Chapter1: 核心计算与交换技术

1. The Network Core: Switching (交换技术)

1. Circuit Switching (电路交换)

- **Definition**: End-to-end resources (bandwidth) are **reserved** (资源独占).
- **Multiplexing**: FDM (Frequency bands), TDM (Time slots).

必考 (Exam Focus)

Exam Focus: 电路交换可以有几个对话同时进行

Network with 4 switches (A, B, C, D) in a ring. Each link has N circuits.

- **Check**: Total active connections on any single link $\le N$.
- **Path Finding**: Connection A-C via B occupies Link A-B **AND** Link B-C.

2. Packet Switching (分组交换)

- **Store-and-Forward**: Router must receive the **entire packet** before transmitting to next link.
- **Message Segmentation**: Breaking a large message into smaller packets (Assgn 1 P31).

必考 (Exam Focus)

Exam Focus: 传输时间计算 [Assgn 1 P31 & Tut 1 Q1]

Sending Data Size F over N links (hops) with rate R.

- **Case A: 不切片 (One huge packet)**:

$$D = N \times \frac{F}{R}$$

- **Case B: 切片 (P packets, each size L)**: The first packet faces full delay; subsequent packets follow in a "pipeline".

$$D = \underbrace{N \times \frac{L}{R}}_{\text{Time for 1st pkt}} + \underbrace{(P-1) \times \frac{L}{R}}_{\text{Time for remaining pkts}} = (\mathbf{N}+\mathbf{P}-\mathbf{1})\frac{\mathbf{L}}{\mathbf{R}}$$

- **Trade-off**: Segmentation reduces delay (pipelining) but adds overhead (headers).

必考 (Exam Focus)

Exam Focus: Binomial & Congestion [Source: Tutorial 1 Q4]

Scenario: N users, active prob $p = 0.1$. Link supports K users.

1. **正好 n 个用户在用** (Q4c Answer):

$$P(X=n) = \binom{N}{n} \cdot \underbrace{p^n}_{\text{active}} \cdot \underbrace{(1-p)^{N-n}}_{\text{inactive}}$$

Match option with p^n, e.g., $(0.1)^n(0.9)^{120-n}$ [Source: Tutorial 1 Q4c]

2. **Combinations Formula (组合数公式)**:

$$\binom{N}{n} = \frac{N!}{n!(N-n)!}$$

Example: From 4 choose 2 ($\binom{4}{2}$) $= \frac{4\times3\times2\times1}{(2\times1)\times(2\times1)} = 6$ [Source: Tutorial 1 Q4 Solution]

3. **拥堵的概率** (活跃人数 $> K$):

$$P(\text{congestion}) = \sum_{i=K+1}^{N} P(X=i) = \sum_{i=K+1}^{N} \binom{N}{i} p^i (1-p)^{N-i}$$

2. Delay Physics (延迟物理学 - Assgn 1 P6 重点)

1. Transmission vs. Propagation

$$d_{trans} = \frac{L}{R} \quad \text{(Size/Bandwidth)} \quad | \quad d_{prop} = \frac{d}{s} \quad \text{(Distance/Speed)}$$

必考 (Exam Focus)

Exam Focus: "Where is the bit?" [Assgn 1 P6]

Host A sends packet L to B (distance m, speed s, rate R). Start at $t = 0$.

- **At $t = d_{trans}$**: The **last bit** has just left Host A.
- **First Bit Position**: At time t, first bit is at distance $x = t \times s$.
- **Comparison**:
  - If $d_{prop} > d_{trans}$: First bit is on the wire, last bit has left A. (Link contains the whole packet).
  - If $d_{prop} < d_{trans}$: First bit has arrived at B, last bit is still at A. (Packet stretches across the link).
- **Equlibrium**: $d_{prop} = d_{trans} \Rightarrow \frac{m}{s} = \frac{L}{R}$.

3. Total Delay Calculation (综合计算)

1. The Four Sources Formula

$$d_{total} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Coach's Notes (避坑指南)

陷阱/易错点: Packetization Delay [Tutorial 2 Q1]

If converting Analog (Voice) to Digital:

- You cannot send bits one by one. You must **fill** a packet first.
- $\mathbf{d_{pack}} = \frac{\text{Packet Size (bits)}}{\text{Encoding Rate (bps)}}$
- Total Time $= d_{pack} + d_{trans} + d_{prop}$.

必考 (Exam Focus)

Exam Focus: Queuing Delay [Tutorial 2 Q4]

New packet arrives. One packet (L) is halfway done (x bits sent). n packets waiting.

$$d_{queue} = \underbrace{\frac{L-x}{R}}_{\text{Residual time}} + \underbrace{n \times \frac{L}{R}}_{\text{Waiting packets}}$$

4. Diagnostic Tools: Traceroute (网络诊断工具 - 必考)

必考 (Exam Focus)

Exam Focus: Traceroute/Tracert Mechanics [Source: Tutorial 2 Q5]

1. Mechanism (原理):

- Uses ICMP packets with incrementing **TTL** (Time To Live).
- $TTL = 1$: 1st router drops packet, sends "Time Exceeded" back. (Measures Hop 1).
- $TTL = 2$: 2nd router drops packet... (Measures Hop 2).

2. Output Columns (输出含义) [Q5a]:

- First column: Hop number (1, 2, 3...).
- Next 3 columns: **Three distinct RTT measurements** (measured in ms) for that hop. (Sends 3 probes per hop).
- Last column: Router Name/IP Address.

3. The Asterisk '*' (星号含义) [Q5b]:

- Means **Request Timed Out** (no response within default 5 sec).
- **Reason**: Packet lost OR (more likely) **Firewall** at that router blocks ICMP packets.

4. Delay Anomaly (反常延迟) [Q5c]:

- **Question**: Why is $RTT(\text{Router } N) > RTT(\text{Router } N+1)$? (Further is faster?)
- **Answer**: RTT includes **Queuing Delay**.
- Queuing is **variable** (stochastic). Hop N might be temporarily congested (high queuing) while Hop $N+1$ is idle later.

Part Supplement: 核心概念大白话讲解

1. Message Segmentation (报文分段) 到底快在哪里？(P31)

想象你要搬 1000 块砖去 3 公里外的工地，中间有 2 个中转站。

- **No Segmentation (整车搬)**: 你把 1000 块砖装一辆大卡车。开到中转站 A，卸货，检查，再装车，再开到 B... 必须等整车到了才能动。

- **Segmentation (车队搬)**：你把砖分给 10 辆小车（Packet）。
- ** 流水线效应 (Pipeline)**：第 1 辆小车刚从中转站 A 出发去 B，第 2 辆小车就可以立刻从起点出发去 A 了！大家都在路上跑，不用干等。
- ** 公式记忆 **：$(N + P - 1)L/R$。
  - $N \times L/R$：第 1 辆车跑完全程（穿过 N 段路）的时间。
  - $(P - 1) \times L/R$：因为是流水线，第 1 辆车到了之后，剩下 $P - 1$ 辆车会紧接着一个接一个到达（每隔 L/R 来一辆）。

2. 详解四种延迟与"Where is the bit?" [Source: Assignment 1 P6]

A. 四种延迟的物理意义 (The 4 Delays)

想象你要送一个车队（数据包）经过收费站（路由器）去往下一个城市：

1. d_{proc} **(处理延迟 - 检查证件)**：收费站大爷看一眼车牌、决定让你走哪条路的时间。每个交换机的处理延迟都要算。
2. d_{queue} **(排队延迟 - 堵车)**：你到收费站时，前面已经有别的车在排队了。你得等它们发完。特点：*这是网络卡顿的罪魁祸首，波动极大（取决于拥堵程度）。*
3. d_{trans} **(传输延迟 - 挤牙膏/过闸口)**：**关键！**这是把你的车队一辆接一辆推过收费站闸口的时间。
   - 车队越长（数据包 L 越大），推得越慢。
   - 闸口手脚越快（带宽 R 越大），推得越快。
   - **公式**：L/R。只跟"推"的动作有关，跟路多远没关系。
4. d_{prop} **(传播延迟 - 路上跑/飞行)**：**关键！**这是车队离开闸口后，在高速公路上狂奔到下一站的时间。
   - 路越远（距离 d 越大），跑得越久。
   - 车速越快（光速 s 越大），跑得越快。
   - **公式**：d/s。跟带宽 R 毫无关系。

B. 考题直觉：比特到底在哪里？[Source: Assignment 1 P6]

这题专门考 d_{trans} 和 d_{prop} 的物理区别。想象你在高空往地上**挤牙膏**：

- d_{trans}：你把整管牙膏从管子里挤出来的时间。
- d_{prop}：牙膏从空中掉到地上的时间。
- **Case 1:** $d_{prop} > d_{trans}$ **(高空作业)**：*场景*：离地很高（链路很长），或者你挤得巨快（带宽很大）。*结果*：你手里的牙膏已经全部挤完了（Last bit left sender），但第一滴牙膏还没落地（First bit hasn't reached receiver）。**整条数据链都悬在半空中（Link 上）。**
- **Case 2:** $d_{prop} < d_{trans}$ **(贴地作业)**：*场景*：离地很近（链路短），或者你挤得很慢（带宽小）。*结果*：你还在费劲地挤最后一点牙膏（Last bit at sender），第一滴早就掉地上了（First bit arrived）。**接收端已经开始收数据了，发送端还没发完。**

Chapter2: 应用层

---

1. Principles & Addressing

Architectures

- **Client-Server (C/S)**: Server always-on, fixed IP. Clients do not talk directly. Scalability limited by server bandwidth.
- **P2P**: No always-on server. Arbitrary end systems communicate. **Self-scalability** (New peers bring service capacity).

Process Addressing (Identifier)

Process ID = **IP Address** (Host) + **Port Number** (Process).

> **必考 (Exam Focus)**
>
> **Socket Identification (Tutorial 5 Q1/Q2)**
> - **UDP Socket**: Identified by **2-tuple**: (Dest IP, Dest Port).
> - **TCP Socket**: Identified by **4-tuple**: (Src IP, Src Port, Dest IP, Dest Port).
> - *Implication:* Web Server (Port 80) distinguishes clients via Src IP/Port. Different source IPs connect to same Dest Port 80 but different *sockets*.

2. Web and HTTP

HTTP Basics (基础概念)

HTTP is **Stateless** (无状态) and uses **TCP** (Port 80).

- **RTT (Round-Trip Time)**: 数据包往返一次所需的时间。
- **TCP Handshake (握手)**: 建立连接需要消耗 $1 \times RTT$。
- **Connection Types (连接类型)**:
  - **Non-Persistent (非持久)**: 1 object per TCP connection.
  - **Persistent (持久)**: Multiple objects per TCP connection.

HTTP Message Anatomy (实战：HTTP 报文解剖)

Scenario: Analyzing raw ASCII from Wireshark (Tutorial 3 Q1).

Raw Message

```
GET /cs453/index.html HTTP/1.1<cr><lf> Host: gaia.cs.umass.edu<cr><lf> User-Agent: Mozilla/5.0
...<cr><lf> Accept-Language: en-us,en;q=0.5<cr><lf> Connection: keep-alive<cr><lf> <cr><lf>
```

Line-by-Line Decoding (逐行解码)

1. **Request Line (请求行 - Line 1)**:
   - `GET`: **Method** (我要下载/获取资源)。
   - `/cs453...`: **Path** (资源路径)。**注意：**这不是完整 URL。
   - `HTTP/1.1`: **Version** (1.1 默认支持持久连接)。
2. **Header Lines (首部行 - Lines 2+)**:
   - **Host**: `gaia.cs.umass.edu` (目标主机)。
     - *Critical*: Web cache 和 Proxy 需要此信息来定位服务器。
   - **User-Agent**: 浏览器身份 (如 Chrome/Firefox)。服务器可据此返回适配内容 (Mobile vs Desktop)。
   - **Connection**:
     - `keep-alive` → **Persistent**: 保持连接，复用通道传后续数据。
     - `close` → **Non-persistent**: 传完即断开。

> **Coach's Notes (避坑指南)**
>
> **Trap: What's MISSING? (隐形考点)**
> - **Q1: Client IP?** (客户端 IP 在哪？)
>   **A: Unknown.** HTTP 是应用层协议（信纸），不包含网络层 IP 地址（快递单）。IP 在 IP Datagram 中。
> - **Q2: Full URL?** (完整地址是什么？)
>   **A**: 报文中只有路径。完整 URL = `http://` + **Host 字段** + **Request Path**。

HTTP Response Time Analysis (响应时间计算)

Ref: Tutorial 3 Q3. 核心考察 RTT 计算逻辑。

1. Cost Model (耗时模型)

设 RTT_0 为 Client 到 Server 的往返时间。

- **New Connection Cost** $= 2 \times RTT_0$
  - 解释：1 RTT (TCP 握手) + 1 RTT (HTTP 请求与响应)。
  - 适用：Non-Persistent 的每次请求，或 Persistent 的第一次请求。
- **Existing Connection Cost** $= 1 \times RTT_0$
  - 解释：连接已建立，只需 1 RTT (HTTP 请求与响应)。
  - 适用：Persistent 的后续对象请求。

2. Scenario Analysis (场景计算)

Task: Fetch 1 Base HTML + N Referenced Objects. (Total $N + 1$ items). **Pre-condition:** RTT_{DNS} is the total time for DNS resolution.

Scenario A: Non-Persistent Serial (串行)

- *Logic*: 每个对象都要新开连接，必须排队 (One by one)。
- *Formula*:

$$\text{Total} = RTT_{DNS} + \underbrace{2RTT_0}_{\text{Base HTML}} + \underbrace{N \times 2RTT_0}_{\text{N Objects}}$$

- *Ex*: If $N = 8$, Delay $= RTT_{DNS} + 18RTT_0$.

Scenario B: Non-Persistent Parallel (并行, k connections)

- *Logic*: 就像用 k 辆车运 N 箱货。需要运送 $\lceil N/k \rceil$ 趟 (Batches)。
- *Critical Step*: **Base HTML 必须先下载** (耗时 $2RTT_0$)，解析出 N 个链接后，才能开启并行下载。
- *Formula*:

$$\text{Total} = RTT_{DNS} + \underbrace{2RTT_0}_{\text{Base HTML}} + \underbrace{\lceil \frac{N}{k} \rceil \times 2RTT_0}_{\text{Parallel Batches}}$$

- *Ex*: If $N = 8, k = 5$. Batches $= \lceil 8/5 \rceil = 2$. Delay $= RTT_{DNS} + 2RTT_0 + 2(2RTT_0) = RTT_{DNS} + 6RTT_0$.

Scenario C: Persistent (持久连接, Pipelining)

- *Logic*: 握手一次，后续直接传。
- *Formula*:

$$\text{Total} = RTT_{DNS} + \underbrace{2RTT_0}_{\text{Base HTML}} + \underbrace{N \times 1RTT_0}_{\text{N Objects}}$$

- *Ex*: If $N = 8$. Delay $= RTT_{DNS} + 2RTT_0 + 8RTT_0 = RTT_{DNS} + 10RTT_0$.

必考 (Exam Focus)

Summary Cheat Sheet (必背公式) *Assumptions: N referenced objects.*

1. **Non-Persistent**: $2RTT_0(1+N)$
2. **Non-Persistent Parallel** (k): $2RTT_0(1+\lceil \frac{N}{k} \rceil)$
3. **Persistent**: $2RTT_0 + N \times RTT_0$

Note: All formulas assume DNS is already resolved or added separately.

3. DNS (Domain Name System)

Map Hostname (www.site.com) ↔ IP. UDP Port 53.

Hierarchy

Root → TLD (.com) → Authoritative (site.com).

- **Iterative**: "I don't know, ask him" (Server returns next server IP).
- **Recursive**: "I'll find out for you" (Server queries next on your behalf).

4. DNS Latency Calculation (Tutorial 3 Q2)

Definitions (定义)

- RTT_L: Round-Trip Time between **Client** ↔ **Local DNS**.
- RTT_r: RTT between **Local DNS** ↔ External Servers (**Root, TLD, Auth**).
- **Assumption**: Local DNS acts as a proxy performing **iterative** queries.

必考 (Exam Focus)

Problem 4: IP Access vs. Hostname Access

1. **Q:** Can you access the webpage by typing this IP? **A: No.**
   - **Key Concept: Virtual Hosting**.
   - **Explanation:** Many websites share the same IP. The web server distinguishes them using the **HTTP `Host` header**.
   - **Mechanism:** Typing the IP sets the Host header to the IP address. The server typically doesn't map the raw IP to the specific website configuration, resulting in a default page or error.

必考 (Exam Focus)

Scenario A: No Cache (Full Query) Local DNS must query the entire hierarchy.

1. **Client ↔ Local**: Request + Final Reply ($1 \times RTT_L$).
2. **Local ↔ Root**: Get TLD address ($1 \times RTT_r$).
3. **Local ↔ TLD**: Get Auth address ($1 \times RTT_r$).
4. **Local ↔ Auth**: Get IP address ($1 \times RTT_r$).

$$\text{Total Delay} = RTT_L + 3RTT_r$$

必考 (Exam Focus)

Scenario B: With Local Cache Local DNS has the IP mapping cached.

- Local DNS replies immediately without contacting external servers.
- Only Client-Local interaction is needed.

$$\text{Total Delay} = RTT_L$$

Coach's Notes (避坑指南)

Variant: Partial Cache (变种考点) Q: What if Local DNS caches the TLD server address but not the final IP? **A**:

- Skip Root ($0 \times RTT_r$).
- Must query TLD ($1 \times RTT_r$) + Auth ($1 \times RTT_r$).
- **Total**: $RTT_L + 2RTT_r$.

File Distribution: C-S vs. P2P

必考 (Exam Focus)

Exam Focus: 文件分发时间计算

Let F = 文件大小 (bits), N = 接收文件的 Peer 数量, u_s = 服务器上传速率 (upload rate of server), d_{min} = 最小的下载速率, u_i = 第 i 个 Peer 的上传速率.

1. Client-Server (C-S) Architecture: The server must upload N copies sequentially.

$$D_{CS} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

- If $u_s/N < d_{min}$: Server is bottleneck → Time is NF/u_s.
- If $u_s/N > d_{min}$: Slowest client is bottleneck → Time is F/d_{min}.

2. Peer-to-Peer (P2P) Architecture: Server uploads at least once; system capacity grows with N.

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum u_i}\right\}$$

- Term 1 (F/u_s): Server sends one copy (min requirement).
- Term 2 (F/d_{min}): Slowest peer download time.
- Term 3 ($\frac{NF}{u_{total}}$): System aggregate upload limit.

Coach's Notes (避坑指南)

Unit Conversion Trap!

Most problems give File Size in **Gbits** or **MBytes**, but Rates in **Mbps** or **Kbps**.

- Always convert everything to **bits** and **bits/sec** (or Mb and Mbps) first!
- 1 Gbit = 1000 Mbits = 10^9 bits (Network definitions usually use decimal 10^3, not 2^{10}, read question carefully).
- Example: $F = 15$ Gbits $= 15,000$ Mbits.

Case A: Server Bottleneck (瓶颈)

Condition: $\frac{u_s}{N} < d_{min}$

必考 (Exam Focus)

Standard Answer Template (背诵):

- **Bottleneck Identification**: Since $u_s/N < d_{min}$, the **bottleneck** is the server's upload capacity.
- **Scheme**: The server transmits the file to all N peers **simultaneously**, dividing its total upload rate u_s equally among them.
- **Justification**: Each peer receives data at a rate of u_s/N. Since $u_s/N < d_{min}$, every peer is capable of downloading at this rate.
- **Time**:

$$\text{Time} = \frac{F}{u_s/N} = \frac{NF}{u_s}$$

Case B: Client Bottleneck (瓶颈)

Condition: $\frac{u_s}{N} > d_{min}$

必考 (Exam Focus)

Standard Answer Template (背诵):

- **Bottleneck Identification**: Since $u_s/N > d_{min}$, the **bottleneck** is the peer with the minimum download rate (d_{min}).
- **Scheme**: The server transmits the file to each peer simultaneously at a rate of d_{min}.
- **Justification**: The total upload rate required from the server is $N \cdot d_{min}$. Since $u_s/N > d_{min}$ implies $u_s > N \cdot d_{min}$, the server has **sufficient bandwidth** to support this.
- **Time**:

$$\text{Time} = \frac{F}{d_{min}}$$

BitTorrent Protocol

Mechanisms

- **Tit-for-Tat**: Users reciprocate by uploading to peers who upload to them at the highest rates. (Prevents free-riding to some extent).
- **Choking**: Temporarily refusing to upload to a peer.
- **Optimistic Unchoking**: Periodically selecting a *random* peer to upload to, regardless of past performance.
  - Purpose: To discover better connections and let new peers (with no data) start downloading.

Tutorial 原题

- **Free-riding**: Can Bob download without uploading? **Yes**, but very slowly. He relies entirely on *Optimistic Unchoking* from others.
- **Sybil Attack**: Can Bob speed up free-riding? **Yes**. By creating multiple identities (different IPs), he increases the probability of being selected for *Optimistic Unchoking* by other peers.

Streaming Video (DASH)

DASH: Dynamic Adaptive Streaming over HTTP.

- **Concept**: Video is split into chunks; each chunk is encoded at multiple rates/qualities. Client requests chunks dynamically based on current bandwidth.
- **Storage Question**: How many files does the server store for N qualities?
  - **Mixed Audio/Video**: Audio & Video mixed in one file. $\Rightarrow$ **N files**.
  - **Separate Audio/Video**: Audio and Video stored separately (flexible). $\Rightarrow$ **2N files** (N video files + N audio files).

Part Supplement: 核心概念大白话讲解

1. P2P 为什么快？(The Intuition)

- **CS 模式**就像老师发卷子，只有老师一个人在发，学生越多，老师越累（时间 NF/u_s），发完所有人的时间线性增长。
- **P2P 模式**就像传八卦，老师告诉一个学生，这个学生转头告诉其他人。**每个人都在贡献上传带宽 **。人越多，帮忙传的人也越多，所以时间不会无限变长，而是趋于稳定。

2. 怎么算 P2P 时间？(The Formula) 不要死记硬背，看短板原理：

- **短板 1（发车）**：服务器至少得先把文件完整吐出来一次吧？$\rightarrow F/u_s$
- **短板 2（接收）**：最笨的那个学生（网速最慢）接收完要多久？$\rightarrow F/d_{min}$
- **短板 3（总水量）**：整个池子需要 $N \times F$ 的水，而所有水龙头（服务器 + 所有 Peers）加起来的出水速度是 u_{total}。灌满池子要多久？$\rightarrow NF/u_{total}$
- **取最大值**：这三个短板里最慢的那个，就是最后完成的时间。

3. BitTorrent 的 "吸血鬼" (Free-riding)

- 系统设计是"谁对我好，我对谁好"（Tit-for-Tat）。
- 但是为了给新人机会，系统会每隔一会"随机选个幸运儿"送数据（Optimistic Unchoking）。
- **Bob 想白嫖**：他不上传，正经途径拿不到数据，只能等着被当成"幸运儿"。
- **Bob 开挂**：他开 100 个小号，被随机砸中当幸运儿的概率就大了，这就是 Sybil Attack。

5. Electronic Mail

- **SMTP**: Push. Client to Server, or Server to Server. Persistent. ASCII.
- **POP3**: Pull. Download & Delete (stateless) or Keep.
- **IMAP**: Pull. Complex, keeps state (folders) on server.

Chapter3: 传输层

Multiplexing & Demultiplexing (Socket Identification)

UDP vs. TCP Socket Identification

- **UDP Socket (Connectionless)**:
  - Identified by a **2-tuple**: (Destination IP, Destination Port).
  - **Behavior**: Packets with the *same* Dest IP and Dest Port are directed to the **same** socket, regardless of Source IP/Port.
- **TCP Socket (Connection-Oriented)**:
  - Identified by a **4-tuple**: (Source IP, Source Port, Destination IP, Destination Port).
  - **Behavior**: A Web Server (Port 80) creates a **unique socket** for each distinct connection. Even if two packets target Port 80, if their Source IP or Source Port differ, they go to different sockets.

Coach's Notes (避坑指南)

陷阱/易错点 (Exam Trap):

- **Q**: 两个不同的客户端 (Host A, Host C) 向同一个 Web 服务器 (Host B, Port 80) 发送 HTTP 请求，它们会进入同一个 Socket 吗？
- **A**: 不会。因为 HTTP 使用 TCP，TCP Socket 由 4-tuple 标识。Host A 和 Host C 的 Source IP 不同，因此服务器会分流 (Demultiplex) 到两个不同的进程/线程中处理。
- **注意**: 返回的数据包 (Reply Segment) 会交换源和目的。例如，Server 回复 Host A 时，Source Port = 80, Dest Port = A's ephemeral port.

必考 (Exam Focus)

Exam Focus: Scenario Analysis (From Problem 2)

- **Context**: Host C (IP: C) runs two browser windows.
- **Process 1**: Source Port 7532 $\rightarrow$ Server B (IP: B), Port 80.
- **Process 2**: Source Port 26145 $\rightarrow$ Server B (IP: B), Port 80.
- **Server Response**:
  - To Process 1: Src Port: 80, Dst Port: 7532, Dst IP: C.
  - To Process 2: Src Port: 80, Dst Port: 26145, Dst IP: C.
- **Key**: The Source Port distinguishes the two processes on the same host.

互联网校验和 (Error Detection)

Calculation Method (1s Complement Sum)

Mechanism: Treat data as a sequence of 16-bit (or 8-bit in tutorial) integers.

1. **Sum**: Add integers using standard binary addition.
2. **Wraparound**: If there is a carry out of the MSB (Most Significant Bit), add 1 to the result.
3. **Checksum**: Take the **1s complement** (flip all bits) of the final sum.

- **检测错误**: The receiver adds all received data words plus the received Checksum.
  - If the result is **all 1s** ($11\dots1$, which is -0 in 1s complement), the data is considered valid.
  - If the result contains any **0**, an error is detected.

错误检测能力分析 (From Tutorial 5 Q3)

- **1-bit Error: Always Detected**.
  - If a single bit flips ($0 \rightarrow 1$ or $1 \rightarrow 0$), the total sum will change. The new sum added to the original checksum will no longer result in all 1s.
- **2-bit Error: Possible to Go Undetected**.
  - If two errors occur that "cancel each other out" during the summation, the error will be missed.
  - *Example*: If one bit in a column flips $0 \rightarrow 1$ and another bit in the **same column** (but different word) flips $1 \rightarrow 0$, the sum for that column remains unchanged.
  - All one-bit errors will be detected, but two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

必考 (Exam Focus)

Calculation Example (From Problem 3): Sum three 8-bit bytes: $00100011, 01001110, 01010100$.

1. Add first two: $00100011 + 01001110 = 01110001$
2. Add third: $01110001 + 01010100 =$ **11000101** (Sum)
3. Checksum (flip bits): $\sim 11000101 =$ **00111010**

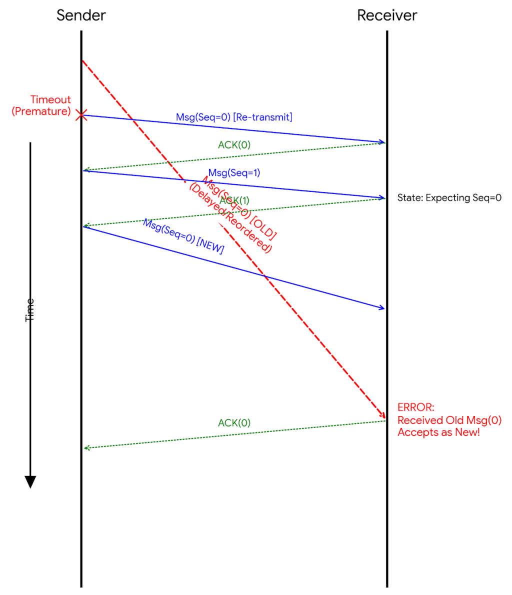
RDT 3.0 Failure under Packet Reordering

The Problem Scenario

- **Question**: If the network can **reorder messages** (not just lose them), will RDT 3.0 (Alternating-Bit Protocol, Seq 0/1) work?
- **Answer**: **No, it fails.**
- **Reason**: The protocol relies on 1-bit sequence numbers (0 and 1). It assumes FIFO (First-In-First-Out). A delayed "Old Packet 0" cannot be distinguished from a "New Packet 0" if they arrive out of order.

Failure Trace Analysis

The failure is triggered by a **Premature Timeout** caused by network delay (packet reordering).



必考 (Exam Focus)

Exam Focus: Failure Step-by-Step (Tutorial 5 Problem 4)

Assume Sender is on left, Receiver on right (as shown in the figure above).

1. **Send Old M0**: Sender transmits packet $M0_{old}$. It gets **delayed** (stuck in network), but NOT lost.
2. **Premature Timeout**: Sender timer expires. Sender thinks M0 is lost.
3. **Resend M0**: Sender retransmits $M0_{rex}$.
4. **Normal Exchange**:
   - Receiver gets $M0_{rex}$, sends $ACK0$.
   - Sender gets $ACK0$, sends $M1$.
   - Receiver gets $M1$, sends $ACK1$.
5. **Send New M0**: Sender gets $ACK1$, sends next packet $M0_{new}$ (Seq wraps back to 0).
6. **The Failure**: The original delayed $M0_{old}$ finally arrives at Receiver.
7. **Receiver Error**:
   - Receiver is waiting for Seq 0 (the $M0_{new}$).
   - It sees $M0_{old}$ (Seq 0).
   - **Outcome**: Receiver **accepts old data as new data**. Protocol fails.

Coach's Notes (避坑指南)

陷阱/易错点 (Why 1-bit Seq is not enough):

- **误区**: 认为有 ACK 和 Timeout 就能保证可靠。
- **真相**: ACK/Timeout 解决的是**丢包**问题。
- 面对**乱序 (Reordering)**, 必须有足够大的序号空间 (Sequence Number Space) 来区分不同轮次的同一个序号（例如区分 "第 1 轮的包 0" 和 "第 2 轮的包 0"）。
- RDT 3.0 只有 1 bit (0/1)，无法区分，所以会把旧包当新包。

两种核心协议 (GBN vs SR)

Mechanism Comparison

- **Go-Back-N (GBN)**:
  - **ACK Type**: **Cumulative ACK**. $ACK(n)$ 表示序号 n 及之前的所有包都已正确接收.
  - **Receiver Behavior**: 只接收按序到达的包。如果中间有包丢失（如 pkt 6 丢了），后续到达的包（pkt 7）会被直接丢弃（Discard），并重复发送对最后一个按序到达包的 ACK（即 ACK 5）.
  - **Sender Behavior**: 一旦超时, 重传所有已发送但未确认的包 (即 "回退 N 步")
- **Selective Repeat (SR)**:
  - **ACK Type**: **Individual ACK**. 接收端会对每个正确接收的包单独发送 ACK.
  - **Receiver Behavior**: 如果 pkt 6 丢了，pkt 7 到达，接收端会缓存（Buffer）pkt 7 并发送 ACK 7。
  - **Sender Behavior**: Maintains a timer for *each* packet. On timeout, retransmits **ONLY** the lost packet.（只重传**超时未收到 ACK** 的那一个包（如只重传 pkt 6）

必考 (Exam Focus)

Exam Focus: Packet Loss Scenario (Problem 1)

Scenario: Window size $N = 8$. Packets 0, 1, ..., 7 sent. **Packet 6 is lost**.

1. **GBN Outcome**:
   - Receiver gets 0-5: Sends ACK 0-5.
   - Receiver gets 7: **Discards** 7 (out-of-order). Re-sends **ACK 5** (Last correct in-order).
   - Sender receives: ACK 0, 1, 2, 3, 4, 5, **5, 5** (Duplicate ACKs).
2. **SR Outcome**:
   - Receiver gets 0-5: Sends ACK 0-5.
   - Receiver gets 7: **Buffers** 7. Sends **ACK 7**.
   - Sender receives: ACK 0, 1, 2, 3, 4, 5, **7**. (Missing ACK 6).

Sequence and Acknowledgement Numbers

Definitions

- **Sequence Number (Seq)**: The byte-stream number of the **first byte** in the segment.
- **Acknowledgement Number (ACK)**: The sequence number of the **next byte** expected by the receiver.
- **Calculation**: $Next\_Seq = Current\_Seq + Data\_Size$.

Scenario Analysis (From Tutorial 6 Q2)

- **Setup**: Host B has received bytes up to 126. Next expected is 127.
- **Transmission**:
  - **Seg 1**: Seq 127, Size 80 bytes. (Range: 127-206).
  - **Seg 2**: Seq 207, Size 40 bytes. (Range: 207-246).

必考 (Exam Focus)

Exam Focus: Timeout & Delayed ACK (Problem 2d)

Scenario: Seg 1 & 2 arrive at B. B sends ACK 207 & ACK 247. **ACK 207 is lost**. A times out.

(场景：Seg 1 和 2 到达 B。B 发送 ACK 207 和 247。ACK 207 丢失。A 发生超时。)

1. **Timeout at A (A 发生超时)**: Since ACK 207 is lost and ACK 247 hasn't arrived, A times out on Seg 1.
   (由于 ACK 207 丢失且 ACK 247 未到达，A 对 Seg 1 触发超时。)
2. **Retransmission (重传)**: A re-sends **Seg 1** (Seq 127). (TCP retransmits oldest unacked).
   (A 重传 **Seg 1**。TCP 总是重传最早未被确认的段。)
3. **Late ACK Arrives (迟到的 ACK)**: ACK 247 arrives at A.
   (ACK 247 到达 A。)
4. **Resolution (解决机制)**:
   - Because TCP uses **Cumulative ACK**, ACK 247 implies "I have everything up to 247".
     (因 TCP 使用 ** 累积确认 **，ACK 247 意味着"247 之前的数据全收到了"。)
   - Sender A marks both Seg 1 and Seg 2 as acknowledged. No further retransmissions.
     (A 将 Seg 1 和 Seg 2 都标记为已确认。A 不会再重传 Seg 2。)
5. **Receiver's Response (接收端响应)**: B receives retransmitted Seg 1. It discards data (duplicate) but sends **ACK 247** (current expectation).
   (B 收到重传的 Seg 1。丢弃重复数据，但再次发送 **ACK 247** 告知当前期望序号。)

Coach's Notes (避坑指南)

陷阱/易错点 (Cumulative ACK Power):

- **误区**: 以为 ACK 207 丢了，Host A 就永远不知道 Seg 1 到了。
- **真相**: 只要后续的 ACK 247 到了，它就**覆盖**了 ACK 207 的功能。累计确认意味着："只要我收到了大的 ACK，小的 ACK 丢了也没事"。

超时计算公式 (From Tutorial 6 Q4)

The Three-Step Algorithm

You must calculate these in exact order for every new sample.

1. **更新 EstimatedRTT (相当于" 移动平均线"，反映正常水平)**:

$$EstimatedRTT_{new} = (1 - \alpha) \cdot EstimatedRTT_{old} + \alpha \cdot SampleRTT$$

(Typically $\alpha = 0.125$)

2. **更新 DevRTT (相当于" 波动率"，反映网络抖动)**:

$$DevRTT_{new} = (1-\beta) \cdot DevRTT_{old} + \beta \cdot |SampleRTT - EstimatedRTT_{new}$$

*(Typically $\beta = 0.25$. Note: Use the **NEW** EstimatedRTT here!)*

3. **更新 TimeoutInterval (意思是不怕慢，就怕慢得离谱。只有慢出 4 倍安全区，才算超时)**：

$$TimeoutInterval = EstimatedRTT_{new} + 4 \cdot DevRTT_{new}$$

必考 (Exam Focus)

Calculation Trace (Problem 4):
Given: Initial Est=100, Dev=5. $\alpha = 0.125, \beta = 0.25$.
Sample 1: 106 ms
- $Est = 0.875(100) + 0.125(106) = 100.75$ ms
- $Dev = 0.75(5) + 0.25|106 - 100.75| = 5.06$ ms
- $Timeout = 100.75 + 4(5.06) = 121$ ms

Sample 2: 120 ms (Use values from Sample 1 as Old)
- $Est = 0.875(100.75) + 0.125(120) = 103.16$ ms
- $Dev = 0.75(5.06) + 0.25|120 - 103.16| = 8.01$ ms
- $Timeout = 103.16 + 4(8.01) = 135.2$ ms

Part Supplement: 核心概念大白话讲解

- **GBN vs SR (快递员的性格)**:
  - **GBN (强迫症)**: " 包裹 6 丢了？那我手里拿到的包裹 7 也不要了，你把 6 和 7 都给我重发一遍！" (丢弃乱序，累计确认)。
  - **SR (好说话)**: " 包裹 6 丢了？没事，包裹 7 我先收下存着。你只把 6 补给我就行。" (缓存乱序，独立确认)。
- **TCP 的累计确认 (Cumulative ACK)**:
  - 就像闯关游戏。ACK 247 意思是" 第 247 关之前的我都通关了"。
  - 哪怕你没收到" 通关第 207 关" 的提示 (ACK 丢失)，只要看到" 通关第 247 关" 的提示，你就知道前面的肯定都过了。

TCP Congestion Control (拥塞控制核心)

1. Phases & Transitions (状态流转)

TCP 通过调整拥塞窗口 (**cwnd**) 来控制发送速率。

- **Slow Start (SS)**:
  - **特征**: 指数增长 (Exponential)，每收到一个 ACK, cwnd += 1 MSS (每 RTT 翻倍)。
  - **条件**: $cwnd < ssthresh$。
  - **目的**: 刚开始或重置后，快速拉升占用带宽。
- **Congestion Avoidance (CA)**:
  - **特征**: 线性增长 (Additive Increase)，每过一个 RTT，cwnd += 1 MSS。
  - **条件**: $cwnd \geq ssthresh$。
  - **目的**: 接近瓶颈时小心试探。
- **Fast Recovery (FR)** (仅 Reno):
  - **特征**: 在收到 3 个重复 ACK 后进入。此时 cwnd 保持较高水平 (ssthresh + 3)，每收到一个重复 ACK，cwnd 还会临时增加 (模拟数据包守恒)，直到新 ACK 到达退出 FR 进入 CA。

2. Reno vs Tahoe (必考区别)

核心考点: 发生丢包时，两者的反应不同。请务必区分 **Timeout** 和 **3 Duplicate ACKs** 两种场景。

Scenario A: Timeout (严重拥塞)

场景描述: ACK 完全没回来，通常意味着网络严重堵塞。

- **TCP Tahoe & Reno (行为一致)**:
  - **ssthresh**: 设为当前 $cwnd/2$
  - **cwnd**: **重置为 1 MSS**
  - **状态流转**: 进入 **Slow Start**

Scenario B: 3 Duplicate ACKs (轻微拥塞)

场景描述: 收到重复 ACK，说明有后续包到达 (网络还能通)，只是中间丢了一个。

- **TCP Tahoe (激进重置)**:
  - **ssthresh**: 设为当前 $cwnd/2$
  - **cwnd**: **重置为 1 MSS**
  - **状态流转**: 进入 **Slow Start**
- **TCP Reno (快速恢复)**:
  - **ssthresh**: 设为当前 $cwnd/2$
  - **cwnd**: **设为 ssthresh + 3** (减半后膨胀)
  - **状态流转**: 进入 **Fast Recovery** (随后进入 CA)

Sequence & ACK Calculation (计算题)

1. Sequence Number Logic

TCP 是字节流协议，Sequence Number 指的是**报文段中第一个字节的编号**。

Coach's Notes (避坑指南)

True/False Trap (Tutorial 8 Q1):
1. **Truth**: Next Seq = m + Data Length. (TCP 计数基于字节，不是包个数！)
2. **Truth**: Host B sends empty ACK segments if no data is available. (必须确认)

必考 (Exam Focus)

Scenario: Seg 1 (Seq=65) and Seg 2 (Seq=92) sent back-to-back.
- **Q: Size of Seg 1?**
- **A**: $Seq_2 - Seq_1 = 92 - 65 = \mathbf{27}$ **bytes**.

2. ACK Number Logic (Cumulative ACK)

ACK 表示接收方**期待接收的下一个字节**。如果中间有丢包，ACK 会卡在第一个丢失字节的编号上。

必考 (Exam Focus)

Scenario: Seg 1 (Seq=65, Size=27) is LOST. Seg 2 (Seq=92) ARRIVES.
- **Receiver State**: 收到了 92，但 65 还没到。
- **Logic**: 只能确认连续数据的末尾。
- **ACK Value**: **65** (一直喊 "我要 65")。
- **Note**: 即使 Seg 2 到了，也不能 ACK 92+Length，必须 ACK 65。

Performance & Utilization (窗口计算)

Utilization Formula

要求信道利用率 (Utilization) 达到一定比例，求最小窗口 N。
Formula:

$$U = \frac{\text{Time to send N packets}}{\text{Total Cycle Time}} = \frac{N \times (L/R)}{RTT + (L/R)}$$

- N: Window Size (packets)
- L/R: Transmission Delay (Time to push one packet)
- RTT: Round Trip Time

Coach's Notes (避坑指南)

Calculation Steps (Tutorial 7 P4):
1. **Unit Conversion**: 将 Packet Size 换算成 bits，将 L/R 换算成 msec (与 RTT 统一)。
$$L = 1200 \text{ bytes} = 9600 \text{ bits}$$
$$d_{trans} = \frac{9600}{10^9} = 9.6\mu s = 0.0096 \text{ ms}$$
2. **Solve Inequality**:
$$\frac{N \times 0.0096}{30 + 0.0096} \geq 0.97 \implies N \geq 3032.22$$
3. **Round Up**: $N = 3033$。

Fairness (公平性)

TCP vs UDP

- **UDP**: 无拥塞控制，不退让，可能挤占带宽。
- **TCP**: 有拥塞控制 (AIMD)，理论上趋向于 R/N 的公平分配。
- **Key Question**: Is TCP fairer? **NO**.
- **Reason**: TCP 应用可以开启 **Parallel Connections** (如 Web 浏览器开 10 个连接下载图片)。即使每个连接都退让，总带宽占用依然比单连接用户大得多。

Part Supplement: 核心概念大白话讲解

- **关于 Tahoe vs Reno**:
  - **Tahoe (老实人)**: 不管是超时还是 3 个重复 ACK，一律心态崩盘，cwnd 回到 1，慢启动重来。

– **Reno (聪明人)**: 如果是 3 个重复 ACK (说明网络还能传 ACK，没死透)，cwnd 减半，然后直接进入拥塞避免 (加法增长)，这叫 "快速恢复"。只有超时 (彻底死透) 才回 1。

- **关于 Sequence Number**:
  – 就像书的页码。第一个包是第 1-100 页 (Seq=1)，第二个包是第 101-200 页 (Seq=101)。
  – 如果你收到 Seq=101 的包，你就知道第一个包肯定是 $101 - 1 = 100$ 字节长。
- **关于 Pipeline 利用率**:
  – 想象一条传送带。停止等待协议就是放一个包裹，等它到了终点拿回回执再放下一个，中间全是空的。
  – Pipeline 就是要一次性放 N 个包裹，铺满传送带。公式就是算出这 N 个包裹的总长度除以传送带循环一次的时间。

Chapter4: 网络层

Fragmentation Parameters

Due to Link Layer MTU (Maximum Transmission Unit) limits, large IP datagrams must be split.

- **Identification (ID)**: Same for all fragments of one original packet. (同一个原始包的所有分片 ID 相同)
- **Flags (MF)**: 1 = More fragments follow; 0 = Last fragment.
- **Fragment Offset**: Position of the fragment data, divided by 8.

必考 (Exam Focus)

Calculation Steps (Tutorial 8 P2): Given: Datagram=1600B (20B Header + 1580B Data), MTU=500B.

1. **Calculate Max Payload**:
$$\text{MaxData} = MTU - Header = 500 - 20 = 480 \text{ bytes}$$
2. **Check 8-Byte Rule**:
$$480 \mod 8 = 0 \quad \text{(Perfect! If not, round down to nearest 8)}$$
3. **Calculate Number of Fragments**:
$$\lceil 1580/480 \rceil = 4 \text{ fragments}$$
4. **Calculate Offset for k-th fragment**:
$$\text{Offset}_k = \frac{(k-1) \times 480}{8}$$
Offsets: $0, 60, 120, 180$.

Coach's Notes (避坑指南)

Why Divide by 8?

- The Offset field in IP Header is only **13 bits**, but Total Length is **16 bits**.
- Hardware constraint requires a scaling factor of $2^3 = 8$.
- **Consequence**: Payload of all non-last fragments **must** be divisible by 8.

Part Supplement: 核心概念大白话讲解

- **关于分片 (Fragmentation)**:
  – 就像搬家。你有一个 1600 斤的大柜子（Datagram），但电梯（MTU）一次只能运 500 斤。
  – 你必须把柜子拆成几块。每块除了木头（Payload），还要包上防撞角（Header，20 斤）。
  – 所以每次只能运 480 斤木头。
- **关于 Offset 为什么除以 8**:
  – 头部里的 "位置记录本"（Offset 字段）格子太小写不下大数字。所以约定：记录本上写 "1"，代表实际的 "8"。填表的时候千万别忘了除以 8！

Longest Prefix Match (最长前缀匹配)

When forwarding a packet, the router selects the entry with the **longest matching subnet mask**. 当一个 IP 地址同时匹配转发表中的多条记录时，路由器会选择 ** 掩码最长（匹配的位数最多）** 的那一条作为转发接口。

Rule: More specific routes (longer prefix) > General routes (shorter prefix).

必考 (Exam Focus)

Handling "Holes" in Ranges (Tutorial 8 P4): How to route a range like Link 2 (`...0001` to `...1111`) while excluding `...0000`?

1. **Strategy**: Do NOT break Link 2 into tiny pieces.
2. **Step 1**: Define Link 2 generally (e.g., prefix `11100001` -> Interface 2).
3. **Step 2**: Define the "Hole" / Exception specifically (e.g., prefix `11100001 00000000` -> Interface 1 or Default).
4. **Logic**: The router will automatically match the "Hole" to the specific rule (Longer Prefix) and the rest to the general rule.

Part Supplement: 核心概念大白话讲解

- **最长前缀匹配 (LPM)**:
  – 就像快递分拣。
  – 规则 A："北京的快递" -> 放左边。
  – 规则 B："北京海淀的快递" -> 放右边。
  – 这是一个 "北京海淀" 的快递，虽然它也属于北京，但规则 B 描述得更精确（前缀更长），所以必须按规则 B 走。
- **打补丁策略**:
  – 如果你想表达 "除了小明，全班都出去"。
  – 不需要列出除了小明外的所有名字。
  – 只需要两条规则：1. "小明 -> 留下"；2. "全班 -> 出去"。
  – 因为 "小明" 比 "全班" 更具体，LPM 原则会保证小明被留下来。

链路状态路由 (Link-State Routing) - Dijkstra 算法【Centralized】

Link-State (LS) routing algorithms require global network knowledge (topology and link costs) available at every node. **Dijkstra's Algorithm** is the standard implementation.

- **前提**: 网络拓扑和所有链路开销 (Link Costs) 对所有节点都是已知的。通常通过 Link State Broadcast 实现
- **Goal**: Compute least-cost paths from one source node to all other destinations.
- **Notation**:
  – $c(x, y)$: 节点 x 到 y 的链路开销。如果不相邻则为 ∞。
  – $D(v)$: 当前从源节点到节点 v 的路径开销估值
  – $p(v)$: 路径中 v 的前驱节点 (Predecessor)。
  – N': 已找到确定最短路径的节点集合。

Dijkstra's Iteration Steps

1. **Initialization**: $N' = \{u\}$ (source). For all neighbors v, $D(v) = c(u, v)$. Others ∞.
2. **Loop**:
   - Find w not in N' with the **minimum** $D(w)$.
   - Add w to N'.
   - **Update neighbors**: For each neighbor v of w (not in N'):
$$D(v) = \min(D(v),\ D(w) + c(w, v))$$
   - **Logic:** New cost = Cost to get to w + Cost from w to v.

必考 (Exam Focus)

Exam Focus: Running Dijkstra (Tutorial 9 P1) Task: Compute shortest path from node **x**. **Step-by-Step Table**:

Step	N'	$D(v), p(v)$	$D(u), p(u)$	$D(w), p(w)$	$D(y), p(y)$	$D(z), p(z)$
0	x	3,x	∞	6,x	6,x	8,x
1	xv	**3,x**	7,v	6,x	6,x	8,x
2	xvu	3,x	**7,v**	6,x	6,x	8,x
3	xvuw	3,x	7,v	**6,x**	8,x	8,x
...	...	...	...	...	...	...

Key Selection: In Step 1, node v has cost 3, node w has 6. Since $3 < 6$, we pick v into N' first. **Update**: After picking v (cost 3), we check neighbor u. Old cost to u was ∞. New cost via v is $D(v) + c(v, u) = 3 + 4 = 7$. Since $7 < \infty$, update $D(u) = 7, p(u) = v$.

Coach's Notes (避坑指南)

Dijkstra Pitfalls (易错点):

- **Not updating correctly**: Always check if $D(w) + c(w, v)$ is smaller than the current $D(v)$.
- **Tie-breaking**: If two nodes have the same minimum cost, you can pick either (usually lexicographical order, e.g., u before w).
- **Oscillations**: In link-state routing, if link costs depend on traffic volume, routes can oscillate (flip-flop) rapidly.

距离向量路由 Distance-Vector Routing) - Bellman-Ford【Distributed】

Distance-Vector (DV) is iterative, asynchronous, and distributed. Nodes only know costs to direct neighbors and receive "vectors" (estimates) from them.

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

- $d_x(y)$: 节点 x 到 y 的最短路径开销。.
- $c(x, v)$: x 到邻居 v 的直接链路开销.
- $d_v(y)$: 邻居 v 报告的它到 y 的距离 (received via gossip).

必考 (Exam Focus)

Exam Focus: DV Table Update (Tutorial 9 P2 & P3) Scenario: Node x has neighbors w (cost 2) and y (cost 5). **Given from neighbors**:

- w reports: "I can reach u in cost 5". ($d_w(u) = 5$)
- y reports: "I can reach u in cost 6". ($d_y(u) = 6$)

Calculation:

$$d_x(u) = \min\{\underbrace{c(x, w) + d_w(u)}_{2+5=7}, \quad \underbrace{c(x, y) + d_y(u)}_{5+6=11}\} = 7$$

Result: x sets its distance to u as 7, and next-hop is w.

Coach's Notes (避坑指南)

Count-to-Infinity Problem:

- **Good news travels fast**: A decrease in link cost propagates quickly.
- **Bad news travels slow**: If a link breaks or cost increases, nodes may bounce incorrect data back and forth, slowly incrementing costs to infinity.
- **Solution**: **Poisoned Reverse**. If z routes through y to get to x, z tells y that its distance to x is ∞ (so y won't route back through z).

Comparison: Link-State vs Distance-Vector

- **1. Network Knowledge (视野范围)**
  - **Link-State (LS)**: **Global**. Each node has a complete map of the entire network topology.
  - **Distance-Vector (DV)**: **Local**. Nodes only know the "distance" to their direct neighbors.
- **2. Communication Strategy (沟通方式)**
  - **LS**: **Broadcast**. Link status info is flooded to **all** nodes in the network.
  - **DV**: **Exchange**. Routing tables are sent only to direct **neighbors**.
- **3. Convergence & Issues (收敛速度)**
  - **LS**: **Fast** and deterministic.
  - **DV**: **Slow**. Can suffer from **Routing Loops** and the "Count-to-Infinity" problem.
- **4. Complexity (复杂度)**
  - **LS**: Message complexity $O(N|E|)$; Computation time $O(N^2)$ (using Dijkstra).
  - **DV**: Varies greatly depending on network changes.
- **5. Robustness (健壮性)**
  - **LS**: **High**. A router acts independently; bad calculations are contained locally.
  - **DV**: **Low**. Errors propagate widely because nodes blindy trust neighbors ("Gossip" effect).

Tutorial 10: IP Subnetting & Routing Protocols

1. IP Subnetting & VLSM (CIDR)

CIDR (Classless Inter-Domain Routing) allows allocating IP addresses more efficiently using Variable Length Subnet Masks (VLSM).

- **CIDR Notation**: $a.b.c.d/x$, where x is the **Network Prefix length**.
- **Block Size**: 2^{32-x} (Total IP addresses in the subnet).
- **Usable Hosts**: $2^{32-x} - 2$ (Minus Network Address & Broadcast Address).
- **Alignment Rule**: The starting address of a subnet must be divisible by its **Block Size**.
- **分配子网时可选的块大小只有：2, 4, 8, 16, 32, 64, 128, 256....**

Coach's Notes (避坑指南)

陷阱: The "Just Missed It" Case (15 Hosts)

- **Scenario**: You need to support **15 hosts**.
- **Calculation**: $15 + 2 = 17$ IPs needed.
- **Block Size 16 (/28)**:
  - Total IPs: 16.
  - Usable Hosts: $16 - 2 = 14$.
  - **Result**: Not enough! ($14 < 15$).
- **Block Size 32 (/27)**:
  - Total IPs: 32.
  - Usable Hosts: $32 - 2 = 30$.
  - **Result**: Correct. (Even though wasteful).
- **Rule of Thumb**: If required hosts $> 2^k - 2$, jump to 2^{k+1}.

必考 (Exam Focus)

Subnet Allocation (Tutorial 10 P1): Given: Base Block 223.1.17.0/24. Subnet 1 (62 hosts) already occupies .0 - .63 (/26).

Goal A: Allocate Subnet 2 (106 hosts)

1. **Calculate Size**: Need $N \geq 106 + 2 = 108$.

$$2^k \geq 108 \implies k = 7 \quad \text{(Host bits)}$$

2. **Determine Mask**: $32 - 7 =$ **/25**. Block Size = 128.
3. **Find Position**:
   - Block [0-127]: Overlaps with Subnet 1 (0-63). **Conflict!**
   - Block [128-255]: Free.
4. **Result**: **223.1.17.128/25**.

Goal B: Allocate Subnet 3 (14 hosts)

1. **Calculate Size**: Need $N \geq 14 + 2 = 16$.

$$2^k \geq 16 \implies k = 4 \quad \text{(Host bits)}$$

2. **Determine Mask**: $32 - 4 =$ **/28**. Block Size = 16.
3. **Find Position**:
   - Range 0-63 (Subnet 1) & 128-255 (Subnet 2) are taken.
   - Free space: 64-127. First available multiple of 16 is **.64**.
4. **Result**: **223.1.17.64/28**.

Coach's Notes (避坑指南)

Pitfalls in Subnetting:

- **The "+2" Rule**: Always add 2 for Network ID and Broadcast ID. If asked for 63 hosts, $63 + 2 = 65$, so you need size 128 (/25), NOT 64 (/26).
- **Alignment**: A /25 subnet (size 128) cannot start at .64. It MUST start at .0 or .128.
- **No Overlap**: Draw a number line to ensure assigned ranges do not cross.

2. Routing Hierarchy (AS & BGP)

The internet is divided into **Autonomous Systems (AS)**. Different protocols are used inside vs. between ASes.

- **Intra-AS Routing (IGP)**: Protocols inside an AS (e.g., **OSPF, RIP**). Focus on **Performance** (shortest path).
- **Inter-AS Routing (EGP)**: Protocols between ASes (e.g., **BGP**). Focus on **Policy** (political/economic control) and **Scale**.
- **eBGP (External)**: Connects border routers of **different** ASes.
- **iBGP (Internal)**: Propagates external info to routers within the **same** AS.

必考 (Exam Focus)

Determining Learning Protocol (Tutorial 10 P3): Scenario: Traffic destined for prefix x (located in AS4). Path: AS4 → AS3 → AS1.

1. **AS4 Border → AS3 Border (3c)**:
   - Communication between **Different ASes**.
   - Protocol: **eBGP**.
2. **AS3 Border (3c) → AS3 Internal (3a)**:
   - Propagating external info (x) within the **Same AS**.
   - Protocol: **iBGP**.
   - *Note: OSPF/RIP is used here only to find the path to 3c, not to learn about x itself.*
3. **AS3 Border (3a) → AS1 Border (1c)**:
   - Communication between **Different ASes**.
   - Protocol: **eBGP**.

Part Supplement: 核心概念大白话讲解

- **关于切蛋糕 (Subnetting)**:
  – **+2 原则**: 就像订酒店房间，如果你带了 106 个人，不能只开 106 间房，因为头尾两间房（网络号、广播号）被锁住了不能住人。所以你要订 128 间的大套房。
  – **对齐原则**: 大套房（/25）只能建在整层楼的起点（.0 或.128），不能建在楼道中间（比如.64）。
- **关于外交 (BGP)**:
  – **AS (自治系统)**: 把 AS 想象成一个独立的“国家”。
  – **Intra-AS (OSPF)**: 国内的导航软件（高德/百度），只管怎么在国内走得最快。
  – **eBGP**: 两国边境的“外交官”，负责交换信息（“去美国走我这边”）。
  – **iBGP**: 国内的新闻联播，负责把外交官听到的消息告诉国内的老百姓（“外交官说，去美国要往东边走”）。

Chapter5: 数据链路层

2D Parity Check (二维奇偶校验)

Extension of single-bit parity. Data is arranged in a matrix to detect and correct errors.

- **Detection Capability**: Can detect 2-bit errors (and any odd number of errors).
- **Correction Capability**: Can **correct** single-bit errors by locating the intersection of the bad row and bad column.
- **Corner Bit**: The parity of the row-parity column and column-parity row must match.

必考 (Exam Focus)

Calculation Steps (Tutorial 11 P1): Given: Data ‘1110 0110 1001 1101‘ (16 bits), Even Parity. Find the checksum field.

1. **Arrange as Matrix**: Split 16 bits into 4×4.
2. **Calculate Parities**:

$$\begin{array}{cccc|l} 1 & 1 & 1 & 0 & \mathbf{1} \leftarrow \text{Row 1 (3 ones} \rightarrow +1) \\ 0 & 1 & 1 & 0 & \mathbf{0} \leftarrow \text{Row 2 (2 ones} \rightarrow +0) \\ 1 & 0 & 0 & 1 & \mathbf{0} \leftarrow \text{Row 3 (2 ones} \rightarrow +0) \\ 1 & 1 & 0 & 1 & \mathbf{1} \leftarrow \text{Row 4 (3 ones} \rightarrow +1) \\ \hline \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \quad {\color{red}\mathbf{0}} \leftarrow \text{Corner Bit} \end{array}$$

3. **Result**: The checksum field contains 9 bits: Row Parities (4) + Col Parities (4) + Corner (1). **Answer**: ‘1001 1100 0‘.

Coach’s Notes (避坑指南)

陷阱警示 (Pitfalls):

- **Corner Bit**: Don’t forget the bottom-right bit! It checks the parity of the ”Row Parity Column” and ”Column Parity Row”. Both must yield the same result.
- **4-Bit Rectangular Error**: If 4 bits form a rectangle of errors, 2D parity **cannot detect** it (Row/Col parity remains even).

Slotted ALOHA (时隙 ALOHA)

Nodes transmit only at the beginning of synchronized slots.

- **Success Condition**: Only ONE node transmits, all others are silent.
- **Max Efficiency**: $1/e \approx 0.37$ (when $N \rightarrow \infty$).

必考 (Exam Focus)

Throughput Calculation (Tutorial 11 P2): General Formula: $P(\text{Node } i \text{ success}) = P(i \text{ sends}) \times \prod_{j \neq i} P(j \text{ silent})$.

Case 1: Asymmetric Nodes (A and B) Given $p_A > p_B$.

- Throughput of A: ${\color{red}p_A(1-p_B)}$. (A sends AND B is silent).
- Total Efficiency: $p_A(1-p_B) + p_B(1-p_A)$.
- **Is throughput proportional?** If $p_A = 2p_B$, is $Th_A = 2Th_B$? **No.** A’s activity suppresses B’s success chance.

Case 2: One ”Big” Node (A) vs Many ”Small” Nodes Node A has prob $2p$, others ($N-1$ nodes) have prob p.

- Throughput of A: $2p \times (1-p)^{N-1}$ (A sends, $N-1$ others silent).
- Throughput of any other node: $p \times (1-2p) \times (1-p)^{N-2}$ (Node sends, A silent, $N-2$ others silent).

CSMA (Carrier Sense Multiple Access)

”Listen before transmit”.

- **Why collisions occur?** Due to **Propagation Delay**. Node B may not hear A’s transmission immediately and starts sending.

Self-learning Switches (自学习交换机)

Switches are plug-and-play devices that filter and forward frames based on MAC addresses.

- **Learning**: Records (Source MAC, Incoming Port) into the Switch Table.
- **Forwarding**:
  – If Dest MAC known: **Unicast** (Selective Forwarding).
  – If Dest MAC unknown: **Flood** (Send to all ports except incoming).

必考 (Exam Focus)

Switch Table Evolution (Tutorial 11 P4): Scenario: A sends to G (Flood); then G responds to A (Unicast). Topology: $A \leftrightarrow S1 \leftrightarrow S4 \leftrightarrow S3 \leftrightarrow G$ (S2 is connected to S4 but not on the path).

Step 1: A → G (Flooding)

- All switches ($S1, S2, S3, S4$) learn **A**’s location.
- S2 receives the flood from S4, learns A is at the interface leading to S4.

Step 2: G → A (Unicast Response)

- Path: $G \rightarrow S3 \rightarrow S4 \rightarrow S1 \rightarrow A$.
- $S3, S4, S1$ learn **G**’s location.
- **Crucial Point**: **S2 does NOT learn G**. The frame is unicast along the path $S3 \rightarrow S4 \rightarrow S1$. S2 is off-path and never sees the frame.

Final Table State:

- S1, S3, S4: Know both A and G.
- S2: Knows **only A**.

Part Supplement: 核心概念大白话讲解

- **二维校验 (2D Parity) 的”神技”**: 单比特校验只能告诉你“出错了”，但不知道哪里错。二维校验就像玩“数独”，横着数不对，竖着数也不对，交叉点就是那个“坏人”。这让它能**纠正** 1 bit 错误。但如果坏成一个矩形（4 个角），校验就全瞎了。
- **Slotted ALOHA 的计算秘诀**: 不要死记公式。想求谁成功，就让**他举手** (p)，同时强迫**其他所有可能会捣乱的人闭嘴** ($1-p$)。不管是 2 个人还是 N 个人，逻辑永远是：$P(\text{Me}) \times \text{AllOthers(Silent)}$。
- **S2 为什么没学到 G (Switching Trap)**: 交换机是“听声辨位”的。
  – A 找 G 时是大喊大叫（泛洪），全网都听到了 A，所以 S2 记住了 A。
  – G 回复 A 时，因为大家都知道 A 在哪，所以 G 是悄悄话传回去（单播）。这条线不经过 S2，S2 根本听不到 G 的声音，自然学不到 G。

Application Layer (Assignment 1)

POP3 Protocol 邮件协议 (Assign 1 P17)

Mail Access Protocol (retrieving mail from server). Stateless (server doesn’t remember history across sessions).

- **Download-and-delete**: Messages are removed after retrieval. User cannot access them from another device. 读完就删。坑点：如果你在手

机上读了邮件，回家用电脑就看不到了。

- **Download-and-keep**: Server keeps a copy. **Trap:** If client is stateless, it might re-download old messages unless client tracks UIDs. 读完服务器还留底。现在的客户端大多默认这个。
- **无状态 (Stateless)**: POP3 服务器不记事，它不知道你上次读到哪了.

Zoom 架构 (Assign 1 Q)

- **Architecture**: Hybrid. Client-Server for signaling/auth; P2P (preferred) or Server-Relay for media streaming.
- **Transport**: **UDP** is preferred for voice/video (speed > reliability). **TCP** used for signaling/chat.

Link Layer (Assignment 3)

Address Resolution Protocol (ARP) (Assign 3 P15)

Mapping IP Address (Network Layer) → MAC Address (Link Layer).

必考 (Exam Focus)

ARP Packet Flow:

- **Query**: "Who has IP X?" → Broadcast MAC (`FF-FF-FF-FF-FF-FF`).
- **Response**: "I have IP X, my MAC is Y" → Unicast MAC.
- **Switch Action**: Floods Broadcast frames; Learns Source MAC from incoming frames.
- **Router Action**: Stops Broadcasts. ARP is local to the subnet.

CSMA/CD (Assign 3 P18)

Carrier Sense Multiple Access / Collision Detection (Ethernet). 核心逻辑：边说边听。为什么有最小帧长限制?

- **Collision Condition**: Sender must transmit long enough to detect a collision returning from the farthest node.
- **Formula**: $T_{trans} \geq 2 \times T_{prop\_max}$
- If $T_{trans} < 2 \times T_{prop}$, sender might finish before knowing a collision occurred (Packet lost without detection).

CRC 校验（循环冗余检查）(Assign 3 P1)

Cyclic Redundancy Check.

必考 (Exam Focus)

Steps to calculate CRC (Remainder R):

1. **Generator** G: $r+1$ bits. (Degree r).
2. **Append Zeros**: Add r zeros to Data D.
3. **Binary Division (XOR)**: Divide $D \cdot 2^r$ by G using XOR subtraction.
4. **Remainder**: The result is the CRC.

Example: $G = 10011$ (5 bits, $r = 4$). Append 0000 to Data. Perform XOR division.

Part 5: 综合大题 - " 打开网页的全过程" (Assignment 3 P31)

Scenario: Connect to network, download URL. No cache.

1. **DHCP (UDP)**: New PC broadcasts "I need IP". DHCP Server replies with IP, Mask, DNS IP, Gateway IP.
2. **ARP (1)**: PC needs Gateway MAC to send outside. Broadcasts "Who has Gateway IP?". Gateway replies.
3. **DNS (UDP)**: PC sends DNS Query to DNS Server IP (via Gateway). Resolves URL → Web Server IP.
4. **TCP Handshake**: PC sends SYN to Web Server IP. (Routing via Gateway → Internet → Server).
5. **HTTP (TCP)**: PC sends HTTP GET. Server replies HTTP 200 OK.

Coach's Notes (避坑指南)

Common Exam Mistake: MAC Addresses

- **Source MAC**: Always the device *sending* the frame on the *current* link.
- **Dest MAC**: Always the device *receiving* the frame on the *current* link (e.g., the Gateway Router, NOT the final Web Server).
- **Src/Dest IP**: End-to-end (PC to Web Server), usually do not change.

Part Supplement: 核心概念大白话讲解

- **ARP Scope**: ARP 喊话（Broadcast）出不去路由器。你在家里喊 "谁是网关"，只有家里的设备听得见。
- **CSMA/CD**: "边说边听"。如果你说话太快（包太短），还没听到远处的回音（冲突信号）就挂了电话，那你就不知道刚才的话被盖住了。所以包必须够长。
- **Switch vs Router**: Switch 是 "二层傻瓜"，只看 MAC，不知道 IP，看到不知道的 MAC 就广播；Router 是 "三层精英"，看 IP，隔离广播，修改 MAC 头转发包。