动态网站开发
【 HTTP 与网络基础】

李博杰

2011-09-24

HTTP 与网络基础

- 计算机网络基础
  - 电路交换与分组交换
  - 时延与丢包
  - 可靠数据传输
  - 因特网协议层次
- HTTP
  - 报文格式
  - 请求行、状态行
  - 首部行
  - 实体主体

电路交换与分组交换

- 电路交换
  - 每条物理链路由 n 条虚电路构成
  - 端到端连接
  - 多路复用
    - 频分多路复用 (FDM)
    - 时分多路复用 (TDM)
  - 打电话
  - 吃饭之前先占座
  - 缺点：静默期电路空闲
  - 优点：通信质量可靠

电路交换与分组交换

- 分组交换
  - 报文 (message) 被划分为数据块：分组 (packet)
  - 分组交换机：输出队列
  - 分组交换性能优于电路交换
    - 假设用户仅有 10% 的时间活动，数据速率 10Mbps，链路输出速率 =100Mbps
    - 若有 35 个用户，仅有 0.0004 的概率超过输出速率（二项式分布）
    - 假设只有一个用户需要传输大量数据……
  - 先打饭，再找座位
  - 手机中的“分组数据”

分组时延

- 分组通过一系列路由器传输，每个节点都有时延
  - 处理时延：确定分组向哪里传送，检查差错，微秒级
  - 排队时延：在输出队列里等待传输，与流量强度关系很大，微秒级到毫秒级
    - 随着流量强度接近 1，平均排队长度越来越长
    - 丢包问题
  - 传输时延：将一个分组的比特推向链路的时间 = 分组长度 / 链路传输速率 (带宽)，微秒级到毫秒级
  - 传播时延：从链路起点传到链路终点 (接近光速)，毫秒级

可靠数据传输

- 丢包了，怎么办
  - 人类的做法：回复确认
  - 发送方若过了一段时间未收到确认，则重新发送
  - 问题：如果“回复确认”丢失了，接收方如何判断这是一条重传的信息还是一条新信息？
  - 让每条信息不同：信息头部加入序号
  - 收到乱序（不是当前所期待的序号）分组，直接丢弃行吗？
  - 收到乱序分组，需要回复 ACK

可靠数据传输

- 传输中发生比特差错，怎么办
  - 检验比特差错：校验码（检验和）
  - 发现出错时人类的做法：你说什么？（特殊分组）
  - “你说什么”又出错了，对方也回一句“你说什么”
- 足够的检验和，不但能检测差错，还能纠正差错
  - 增加了网络传输的开销，实现上相对困难
- 收到错误的分组时直接忽略
  - 发送方一直收不到回复，超时后自动重传
  - 不管是分组丢失、错误还是过度时延，一律重传

可靠数据传输协议

- “ 停等”方式虽然正确，但效率不理想
  - 一个分组丢包，需要等到超时才能收到有效分组
  - 网络协议设计不合理，限制了硬件功能
- 选择重传
  - 只要此序号的分组尚未收到，照单全收
  - 只要正确接收，则确认之
  - 每个分组都有一个 (逻辑) 定时器，以便超时重传

什么是协议

- 人类的通话协议
  - 用什么来开始通话？
  - 收到应答报文后有什么不同反应？
  - 通话双方是否必须执行相同的协议？
- 人类协议
  - 拨电话 - 喂 (你好) – 今天有没有作业？ - ***
- 计算机协议
  - TCP SYN – TCP ACK – GET url – document
  - 可靠数据传输协议

协议层次

- 邮政服务的层次
  - 内容层：内容 -> 信纸：写信：读信
  - 包装层：信纸 -> 信封：装信封，贴邮票：拆信
  - 邮局层：信封 -> 包裹：根据目的地打包：拆包
  - 运输层：包裹 -> 车辆：运送至目的地：按邮局分类
- 模块化：改变服务的实现而不影响系统其他部分

因特网协议层次

- 应用层：网络应用程序的数据传输格式
- 运输层：在应用程序端点间传送应用层报文
  - TCP：面向连接的，可靠传输，拥塞控制
  - UDP：无连接服务，无可靠性保证
- 网络层：将数据报从一台主机移动到另一台主机
- 链路层：将帧从一个节点移动到下一个节点
- 物理层：从物理上传输帧中的每个比特
- 模块化：改变服务的实现而不影响系统其他部分

客户机与服务器

- 对等 (P2P) 体系结构
  - BitTorrent, eMule, Skype...
- 客户机 - 服务器体系结构
  - C/S
  - B/S
  - 客户机向总是打开的服务器发出请求，客户机之间相互不通信
  - 某些 P2P 应用也要用到客户机 - 服务器，如即时通信软件，服务器跟踪登录用户的 IP 地址，用户的主机之间直接发送信息。

运输层多路复用

- 一台主机如何分辨不同的进程？
  - 每个套接字 (socket) 需要有唯一的标识符
  - 每个报文需要携带源主机、目的主机套接字的标识符
- 端口号
  - 0~1023： 周知端口号 (HTTP 80, FTP 21, SSH 22)
  - 1024~65535
- 在客户机 - 服务器应用中，一般客户端自动分配端口号，服务器端分配特定的端口号

HTTP

- 以 TCP 作为运输层协议（可靠数据传输）
- 无状态协议（协议本身不保存客户机信息）
  - 由应用程序依靠 Cookie 保存客户机信息
- 单向连接
  - 服务器不能向客户端主动发起连接
  - 要实现实时连接，或者使用持久连接保持不中断，或者由客户端定时主动发起连接

HTTP 报文格式

- RFC 2616
- HTTP-message = Request | Response
- Generic-message = start-line
  *(message-header CRLF)
  [message-body]
- Start-line = Request-Line | Status-Line
- CRLF = <Carriage Return><Line Feed>

Structure

- Request
  - Request Line
  - Header Fields
  - CRLF
  - Entity Header
  - Entity
- Response
  - Status Line
  - Header Fields
  - CRLF
  - Entity Header
  - Entity

Message Header

- Message-header = field-name ":" [field-value]
- Field-name = token
- Field-value = *(field-content | LWS)
- Field-content = *TEXT | <combination of token, separators, and quoted-string>
- TEXT = <any OCTET except CTLs, but including LWS>
- CTL = <any US-ASCII control character>
- LWS = <Linear white space>

Request Line

- Request Line = Method SP Request-URI SP HTTP-Version CRLF
- Request-URI = ”*” | absoluteURI | abs_path | authority
- GET http://gewu.ustc.edu.cn/gewu/ HTTP/1.1
- GET /pub/index.html HTTP/1.1
  Host: localhost

Request Methods

- GET
  - Retrieve whatever information identified by the Request-URI
- POST
  - Annotation of existing resources
  - (*) Posting a message
  - (*) Providing a block of data (e.g. Submit a form)
  - Extending a database through an append operation
- OPTIONS, HEAD, PUT, DELETE, TRACE, CONNECT

Header Fields

- Cache-Control
- Connection
- Content-Encoding
- Content-Length
- Content-Type
- Host
- Location
- Referer
- User-Agent

Response

- Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
- Status Code
  - 1xx: Informational
  - 2xx: Success (200 OK)
  - 3xx: Redirection (301 Moved Permanently, 302 Found, 304 Not Modified)
  - 4xx: Client Error (400 Bad Request, 403 Forbidden, 404 Not Found, 413 Request Entity Too Large)
  - 5xx: Server Error (500 Internal Server Error, 502 Bad Gateway)

Cookie

- 无状态的：简化了服务器的设计，利于并发
  - 如何根据用户显示不同的内容？
- Cookie (RFC 2109)
  - HTTP 响应报文： cookie 首部行
  - HTTP 请求报文： cookie 首部行
  - 用户端系统： cookie 文件（浏览器管理）
  - 服务器端系统： cookie 数据库
    - 如 PHP 的 SESSION 机制

Cookie

- 客户机：普通 HTTP 请求
- 服务器： HTTP 响应 +Set-cookie 首部行
- 客户机： HTTP 请求 +Cookie 首部行
- 服务器：普通 HTTP 响应
- 客户机： HTTP 请求 +Cookie 首部行
- 服务器：普通 HTTP 响应
- ……

Web 缓存

- 客户机 -> 代理服务器
- 代理服务器检查是否有缓存
  - 有缓存且未过期，直接返回缓存
  - 否则向原始服务器发送请求，取回数据，向客户机返回此数据；根据 HTTP 协议确定是否缓存。
- 减少客户机的响应时间
- 减少网络中的流量
- 浏览器本身也会进行缓存 (Ctrl+F5)

Web 缓存

- 如何判断缓存内容是否过期
- 条件 GET
  - Client: GET
  - Server: Last-Modified: <datetime>
  - Client: If-Modified-Since: <datetime>
  - Server: 304 Not Modified

Thanks!

- Question Time!