2021 年秋季学期“计算机网络”(011144.01)
期中复习—知识点梳理

助教: 董寅灏 朱映 白欣雨 李昱祁 王澍民

2021 年 11 月 于中国科学技术大学

期中考试注意事项

1. 考试时间: 2021 年 11 月 12 日 (周五) 9:45 ~ 11:45
2. 考试地点: 3B103
3. 考试形式: 半开卷 (可带一**张正反面**写有笔记 **A4** 纸)
4. 考试范围: 教材《计算机网络: 自顶向下方法 (**第 7 版**)》**前三章**
5. 考试题型: 单项选择题、简答题、计算题
6. 成绩占比: 期中考试成绩占课程总成绩的 25%
7. 复习参考: 教材、PPT、作业、测验.....

1 计算机网络和因特网

1.1 什么是因特网

1. 因特网定义 1 (具体构成): 由一群**遵循 TCP/IP 协议**的 ISP，按照**松散的层次结构**组织而成的**网络的网络**
2. 因特网定义 2 (服务): 为分布式应用提供通信服务的基础设施

1.2 网络边缘

1. 网络的结构 = 终端 (end systems) + 将终端连接到其边缘路由器的物理链路 (access networks) + 路由器和通信链路组成的网状网络 (network core)
2. 接入网: 住宅接入、企业接入、移动接入.....
3. 物理媒体: 双绞线、同轴电缆、光纤、电磁波.....

1.3 网络核心

1. 任务: 将数据包从发送侧的边缘路由器，传送到接收侧的边缘路由器
2. 基本问题: 数据包如何在网络核心中高效地传递?

(a) 分组传输延迟小

(b) 网络吞吐量高

3. 通信网络中移动数据的两种基本方法:

(a) 电路交换 (独占信道): 电话网使用

(b) 分组交换 (复用信道): 计算机网络使用

4. 分组交换 (**重点理解**)

(a) 过程:

i. 主机将要传输的数据分段，并组装成一系列**分组**

ii. **交换:** 在传输路径上，交换设备从一条链路上接收分组，将其发送到另一条链路上

iii. **存储转发:** 交换设备在接收到完整的分组后，才可以开始转发

(b) 存储转发引入**排队延迟和丢包**

i. 排队延迟: 分组在输出链路的缓存中排队，引入延迟

ii. 丢包: 若输出链路的缓存满，溢出的分组被丢弃

iii. 当大量分组集中到达时，排队延迟和丢包较严重

5. 电路交换: 通话前完成两部电话机之间的电路接续，通话结束后释放整条电路。本质是**预留资源和独占资源**。

6. 分组交换与电路交换的对比? 为什么因特网使用分组交换? 可能引入什么问题? (**重点理解**，课件 34-37 页)

1.4 衡量网络性能的主要指标

1. 延迟 (**重点理解**): 分组从源终端到达目的终端的时间

(a) 分组延迟的来源

i. 节点处理延迟 d_{proc}: 检查错误，确定输出链路的时间。典型地为几个微秒或更低

ii. 排队延迟 d_{queue}: 在输出缓存等待传输的时间。**差异很大，取决于链路负载**

iii. 传输延迟 d_{trans}: 将分组发送到链路上的时间。用 L bits 表示分组长度，R bps 表示链路传输速率 (带宽)，则 $d_{trans} = L/R$。微秒 ~ 毫秒，主要取决于链路速率

iv. 传播延迟 d_{prop}: 分组在链路上传播的时间。用 d 表示物理链路的长度，s 表示该链路的传播速率，则 $d_{prop} = d/s$。几微秒 ~ 几百毫秒，主要取决于链路长度

(b) 节点的总延迟 $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$，**这些延迟成分所起的作用可能会有很大的不同**

(c) 端到端延迟: 分组传输路径上所有节点的节点延迟之和

2. 丢包率: 未成功交付到目的终端的分组比例。队列长度是一个重要的参数: 队列太短，丢包率增大；队列太长，排队延迟增大 (也会造成间接丢包)

3. 吞吐量: 单位时间内向接收端成功交付的数据量。端到端吞吐量与瓶颈链路的带宽、以及链路上的负载有关

1.5 协议层次及其服务模型

1. 网络协议定义了:
   (a) 通信实体之间交换的报文的格式和次序
   (b) 在发送/接收报文、或其它事件后采取的动作
2. 分层: 将网络按功能划分成一系列水平的层次，每一层实现一个功能 (服务)。每一层的功能实现都要依赖其下层提供的服务
   (a) 在不同系统的同一层上: 对等实体执行该层的协议
   (b) 相同系统的上下层: 调用服务/提供服务；封装/解封装分组
   (c) 不同系统的不同层: 不直接通信
3. Internet 协议栈 (**重点理解**)
   (a) 应用层: 在应用程序之间传输应用特定的报文 (message)。例如: FTP, SMTP, HTTP
   (b) 传输层: 在应用程序与网络的接口间 (进程-进程) 传输报文段 (segment)。例如: TCP, UDP
   (c) 网络层: 在源主机和目的主机 (终端-终端) 之间传输分组 (packet)。例如:IP, 路由协议
   (d) 链路层: 在相邻设备之间传输帧 (frame)。例如: PPP, Ethernet
   (e) 物理层: 在物理媒体上传输比特 (bit)
4. ISO/OSI 参考模型: 附加两个层
   (a) 表示层: 使通信的应用程序能够解释交换数据的含义，这些服务包括数据压缩和数据加密以及数据描述
   (b) 会话层: 提供了数据交换的定界和同步功能，包括了建立检查点和恢复方案的方法

2 应用层

2.1 网络应用程序体系结构

1. **客户/服务器模型**: 在客户/服务器 (Client/Server, C/S) 模型中，有一个总是打开的主机称为服务器，它服务于许多来自其他称为客户机的主机请求。其工作流程如下:
   - 服务器处于接收请求的状态。
   - 客户机发出服务请求，并等待接收结果。
   - 服务器收到请求后，分析请求，进行处理，将结果并发送给客户机。
2. **P2P 模型**: 在 P2P 模型中，各计算机没有固定的客户和服务器划分，任意一对计算机称为对等方 (Peer)，可以直接相互通信。相比 C/S 架构其优点主要体现如下:
   - 减轻了服务器的计算压力，消除了对某个服务器的完全依赖，将任务分配到各个结点上，提高了系统效率和资源利用率。
   - 可扩展性好。
   - 网络健壮性强，单个结点的失效不会影响其他结点。

   P2P 模型也有缺点，在获取服务的同时，还要给其他结点提供服务，因此会占用较多的内存。

2.2 超文本传输协议 (HTTP)

概念

网页由一些对象 (object) 组成。对象简单来说就是文件，可以是 HTML 文件、图像等，每个对象通过一个 URL 获取。URL 一般的格式为:

< 协议 >://< 主机 >:< 端口 >/< 路径 >

注意，< 端口 > 与 < 路径 > 可以省略。

HTML 是一种格式语言，用于描述页面的内容和显示方式。HTML 文件是用 HTML 语言编写的超文本文件，Web 页通常包含一个 HTML 基本文件和若干引用对象。

工作过程

1. 每个万维网站点都有一个服务器进程，它不断地监听 80 端口 (默认)，当监听到连接请求后便与浏览器建立连接。
2. TCP 连接建立后, 浏览器就向服务器发送 HTTP 请求。
3. 服务器收到 HTTP 请求后，将构建所请求的 Web 页的必需信息，通过 HTTP 响应返回。
4. 浏览器将信息进行解释，然后将网页显示给用户。
5. 最后 TCP 连接释放。

特点

1. HTTP 是无状态的，同一个客户第二次访问同一个服务器上的页面时，服务器的响应与第一次被访问时的相同。
2. HTTP 采用 TCP 作为传输层协议，保证了数据的可靠传输。注意，HTTP 本身是无连接的，通信的双方在交换 HTTP 报文之前不需要先建立 HTTP 连接。
3. HTTP 既可以使用非持久连接，也可以使用持久连接 (HTTP/1.1 支持)。对于非持久连接，每个网页元素对象的传输都需要单独建立一个 TCP 连接；持久连接指万维网服务器在发送响应后仍然保持这条连接，使同一个客户和服务器可以继续在这条连接上传送后续的 HTTP 请求与响应报文。

持久连接又分为非流水线和流水线两种方式。对于非流水线方式，客户在收到前一个响应后才能发出下一个请求。HTTP/1.1 的默认方式是使用流水线的持久连接，这种情况下，客户每遇到一个对象引用就立即发出一个请求，因此客户可以逐个地连续发出对各个引用对象的请求。如果所有的请求和响应都是连续发送的，那么所有引用的对象共计经历 1 个 RTT 延迟，而不是像非流水线方式那样。每个引用都必须有 1 个 RTT 延迟。

HTTP 报文结构

HTTP 是面向文本的，因此报文中的每个字段都是一些 ASCII 码串。有两类 HTTP 报文:

2.3 文件传输协议 (FTP)

工作原理

FTP 是因特网上使用得最广泛的文件传输协议。FTP 允许客户指明文件的类型与格式并允许文件具有存取权限。

FTP 采用客户/服务器的工作方式，它使用 TCP 可靠的传输服务。一个 FTP 服务器可同时为多个客户端同时服务。FTP 的服务器进程由两大部分组成: 一个主进程，负责接收新的请求；若干从属进程，负责处理单个请求。其工作步骤如下:

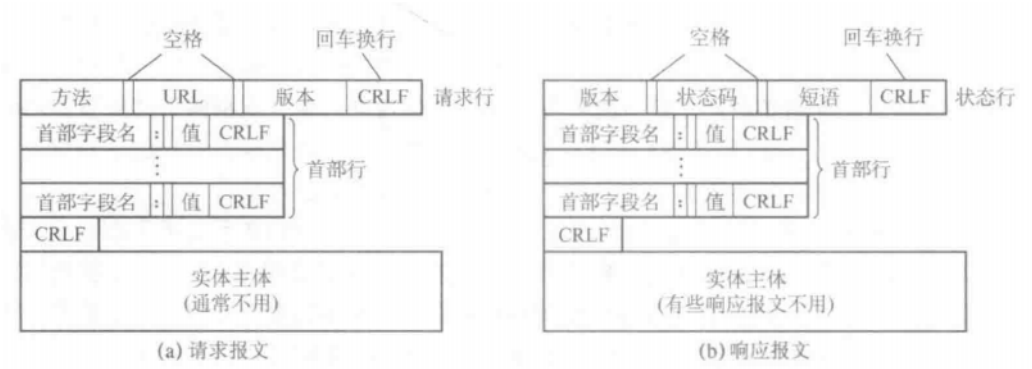

图 1: HTTP 报文结构

1. 打开熟知端口 21 (控制端口)。
2. 等待客户进程发连接请求。
3. 启动从属进程来处理客户进程发来的请求。
4. 回到等待状态，继续接收其他客户进程的请求。

可自行了解 FTP 主动与被动模式的区别。

控制连接

服务器监听 21 号端口，等待客户连接，控制连接用来传输控制信息 (如连接请求、传送请求等)。在传输文件时还可以使用控制连接 (如客户在传输中途发一个中止传输的命令)，因此控制连接在整个会话期间一直保持打开状态。

数据连接

数据连接用来连接客户端和服务器端的数据传送进程，数据传送进程实际完成文件的传送，在传送完毕后关闭数据传送连接并结束运行。

2.4 电子邮件协议

系统组成

用户代理 (UA): 用户与电子邮件系统的接口。用户代理使用户能够发送和接收邮件。通常情况下，用户代理就是一个运行在 PC 上的程序，比如 Outlook、Foxmail 等。

邮件服务器: 电子邮件系统的核心。邮件服务器的功能是发送和接收邮件，同时还要向发信人报告邮件传送的情况 (已交付、被拒绝、丢失等)。邮件服务器采用客户/服务器方式工作，它能够同时充当客户和服务器。

邮件发送协议和读取协议: 邮件发送协议用于用户代理向邮件服务器发送邮件或在邮件服务器之间发送邮件，通常使用的是 SMTP；邮件读取协议用于用户代理从邮件服务器读取邮件，如 POP3。

电子邮件格式

一个电子邮件分为信封和内容两大部分，邮件内容又分为首部和主体两部分。RFC822 规定了邮件的首部格式，邮件的主体部分则让用户自由撰写。用户写好首部后，邮件系统自动地将信封所需的信息提取出来并写在信封上。

邮件内容的首部包含一些首部行，**To** 是必需的关键字，后面填入一个或多个收件人的电子邮件地址。电子邮件地址的规定格式为: 收件人邮箱名 @ 邮箱所在主机的域名。**Subject** 是可选关键字，是邮件的主题，反映了邮件的主要内容。还有一个必填的关键字是 **From**，它通常由邮件系统自动填入。

多用途网际邮件扩充 (MIME)

SMTP 只能传送一定长度的 ASCII 码，许多其他非英语国家的文字就无法传送，且无法传送可执行文件及其他二进制对象，因此提出了多用途网络邮件扩充 (Multipurpose Internet Mail Extensions, MIME)。MIME 的意图是继续使用目前的格式，但增加了邮件主体的结构，并定义了传送非 ASCII 码的编码规则，即 MIME 邮件可在现有的电子邮件程序和协议下传送。

SMTP 与 POP3

简单邮件传输协议 (Simple Mail Transfer Protocol, SMTP) 是一种提供可靠且有效的电子邮件传输的协议，它控制两个相互通信的 SMTP 进程交换信息。SMTP 使用客户/服务器方式，TCP 连接 (端口号为 25)。

邮局协议 (Post Office Protocol, POP) 是一个非常简单但功能有限的邮件读取协议。POP3 采用的是“拉” (Pull) 的通信方式，当用户读取邮件时，用户代理向邮件服务器发出请求，“拉”取用户邮箱中的邮件。POP 也使用客户/服务器的工作方式，在传输层使用 TCP, 端口号为 110。POP 有两种工作方式: “下载并保留”和“下载并删除”。

2.5 域名系统 (DNS)

域名系统 (Domain Name System, DNS) 是因特网使用的命名系统，用来把便于人们记忆的具有特定含义的主机名转换为便于机器处理的 IP 地址。DNS 系统采用 C/S 模型，其协议运行在 UDP 之上，使用 53 端口。

层次域名

因特网采用层次树状结构的命名方法。任何一个连接到因特网的主机或路由器，都有一个唯一的层次结构名称，即域名 (Domain Name)。域 (Domain) 是名字空间中一个可被管理的划分。域还可以划分为子域，而子域还可以继续划分，这样就形成了顶级域、二级域等。

关于域名中的标号有以下几点需要注意:

- 标号中的英文不区分大小写。标号中除连字符 (-) 外不能使用其他的标点符号。
- 每个标号不超过 63 个字符，多标号组成的完整域名最长不超过 255 个字符。
- 级别最低的域名写在最左边，级别最高的顶级域名写在最右边。

顶级域名 (Top Level Domain, TLD) 分三大类:

- 国家顶级域名 (nTLD)。国家和某些地区的域名，如“.cn”表示中国，“.us”表示美国。
- 通用顶级域名 (gTLD)。常见的有“.com” (公司)、“.net” (网络服务机构)、“.org” (非营利性组织) 等。
- 基础结构域名。这种顶级域名只有一个，即 arpa，用于反向域名解析，因此又称反向域名。

域名服务器

因特网的域名系统被设计成一个联机分布式的数据库系统。一个服务器所负责管辖的 (或有权限的) 范围称为区，每个区设置相应的权限域名服务器，用来保存该区中的所有主机的域名到 IP 地址的映射。每个域名服务器不但能够进行一些域名到 IP 地址的解析，而且还必须具有连向其他域名服务器的信息。

下面介绍 4 类域名服务器。

根域名服务器是最高层次的域名服务器，所有的根域名服务器都知道所有的顶级域名服务器的 IP 地址。若要对因特网上任何一个域名进行解析，只要自己无法解析，就首先要求助于根域名服务器。根域名服务器用来管辖顶级域 (如.com)。

顶级域名服务器负责管理在该顶级域名服务器注册的所有二级域名。收到 DNS 查询请求时，就给出相应的回答 (可能是最后的结果，也可能是下一步应当查找的域名服务器的 IP 地址)。

授权域名服务器 (权限域名服务器)，每台主机都必须在授权域名服务器处登记。许多域名服务器都同时充当本地域名服务器和授权域名服务器。授权域名服务器能将其管辖的主机名转换为该主机的 IP 地址。

本地域名服务器对于域名系统很重要，当一台主机发出 DNS 查询请求时，这个查询请求报文就发送给该主机的本地域名服务器。

注: 在 Windows 系统中配置“本地连接”时，就需要填写 DNS 地址，这个地址就是本地 DNS(域名服务器) 的地址。

解析过程

递归查询: 如果本地主机所询问的本地域名服务器不知道被查询域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向根域名服务器继续发出查询请求报文，在这种情况下，本地域名服务器只需向根域名服务器查询一次，后面的几次查询都是递归地在其他几个域名服务器之间进行。

迭代查询: 当根域名服务器收到本地域名服务器发出的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地域名服务器下一步应当向哪个顶级域名服务器进行查询。然后让本地域名服务器向这个顶级域名服务器进行后续的查询，如此迭代。

Tips: 为了提高 DNS 的查询效率，并减少因特网上的 DNS 查询报文数量，在域名服务器中广泛地使用了高速缓存。当一个 DNS 服务器接收到 DNS 查询结果时，它能将该 DNS 信息缓存在高速缓存中。当另一个相同的域名查询到达该 DNS 服务器时，该服务器就能够直接提供所要求的 IP 地址。因为主机名和 IP 地址之间的映射不是永久的，所以 DNS 服务器将在一段时间后丢弃高速缓存中的信息。

3 传输层

3.1 概述和传输层服务

1. 传输层位于网络层之上，用于提供**逻辑通信功能**，从应用程序角度看运行不同进程的主机好像直接相连一样。应用只需要使用，不用考虑底层设施细节。(课本两个家庭信件例子)

2. 传输层把报文转换成传输层分组，称为传输层报文段 (segment)。网络传输层可以使用多种协议，包括 **TCP 和 UDP 协议**。数据报本书中指网络层分组

3. 因特网网络层协议叫网际协议 (IP)，服务模型 “**尽力而为交付服务**”，不做任何确保，因此被称为不可靠服务。即使网络层协议不能提供可靠的服务，传输协议也可以为应用程序提供可靠的数据传输服务

3.2 多路复用与多路分解

1. 将主机间交付扩展到进程间交付被称为传输层多路复用与多路分解。

2. 多路分解和多路复用: 在**接收端**，传输层检查每个传输层报文段中具有的几个字段，标识出接收套接字，进而将报文段定向到该套接字。将传输层报文段中的数据交付到正确的套接字的工作称为**多路分解**。在**源主机**从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 (这将在以后用于分解) 从而生成报文段，然后将报文段传递到网络层，所有这些工作称为**多路复用**。套接字提供了同一主机或者不同主机上的进程之间的通信。

3. HTTP 服务器使用 80 端口，为每个连接生成一个新进程，每个进程都有自己的套接字 (或者生成子线程只使用一个套接字，可以有多个不同套接字连接到同一个进程；线程比进程更为轻量级)。但是这并不代表 TCP 是点对多或者多对多的!

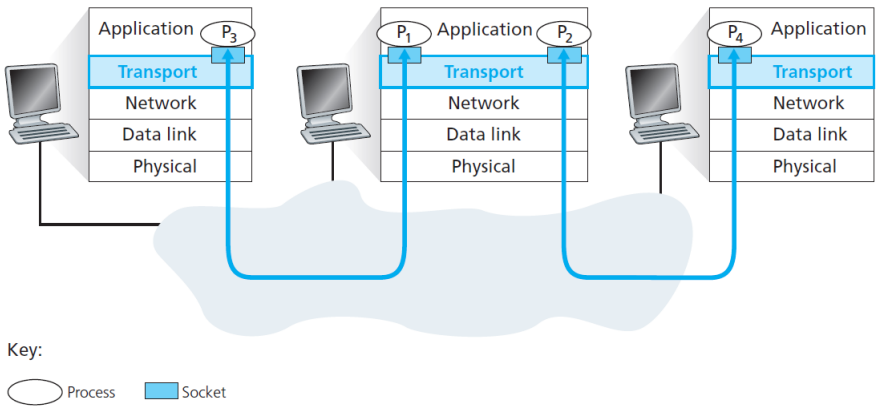

图 2: 传输层的多路复用与多路分解

3.3 无连接传输:UDP

1. UDP 套接字由一个二元组标识 (目的 IP 地址和目的端口号)。TCP 套接字由四元组标识 (源 IP 地址，源端口，目的 IP 地址，目的端口)。应用程序创建套接字后传递信息，系统为套接字分配端口号，保证不同目的地的数据可以到达正确进程。

2. UDP 无非就是对网络层协议增加了一点复用/分解服务。发送之前双方**无需握手**，所以称其为无连接的，最典型的例子是 DNS。进行一次查询时，程序构造了一个 DNS 查询报文并将其交给 UDP。主机端的 UDP 为此报文添加首部字段，然后交给网络层。网络层封装进一个 IP 数据报中，然后将其发送给一个 DNS 服务器

3. 尽管 UDP 只能尽力而为，但是由于 UDP 有以下特点: 关于控制何时发送什么数据更为精细；无需建立连接；无需维护连接状态；分组首部开销小 (只有 8 字节，TCP20 字节)。所以很多延迟敏感应用采用 UDP 而非 TCP

4. UDP 报文段由源端口、目的端口号、长度、检验和 (仅提供差错检测，无恢复能力) 以及应用数据 (报文) 组成。前四个每个占用 8 比特，共 64 比特 8 字节

3.4 可靠数据传输原理

1. 目的: 在不可靠的网络层上实现可靠的传输协议

2. 构造过程:

   (a) rdt 1.0: 经完全可靠底层信道的可靠数据传输。底层完全可靠，无需任何反馈，状态机简单

   (b) rdt 2.0: 经具有比特差错信道的可靠数据传输

      i. 需要控制报文表达肯定或者否定确认，重复传递有误内容，因此也被称为自动重传请求协议。要求差错检测、接收方反馈和重传

      ii. 缺点: 没有考虑到**确认报文受损**的可能性 (ACK or NAK)

      iii. 解决方法: 增加足够的检验比特从而可以纠错；或者收到模糊确认时重传。但是又需要解决**冗余分组**的问题，即采用目前广泛应用的**对分组编号**

   (c) rdt 2.1: 解决冗余分组问题，采用 0、1 序号

   (d) rdt 2.2: 不需要 NAK 分组，接收方对**上次正确接受的分组**重发 ACK 来告知发送方

(e) rdt 3.0: 经具有比特差错信道的可靠数据传输

i. 是一种停等协议。又因为采用 0、1 序号，又称**比特交替**协议

ii. 采用倒计数**定时器**，每发送一个分组启动一个定时器，收到 ACK 终止，否则超时重传对应分组

(f) 小结: 可靠数据传输需要检验和、序号、计时器、肯定和否定确认分组技术

(g) 问题:rdt 3.0 性能很差。根据发送方利用率公式可以计算其利用率。解决方法是不采用停等，允许发送多个分组无需确认，即**流水线技术**

$$U_{sender} = \frac{L/R}{\mathrm{RTT} + L/R}$$

(h) 流水线技术要求: 增加序号范围，发送、接收方能缓存多个分组，正确处理丢包、损坏与延时过大问题

3. **GBN** 回退 N 步协议

(a) 允许发送方在有多个分组可用时发送多个分组而无需等待确认，但未确认分组不能超过 N，N 被称为窗口长度，因此本协议也叫做**滑动窗口协议**。采用窗口而不是无限发送的原因是进行流量控制

(b) 现实中序号采用固定长度比特承载，序号范围 $[0, 2^k - 1]$，涉及序号的运算需要采用模 2^k 运算

(c) GBN 采用**累积确认**，接收方发送的 ACK N 表明正确收到小于等于 N 的所有分组。接收方会丢弃所有失序分组。相当于接受窗口大小为 1 的选择重传协议

(d) 回退 N 步出现在丢失或超时的情况。发送方此时**重传所有**已发送但未被确认的分组 (最多为 N)。所以发送方只对窗口基序号分组维护一个计时器

4. **SR** 选择重传

(a) 和 GBN 相比增加了接收窗口，失序分组会被缓存；增加了计时器，对所有发送但未确认的分组计时，只需重发超时分组

(b) 发送和接收方窗口长度在相等的情况下应小于等于序号空间大小的一半。否则极端情况无法确认分组是一次重传还是新分组

5. 小结: 进一步采用了检验和、定时器、序号、确认、否定确认、窗口和流水线技术

3.5 面向连接的传输:TCP

1. 定义:TCP 是因特网传输层的面向连接的可靠的传输协议，依赖于包括差错检测、重传、累积确认、定时器以及用于序号和确认号的首部字段等技术。由于 TCP 在发送数据之前需要握手，双向通信，进程对应，所以被称为**面向连接的**、**全双工**、**点对点的**。它兼有 GBN 和 SR 的特点

2. TCP 组成包括一台主机上的**缓存**、**变量和与进程连接的套接字**，以及另一台主机上的另一组缓存、变量和与进程连接的套接字。在这两台主机之间的网络元素 (路由器、交换机和中继器) 中，没有为该连接分配任何缓存和变量连接双方需要为连接初始化变量，连接的状态仅存在于端系统中而不存在于中间的网络中

3. TCP 根据 MTU 确认最大报文长度，分段从发送缓存中取出要发送的内容，包装上 TCP 首部后传给网络层。接收方同样将收到的报文放入接受缓存，供应用程序读取

4. TCP 首部结构
   (a) 和 UDP 类似的源端口号、目的端口号、数据、因特网检验和 (各 2 字节共 8 字节)
   (b) 序号和确认号 (各 4 字节共 8 字节) 文档首字节序号为 0(可以自行确定)，对每个字节编号。确认号为期望接收到的下一字节序号，只确认从第一个到第一个丢失为止的所有字节，所以 TCP 被称为提供累积确认。
   (c) 接受窗口字段 (2 字节) 指示接收方愿意接受的字节数量
   (d) 首部长度字段 (4 比特可变)
   (e) 选项 (可选，可变长) 协商最大报文长度等
   (f) 标志 (6 比特) 包括 ACK，RST，SYN，FIN，PSH，URG
5. 因为采用超时/重传机制，所以需要**估计往返时间**来确认超时间隔长度。计算过程:
   (a) 计算 SampleRTT
   (b) 更新 EstimatedRTT(α 参考值 0.125)

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

   (c) 更新 DevRTT(β 参考值 0.25)

$$\text{DevRTT} = (1-\beta)\cdot\text{DevRTT} + \beta\cdot|\text{SampleRTT} - \text{EstimatedRTT}|$$

   (d) 设置超时间隔 (初始推荐 1s) 大多数 TCP 实现中超时间隔在超时事件发生时直接加倍，简易形式的拥塞控制

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4\cdot\text{DevRTT}$$

6. 流量控制: 发送速率与接受速率相匹配，这种形式的发送方控制称为**拥塞控制**。发送方确保接受窗口足够大防止接收方溢出

$$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

$$\text{LastByteSent} - \text{LastByteAcked} \le \text{rwnd}$$

7. 三**次握手**: 第一步，客户端发送 SYN 报文段，选择初始序号；第二步，服务器分配资源，返回 SYNACK 报文段；第三步，客户端分配资源，可以开始发送资源。挥手 (断开连接) 操作为 4 步

3.6 拥塞控制原理

1. 为了处理网络拥塞，需要限制发送方。原因: 路由器缓存，链路容量，其他因素。
2. TCP 采用端到端拥塞控制。网络辅助拥塞控制: 网络层构件比如路由器可以反馈关于链路中的拥塞情况 (RM 资源管理信元)，显式通知发送方降低速率

3.7 TCP 拥塞控制

1. 拥塞窗口 cwnd(congestion window) 限制发送方能向网络中发送流量的速率

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min\{\text{cwnd}, \text{rwnd}\}$$

2. **TCP 拥塞控制算法 (重点理解)**

   (a) 慢启动: cwnd 初始设置为一个 MSS，之后如果没有超时或冗余 ACK 则每一个传输回合翻一倍。监测到超时，将 ssthresh(慢启动阈值) 设置为 cwnd/2，再设置 cwnd = 1，重新开始慢启动；监测到 cwnd = ssthresh，进入拥塞避免状态；监测到三个冗余 ACK，ssthresh = cwnd/2, cwnd = ssthresh+3，进入快速恢复

   (b) 拥塞避免: 进入拥塞避免表明此时 cwnd 大约为上一次拥塞时的一半大小，所以每个 RTT 都只增加 1。监测到超时，ssthresh = cwnd/2, cwnd = 1，进入慢启动；监测到三个冗余 ACK，ssthresh = cwnd/2, cwnd = ssthresh+3，进入快速恢复

   (c) 快速恢复: 只有 TCP Reno 版本拥有，TCP Tahoe 没有，后者不论超时还是冗余 ACK 都重新慢启动。要进入这个状态必须要三个冗余 ACK，这个状态下针对缺失的报文段每收到一个重复 ACK 让 cwnd 增加 1。如果超时则 ssthresh = cwnd/2, cwnd = 1，进入慢启动；如果得到新的 ACK 表明已经成功发送，设置 cwnd = ssthresh，进入拥塞避免

   (d) **快速重传**: 如果 TCP 发送方接收到同一个数据的三次冗余 ACK，立即重传丢失分组，忽视计时器。因为冗余 ACK 表示网络中可能并没有阻塞，只是因为某些原因导致部分报文丢失，所以此时仍然将拥塞窗口设置为 1 进入慢启动是浪费带宽的。可以提高链路吞吐率

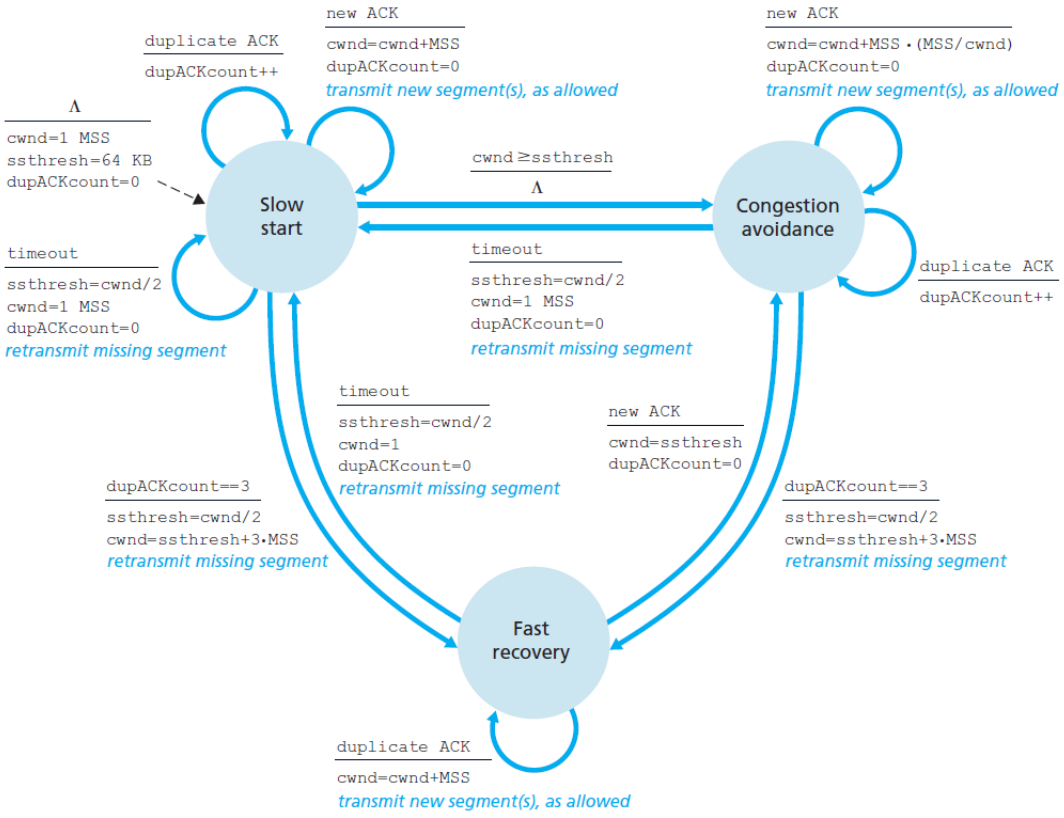

图 3: TCP 拥塞控制的 FSM 描述

3. 举例 (课本例)。如图3所示，这个例子中，初始门限为 8，一开始指数增长，在第八回合检测到冗余 ACK。两种协议都设置门限为一半，TCP Reno 设置 cwnd 为门限加 3，TCP Tahoe 重启动。这张图中没有收到新的 ACK，因为 TCP Reno 的直线一直增长而没有降低为 ssthresh。可以看出拥塞控制的特征: 加性增，乘性减

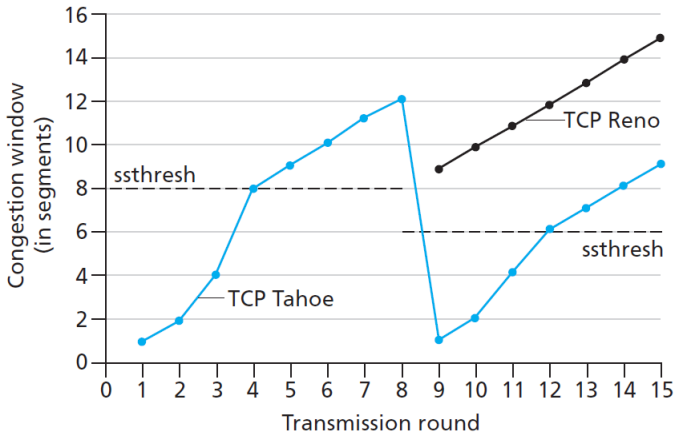

图 4: TCP 拥塞窗口的演化 (Tahoe 和 Reno)