# 第七章 网路协议基础

## ARP协议

### ARP协议基础

**1. ARP的需求背景**

* 问题：IP地址用于网络层寻址，但数据链路层需MAC地址进行通信。当设备只知道目标IP而不知其MAC时，需通过ARP协议动态解析。
* 案例：PC1（IP\_A）想访问PC2（IP\_B），PC1需先获取PC2的MAC\_B。

**2. ARP工作原理**

核心流程：

* 发送方检查本地ARP缓存表是否有目标IP的MAC记录。
* 若不存在，**广播**发送ARP请求（目标MAC为FF-FF-FF-FF-FF-FF）。
* 目标主机（IP匹配者）**单播**回复ARP响应告知自身MAC。
* 发送方更新ARP表，后续通信直接使用缓存MAC。

### ARP请求与响应过程

**1. ARP请求（广播）**

报文关键字段：

表7.1 ARP请求字段

|  |  |  |
| --- | --- | --- |
| **字段** | **值** | **说明** |
| Sender MAC | PC1\_MAC | 请求方MAC地址 |
| Sender IP | IP\_A | 请求方IP地址 |
| Target MAC | 00:00:00:00:00:00 | 全0（未知） |
| Target IP | IP\_B | 目标IP地址 |
| Opcode | 1 (Request) | 操作码（请求） |

**2. ARP响应（单播）**

报文关键字段：

表7.2 ARP响应字段

|  |  |  |
| --- | --- | --- |
| **字段** | **值** | **说明** |
| Sender MAC | PC2\_MAC | 响应方MAC地址 |
| Sender IP | IP\_B | 响应方IP地址 |
| Target MAC | PC1\_MAC | 请求方MAC地址 |
| Target IP | IP\_A | 请求方IP地址 |
| Opcode | 2 (Reply) | 操作码（响应） |

**3. ARP缓存表**

查看方法：

arp -a # Windows/Linux查看ARP表

show arp # Cisco设备查看ARP表

老化时间：通常300秒（可配置）。

**4. 全F与全0的区别**

表7.3 F与0的答疑

|  |  |  |
| --- | --- | --- |
| **字段位置** | **值** | **说明** |
| 以太网帧头目标MAC | FF:FF:FF:FF:FF:FF（全 F） | 确保ARP请求以广播形式发送到整个局域网 |
| ARP报文Target MAC | 00:00:00:00:00:00（全 0） | 表示发送方在请求阶段未知目标 MAC，等待目标设备响应填充 |

### 跨网段ARP请求与响应

**1. 跨网段通信的基本前提**

当设备A（IP\_A）与设备B（IP\_B）处于不同子网时，数据通信必须通过路由器（网关）转发。

* 设备A不会直接发送ARP请求到目标IP\_B（因为IP\_B不在同一子网）。
* 设备A会将数据包发送给默认网关（路由器接口的MAC地址），由路由器负责转发。
* ARP请求仅用于获取默认网关的MAC地址，而非目标设备的MAC地址。

**2. 跨网段ARP请求与响应流程**

假设以下场景：

* 设备A：IP\_A = 192.168.1.100/24，默认网关 = 192.168.1.1（路由器R1的接口）。
* 设备B：IP\_B = 10.1.1.100/24，默认网关 = 10.1.1.1（路由器R2的接口）。
* 路由器R1与R2互联：R1的接口IP = 172.16.1.1，R2的接口IP = 172.16.1.2。

步骤1：设备A发送数据到设备B

判断目标IP是否在同一子网：

* 设备A计算 IP\_A & 子网掩码 和 IP\_B & 子网掩码，发现不在同一子网。

查找默认网关MAC地址：

* 检查ARP缓存表是否有网关（192.168.1.1）的MAC记录。
* 若不存在，设备A广播发送ARP请求，询问 192.168.1.1 的MAC地址。

步骤2：ARP请求与响应（获取网关MAC）

ARP请求（广播）：

以太网帧头：

- 目标MAC: FF:FF:FF:FF:FF:FF

- 源MAC: MAC\_A

ARP报文：

- Sender MAC: MAC\_A

- Sender IP: 192.168.1.100

- Target MAC: 00:00:00:00:00:00

- Target IP: 192.168.1.1

ARP响应（单播）：

路由器R1回复自身接口MAC（MAC\_R1）：

以太网帧头：

- 目标MAC: MAC\_A

- 源MAC: MAC\_R1

ARP报文：

- Sender MAC: MAC\_R1

- Sender IP: 192.168.1.1

- Target MAC: MAC\_A

- Target IP: 192.168.1.100

步骤3：数据包转发到路由器

设备A将数据包封装为：

以太网帧头：

- 目标MAC: MAC\_R1（网关）

- 源MAC: MAC\_A

IP包头：

- 源IP: 192.168.1.100

- 目标IP: 10.1.1.100

步骤4：路由器处理数据包

路由器R1查路由表：

* 确定下一跳为R2（172.16.1.2）。

路由器R1发送ARP请求（跨网段）：

* 若R1的ARP表中无 172.16.1.2 的MAC记录，广播ARP请求。

路由器R2响应ARP请求：

* 回复自身接口MAC（MAC\_R2）。

步骤5：数据包最终到达设备B

路由器R2将数据包转发到设备B所在的子网（10.1.1.0/24）：

以太网帧头：

- 目标MAC: MAC\_B（由R2通过ARP获取）

- 源MAC: MAC\_R2

IP包头：

- 源IP: 192.168.1.100（保持不变）

- 目标IP: 10.1.1.100

**3. 代理ARP（Proxy ARP）在跨网段中的应用**

如果路由器启用了代理ARP，跨网段通信可能无需配置默认网关（依赖路由器代答ARP请求）。

场景示例：

* 设备A（192.168.1.100/24）无默认网关配置，尝试访问 10.1.1.100。
* 路由器R1启用代理ARP，接口IP为 192.168.1.1。

流程：

设备A发送ARP请求：

* 目标IP为 10.1.1.100（不同子网）。

路由器R1代答ARP：

* 发现 10.1.1.100 属于其他子网，但路由器知道如何路由。
* 回复自身MAC（MAC\_R1）给设备A。

数据包转发：

* 设备A将数据包发送给R1，后续由路由器完成跨网段转发。

### 代理ARP与免费ARP

**1. 代理ARP（Proxy ARP）**

作用：路由器代替其他设备响应ARP请求，用于跨子网通信或子网掩码不匹配场景。

场景：

* 主机A（192.168.1.2/24）试图访问主机B（192.168.2.3/24），子网不同但未配置网关。
* 路由器启用代理ARP，响应A的ARP请求，伪装成B的MAC。

配置示例（Cisco路由器）：

interface GigabitEthernet0/0

ip proxy-arp ! 开启代理ARP

**2. 免费ARP（Gratuitous ARP）**

作用：

* IP冲突检测：设备启动时发送ARP请求查询自身IP是否已被占用。
* MAC地址更新：设备更换网卡后，广播免费ARP更新其他主机的ARP缓存。

报文特征：

* Sender IP = Target IP（均为自身IP）。
* Opcode = 1（请求），但无需响应。

### 拓展

**1. ARP欺骗（ARP Spoofing）**

原理：攻击者伪造ARP响应，将自身MAC绑定到他人IP，劫持流量。

防御：

* 静态ARP绑定：手动设置IP-MAC映射。
* ARP防火墙：检测异常ARP活动。

**2. 反向ARP（RARP）**

用途：通过MAC地址获取IP地址（现已被DHCP替代）。

## TCP及UDP协议

### 传输层核心作用

**1. 核心功能**

* 端到端通信：为不同主机上的应用进程提供逻辑通信（通过端口号标识服务）。
* 可靠性保障（TCP）：确保数据完整、有序、无丢失传输。
* 复用与分用：多个应用进程共享同一网络接口（通过端口号区分）。
* 流量控制（TCP）：防止发送方淹没接收方。
* 拥塞控制（TCP）：避免网络过载。

**2. 与网络层的关系**

* 网络层（IP协议）：负责主机到主机的通信（IP地址寻址）。
* 传输层：在IP基础上，实现进程到进程的通信（端口号寻址）。

### TCP协议详解

**1. TCP协议概述**

特点：面向连接、可靠传输、全双工通信、字节流传输。

适用场景：要求数据完整性的应用（如网页浏览、文件传输）。

关键字段：

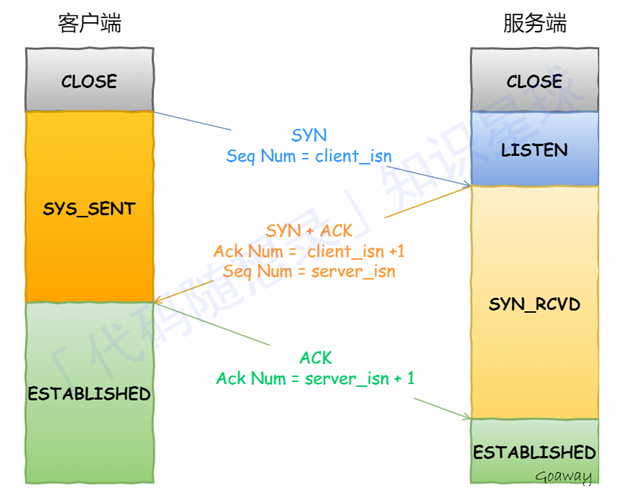
* 源端口/目的端口（16位）
* 序列号（32位）
* 确认号（32位）
* 窗口大小（16位，流量控制）
* 标志位（SYN、ACK、FIN、RST等）

**2. TCP三次握手（建立连接）**

目的：协商初始序列号（ISN），确认双方通信能力。

流程：

* SYN（Client → Server）：SYN=1, seq=x
* SYN-ACK（Server → Client）：SYN=1, ACK=1, seq=y, ack=x+1
* ACK（Client → Server）：ACK=1, seq=x+1, ack=y+1



为什么是三次握手？

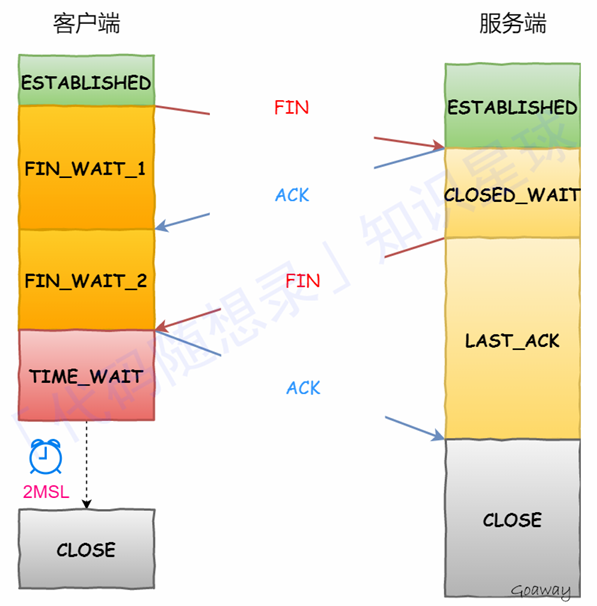
* 避免历史重复连接初始化（两次无法防止失效请求）。
* 确保双方收发能力正常（三次是最小可靠次数）。

**3. TCP四次挥手（断开连接）**

目的：双方确认数据发送完毕，安全关闭连接。

流程：

* FIN（A → B）：FIN=1, seq=u
* ACK（B → A）：ACK=1, seq=v, ack=u+1
* FIN（B → A）：FIN=1, ACK=1, seq=w, ack=u+1
* ACK（A → B）：ACK=1, seq=u+1, ack=w+1



TIME\_WAIT状态：

* 等待2MSL（最大报文生存时间），确保最后一个ACK被接收。
* 防止旧连接数据干扰新连接。

**4. RST（强制断开连接）**

* 触发条件：异常情况（如端口未监听、连接超时、数据错误）。
* 作用：立即终止连接，无需四次挥手。
* 抓包特征：RST=1，通常伴随ACK标志。

### UDP协议详解

**1. UDP协议概述**

特点：无连接、不可靠传输、面向报文、低延迟。

适用场景：实时性要求高、可容忍少量丢包的应用（如视频会议、DNS查询）。

关键字段：

* 源端口/目的端口（16位）
* 长度（16位，首部+数据总长度）
* 校验和（可选，用于错误检测）

**2. UDP与TCP对比**

表7.4 UDP与TCP对比

|  |  |  |
| --- | --- | --- |
| **特性** | **TCP** | **UDP** |
| 连接性 | 面向连接 | 无连接 |
| 可靠性 | 可靠传输（确认重传） | 不可靠传输 |
| 传输方式 | 字节流 | 数据报文 |
| 速度 | 较慢（握手、控制机制） | 极快（无控制开销） |
| 头部开销 | 20字节（最小） | 8字节 |
| 典型应用 | HTTP、FTP、SSH | DNS、VoIP、在线游戏 |

### TCP与UDP应用案例

**1. TCP应用案例**

* HTTP/HTTPS：网页传输需数据完整（如银行交易）。
* FTP：文件传输不允许丢包。
* SSH：远程登录需可靠会话。

**2. UDP应用案例**

* DNS查询：快速响应优先，失败可重试。
* 实时音视频（Zoom/Skype）：延迟敏感，丢包影响小于延迟。
* 在线游戏：高频状态更新，少量丢包可接受。

### 拓展

**1. TCP拥塞控制**

机制：慢启动、拥塞避免、快速重传、快速恢复。

目标：动态调整发送速率，避免网络拥塞。

**2. UDP的可靠性增强**

QUIC协议（基于UDP）：Google提出，在应用层实现可靠性（用于HTTP/3）。

**3. 端口号分类**

知名端口（0~1023）：HTTP（80）、HTTPS（443）、SSH（22）。

注册端口（1024~49151）：MySQL（3306）、Redis（6379）。

动态端口（49152~65535）：客户端临时使用。

## DNS协议

### DNS需求背景

**1. 为什么需要DNS？**

问题：互联网通过IP地址标识设备，但IP地址（如192.168.1.1）难以记忆，需将域名（如www.example.com）映射为IP地址。

核心需求：

* 域名解析：将人类可读的域名转换为机器可识别的IP地址。
* 负载均衡：通过DNS轮询分配请求到不同服务器。
* 服务发现：定位邮件服务器（MX记录）、CDN节点等。

**2. 历史演进**

早期方案：/etc/hosts文件手动维护域名与IP映射（不适用于大规模网络）。

DNS诞生：1983年Paul Mockapetris设计，实现分布式、层次化的域名解析系统。

### DNS概述及工作原理

**1. DNS系统组成**

域名空间：树状结构（根域、顶级域、二级域等）。

DNS服务器：

* 根DNS服务器：全球13组，管理顶级域（如.com、.org）。
* 顶级域DNS服务器：管理特定后缀域名（如.com下的所有域名）。
* 权威DNS服务器：管理具体域名的解析记录（如example.com）。
* 递归DNS服务器（本地DNS）：为用户提供查询代理服务（如运营商DNS）。

**2. DNS查询类型**

递归查询：客户端要求DNS服务器必须返回最终结果（客户端 → 本地DNS）。

迭代查询：DNS服务器返回下一级服务器地址，客户端自行查询（本地DNS → 根/顶级域）。

### DNS报文结构

**1. DNS报文格式**

头部（Header）：包含事务ID、标志位（查询/响应）、问题数等。

问题部分（Question）：查询的域名和记录类型（如A记录）。

回答部分（Answer）：返回的解析结果（如IP地址）。

权威部分（Authority）：指向域名的权威DNS服务器。

附加部分（Additional）：额外信息（如权威DNS的IP地址）。

**2. DNS查询与响应示例**

查询报文：

Header:

- Transaction ID: 0x1234

- Flags: 0x0100（标准查询）

Question:

- Name: www.example.com

- Type: A（IPv4地址）

响应报文：

Header:

- Transaction ID: 0x1234

- Flags: 0x8180（响应+递归可用）

Answer:

- Name: www.example.com

- Type: A

- TTL: 300

- Data: 93.184.216.34

### DNS常见记录类型及作用

表7.5 DNS常见记录类型及作用

|  |  |  |  |
| --- | --- | --- | --- |
| **记录类型** | **全称** | **作用** | **应用场景** |
| A | Address Record | 将域名解析为IPv4地址 | www.example.com → 192.0.2.1 |
| AAAA | IPv6 Address Record | 将域名解析为IPv6地址 | www.example.com → 2001:db8::1 |
| NS | Name Server Record | 指定域名的权威DNS服务器 | example.com → ns1.example.com |
| MX | Mail Exchange Record | 指定域名的邮件服务器 | example.com → mail.example.com |
| CNAME | Canonical Name Record | 域名别名（指向另一个域名） | blog.example.com → www.example.com |
| TXT | Text Record | 存储文本信息（如SPF、DKIM验证） | 邮件防伪、域名所有权验证 |
| PTR | Pointer Record | 反向解析（IP → 域名） | 192.0.2.1 → www.example.com |

### DNS请求与响应过程详解

**1. 本地DNS缓存**

* 客户端首先检查本地缓存（浏览器、操作系统）是否有域名解析结果。
* 若缓存命中且未过期（TTL有效），直接使用缓存结果。

**2. 递归查询流程**

客户端向本地DNS服务器发送查询请求（如8.8.8.8）。

本地DNS服务器依次查询：

* 根DNS → 获取.com的顶级域DNS地址。
* 顶级域DNS → 获取example.com的权威DNS地址。
* 权威DNS → 返回www.example.com的A记录。

本地DNS缓存结果并返回给客户端。

**3. 抓包分析（Wireshark）**

过滤条件：dns

关键字段：

* Transaction ID：匹配请求与响应。
* Flags：区分查询（0x0100）与响应（0x8180）。
* Queries/Answers：查看解析结果。

### 拓展

**1. DNS安全**

* DNS劫持：篡改DNS响应，引导用户至恶意网站。
* DNSSEC：通过数字签名验证DNS响应真实性。

**2. CDN与DNS**

地理解析：根据用户位置返回最近的服务器IP（通过A记录实现负载均衡）。

**3. 为什么DNS使用UDP？**

UDP无连接、速度快，适合小数据量的查询（默认端口53）。

若响应超过512字节，自动切换为TCP（如DNSSEC场景）。

**4. TTL的作用是什么？**

Time To Live：控制DNS记录在缓存中的存活时间（单位：秒）。

短TTL：快速更新记录（如故障转移）。

长TTL：减少查询压力（如静态资源域名）。

**5. MX记录的优先级如何工作？**

数值越小优先级越高（如10 mail1.example.com优先于20 mail2.example.com）。

## HTTP与HTTPS协议

### HTTP协议

**1. HTTP需求背景**

核心问题：早期互联网需要一种标准化方式传输超文本（如网页），但缺乏统一的请求-响应规则。

设计目标：

* 简单、灵活：支持多种数据类型（HTML、图片、视频）。
* 无状态：每次请求独立，简化服务器设计。

演进历史：

* HTTP/0.9（1991）：仅支持GET方法，无头部。
* HTTP/1.0（1996）：引入状态码、头部字段。
* HTTP/1.1（1997）：持久连接、分块传输。
* HTTP/2（2015）：二进制协议、多路复用。

**2. HTTP概述及工作原理**

基本特性：

* 无状态：服务器不保存客户端状态（依赖Cookie/Session）。
* 明文传输：数据未加密，易被窃听或篡改。
* 基于TCP：默认端口80，需三次握手建立连接。

请求-响应模型：

* 客户端（浏览器）发送HTTP请求（如GET /index.html）。
* 服务器返回HTTP响应（如200 OK + HTML内容）。

**3. HTTP报文结构**

请求报文：

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

Accept: text/html

响应报文：

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 1234

<html>...</html>

关键字段：

表7.6 HTTP报文关键字段

|  |  |  |
| --- | --- | --- |
| **请求头** | **响应头** | **说明** |
| Host | Server | 目标主机/服务器软件信息 |
| User-Agent | Content-Type | 客户端标识/响应数据类型 |
| Cookie | Set-Cookie | 客户端/服务器管理会话状态 |
| Accept | Content-Length | 客户端支持的数据类型/数据长度 |

**4. HTTP方法**

表7.7 HTTP方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **作用** | **幂等性** | **安全性** |
| GET | 获取资源 | 是 | 是 |
| POST | 提交数据（如表单） | 否 | 否 |
| PUT | 更新资源（全量替换） | 是 | 否 |
| DELETE | 删除资源 | 是 | 否 |

**5. 状态码**

表7.8 HTTP状态码

|  |  |  |
| --- | --- | --- |
| **状态码** | **类别** | **常见示例** |
| 1xx | 信息性响应 | 100（继续） |
| 2xx | 成功 | 200（OK）、201（Created） |
| 3xx | 重定向 | 301（永久重定向）、302（临时重定向） |
| 4xx | 客户端错误 | 404（未找到）、403（禁止访问） |
| 5xx | 服务器错误 | 500（内部错误）、503（服务不可用） |

### HTTPS协议

**1. HTTPS需求背景**

HTTP的安全缺陷：

* 窃听风险：明文传输易被中间人截获（如密码、信用卡号）。
* 篡改风险：攻击者可修改传输内容（如插入广告、恶意脚本）。
* 伪装风险：仿冒网站欺骗用户（如钓鱼网站）。

解决方案：通过加密（SSL/TLS）保障数据机密性、完整性、身份认证。

**2. HTTPS概述及工作原理**

基本特性：

* 加密传输：数据经SSL/TLS加密，防窃听与篡改。
* 身份认证：通过数字证书验证服务器身份。
* 基于TCP：默认端口443，兼容HTTP语义。

核心流程：

* TCP三次握手：建立连接。
* TLS握手：协商加密算法、交换密钥、验证证书。
* 加密通信：应用数据通过对称加密传输。

**3. TLS/SSL握手过程**

Client Hello：

* 客户端支持TLS版本、加密套件列表、随机数。

Server Hello：

* 服务器选择加密套件、发送随机数、证书（含公钥）。

证书验证：

* 客户端验证证书合法性（颁发机构、有效期、域名匹配）。

密钥交换：

* 客户端生成预主密钥，用服务器公钥加密后发送。

生成会话密钥：

* 双方通过随机数和预主密钥生成对称密钥。

加密通信：

* 使用对称密钥加密HTTP数据。

**4. HTTPS报文结构**

加密后数据：HTTP报文经TLS加密后传输，无法直接阅读明文。

抓包示例（Wireshark）：

* 过滤条件：tcp.port == 443，可看到TLS握手和加密数据流。

### HTTP与HTTPS对比

表7.9 HTTP与HTTPS对比

|  |  |  |
| --- | --- | --- |
| **特性** | **HTTP** | **HTTPS** |
| 协议 | 明文传输 | 加密传输（SSL/TLS） |
| 端口 | 80 | 443 |
| 安全性 | 无加密，易受攻击 | 防窃听、篡改、伪装 |
| 性能 | 无加密开销，更快 | 加密增加延迟（可优化） |
| 应用场景 | 非敏感信息传输 | 登录、支付、API接口 |

### HTTP/HTTPS案例分析

**1. HTTP明文传输风险**

* 场景：用户通过HTTP登录，密码明文传输。
* 抓包分析：使用Wireshark可直接看到POST请求中的密码字段。

**2. HTTPS加密保护**

* 场景：用户访问https://bank.example.com，提交信用卡信息。
* 抓包分析：仅能看到TLS握手过程，应用数据为加密密文。

**3. 混合内容（Mixed Content）**

* 问题：HTTPS页面加载HTTP资源（如图片、脚本），浏览器警告不安全。
* 解决方案：确保所有资源使用HTTPS加载。

### 拓展

**1. 数字证书与CA**

证书内容：域名、公钥、颁发机构、有效期。

CA（证书颁发机构）：受信任的第三方（如Let's Encrypt、DigiCert）。

证书链：根证书 → 中间证书 → 域名证书。

**2. 性能优化**

TLS会话恢复：通过Session ID或Session Ticket减少握手开销。

OCSP Stapling：服务器主动提供证书状态，减少客户端查询延迟。

**3. HTTP/2与HTTP/3**

HTTP/2：多路复用、头部压缩、服务器推送。

HTTP/3：基于QUIC（UDP），解决队头阻塞，提升弱网性能。

### 浏览器地址栏输入URL回车后涉及到的流程

**1. 查找DNS缓存**

* 先查找浏览器DNS缓存，看是否存放目标网络的IP地址；
* 如果不在浏览器缓存，则浏览器将对操纵系统发起系统调用，查询操作系统本地缓存；
* 如果不在操作系统本地缓存，则浏览器会查询与之相连的路由器缓存；
* 如果不在路由器缓存，则浏览器会检查ISP【本地通信服务商】缓存；

若以上四步均没有查询到目标网络的IP地址，则发起DNS查询。

**2. 发起DNS查询**

判断DNS服务器和我们的主机是否在同一子网内

* 在同一子网，则采用ARP地址解析协议对DNS服务器进行ARP查询
* 不在同一子网，则采用ARP地址解析协议对默认网关进行查询

若此时还是查询不到IP地址，则根据拿到DNS服务器或者默认⽹关的IP地址，继续进⾏DNS请求

使用53端⼝先向本地DNS服务器发送UDP请求包，此处一般使用UDP协议（如果响应包太⼤，则使用TCP协议）

没有查询到IP地址：

则它会发送一个递归查询请求，一层一层向高层DNS服务器查询，直到查询到IP地址，则将结果返回

【解释：DNS是分布式域名服务器，每台服务器只维护⼀部分IP地址到网络地址的映射，没有任何⼀台服务器能够维持全部的映射关系】。

**3. 封装TCP数据包**

拿到IP地址后，根据URL中的端⼝可知端⼝号【HTTP：80；HTTPS：443】，一般先会先尝试建立HTTP连接；

准备TCP数据包：

步骤：

* 将应用层传递下来的实际数据，在传输层添加TCP首部；
* 将传输层传下来的数据在⽹络层添加IP首部；
* 将网络层传输下来的数据，在数据链路层添加以太网首部，并在传输介质中进⾏传输。

**4. 浏览器与目标服务器建立TCP连接**

经过上述DNS和ARP查询流程后，浏览器会收到目标服务器的IP和MAC地址，然后经过三次握手后建立TCP连接；

使用HTTP协议：

浏览器发送请求到服务器，如果使用的是HTTP协议，则服务器直接返回结果；

使用HTTPS协议：

如果不是HTTP协议，则服务器会返回一个以3开头的重定向消息，告诉浏览器使用的HTTPS，IP没变，只是端口号变成443；完成四次挥手；

重新建立TCP连接，将端口号修改为443，同时沟通好双方的使用的认证算法、加密和解密算法，在次过程中也会检查对方的CA安全证书，采用SSL加密技术进⾏传输数据。

**5. 浏览器发送HTTP/HTTPS请求到web服务器**

主要使用两种请求方式：

* 浏览器发送get请求，要求目标服务器提供输⼊的网页；
* 浏览器发送post请求，表示填写的是表单。

**6. 服务器处理请求并返回一个响应**

服务器会从浏览器接受请求并将其传递给请求处理程序并响应；

**7. 服务器发回一个HTTP响应**

⼀般响应包包含：请求的网页以及状态码，压缩类型，如何缓存的页面，设置的cookie；

**8. 浏览器显示HTML页面**

* 渲染HTML骨架；涉及到Ajax技术；
* 检查HTML标记并发送GET请求以获取网页上的其他元素【图像、CSS样式、JS文件等】，该静态文件一般由浏览器缓存，再次访问，不用重新请求；
* 最后会看到请求色彩斑斓的网页。

## DHCP协议

### DHCP需求背景

**1. 为什么需要DHCP？**

手动分配IP的痛点：

* 管理繁琐：大型网络中手动配置IP地址效率低下。
* 易冲突：人工操作易导致IP重复分配。
* 灵活性差：设备移动或网络变更需重新配置。

核心需求：

* 动态分配：自动为设备分配IP地址、子网掩码、网关、DNS等参数。
* 租期管理：IP地址按需分配，避免资源浪费。

### DHCP协议详解

**1. DHCP概述**

全称：Dynamic Host Configuration Protocol（动态主机配置协议）。

作用：自动为网络设备分配IP地址及其他配置参数。

协议版本：

* DHCPv4：用于IPv4网络（默认端口67/UDP服务器，68/UDP客户端）。
* DHCPv6：用于IPv6网络（端口546/UDP客户端，547/UDP服务器）。

**2. DHCP工作原理（DORA流程）**

Discover（客户端广播）：

* 客户端发送DHCP Discover报文，寻找可用DHCP服务器。

Offer（服务器单播/广播）：

* 服务器回应DHCP Offer，提供IP地址及配置参数。

Request（客户端广播）：

* 客户端选择某服务器的Offer，发送DHCP Request确认请求。

Ack（服务器单播/广播）：

* 服务器最终确认分配，发送DHCP Ack，客户端正式使用IP。

**3.** **DHCP报文结构**

通用字段：

表7.10 DHCP报文结构

|  |  |
| --- | --- |
| **字段** | **说明** |
| Op | 报文类型（1=请求，2=响应） |
| Htype/Hlen | 硬件类型/地址长度（如以太网为1/6） |
| Xid | 事务ID（匹配请求与响应） |
| Flags | 广播标志（客户端未分配IP时设为广播） |
| Client MAC | 客户端MAC地址 |
| Your IP | 服务器分配给客户端的IP地址 |
| Options | 可配置参数（子网掩码、网关、DNS、租期等） |

常见报文类型：

表7.11 DHCP报文类型

|  |  |
| --- | --- |
| **类型** | **说明** |
| DHCP Discover | 客户端发起，寻找可用服务器 |
| DHCP Offer | 服务器响应，提供IP配置 |
| DHCP Request | 客户端确认选择某服务器的Offer |
| DHCP Ack | 服务器最终确认分配 |
| DHCP NAK | 服务器拒绝请求（如IP已分配） |
| DHCP Release | 客户端主动释放IP地址 |

### DHCP案例

**1. 家庭网络**

场景：家庭路由器内置DHCP服务器，为手机、电脑分配IP地址。

流程：

* 手机连接Wi-Fi，发送DHCP Discover。
* 路由器回应DHCP Offer（如IP：192.168.1.100）。
* 手机发送DHCP Request确认。
* 路由器发送DHCP Ack，手机获得IP并联网。

**2. 企业网络**

场景：多子网环境中，通过DHCP中继实现跨网段IP分配。

流程：

* 客户端在子网A发送DHCP Discover（广播）。
* 中继代理（路由器）将请求转发至子网B的DHCP服务器。
* 服务器通过中继代理回应DHCP Offer。
* 客户端完成DORA流程获取IP。

### DHCP中继与DHCP Snooping

**1. DHCP中继（Relay Agent）**

应用场景：跨子网环境中，客户端与DHCP服务器不在同一广播域。

工作原理：

* 中继代理监听客户端的DHCP广播报文。
* 将报文单播转发至指定DHCP服务器（修改报文中的giaddr字段为客户端子网地址）。
* 服务器根据giaddr确定客户端所属子网，分配对应IP池中的地址。

**2. DHCP Snooping**

安全威胁：恶意用户伪造DHCP服务器（Rogue DHCP），分配错误IP或劫持流量。

工作原理：

信任端口与非信任端口：

* 信任端口：连接合法DHCP服务器（允许所有DHCP报文）。
* 非信任端口：连接客户端（仅允许DHCP请求，过滤服务器响应）。

绑定表（Binding Table）：

* 记录客户端MAC、IP、租期、VLAN、端口等信息。

过滤非法报文：

* 非信任端口的DHCP Offer/Ack报文被丢弃，防止Rogue DHCP攻击。

### 拓展

**1. DHCP租期更新**

* T1时间（50%租期）：客户端尝试向原服务器续租（单播DHCP Request）。
* T2时间（87.5%租期）：客户端广播DHCP Request，寻找其他服务器。
* 租期到期：客户端释放IP，重新发起DORA流程。

**2. DHCPv6与IPv6**

* 无状态模式：设备通过RA（Router Advertisement）获取网络前缀，自行生成IP。
* 有状态模式：类似DHCPv4，服务器分配完整IP地址。

**3. 常见问题排查**

IP冲突：检查DHCP地址池范围，确认无静态IP重叠。

客户端无法获取IP：

* 检查DHCP服务器是否在线，地址池是否耗尽。
* 抓包分析DORA流程是否完整（Wireshark过滤bootp）。

**4. 客户端收不到DHCP Offer的可能原因？**

* DHCP服务器未启动或地址池耗尽。
* 客户端与服务器间存在防火墙阻止UDP 67/68端口。

**5. DHCP Snooping如何防御中间人攻击？**

* 通过绑定表验证客户端IP-MAC-VLAN-端口的一致性，丢弃非法报文。

**6. DHCP中继的giaddr字段作用？**

* 告知服务器客户端所在子网，确保分配对应子网的IP地址。

## FTP协议

### FTP需求背景

**1. 为什么需要FTP？**

核心需求：早期互联网需要一种标准化方式在不同系统（如Windows、Unix）之间传输文件。

关键特性：

* 跨平台支持：兼容不同操作系统。
* 可靠传输：基于TCP协议，确保文件完整性。
* 目录管理：支持文件列表、目录切换、文件删除等操作。

历史地位：在HTTP普及前，FTP是文件共享的主要协议（现多用于内部网络或特定场景）。

**2.** **FTP的优缺点**

表7.12 FTP的优缺点

|  |  |
| --- | --- |
| **优点** | **缺点** |
| 支持大文件传输 | 明文传输（用户名、密码易被窃听） |
| 支持断点续传 | 配置复杂（需处理主动/被动模式） |
| 可管理远程文件系统 | 默认使用多个端口，防火墙难配置 |

### FTP协议概述

**1. 基本概念**

双通道设计：

* 控制连接：默认端口21，传输命令（如登录、文件操作）。
* 数据连接：默认端口20（主动模式）或随机端口（被动模式），传输文件内容。

传输模式：

* ASCII模式：文本文件自动转换格式（如换行符）。
* 二进制模式：原样传输文件（如图片、压缩包）。

**2. FTP工作模式**

（1）主动模式（Active Mode）

流程：

* 客户端随机端口N连接服务器21端口（控制连接）。
* 客户端发送PORT命令告知服务器自己的数据端口（N+1）。
* 服务器从20端口主动连接客户端的数据端口（N+1）。
* 文件传输完成后，数据连接关闭。

防火墙问题：客户端需开放数据端口，可能被防火墙拦截。

（2）被动模式（Passive Mode）

流程：

* 客户端随机端口M连接服务器21端口（控制连接）。
* 客户端发送PASV命令，服务器回应随机数据端口P。
* 客户端从端口M+1连接服务器的数据端口P。
* 文件传输完成后，数据连接关闭。

防火墙优势：服务器开放数据端口，客户端无需配置入站规则。

**3. 主动模式 vs 被动模式**

表7.13 FTP主动被动区别

|  |  |  |
| --- | --- | --- |
| **特性** | **主动模式** | **被动模式** |
| 数据连接方向 | 服务器主动连接客户端 | 客户端主动连接服务器 |
| 客户端防火墙 | 需开放数据端口（易被拦截） | 无需特殊配置（推荐） |
| 服务器防火墙 | 无需特殊配置 | 需开放随机数据端口（需配置NAT） |
| 应用场景 | 内网环境或无防火墙限制 | 公网或客户端有严格防火墙 |

### FTP案例

**1. 文件下载流程（被动模式）**

建立控制连接：

* 客户端（端口1234） → 服务器21端口：TCP三次握手。

用户认证：

* 客户端发送：USER anonymous
* 服务器响应：331 Password required
* 客户端发送：PASS guest@
* 服务器响应：230 Login successful

进入被动模式：

* 客户端发送：PASV
* 服务器响应：227 Entering Passive Mode (192,168,1,100,203,112)

数据端口 = 203×256 + 112 = 52080

发起数据连接：

* 客户端（端口1235） → 服务器52080端口：TCP三次握手。

传输文件：

* 客户端发送：RETR file.txt
* 服务器响应：150 Opening data connection
* 文件传输完成后，服务器响应：226 Transfer complete

**2. 常见FTP客户端工具**

FileZilla：开源跨平台FTP客户端，支持主动/被动模式切换。

命令行工具：

ftp ftp.example.com # 连接FTP服务器

ls # 列出文件

get file.txt # 下载文件

put file.txt # 上传文件

### FTP协议报文结构

**1. 控制连接报文（明文）**

命令格式：命令 参数\r\n

示例：

USER anonymous\r\n

PASS guest@\r\n

RETR file.txt\r\n

响应格式：状态码 描述\r\n

示例：

220 Welcome to FTP Server\r\n

230 Login successful\r\n

226 Transfer complete\r\n

常见状态码：

表7.14 FTP常见状态码

|  |  |
| --- | --- |
| **状态码** | **说明** |
| 200 | 命令成功执行 |
| 220 | 服务就绪 |
| 230 | 登录成功 |
| 550 | 文件不可用（权限不足） |

**2. 数据连接报文**

二进制模式：直接传输文件字节流。

ASCII模式：自动转换文本文件格式（如\n → \r\n）。

### 拓展

**1. FTP安全性问题**

明文传输：用户名、密码、文件内容均未加密，易被嗅探。

解决方案：

* FTPS：基于SSL/TLS加密的FTP（端口990控制连接）。
* SFTP：SSH文件传输协议（与FTP无关，基于SSH加密）。

**2. FTP与防火墙/NAT**

* 主动模式问题：NAT设备无法正确映射服务器主动连接的数据端口。
* 被动模式配置：需在服务器端配置被动端口范围，并在防火墙/NAT中开放。

**3. 匿名FTP**

* 匿名登录：用户名为anonymous，密码为任意邮箱（如guest@）。
* 风险：若服务器配置不当，可能导致未授权文件访问。

**4. 客户端无法连接FTP服务器？**

检查防火墙是否放行21端口（控制连接）及被动模式端口范围。

确认服务器是否支持主动/被动模式。

**5. 文件传输中断如何处理？**

使用支持断点续传的客户端（如FileZilla），重新连接后继续传输。

**6. 如何提升FTP安全性？**

改用FTPS或SFTP协议，禁用匿名登录，限制用户权限。

## 邮件协议

### 邮件协议需求背景

**1. 为什么需要邮件协议？**

核心需求：实现用户间跨网络、跨系统的可靠邮件传递。

关键挑战：

* 异构系统兼容：不同邮件服务器（如Gmail、Outlook）需互通。
* 异步通信：收件人不在线时仍能接收邮件。
* 安全性：防止邮件伪造、窃听、篡改。

**2. 邮件系统组成**

* 用户代理（UA）：客户端（如Outlook）或网页端（如Gmail）。
* 邮件服务器：负责存储、转发邮件（如SMTP服务器、POP3服务器）。
* 协议：SMTP（发）、POP3/IMAP（收）、HTTP（网页邮件）。

### 发送邮件过程（SMTP）

**1. 流程概述**

用户撰写邮件：通过客户端或网页端填写收件人、主题、内容。

提交到发送方SMTP服务器：

* 客户端使用SMTP协议将邮件发送到发件人所属的SMTP服务器（如smtp.gmail.com:587）。

SMTP服务器中继：

* 发送方SMTP服务器通过DNS查询收件人域名的MX记录，找到接收方SMTP服务器（如mail.example.com:25）。

接收方SMTP服务器存储邮件：

* 邮件存入收件人邮箱（若地址有效），等待用户收取。

**2. SMTP协议交互**

命令示例（明文通信）：

EHLO example.com # 客户端标识

MAIL FROM:<sender@example.com>

RCPT TO:<receiver@example.com>

DATA # 开始传输邮件内容

From: sender@example.com

To: receiver@example.com

Subject: Hello

This is the message body.

. # 结束DATA部分

QUIT

响应码：

表7.15 SMTP响应码

|  |  |
| --- | --- |
| **状态码** | **说明** |
| 220 | 服务就绪 |
| 250 | 请求操作成功 |
| 354 | 开始邮件输入 |

### 接收邮件过程（POP3/IMAP）

**1. POP3协议（邮局协议版本3）**

特点：下载邮件到本地后删除服务器副本（默认配置），适合单设备访问。

交互流程：

* 授权阶段：客户端认证（用户名/密码）。
* 事务阶段：列出邮件、下载邮件。
* 更新阶段：删除服务器邮件（可选）。

命令示例：

USER receiver@example.com

PASS password

LIST # 列出邮件列表

RETR 1 # 下载第1封邮件

DELE 1 # 删除第1封邮件

QUIT

**2. IMAP协议（互联网邮件访问协议）**

特点：邮件保留在服务器，支持多设备同步和文件夹管理，适合多设备访问。

核心功能：

* 在线管理邮件（标记已读、移动至文件夹）。
* 同步服务器与客户端的邮件状态。

### 常见邮件协议对比

表7.16 常见邮件协议对比

|  |  |  |  |
| --- | --- | --- | --- |
| **协议** | **端口（明文/加密）** | **功能** | **适用场景** |
| SMTP | 25/587（STARTTLS）  465（SMTPS） | 发送邮件 | 邮件提交与中继 |
| POP3 | 110/995（POP3S） | 下载邮件到本地 | 单设备离线访问 |
| IMAP | 143/993（IMAPS） | 同步管理服务器邮件 | 多设备在线访问 |
| HTTP | 80/443 | 网页邮件交互 | Gmail、Outlook网页版 |

### 网页版与客户端版收发邮件案例

**1. 网页版邮件（如Gmail）**

发送流程：

* 浏览器通过HTTPS将邮件提交到Gmail的SMTP服务器。
* Gmail服务器使用SMTP协议中继邮件至收件人服务器。

接收流程：

* 浏览器通过HTTPS从Gmail的IMAP服务器拉取邮件列表。
* 用户点击邮件后，服务器返回邮件内容。

**2. 客户端版邮件（如Outlook）**

发送流程：

* Outlook通过SMTP协议将邮件发送到配置的SMTP服务器（如smtp.office365.com:587）。

接收流程：

* Outlook通过POP3/IMAP协议从邮件服务器下载邮件。

### 协议报文结构

**1. SMTP报文（明文）**

请求：

MAIL FROM:<sender@example.com>\r\n

RCPT TO:<receiver@example.com>\r\n

DATA\r\n

...邮件头及正文...

\r\n.\r\n

响应：

250 OK\r\n

354 Start mail input\r\n

**2. POP3报文（明文）**

请求：

RETR 1\r\n

响应：

+OK 1200 octets\r\n

...邮件内容...

\r\n.\r\n

### 拓展

**1. 邮件安全**

* SSL/TLS加密：SMTPS（465）、IMAPS（993）、POP3S（995）。
* SPF/DKIM/DMARC：防伪造、防钓鱼技术。

**2. 邮件格式（MIME）**

* 多部分内容：支持文本、附件、HTML格式。
* 编码：Base64编码处理二进制附件。

**3. 垃圾邮件过滤**

* 贝叶斯过滤：基于内容关键词概率分析。
* 黑名单机制：拦截已知垃圾邮件服务器IP。

**4. 邮件发送失败的可能原因？**

SMTP服务器配置错误（端口、加密方式）。

收件人地址不存在或服务器拒收（检查SPF/DKIM记录）。

**5. POP3与IMAP如何选择？**

POP3：单设备离线访问，节省服务器存储空间。

IMAP：多设备同步，需服务器支持。

**6. 为什么网页邮件使用HTTP而非SMTP？**

HTTP更适合交互式操作（如富文本编辑、实时刷新），SMTP仅用于后端邮件传输。

## RADIUS协议

### AAA概述

**1. AAA定义**

* 认证（Authentication）：验证用户身份（如用户名/密码）。
* 授权（Authorization）：确定用户可访问的资源或权限。
* 计费（Accounting）：记录用户资源使用情况（如时长、流量）。

**2. AAA应用场景**

网络接入控制：Wi-Fi认证、VPN登录、拨号上网。

设备管理：路由器、交换机管理员登录验证。

### RADIUS需求背景

**1. 为什么需要RADIUS？**

集中化管理：早期网络设备各自维护用户认证信息，管理成本高。

标准化协议：不同厂商设备需统一认证交互方式。

安全增强：支持加密传输（如密码隐藏）、审计日志。

**2. 替代方案对比**

TACACS+：Cisco主导，更适用于设备管理（如CLI权限分级）。

Diameter：RADIUS升级版，支持更多扩展（如LTE网络）。

### RADIUS协议详解

**1. RADIUS概述**

全称：Remote Authentication Dial-In User Service。

作用：实现AAA功能的客户端-服务器协议。

特点：

* 基于UDP（端口1812认证，1813计费）。
* 支持多种认证方式（PAP、CHAP、EAP）。
* 灵活扩展属性（Attribute-Value Pairs, AVPs）。

**2. RADIUS工作原理**

基本交互流程（以Wi-Fi认证为例）：

用户发起连接：

* 设备（如手机）尝试连接Wi-Fi接入点（AP）。

NAS（网络接入服务器）转发请求：

* AP（作为RADIUS客户端）向RADIUS服务器发送Access-Request报文。

RADIUS服务器验证：

* 检查用户凭证（如用户名/密码）。
* 返回Access-Accept（成功）或Access-Reject（失败）。

授权与计费：

* 若认证成功，服务器下发授权策略（如VLAN、带宽限制）。
* 开始计费会话（发送Accounting-Start报文）。

**3. RADIUS报文结构**

报文头（Header）：

表7.17 RADIUS报文头

|  |  |  |
| --- | --- | --- |
| **字段** | **长度 (字节)** | **说明** |
| Code | 1 | 报文类型（如1=Access-Request） |
| Identifier | 1 | 事务ID（匹配请求与响应） |
| Length | 2 | 报文总长度 |
| Authenticator | 16 | 认证字（用于报文完整性校验） |

属性（Attributes）：

表7.18 RADIUS报文属性

|  |  |  |
| --- | --- | --- |
| **属性类型** | **值（示例）** | **说明** |
| User-Name | user1 | 用户名 |
| User-Password | 加密值 | 用户密码（MD5加密） |
| NAS-IP-Address | 192.168.1.100 | NAS设备IP地址 |
| Framed-IP-Address | 10.1.1.100 | 为用户分配的IP地址 |
| Service-Type | 2（Framed） | 服务类型（如拨号上网） |

### RADIUS案例

**1. 企业Wi-Fi认证**

场景：员工连接企业Wi-Fi需输入AD域账号。

流程：

* 员工设备连接SSID，触发RADIUS认证。
* AP（RADIUS客户端）转发请求至RADIUS服务器（如FreeRADIUS）。
* RADIUS服务器与AD域控制器同步验证账号。
* 认证通过后，分配VLAN并记录会话日志。

**2. ISP拨号上网**

场景：用户通过PPPoE拨号接入互联网。

流程：

* 用户输入宽带账号密码，发起PPPoE连接。
* BRAS（Broadband Remote Access Server）发送RADIUS请求。
* RADIUS服务器验证用户并下发带宽策略。

### 拓展

**1. RADIUS与EAP**

EAP（Extensible Authentication Protocol）：在RADIUS中扩展支持更安全的认证方式（如EAP-TLS、PEAP）。

应用场景：WPA2-Enterprise无线认证（如802.1X）。

**2. RADIUS属性扩展**

厂商特定属性（VSA）：允许厂商自定义属性（如Cisco的Cisco-AVPair）。

格式：Vendor-Specific (26) + 厂商ID + 子属性。

**3. 安全性增强**

共享密钥（Shared Secret）：客户端与服务器预共享密钥，用于加密敏感属性（如密码）。

报文完整性：通过Authenticator字段验证报文未被篡改。

**4. RADIUS为什么使用UDP而非TCP？**

实时性：UDP无连接开销，适合频繁的认证请求。

轻量级：RADIUS报文通常较小，UDP足够可靠（应用层可重试）。

**5.** **RADIUS与TACACS+的区别？**

表7.19 RADIUS与TACACS+的区别

|  |  |  |
| --- | --- | --- |
| **特性** | **RADIUS** | **TACACS+** |
| 传输协议 | UDP | TCP |
| 加密方式 | 仅加密密码 | 加密整个报文 |
| 授权粒度 | 较粗（基于会话） | 精细（基于命令级） |
| 主要场景 | 网络接入 | 设备管理（如路由器） |

**6. 如何排查RADIUS认证失败？**

* 检查共享密钥是否一致。
* 确认NAS与RADIUS服务器网络连通性。
* 查看服务器日志（如FreeRADIUS的/var/log/freeradius/radius.log）。