计算机网络

计科210X 甘晴void 202108010XXX

期中复习

第一章：分组交换和电路交换的比较

应用层：要看懂HTTP报文，知道cookie，持续连接非持续连接区别，邮件工作的过程，DNS 原理，后面的没空可以不看了

运输层： UDP和TCP比较，可靠数据传输的rdt不需要知道那么仔细，知道SR和GBN就可以，TCP要知道报文确认号序号，流量控制和拥塞控制有什么区别，拥塞控制算法

1计算机网络和因特网

1.1 因特网

主机，端系统，通信链路，分组交换机，传输速率bps，分组，路由器，链路层交换机，路径（一个分组经过的链路和分组交换机）

因特网服务提供商ISP（InternetServiceProvider），传输控制协议TCP（TransmissionControlProtocol），网际协议IP（InternetProtocol），请求评论（RFC）

互联网是什么？节点和边，通过网络互联设备连接起来

分布式应用以及为它提供通信服务的基础设施

分布式应用程序，套接字接口

1.2 网络边缘

客户（client），服务器（server），

CS模式，P2P模式

接入网，

采用网络设施的面向连接服务

1.3 网络核心

电路交换：频分复用FDM，时分复用TDM（独享）

分组交换（将长报文划分为较小的数据块）：虚电路网络，数据报网络

（电路交换会预留资源，分组交换不会）

分组交换机

存储转发传输（交换机开始向输出链路传输前必须先收到整个分组）：d(端到端)=NL/R,N表示中间路径数

★分组交换优点：共享性；坏处：丢包/时延

★分组交换与电路交换的对比：

- 提供了比电路交换更好的带宽共享
- 比电路交换更简单有效，实现成本更低

包含计算

1.4 分组交换的时延/丢包和吞吐量

节点总时延=传输时延+传播时延+排队时延+处理时延【P25】

处理时延：微秒或更低

排队时延：毫秒到微秒

传输时延：分组推向链路，L/R，分组长度/链路带宽，毫秒到微秒

传播时延：链路到被接收，d/s，物理链路长度/传播速度，毫秒量级

车队类比

流量强度I=La/R,，所有分组由Lbit组成，R为传输速率，a为到达分组的平均速率

流量强度I不能大于1，I接近1时，平均排队时延趋向于无穷大

吞吐量，取决于瓶颈链路的传输速率

1.5 协议层次与服务模型

协议栈，5个层次，应用层（报文），运输层（报文段），网络层（数据报），链路层（帧），物理层

应用层：HTTP,SMTP,FTP,DNS等；分组：报文

运输层：TCP,UDP（可靠性，流量控制，拥塞控制）；分组：报文段

网络层：IP；分组：数据报

服务

原语

服务访问点SAP

2 应用层原理

2.1 应用层协议原理

2.1.1 应用程序体系结构

客户-服务器体系结构，P2P体系结构（自扩展性）

2.1.2 数据通信

2.1.3 可供应用程序使用的运输服务

可靠数据传输，吞吐量（带宽敏感应用<>弹性应用），定时，安全性

2.1.4 因特网提供的运输服务

TCP：面向连接的服务，可靠的数据传送服务，拥塞控制机制

UDP：

安全套接字SSL（SecureSocketLayer）

2.2 Web和Http

2.2.1 HTTP

超文本传输协议（HyperTextTransferProtocol）

HTTP，建立在TCP上，默认端口80，无状态协议

2.2.2 非持续连接和持续连接

非持续连接，持续连接（所有请求响应经一个单独TCP连接发送）

往返时间（RTT,RoundTripTime），总响应时间=2*RTT+传输HTML文件的时间

2.2.3 HTTP报文格式

HTTP请求报文：

```
GET /somdir/page.html HTTP/1.1  #方法，URL和HTTP版本字段
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

方法包括：GET,POST,HEAD,PUT,DELETE

HTTP响应报文：

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Gug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Gug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data ......)
```

状态码包括

- 200 OK
- 301 Moved Permanently
- 400 Bad Request
- 404 Not Found
- 505 HTTP Version Not Supported

2.2.4 Cookie

HTTP是一个无状态协议，服务器不保存用户的信息。使用cookie可以让服务器标识一个客户，提供服务。cookie有以下4个组件：

- 响应报文中的一个cookie首部行
- 请求报文中的一个cookie首部行
- 用户端系统中保留一个cookie文件，由浏览器管理
- 位于Web站点的一个数据库，记录用户信息

2.2.5 Web缓存

Web缓存器（WebCache）也叫代理服务器（proxy server）

优先请求本地代理服务器，若得不到满足再请求

能够代表初始Web服务器满足HTTP请求。Web缓存器可以在存储器空间中保留最近请求过的对象的副本。使用Web缓存器可以大大减少对客户请求的响应时间，还能够大大减少一个机构的接入链路到因特网的通信量。

Web缓存器通常由ISP购买和安装

Web缓存器既是服务器又是客户

Web缓存器带来的一个问题是存放在缓存器中的副本可能不是最新的，因此需要有方式去证实请求的对象是最新的。解决：使用条件GET方法，在首部行中添加"If-Modified-Since"，这样只在指定日期后对象被修改过，才发送该对象。返回304NotModified表示未修改

2.2.* FTP

默认端口21

server主动向client的20号端口建立连接。

有状态协议

2.3 因特网中的电子邮件

默认端口 25

三个重要组成部分：用户代理，邮件服务器，简单邮件传输协议（SMTP,SimpleMailTransferProtocol）

过程：用户代理---->邮件服务器----SMTP---->邮件服务器---->用户代理

SMTP（推协议）

SMTP是一个推协议，由发送邮件的服务器将文件发给接收邮件的服务器，接收邮件的用户是不能使用该协议获取接收到的邮件的。获取接收到的邮件使用的是邮件访问协议。流行的邮件访问协议有：POP3,IMAP,HTTP。

POP3（拉协议）

POP3协议简单，但功能有限。用户代理打开与邮件服务器的TCP连接后，POP3开始工作。POP3工作有三个阶段：

- 特许：用户代理发送用户名和口令以鉴别用户。
- 事务处理：用户代理取回报文，对报文进行删除标记，取消删除标记，获取邮件的统计信息。获取报文有以下两种方式：下载并保留 || 下载并删除
- 更新：结束POP3会话，删除被标记为删除的报文。

IMAP（拉协议）

使用POP3协议只能获取报文，不能在服务器上创建文件夹对报文进行管理。为了解决这个或一些其他问题，产生了IMAP协议。IMAP服务器把每个报文与一个文件夹联系起来，报文到达时与INBOX文件夹相关联，而收件人能够把邮件移到一个新的文件夹中，阅读，删除或是移动到别的文件夹。IMAP还提供了查询邮件的命令，且IMAP维护了IMAP会话的用户状态信息(文件夹的名字等)。另外，IMAP还允许用户代理获取报文某些部分的命令。这样当用户处于低带宽连接时，可以选择性的只取回(MIME)报文的一部分。

基于Web的电子邮件

今天，基于Web的邮件已经非常常见了。这种电子邮件，用户代理就是浏览器，用户和远程邮箱之间的通信通过HTTP进行。用户发送和获取电子邮件时，都通过HTTP协议在浏览器和邮件服务器之间进行报文传输，但是邮件服务器之间发送和接收报文仍然使用SMTP。

用户代理--HTTP-->邮件服务器--SMTP-->邮件服务器--HTTP-->用户代理

2.4 DNS：因特网的目录服务

4.1 DNS概述与DNS服务

因特网上的主机可以使用多种方式标识。一种易于记忆的方式是主机名，而路由器所需要的标识则是更为具体的IP地址。将主机名转换到IP地址，就是域名系统DNS的任务。

DNS是由分层的DNS服务器实现的分布式数据库。

DNS是使主机能够查询分布式数据库的应用层协议。

（期中考试20分）DNS协议运行在★★★UDP★★★上，使用53端口。

（期中考试10分）A,NS,CNAME,MX分别是什么

除了进行主机名到IP地址的转换，DNS还提供了以下服务：

主机别名：一个主机除了规范主机名外，还可以有其他别名。通过DNS可以获取别名对应的规范主机名和IP地址。

邮件服务器别名：和主机别名相同，邮件服务器也可以使用别名，通过DNS获取规范主机名和IP地址。

负载分配：一个站点可能会被冗余分布在多台服务器上，有不同的IP地址。这些IP地址构成了一个IP地址集，DNS服务器可以在返回这些IP地址时循环改变次序，客户通常选择先返回的IP地址请求服务，这样通过更改返回IP地址的顺序，就能起到负载分配的作用。

4.2 DNS工作原理

使用单个DNS服务器有单点故障、通信容量，远距离集中(高时延)，维护等问题，因此DNS采用了分布式的设计方案。按照层次，DNS服务器分为三种：根DNS服务器；顶级域服务器；权威DNS服务器。除了以上三个层次外，还有一类重要的DNS服务器是本地DNS服务器。本地DNS服务器一般离主机很近，起到代理的作用。在进行DNS查询时，有两种查询方式：递归查询和迭代查询。通常，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。

为了改善时延性能和减少报文传输，DNS还广泛使用了缓存技术。DNS服务器会缓存主机名/IP地址对，一段时间后才丢弃缓存信息。

2.5 P2P文件分发

【计算题P95】两个公式

使用P2P体系结构的网络应用对总是打开的基础设施服务器有最小的依赖。P2P体系结构具有内在自扩展性，无论有多少对等方，P2P体系结构的应用的文件分发时间都小于客户-服务器体系结构的应用。BitToreent是一种用于文件分发的流行P2P协议，以下是BitTorrent的部分功能原理。

在BitTorrent协议中，参与一个文件分发的所有对等方的集合称为一个洪流。洪流中的对等方互相下载等长度的文件块(典型长度256KB)。对等方刚进入洪流时没有块，一段时间后就持有了块，并可为其他对等方上载块。每个洪流有一个追踪器，当对等方加入洪流时，将向追踪器注册自己，并周期性通知追踪器自己仍在洪流中。

当一个对等方A加入洪流后，追踪器会给出一个对等方子集及这些对等方的IP地址，A可以从中选择临近的对等方建立TCP连接，交换块列表。选择向邻居请求哪些块使用的策略是最稀缺优先，首先请求邻居中副本最少的块，从而加快这些块的分发。而响应邻居的哪些请求则使用的是一种对换算法，优先响应前四个给A提供数据最高速率的邻居(他们被称为疏通)，并且还会每30s随机选择一个对等方B，如果B的速率够高，就把B换进前四位列表。这样对等方能够趋向于找到彼此的协调的速率上载。除了这五个对等方外，其他相邻的对等方不会收到A的块。这种激励机制被称为“一报还一报”。

2.6 视频流和内容分发网

6.1 视频流

视频是一系列的图像，可以被压缩，通常用比特率来衡量质量。视频可以被压缩到不同的比特率，让用户根据网络带宽来选择观看的版本。对流式视频最重要的性能度量是平均端到端吞吐量，流式视频应用得到的平均吞吐量至少与压缩视频的比特率一样大。

在HTTP流中，视频只是一个普通的文件。用户请求视频文件，将收到的字节进行缓存，一旦超过了预先设定的门限就开始播放，将字节处理成帧，并将这些帧解压缩展现在视频上。视频编码为不同比特率的版本经HTTP传输，被称为经HTTP的动态适应流(DASH)。使用DASH后，视频有不同版本存放在服务器中，服务器会有一个告示文件，提供每个版本的URL和比特率信息。客户可以根据可用带宽指定URL和一个字节范围，对视频数据块请求。

6.2 内容分发网

因特网视频公式每天都需要向百万计的用户发送数据，如果在数据中心存储所有视频，会产生很大的问题。主要有三个问题：1.如果用户离数据中心太远，很可能产生停滞时延。2.流行的视频可能经相同链路发送多次，浪费带宽。3.单个数据中心出现单点故障，就不能分发视频流了。为了解决分发视频数据的问题，，几乎所有的视频流公司都使用内容分发网CDN。CDN管理分布在多个位置的服务器，将用户请求定位到能提供最好服务的CDN位置。CDN包括专用CDN(内容提供商自己的)和第三方CDN(代表多个内容提供商分发内容)。

CDN通常集群部署。每个集群只保留一些视频，当用户请求视频时，如果集群中没有该视频，再从其他集群或中心仓库拉取视频到集群，并在本地存储一个副本。CDN的部署通常采用两种原则：

深入：在全球接入ISP中部署服务器集群，靠近端用户。

邀请做客：在少量关键的位置建造大集群，邀请ISP做客。

使用CDN，用户的请求需要被重定向到CDN服务器，这通常是由DNS进行截获和重定向的。DNS检测到URL中有video以及内容提供商的名字，就返回一个CDN域的主机名，用户将再次发送请求到这个CDN域的DNS系统，并得到CDN节点的IP地址。CDN的集群选择策略是CDN部署的核心，一种简单的策略是指派用户到地理上最为临近的集群，也有周期性实时测量集群和用户到集群时延和丢包性能来选择集群的策略。

2.7 套接字编程

网络应用程序有两类，一种使用协议标准(如RFC)定义的操作实现的，另一种是使用专用的应用层协议。如果是开发专用的网络应用程序，应该避免使用熟知端口号。另外一个实现网络应用程序的问题是，该选择TCP还是UDP。这两种协议有各自的特点。以下是分别使用UDP和TCP实现的简单客户-服务器程序(Python实现)。

7.1 UDP套接字编程

```
# Client
from socket import *

def main():
    serverName = '127.0.0.1'
#本机IP
    serverPort = 12000
#端口
    clientSocket = socket(AF_INET, SOCK_DGRAM)
#IPV4,UDP
    message = input("Input lowercase sentence:")
    clientSocket.sendto(message.encode(),
(serverName,serverPort))
    modifiedMessage,serverAddress = clientSocket.recvfrom(2048)
    print(modifiedMessage.decode())
    clientSocket.close()
    return

if __name__ == "__main__":
    main()

# Server
from socket import *

def main():
    serverPort = 12000
    serverSocket = socket(AF_INET,SOCK_DGRAM)
```

```
    serverSocket.bind(('',serverPort))
    message,clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),clientAddress)

if __name__ == '__main__':
    main()
```

7.2 TCP套接字编程

```
# Client
from socket import *

def main():
    serverName = '127.0.0.1'
    serverPort = 12000
    clientSocket = socket(AF_INET,SOCK_STREAM)
#IPv4,TCP
    clientSocket.connect((serverName,serverPort))
#建立连接
    sentence = input("Input lowercase sentence:")
    clientSocket.send(sentence.encode())
#已建立连接，不需要地址了
    modifiedSentence = clientSocket.recv(2048)
    print("From Server:",modifiedSentence.decode())
    clientSocket.close()

if __name__ == '__main__':
    main()
# Server
from socket import *

def main():
    serverPort = 12000
    serverSocket = socket(AF_INET,SOCK_STREAM)
    serverSocket.bind(('',serverPort))
    serverSocket.listen(1)
#欢迎套接字，等待建立连接
    print("The server is ready to receive")
    while(1):
```

```
27          connectionSocket,addr = serverSocket.accept()
    #客户专用套接字
28          sentence = connectionSocket.recv(2048).decode()
29          modifiedSentence = sentence.upper()
30          connectionSocket.send(modifiedSentence.encode())
31          connectionSocket.close()
32
33  if __name__ == "__main__":
34      main()
```

3 传输层

3.1 概述和运输层服务

运输层协议为运行在不同主机上的应用进程之间提供了逻辑通信功能。运输层协议是在端系统实现的，将应用程序的报文转换为较小的块，加上运输层首部生成运输层分组，称为报文段，将报文段交给网络层，由网络层发送到目的地。

因特网提供了两种可用的运输层协议：UDP(用户数据报协议)，TCP(传输控制协议)。UDP提供不可靠服务，而TCP提供面向连接的可靠数据传输，还提供拥塞控制。网络层的协议即网际协议IP，提供的是不可靠服务，不确保报文段的交付和按序交付，因此本章主要讨论的一个问题是TCP如何提供可靠数据传输，另一个主要讨论的问题是TCP如何实现拥塞控制。

TCP:拥塞控制，流量控制，建立连接

UDP:没有在IP之上提供更多服务（IP主机到主机，UDP进程到进程）

都不提供延时与带宽保证，数据交付和交错检查（UDP仅能提供的）

3.2 多路复用与多路分解

运输层从紧邻其下的网络层接收报文段。运输层负责将报文段进程交付给主机上运行的适当进程。一个进程有一个或多个套接字，运输层实际上是将数据交给中间的套接字。接收主机可能同时有多个套接字，因此每个套接字都有一个唯一的标识符。为了将运输层报文段定向到合适的套接字，运输层报文段有一些字段，包含了套接字的标识符。

多路分解：（运输层→套接字，向上）通过这些字段，将运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。

多路复用：（套接字→网络层，向下）从不同套接字中收集数据块，将数据块封装并添加首部信息产生报文段，发送到网络层的过程工作称为多路复用。

每个套接字都有唯一的标识符，每个报文段都有特殊字段来指示该报文要交付到的套接字。套接字需要使用端口号进行标识。端口号是一个16bit的数，其中0-1023为熟知端口号。保留给HTTP,FTP等熟知的应用层协议。

UDP套接字：二元组标识，包含一个目的IP地址(标识主机)和一个目的端口号(标识具体的套接字)。接收主机的运输层收到报文段后，通过首部中的目的端口号，将报文段定向到相应的套接字。

TCP套接字：四元组(源IP地址，源端口号，目的IP地址，目的端口号)标识。这四个值标识了一个连接，接收主机通过四个值将报文段定向到对应套接字。由于TCP是面向连接的，不同的源IP地址和源端口号与同一个目的IP的同个目的端口号建立的是不同的连接，使用的也是不同的套接字。但是初始创建连接时，连接还没有建立，因此使用的是同一个套接字(欢迎套接字)。连接建立后，接收端进程将创建一个新的套接字供该连接使用。

3.3 无连接运输UDP

UDP是无连接的运输，只在IP的基础上进行复用和分解，并增加了少量的差错检测。UDP不能提供可靠的传输服务，但有以下优点：

- 控制更加精细。采用UDP时，UDP会将数据打包直接传递给网络层，不像TCP存在拥塞控制机制，发送可能受到遏制。
- 无需建立连接。UDP不会引入建立连接的时延。因此DNS就是运行在UDP之上的。
- 无连接状态。UDP不维护连接状态。而TCP则需要维护包括接收和发送缓存，拥塞控制参数和确认号等连接状态。
- 分组首部开销小。仅有8字节。

除了不能提供可靠传输，UDP还可能导致其他问题，因为缺乏拥塞控制，UDP可能导致发送方接收方之间的高丢包率，挤垮TCP会话。

UDP报文段由源端口号，目的端口号，长度和校验和组成，每个字段2字节，共8个字节。其中的校验和提供了差错检测功能，因为链路层协议可能没有提供差错检测，而报文段的传输可能经过一条没有使用差错检测协议的链路。因此UDP在端到端基础上，在运输层提供差错检测，这称为端到端原则。

校验：

发送方：对报文段中所有16比特字的和（任何时候若溢出要回滚），进行反码运算。

接收方：若无差错，和将是0xFFFF

3.4 可靠数据传输原理

可靠数据传输协议（ReliableDataTransferProtocol）

可靠传输问题不仅在运输层出现，也在链路层及应用层出现。实现可靠传输服务是可靠数据传输协议的任务。一般情况下，都是在下层协议提供不可靠数据传输(udt)下，建立可靠数据传输协议。由于可靠数据传输(rdt)不仅适用于运输层，收发端交换的数据以下称为分组，而不是运输层的报文段。

3.4.1 构造可靠数据传输协议（**rdt1.0,2.0,2.1,2.2,3.0**）

①经完全可靠信道的可靠数据传输 rdt1.0

经完全可靠信道的可靠数据传输，接收与发送方不需要进行任何通信。发送方只需接收高层的数据，产生分组并发送到信道中，而接收方只需要从信道接收分组，从分组中取出数据交给上层。

②经具有比特差错信道的可靠数据传输 rdt2.0

实际情况下，底层信道的模型是会出现比特差错的模型，在分组的传输，传播或缓存的过程中都可能出现比特差错。为了处理差错，接收方需要对接收到的分组进行确认，发出肯定确认告知发送方分组被接收，发出否定确认告知发送方重发分组。基于这种重传机制的可靠数据传输协议称为自动重传协议(ARQ)。ARQ协议需要三种功能处理比特差错的情况：

- 差错检测：通过检验和字段，使接收方能够检测到比特差错。
- 接收方反馈：接收方需要进行肯定确认(ACK)，否定确认(NAK)，只需要一个bit的分组就可以进行确认。
- 重传：接收方接收到有差错的分组时，发送方重传分组。

采用以上这种重传机制，发送方在发送分组后，将等待接收方的确认分组，决定是否重传还是传输新的分组。由于这种行为，rdt2.0这样的协议被称为停等协议。

③rdt2.1

由于信道本身可能出现差错，因此必须考虑确认分组ACK或NAK出现差错的情况。在这种情况下，发送方不知道接收方是否正确收到了分组。解决该问题的方式是，如果发送分组不能确定接收方是否接收到了正确的分组，就重新发送分组。这种方式给信道引入了冗余分组。但是接收方还需要知道接收到的是新的分组还是重传的分组，因此对于分组需要进行一个序号，需要只需要为0或1，让接收方能够区别分组是否和上一个接收到的分组一样就可以。

④rdt2.2

用对前一分组的正向确认替代对当前分组的反向确认（用ACK0替代NAK1回应P1）

这个方法没有NAK状态，只有ACK。

⑤经具有比特差错的丢包信道的可靠数据传输 rdt3.0

除了比特差错，丢包也是常见的一种情况。解决丢包的方式和处理比特差错一样，如果没有接收到确认分组(发送的分组丢失或确认的ACK丢失，又或者只是分组或ACK延时)，就重传数据分组。新的问题是，到底等待多久才能判断分组丢失。这个时间至少大于一个往返时延，要根据情况设置。为了实现以上这种基于时间的重传机制，还需要一个倒计数定时器，发送方每发送一个分组就启动一个定时器，定时器超时就进行重传。解决冗余分组的方式则和rdt2.0相同。因为分组序号在0和1之间交替，rdt3.0也被称为是比特交替协议。

可以对抗丢失/出错

3.4.2 流水线可靠数据传输协议

以上所形成的以停等协议为核心的可靠数据传输协议是功能正确的，但是在性能方面存在大的问题，因为发送方的信道利用率太低了，大量时间都在等待确认分组。而确认分组至少需要一个往返时延RTT才能回到发送方，还要加上协议处理时间和中间路由器的时延。这个性能问题的解决办法就是不以停等方式运行，允许发送方发送多个分组。例如发送方可以发送三个分组后再等待确认，这样的方式被称为流水线。不过，在提升性能的同时，采用流水线也会带来新的问题。

必须增加序号范围，因为输送中的分组有多个。
发送方和接收方需要缓存多个分组。
需要处理丢失，损坏及延时过大的分组。两种基本方法是：回退N步；选择重传。

3.4.3 回退N步(GBN)

在回退N步协议GBN中，允许发送方发送多个分组，不需等待确认，但是未确认的分组数不能超过某个最大允许数N。定义基序号base为最早未确认分组的序号，下一个序号nextseqnum为下一个待发分组的序号，则：

GBN协议也常被称为滑动窗口协议。分组的序号在分组首部的字段(k位)，范围为0-2^k-1。
GBN发送方响应三种事件：

上层调用：窗口未满，则发送分组，否则告知上层。上层可能过一会儿再试，也可能发送方直接把上层数据缓存。
接收ACK：GBN协议中，对分组的确认方式为累积确认。接收到n的ACK，则序号小于等于n的分组都被确认接收。
超时：如果出现超时，重传所有已发送但未被确认过的分组。GBN协议中只有一个定时器，最早发送且未被确认的分组启动计时器，接收到确认分组则重启计时器。
GBN协议中，接收方会丢弃所有失序的分组，因为发送方会重传这些分组。直接丢弃这些分组，接收缓存简单，不需要缓存失序分组，接收方唯一需要维护的信息就是下一个按序接收的分组的序号。

3.4.4 选择重传(SR)

如果采用GBN协议，一旦最早发送的未确认分组超时，就需要从该分组开始重传，且接收方接收到的不按序的分组都会被丢掉。在信道差错率增加，时延很大的情况下，GBN可能大量重传分组。选择重传协议SR改进了这一点，只让发送方重传可能出错的分组。接收方将缓存失序的分组，并逐个对分组进行确认，连续的几个分组都收到后一起交付给上层。而发送方的每一个分组都有自己的定时器，超时后只重传一个分组，如果收到了最小未收到确认的分组的对应ACK，就将窗口重新移动到具有最小序号的未确认分组处。

对于已收到的那些分组，接收方会重新进行确认，这是有必要的，否则发送方可能停留在一个固定的窗口(因为不进行累积确认了，不重新确认窗口就不移动了)。

另外，考虑到序号是有限的，发送方与接收方窗口的不同步会导致严重的问题。例如，假设序号只有0,1,2，一种情况是正常的按照0,1,2,0来发送分组，另一种情况是分组0的ACK丢失了，发送次序是0,1,2,0(重传0)，因为有这两种不同的情况，接收方无法确认接收到的分组是重复发送的还是新的分组。因此必须限制窗口的大小，避免这种情况的发生。窗口长度必须小于或等于序号空间大小的一半。这样假设发送方没有收到正确的ACK，窗口不移动，接收方接收到了分组，一直向后移动，最后的结果就是发送方的窗口占了一半序号，接收方的窗口占了一半的序号，接收方的窗口最大值刚好停在发送方的窗口最小值前面，而不会发生序号重复。

3.5 面向连接的传输：TCP

面向连接，全双工（两方都可以同时收发），三次握手确定连接，通过连接发送数据

TCP连接建立后，应用进程就可以互相发送数据了。客户进程通过套接字将数据交给TCP，TCP将数据存储到连接的发送缓存中，接下来TCP会从发送缓存中取出数据交给网络层。TCP可取出并添加到报文段的数据大小受限于最大报文长度MSS，而MSS又通常根据最大链路层帧长度(最大传输单元MTU)来设置，MSS要保证一个报文段加上TCP/IP首部长度(40字节)适合链路层帧。以太网和PPP链路层协议都有1500字节的MTU，因此MSS的典型值为1460字节。TCP接收端接收到报文段，将数据放到接收缓存中，应用程序从该缓存中读取数据。

根据以上的讨论，TCP连接的组成包括：一台主机上的缓存，变量和与进程连接的套接字，另一台主机上的缓存，变量和与进程连接的套接字。

TCP报文结构

序号和确认号

发送的ACK是我期望从对方那边得到的期望序号

3.5.3 往返时间的估计与超时

- EstimatedRTT = (1-a) EstimatedRTT + a SampleRTT
- a推荐值取1/8
- DevRTT = (1-b) DevRTT + b | SampleRTT - EstimatedRTT |
- b推荐值0.25
- TimeoutInterval = EstimatedRTT + 4 DevRTT

可靠数据传输 使用单一重传定时器

超时间隔加倍

快速重传：监测到3个冗余ACK（期望序号比应该的ACK高）就执行快速重传，比超时重传要早

产生TCP的ACK的建议（书P163）

3.5.5 流量控制

接收窗口（receive window）由发送方维护

接受方接收到报文段后，将报文段存放到接收缓存当中，应用进程从中取出数据。如果应用程序读取的速度较慢，而发送方发送的数据太多太快，就会导致接收缓存溢出。因此TCP提供了流量控制服务，使发送方的发送速率与接收方应用程序的读取速率匹配。这与拥塞控制很相似，但是完全是出于不同的原因。

TCP通过让发送方维护一个称为接收窗口的变量来进行流量控制。接收窗口表示接收方可用缓存的大小。发送方需要维护两个变量：

lastbyteread：读出的最后一个字节
lastbyterecv：已到达接收缓存的最后一个字节
不允许接收缓存溢出，则lastbyterecv-lastbyteread<=rcvbuffer。则接收窗口为：

rwnd=rcvbuffer-[lastbyterecv-lastbyteread]

rwnd是一个动态的值。在起始时，rwnd=recvbuffer，此后接收主机会维护该值，并将其发给发送主机，而发送主机需要进行如下的控制：

lastbytesent−lastbyteacked<=rwnd

为了避免发送主机收到rwnd=0后被阻塞，如果接收窗口为0，发送主机将发送一个只有一个字节的报文段，接收主机将会回发确认，并在确认报文中包含一个新的非0rwnd值。

TCP提供了以上的流量控制服务，UDP是不提供这样的服务的。因此如果使用UDP，进程从缓存中读取报文段的速度不够快，缓存将会溢出，并丢失报文段。

3.5.6 TCP连接管理

TCP建立：三次握手

两次连接的问题：维持虚假半连接，接收旧数据

TCP结束：四次挥手

3.6 拥塞控制原理

流量控制<>拥塞控制，相同点：都是对发送方进行遏制

- 流量控制：发送方的发送速率与接收方的读取速率相匹配，以取消出发送方使接收方缓存溢出的可能性
- 拥塞控制：避免网络拥堵

3.6.1 拥塞原因与代价

通常出现拥塞有以下几种情况：

- （情况1：两个发送方和一台具有无穷大缓存的路由器）当发送速率超过吞吐量时，平均排队分组数不断增长，源与目的之间的时延也变为无穷大(假设不停发送，有无限大的缓存)。
- （情况2：两个发送方和一台具有有限缓存的路由器）当缓存已满，部分分组被丢弃，这将引起发送方的重传，此外，提前发生超时还可能导致发送方重传没有丢失的分组。
- （情况3：4个发送方和具有有限缓存的多台路由器及多跳路径）在有许多跳的情况下，不同的链路中不同连接的载荷不同。

从以上几种情况来看，由于拥塞丢弃分组产生了以下的代价：

- 分组的到达速率接近链路容量时，分组经历巨大的排队时延
- 发送方必须重传以补偿因为缓存溢出而丢弃的分组
- 发送方在遇到大时延时进行的不必要重传会引起路由器利用链路带宽来转发不必要的分组副本
- 一个分组沿一条路径被丢弃时，每个上游路由器用于转发该分组的传输容量最终被浪费掉了

3.6.2 拥塞控制方法

端到端拥塞控制：端系统自己来观察，网络层不提供信息

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

ATM ABR

RM(资源管理)信元：NI位（轻微拥塞指示），CI位（拥塞指示），ER字段（中间路由器将自己可提供带宽写入）

3.7 TCP拥塞控制

当出现了网络拥塞时，TCP应该降低发送速率，进行拥塞控制。实现这样的控制，有三个问题需要解决。

TCP发送方如何感知到与目的地之间的路径上存在拥塞，或是不存在拥塞
TCP如何限制其向连接上发送流量的速率
采用什么算法改变发送速率
下面逐个解决以上的问题。

1.拥塞检测与速率的确定

当发生拥塞时，路径上的路由器缓存溢出，就会引起丢包。对于发送方而言，丢包导致的是出现超时和收到3个冗余ACK。如果发生了这两种情况，就表示发生了拥塞，就应该减小发送速率。这就解决了拥塞检测的问题。在没有拥塞时，为了充分利用所有可用的带宽，TCP发送方还应该可以加快发送的速率。当TCP发送方接收到确认报文段ACK时，就认为一切顺利，网络不拥塞，可以增加发送速率。

2.限制速率

TCP采用了一个拥塞窗口cwnd来限制发送速率。注意这与流量控制相似但是不同。发送方未被确认的数据量不会超过cwnd与rwnd的最小值，即LastByteSent-LastByteAcked<=min{rwnd，cwnd}。由于此处仅探讨拥塞控制，先假设接收缓存无限大，这样发送速率就只与cwnd有关。通过cwnd的值，就可以调整向连接发送数据的速率了。

3.TCP拥塞控制算法

解决了上面两个问题，只要使用特定的算法控制cwnd就可以实现拥塞避免了。TCP拥塞控制算法包括三个部分：慢启动；拥塞避免；快恢复。

慢启动

- 起始时，cwnd的值以一个MSS开始，每当一个报文被确认，就增加一个MSS（对每个ACK都将cwnd增加一个MSS），故相当于*2
- 出现超时丢包：cwnd设置为1，慢启动阈值ssthresh值设置为检测到拥塞时的cwnd/2，重新开始慢启动。
- cwnd达到ssthresh：结束慢启动，进入拥塞避免模式

- 出现三个冗余ACK：ssthresh=cwnd/2，cwnd=ssthresh+3MMS，结束慢启动，进入快速恢复模式

拥塞避免

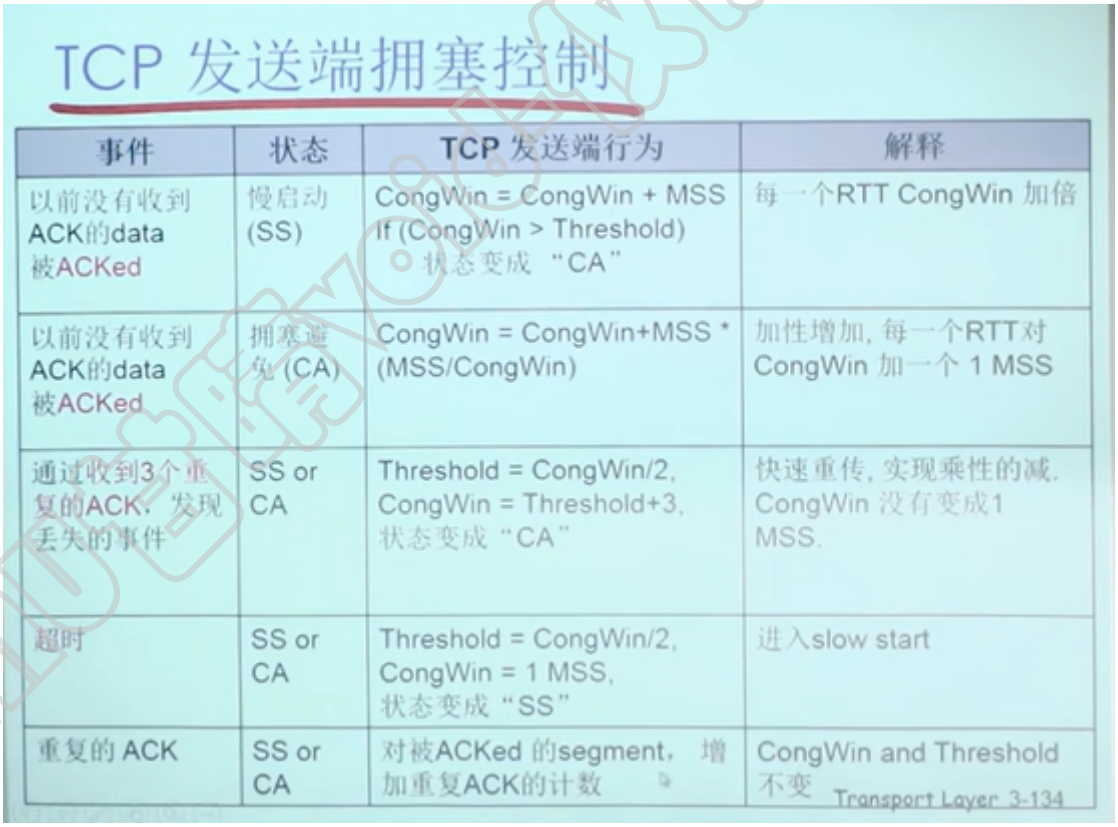
- 收到上一组全部cwnd的报文，增加一个MSS（线性加）
- 出现超时丢包：与慢启动的反应相同，返回并重新开始慢启动
- 出现三个冗余ACK：ssthresh=cwnd/2，cwnd=ssthresh+3MMS，结束拥塞避免，进入快速恢复模式

快速恢复

- 对于每个冗余的ACK，cwnd++，在对丢失报文段的ACK到达后，cwnd=ssthresh，进入拥塞避免模式
- 出现超时丢包，与慢启动的反应相同，返回并重新开始慢启动

TCP Tahoe（丢包全部按超时处理，进入慢启动）

TCP Reno(含快速恢复)



早期的TCP版本Tahoe没有快恢复机制，无论是收到冗余ACK还是丢包，都直接将cwnd设置为1。TCP的较新版本TCP Reno 则加入了快恢复机制。现在已有Reno的许多变种，例如Vegas，该算法在分组发生丢失之前检查源与目的地之间的拥塞，当检测出快要发生的分组丢失时(通过RTT的变化)，线性降低发送速率。

7.2 对TCP吞吐量的宏观描述

有了TCP的拥塞控制算法，可以考虑长期存活的TCP连接的吞吐量了。

当窗口长度是w，往返时间为RTT，TCP的发送速率大约是W/RTT，在一个窗口内发送了w后，收到前一半发送的分组的ACK，因此发送速率约为W/RTT。不考虑慢开始阶段(指数增长，很快结束)，发生丢包时，假设速率减半，再增加到W/RTT。在以上情况下，TCP的速率重复从W/2RTT到W/RTT的过程，在两个值之间线性增长，可以得出一个高度理想化的TCP稳态动态性模型：

一条连接的平均吞吐量 = 0.75 * W /RTT

在高带宽路径中，结合MSS和丢包率L，可以得出一条TCP连接的吞吐量公式：

一条连接的平均吞吐量 = 1.22 * MSS / (RTT * sqrt(L))

7.3 公平性

可以在吞吐量图上进行演示

如果有多条TCP连接，通过同一段瓶颈链路，TCP趋于给竞争的多条TCP连接提供平等的带宽共享。书上给出了理想化的证明，在连接有相同MSS和RTT的情况下，公平性是可以保证的。

现实中存在的不公平挑战：

- 多媒体应用使用UDP连接压制TCP流量
- Web浏览器常使用并行连接传输多个对象

7.4 明确拥塞通告

拥塞控制的另一类是网络明确向TCP双方发送拥塞信号，这种形式的网络辅助拥塞控制称为明确网络拥塞控制通告ECN。这种形式下，路由器使用一种ECNbit指示该路由器正在经历拥塞，并将该标记携带在IP数据报中，发送给目的主机，再由目的主机告知源主机，实现拥塞控制。

总结

第三章 总结(1/2)

- 传输层提供的服务
  - 应用进程间的逻辑通信
    - Vs 网络层提供的是主机到主机的通信服务
  - 互联网上传输层协议：UDP TCP
    - 特性
- 多路复用和解复用
  - 端口：传输层的SAP
  - 无连接的多路复用和解复用
  - 面向连接的多路复用和解复用
- 实例1：无连接传输层协议 UDP
  - 多路复用解复用
  - UDP报文格式
  - 检错机制：校验和
- 可靠数据传输原理
  - 问题描述
  - 停止等待协议：
    - Rdt1.0 rdt2.0,2.1 ,2.2 Rdt 3.0
  - 流水线协议
    - GBN
    - SR

第三章 总结（2/2）

- 实例2：面向连接的传输层协议-TCP
  - 概述：TCP特性
  - 报文段格式
    - 序号，超时机制及时间
  - TCP可靠传输机制
  - 重传，快速重传
  - 流量控制
  - 连接管理
    - 三次握手
    - 对称连接释放
- 拥塞控制原理
  - 网络辅助的拥塞控制
  - 端到端的拥塞控制
- TCP的拥塞控制
  - AIMD
  - 慢启动
  - 超时之后的保守策略

4 网络层：数据平面

4.1 导论

路由（控制平面，全局）<>转发（数据平面，局部）

传统方式：基于目标地址+转发表

SDN方式：基于多个字段+流表

保证：服务模型，带宽，丢失，保序，延迟，拥塞反馈

4.2 路由器工作原理

路由器结构组件（4个）：

- 输入端口，交换结构，输出端口，路由选择处理器

输入端口处理和基于目的地的转发：最长前缀匹配规则

“匹配+动作”抽象

交换结构：

- 经内存交换：由CPU（路由选择处理器）直接控制
- 经总线交换：输入端口经一根共享总线将分组直接传送到输出端口
- 经互联网络交换：纵横式交换机
- 内存/总线一次只能处理一个，纵横式只要彼此输出端口不同，一次可以处理n个

交换速率：N倍输入/输出端口速率，否则会成为瓶颈

输出端口处理

何处出现排队：输入排队，输出排队

- 输入排队：线路前部阻塞（HOL）：由于队列前面的受到竞争而阻塞，后面的即使不存在竞争也要等待
- 输出排队：弃尾，主动队列管理（AQM）

多少缓存？

- 缓存数量B=平均往返时延RTT*链路容量C（少量TCP流量）

- B=RTT*C/sqrt(N) （大量TCP流量，N条）

分组调度：FIFO，优先权排队（同一优先权分组之间使用FIFO，另：非抢占式优先权排队），循环和加权公平排队（）

★4.3 网际协议：IPV4，寻址，IPV6及其他

★IPv4数据报格式

【补充】

★IPv4数据报分片

MTU（最大传送单元）

标识，标志（最后为0，其余为1），片偏移（计算题）

★IPv4编址（地址协议）

一个IP地址与一个接口（主机与物理链路的边界）相关联，而不是该接口的主机或路由器

子网划分【考题：计算】

- 网络地址（子网掩码确定的子网地址，后面部分全为0）
- 主机地址（在网络地址和广播地址之间）
- 广播地址（子网掩码确定的子网地址，后面部分全为1）

（不考）特殊IPv4地址：

- 默认路由：0.0.0.0
- 环回地址：127.0.0.0/8（测试IP协议栈是否正常）
- 链路本地地址：169.254.0.0/16

（不考）地址分类（已经被淘汰了）

- A类地址（N.H.H.H）默认子网掩码（255.0.0.0）

- B类地址（N.N.H.H）默认子网掩码（255.255.0.0）
- C类地址（N.N.N.H）默认子网掩码（255.255.255.0）

无类别域间路由选择（CIDR）

- a.b.c.d/x前x最高比特构成IP地址的网络部分（网络前缀/前缀）

路由聚合/地址聚合/路由摘要：使用单个网络前缀通告多个网络的能力（书P221案例，通过申明更长的前缀，可以实现搬家的效果）

★动态主机配置协议（DHCP，DynamicHostConfigurationProtocal）

4个步骤【补充】

- DHCP发现报文
- DHCP提供报文
- DHCP请求报文
- DHCP ACK报文

★网络地址转换NAT（NetworkAdressTranslation）

10.0.0.0/8这类地址被保留用于专用网络，仅在给定的网络中才有意义。

NAT自己运行一个DHCP服务器给子网用户分IP地址，并从ISP的DHCP服务器得到自己的IP地址。

NAT转换表（WAN端，LAN端）

NAT缺点：

- （书上）服务器进程在周知端口号上等待入请求
- （书上）P2P对等方在充当服务器时需要接受入连接
- （书上）违背IP设计理念，破坏端到端模型
- 延迟增加，
- 应用限制
- 端口限制（端口数量有限）
- 单点故障
- 未标准化
- 突破NAT穿越问题很麻烦

（不考）分类

- FullCone NAT（只要知道内部主机的IP和Port，就可以发送数据包）IP和端口都不限制
- Restricted Cone NAT（只有内网主机主动请求过的外部主机，该外部主机可以使用自己的IP+任意端口，可以向内网主机发送）限制了IP不限制端口
- Port Restricted Cone NAT（只有内网主机主动请求过的外部主机，该外部主机可以使用自己的IP+自己的端口，可以向内网主机发送）限制了IP+端口
- Symmetric NAT（只有收到内网主机发送的数据才能往回发送，无法实现UDP-P2P通信）

（不考）NAT穿越

- 中继，逆向链接，
- UDP打洞（QQ的原理，A与B连接S进行注册，A向B通信时，A向S通信，要求B向S提供连接，一旦都打开，A可直接与B通信）

（老师给出）NAT的问题

- 结构上不合理（与IP提供端到端可达性相悖）；单点故障；没有标准化；突破NAT穿越问题很麻烦；NAT不等同于防火墙；端口号限制

IPv6

IPv6数据报格式：版本，流量类型，流标签，有效载荷长度，下一个首部，跳限制，源地址，目的地址，数据

★注意：IPv6地址共128bit，IPv6不允许在中间路由器上进行分片/重新组装，并因此取消了首部检验和，同时取消了选项字段。

IPv4到IPv6的迁移：建隧道，在中间使用IPv4的路段使用IPv4封装IPv6

（不考）4.4 通用转发和SDN

第4章补充

IP定位工具

- ping工具
- Traceroute工具
- HTTPHead
- CDN工具
- DNS工具

网络空间测绘技术

SDN三大特征

- 集中控制、分离数据平面和可编程性

P4 可编程协议无关报文处理语言

SD-WAN 广域网组网

5 网络层：控制平面

5.1 概述

路由选择算法

分类方式（集中式/分散式，静态/动态，负载敏感/负载迟钝）

5.2 路由选择算法

5.2.1 链路状态路由选择算法（LS）

【具体内容……】Dijkstra算法

问题：拥塞敏感的路由选择的振荡

解决：确保并非所有路由器同时运行LS算法，避免自同步（发送链路通告的时间随机化）

5.2.2 距离向量路由选择算法（DV）

【具体内容……】Bellman-Ford算法

迭代的，异步的，分布式的算法

问题：

- 好消息传播快，坏消息传播慢（算法本身对于信息的真伪无法甄别）
- 无穷计数问题：（链路突然变坏的消息没有及时被别的节点收到，误以为那条路仍然是好的，且这个消息会干扰到直接与那条路相连的节点）
- （解决：毒性逆转：z通告y路由选择到目的地x，则z向y通告它到x的距离是无穷大【三个以上节点的环路无法检测】）
- （书上没有）解决：水平分割：路由器不使用接收更新的同一接口来通告同一网络（从根本上解决问题）

LS和DV比较

- 报文复杂性：
- 收敛速度：DV算法收敛慢，会遇到无穷计数问题
- 健壮性：LS较好，DV上一个不正确的节点计算会扩散全网

5.3 OSPF：因特网中自治系统内部的路由选择

开放最短路优先(OSPF)算法。

OSPF是一种链路状态协议，使用洪泛链路状态信息和Dijkstra最短路径算法。每台路由器在本地运行Dijkstra算法，构建一个关于整个自治系统的完整拓扑图。

使用OSPF时，路由器向自治系统内所有其他路由器广播路由选择信息（不仅仅是向邻居广播），当链路状态变化或经过一个特定周期，路由器就会广播链路状态信息。

OSPF通告包含在OSPF报文中，直接由IP承载。

OSPF优点如下：

- 安全：使用鉴别，仅有受信任的路由器能参与一个AS内的OSPF协议。
- 多条相同开销的路径：对于多条相同开销路径，OSPF允许使用多条路径，这样可以负载均衡。
- 对单播与多播路由选择的综合支持。
- 支持在单个AS中的层次结构：一个OSPF自治系统能够层次化配置多个区域，每个区域都运行自己的OSPF链路状态路由选择算法，路由器只在区域内进行广播，一台或多台边界路由器负责为流向区域外的分组提供路由选择。

★5.4 BGP：边界网关协议（Border Gateway Protocol）

对于BGP来说，只要把分组送到分组的目标网络，不需要考虑送到目标主机的问题。因此BGP中，一个目的地是一个地址前缀，即一个网络。BGP通过两个手段将分组送到目标网络：

- 从邻居AS获取前缀的可达性信息：BGP允许每个子网向因特网中的其他部分通告自己的存在，这样一个AS就知道一个子网的存在，才可能将分组送达这个子网。即“我在这里”
- 确定最好的路由路径：基于策略和可达性信息，路由器会选择一条尽可能好的路由路径传送分组。

通告BGP路由信息

- 路由器只有两种：网关路由器（在AS边缘），内部路由器（在AS内）
- 路由器通过使用179端口的半永久【TCP连接】交换路由选择信息，每条路由器之间的连接和发送的BGP报文，称为BGP连接
- 跨越两个AS的BGP连接为外部BGP(eBGP)，
- AS内部路由器的BGP会话为内部BGP(iBGP)

BGP属性

- AS-PATH
- NEXT HOP：AS-PATH起始路由器的IP地址

确定最好的路由

- 热土豆路由选择：尽可能快地（最低开销地）送出本AS，即最靠近NEXT-HOP
- 路由选择算法：依次采取以下规则：指派一个本地偏好作为属性之一，选择具有最短AS-PATH（AS-PATH就是从当前AS到达一个AS的路径）的路由，使用热土豆路由选择，使用BGP标识符

- 注意：当在转发表中对于热土豆路由选择增加AS向外前缀时，AS间路由选择协议（BGP）和AS内部路由选择协议（OSPF）都要用到。

IP任播

- 常用于DNS
- 用处：在分散不同的地理位置，替换不同服务器上的相同内容
- 让每个用户从最靠近的服务器访问内容
- 原理：为多台服务器指派相同的IP地址，并使用标准BGP通告该IP地址。这样当某台BGP路由器收到对于该IP地址的多个路由通告时，它会认为这是对相同物理位置的不同路径，因而根据本地的BGP路由选择算法选择一个较优的路径，事实上访问了最靠近的内容
- 在CDN网络实践上并没有采用：BGP路由选择变化能导致相同的TCP连接的不同分组到达Web服务器的不同实例（即到达了不同的服务器上，虽然是同一个IP）

路由选择策略

- 客户网络<>提供商网络
- 接入ISP：所有进入接入ISP网络的流量必定以该网络为目的地，所有离开该接入ISP网络的流量必定源于该网络。（我理解为该网络不承担中转任务，只作为终端节点的客户网络）
- 多宿接入ISP：连接了多个提供商网络的客户网络。
- 提供商网络间结算由它们自己协商

拼装在一起

如果要创建一个具有服务器的小型公司网络

- 首先与本地ISP签合同，将网关路由器与ISP路由器相连，获取IP地址范围，分配IP地址，Web服务器，电子邮件服务器，DNS服务器，网关路由器以及其它设备都需要IP地址。
- 与因特网注册机构签合同，获取域名，向注册机构提供我的DNS服务器的IP地址，注册机构将在顶级域名服务器中为我的DNS服务器添加一个表项
- 本地ISP使用BGP向所有与它连接的ISP通告我的前缀，最终所有因特网路由器都将得知我的前缀。

【不考】5.5 SDN控制平面

★5.6 ICMP：因特网控制报文协议

主机和路由器用来彼此沟通网络层信息，用途：差错报告。

ping的实现，ICMP源抑制报文，Traceroute的实现（向不可达UDP端口号发送TTL逐次递增的报文）

补充：测试工具：

- ping的参数（-l size指定数据包大小，-i ttl设置存活，-w timeout设置超时时间）
- traceroute（tracert）
- netstat
- pathping（综合ping和traceroute的功能）
- tcping
- http-ping（解决针对ICMP-ping的伪装）
- dig（复现DNS查询过程，包括上级服务器等）

【】5.7 网络管理和SNMP

SNMP Protocol

6 链路层和局域网

6.1 链路层概述

节点，链路，链路层帧（在链路之间传递的数据报被封装成链路层帧），

链路层提供可能的服务

- 成帧（framing）
- 链路接入，媒体访问控制（MediumAccessControl,MAC）
- 可靠交付（很多有线链路层协议不提供可靠交付）

- 差错检测和纠正（EDC）

链路层在何处实现？网络适配器/网络接口卡（NIC）

链路层是硬件和软件的结合体

6.2 差错检测和纠正技术

【不重要】

- 奇偶校验，
- 检验和方法（运输层）：用软件实现。因特网检验和：TCP和UDP对首部&数据字段计算，其他协议只对首部计算
- 循环冗余检测（链路层）：硬件实现，可以使用复杂方法。

循环冗余检测（CRC, Cyclic Redundancy Check）

首先协商一个r+1比特模式，生成多项式G（最左边是1，剩下为r位）。

发送方：对于给定要保护的多项式D，左移r位，右边空出来的部分填充R。变换后的式子是（D*2^r XOR R）。发送方要求出R，使得该式子可以整除G（整除的过程也相当于做异或）。

R = remainder[（D*2^r）/G]，remainder的意思是取整除结果的后r位

接收方：若接收到的式子是可以整除G的，就是正确的。

这种方法可以检测小于r+1比特的突变。

★6.3 多路访问链路和协议

多路访问协议

- 信道划分协议（Channel Partition Protocol）
- 随机接入协议（Random Access Protocol）
- 轮流协议（Taking-turns Protocol）

信道划分协议

- 时分多路复用（TDM）
- 频分多路复用（FDM）
- 码分多址（CDMA）对每个节点分配一个不同的编码，接收方根据编码来接收，不同节点能同时传输（由于有混码问题，现在不再使用，【不重要】）

随机接入协议

- 时隙ALOHA：有一个新帧要发送的时候，在下一个时隙传输整个帧；若无碰撞，成功发出；若有碰撞，在之后以概率p在之后的每个时隙中重传该帧直到成功传出。（节点只在时隙起点开始传输帧，节点同步，每个节点知道时隙什么时候开始）
- ALOHA：没有引入时隙，规则同上。有一个新帧要发送的时候，立刻发送；若有碰撞，在该碰撞帧传递完后，立即以概率p重传以概率(1-p)等待这段时间，之后也是类似的。
- 载波侦听多路访问（CSMA，Carrier Sense Multile Access）：【载波侦听】传输前先听信道，若有节点正在发送，就等它们传输完再等一小会儿自己再传输（还会发生碰撞，为什么？信道传播时延）
- 具有碰撞检测的载波侦听多路访问（CSMA/CD）：【载波侦听】跟上面一样+【碰撞检测】当检测到另一个节点正在传输干扰帧时，停止传输并随机等待一定时间（二进制指数后退binary exponential backoff算法：在该帧连续经历了n次碰撞后，等概率地从{0，1，.....,2^n-1}中选择一个值作为K值。以太网中，K*512比特时间为在开始“侦听-当空闲时传输”前的等待时间）
- CSMA/CD效率；近似公式：效率=1/（1+5d[prop]/d[trans]），d[prop]为信号能量在仍任意两个适配器之间传播所需的最大时间，d[trans]表示传输一个最大长度的以太网帧的时间

轮流协议

- 轮询协议：主节点通知每个节点它可以传输的帧的最多数量（缺点：引入轮询时延，若主节点有故障整个网络瘫痪）
- 令牌传递协议：没有主节点，不同节点之间传递一个被称为“令牌”的特殊帧（缺点，任意节点出问题使令牌没有正确传递，网络都会瘫痪）

DOCSIS：用于电缆因特网接入的链路层协议【不重要】

★6.4 交换局域网LANS（LocalNetwork）

链路层地址，又称：LAN地址，物理地址，MAC地址

MAC地址

- 主机和路由器（的适配器）具有MAC地址
- MAC地址是唯一的（由IEEE管理）
- MAC广播地址：FF-FF-FF-FF-FF-FF

ARP（Address Relsolution Protocol）

- 作用：将IP地址转换为MAC地址
- 区别：DNS为在因特网中任何地方的主机解析主机名，而ARP只为在同一个子网上的主机和路由器接口解析IP地址
- ARP表（保存IP地址和MAC地址，有TTL，通常为20mins）
- ARP是跨越链路层和网络层的协议

在子网内发送：

- 若ARP表中有对应IP的MAC地址，让适配器直接发送即可。
- 若ARP表没有对应IP的MAC地址，向适配器传递ARP查询分组，适配器用广播MAC地址FF-FF-FF-FF-FF-FF发送使用链路层帧封装好的ARP分组；接收到的每个适配器都会检查自己的MAC地址是不是要找的，如果是，使用标准帧回应。

发送到子网外：

- 路由器的每个接口都有一个IP地址，一个ARP模块和一个适配器
- 源主机A向该子网的网关路由器B发送数据包
- 若主机A没有该路由器B的MAC，使用ARP寻址
- 网关路由器B向目标子网发送数据包
- 若网关路由器B没有目标主机C的MAC，使用ARP寻址

以太网

- 集线器（物理设备）<>交换机
- 集线器作用于各比特，在一个接口上收到一个比特时，将能量强度放大，然后将该比特向其他所有接口传输出去（相当于一次只能接通一条线路）
- 以太网帧结构：数据字段（46-1500字节，超过1500字节需要分片），目的地址（6字节），源地址（6字节），类型字段（2字节：IP?ARP?.....），CRC（循环冗余检测，4字节），前同步码（8字节：用于唤醒适配器并且同步时钟）
- 以太网技术向网络层：提供无连接服务（不握手），提供不可靠服务（不回ACK也不否认）

链路层交换机

- 作用：接收入链路层帧并转发到出链路
- 对于子网中的主机和路由器来说是透明的
- 过滤：决定某帧是转发到某个接口还是丢弃
- 转发：决定某帧应该被导向某个接口

对于帧的处理

- 对于从x接口到达的帧：
- 若没有其目的MAC对应的表项，向除到来接口x外所有口广播该帧
- 若有其目的MAC对应的表项且接口为x，丢弃该帧
- 若有其目的MAC对应的表项且接口为y≠x，向接口y的输出缓存转发该帧

交换机表的更新

- 交换机表：
- 表项为MAC地址，接口，时间
- 具有自学习能力：初始为空，对于T时间从x接口接收到的MAC地址为M的入帧，在交换机表中加入一条（M,x,T），在老化期后若再没收到，就删除该表项
- 交换机为“即插即用设备”，全双工（每个接口都能同时收发）

链路层交换机<>路由器

- 交换机即插即用，路由器需要人为配置IP地址
- 路由器对分组处理时间更长
- 路由器通常不会发生路由器循环，即使有冗余路径
- 路由器对第二层的广播风暴提供了保护

虚拟局域网

之前的局域网存在缺陷：缺乏流量隔离，每个最小单位太多时导致交换机的无效使用，管理用户困难

VLAN（Virtual LAN）允许用一个单一物理局域网的基础设施定义多个虚拟局域网

VLAN干线连接（每个交换机上有一个特殊的端口用来连接一个交换机，一种特殊的帧802.1Q用于跨越VLAN干线）

【】6.5 链路虚拟化：网络作为链路层

PARP

MPLS

【】6.6 数据中心网络

★6.7 回顾：Web页面请求的历程

【书P326-330】

DHCP(UDP)→DHCP服务器返回分配的IP地址，默认网关路由器IP地址，DNS服务器IP地址

→ARP→查询默认网关路由器MAC地址

→DNS(UDP)→查询www.hnu.edu.cn的IP地址（路由器根据自治区内OSPF知道DNS服务器该如何转发）

→TCP三次握手，HTTP GET

7 无线网络和移动网络

7.1 概述

无线主机，无线通信链路（两个主要特性：覆盖区域，链路速率）

基站（蜂窝塔，接入点等），无线主机与基站“相关联”：该主机位于该基站的无线通信覆盖范围内，该主机使用该基站中继它与更大网络之间的数据

基础设施模式（通过基站相连），自组织网络（无基础设施）

基于一个无线跳/多个无线跳分组：

- 单跳，基于基础设施：802.11，4G LTE数据网络
- 单跳，无基础设施：蓝牙网络，具有自组织模式的802.11

- 多跳，基于基础设施：无线传感网络，无线网状网络
- 多跳，无基础设施：移动自组织网络

7.2 无线链路和网络特征

★有线链路与无线链路的区别：

- 递减的信号强度（路径损耗）
- 来自其它源的干扰（同一个频段发射的电波源互相干扰）
- 多径传播（电磁波受到物体和地面的反射，使得信号变模糊）
- 隐藏终端问题：A和C都向B通信，但由于物理阻隔，A和C彼此不知道对方的存在

对于CDMA编码的讨论【略】

7.3 WiFi：802.11 无线 LAN

802.11结构：

- 基本服务集（BSS,Basic Service Set），一个BSS包含一个或多个无线站点和中央基站
- 接入点（AP,Acess Point）

信道与关联

- AP被安装时，会分配一个服务器标识符（SSID,Service Set Identifier）和一个信道号
- 每个AP周期性地发送信标帧（包括该AP的SSID和MAC地址）
- 我的无线站点扫描信道并连接可用的AP用于关联
- 扫描方式可以有被动扫描和主动扫描（广播探测帧）
- 确定想关联的AP后，我的无线主机向该AP发送一个关联请求帧，该AP使用关联响应帧回应
- 关联后，我的主机通过关联的AP向子网发送DHCP发现报文，以获取在该AP子网中的一个IP地址

802.11 MAC协议

CSMA/CA（collision avoidance）带碰撞避免的CSMA

★802.11与以太网的区别

- 802.11使用碰撞避免而非碰撞检测（接收信号的强度远远小于发送强度，无法同时发送/接收信号，隐藏终端问题和衰减问题，没法做到碰撞检测）
- 由于误比特率高，802.11使用链路层确认/重传（ARQ）方案

链路层确认：

目的站点收到一个帧后，等待一小段时间（短帧间间隔SIFS），发挥一个确认帧，若发送站点一定时间内未收到确认帧，就假定错误并重传该帧，若多次错误就抛弃该帧。

CSMA/CA：

- 站点最初监听到信道空闲，在短时间（分布式间隔帧DIFS）后发送该帧
- 否则，该站点选取一个随机回退值，并在侦听信道空闲时递减该值，若信道忙，该值不变。
- 当计数减为0时，该站点发送整个数据帧并等待确认
- 若收到确认，则该帧成功发送。若未收到确认，重新进入选取回退值的阶段，并在一个更大的范围内选择回退值

处理隐藏终端：RTS和CTS

- 请求发送控制帧RTS
- 允许发送控制帧CTS
- 发送方先向AP发RTS并预约传输DATA帧和收回ACK帧需要的时间
- AP广播CTS，回应发送方并授权许可，同时该CTS还能指示其它站点在预约期内不要发送

使用802.11作为一个点对点链路

- 定向天线

7.3.3 IEEE 802.11帧

有4个地址（我们关注3个）

- 地址2：传输该帧的站点的MAC地址
- 地址1：要接受该帧的无线站点的MAC地址
- 地址3：AP接入的路由器接口的MAC地址
- （前两个负责无线通信，最后一个确认路由器MAC）

7.3.4 在相同的IP子网中的移动性

- 我的无线站点在远离AP1后解除与AP1的关联，并与AP2关联。
- 怎么让交换机知道？在新关联形成后，让AP2以我的无线站点的源地址向交换机发送以太网广播帧

LTE：long term evolution

课程总结

架构

- 分层架构(internet)、集中/分布式、端到端、C/S、B/S、P2P

原则 - principles

- 端到端：保持一个极简的网络核心，将复杂留在端
- 传输层：不可靠网络层上的可靠通信、连接建立/拆除和握手、拥塞和流量控制以及多路复用
- 网络层：确定两个路由器之间的“良好”路径、互连大量异构网络以及管理现代网络的复杂性、控制平面/数据平面
- 链路层：共享多路访问信道，处理/避免冲突，数据包转发与寻址
- 分层之间的连接：DNS，DHCP，ARP
- 性能指标：时延，吞吐率
- 这些原则具有较长“保质期”—在今天的网络标准和协议过时之后，它们所体现的原则仍然很重要和相关。
- 掌握与这些原则相关的基本问题和解决方法，以后能够快速了解几乎任何网络技术。

发展与竞争

- 技术演进：HTTP1.0->3.0,TCP算法，IPV4->IPV6，以太网（10M->100G），无线网（2G->6G,wifi->wifi7）
- 产业兴衰：伴随网络发展，传统业务比如有线电视，语音电话，短信不断衰落;互联网->移动互联网->产业互联网

争论与妥协

- 电路交换-包交换，集中-分布式，端到端-NAT，IPV4-IPV6，非二层交换、硬件-软件，符号主义-连接主义，

矛盾与演进

- 对立统一（中心化/去中心化）、在应用中螺旋上升

备考

网路层路由，路由器怎么工作，路由协议，路由算法，路由器转发动作

数据平面和控制平面分离，不考

IPv4地址分配，NAT★P225，

ICMP不考

OSPF BGP