Review_题型整理

Class	CAN 201
Type	Seminar
Reviewed	☐

- ☐ 记忆一下从点击链接到显示页面的全流程
- ☐ 五个层的error detection方法分别是什么
- ☐ 通过DNS得到IP的流程是什么
- ☐ 看一下CRC计算过程
- ☐ 再看一下不同风险都可以通过什么方法来解决，或者是什么方法可以解决或者带来什么风险
- ☐ SDN流表

四、SDN流表的工作流程

1. **数据包到达交换机：**
   - 当数据包到达交换机时，交换机会根据现有的流表逐条匹配数据包的头部信息。
2. **流表匹配：**
   - 交换机检查流表中的流条目，寻找第一个与数据包匹配的流条目。
3. **执行流条目动作：**
   - 如果找到匹配的流条目，交换机执行该条目的指令和动作，如转发、修改、丢弃等。
4. **缺少匹配的流条目：**
   - 如果流表中没有匹配的流条目，交换机会根据配置，将数据包发送到SDN控制器进行处理。
5. **控制器下发流条目：**
   - SDN控制器根据网络策略和当前网络状态，生成适当的流条目，并通过开放接口（如OpenFlow）下发到交换机。
6. **交换机更新流表：**
   - 交换机接收到控制器下发的流条目后，更新流表，并对当前和未来匹配的数据包执行相应的处理。

Q1

一些比较重要的知识点

1. Delay（时延）

定义：

时延是指数据从发送端传输到接收端所需的时间。

组成：

网络时延可以分为以下几个部分：

1. **传输时延（Transmission Delay）：**
   - 数据包的大小除以链路的传输速率。
   - 公式：$\text{Transmission Delay} = \frac{\text{Packet Size}}{\text{Transmission Rate}}$
   - **例子**：发送一个 1MB 的文件到传输速率为 10 Mbps 的链路，需要 $1 \times 8 / 10 = 0.8$ 秒。
2. **传播时延（Propagation Delay）：**
   - 信号在物理媒介（如光纤、电缆）中传播所需的时间。
   - 公式：$\text{Propagation Delay} = \frac{\text{Distance}}{\text{Propagation Speed}}$
   - **例子**：光纤中的传播速度约为 $2 \times 10^8 \, \text{m/s}$，如果传输距离为 2000 km，传播时延为 $\frac{2000 \times 1000}{2 \times 10^8} = 0.01$ 秒。
3. **排队时延（Queuing Delay）：**
   - 数据包在路由器或交换机队列中等待处理的时间。
   - **影响因素**：
     - 网络拥塞程度。
     - 队列的长度。
4. **处理时延（Processing Delay）：**
   - 路由器或交换机分析数据包头并做出转发决策的时间。

---

2. Loss（丢包）

定义：

丢包是指在数据传输过程中，部分数据包未能成功到达目的地。

原因：

1. **拥塞**：
   - 路由器的缓冲区满了，无法接收更多数据包。
2. **网络故障**：
   - 链路中断或设备故障。
3. **信号干扰**：
   - 无线通信中，信号受干扰导致数据丢失。

影响：

- 丢包会导致应用程序性能下降，例如网页加载变慢或视频流卡顿。
- 对于基于 TCP 的传输协议，丢包会触发重传机制，进一步增加时延。

---

3. Throughput（吞吐量）

定义：

吞吐量是指单位时间内通过网络的实际数据量，通常用 Mbps 或 Gbps 表示。

影响因素：

1. **网络带宽**：
   - 带宽越高，理论上吞吐量越高。
2. **网络拥塞**：
   - 拥塞会降低有效吞吐量。
3. **协议开销**：
   - TCP 等协议的报头会占用一定的传输资源，降低有效吞吐量。

例子：

如果网络连接的带宽是 100 Mbps，但由于拥塞，实际吞吐量可能只有 50 Mbps。

---

4. Bandwidth（带宽）

定义：

带宽是指网络的最大传输速率，表示网络能够在单位时间内传输的最大数据量。

类型：

1. **理论带宽（Theoretical Bandwidth）：**
   - 链路的最大传输能力。
   - **例子**：光纤的理论带宽可能高达数 Tbps。
2. **有效带宽（Effective Bandwidth）：**
   - 实际传输过程中能够达到的带宽，通常低于理论值。

与吞吐量的区别：

- **带宽**是理论上的最大能力，**吞吐量**是实际的传输数据速率。
- 举例：一条高速公路的带宽是 5 条车道，但实际吞吐量取决于车流量和交通规则。

指标	定义	影响因素	示例
Delay (时延)	数据包从源到目的地的传输时间。	距离、链路速率、处理时延、队列时延等。	一个 HTTP 请求可能需要 200 ms 的总时延。
Loss (丢包)	数据包未成功到达目的地。	拥塞、信号干扰、硬件故障等。	视频播放卡顿，需重传数据包。
Throughput (吞吐量)	单位时间内成功传输的数据量。	带宽、拥塞、协议开销等。	100 Mbps 的网络中，实际吞吐量可能是 80 Mbps。
Bandwidth (带宽)	网络的最大传输能力。	物理链路特性。	光纤网络的带宽可能高达 1 Gbps。

五层

1. **应用层** (Application Layer)
   - 提供网络服务接口给用户或应用程序。处理高层协议，例如HTTP、FTP、SMTP等。
     (Provide network service interfaces to users or applications, handling high-level protocols like HTTP, FTP, SMTP.)
2. **传输层** (Transport Layer)
   - 提供端到端的通信服务，包括可靠性、流量控制和分段重组。常见协议有TCP和UDP。
     (Provide end-to-end communication services, including reliability, flow control, and segmentation/reassembly. Common protocols are TCP and UDP.)
3. **网络层** (Network Layer)
   - 负责数据包的路由选择和转发，确定从源到目的地的最佳路径。使用IP协议。
     (Responsible for routing and forwarding data packets, determining the best path from source to destination using IP protocol.)
4. **链路层** (Link Layer)
   - 控制数据帧在局域网中的传输，包括错误检测和点对点通信。常见协议有以太网（Ethernet）。
     (Control data frame transmission within local area networks, including error detection and point-to-point communication. Common protocols include Ethernet.)
5. **物理层** (Physical Layer)
   - 传递比特流，处理物理信号的传输，例如通过光纤、电缆或无线电波传递信号。
     (Transmit bitstreams and handle the physical signal transmission, such as through fiber optics, cables, or radio waves.)

TCP与UDP的主要区别

特性	TCP	UDP
连接方式	面向连接，需建立连接（三次握手）	无连接，直接发送数据报
可靠性	可靠传输，保证数据包按序到达，提供重传机制	不可靠传输，不保证数据包的到达和顺序
传输方式	面向字节流，数据以连续流的形式传输	面向数据报，数据以独立的报文形式传输
流量控制	有流量控制机制，防止发送方过快发送导致接收方溢出	无流量控制机制
拥塞控制	有拥塞控制机制，动态调整传输速率	无拥塞控制机制
头部开销	较大（通常20字节以上，包含序列号、确认号、窗口大小等信息）	较小（通常8字节，包含源端口、目标端口、长度、校验和）
传输速度	相对较慢，由于需要建立连接和维护可靠性	较快，适合对速度要求高的应用
适用场景	需要可靠性和数据完整性的应用，如网页浏览、文件传输	实时性要求高且能容忍部分数据丢失的应用，如在线游戏、视频直播
错误处理	通过重传、校验和等机制处理错误	仅进行基本的错误检测，错误恢复由应用层处理

C/S and P2P

CS分发时间公式：

$$T_{\text{Client-Server}} = \max\left(\frac{F \times N}{u_{\text{server}}}, \frac{F}{d_{min}}\right)$$

P2P分发时间公式:

$$D_{\text{P2P}} = \max\left\{\frac{F}{U_s}, \frac{F}{d_{\min}}, \frac{NF}{U_s + \sum_{i=1}^{N} u_i}\right\}$$

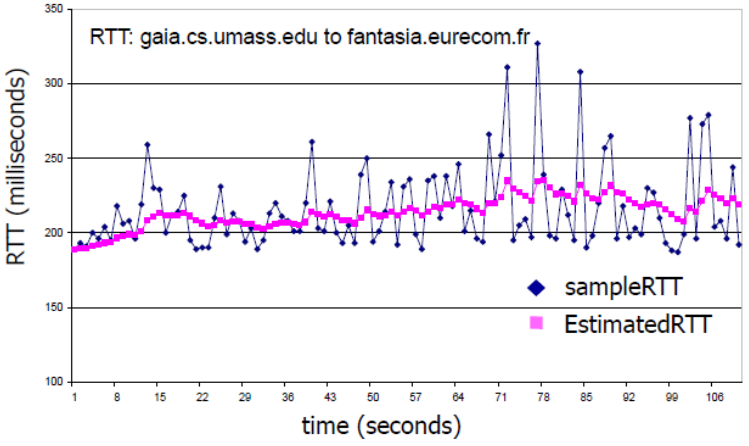
RTT估算

TCP Round Trip Time and Timeout

- **如何设置TCP超时值**：超时值应该比RTT（往返时间）更长，以避免过早重传。然而，RTT可能会有较大的波动：
  - 如果超时值设置得过短，可能会导致不必要的重传。
  - 如果超时值设置得过长，会导致丢失段时的响应过慢。
- **如何估算RTT**：TCP通过测量从发送数据段到收到确认的时间来估算RTT（SampleRTT），但RTT会有波动，因此需要使用平滑算法对RTT进行估算。
- **公式**：`EstimatedRTTn = (1-α) * EstimatedRTTn-1 + α * SampleRTT`

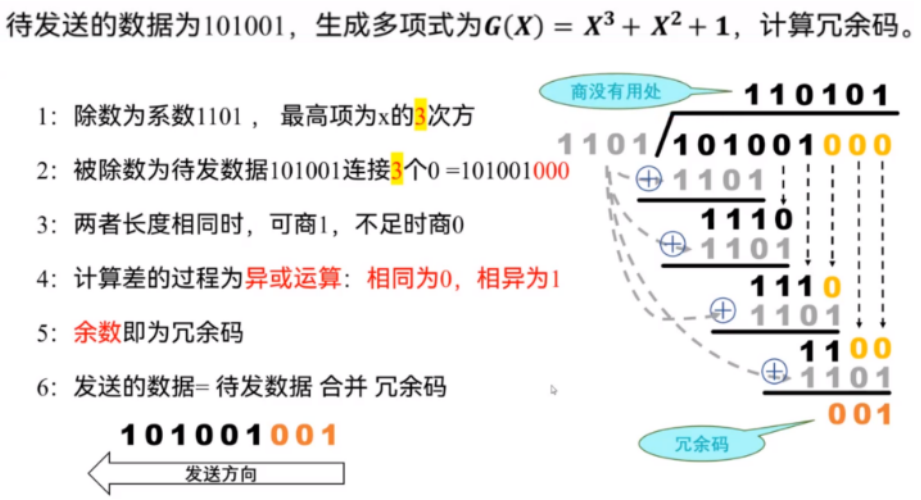
$$\mathbf{EstimatedRTT_n = (1-\alpha)*EstimatedRTT_{n-1} + \alpha*SampleRTT}$$

- 这里使用了指数加权移动平均算法（EWMA），其中α的典型值为0.125。该算法使得最近的RTT样本对估算结果影响较大，而较老的样本影响逐渐减小。
- **超时间隔（Timeout Interval）**：最终的超时值是估算的RTT值加上一个安全边际（通常与RTT偏差相关）。



- **超时间隔（Timeout Interval）**：超时间隔是由估算的RTT（EstimatedRTT）加上一个“安全边际”计算得出的。当RTT的变化较大时，安全边际会增大，以防止超时过早发生。
- **估算RTT偏差（DevRTT）**：DevRTT反映了实际RTT与估算RTT之间的差异，它的计算公式为：
  - `DevRTT = (1-β) * DevRTT + β * |SampleRTT - EstimatedRTT|`
  - 通常，β的值为0.25。
- **最终的超时间隔计算公式**：`TimeoutInterval = EstimatedRTT + 4 * DevRTT`。这个公式将EstimatedRTT与其偏差结合在一起，确保超时间隔能够适应网络条件的波动。

CRC



Q2

总结对比

- 这里主要来说就是对于HTTP1.1的细节的理解 以及记忆, 还有对于请求和回应的信息的理解
- 还有一种题型就是checksum的题型

题型1—对于请求信息的理解

- 这里提供一个全面的案例来做详细的每行讲解
- 一般问题的答案都可以这里找到
- 这里\r\n代表分隔符, 每个分隔符后面的就是其他内容了

```
GET /index.html HTTP/1.1\r\n
Host: www.example.com\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.9\r\n
Accept-Encoding: gzip, deflate, br\r\n
Connection: keep-alive\r\n
```

▼ 内容详解

1. `GET /index.html HTTP/1.1\r\n`
   - **内容解析：**
     - `GET`：HTTP 方法，表示从服务器获取资源。常用的还有：
       - **POST**：提交数据给服务器。
       - **PUT**：更新资源。
       - **DELETE**：删除资源。
     - `/index.html`：请求的资源路径，相对于服务器根目录。
     - `HTTP/1.1`：使用的 HTTP 协议版本。HTTP 版本的区别：
       - **HTTP/1.0**：每个请求创建一个新的连接。
       - **HTTP/1.1**：支持持久连接（`keep-alive`）。
       - **HTTP/2**：多路复用，性能更高。

- **延伸：**
  - 确保路径正确，如果有查询参数，例如 `/index.html?user=admin`，则需要分析参数含义。
  - HTTP 方法需要根据业务场景选择合适的方法。

---

2. `Host: www.example.com\r\n`

- **内容解析：**
  - 指定请求的目标主机名或域名。在 HTTP/1.1 中是必须字段。
  - 用于在服务器支持多个站点（虚拟主机）时区分请求。
- **延伸：**
  - 如果省略 `Host` 字段，HTTP/1.1 会返回 `400 Bad Request`。
  - 如果是 HTTPS 请求，则使用 `SNI (Server Name Indication)` 指定主机。

---

3. `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36\r\n`

- **内容解析：**
  - 指定客户端的标识，包括浏览器、操作系统等信息。
  - 示例：
    - `Mozilla/5.0`：历史遗留，表示兼容 Netscape 浏览器。
    - `Windows NT 10.0; Win64; x64`：操作系统信息（Windows 10, 64 位）。
    - `AppleWebKit/537.36`：浏览器内核版本。
    - `Chrome/91.0.4472.124`：浏览器版本。
- **延伸：**
  - 可以用来识别爬虫或恶意请求（如非标准 User-Agent）。
  - User-Agent 伪造：某些爬虫或工具可能会伪装成合法浏览器。

---

4. `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*`

- **内容解析：**
  - 指定客户端可接受的响应内容类型（MIME 类型）。
  - `q=0.9` 表示优先级（权重），值越高优先级越高。
- **延伸：**
  - 确认服务端返回的 MIME 类型是否与客户端匹配。
  - 常见 MIME 类型：
    - `text/html`：HTML 文档。
    - `application/json`：JSON 数据。
    - `image/png`：PNG 图片。

---

5. `Accept-Language: en-US,en;q=0.9\r\n`

- **内容解析：**
  - 指定客户端偏好的语言。
  - `en-US` 优先（美式英语），`en` 其次（通用英语）。
- **延伸：**
  - 服务端可以根据 `Accept-Language` 返回多语言内容。
  - 可以用于语言版本的网站（如 `example.com/fr`）。

6. `Accept-Encoding: gzip, deflate, br\r\n`

- **内容解析：**
  - 指定客户端支持的内容压缩格式：
    - `gzip`：常见压缩格式，兼容性高。
    - `deflate`：较轻量的压缩。
    - `br`：Brotli，性能更优（但不广泛支持）。
- **延伸：**
  - 确认服务端是否支持这些压缩格式。
  - 压缩可以显著减少传输的数据量，提高性能。

---

7. `Connection: keep-alive\r\n`

- **内容解析：**
  - 指定是否保持连接。
  - `keep-alive`：长连接，减少重复建立 TCP 连接的开销。
- **延伸：**
  - HTTP/2 默认启用长连接，不需要显式指定。

- 这里的常见题型包括:
  - 提取被请求的URL
  - HTTP的版本
  - 如何获取IP地址
  - 是否是可持续连接

▼ GPT猜测的可以根据提供信息来提问的问题

1. 请求的资源路径是什么？

- **Answer:** `/sat/index.html` （请求行中的路径部分）

---

2. 使用了哪种 HTTP 方法发送请求？

- **Answer:** `GET` （请求行的第一个单词）

---

3. HTTP 使用的是哪个版本？

- **Answer:** `HTTP/1.1` （请求行的最后一部分）

---

4. 客户端正在请求的主机名是什么？

- **Answer:** `sat.xjtlu.edu.cn` （`Host` 头字段的值）

---

5. 客户端使用的浏览器和操作系统是什么？

- **Answer:**
  - 浏览器：`Netscape/7.2`
  - 操作系统：`Windows NT 7.1` （从 `User-Agent` 字段中解析）

---

6. 客户端偏好的响应内容类型有哪些？

- **Answer:**
  - `text/html`

- `application/xhtml+xml`
- `application/xml;q=0.9`
- `text/plain;q=0.8`
- `image/png,*/*;q=0.5`（`Accept` 头字段中列出的内容）

---

7. 客户端支持哪些内容编码？

- **Answer:**
  - `gzip`
  - `deflate`（从 `Accept-Encoding` 头字段中提取）

---

8. 客户端支持的语言是什么？

- **Answer:**
  - `en-US`（优先）
  - `en`（次优，权重为 0.5）（从 `Accept-Language` 头字段中解析）

---

9. 客户端是否请求持久连接？

- **Answer:** 是（`Connection: keep-alive` 表明客户端请求持久连接）

---

10. 客户端希望连接保持多长时间？

- **Answer:** `300` 秒 （`Keep-Alive` 头字段的值）

---

11. 如果服务器资源未被修改，客户端希望返回什么状态码？

- **Answer:** `304 Not Modified`（`If-Modified-Since` 头字段通常触发此行为，字段未列出，但常用于缓存验证）

---

12. `User-Agent` 字段的作用是什么？

- **Answer:** 标识客户端的浏览器、操作系统和版本，供服务器优化内容。

---

13. 为什么需要 `Host` 字段？

- **Answer:** 在同一 IP 地址上托管多个虚拟主机时，用于区分请求目标。

---

14. 客户端可以接受的字符集是什么？

- **Answer:**
  - `ISO-8859-1`
  - `utf-8`（优先级略低）
  - `*`（任意字符集，权重最低）（从 `Accept-Charset` 头字段中解析）

---

15. `Connection: keep-alive` 的优势是什么？

- **Answer:** 减少了频繁建立和关闭连接的开销，提高了传输效率。

---

16. 如果省略了 `Host` 字段，服务器会如何响应？

- **Answer:** HTTP/1.1 请求中省略 `Host` 会导致服务器返回 `400 Bad Request` 错误。

---

17. `Accept-Encoding` 和压缩的作用是什么？

- **Answer:** 表明客户端支持的压缩格式（如 `gzip` 和 `deflate`），可以减小响应数据的大小，提高传输速度。

---

18. `Keep-Alive: 300` 是否强制服务器保持连接？

- **Answer:** 不是，这只是客户端的请求，服务器可能会提前关闭连接。

---

19. 客户端请求的资源路径与 Referer 字段有什么关系？

- **Answer:**
  - Referer（未显示）表明请求的来源页面，有助于分析请求的上下文。

---

20. 如果服务器不支持客户端请求的 MIME 类型，会发生什么？

- **Answer:** 服务器可能返回 `406 Not Acceptable` 或默认类型的内容。

---

21. `Accept` 字段中的优先级 (q) 参数有什么作用？

- **Answer:** 定义客户端对不同 MIME 类型的偏好，值越高优先级越高。

---

22. 浏览器如何解析 `sat.xjtlu.edu.cn` 的 IP 地址？

- **Answer:** 通过 DNS 查询，将域名解析为服务器的 IP 地址。

---

23. 为什么会发送 `Connection: close` 而不是 `keep-alive`？

- **Answer:** 在短连接中使用 `Connection: close` 可以释放服务器资源，减少不必要的连接占用。

---

24. `User-Agent` 字段中的伪造信息对服务器有何影响？

- **Answer:** 伪造的 `User-Agent` 可能误导服务器优化内容，甚至规避安全机制。

---

25. 如何使用 `Accept-Language` 提供本地化内容？

- **Answer:** 服务器根据 `Accept-Language` 返回相应语言的资源（如英文或中文页面）。

(23-24Resit)

- 这里的`<cr><lf>` 和上面的`\r\n` 一样

```
GET /sat/index.html HTTP/1.1<cr><lf>Host:
sat.xjtlu.edu.cn<cr><lf>User-Agent: Mozilla/5.0 (Windows;U; Windows NT
7.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:
text/xml, application/xml, application/xhtml+xml, text /html;q=0.9,
text/plain;q=0.8,image/png,*/*;q=0.5 <cr><lf>Accept-Language: en-
us,en;q=0.5<cr><lf>Accept- Encoding: zip,deflate<cr><lf>Accept-
Charset: ISO -8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive:
300<cr><lf>Connection:keep-alive<cr><lf><cr><lf>
```

a) What is the URL of the document requested by the browser? (3 points)
b) What version of HTTP is the browser running? (3 points)
c) How does the browser get IP address? (4 points)

Answer

```
a): http://sat.xjtlu.edu.cn/sat/index.html
b): HTTP/1.1
c):
The browser resolves the IP address of the Host (sat.xjtlu.edu.cn) using the DNS (Domain Name
The browser sends a DNS query to its configured DNS server.
The DNS server responds with the IP address of the host.
If the browser has previously resolved the IP address, it might retrieve it from its cache ir
```

（22-23Final）

GET /can201/index.html HTTP/1.1<cr><lf>Host: sat.xjtlu.edu.cn<cr><lf>User-Agent: Mozilla/5.0 (Windows;U; Windows NT 7.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) <cr><lf>Accept: text/xml, application/xml, application/xhtml+xml, text /html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5 <cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept- Encoding: zip,deflate<cr><lf>Accept-Charset: ISO -8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr><lf>Connection:keep-alive<cr><lf><cr><lf>

1. What is the URL of the document requested by the browser? (3 points)
2. Which version of HTTP is the browser running? (2 points)
3. Does the browser request a non-persistent or a persistent connection? (3 points)
4. What is the IP address of the client host where the browser is running? How does the browser get the web server's IP address? (7 points)
5. What type of browser initiates this message? Why is the browser type needed in an HTTP request message? (5 points)

Answer

```
a): http://sat.xjtlu.edu.cn/can201/index.html
b): HTTP/1.1
c): persistent connextion //这个一般可以从Connextion这里判断,如果是keep-alive的话那是可持续的
d): The client's IP address not include in the HTTP request.
The browser uses the DNS (Domain Name System) to resolve sat.xjtlu.edu.cn into an IP address
The browser queries the local DNS resolver or DNS server.
If the IP address is not cached, the DNS server recursively queries authoritative DNS servers
The resolved IP address is returned to the browser.
e): Mozilla/5.0 (Windows; Windows NT 7.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2
Serve content optimized for specific browsers or platforms.
Differentiate between human users and automated requests (e.g., bots or crawlers).
Track analytics or usage patterns based on client type.
```

题型2—对于回应信息的理解

- 这里也用一个全面的例子来详细讲解每一行的内容

▼ GPT推测问题

1. 响应的 HTTP 状态码是什么？表示什么？

- **问题：** HTTP 响应状态码是 200，表示什么含义？
- **答案：**
  - 状态码是 `200 OK`，表示请求成功，服务器正常返回了所请求的资源。

---

2. 响应是由什么服务器生成的？

- **问题：** 响应头中，`Server` 字段显示了哪个服务器及其版本？
- **答案：**
  - 服务器是 `Apache/4.0.52`。

---

3. 响应头中的日期是什么时间？

- **问题：** `Date` 字段指示的时间是什么？它有什么作用？
- **答案：**
  - 时间是 `Tue, 07 Mar 2023 12:39:45 GMT`。
  - 它表示服务器生成响应的时间，客户端可以用来校准时间或验证响应的时效性。

---

4. 文档的最后修改时间是什么？

- **问题：** 响应头中的 `Last-Modified` 字段表示什么时间？
- **答案：**
  - 时间是 `Sat, 10 Dec 2022 18:27:46 GMT`，表示资源最后一次被修改的时间。

---

5. 响应的内容长度是多少？

- **问题：** `Content-Length` 字段的值是多少？它表示什么？
- **答案：**
  - 值是 `3874`，表示响应正文的字节数。

---

6. 响应的内容类型是什么？

- **问题：** `Content-Type` 字段的值是什么？它有什么作用？
- **答案：**
  - 值是 `text/html; charset=utf-8`，表示内容是 HTML 文档，并使用 UTF-8 编码。

---

7. 服务器是否支持持久连接？

- **问题：** 响应中是否支持 `keep-alive` 连接？如何判断？
- **答案：**
  - 是的，支持持久连接。
  - `Connection: keep-alive` 和 `Keep-Alive: timeout=max=100` 头表明服务器同意保持连接并指定超时时间。

---

8. 响应的正文使用了什么 HTML 文档类型？

- **问题：** 从响应正文中可以看出 HTML 文档类型是什么？
- **答案：**
  - 文档类型是 `HTML 4.0 Transitional`，由 `<!doctype html public "-//w3c//dtd html 4.0 transitional//en">` 指定。

---

9. 文档的标题是什么？

- **问题：** 从 HTML 的 `<title>` 标签中可以得知文档的标题是什么？
- **答案：**
  - 标题是：`Unleash Yourselves and Dare to be Rationally Unconventional`。

---

10. 文档的生成工具是什么？

- **问题：** 从响应体的 `<meta>` 标签中，可以知道该文档是用什么工具生成的吗？
- **答案：**
  - 生成工具是：`Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape`。

---

11. 响应的正文结构包含哪些主要部分？

- **问题：** 从 HTML 的结构中，可以看出哪些主要部分？
- **答案：**
  - 包括：
    - `<html>`：HTML 文档开始。
    - `<head>`：包含元信息（如标题、编码）。
    - `<body>`：实际内容，包括标题 `<H1>` 和段落 `<p>`。

---

12. 响应正文的第一行内容是什么？

- **问题：** HTML 内容的第一行是什么？
- **答案：**
  - 第一行是 `<!doctype html public "-//w3c//dtd html 4.0 transitional//en">`。

---

13. 响应正文的主要信息是什么？

- **问题：** HTML 文档中 `<body>` 的主要内容是什么？
- **答案：**
  - 内容是：
    - 标题：`Unleash Yourselves and Dare to be Rationally Unconventional`。
    - 段落：`Last Modify Date: 1 Dec 2022`。

---

14. 如果文档未修改，客户端会收到什么状态码？

- **问题：** 如果客户端的缓存是最新的，服务器可能返回什么状态码？
- **答案：**
  - 状态码是 `304 Not Modified`。

---

15. 为什么 `Keep-Alive` 超时时间是 100？

- **问题：** `Keep-Alive: timeout=max=100` 中的 100 秒有什么作用？
- **答案：**
  - 它表示服务器允许连接保持活动状态的最长时间是 100 秒，以提高传输效率。

```
HTTP/1.1 200 OK\r\n
Date: Mon, 18 Dec 2023 10:15:30 GMT\r\n
Server: Apache/2.4.41 (Ubuntu)\r\n
Last-Modified: Wed, 13 Dec 2023 15:45:00 GMT\r\n
Content-Length: 348\r\n
Content-Type: text/html; charset=UTF-8\r\n
Connection: keep-alive\r\n
\r\n
<html>
<head>
    <title>Example Page</title>
</head>
<body>
    <h1>Welcome to the Example Page!</h1>
</body>
</html>
```

▼ 详细讲解

1. `HTTP/1.1 200 OK\r\n`

- **作用：** 响应行，提供协议版本、状态码以及状态描述。
  - `HTTP/1.1`：使用的协议版本。
  - `200`：状态码，表示请求成功。
  - `OK`：状态描述，进一步解释状态码。
- **延伸：** 常见的状态码包括：
  - **1xx**：信息响应（如 `100 Continue`）。
  - **2xx**：成功响应（如 `200 OK`, `201 Created`）。

- **3xx**：重定向（如 `301 Moved Permanently`，`302 Found`）。
- **4xx**：客户端错误（如 `400 Bad Request`，`404 Not Found`）。
- **5xx**：服务器错误（如 `500 Internal Server Error`）。

---

2. `Date: Mon, 18 Dec 2023 10:15:30 GMT\r\n`

- **作用：** 指示服务器生成响应的时间。
  - `Mon, 18 Dec 2023 10:15:30 GMT`：日期和时间，使用 GMT 格式。
- **延伸：**
  - 时间可以用于缓存策略，或客户端校准时间。
  - 如果没有 `Date`，客户端可能无法判断响应是否及时。

---

3. `Server: Apache/2.4.41 (Ubuntu)\r\n`

- **作用：** 指示服务器的软件及版本。
  - `Apache/2.4.41`：服务器类型和版本。
  - `(Ubuntu)`：运行环境。
- **延伸：**
  - 不建议暴露过多服务器信息，可能会被攻击者利用。

---

4. `Last-Modified: Wed, 13 Dec 2023 15:45:00 GMT\r\n`

- **作用：** 表示资源最后修改的时间。
  - `Wed, 13 Dec 2023 15:45:00 GMT`：最后修改时间。
- **延伸：**
  - 客户端使用 `If-Modified-Since` 头与该时间对比，判断是否需要重新下载资源。

---

5. `Content-Length: 348\r\n`

- **作用：** 指示响应主体的字节长度。
  - `348`：主体的字节数。
- **延伸：**
  - 如果长度不匹配，可能导致客户端无法正确解析响应。

---

6. `Content-Type: text/html; charset=UTF-8\r\n`

- **作用：** 指示响应内容的 MIME 类型及字符集。
  - `text/html`：表示响应是 HTML 文档。
  - `charset=UTF-8`：内容的字符编码是 UTF-8。
- **延伸：**
  - 常见 MIME 类型包括：
    - `text/plain`：纯文本。
    - `application/json`：JSON 数据。
    - `image/png`：PNG 图片。

---

7. `Connection: keep-alive\r\n`

- **作用：** 表示保持 TCP 连接。
  - `keep-alive`：客户端希望复用连接，减少建立连接的开销。

- **延伸：**
  - 如果是 `Connection: close`，服务器在响应后关闭连接。

---

8. 空行 `\r\n`

- **作用：** 分隔响应头和响应主体。
- **延伸：**
  - HTTP 协议规定，空行后开始传输响应内容。

---

9. 响应主体

```
<html>
<head>
    <title>Example Page</title>
</head>
<body>
    <h1>Welcome to the Example Page!</h1>
</body>
</html>
```

- **作用：** 响应的实际内容。
  - HTML 代码可以被浏览器解析并呈现为网页。
- **延伸：**
  - 响应主体的格式取决于 `Content-Type` 头。
  - 如果是二进制内容（如图片、视频），则客户端需要根据 MIME 类型正确处理。

> HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2023 12:39:45GMT+8<cr><lf>Server: Apache/4.0.52<cr><lf>Last-Modified: Sat, 10 Dec 2022 18:27:46 GMT+8<cr><lf>Accept-Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>Keep-Alive: timeout=max=100<cr><lf>Connection: Keep-Alive<cr><lf>Content-Type: text/html; charset=utf-8<cr><lf><cr><lf><!doctype html public ”-//w3c//dtd html 4.0 transitional//en”><lf><html><lf><head><lf> <meta http-equiv=”Content-Type”content=”text/html; charset=utf-8”><lf> <meta name=”GENERATOR” content=”Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]”><lf> <title>Unleash Yourselves and Dare to be Rationally Unconventional</title><lf></head><lf><body><H1>Unleash Yourselves and Dare to be Rationally Unconventional</H1> Last Modify Date: 1 Dec 2022 ...<much more document text following here (not shown)>

a) Was the server able to successfully find the document or not? What time was the document reply provided? (2 points)
b) When was the document last modified? (2 points)
c) How many bytes are there in the document being returned? (2 points)
d) What are the first 9 bytes of the document being returned? (2 points)
e) Did the server agree to a persistent connection? (2 points)

- 一般来说对于回应信息来说两个分隔符后面就是主体内容了

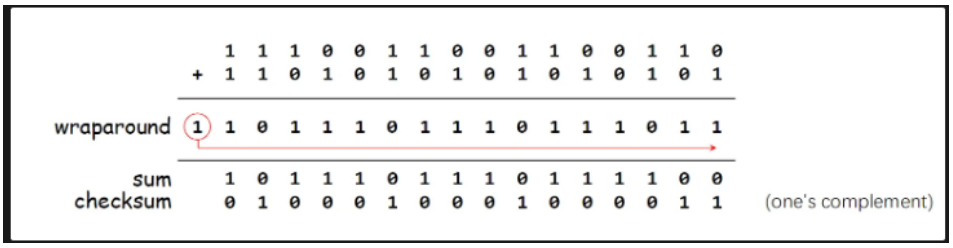
Answer

```
a): Yes, the server successfully found the document. This is indicated by the status
line HTTP/1.1 200 OK, where 200 OK means the request was successful.
The document reply was provided on Tue, 07 Mar 2023 12:39:45 GMT,
as shown in the Date header.
b): The document was last modified on Sat, 10 Dec 2022 18:27:46 GMT,
```

```
as indicated by the Last-Modified header.
c): The document being returned is 3874 bytes in size, as specified by
the Content-Length header.
d): The first 9 bytes of the document are <!doctype,
as these are the beginning characters of the HTML response body.
e): Yes, the server agreed to a persistent connection. This is indicated by
the Connection: keep-alive header and the Keep-Alive: timeout=max=100 header,
which specify that the connection should remain open.
```

题型3—checksum

- 这个重点就是了解checksum的原理以及用法
- 以下是一组checksum 的案例
  - 这里将两组二进制相加，如果溢出的那么就再将溢出的加回去
  - 将最终的结果取1'补码， 也就是各个位置的数取反
  - 一般来说发送端都会将原本的二进制码以及checksum这个码一起发送给接收端
  - 接收端来将所有的二进制码加起来包括这个checksum，如果结果是每个位置上都为1，那么代表这个数据没有问题，如果出现了0那么代表就是有问题的



- 一般会额外引申出一组问题，那就是可不可以判断1'bit的错误
  - 这里的答案是可以的，因为对于1'bit的错误来说这里有一个地方发生改变那就绝对会出现0
- 还有一种就是判断2'bit的错误可不可以检测出
  - 这个答案是不一定，也就是两个地方出现的问题同时出现也可能导致checksum没有问题，也就是两个错误刚好隐藏了问题，导致不一定能通过checksum来发现问题

（22-23Resit）

Due to the issues such as interferences and device errors, bit errors often occur in networking communication. In order to receive the correct information through the internet, error detection and correction methods are applied in some layers.

1. Suppose you have the following three 8-bit bytes: `01010011`, `11001100`, `00001100`. What is the 1s complement of the sum of these 8-bit bytes? (10 points)
2. With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error? (10 points)

Answer

```
a): 01010011 + 11001100 = 100111111 （溢出）
处理溢出：00111111 + 1 = 01000000
再加第三个字节：01000000 + 00001100 = 01001100
取得1's 补码
10110011
b):
接收端如何检测错误？
接收端将所有收到的数据（包括校验和）相加，结果如果是全 1（11111111），则无错误；否则，有错误。
```

是否可能 1-bit 错误未被检测到？
不可能。任何单比特错误都会改变校验和结果，无法为全 1。

是否可能 2-bit 错误未被检测到？
可能。如果两个比特的错误相互抵消（一个 0->1，另一个 1->0），校验和可能仍为全 1，错误未被检测到。

(19-20Final)

a) UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. (10 points)

b) Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error? (10 points)

Answer

a): 1's 补码为：11010001
b):
UDP 使用 1's 补码的原因：
高效处理溢出，适合硬件实现。

错误检测能力：
1-bit 错误一定可以检测到。
2-bit 错误可能未被检测到（如果错误相互抵消）。

Q4

Class A

- 示例地址：`10.0.0.0`
- 默认掩码：`255.0.0.0`
- 掩码长度：`/8`
- 地址范围：`10.0.0.0 - 10.255.255.255`

Class B

- 示例地址：`172.16.0.0`
- 默认掩码：`255.255.0.0`
- 掩码长度：`/16`
- 地址范围：`172.16.0.0 - 172.16.255.255`

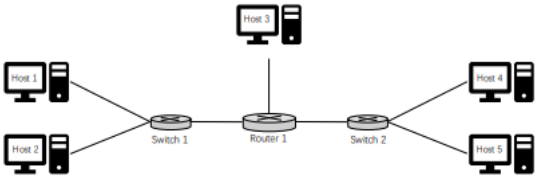
Class C

- 示例地址：`192.168.1.0`
- 默认掩码：`255.255.255.0`
- 掩码长度：`/24`
- 地址范围：`192.168.1.0 - 192.168.1.255`

题型1

- 一般来说会问有几个子网，子网的判断条件就是路由器之间或者路由器自己分出去几个支线

（23-24Resit）



这个只需要说明有几个子网并且说明每个子网里面有什么即可

- !!! 注意Switch 并不列入子网的描述中

Answer

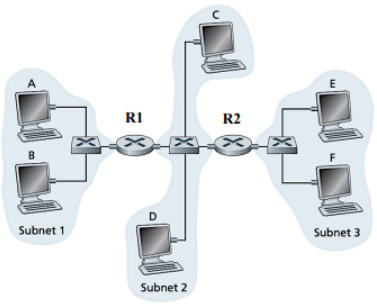
```
3 subnet
subnet1: Interface of Host1, Interface of Host1 and Router1's interface which connect to Swit
subnet2: Interface of Host3 and Router1' interface witch connect to Host3
subnet3: Interface of Host4, Interface of Host5 and Router1'  interface witch connect to Swit
```

特殊案例

- 如果这里将R1和R2之间的Switch替换成R3之后这里有几个子网

Answer

```
6 subnet
subnet 1: A, B, and R1's interface
subnet 2: R1's interface and R2's interface(betwee
subnet 3: R2's interface and R3's interface(betwee
subnet 4: C and R3's interface
subnet 5: D and R3's interface
subnet 6: E, F, and R2's interface
```



(21-22Final)

- **路由器之间的每个连接都算一个子网，不管里面有没有东西**

题型2

- 计算最大IP地址数量，以及对应掩码

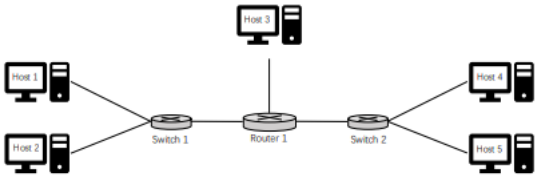


Fig. 3.

If the interface of Router 1 linked with Host 3 has an IP address 10.0.2.1/25, which would be the largest IP address that can be assigned to the interface of Host 3 and the corresponding network mask? (4 points)

- 这里的地址提供的是IPv4的地址，也就是32位，这里的25代表网络层占用了25位，所以留给地址表达的就只剩下 $2^{(32-17)} = 128$，排除`10.0.2.0`和`10.0.2.127`不可用之外最大的就是`10.0.2.126`
  - `10.0.2.0/25`是用来标示整个网络的, 表示这个子网
  - `10.0.2.127`是用来作为广播地址使用的, 也就是发送到这个低智商会广播给所有子网内的设备
- 对于掩码的转化:
  - 这里的25代表在32位中的前25位是1,后面剩下的7位都是0,也就是`11111111.11111111.11111111.10000000` = `255.255.255.128`
  - 其他的对于24的来说就是, 剩下的以此类推

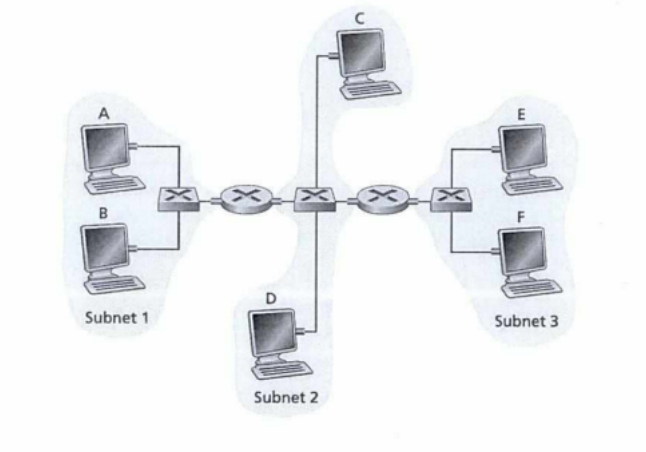
    `11111111.11111111.11111111.00000000` = `255.255.255.0`

Answer

```
10.0.2.126
255.255.255.128
```

题型3

- 为每个接口标记ip地址以及MAC码



a) Assign IP addresses to all of the interfaces. For Subnet 1 use addresses of the form 192.168.1.xxx; for Subnet 2 uses addresses of the form 192.168.2.xxx; and for Subnet 3 use addresses of the form 192.168.3.xxx. (5 points)

b) Assign MAC addresses to all of the adapters. (5 points)

- 一般来说这里就根据提供给的信息来推测
- 没有提供给你的信息来自己编造即可
- Adapt一般来说就是Network Interface,所以给每个接口标注一下就好了, Switch不算在内
- 这里最好把host标为字母, router标为数字, 我也不知道为什么

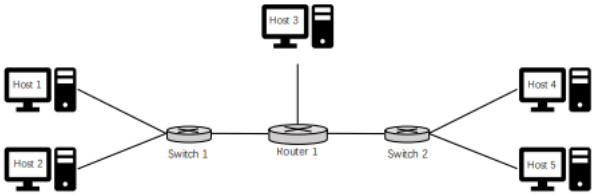
Answer

```
A: 192.168.1.2 B: 192.168.1.3 //这里2,3可以是任何数,不一定非是2,3
C: 192.168.2.2 D: 192.168.2.3
E: 192.168.3.2 E: 192.168.3.3

A: aa-aa-aa-aa-aa-aa B: bb-bb-bb-bb-bb-bb
Router1 in subnet1: 11-11-11-11-11-11
Router1 in subnet2: 22-22-22-22-22-22//以此类推
```

题型4

- 详细讲出从一个Host转发数据包到另一个Host的过程



3. Assuming the interface of Host 1 has an IP address 10.0.1.2, and the adapter for that interface has a MAC address aa-aa-aa-aa-aa-aa; the interface of Router 1 lined with Switch 1 has an IP address 10.0.1.1, and the adapter for that interface has a MAC address 11-11-11-11-11-11. Now, consider sending an IP datagram from Host 1 to Host 5. Suppose Host 1 has an empty ARP table, while Router 1 has the up-to-date ARP table and routing table respectively. Describe all the steps to succeed in sending the IP datagram. (8 points)

- APR代表IP和MAC映射的表, Host想要转发东西必须通过MAC地址来发送
- 这里分三种情况, 一般的题都会在情况2, 3出题, 情况1比较简单, 本题属于情况2

▼ 1. **发送端和接收端都在一个子网内**

假设有两个主机 A 和 B，在同一子网中，A 要向 B 发送数据：

1. **主机 A 检查目标 IP 地址：**
   - 主机 A 通过子网掩码判断目标 IP 地址是否在同一子网。
   - 如果目标 IP 地址在同一子网，数据将直接发送。
2. **主机 A 查询目标的 MAC 地址：**
   - 主机 A 检查自己的 ARP 表，是否已缓存目标 IP 地址对应的 MAC 地址。
   - 如果没有缓存，则发送 ARP 请求，目标是主机 B 的 IP 地址。
3. **主机 B 响应 ARP 请求：**
   - 主机 B 返回自己的 MAC 地址。
   - 主机 A 将 B 的 MAC 地址存入 ARP 表中。
4. **主机 A 封装数据帧：**
   - 使用数据链路层的 MAC 地址进行帧封装：
     - 源 MAC 地址：主机 A 的 MAC 地址。
     - 目标 MAC 地址：主机 B 的 MAC 地址。
   - 使用网络层的 IP 地址进行 IP 数据报封装：
     - 源 IP 地址：主机 A 的 IP 地址。
     - 目标 IP 地址：主机 B 的 IP 地址。
5. **主机 A 发送帧到主机 B：**
   - 以太网帧通过局域网传输到目标 MAC 地址对应的设备（主机 B）。
6. **主机 B 接收并解封装数据：**
   - 主机 B 根据 MAC 地址接收帧，解封装得到 IP 数据报。
   - 主机 B 检查 IP 数据报中的目标地址，确认是发给自己的，完成处理。

▼ 2. **发送端和接收端在相邻的两个子网内**

假设设备 A 在子网 1，设备 B 在子网 2，子网 1 和子网 2 是相邻的，直接通过一台路由器（Router R）的接口连接。

1. **设备 A 判断目标 IP：**
   - 设备 A 根据子网掩码判断目标设备 B 的 IP 是否在同一子网。
   - 发现目标 IP 不在同一子网，因此需要发送数据给默认网关（Router R）。
2. **设备 A 获取默认网关的 MAC 地址：**
   - A 发送 ARP 请求以解析默认网关（Router R）的 MAC 地址。
   - Router R 的接口（连接到子网 1）回复 MAC 地址，A 记录此信息。
3. **设备 A 发送数据到 Router R：**

- 数据帧封装：
  - **源 MAC 地址：** A 的 MAC 地址。
  - **目标 MAC 地址：** Router R 子网 1 接口的 MAC 地址。
  - **源 IP 地址：** A 的 IP 地址。
  - **目标 IP 地址：** B 的 IP 地址。

4. **Router R 转发数据到子网 2：**
   - Router R 接收数据帧，解封装后检查 IP 数据报。
   - Router R 查找路由表，发现目标 IP 属于子网 2，下一跳就是直接连接的子网 2。
   - Router R 查询 ARP 表，获取设备 B 的 MAC 地址（如果不存在，则发送 ARP 请求）。
   - 数据帧封装：
     - **源 MAC 地址：** Router R 子网 2 接口的 MAC 地址。
     - **目标 MAC 地址：** B 的 MAC 地址。
     - **源 IP 地址：** A 的 IP 地址。
     - **目标 IP 地址：** B 的 IP 地址。
5. **设备 B 接收数据：**
   - B 接收帧，解封装后检查 IP 数据报，发现目标是自己。
   - 数据处理完成。

▼ 3.**发送端和接收端不在相邻的两个子网内**

假设设备 A 在子网 1，设备 B 在子网 3，子网 1 和子网 3 之间通过两台路由器（Router R1 和 Router R2）连接。

1. **设备 A 判断目标 IP：**
   - 设备 A 根据子网掩码判断目标设备 B 的 IP 是否在同一子网。
   - 发现目标 IP 不在同一子网，因此需要发送数据给默认网关（Router R1）。
2. **设备 A 获取默认网关的 MAC 地址：**
   - A 发送 ARP 请求以解析默认网关（Router R1）的 MAC 地址。
   - Router R1 的接口（连接到子网 1）回复 MAC 地址，A 记录此信息。
3. **设备 A 发送数据到 Router R1：**
   - 数据帧封装：
     - **源 MAC 地址：** A 的 MAC 地址。
     - **目标 MAC 地址：** Router R1 子网 1 接口的 MAC 地址。
     - **源 IP 地址：** A 的 IP 地址。
     - **目标 IP 地址：** B 的 IP 地址。
4. **Router R1 转发数据到 Router R2：**
   - Router R1 接收数据帧，解封装后检查 IP 数据报。
   - Router R1 查找路由表，发现目标 IP 属于子网 3，下一跳是 Router R2。
   - Router R1 查询 ARP 表，获取 Router R2 的 MAC 地址（如果不存在，则发送 ARP 请求）。
   - 数据帧封装：
     - **源 MAC 地址：** Router R1 的 MAC 地址（连接到 R2 的接口）。
     - **目标 MAC 地址：** Router R2 的 MAC 地址。
     - **源 IP 地址：** A 的 IP 地址。
     - **目标 IP 地址：** B 的 IP 地址。
5. **Router R2 转发数据到子网 3：**

- Router R2 接收数据帧，解封装后检查 IP 数据报。
- Router R2 查找路由表，发现目标 IP 属于子网 3。
- Router R2 查询 ARP 表，获取设备 B 的 MAC 地址（如果不存在，则发送 ARP 请求）。
- 数据帧封装：
  - **源 MAC 地址：** Router R2 子网 3 接口的 MAC 地址。
  - **目标 MAC 地址：** B 的 MAC 地址。
  - **源 IP 地址：** A 的 IP 地址。
  - **目标 IP 地址：** B 的 IP 地址。

6. **设备 B 接收数据：**
   - B 接收帧，解封装后检查 IP 数据报，发现目标是自己。
   - 数据处理完成。

步骤	相邻子网	不相邻子网
判断目标 IP	设备直接发送到默认网关（Router）。	设备发送到默认网关，由中间路由器逐跳转发到目标子网。
ARP 表查询	只需查找或请求默认网关和目标设备的 MAC 地址。	每个路由器接口之间的 MAC 地址都需要查找或请求。
转发过程	路由器直接转发到目标设备所在的子网。	路由器逐跳转发，直到目标设备所在的子网。
封装和解封装次数	1 次封装（设备到路由器），1 次解封装（路由器到目标设备）。	每经过一个路由器封装和解封装一次，最终到达目标设备。

Answer

1. Host 1 checks if the target IP is in the same subnet

- Host 1 uses the subnet mask (assumed `/24`) to check if the target IP address `10.0.3.x` is in the same subnet.
- Host 1 determines the target IP is outside its subnet, so it needs to send the datagram to the default gateway (Router 1).

2. Host 1 resolves the MAC address of the default gateway (Router 1)

- Since Host 1's ARP table is empty, it sends an ARP request to resolve the MAC address of Router 1's IP `10.0.1.1`.
- Router 1 replies with its MAC address `11-11-11-11-11-11`.
- Host 1 stores Router 1's MAC address in its ARP table.

3. Host 1 encapsulates the frame and sends it to Router 1

- Host 1 encapsulates the datagram into an Ethernet frame:
  - **Source MAC Address:** `aa-aa-aa-aa-aa-aa` (Host 1's MAC).
  - **Destination MAC Address:** `11-11-11-11-11-11` (Router 1's MAC).
  - **Source IP Address:** `10.0.1.2`.
  - **Destination IP Address:** `10.0.3.x`.
- Host 1 sends the frame to Router 1.

4. Router 1 decapsulates and checks the target IP

- Router 1 receives the frame, decapsulates the Ethernet header, and inspects the IP datagram.
- Router 1 looks up its routing table and identifies the destination IP `10.0.3.x` belongs to Subnet 3 (`10.0.3.0/24`).
- Router 1 determines the next hop is its interface connected to Subnet 3.

5. Router 1 resolves the MAC address of Host 5

- Router 1 checks its ARP table for Host 5's MAC address.
- If the MAC address is not in the ARP table, Router 1 sends an ARP request for Host 5's IP `10.0.3.x`.
- Host 5 replies with its MAC address (assume `55-55-55-55-55-55`).

6. Router 1 encapsulates the frame and sends it to Host 5

- Router 1 encapsulates the datagram into a new Ethernet frame:
  - **Source MAC Address:** Router 1's MAC address for Subnet 3.
  - **Destination MAC Address:** `55-55-55-55-55-55` (Host 5's MAC).
  - **Source IP Address:** `10.0.1.2`.
  - **Destination IP Address:** `10.0.3.x`.
- Router 1 sends the frame to Host 5.

7. Host 5 receives and processes the datagram

- Host 5 receives the frame, decapsulates it, and inspects the IP datagram.
- Host 5 confirms the IP datagram is addressed to itself and processes the data.

Q5

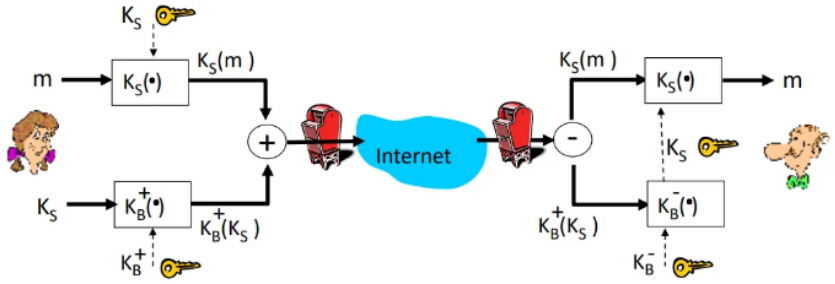
- 这里主要考察的重点就是三个类型的加密
  - **机密性**
  - **完整性**
  - **机密性+完整性**
- 一般来说也会有以下五种潜在威胁
  - **Eavesdropping（窃听）**
  - **Hijacking（劫持）**
  - **Denial of Service-DoS （拒绝服务）**
  - Impersonation（冒充）
  - Message Insertion（消息插入）

威胁类型	影响的属性	原因
Eavesdropping (窃听)	保密性 (Confidentiality)	未经授权获取通信内容，泄露敏感数据。
Hijacking (劫持)	保密性 (Confidentiality)	劫持过程中可能窃取敏感信息。
	完整性 (Integrity)	劫持后可能篡改数据，破坏真实性和可信性。
	可用性 (Availability)	劫持可能导致资源不可用或服务中断。
Denial of Service (DoS)	可用性 (Availability)	通过耗尽资源或流量攻击使服务不可访问。
Impersonation (冒充)	保密性 (Confidentiality)	冒充者可能访问敏感数据，破坏数据的保密性。
	完整性 (Integrity)	冒充者伪造身份发送篡改的消息，破坏数据的真实性。
Message Insertion (消息插入)	完整性 (Integrity)	插入伪造的消息，破坏数据的真实性和可信性。

机密性

- 这一部分就是利用**不对称加密**来进行发送了
- 需要注意的点
  - 机密性是要保证利用对方的公钥来加密
  - 不需要身份验证
  - 不确定用机密邮件算不算违规？

目标：Alice希望向Bob发送机密电子邮件 m。



- **过程**：
  1. Alice生成一个**随机的对称密钥** K_S
  2. Alice用对称密钥 K_S加密消息 m: $K_S(m)$
  3. Alice用Bob的公钥 $K_B{}^+$加密对称密钥 K_S: $K_S{}^+(K_S)$
  4. Alice将以下两部分发送给Bob：
     - 加密的消息 $K_S(m)$。
     - 加密的对称密钥 $K_B{}^+(K_S)$。
- **优势**：
  - 对称密钥加密高效。
  - 公钥加密用于安全传输对称密钥。

Bob接收到 $K_S(m)$ 和 $K_B{}^+(K_S)$，解密过程如下：

1. 用Bob的私钥 $K_B{}^-$解密 $K_B{}^+(K_S)$，恢复对称密钥 K_S：

$$K_B{}^-(K_B{}^+(K_S)) = K_S$$

2. 用对称密钥 K_S 解密消息 $K_S(m)$，恢复原始消息 m：

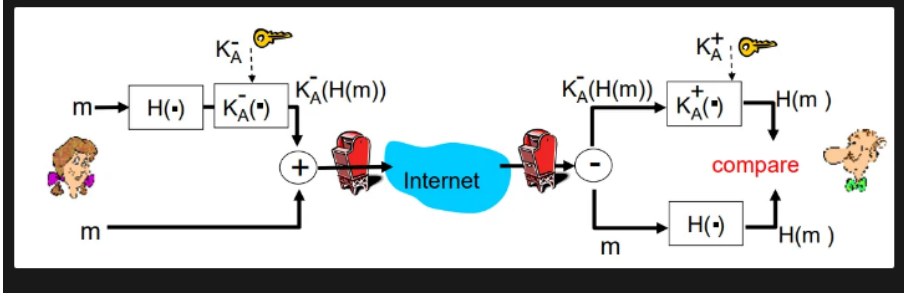
$$K_S{}^{-1}(K_S(m)) = m$$

完整性

- 一般来说这个完整性就要用到**数字签名（Digtal signatures)**以及**信息摘要（Meaage Digests）**
- 这里**数字签名**用来作为验证数据完整性以及身份验证的手段
  - 这里数据完整性书要是通过将原始信息以及加密信息同时发送到对方那里来让对方对比原始信息以及被加密信息是否相同，如果相同则代表信息完整，如果不相同则代表信息不完整
  - 这里的身份验证是指：在发送的时候发送方利用自身的私钥来加密数据，之后将加密信息以及原始信息一同发送到接收方，接收方只能用对方的公钥来解密信息从而进行验证

目标：Alice希望提供消息的发送者认证和完整性保证。

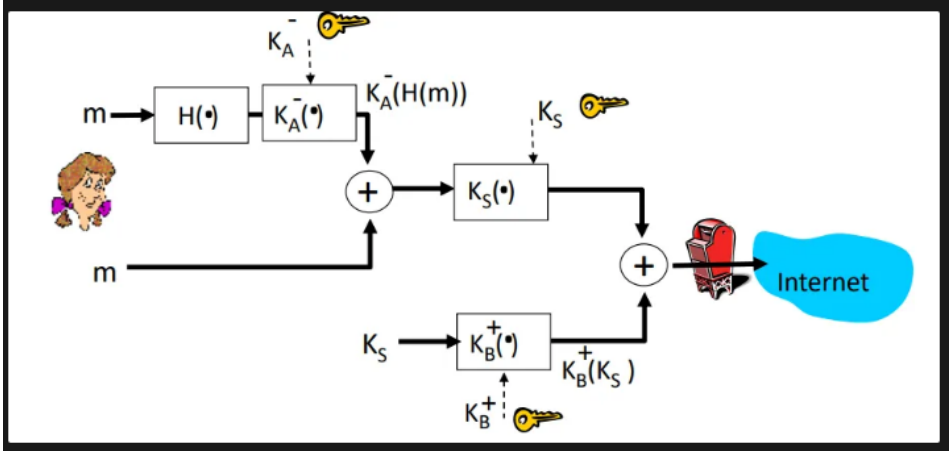
- **过程：**
  1. Alice对消息 m 计算哈希值 H(m)。
  2. 用Alice的私钥 K_A^- 对哈希值 H(m) 进行签名：$K_A^-(H(m))$
  3. Alice将消息 m 和数字签名 $K_A^-(H(m))$一起发送给Bob。
- **Bob的验证过程：**
  1. Bob接收到 m 和 $K_A^-(H(m))$。
  2. 用Alice的公钥 K_A^+ 解密签名，得到哈希值 H(m)：$K_A^+(K_A^-(H(m))) = H(m)$
  3. Bob对消息 m 重新计算哈希值 H'(m)。
  4. 比较 H(m) 和 H'(m)：
     - 如果相等，则认证通过，说明消息未被篡改且确实由Alice签署。



机密性+完整性

- 这一块一般就是用对方的公钥给完整性这一块加密一下就行了，逻辑就是机密性套完整性

综合安全（机密性 + 认证 + 完整性）



- Alice希望同时实现机密性、发送者认证和消息完整性。
- **方法**：
  1. Alice生成对称密钥 K_S。
  2. Alice对消息 m 计算哈希值 H(m)，并用私钥签名：$K_A^-(H(m))$
  3. Alice用 K_S 加密消息 m 和签名 $K_A^-(H(m))$：$K_S(m, K_A^-(H(m)))$
  4. Alice用Bob的公钥加密对称密钥 K_S：$K_B^+(K_S)$
  5. Alice将以下内容发送给Bob：
     - $K_S(m, K_A^-(H(m)))$
     - $K_B^+(K_S)$
- **Bob的解密过程**：
  1. 用私钥 K_B^- 解密 $K_B^+(K_S)$，获得对称密钥 K_S。
  2. 用 K_S 解密 $K_S(m, K_A^-(H(m)))$，获得 m 和签名 $K_A^-(H(m))$。
  3. 用Alice的公钥 K_A^+ 验证签名 $K_A^-(H(m))$，检查哈希值是否匹配。

对称加密和非对称加密

对称加密 (Symmetric Encryption)

对称加密是加密和解密使用相同的密钥。

优点：

1. **速度快**：
   - 算法计算简单，通常比非对称加密快得多，适合处理大规模数据。
2. **资源占用少**：
   - 计算开销小，适用于性能受限的设备，如嵌入式系统。
3. **实现简单**：
   - 算法较成熟且易于实现。

缺点：

1. **密钥分发问题**：
   - 加密双方必须提前安全地共享密钥，密钥传递过程中可能被截获。
2. **密钥管理复杂**：
   - 随着用户数量增加，所需的密钥数量呈指数增长。例如，N 个用户需要 N×(N−1)/2 个密钥。
3. **缺乏认证功能**：
   - 无法确认消息的发送者是否可信，仅能保证数据保密性。

非对称加密 (Asymmetric Encryption)

非对称加密使用一对密钥：公钥 (Public Key) 和私钥 (Private Key)。公钥用于加密，私钥用于解密。

优点：

1. **密钥分发更安全**：
   - 公钥可以公开传播，不需要安全通道分发密钥，私钥始终保留在本地，减少密钥泄露的风险。
2. **支持数字签名**：
   - 可用于验证消息发送者身份，实现消息的不可否认性和完整性。
3. **管理简单**：
   - 每个用户只需要一对密钥，无需为每对通信方创建密钥。

缺点：

1. **速度慢**：
   - 算法复杂，计算速度远低于对称加密，不适合加密大数据。
2. **资源占用高**：
   - 需要更多计算资源，特别是对性能较低的设备不友好。
3. **实现复杂**：
   - 算法实现和密钥管理更复杂，增加了开发和维护的难度。

特性	对称加密	非对称加密
加密/解密速度	快	慢
密钥管理	复杂，需要共享密钥	简单，仅需公钥和私钥
安全性	依赖密钥分发的安全性	更高，公钥可公开
适用场景	数据传输、大规模数据加密	数字签名、密钥交换