CS3611 计算机网络

课程大作业

520030910002 周怡清 520030910238 王艺诺

摘要

计算机网络近年来它获得了飞速的发展，现在已成为我们社会结构的一个基本组成部分，被广泛应用于社会的各个领域。计算机技术的发展表现为网络化，同时，计算机网络将呈现出全连接的，开放的，传输多媒体信息的特点。在计算机网络中，代理服务器是一种重要的服务器安全功能，它的工作主要在开放系统互联（OSI）模型的会话层，从而起到防火墙和内容响应的作用。此外还包括安全性、缓存、内容过滤、访问控制管理等功能。

本文的主要工作如下：

（1）参考课本 [1]，基于互联网体系结构模型概括了计算机网络各层次相关内容，包括但不限于功能，结构，定义，相关协议标准，以及各层之间的接口等，并探讨了近年来网络的发展和代理服务器的应用与发展。

（2）基于 Python 套接字编程实现了一个具有缓存功能，支持现行各种 http 协议的代理服务器，并展示了代理服务器的使用方法和运行结果。

目录

1 综述 **5**

1.1 概述 5

1.2 应用层 5

1.2.1 概念定义 5

1.2.2 典型应用 6

1.3 运输层 7

1.3.1 传输控制协议（TCP） 7

1.3.2 用户数据报协议（UDP） 10

1.3.3 多路复用与多路分解 10

1.4 网络层 10

1.4.1 路由 11

1.4.2 路由器算法 11

1.4.3 网络层协议–IP 12

1.4.4 SDN 15

1.4.5 网络管理 16

1.5 链路层 17

1.5.1 差错检验和纠正技术 17

1.5.2 MAC 协议 17

1.5.3 交换局域网 18

1.6 物理层 19

1.7 无线网络和移动网络 19

1.7.1 无线网络 19

1.7.2 移动网络 20

1.8 互联网发展情况 20

1.9 代理服务器的最新发展 21

1.10 小结 21

2 Web 缓存代理服务器实验 **22**

2.1 Web 缓存代理服务器 22

2.2 项目实现及功能介绍 24

2.2.1 缓存器 24

2.2.2 代理服务器 24

2.3 技术细节 24

2.3.1 多线程代理服务 24

2.3.2 键盘终止 24

2.3.3 头部字段分析 24

2.3.4 ssl 加密处理 25

2.3.5 使用 select 模块实现 http2.0 25
2.3.6 Web 动态分配缓存 26
2.4 实验效果展示 26

3 总结与展望 **28**

术语表

中文名称	英文简写	解释
对等	P2P	
超文本传输协议	HTTP	
域名系统	DNS	
资源记录	RR	包含（Name,Value,Type,TTL）的四元组
内容分发网	CDN	一种透过互联网互相连接的计算机网络系统，利用最靠近每位用户的服务器提供高性能、可扩展性及低成本的网络内容传递给用户。
经 HTTP 的动态适应流	DASH	是基于 HTTP 的使用 TCP 传输协议的流媒体传输技术。
传输控制协议	TCP	可靠的面向连接服务的运输层协议
用户数据报协议	UDP	不可靠，无连接的运输层协议
可靠数据传输协议	rdt	运输层为上层提供可靠数据传输的抽象服务的协议
回退 N 步	GBN	流水线可靠数据传输中差错恢复的基本方法
选择重传	SR	
软件定义网络	SDN	
网络地址转换	NAT	
无类别域际路由选择	CIDR	
网络接口卡	NIC	又名网络适配器，工作在链路层的网络组件，是局域网中连接计算机和传输介质的接口，核心是链路层控制器。
链路层地址	MAC 地址	又名 LAN 地址，物理地址
信噪比	SNR	无线网络接收方收到的信号与噪声强度的相对测量

1 综述

1.1 概述

计算机网络是如今最重要的信息通信和信息服务基础设施，它用以支持计算机主机之间的数据交互。本文将以因特网为例来讨论计算机网络。因特网由大量简单网络互联构成，是当今世界最大的互联网。主机与主机通过通信链路和分组交换机连接到一起，。因特网的边缘部分由主机构成，主机通过不同的物理介质与边缘路由器相连，形成接入网。因特网的核心由分组交换机与连接路由器的链路构成，为因特网网边缘提供通信服务。主机通过**因特网服务提供商（Internet Service Provider, ISP）**接入网络，不同层级的 ISP 互联，最终为主机供应内容。

1977 年，国际标准化组织 (ISO) 创建了互联网标准体系结构模型 (OSI)，从而统一了开放系统互联的网络标准。OSI 是一种 7 层的网络模型。每一层向它的上一层提供服务，最终实现应用程序的交互。服务由各层的协议实现，由各层组织这些协议。本文将引入如表 1所示的 5 层协议模型。

OSI 参考协议模型	5 层协议模型
应用层 表示层 会话层	应用层
运输层	运输层
网络层	网络层
链路层	链路层
物理层	物理层

表 1: OSI 参考协议模型与 5 层协议模型服务对比

互联网也经历了多次的更新迭代后形成了如今的层次模型。网络的发展历史可以用图 1表示

图 1: 网络的发展历史

1.2 应用层

1.2.1 概念定义

网络应用层的程序限制在端系统上，不包括网络核心设备的应用程序软件。主要的应用程序体系结构有**客户-服务器体系结构**和**对等（P2P）**体系结构。客户-服务器体系结构的主机分为服务器和客户两种，服务器服务来自客户的请求。其特点为客户之间不会直接通信，且服务器总是打开且具有固定的周知的 IP

地址。P2P 体系结构则是高度非集中的，每个对等方都兼有客户与服务器的功能，P2P 体系结构的主要特点是自扩展性。

网络应用程序由成对的进程组成，进程与网络之间，即应用层与运输层之间通过套接字连接，不同端系统之间通过网络交换报文实现通信。参与一次通信的两个进程中，发起会话的被称为客户，开始时等待连接的被称为服务器。IP 地址与端口号标识主机与网络应用。应用层协议定义了报文传递的相关内容。

1.2.2 典型应用

客户-服务器体系结构的典型应用和相关协议

Web 使用的是客户-服务器应用程序体系结构。其核心是其应用层使用的**超文本传输协议（HTTP）**，它定义了 Web 页面的交互过程。一个 HTTP 请求报文包括了请求行、首部行和实体体，其中，请求行包含了方法字段、URL 字段、HTTP 版本字段。HTTP 请求报文的方法主要有 GET 和 POST，分别用于请求指定的页面信息和向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。在格式上，GET 把参数包含在 URL 字段中，POST 通过请求的实体体传递参数。在基础的请求报文外，Web 还引入了 Web 缓存来减短响应时长和降低流量，引入了条件 GET 机制判断缓存中内容是否为最新版本。响应报文结构类似，通过状态码表示响应。

报文中的 cookie 首部行和用户端系统，Web 后端数据库共同实现了 cookie 技术来识别用户和进行行为跟踪。HTTP 依据每个请求及其响应是否通过单独的 TCP 连接分为非持续性连接和持续性连接，持续性俩姐能够进一步提高效率。

邮件系统结构上有用户代理和邮件服务器两部分，邮件服务器间使用一种推协议——**简单邮件传输协议（SMTP）**进行通信，SMTP 服务器内部以队列形式存储报文，报文包括环境首部信息和报文体。用户通过本地运行的用户代理程序获取服务器上的报文，遵循 POP3，IMAP 等邮件访问协议。

域名系统（DNS）是因特网上作为域名和 IP 地址（也包括主机别名，邮件服务器别名）相互映射的一个分布式，层次数据库，用户使用主机名通过 DNS 进行域名解析获取对应的 IP 地址，实现对互联网更便捷的访问。客户通过本地 DNS 服务器以递归或迭代的方式逐层向根 DNS 服务器，顶级域 DNS 服务器和权威 DNS 服务器查询最终获得资源记录，DNS 缓存使得查询过程能够绕过某些层次的服务器而改善时延性能。域名具有唯一性，需要通过向注册登记机构注册获得。

P2P 文件分发

比特流（BitTorrent）是 P2P 文件分发的一种常用协议。它采用高效的软件分发系统和点对点技术共享大体积文件。比特流工作方式为分配器或文件的持有者将文件发送给其中一名用户，再由这名用户转发给其它用户，用户之间相互转发自己所拥有的文件部分，直到每个用户的下载都全部完成。由于 P2P 结构的自扩展特性，其文件分发的最小时间总是小于客户-服务器体系结构且随对等方数量增长而增长缓慢，不会超过 1 小时。

视频流和内容分发网

HTTP 中将视频作为普通文件传输的策略因只有一种编码机制不考虑带宽存在客户端延迟等缺陷，因此产生了经 HTTP 的动态适应流（DASH）协议，视频被分块并采用速率不同的多种编码方式存储在不同

的地方，用户依据带宽量运行速率决定算法来动态选择不同版本的块。视频公司利用内容分发网（CDN）向用户发送视频等内容，CDN 有深入和邀请做客两种服务器安置原则，深入通过将服务器深入到 ISP 的接入网中，靠近端用户从而改善用户感受的时延和吞吐量；邀请做客通过在少量关键点建造大集群来实现较低的维护和管理开销。

1.3 运输层

运输层为运行在不同主机上的应用进程提供逻辑通信功能，运输层的发送端将应用层的报文转分组为报文段传递给网络层发送至目的地，接收端执行相反的流程。运输层协议有**传输控制协议（TCP）**与**用户数据报协议（UDP）**，能够为进程提供可靠数据传输，吞吐量，定时和安全性等服务。不同的应用会根据两种运输层协议的特点和缺点来选择相应的方案。表 2展示了常见应用所使用的运输层协议。

应用	应用层协议	下面的运输层协议
电子邮件	SMTP	TCP
Web	HTTP	TCP
文件传输	FTP	TCP
流式多媒体	通常专用	UDP 或 TCP
网络管理	SNMP	通常 UDP
名字转换	DNS	通常 UDP

表 2: 流行的因特网应用及其下面的运输协议

1.3.1 传输控制协议（TCP）

TCP 特征与报文结构

TCP 模型是点对点的，提供面向连接服务（开始前握手，初始化状态变量，结束后拆除连接）和可靠数据传输服务（无差错，按照适当顺序），拥塞控制机制等。TCP 报文包括首部字段与数据字段，具体内容如图 2所示。其中序号字段用于实现可靠传输服务。基于报文段结构实现三次握手机制。其中，三次握手中有两个特殊的报文段，即 **SYN 报文段：同步序列编号报文段**和 **ACK 报文段：确认报文段**。

1. 客户端发送请求到服务器，服务器知道客户端发送，自己接收正常。`SYN=1, seq = x`。
2. 服务器发给客户端,客户端知道自己发送、接收正常,服务器接收、发送正常。`ACK=1,ack=x+1,SYN=1,seq=y`。
3. 客户端发给服务器:服务器知道客户端发送,接收正常,自己接收,发送也正常。`seq=x+1,ACK=1,ack=y+1`。

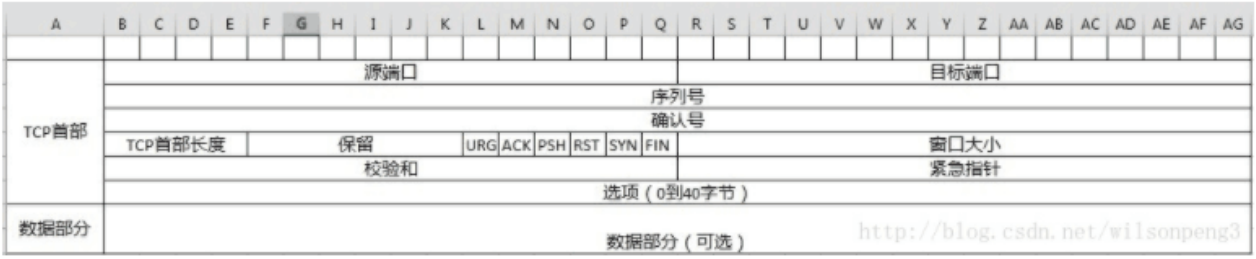


图 2: TCP 报文段结构

三次握手的 TCP 连接建立决定了 TCP 状态如图 3。

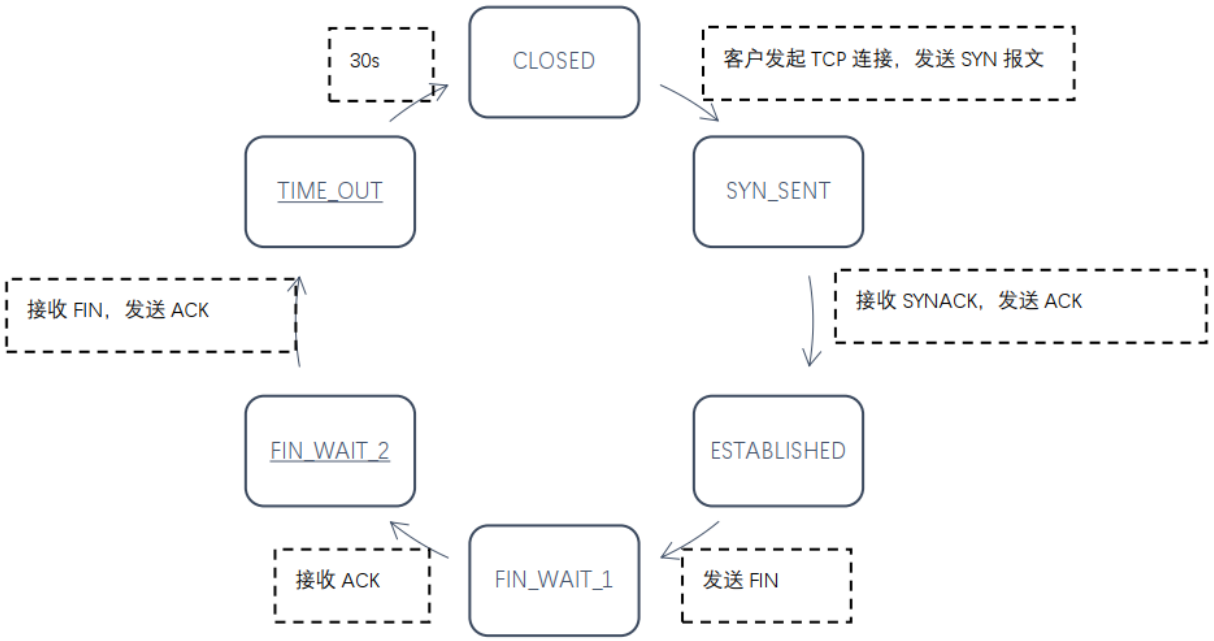


图 3: TCP 状态

SYN 洪泛攻击

SYN 洪泛攻击是一种利用 TCP 的三次握手机制的 Dos 攻击，攻击端发送大量伪造的 IP 地址向被攻击端发出请求使得被攻击端因为发出的响应报文永远发送不到目的地而无法关闭连接从而导致了主机资源耗尽。可以通过降低超时时间使得主机更快的释放半连接的占用或采用 SYN cookie 设置，即如果短时间内连续收到某个 IP 的重复 SYN 请求，则认为受到了该 IP 的攻击，丢弃来自该 IP 的后续请求报文等方式来抵御 SYN 攻击。SYN cookie 设置后的 TCP 状态更新为图 4。

可靠数据传输

可靠数据传输问题在应用层，运输层，链路层都会出现。应用层的可靠数据传输可简化为在较低层不可靠点对点通信的基础上通过**可靠数据传输协议（rdt）**实现为上层提供可靠的信道传输的抽象服务。rdt1.0 在假设底层信道完全可信的基础上，通过独立的发送方和接收方有限状态机定义状态与操作方式。rdt2.0 引入了差错检测，接收方反馈（肯定与否定确认）和重传机制实现了一个停等协议。rdt2.1 添加了分组序号，接收端用冗余 ACK 来表示失序情况。rdt2.2 在 ACK 中显式的加入分组序号代替 NAK 的功能。rdt3.0 加入了倒数定时器处理丢包问题，实现了较为完善的可靠数据传输协议。流水线技术改善了停等协议造成的发送方低利用率问题，使用**回退 N 步（GBN）**和**选择重传（SR）**进行差错恢复。GBN 协议发送端与接收端均存在长度为 N 的滑动窗口用于存放不同状态的封包，分别使用累计确认和丢弃所有失序分组的原则处理分组。SR 发送端仅对未收到的分组进行重传，接收端缓存失序分组至所有丢失分组都被接收后再向上层传递。

TCP 在 IP 的不可靠服务之上建立可靠的数据传输。发送方从上层应用程序接收数据封装后交给 IP 并启动定时器，采用累计确认处理 ACK，超时或者收到 3 个冗余 ACK 时会进行重传，并且超时间隔在

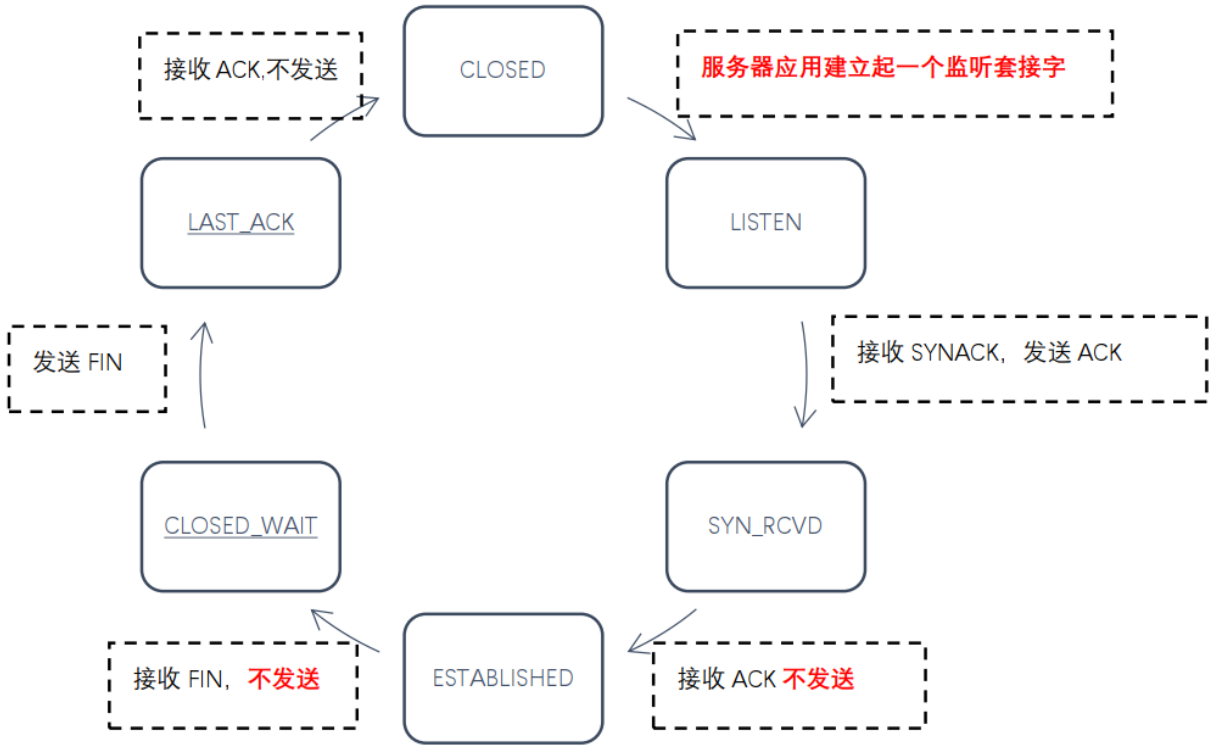


图 4: SYN cookie 设置后的 TCP 状态

每次重传后呈指数增长。使用选择确认与选择重传结合的差错恢复机制，接收方可以有选择的确认失序报文段。

往返时间与超时

TCP 通过**往返时延（RTT）**来判断报文的超时。TCP 为传输一次的报文段测量样本 RTT 并通过指数加权移动平均维护样本 RTT 的均值来估算发送方与接收方的往返时间。超时间隔设置为样本 RTT 均值加一个与样本波动情况成正比的余量。这些参数随着报文段更新而更新，计算公式中的参数通常采用经验值。

样本 RTT 通过采样 RTT 和原来的样本 RTT 加权平均来更新。在等式 (1) 中，依经验取 $\alpha = \frac{1}{8}$。

$$EstimatedRTT = (1-\alpha) \cdot EstimatedRTT + \alpha \cdot SampleRTT \tag{1}$$

样本 RTT 表示了预估下一次的 RTT。如果下一次的 RTT 超出样本 RTT 太多，则认为超时丢包。这个阈值通常用 RTT 偏差给定。在等式 (2) 中，依经验取 $\beta = \frac{1}{4}$。

$$\begin{aligned} DevRTT &= (1-\beta)DevRTT + \beta(SampledRTT - EstimatedRTT) \\ TimeoutInterval &= EstimatedRTT + 4 \cdot DevRTT \end{aligned} \tag{2}$$

拥塞控制

拥塞是指在分组交换网络中传送分组的数目太多时，由于存储转发节点的资源有限而造成网络传输性能下降的情况。在不同的路由资源情况下，拥塞会导致巨大的排队时延，重传补偿缓存溢出而导致分组丢失，冗余内容占用带宽和分组被丢弃导致的传输容量浪费等代价。依据网络层是否为运输层拥塞控制提供显示帮助将拥塞控制分为端到端拥塞控制和网路辅助拥塞控制。

TCP 的拥塞控制是一种端到端的拥塞控制，基于数据传输单向性，接收方缓存空间足够大，发送**拥塞窗口（congestion window ,cwnd)** 的大小, 由网络的拥塞程度来决定和以 TCP 报文段的个数为讨论问题的单位的假设。当不拥塞时，传输速率加性增，出现丢包时，传输速率就乘性减。有慢启动，拥塞避免，快速恢复 3 种拥塞控制算法。其中慢启动与拥塞避免依据发送端的 cwnd 和**慢启动阈值（ssthresh）**变量的相对大小来交替使用。

- **慢启动** 初始时将 cwnd 设为小值（1 个最大报文段长度 (MSS)）。每一个 RTT 速率翻一倍。如果出现丢包，则再次将 cwnd 设为 1 MSS。将 ssthresh 设置为 cwnd/2, 再次慢启动，直至达到阈值，转变到拥塞避免算法。
- **拥塞避免** 每次 RTT 增加 1MSS。若出现超时，则吧 cwnd 设置为 1MSS；若出现丢包，则更新 ssthresh 为 cwnd 的一般；若出现 3 个冗余 ACK，则认为发生了轻微超时，设置 cwnd 为原来的一半，进入快速恢复状态。
- **快速恢复** 对于引起快速恢复的报文段，每个冗余 ACK 都将 cwnd 增加 1MSS。直至引起快速恢复状态的报文的 ACK 收到，再次转换状态为拥塞避免。

1.3.2 用户数据报协议（UDP）

UDP 是一种提供最小服务的轻量级服务，无连接（通信双方的运输实体不执行握手），不保证安全性和有序性。这也使得其具有不存在连接建立的时延，不需要维护拥塞控制参数等连接状态，分组首部开销小等优点。基于这些优点，UDP 被广泛运用于多媒体传输等应用中。

UDP 报文段由首部四个字段和应用数据组成，端口号用于执行分解功能，长度字段指示了报文段中的字节数，检验和用于和接收到的报文段中 16 比特字的和的反码比较是否相等来判断是否出现了差错。

1.3.3 多路复用与多路分解

运输层的多路复用与多路分解就是将主机交付扩展到进程交付。多路分解的工作方式可以概括为主机上的每个套接字被分配一个端口号，当报文到达主机时，运输层检查报文段中的目的端口号，并将其定向到相应的套接字。多路复用就是从源主机的不同套接字中收集数据块，并为每个数据块封装上首部信息从而生成报文段，然后将报文段传递到网络层中去。无连接和有连接的多路复用与多路分解分别适应于 UDP 与 TCP 的套接字仅用端口号和用四元组标识的机制。(我觉得可以配图)

1.4 网络层

网络层的目的是实现两个端系统之间的数据透明传送，有转发和路由选择等重要功能。路由器是实现数据中转的典型设备，每台路由器的功能就是网络层的数据平面的功能，主要控制路由器内部从输入端到

输出端的数据报转发。控制平面主要控制数据报沿着源和目的地主机之间的路由器路径最终实现端到端的传输，也就是路由选择，控制平面通过**软件定义网络（SDN）**远程配置转发表。因特网的网络层提供了单一的尽力而为服务，即对任何一项服务都不能做出保障。

1.4.1 路由

路由器结构功能

路由器的体系结构如图 5所示。分组经过物理层与链路层处理后网络层的队列中排队等待处理，输入端口通过最长前缀匹配规则在转发表中查找该分组对应的输出端口，实现基于目的地的转发，由于交换线路的阻塞，输入端口的报文可能会排队。交换结构是路由器的核心，常用的交换方式有经内存交换（一次只能执行一个内存读/写），经总线交换（带宽受总线速率的限制），经互联网的交换（非阻塞的，纵横式网络并行转发多个分组）。输出端口取出已经在其内存中的分组并将其发送到输出链路上，当内存不足时，通过弃尾等主动对列管理策略来腾出空间。输出端口使用先进先出，优先权排队，循环和加权公平排队等分组调度方式来确定分组的转运顺序。

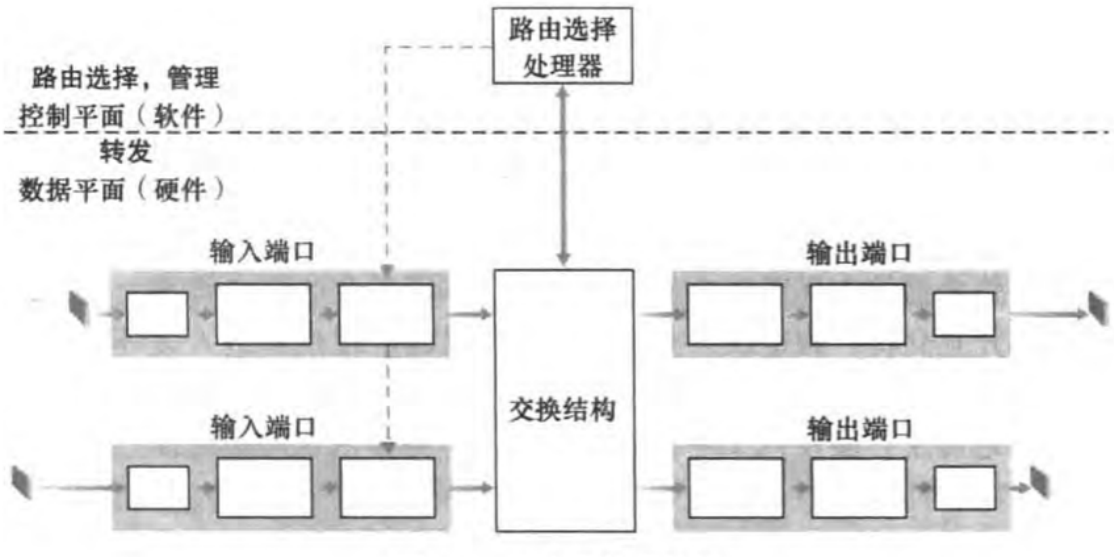


图 5: 路由器体系结构

1.4.2 路由器算法

路由选择算法用于确定一条从发送方到接收方的过程中通路由器的好的路径，建立在抽象图的模型上。分为集中式路由选择算法和分散式路由选择算法。集中式路由选择算法要求在算法开始之前获取所有节点的连通性和链路的开销的完整信息，常被称为**链路状态（LS）算法**。分散式路由选择算法每个节点仅拥有与其直接相连的链路的开销，**距离向量算法（DV）**是其中的一种。

Dijsktra

Dijkstra 算法维护一个当前已知最低开销路径的集合，每次迭代选择未加入该集合中具有最低开销的节点加入并更新其它未加入节点的最低开销直至所有节点都属于该集合。某个节点被加入集合前最后一次

对它进行更新的节点就是源点到其的最短路径上的前一个节点。其时间复杂度为 $O(n^2)$, 需要通过每台路由器发送链路通告的时间随机化来避免振荡。LS 算法如算法 1的核心就是 Dijsktra。

Algorithm 1 源节点 u 的链路状态 (LS) 算法

Input: $G = (V, E)$
Output: Broadcast link state
1: **Initialization**
2: $V' \leftarrow \{u\}$
3: **for all** $v \in V$ **do**
4: **if** v is a neighbor of u **then**
5: $D(v) \leftarrow c(u, v)$
6: **else**
7: $D(v) \leftarrow \infty$
8: **end if**
9: **end for**
10: **repeat**
11: Find w not in V' such that $D(w)$ is a minimum
12: Add w to V'
13: Update $D(v)$ for each neigbor of w and not in V'
14: **until** $V' = V$

Bellman-Ford

Bellman-Ford 算法对于每个节点维护一个到所有其他节点的距离向量，异步的向它所有的邻居发送它保存的距离向量的副本，并且使用它从邻居处接受到的新的距离向量更新自己的距离向量，获取该点到其他点的更短的距离。这种方法已被证明是可以收敛到实际最短距离的数值的。距离向量算法存在收敛速度慢，可能遇到路由选择回路而导致无穷计数等缺点，可以使用水平分裂，增加毒性逆转等方法缓解。DV 算法 (如算法 2) 就运用到了 Bellman-Ford。

拓扑相关的路由算法

除了以上两种经典算法，还有一些常用的其他路由算法。分层算法通过将路由器分组要求组内彼此信息可知从而抽象为一个点来缓解网络规模扩大带来的资源需求量上升问题。广播路由采用多目的地路由，生成树算法和逆向路径传输等算法向所有节点发送报文，可能需要较大的发送量或出现重复问题。多址路由选择通过每个点建立生成树或一组一颗生成树，使用 IP 的 D 类地址，每次传输仅沿相应的生成树转发。

1.4.3 网络层协议–IP

IP 协议是网络层目前使用的著名协议，有 4 和 6 两个版本，两者功能几乎一致，均标识对应的通信流类别。其数据报格式如图 6所示，IPv6 的重要变化也体现在数据报格式中：其将 IP 地址由 32 比特扩展为 128 比特，且引入了任播地址；Ipv6 还简化了首部，删去的字段都以可选扩展头部的形式出现在报文里；预见性的添加了流标签要求发送方对某些流特殊处理。

IP 协议提供一种统一的地址格式–IP 地址，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。IP 地址由网络号和主机号组成，网络号由权威组织 TCANN 在全球范围统一分配，主机号由本地管理员手动或采取动态主机配置协议分配。采用点分十进制表示，依据网络

Algorithm 2 距离向量 (DV) 算法

Input: $G = (V, E)$

Initialization
for all destination $y \in V$ **do**
  if y is not a neighbor **then**
    $D_x(y) = \infty$
  else
    $D_x(y) = c(x, y)$
  end if
end for
for all neigbor w **do**
  $D_w(y) = c(w, y)$ for all destinations $y \in V$
end for
for all neighbor w **do**
  send distance vector $\mathbf{D_x} = \{D_x(y) : y \in V\}$ to w
end for
loop
  wait
  if A link cost changes or receive a distance vectore **then**
    for all $y \in V$ **do**
      $D_x(y) = \min_v\{c(x, v) + D_v(y)\}$
    end for
  end if
  if $D_x(y)$ changed for any destination y **then**
    send distance vector $\mathbf{D_x} = [D_x(y) : y \in V]$ to all neighbors
  end if
end loop

规模由大到小，分为 A,B,C,D,E 五大类，其中 D 类地址用于多播，E 类地址作为保留地址用于实验。为了优化网络性能，减少流量，网络被划分为不同子网，引入与 IP 地址格式相同的子网掩码来确定 IP 地址中的网络号与主机号长度，子网掩码与网络号对应的所有位都是 1，与主机号对应的位都是 0，也可以通过在十进制 IP 地址后面加”/x”表示网络号占 x 位。

无类别域际路由选择 (CIDR) 和**网络地址转换 (NAT)** 是在 IPv4 协议基础上提出的扩展算法。CIDR 目的在于解决地址耗尽，将好几个 IP 网络结合在一起，对原有的类别路由选择进程进行了重新构建，用前缀取代了原有结构对地址网络部分的限制，减少由核心路由器运载的路由选择信息的数量。NAT 是一种将私有地址转化为合法的 IP 地址的技术，广泛应用于各种类型的网络中，能够解决 IP 地址不足，隐藏并保护网络内部的计算机，从而有效避免来自网络外部的攻击。

IPv6 协议的优点从数据报中就能够体现出来，但由于现在网络规模巨大，通过标志日迁移实现改变网络层协议是不可行的，广泛采用的方法是通过建隧道进行渗透，隧道结构如图 7所示。

隧道技术包含了以下步骤。

1. 隧道入口节点 (封装路由器) 创立一个用于封装的 IPv4 报文头，并传送此被封装的分组。
2. 隧道出口节点 (解封装路由器) 接收此被封装的分组，如果需要重新组装此分组，移去 IPv4 报文头，并处理收到的 IPv6 分组。
3. 封装路由器或许需要为每个隧道记录维持软状态信息，这类参数诸如隧道 MTU，以便处理转发的 IPv6 分组进隧道。

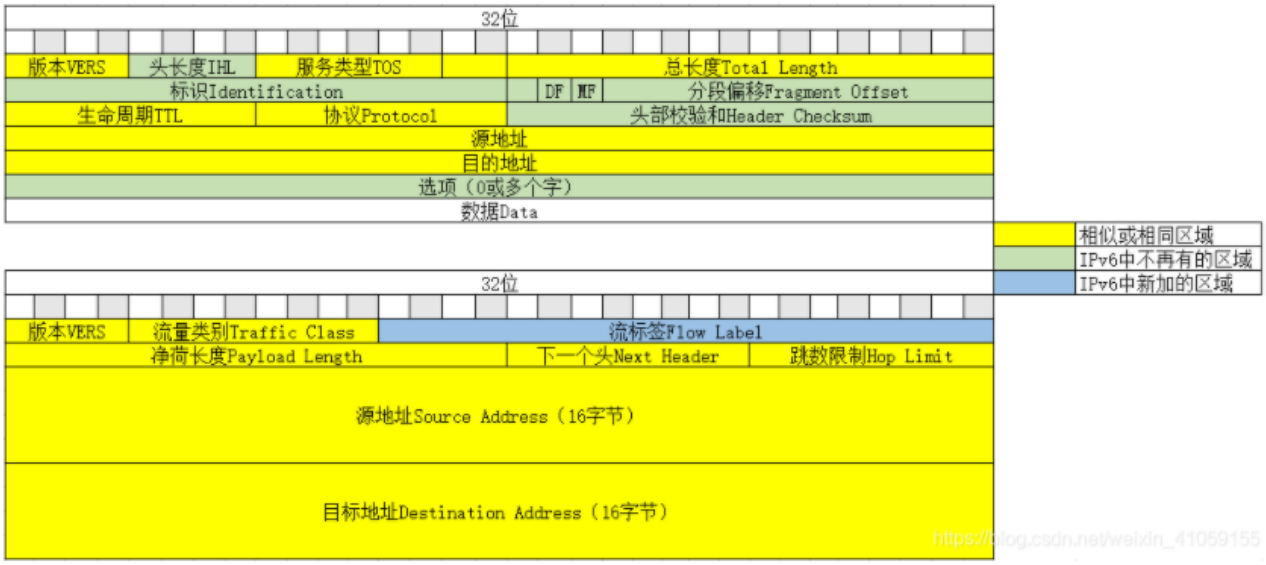


图 6: IPv4 与 IPv6 数据报格式对比 [2]

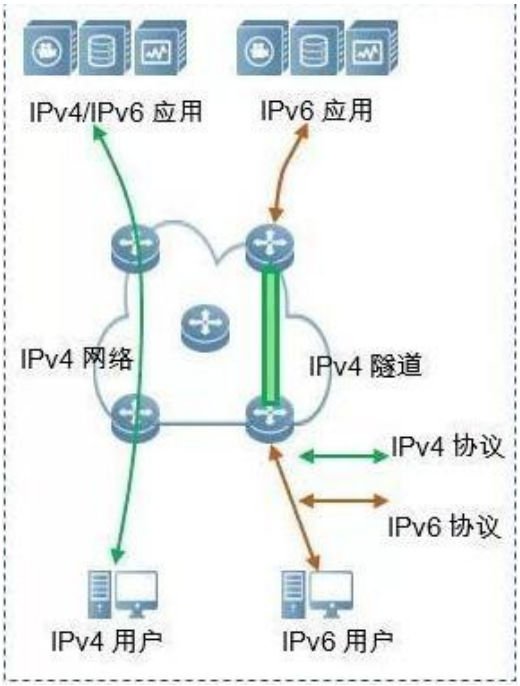


图 7: 隧道技术示意图

OSPF

通过将路由器组织进自治系统可以解决网络中路由器数目过大问题并实现组织的管理自治。**开放最短路优先（OSPF）**是一种自治系统内部的路由选择协议，它使用了泛洪链路状态信息和 Dijsktra 算法，报文由 IP 承载，自己实现可靠报文传输，链路状态广播等功能。OSPF 通过认证，序号等保障路由器之间交换的安全性。可以实现单个自治系统内的区域划分，综合支持单播与多播路由选择，并使自身的开销可控。缺点是其配置相对复杂。

BGP

边界网关协议（BGP）是所有 AS 系统间通信需要运行的 ISP 间路由选择协议。由于 BGP 中的分组路由的目的地是一个 CIDR 化的前缀（表示一个子网或一个子网的集合），因此其实现手段为从邻居 AS 获得前缀的可达性信息，然后确定到达该前缀的最好路由。获取前缀可达信息的过程是通过网关路由器在 AS 之间传递 BGP 报文告知某个子网的存在和到达路径，同一个 AS 内部则通过 iBGP 连接来传播可达性信息。确定最好的路由的算法有热土豆路由选择和路由器选择算法。热土豆路由选择的主要思想史尽可能快的将分组送出 AS 而不考虑余下部分的开销。路由器选择算法则是 AS 内部由管理员决定，AS 间以 AS 跳数作为距离测度使用 DV 算法决定路径。在遇到同样性能的路径时，使用热土豆路由选择，仍有剩余则使用 BGP 标识符来选择路由。

BGP 也可以用来实现 IP 任播，为物理位置不同的服务器指派相同的地址，让用户从距离最近的服务器获取内容。BGP 用于挑选到用户目标 IP 地址最好的路由。

ICMP

因特网控制报文协议（ICMP）是 TCP/IP 协议族的一个子协议，用于在 IP 主机、路由器之间传递控制消息（指网络通不通、主机是否可达、路由是否可用等网络本身的消息）。其报文包括类型字段和编码字段，包含引起该报文首词生成 IP 数据报首部和前 8 个字节。Traceroute 程序就是通过 ICMP 报文实现的，源主机在发送具有不可达端口号 UDP 后启动相应计时器，通过 ICMP 报文返回源主机的时间确定往返时延，路径当中的路由器数量和标识。

1.4.4 SDN

软件定义网络（SDN）将控制平面与数据平面分离，采用集中化的网络控制器来实现控制平面的功能，通过南向接口把算出来的流表，发给每个设备中的控制代理，这个代理把流表装载在分组交换机，之后根据这个流表作出相应的动作。数据平面交换机按照流表（有控制平面设置的控制逻辑）进行转发等动作。相比于传统方式，SDN 实现由垂直集成到水平集成。SDN 体系结构特征包括了

- 基于流的转发
- 数据平台与控制平台分离
- 网络控制功能
- 可编程网络

OpenFlow 协议

OpenFlow 协议定义了交换机在报文匹配失败时向控制器申请流表的方法，当交换机收到一个不能被当前流表各条流匹配的数据包时，通过将失配报文的相关信息封装在 Packet-In 消息中发送给控制器，让控制器知晓报文失配情况，由控制器通过 Flow-Mod 等消息向交换机安装新流表。

通用转发

通用转发过程上使用一张匹配加动作表（OpenFlow 中的流表）将基于目的的转发表一般化了。OpenFlow 的匹配抽象允许对来自三个层次的协议首部所选择的字段进行匹配。流表项中的动作列表按规定次序执行决定了应用于与流表项匹配的分组的处理。在匹配与动作中操作中 OpenFlow 有简单转发，负载均衡，充当防火墙等实例。

SDN 控制器

SDN 控制器和 SDN 网络控制应用程序构成了 SDN 控制平面。SDN 控制器工能上自底向上可划分为通信层，网络范围管理层，对于网络控制应用程序层的接口。分别负责传送控制器与交换机，主机等受控设备间的信息，维护各种受控设备最新状态的流表，与网络控制应用程序交互。每层的功能如下。

- **通信层** SDN 控制器和受控网络设备之间的通信。
- **网络范围管理层**控制器获得相关主机或交换机的状态
- **对于网络控制应用程序层的接口**

1.4.5 网络管理

网络管理框架中有如图 8所示的关键组件。简单网络管理协议（SNMP）是一个应用层协议，用于在管理服务器和代表管理服务器执行的代理之间传递网络管理控制和信息报文。依据目的不同有不同类型的报文格式。该协议经历了三个版本的演变逐步完善了报文功能，增加了错误码和安全保护机制等。

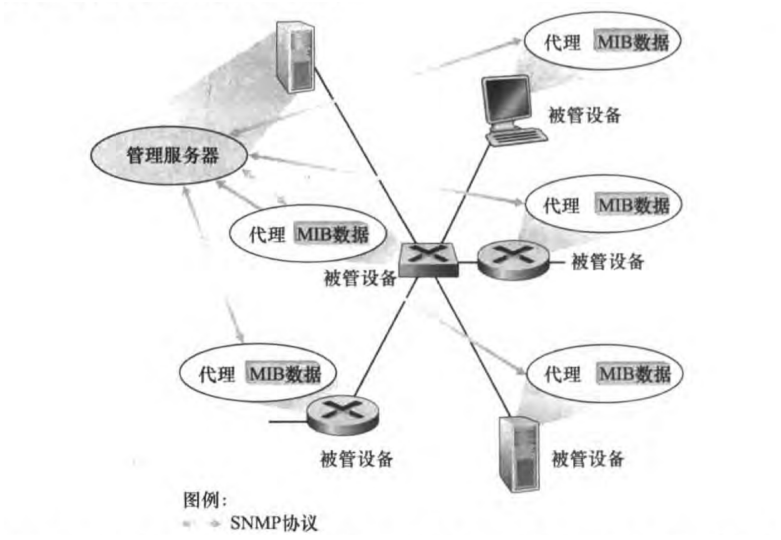


图 8: 网络管理的组件

1.5 链路层

链路层是协议栈中软件与硬件交接的地方，其主体部分是网络接口卡（NIC）中的链路层控制器，实现包括成帧，链路接入，可靠交付，差错检验和纠正在内多种链路层服务；组装链路层寻址信息和几号激活控制器硬件等高层链路层功能在 CPU 上的软件中实现。链路层最基本的服务是将源自网络层来的数据可靠地传输到相邻节点的目标机网络层，链路层的结构，协议等都是为了实现该功能而设计。

1.5.1 差错检验和纠正技术

比特级差错检验和纠正技术用于保护网络层传递下来的数据报和链路帧首部的链路级寻址信息，序号等字段。常见的技术包括奇偶校验，检验和和循环冗余检测等。

奇偶校验是差错检测中最简单的方式。单个奇偶校验位可以通过判断接收方比特位为 1 的个数奇偶性是否与发射方相同来检测出奇数个比特差错，二维的奇偶校验则可以通过存在奇偶校验差错的行和列的索引来实现两位检错和一位纠错。

因特网检验和将数据的字节作为 16 比特整数处理并求和取反码携带在报文首部，接收方通过判断收到的数据和与该值求和再取反码的值是否全为 1 来判断数据是否出现了差错。

循环冗余检测（CRC）技术将发送的比特串看做系数为 0 或 1 的多项式，进行多项式计算，并将得到的结果附在帧的后面，接收设备也执行类似的算法，以保证数据传输的正确性和完整性。

1.5.2 MAC 协议

多路接入控制协议（MAC）用于解决广播链路中协调多个发送和接收节点对一个共享广播信道的访问。有信道化分，随机接入和轮流三种类型。

信道化分协议的技术有在时间划分为时间帧进一步划分的时隙内传输分组比特的时分多路复用，将信道划分为带宽相等的不同频段公平分配给 N 个节点的频分多路复用和为每个节点分配不同的编码的码分多址技术。

随机接入协议中一个节点总是以信道的全部速率进行发送，经历碰撞的节点独立的选择随机延时重复发送帧直至无碰撞通过。纯 ALOHA 协议操作简单，接收到帧立即传输，发生碰撞立即以概率 p 重传，因其完全分散而导致最大传输效率仅为 1/(2e)。时隙 ALOHA 协议在此基础上将时间分块，当在某个时隙结束之前检测到碰撞，在后续每个时隙终以概率 p 重传，最高传输效率提高了 1 倍，但其成功吞吐率仍然不高。载波侦听多路访问（CSMA）通过载波侦听和碰撞检测（CD）来减少传输过程中发生的碰撞，在适配器侦听到信道空闲后才开始传输帧，如果传输过程发现其他适配器的信号能量，就立即停止并等待一个随机时延再继续传输，CSMA/CD 的传输效率可以用近似公式 1/(1+5tprop/ttrans) 来计算。

轮流协议主要有有轮询协议和令牌传递协议。轮询协议规定主节点以循环的方式通知每个节点其最大传输帧的数量，消除了碰撞和空时隙但引入了轮询时延。令牌传递协议不规定主节点，而是在所有节点之间以某种固定次序交换一个成为令牌的小的特殊节点，某个节点持有令牌时可发送其最大数目的帧并将令牌转发给下一节点。

1.5.3 交换局域网

链路层寻址和 ARP

链路层的交换局域网使用链路层地址来转发链路层帧通过交换机网络。链路层地址（MAC 地址）属于主机或路由器的网络接口，MAC 地址长度是 48 比特，由 16 进制的数字组成，前 24 位组织唯一标志符由 IEEE 的注册管理机构给不同厂家分配，后 24 位扩展标识符由厂家自己分配的，同一个厂家生产的网卡中 MAC 地址后 24 位是不同的。与 IP 地址不同，MAC 地址具有扁平结构且不随主机移动而改变。地址解析协议（ARP）用于转换 IP 地址与 MAC 地址，通过保存在主机或路由器内存中的 ARP 表进行地址间的映射，表项随时间而更新。在跨越子网的数据报传递过程中，数据报需通过不同子网间路径上和子网入口的路由器传递，通过 ARP 获取每一步目的地的 MAC 地址。

以太网

以太网是目前最流行的局域网技术，其中心结构由开始的集线器发展为交换机，其传送的帧包含承载 IP 数据报的数据字段，目的和源地址，CRC 和用于唤醒接收适配器的前同步码。为网络层提供不可靠服务。

链路层交换机

链路层交换机负责接收入链路层帧并将它们转发到出链路，借助交换机表完成对帧的过滤和转发功能，交换机表通过自学习动态的建立交换机表。交换机具有能够消除碰撞，将不同的链路彼此隔离和易于进行网络管理的性质。与同样进行转发分组的路由器相比，交换机是即插即用的，处理网络第二层的帧，将活跃拓扑限制为一棵生成树，对广播风暴不提供任何保护措施；而路由器不是即插即用的，处理的是第三层的数据报，没有生成树限制，所以可以以丰富的拓扑结构建立因特网，且能够对第二层广播风暴提供防火墙保护。基于以上特点，交换机常被应用在小型网络，几千台主机形成的大型网络则还需要包括路由器。

虚拟化

虚拟局域网（VLAN）是一组逻辑上的设备和用户，这些设备和用户并不受物理位置的限制，可以根据功能、部门及应用等因素将它们组织起来，相互之间的通信就好像它们在同一个网段中一样。可以用于解决交换机的无效使用，缺乏流量隔离，和雇员组间移动困难的缺点。网络管理员将交换机端口分组，一个端口的广播流量仅能到达组内的其他端口。不同组间通过一个共享相同的物理交换机进行流量交换。除了上述分组隔离模式，也可以通过配置每台交换机的特殊端口结合帧首部的 VLAN 标签实现 VLAN 干线连接形成一种扩展性互联交换机的方法。

多协议标签交换（MPLS）是一种 IP 骨干网技术。MPLS 在无连接的 IP 网络上引入面向连接的标签交换概念，将第三层路由技术和第二层交换技术相结合，充分发挥了 IP 路由的灵活性和二层交换的简捷性，体现链路层的虚拟化。网络层的控制平面负责产生和维护路由信息以及标签信息，数据平面负责普通 IP 报文的转发以及带 MPLS 标签报文的转发。

数据中心网络 (DCN)

数据中心网络（DCN) 是基于数据中心内的流量呈现出典型的交换数据集中、东西流量增多等特征提出的一种分层次的网络结构，通过第一层交换机前接入负载均衡器实现负载均衡。为了降低数据中心的费用，提高其在时延和吞吐量上的性能，数据中心网络呈现扁平化、网络虚拟化以及可以编程和定义的新发展趋势。

1.6 物理层

计算机网络物理层的传输介质常见的有五种，分别是双绞线，同轴电缆，光缆，无线传输和卫星通信。

比特与信号是指数字数据最终都被表示成比特串数字传输的本质就是把比特串变成信号。其中包括数字信号的傅里叶分析，数据传输，Nyquist 定理，Shannon 定理。

多路复用技术以同时携带多路信号来高效率地使用传播介质，进而使传输系统得到更高效的利用，其方式主要有两种，一是频分多路复用 FDM，二是时分多路复用 TDM。

传输模式主要有三种，一是串行传输和并行传输 (计算机网络采用串行传输)，二是基带传输和宽带传输，三是异步传输和同步传输 (在数据通信中，通常采用异步传输和同步传输两种方式)。

物理层的协议体现在物理接口上，包括适用于计算机或终端与 Modem 间的典型的串行接口 RS-232-C，常用的并行口如打印机，和最常用的网络接口 RJ-45。

电话系统是公用电话交换网，简称 PSTN，一般为树状结构，并且由于数字传输的明显优势，所以电话系统呈现数字化的发展趋势。电话系统主要由本地回路、主干线、交换局三部分组成。

Internet 的本地接入主要有拨号接入、ADSL 接入和 Internet over Cable。

1.7 无线网络和移动网络

1.7.1 无线网络

无线连接和网络特征

无线网络的要素包括无线主机，无线通信链路和基站，目前主流的结构结构是基于基础设施接入有线网络 () 如图 9所示，也有目前尚不成熟的自组织模式，即不存在基础设施结构，网络节点自己组织成网络并在其中进行路由。无线网络与有线网络的区别主要在于它信号传输的过程中会由于路径损耗而导致信号强度递减，更容易受到其他源的干扰和可能会由于反射出现多径传播。接收方采用不同的调制技术会获得比特差错率不同的信息，经过传输后的信息信噪比越大，越容易被正确提取。信息传输的双方也有可能因为物理间隔等因素产生隐藏终端问题。无线 LAN 和蜂窝技术当中常用协议有链路层的码分多址（CDMA）协议等。

WIFI

WIFI 体系结构的基本构件模块是**基本服务集（BSS)**，一个 BBS 中有一个作为接入点（AP）的中央基站，它在安装时被赋予了服务集标识符和信道号，它们通过互联设备接入因特网中。WIFI 标准要求每个 AP 周期性的发送信标帧，无线主机通过被动扫描或主动扫描的方式扫描信道，选择可用 AP 进行关联。WIFI 为无线局域网选择了类似于 CSMA/CD 的带碰撞避免的 CSMA，不同的是它并不实现碰撞检

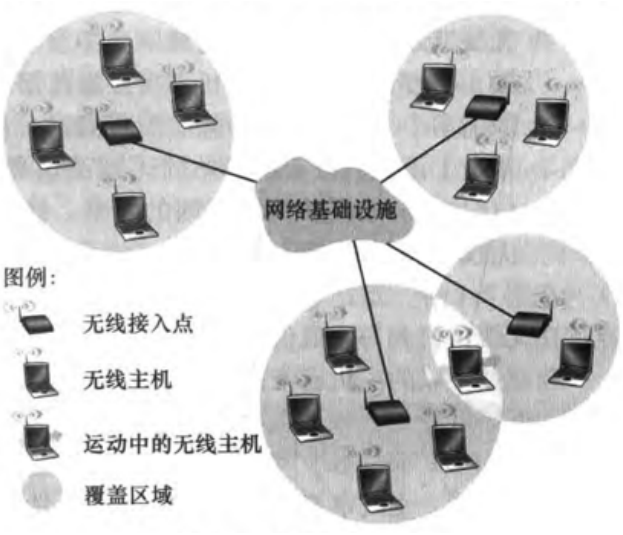


图 9: 无线网络结构

测，数据帧一旦开始发送就不会返回，发送端在未收到确认帧的情况下会进行重传。其他避免碰撞的方法还有如利用短请求发送和短允许发送控制帧预约信道访问等。

蜂窝因特网接入

蜂窝技术由早期主要用于语音通信，到二代语音与电话网连接，，再到三代通过智能手机完整的运行 TCP/IP 协议从而将蜂窝数据网接入因特网，逐步实现了将因特网扩展到蜂窝用户。第四代蜂窝技术实现了一个全 IP 核心网和一个加强的无线电接入网。在速度上有显著的提升。

1.7.2 移动网络

无线网络的移动性有很多方面，移动的网络的体系架构如图 10。用户的移动性对网络透明，可以通过它处于的外部网络像邻居通告移动节点位置或通过移动节点与归属代理之间的协议来更新节点位置。移动节点间接路由选择的方法由移动 IP 标准规定，移动 IP 标准包括代理发现，像归属代理注册和数据报间接路由选择三部分。直接路由选择克服了三角路由选择的低效问题，但需要移动用户定位协议让通信者代理获得移动节点的 COA，也需要解决数据路由选择到新的外部网络的问题。相比于有线网络，无线网络及其移动性对运输层和应用层产生的差异较小，但运输层协议在无线网络上的性能会受到无线信道高比特差错率等性质的影响。

1.8 互联网发展情况

随着技术的不断成熟，网络领域也不断扩展相关标准也在不断更新换代，如方兴未艾的第五代移动通信技术（5G）就是移动通信技术的进一步发展，其具有高速率、低时延和大连接等显著优势，是实现人机物互联的网络基础设施。深圳从 2017 年 10 月开通首个 5G 试验站点以来，5G 产业链发展快速推进。2018 年 6 月 3GPP 发布了第一个 5G 标准 (Release-15)，支持 5G 独立组网，重点满足增强移动宽带业务。2020 年 6 月 Release-16 版本标准发布，重点支持低时延高可靠业务，实现对 5G 车联网、工业互联网等应用的支持。Release-17(R17) 版本标准重点实现差异化物联网应用，实现中高速大连接，在今年三月也已冻结，近日即将转入可执行阶段。5G 作为一种新型移动通信网络，不仅要解决人与人通信，为用

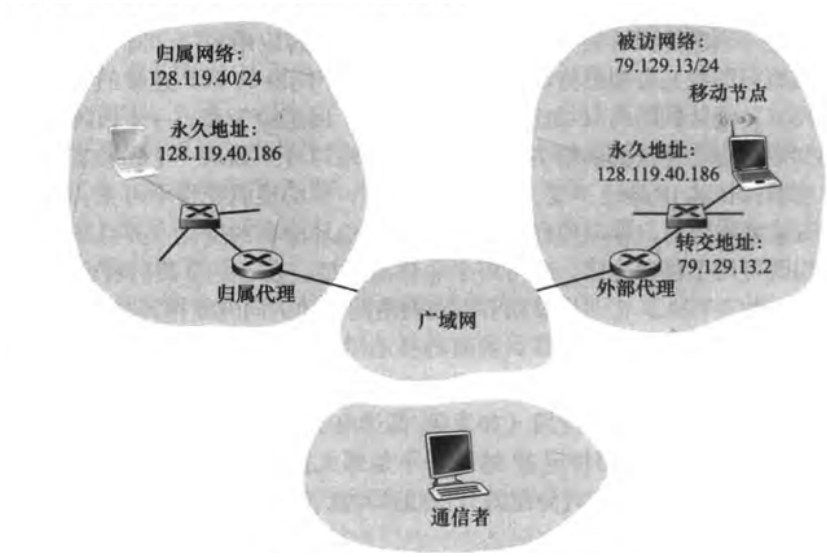


图 10: 移动网络体系结构的初始要素

户提供增强现实、虚拟现实、超高清 (3D) 视频等更加身临其境的极致业务体验，更要解决人与物、物与物通信，也就是物联网问题。因此物联网的核心与基础就是互联网，其用户端延伸和扩展到了任何物品与物品之间，通过连接各种设备创建了一个虚拟网络进行信息交换和通信。物联网一般为无线网，而由于每个人周围的设备可以达到一千至五千个，所以物联网可能要包含 500 兆至一千兆个物体。通过物联网可以用中心计算机对机器、设备、人员进行集中管理、控制，也可以对家庭设备、汽车进行遥控，以及搜索位置、防止物品被盗等，物联网通过智能感知、识别技术与普适计算等通信感知技术，广泛应用于网络的融合中，也因此被称为继计算机、互联网之后世界信息产业发展的第三次浪潮。此外还有大数据，人工智能，区块链，，深度学习等领域的发展都与计算机网络的发展息息相关。[3]

1.9 代理服务器的最新发展

近年来，代理服务器在许多领域都有广泛的应用。陈爱林, 乐全明 [4] 在 2010 年提出了智能变电站和调度主站无缝通信方面代理服务器的应用，为智能电网异构系统间实现无缝连接提供技术支撑。徐浩诚刘, 利军, 黄青松 [5] 在 2016 年提出了将代理服务器中的缓存置换策略应用于医学图像自适应分层切割，结合医学影像的特点减少网络的宽带消耗、减轻服务器的负载。魏宁, 周公建, 钱江 [6] 在 2020 年开发了地铁信号系统中的代理服务器，结合地铁信号系统特征，提高了访问地铁终端的效率。

在技术方面, 当下的代理服务器的主要研究都集中于 Web 代理服务器缓存替换策略。2011 年，张旺俊 [7] 利用更加优化的用户访问特性预估算法，进一步改善网络带宽性能、解决网络堵塞和用户访问延时时间过长等问题。2017 年，赵中全, 刘丹 [8] 利用树扩展朴素贝叶斯分类器对 Web 日志数据进行分类，从而提高缓存命中率和字节命中率，提高代理服务器的效率。

近年来，随着短视频的兴起，如何解决庞大的多媒体流量也是目前网络市场需要攻克的核心，而代理服务器在其中的应用也起着重要的作用。

1.10 小结

通过对计算机网络结构的梳理，总结出网络各层的主要功能如下：

应用层主要解决最终通信双方数据传输问题，即不同结点上两个对应应用进程之间的通信。主要协议有：HTTP ,SMTP,SNMP,DNS,POP3 等。表示层与会话层在五层模型中合并到应用层，分别定义了数据格式以及加密和如何开始、控制和终止一个会话。运输层输层定义传输数据的协议端口号，以及流控和差错校验。主要协议协议有：TCP UDP 等，数据包离开网卡即进入网络运输层。网络层主要进行逻辑地址寻址，实现不同网络之间的路径选择（即路由选择）。主要协议有：ICMP IGMP IP（IPV4 IPV6）ARP 等。数据链路层主要建立逻辑连接、进行硬件地址寻址、差错校验等功能。物理层规定数据传输格式、电气意义、编码方式、复用方式、信道估计、同步方式等，设备如集线器等定义在物理层。

计算机网络的分层模型简化了体系设计上的复杂性，也为我们研究其结构功能提供了相对清晰的思路。学习计算机网络体系结构对我们今后学习和使用网络都提供了很大的帮助。

2 Web 缓存代理服务器实验

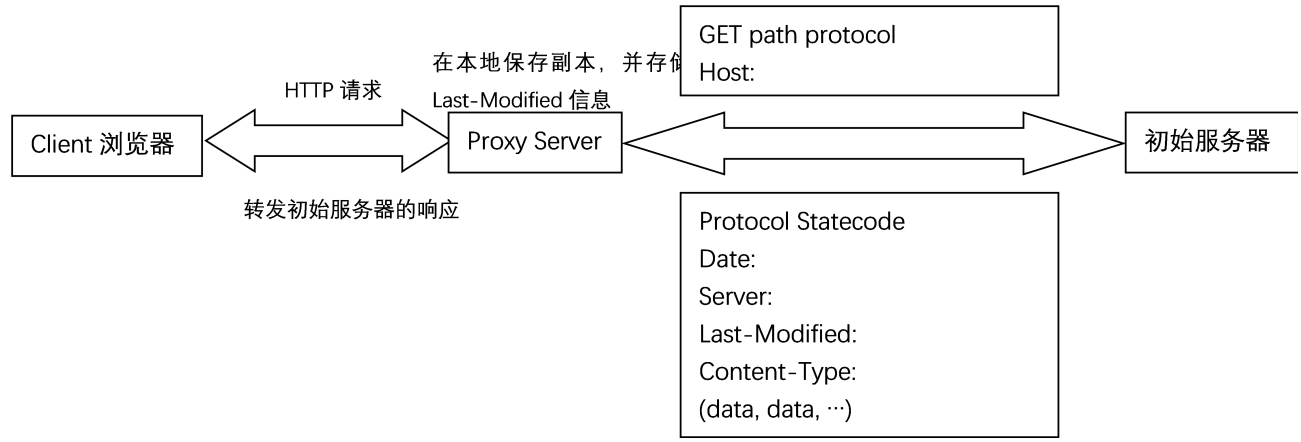
2.1 Web 缓存代理服务器

Web 缓存器 (Web cache) 也叫代理服务器 (proxy server)，能够代替初始服务器服务客户主机的 HTTP 请求。代理服务器有自己的本地磁盘，能够存储最近请求过的文件副本。如果将代理服务器配置给客户主机，当客户主机请求一个网页时，将会发生

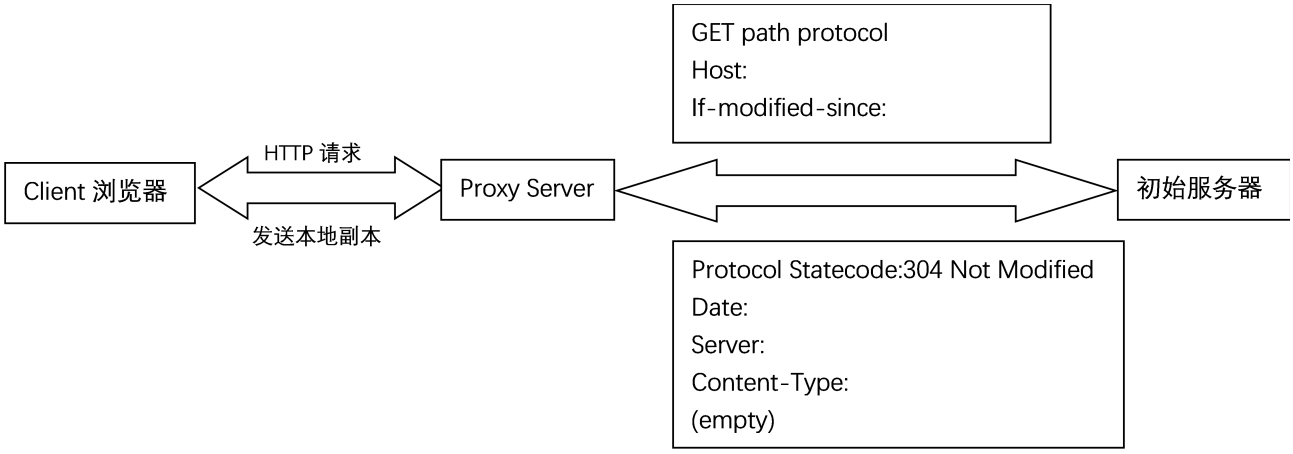
- 浏览器创建一个到代理服务器的 TCP 连接，并发送 HTTP 请求。
- 代理服务器检查本地是否有副本。
- 若有本地副本
  - 代理服务器向初始服务器发送带修改时间的请求报文确认副本是否过期（内容是否有更改）。
  - 如果没有过期，则初始服务器返回一个空内容的响应，代理服务器向客户浏览器发送一个本地副本。
  - 如果过期，那么初始服务器返回更新的内容，代理服务器向客户浏览器发送更新版本，并更新本地副本。
- 若没有本地副本
  - 代理服务器向初始服务器发送请求报文。
  - 初始服务器向代理服务器返回一个响应报文，代理服务器向客户浏览器发送报文，并保存到本地副本。

其中，在检查副本更新中，代理服务器需要修改请求报文，即**条件 GET** 方法。在条件 GET 中，当一个 HTTP 请求发生，客户浏览器，代理服务器，初始服务器之间的交互如图 11。

可以看到，当客户浏览器通过代理服务器发送 HTTP 请求时，客户浏览器的地址，套接字端口等信息不会暴露给初始服务器，因此代理服务器能够充当防火墙的作用。除了基础的请求响应和缓存功能，一些代理服务器还具备内容过滤、访问控制管理的功能。



(a) 代理服务器没有本地副本



(b) 代理服务器有本地副本——以副本未过期为例

图 11: 客户浏览器，代理服务器，初始服务器之间的交互

2.2 项目实现及功能介绍

2.2.1 缓存器

为了展示 Web 缓存器会把旧的副本清理的特性，以及 Web 文件以一份缓存 (buffer) 为单位接收的特性，本项目使用一个类`Cache`来模拟一个代理服务器的本地磁盘。

2.2.2 代理服务器

本项目的代理服务器支持多种协议包括`HTTP1.0`, `HTTP1.1`, `HTTP2.0`, `HTTPS`（即加密 ssl）。通过本代理可以，本地浏览器可以快速访问各种网站。

2.3 技术细节

2.3.1 多线程代理服务

根据课本 [1] 套接字编程所给出的参考代码，python 套接字编程中监听的参数就可以理解为多线程的线程数。事实上，监听数等于服务器拒绝 (超过限制数量的) 连接之前，操作系统可以挂起的最大连接数量，相当于排队的数量，并不等于多线程的线程数量。因此，要实现真正的多线程代理服务器，需要使用到 python 内置的多线程方法。并且，在实验过程中可以发现，如果只是监听多个窗口，网站的访问效率很低。

在本次项目中，不仅代理服务器处理 HTTP 请求是多线程的，处理一个 TCP 连接也用到了多线程的方法。多线程部分的核心代码如下所示

```
t = threading.Thread(target=thread_server, args=(tcpSerSock, webcache))
t.setDaemon(True)
t.start()
```

2.3.2 键盘终止

在之前 lab2 的套接字编程的 web server 中可以发现，并不能够使用键盘打断，只有关闭终端才能停止程序。项目参考了 [9] 中的做法，使程序能够被键盘终止。

```
try:
  ...
except KeyboardInterrupt:
  print('exit')
```

2.3.3 头部字段分析

请求报文的头部有许多重要的信息，如方法 (GET, POST, CONNECT)，主机，请求的文件，客户机的套接字等等。代理服务器能过够解析头部的字段，判断方法，主机地址等等重要信息，从而便于之后针对不同的请求报文做出不同的响应，以及给初始服务器发送相应的请求。

```
def parse_header(header): #从header中获得字段信息
    base, l = 0, len(header)
    fields = dict()
```

```
    i = header.find('\n') - 1
    firstline = header[base: i]
    fields['method'], fields['path'], fields['version'] = firstline.split()
    base = i + 1
    while i < l:
        if header[0] == '\r' and 0 < l-1 and header[1] == '\n':
            break
        while i<l and header[i] != ':':
            i += 1
        if i < l:
            name = header[base:i].strip()
            base = i + 1
            while i < l and not (header[i] == '\n' and header[i-1] =='\r' ):
                i += 1
            value = header[base:i-1].strip()
            fields[name] = value
            base = i + 1
    return fields
```

2.3.4 ssl 加密处理

现行的大多数网站用到了 https 的 ssl 加密处理，表现为 TCP 连接方法是`CONNECT`，此时需要代理服务器向初始服务器发送符合 https 要求的请求报文。

```
def send_https_response(proxySock, tcpCliSock):
    data = b"HTTP/1.0 200 Connection Established\r\n\r\n"
    tcpCliSock.sendall(data)
    # communicate(tcpCliSock, proxySock)
    t = threading.Thread(target=communicate, args = (tcpCliSock, proxySock))
    t.setDaemon(True)
    t.start()

def communicate(sock1, sock2):
    try:
        while True:
            data = sock1.recv(4096)
            if not data:
                return
            sock2.sendall(data)
    except:
        pass
```

2.3.5 使用 select 模块实现 http2.0

http2.0 要求持续连接，因此不能用课本套接字编程所给的`sendall`方法来实现。项目使用了 select 实现了端到端直传，从而让客户端和服务器自行判断是否结束，从而实现 http2.0 的持续连接。

```
(rlist, wlist, elist) = select.select([proxySock], [], [], 3)
if rlist:
  data = rlist[0].recv(4096)
    if len(data) > 0:
```

```
        tcpCliSock.send(data)
    else:
        break
```

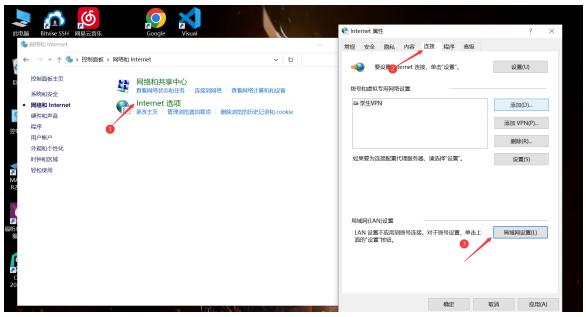
2.3.6 Web 动态分配缓存

为了模拟实际情况下代理服务器内部的磁盘空间有限，而大量主机的大量访问可能会造成磁盘空间满出，本项目使用了一个缓存器类来存放网站内容的缓存。在这个缓存器内，每个 Html 文件以一个数据列表的形式存储，列表中每个元素为一个缓存长度的数据。每个文件都包含一个时间标签，如果有限的存储空间满了后仍有新的访问需要加入新的副本，则会按照时间标签删除最老的副本。由于全部数据都以一样长度的二进制数据 buffer 存放，因此在返回响应的时候，也可以以一份一份 buffer 的形式响应。

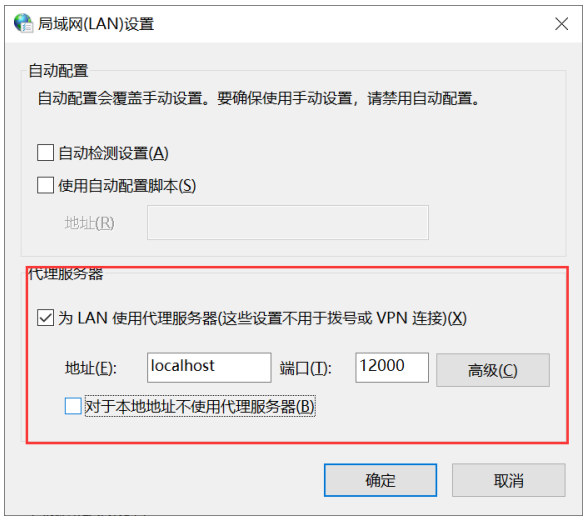
2.4 实验效果展示

使用方法

首先修改本地的代理,将代理如图 12改成对应的本地 IP 和本地端口。然后运行文件`MyProxyServer.py`，即可访问各种网站。



(a) 第一步



(b) 第二步

图 12: 代理服务器设置

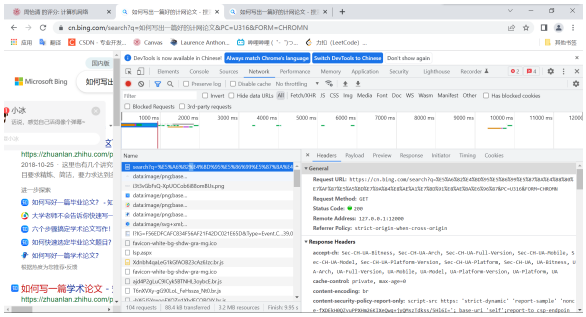
效果展示

访问 bing 搜索引擎图 13

访问上海交通大学 Canvas 系统图 14

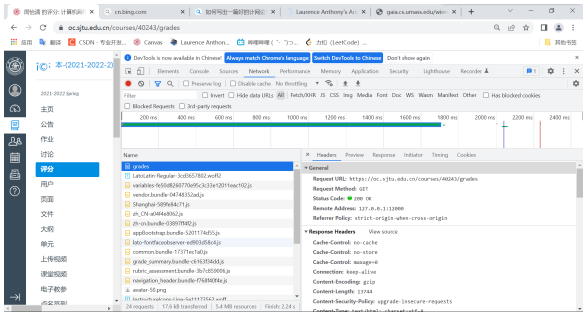
终端中的输出情况图 15

(a) 搜索网页

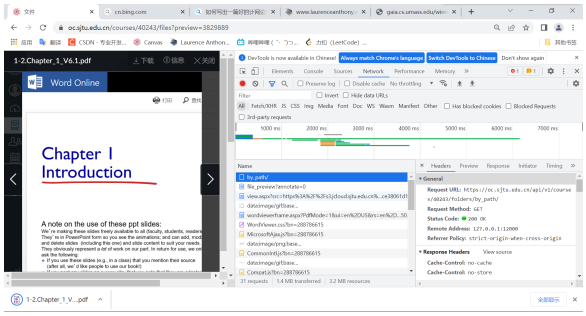


(b) 远程响应的主机 ip

图 13: 访问 bing 搜索引擎



(a) 网页及远程响应的主机 ip



(b) 下载文件

图 14: 访问上海交通大学 Canvas 系统

```
start server
Thread-2---- Received a connection from ('127.0.0.1', 56594)
Thread-3---- Received a connection from ('127.0.0.1', 56595)
Thread-6---- Received a connection from ('127.0.0.1', 56598)
Read from cache
Thread-8---- Received a connection from ('127.0.0.1', 56601)
Read from cache
Thread-10---- Received a connection from ('127.0.0.1', 56607)
Thread-11---- Received a connection from ('127.0.0.1', 56610)
Exception in thread Thread-11:
Traceback (most recent call last):
  File "C:\Program Files\Python38\lib\threading.py", line 932, in _bootstrap_inner
    self.run()
  File "C:\Program Files\Python38\lib\threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "d:/CS3611-计算机网络/CS3611-proj/MyProxyServer.py", line 116, in thread_proxy
    send_response(proxySock, tcpCliSock, filename, cache)
  File "d:/CS3611-计算机网络/CS3611-proj/MyProxyServer.py", line 235, in send_response
    data = rlist[0].recv(4096)
ConnectionResetError: [WinError 10054] 远程主机强迫关闭了一个现有的连接。
Thread-13---- Received a connection from ('127.0.0.1', 56614)
Thread-15---- Received a connection from ('127.0.0.1', 56616)
Read from cache
```

图 15: 终端中的输出情况

3 总结与展望

本文梳理了课本的知识点，沿用自顶向下的分析方式，对于各层的服务模型、各层之间相互协作的关系以及网络层次模型的设计逻辑有了更进一步的阐释。项目还实现了一个代理服务器，可以满足本地主机与外部服务器的交互需求，也进一步加深了对网络服务的理解。

致谢

非常感谢阮娜老师一个学期的 CS3611 学习中的指导与帮助。同时也非常感谢孙泽□助教学长在课程过程中答疑解惑以及对我们的实验辅导。

分工说明

组员都参与了代码、综述和报告部分，工作量相等。

参考文献

[1] James.F.Kurose, *Computer Networking: A Top-Down Approach, Seventh Edition.* Pearson Education, Inc., 2017.

[2] 是曹大大. Ipv4 和 ipv6 报文格式介绍和对比. (2019, Febrary 28). [Online]. Available: https://blog.csdn.net/weixin_41059155/article/details/88015055

[3] duozhishidai. 物联网和互联网之间，主要有什么关系？. (2019, May 4). [Online]. Available: https://blog.csdn.net/duozhishidai/article/details/89812292

[4] . 孙永辉, "代理服务器在智能变电站和调度主站无缝通信中的应用," 计算机工程与应用, pp. 99–101, 2010.

[5] . 徐浩诚, 刘利军, "代理服务器中医学图像自适应分层切割缓存置换策略," 现代电子技术, pp. 72,78–79, 2016.

[6] . 魏宁, 周公建, "地铁信号系统中代理服务器的设计与实现," 铁道通信信号, pp. 84,86–87, 2020.

[7] 张旺俊, "Web 缓存替换策略与预取技术的研究," Master's thesis, 中国科学技术大学, 2011.

[8] . 赵中全, "基于树扩展朴素贝叶斯分类器的 web 代理服务器缓存优化," 计算机工程, pp. 115–119, 2017.

[9] 谁用了我的英文名. Python 实现的简易 http 代理服务器. (2019, March 14). [Online]. Available: https://blog.csdn.net/qq_26946497/article/details/88554922