计算机网络：自顶向下方法(Computer Networking : A Top-Down Approach) 术语概念整理

> **Author: MQ_Adventure**

> **Duration: June, 2022**

第一章：计算机网络和因特网

1.1 什么是因特网

主机 (host) / 端系统 (end system)

通信链路 (communication link)

分组交换机 (packet switch)：包括路由器和链路层交换机，作用：forward packets

传输速率 (transmission rate): bandwidth

分组 (packet)

路由器（router）

链路层交换机（link-layer switch）

路径（path）：一个分组所经历的一系列通信链路和分组交换机成为通过该网络的路径

因特网服务提供商 （ISP）：端系统通过 ISP 接入因特网，ISP 自身就是一个由多台分组交换机和多段通信链路组成的网络

请求评论（RFC）& 因特网工程任务组（IETF）：互联网标准

套接字接口（socket interface）：一套发送程序必须遵循的规则集合

协议（protocol）：定义了在两个或多个通信实体之间交换的报文的格式（format）和顺序(order)，以及报文发送和/或接受一条报文或其他时间所采取的动作(action taken)

1.2 网络边缘

接入网（access network）：将端系统物理连接到其边缘路由器的网络

数字用户线（DSL）& 数字用户线接入复用器（DSLAM）：在用户一侧，一个分配器把到达家庭的数据信号和电话信号分隔开，并将数据信号转发给 DSL 调制解调器； 在电话公司，DSLAM 把数据和电话信号分隔开，并将数据送往因特网。

电缆（fiber cable）：共享广播媒体

数字用户线接入复用器（DSLAM）：将数据和电话信号分隔开，并将数据送往因特网

不对称接入（asymmetric）：上行速率和下行速率不同

混合光线同轴（HFC）：同时使用光纤和同轴电缆

物理媒体：同轴电缆（coaxial cable）、光纤（fiber opic cable）、卫星（satellite）

1.3 网络核心

存储转发传输（store-and-forward transmission）：在交换机能够开始向输出链路传输该分组的第一个比特之前，必须接收到整个分组

输出缓存（output buffer）

排队时延（queuing delay）

分组丢失（packet loss）

路由选择协议（routing protocol）

电路交换（circuit switching）：dedicated resource，no sharing，恒定速率，预留资源（缓存，链路传输速率）

分组交换（packet switching）：更好的带宽共享，更加简单高效，实现成本更低

频分复用（FDM）：different channels transmitted in different frequency bands（不同的频道在不同的带宽里面传输）

时分复用（TDM）：不同的频道在不同的时隙里面传输

存在点（PoP）:其中客户 ISP 可以与提供商 ISP 连接

多宿（multi-home）：可以与两个或者更多提供商 ISP 连接

对等 （peer）：可以直接将相同等级结构的 ISP 连接，使得它们之间的所有流量经过直接连接而不是通过上游的中间 ISP 传输

因特网交换点（IXP）：是一个汇合点，多个 ISP 能够在这里一起对等

内容提供商网络（content provider network）: private network that connects its data centers to Internet, often bypassing tier-1, regional ISPs

---

1.4 分组交换网中的时延、丢包和吞吐量

处理时延（processing delay）: 检查分组首部和决定将该分组导向何处所需要的时间等等处理时间

排队时延（queuing delay）：取决于网络的拥塞程度

传输时延（transmission delay）：将所有分组的比特推向链路所需要的时间

传播时延（propagation delay）：从链路起点到路由器传播所需要的时间

流量强度（traffic intensity）：
La/R $(a$是分组到达队列的平均速率，L是一个分组含有的比特数量$)$

端到端时延：$d_{end-to-end} = N(d_{proc} + d_{trans} + d_{prop})$

瓶颈链路（bottleneck link）: link on end-end path that constrains end-end throughput

---

1.5 协议层次及其服务模型

协议栈（protocol stack）：各层的所有协议

报文（message）：位于应用层的信息分组

报文段（segment）：位于运输层的分组

数据报（datagram）：位于网络层的分组

帧（frame）：链路层分组

1.6 面向攻击的网络

病毒（virus）：需要某种形式的用户交互来感染用户设备的软件

蠕虫（worm）：一种无须任何明显用户交互就能进入设备的恶意软件

拒绝服务攻击（Denial-of-Service (DoS) attack）： 1. select target 2. break into hosts around the network 3. send packets to target from compromised hosts.

分组嗅探器（packet sniffer）

IP 哄骗（IP spoofing）：将具有虚假源地址的分组诸如因特网

如何连接端系统和边缘路由器：接入网（access network）（家庭接入网，机构接入网（以太网））

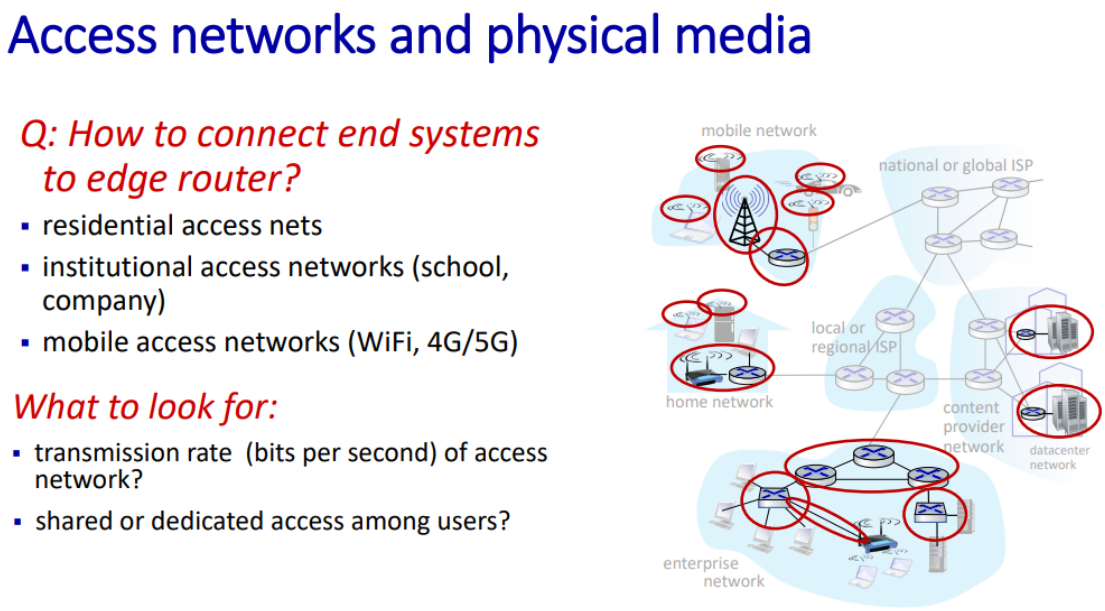
互联接入 ISP 的中心目标：使得所有端系统能够彼此发送分组

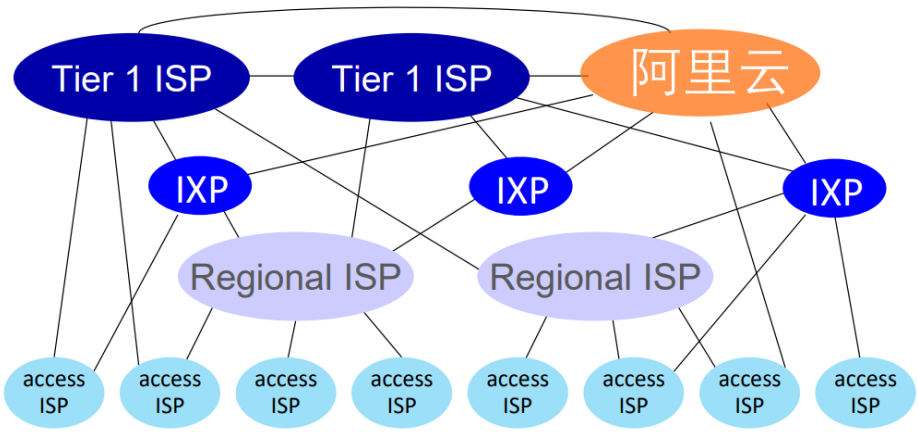
OSI 比传统的五层协议结构多出来会话层以及表示层

链路层交换机实现了第一层和第二层，路由器实现了第一层到第三层协议栈

一个分组具有两种类型的字段：首部字段和有效载荷字段(payload field)

DoS攻击类型：弱点攻击，带宽洪泛，连接洪泛





第二章：应用层

2.1 应用层协议原理

客户-服务器体系结构

- 客户相互之间不直接通信
- 服务器总是开启，并且有固定的周知的地址，也就是 IP 地址

P2P体系结构

- 可扩展性（scalability）好
- 客户端之间可以直接通信

在相同端系统之间的进程使用进程间通信机制相互通信，在不同端系统之间的进程通过跨越计算机网络交换报文而相互通信。

套接字是同一台主机内应用层和运输层之间的接口。

标识进程需要的信息：1. 主机地址 2.在目的主机中指定接收进程的标识符

应用程序对运输层协议的要求：

- 可靠数据运输（是否可以确保由应用程序的一段发送的数据正确、完全地交付给该应用程序的另一端）
- 吞吐量保证（具有吞吐量要求的应用成为带宽敏感的应用）
- 定时保证
- 安全性

TCP 服务模型包括 1. 面向连接服务 2. 可靠数据传输服务

TCP 和 UDP 都没有提供加密机制，且不能够提供定时和吞吐量的保证

2.2 Web 和 HTTP

Web 的应用层协议是超文本传输协议（HTTP）

Web 页面是由对象组成的，一个 HTML 文件也是一个对象，每个对象都存储在服务器端，每个对象由 URL 进行寻址，每个 URL 地址包含主机名 + 路径名

HTTP 使用 TCP 作为它的支撑运输协议，并且 HTTP 是一个无状态协议（stateless protocol）

持续连接（persistent connection）：所有的请求及其相应经相同的 TCP 连接发送，HTTP 在默认方式下使用持续连接

往返时间（Round-Trip Time, RTT）：一个短分组（即不考虑传输时延）从客户到服务器然后再返回客户所花费的时间。包括分组传播时延、分组在中间路由器和交换机上的排队时延和分组处理时延。由于涉及到三次握手，所以总的相应时间就是两个 RTT （一个用于建立 TCP 连接，一个用于请求以及接受对象）加上服务器传输 HTML 文件的时间。

HTTP 报文使用 ASCII 文本书写

HTTP 请求报文至少有一行，为请求行（request line），有三个字段（方法字段（GET, POST,...），URL 字段以及 HTTP 版本字段）。后继的就称为首部行（header line）。请求报文最后是实体体，如果使用 GET 方法，实体体为空，如果使用 POST 方法，实体体用来提交表单。

HTTP 响应报文 有三个部分，一个初始状态行（status line）（状态行有三个部分，协议版本字段、状态码以及相应状态信息），一个首部行，一个实体体（报文的主要部分）。

HTTP响应报文首部行里面的 Date 信息是指服务器检索、插入、发送该响应报文的时间，而不是对象创建或者最后修改的时间。而 Last-Modified：才是指示了对象创建或者最后修改的日期和时间。Content-Length：首部行指示了被发送对象中的字节数。

Cookie 技术四大组件：

- HTTP 响应报文当中的 cookie 首部行（Set-cookie: xxxx）
- HTTP 请求报文中的 cookie 首部行 (Cookie: xxxx)
- 在用户端系统中保留有一个 cookie文件，并由用户浏览器管理
- 位于 Web 站点的一个后端数据库

在用户端系统拥有对应网页的 cookie 文件之后，每请求一个 Web 页面，其浏览器都会查询该 cookie 文件并抽取她对这个网站的识别码，放到请求报文中包括识别码的 cookie 首部行中。

Web 缓存器（Web cache），代理服务器（proxy server）。可以大大减少对客户请求的响应时间，并且可以大大减少一个机构的接入链路到因特网的通信量。

条件 GET 方法，如果 1. 请求报文使用 GET 方法 2. 请求报文中包含一个 If-Modified-Since 首部行，那么这个报文就是一个条件 GET 请求报文。条件 GET 报文是 Web 代理缓存向服务器发起的，如果在该日期之后没有被修改，服务器不会在响应报文中包含该对象

2.3 因特网中的电子邮件

因特网电子邮件的三大部分：用户代理（user agent）、邮件服务器（mail server）、简单邮件传输协议（SMTP）

SMTP 基于 TCP，报文体部分只能使用 7比特 ASCII 表示。而使用HTTP传送前不需要将多媒体数据编码为 ASCII 码

SMTP 不使用中间邮件服务器发送邮件，如果没有发送成功，那么邮件就会停留在发送方的邮件服务器当中，等待新的尝试。

在 SMTP 握手的阶段，SMTP客户指示发送方的邮件地址和接收方的邮件地址。且报文的发送是持续性的，建立在相同的TCP 之上。

对于 SMTP 的每个报文，客户用一个新的 MAIL FROM 开始，用一个独立的句点指示该邮件的结束，即以 CRLF . CRLF 指示报文的结束，并且仅当所有邮件发送完成之后才会发送 QUIT。

SMTP 与 HTTP 之间的对比：

- HTTP 主要是一个 拉协议（pull protocol），而 SMTP 是一个推协议（push protocol）
- SMTP 要求每个报文采用7比特 ASCII 码格式，而 HTTP 数据不受这种限制
- HTTP 把每个对象都封装到它自己的 HTTP 相应报文当中，即一个对象对应一个报文，而 SMTP 则把所有报文对象放在一个报文当中

SMTP 报文格式：首部行 + 空白行 + 报文体

首部行中的 FROM: 和 TO: 不同于之前的 SMTP 命令，前者是 SMTP 报文的一部分，而后者是 SMTP 握手协议的一部分。

邮件发送过程：发送方代理 ->（SMTP） 发送方邮件服务器 ->(SMTP) 接收方邮件服务器 ->（POP3、IMAP 或者 HTTP） 接收方用户代理

POP3 工作流程：特许（authorization）(通过用户名以及口令鉴别用户)、事务处理、更新（出现在quit之后，结束POP会话并且删除在事务处理中标记为删除的报文）

下载并删除与下载并保留的区别在于用户下载邮件之后在其他客户端是否仍然可以访问该邮件。

在用户代理与邮件服务器之间的POP3会话期间，POP3服务器会保留状态信息，但是并不会在会话期间携带状态信息

POP3 协议没有给用户提供任何创建远程文件夹并为报文指派文件夹的方法，而 IMAP 有在远程创建文件夹以及将邮件从一个文件夹移动到另一个文件夹的命令。IMAP 另一个重要的特性在于允许用户代理处理获取报文某些部分的命令。

用户代理从邮件服务器获取邮件还可以通过 HTTP 协议实现。但是，邮件服务器在于其他的邮件服务器之间发送和接收邮件时，仍然使用的是 SMTP

2.4 DNS: 因特网的目录服务

DNS 协议建立在 UDP 之上，使用 53 号端口

DNS 服务：

- 主机名到 IP 地址的转换
- 主机别名（host aliasing），获得规范主机名（canonical hostname）
- 邮件服务器别名（mail server aliasing）
- 负载分配（load distribution）

DNS 服务器集中式设计带来的问题：

- 单点故障（a single point of failure）
- 通信容量（traffic volume）
- 远距离的集中式数据库（distant centralized database）
- 维护（maintenance）

三中类型的 DNS 服务器：根 DNS 服务器、顶级域（TLD）服务器、权威DNS服务器

本地DNS 不属于服务器的层次结构，每个 ISP 都有一台本地的 DNS 服务器

递归查询（recursive query）和迭代查询（iterative query）

从请求主机到本地 DNS 服务器的查询是递归的，其余的查询是迭代的。

资源记录是四元组（Name, Value, Type, TTL），TTL 是该记录的生存时间，决定了资源记录应当从缓存中删除的时间

获得邮件服务器的规范主机名使用 MX，获得其他服务器的规范主机名使用 CNAME

一个主机名能够有多个 IP的原因？在DNS 回答报文中的回答区域中可以包含多条RR。

2.5 P2P 文件分发

参与一个特定文件分发的所有对等方的集合被称为一个洪流（Torrent）

最稀缺优先（rarest first）：针对一个对等方没有的块在它的邻居中决定最稀缺的块，并且首先请求那些最稀缺的块。目标是均衡每个块在洪流中的副本数量。

一报还一报（tit-for-tat）：10秒计算一次最高速率流入的4个邻居，30秒随机选择一个邻居向其发送块。

2.6 视频流和内容分发网

对流式视频的最为重要的性能度量是平均端到端吞吐量

经 HTTP 的动态适应性流（Dynamic Adaptive Streaming over HTTP, **DASH**）,将视频编码成几个不同的版本，每个版本具有不同的比特率，对应于不同的质量水平。客户使用 HTTP GET 请求一次选择一个不同的块。客户首先请求的是告示文件，得知各种各样的版本。

CDN 服务器安置原则

- 深入（靠近端用户，改善用户感受时延和吞吐量）
- 邀请做客（将集群放在IXP中）

在请求的时候，权威服务器会将DNS请求移交给对应的CDN进行处理，然后LDNS会请求对应 的 CDN

集群选择策略：

- 地理上最为邻近（geographically closest）
- 实时测量（real-time measurement）

2.7 套接字编程：生成网络应用

第三章：运输层

3.1 概述和运输层服务

运输层是在端系统中而不是在路由器中实现的

网络层提供不同主机之间的通信，而运输层提供不同主机之间进程的逻辑通信.

3.2 多路复用和多路分解

多路复用与多路分解：将网络层提供的主机到主机交付服务延伸到主机上的应用程序提供进程到进程的交付服务

多路复用（multiplexing）：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，从而生成报文段（segment），然后将报文段传递到网络层。要求是：1. 套接字有唯一标识符。 2. 每个报文段有特殊字段（源端口号字段和目的端口号字段）来指示该报文段所要交付到的套接字。

多路分解（demultiplexing）: 将运输层报文段中的数据交付到正确的套接字的工作。

一个 UDP 套接字是由包含目的 IP 地址 和一个目的端口号的二元组标识的。

TCP 套接字是由一个四元组（源 IP 地址，源端口号，目的 IP 地址，目的端口号）来标识的。

3.3 无连接运输：UDP

使用 UDP 的应用是可能实现可靠数据传输的，可以通过在应用程序自身中建立可靠性机制来完成。

UDP 报文头部仅仅包括 源端口号、目的端口号（用于定位套接字来执行分解功能）、长度字段（指示UDP报文段中的字节数（首部 + 数据））以及检验和字段，总共8字节。

UDP 检验和，发送方的 UDP 对报文段中的所有 16 比特字的和进行反码运算，求和遇到的任何溢出都要进行回卷，最后对结果取反码。接收方认证的时候将所有结果相加，如果和不是全1，则代表出错。

UDP 提供了差错检测，但是不能差错恢复。

检验和作用：检测在一个传输分组中的比特错误

3.4 可靠数据传输原理

自动重传请求协议（ARQ）处理比特差错情况：1. 差错检测 2. 接收方反馈（ACK, NAK） 3. 重传

停等协议（stop-and-wait）：在发送方确信接收方已正确接收分组之前，发送方不会发送新数据这样的协议

解决流水线差错的办法：回退N步（GBN）以及选择重传（SR）

对 GBN （滑动窗口）协议，限制发送但未确认的分组数量，为了进行流量控制。

在 GBN 协议当中，对序号为 n 的分组的确认采取累积确认（cumulative acknowledgement）的方式，也就是说，表明接收方已经正确接收到序号为 n 的以前且包括n在内的所有分组。

定时器：用于超时重传一个分组。

3.5 面向连接的运输：TCP

TCP 是面向连接的（connection-oriented）、全双工（full-duplex service）的、点对点的（point-to-point）

TCP 从缓存中取出并放入报文段的数据数量受限于最大报文段长度（MSS），MSS 需要根据最大链路层帧长度（最大传输单元（MTU））来设置。设置 MSS 要保证一个TCP报文段加上 TCP/IP 首部长度将适合单个链路层帧。MSS 是指在报文段里应用层数据的最大长度，而不是指包括首部的TCP报文段的最大长度。

TCP 报文字段：源端口号，目的端口号，检验和，序号，确认号，接收端窗口（用于流量控制），首部长度，选项，标志。

报文段序号(sequence number)：TCP 把数据看成无结构、有序的字节流。隐式地给数据流每一个字节编号，根据 最大报文段长度（MSS）给每一个报文段填充首字节的序号作为报文段的序号。

报文段确认号（acknowledgement number）：是接收主机希望从发送主机收到的下一字节的序号。

捎带（piggybacked）：对数据的确认被装载在一个承载服务器到客户的数据的报文段中。

TCP 决不为已被重传的报文段计算样本 RTT，仅为传输一次的报文段测量样本RTT。

$$TimeoutInterval = EstimatedRTT + 4 * DevRTT$$

接收端回复的 ACK 是 发送方 $Seq + len(data)$ 的值

超时间隔加倍：每次 TCP 重传时都会将下一次的超时间隔设为先前值的两倍。

接收方会在比期望序号大的失序报文段到达并且检测出了间隔之后，发送冗余 ACK，只是下一个期待字节的序号

一旦受到 3 个冗余 ACK，TCP 就执行快速重传（fast retransmit），即在该报文段的定时器过期之前重传丢失的报文段。

TCP 确认是累积式的，正确接受但是失序的报文段不会被接收方逐个确认，和 SR 不同。但是接收方会缓存起来，这点和 GBN 不接受直接丢弃有显著差异。并且 TCP只会重传至多一个报文。

流量控制（flow-control）：发送方维护接收窗口（receive window）来表示接收方还有多少可用的缓存空间。

发送方在整个连接生命周期需要保证：$LastByteSent - LastByteAcked \leq rwnd$

当接收方的接收窗口为 0 时，发送方还将继续发送只有一个字节数据的报文段。接收方清空缓存之后再返回含有非零接收窗口值的确认报文。

TCP 连接建立过程（three-way handshake）：

1. 客户端 TCP 向服务器端 TCP 发送不含有应用层数据的 SYN 报文段，并且随机选择初始序号放入序号 字段当中。
2. 服务器端 TCP 接收到 SYN 报文之后，为该 TCP 连接分配 TCP 缓存和变量，将 SYN 依然置 1，确认号字段 为 客户端的初始序号 + 1，选择自己的初始序号，成为 SYNACK 报文段
3. 接收到 SYNACK 报文段之后，客户端也要给该连接分配缓存和变量，SYN 置为0，确认号为服务器的初始序号 + 1，并且可以携带客户到服务器的数据。

两次握手无法确认客户端正常，且无法区分历史连接。

四次挥手：客户端向服务器端发送 FIN 置为1的报文段，服务器先发送确认报文段，紧接着发送 FIN 置为 1 的报文段，最后客户端进行确认，最终释放该连接的所有资源。

3.6 拥塞控制原理

网络拥塞代价：

1. 会有丢包现象，数据会有损失
2. 发送方必须执行重传以补偿因为缓存溢出而丢弃的分组
3. 发送方在遇到大时延时所进行的不必要重传会引起路由器利用其链路带宽来转发不必要的分组副本。
4. 当一个分组沿着一条路径被丢弃时，每个上游路由器用于转发给分组到丢弃该分组而使用的传输容量最终被浪费掉了。

拥塞控制方法：

- 端到端拥塞控制：网络层没有为运输层拥塞控制提供显式支持。通过TCP 报文段的丢失来指示网络拥塞。
- 网络辅助的拥塞控制：路由器向发送方提供关于网络中拥塞状态的显式反馈信息。

3.7 TCP 拥塞控制（congestion control)

运行在发送端的 TCP 拥塞控制机制维护拥塞窗口（congestion window），对一个 TCP 发送方能向网络中发送流量的速率进行了限制，在一个发送方中未被确认的数据量不会超过接收窗口与拥塞窗口的最小值。

TCP 的自计时（self-clocking）：TCP 使用确认来增大它的拥塞窗口长度

TCP 拥塞控制算法

1. 慢启动（slow-start）：拥塞窗口的大小以 1 个 MSS开始并且每当传输的报文段首次被确认就增加一个 MSS，发送速率以指数级增长。有三种方法会停止指数级增长：
   - 超时指示的丢包事件发生，则判断遇到拥塞，此时cwdn 设置为 1，并且将慢启动阈值设置为减半前 cwnd / 2。
   - cwnd 值等于 慢启动阈值，此时进入拥塞避免模式
   - 接收到三个冗余 ACK，则快速重传并且进入快速恢复状态。
2. 拥塞避免：进入拥塞避免状态之后，每过一个 RTT，拥塞窗口的大小才会增加一个 MSS，类似于线性增长。有两种情况结束线性增长：
   - 超时： cwnd 设置为 1MSS，慢启动阈值更新为cwnd减半前的一半
   - 三个冗余 ACK：cwnd 减半，慢启动阈值更新为cwnd减半前的一半，进入快速恢复状态
3. 快速恢复：对于引起 TCP进入快速恢复状态的缺失报文段，对收到的每个冗余 ACK，拥塞窗口的值增加一个 MSS。

TCP 的拥塞控制是，每个 RTT 内拥塞控制窗口线性增加 1 MSS，然后出现 3 个冗余 ACK 事件时拥塞控制窗口减半。加性增、乘性减（AIMD）

TCP 趋于在竞争的多条 TCP 连接之间提供对一段瓶颈链路带宽的平等分享。

明确拥塞通告（ECN）：允许网络明确向 TCP 发送方和接收方发出拥塞信号。

3.8 小结

最简单的运输层协议就是在网络层的基础上提供了多路复用以及多路分解

若网络层协议不能向运输层报文段提供时延或者带宽保证，那么运输层协议就不能向进程间发送的报文提供时延或者带宽保证。

可以通过提供 确认、定时器、重传以及序号机制来完成可靠数据传输。

第四章：网络层：数据平面

4.1 网络层概述

计算题汇总

延迟计算（时延）

P2P & CS 模式的文件分发计算

u_s表示服务器接入链路的上载速率，u_i表示第i对等方接入链路的上载速率，d_i表示了第i对等方接入链路的下载速率，F表示被分发的文件长度(以比特计算)，N表示要获得的该文件副本的对等方的数量

CS模式的文件分发时间

- 对服务器来说，分发该文件的时间是 NF/u_s
- 对对等方来说，用 d_{min} 表示具有最小下载速率的对等方的下载速率，即 $d_{min} = min\{d_1, d_p, \dots, d_N\}$， 因此最小分发时间至少为 F/d_{min}

  综上得到 CS 模式文件分发时间 $D_{cs} \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{min}}\}$

P2P模式的文件分发时间

- 服务器只需要发送一次文件副本，所以用的时间 F/u_s
- 和 CS 模式一样，对于对等方来说，最小分发时间至少为 F/d_{min}
- 与 CS 模式不一样的是，P2P 模式下所有文件的传输由整个系统的上载时间承担，故时间为 $NF/u_s + \sum_{i=1}^{N} u_i$

  综上得到 P2P 模式下文件分发时间 $D_{P2P} \geq \max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s+\sum_{i=1}^{N} u_i}\}$

检验和计算

UDP 检验和，发送方的 UDP 对报文段中的所有 16 比特字的和进行反码运算，求和遇到的任何溢出都要进行回卷，最后对结果取反码。接收方认证的时候将所有结果相加，如果和不是全1，则代表出错。