布局。
  - ✧ JavaScript：为网页提供动态功能（如交互、动画等）。



1.1.6.4.3 网络层

网络层是万维网架构的基础，负责将客户端和服务器连接起来，确保数据的传输。

- 作用：

网络层通过互联网提供物理连接，利用网络协议（如 TCP/IP）确保数据包能够在客户端和服务器之间传递。互联网的基础设施（包括路由器、交换机等设备）构成了这个层次。

1.1.6.4.4 URL

URL（Uniform Resource Locator，统一资源定位符）是用于标识和定位互联网上资源的地址，通常指向某个文件、网页或其他网络资源。它是万维网体系结构中至关重要的组成部分，通过 URL，浏览器和其他客户端能够定位并请求特定的资源。

例如

https://www.example.com:443/products?id=1234&category=books#section2

通常由以下几个部分组成：

1. **协议（Scheme）**：指定访问资源的协议，常见的如 http、https、ftp 等。
   - ✧ 示例：https
2. **主机名（Host）**：服务器的地址，通常是域名或 IP 地址。
   - ✧ 示例：www.example.com
3. **端口（Port）**：指定服务器的端口，HTTP 默认端口为 80，HTTPS 为 443。
   - ✧ 示例：:443

4. **路径（Path）**：资源在服务器上的具体位置。
   - ✧ 示例：/products/1234
5. **查询字符串（Query String）**：附加的参数，用于传递数据给服务器。
   - ✧ 示例：?id=1234&category=books
6. **片段标识符（Fragment Identifier）**：指向页面中的特定部分。
   - ✧ 示例：#section1

1.1.6.5 HTTP 协议（HyperText Transfer Protocol）

HTTP（超文本传输协议）是万维网（WWW）中用于客户端（通常是浏览器）与服务器之间进行通信的应用层协议。它定义了客户端与服务器之间请求和响应的格式，是 Web 浏览器与 Web 服务器之间的标准协议。HTTP 是**无状态的**、**面向请求-响应**的协议。

1.1.6.5.1 HTTP 的基本概念

1.1.6.5.2 HTTP 请求与响应（工作原理）

HTTP 的基本工作原理是客户端（通常是 web 浏览器）向服务器发送请求，服务器接收到请求后，返回相应的资源。这些资源可以是网页、图像、音频文件、视频等。

1. **建立连接**：客户端与服务器之间建立连接。在传统的 HTTP 中，这是基于 TCP/IP 协议的。最近的 HTTP/2 和 HTTP/3 则使用了更先进的传输层协议，例如基于 TCP 的二进制协议（HTTP/2）或基于 UDP 的 QUIC 协议（HTTP/3）。
2. **发送请求**：客户端向服务器发送请求，请求中包含要访问的资源的 URL、请求方法（GET、POST、PUT、DELETE 等）、请求头（例如，Accept、User-Agent）以及可选的请求体（对于 POST 或 PUT 请求）。
3. **处理请求**：服务器接收到请求后，根据请求中的信息找到相应的资源，执行相应的处理操作。这可能涉及从数据库中检索数据、生成动态内容或者简单地返回静态文件。
4. **发送响应**：服务器将处理后的结果封装在响应中，并将其发送回客户端。响应包含状态码（用于指示请求的成功或失败）、响应头（例如，Content-Type、Content-Length）以及可选的响应体（例如，HTML 页面、图像数据）。
5. **关闭连接**：在完成请求-响应周期后，客户端和服务器之间的连接可以被关闭，除非使用了持久连接（如 HTTP/1.1 中的 keep-alive）

1.1.6.5.3 HTTP 消息结构

1.1.6.5.3.1 客户端请求

客户端发送一个 HTTP 请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据四个部分组成。

- **请求行**（Request Line）：
  - ✧ **方法**：如 GET、POST、PUT、DELETE 等，指定要执行的操作。
  - ✧ **请求 URI**（统一资源标识符）：请求的资源路径，通常包括主机名、端口号（如果非默认）、路径和查询字符串。
  - ✧ **HTTP 版本**：如 HTTP/1.1 或 HTTP/2。

  请求行的格式示例：GET /index.html HTTP/1.1

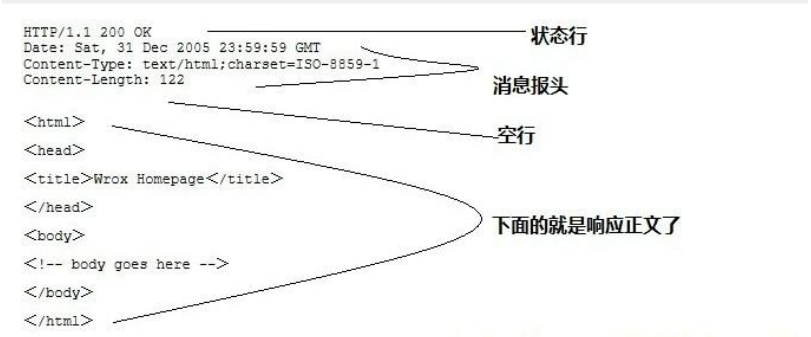
- **请求头**（Request Headers）：
  - ✧ 包含了客户端环境信息、请求体的大小（如果有）、客户端支持的压缩类型等。
  - ✧ 常见的请求头包括 Host、User-Agent、Accept、Accept-Encoding、Content-Length 等。

- **空行**：
  - ✧ 请求头和请求体之间的分隔符，表示请求头的结束。

- **请求体**（可选）：
  - ✧ 在某些类型的 HTTP 请求（如 POST 和 PUT）中，请求体包含要发送给服务器的数据。

1.1.6.5.3.2 服务器响应

HTTP 响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。



- **状态行（Status Line）：**
  - ✧ HTTP 版本：与请求消息中的版本相匹配。
  - ✧ 状态码：三位数，表示请求的处理结果，如 200 表示成功，404 表示未找到资源。
  - ✧ 状态信息：状态码的简短描述。

  状态行的格式示例：HTTP/1.1 200 OK
- **响应头（Response Headers）：**
  - ✧ 包含了服务器环境信息、响应体的大小、服务器支持的压缩类型等。
  - ✧ 常见的响应头包括 Content-Type、Content-Length、Server、Set-Cookie 等。
- **空行：**
  - ✧ 响应头和响应体之间的分隔符，表示响应头的结束。
- **响应体（可选）：**
  - ✧ 包含服务器返回的数据，如请求的网页内容、图片、JSON 数据等。

1.1.6.5.4 HTTP 方法

HTTP 方法（也称为 HTTP 动词）用于指定客户端请求服务器资源时的操作类型。常见的 HTTP 方法包括：

- GET：请求指定的资源。数据通过 URL 传递，通常用于获取数据，不会修改服务器上的资源。
- POST：向服务器提交数据（如表单数据），用于创建或更新资源。
- PUT：更新指定的资源。通常用于完全替换目标资源。
- DELETE：删除指定的资源。
- HEAD：与 GET 类似，但只返回响应头，不返回响应体。用于获取资源的元数据。
- PATCH：部分更新指定的资源，只修改资源的一部分。
- OPTIONS：请求服务器支持的 HTTP 方法，通常用于检查某个资源的可操作性。
- TRACE：回显服务器收到的请求，主要用于调试和诊断。

这些方法帮助服务器了解客户端请求的类型，以便适当处理请求和响应。

1.1.6.5.5 HTTP 状态码

HTTP 状态码是服务器返回的数字代码，表示 HTTP 请求的处理结果。状态码分为五类，每类代表不同的含义。

1. **1xx（信息性状态码）：表示请求已被接收，继续处理。**
   - ✧ 100 Continue：客户端应继续发送请求的其余部分。

2. **2xx（成功状态码）：表示请求成功，服务器成功处理了请求。**
   - ✧ 200 OK：请求成功，服务器返回请求的内容。
   - ✧ 201 Created：资源创建成功（如 POST 请求）。

3. **3xx（重定向状态码）：表示需要客户端进一步操作才能完成请求。**
   - ✧ 301 Moved Permanently：请求的资源已被永久移动到新位置。
   - ✧ 302 Found：资源临时移动到新位置。

4. **4xx（客户端错误状态码）：表示请求有错误，客户端需要修改请求。**
   - ✧ 400 Bad Request：请求有语法错误，服务器无法理解。
   - ✧ 404 Not Found：请求的资源不存在。

5. **5xx（服务器错误状态码）：表示服务器处理请求时发生了错误。**
   - ✧ 500 Internal Server Error：服务器内部错误，无法处理请求。
   - ✧ 502 Bad Gateway：网关或代理服务器收到无效响应。

1.1.6.5.6 Cookie

HTTP **无状态协议**，服务器 **用 cookies 保持用户状态，**Cookie 是 Web 服务器用于存储在客户端浏览器中的小块数据。它在客户端和服务器之间传递，用于保持会话信息、用户偏好设置等。

1.1.6.5.6.1 工作原理

- **设置 Cookie**：当用户访问一个网站时，Web 服务器会通过 HTTP 响应头中的 Set-Cookie 字段向客户端发送 Cookie 数据。
- **返回 Cookie**：在后续的请求中，浏览器会将对应的 Cookie 信息附加到 HTTP 请求头中，通过 Cookie 字段发送给服务器。

1.1.6.5.6.2 常见用途

✧ 会话管理：用于保持用户的登录状态，避免每次请求都需要重新认证。
✧ 个性化设置：存储用户偏好设置，如语言、主题等。
✧ 追踪与分析：用于追踪用户的浏览行为和分析网站流量。

1.1.6.5.6.3 Cookie 的属性

✧ expires：指定 Cookie 的过期时间。
✧ path：指定 Cookie 适用的路径。
✧ domain：指定 Cookie 适用的域名。
✧ secure：仅在 HTTPS 连接中发送 Cookie。
✧ HttpOnly：仅允许服务器访问 Cookie，客户端脚本无法读取。

1.1.6.5.7 HTTPS（HTTP Secure）

HTTPS（超文本传输安全协议）是基于 HTTP 的安全版本，它通过 SSL/TLS 协议对 HTTP 请求和响应的数据进行加密，确保数据的机密性和完整性，并提供身份验证功能。

工作原理

✧ **SSL/TLS 协议**：HTTPS 通过 SSL/TLS 加密层保障通信的安全性。SSL/TLS 在 HTTP 协议和应用程序之间提供加密和身份验证。
✧ **加密**：使用公钥加密技术，在客户端和服务器之间传递加密的 HTTP 请求和响应数据，防止中间人攻击和数据泄露。
✧ **身份验证**：SSL/TLS 使用数字证书验证服务器的身份，确保客户端连接的是合法的服务器。

HTTPS 的优势

✧ 加密通信：通过加密确保数据在传输过程中不被窃取。
✧ 数据完整性：确保传输的数据在传递过程中没有被篡改。
✧ 身份验证：通过 SSL 证书验证服务器的真实性，防止钓鱼网站。

HTTPS 与 HTTP 的区别

✧ HTTP：数据以明文形式传输，容易受到中间人攻击。
✧ HTTPS：通过 SSL/TLS 加密传输，数据在传输过程中是加密的，安全性更高。

1.1.6.6 CDN

CDN（内容分发网络）是一种通过分布在全球各地的服务器网络来优化用户访问速度的技

术。它的目的是减少因地理位置或网络延迟导致的加载速度慢、带宽过载等问题，提升互联网内容（如网页、图片、视频等）的传输效率和可靠性。

1.1.6.6.1 工作原理

CDN 通过将内容分布到多个地理上分散的服务器节点上（称为边缘节点），从而使用户能够从离他们最近的服务器获取数据。这样可以减少延迟，提高下载速度，减轻源服务器的负担。

- ✧ **请求**：用户请求资源时，CDN 选择最近的边缘节点提供服务。
- ✧ **缓存**：边缘节点缓存常用内容，避免每次都从源服务器请求。
- ✧ **加速**：通过优化路由和分发，提升用户访问速度。

1.1.6.6.2 主要优点

- ✧ **降低响应时延**：通过将内容存储在离用户近的边缘节点，减少数据传输距离，显著提高访问速度。
- ✧ **避免网络拥塞与源服务器过载**：通过分发流量到多个节点，减轻源服务器负担，并防止 DDoS 攻击。
- ✧ **良好的可扩展性**：分布式架构使得 CDN 易于扩展，可以应对大规模并发用户的请求。
- ✧ **用户透明**：CDN 对最终用户透明，用户不需感知内容来自 CDN 网络。

1.1.6.6.3 关键技术

- ✧ **DNS 重定向**：通过 DNS 将用户请求引导至最近的 CDN 节点，减少延迟和网络拥塞。
- ✧ **HTTP 重定向**：源服务器通过 HTTP 状态码（30X）和 Location 头部将请求转发至 CDN 节点。
- ✧ **负载均衡 DNS**：根据 CDN 节点的负载和网络状况，动态选择最优节点来处理请求，避免过载。
- ✧ **边缘缓存**：CDN 在边缘节点缓存静态资源，减少源服务器负担，提升加载速度。
- ✧ **智能路由**：基于网络状况，动态选择最佳路径传输数据，进一步减少延迟。

1.1.6.7 其它应用层协议

1.1.6.7.1 FTP 协议（File Transfer Protocol）

FTP（文件传输协议）是一种标准的网络协议，用于在客户端和服务器之间传输文件。它最常用于将文件从计算机上传到服务器，或从服务器下载文件到计算机。

- **核心思想**

FTP 通过在客户端和服务器之间建立两个连接：一个用于控制命令（控制连接），另一个用于传输数据（数据连接）。这种分离控制和数据的方式可以提高文件传输的效率。

- **工作原理**
  - ✧ **控制连接**：当客户端与 FTP 服务器建立连接时，客户端会通过控制连接发送命令来请求文件、删除文件、列出文件等操作。这是一个持久连接，通常使用 TCP 的 21 端口。

- **数据连接**：用于实际传输文件内容。根据不同模式（主动模式或被动模式），数据连接可以由客户端或服务器发起。数据连接使用随机的端口（通常是 20 端口）。

1.1.6.7.2 Telnet 协议

Telnet 是一种网络协议，允许用户通过命令行远程登录到远程计算机，通常用于管理和操作远程设备。它基于 TCP 协议，通过端口 23 进行连接。

- **核心思想**

Telnet 使得远程计算机能够提供交互式的命令行界面，用户通过它可以直接在远程计算机上执行命令、运行程序等，类似于本地使用命令行操作。它的目标是提供透明的远程计算机访问体验。

- **工作原理**
  - **连接建立**：Telnet 客户端与 Telnet 服务器建立 TCP 连接，通常使用端口 23。
  - **交互**：一旦连接成功，用户可以通过命令行输入指令，远程计算机将相应的结果返回给客户端，用户可以在本地看到输出。
  - **无加密**：Telnet 协议本身没有加密，所有数据，包括用户名和密码，都是以明文形式传输。这意味着，任何在网络中间的人都可以拦截和读取这些信息。
  - **安全性**：由于数据传输是明文的，Telnet 容易受到中间人攻击，因此它已经被更安全的协议（如 SSH）取代。SSH 提供了加密传输和身份验证功能，确保数据在网络传输过程中的安全。

1.1.6.7.3 WebSocket 协议

WebSocket 是一种网络协议，提供全双工通信，允许客户端和服务器在单个 TCP 连接上进行实时、低延迟的数据交换。适用于实时应用，如在线聊天、游戏、实时数据推送等。

- **核心思想**

WebSocket 通过一个持久的连接实现客户端与服务器之间的双向通信。与传统的 HTTP 请求-响应模式不同，WebSocket 保持连接状态，数据可以在连接期间双向流动，减少了请求和响应的延迟。

- **工作原理**
  1. **握手过程**
     客户端发起 HTTP 请求并请求升级为 WebSocket 连接。服务器响应 HTTP 101 状态码确认协议切换。
  2. **数据传输**
     连接建立后，客户端与服务器可以随时双向发送消息。数据通过“帧”进行传输，每个帧包含控制信息和数据负载。
  3. **连接关闭**
     任何一方都可以通过发送“关闭帧”来结束连接。
- **特点**
  - **全双工通信**：客户端和服务器可以同时发送数据，通信是双向的。
  - **低延迟**：建立连接后，客户端和服务器可以实时交换数据，减少延迟。

- **高效**：WebSocket 消息框架轻量，不需要重复传输 HTTP 头部信息，节省带宽。
- **持久连接**：WebSocket 使用持久连接，避免了传统 HTTP 的频繁建立和断开连接。

- **应用场景**
  - **在线聊天**：支持即时消息传递。
  - **实时数据推送**：如股票行情、新闻更新、天气预报等。
  - **多人在线游戏**：实时同步玩家操作。
  - **物联网（IoT）**：设备和服务器之间的实时通信。

- **WebSocket 与 HTTP 的区别**

特性	HTTP	WebSocket
通信模式	请求-响应（单向）	全双工（双向）
连接建立	每次请求建立连接	持久连接
数据传输效率	需要重复发送头部信息	高效，仅传输数据
延迟	较高	较低，实时通信