

408 计算机网络

- 1.1 信息时代的计算机网络
- 1.2因特网概述
- 1.3电路交换、分组交换和报文交换
- 1.4计算机网络的定义和分类
- 1.5计算机网络的性能指标
- 1.6计算机网络的体系结构
- 1.7我国的计算机网络发展情况

internet:互联网（互连网），通用名词，任意通信协议

Internet:因特网，专用名词，TCP/IP协议族

因特网发展的三个阶段：

1969	1985	1993
ARPANET向互连网发展	三级结构因特网	多层次ISP结构的因特网
1969年，第一个分组交换网 ARPANET	1985年，NSFNET(主干网、地区网和校园网)	1993年，NSFNET被替代，由各种ISP运营
70年代中期，研究多种网络的互连	1990年，ARPANET任务完成，正式关闭	1994年，WWW技术推动因特网迅速发展
1983年，TCP/IP协议成为ARPANET的标准协议（因特网诞生时间）	1991年，因特网初步商业化，开始收费	1995年，NSFNET停止运作，因特网彻底商业化

ISP：Internet Service Provider, 中国为电信、移动、联通，各用户通过ISP接入因特网。

因特网的标准化工作

因特网的标准化工作是面向公众的，其任何一个建议标准在成为因特网标准之前都以RFC技术文档的形式在因特网上发表。

RFC(Request For Comments)的意思是“请求评论”。任何人都可以从因特网上免费下载RFC文档(<http://www.ietf.org/rfc.html>)，并随时对某个RFC文档发表意见和建议。

电路交换：

报文交换：

分组交换：

电路交换	报文交换	分组交换
(1) 建立连接：分配通信资源 (2) 通话：一直占用通信资源 (3) 释放连接：归还通信资源。		报文交换是分组交换的前身。在报文交换中，报文被整个地发送，而不是拆分成若干个分组进行发送。交换节点将报文整体接收完成后才能查找转发表，将整个报文转发到下一个节点。

电路交换	报文交换	分组交换
	没有建立连接和释放连接的过程。 分组传输过程中逐段占用通信链路有较高的通信线路利用率。 交换节点可以为每一个分组独立选择转发路由，使得网络有很好的生存性	
计算机之间的数据传送是突发式的，当使用电路交换来传送计算机数据时，其线路的传输效率一般都会很低，线路上真正用来传送数据的时间往往不到10%甚至1%。	分组首部带来了额外的传输开销。 交换节点存储转发分组会造成一定的时延。 无法确保通信时端到端通信资源全部可用，在通信量较大时可能造成网络拥塞。分组可能会出现失序和丢失等问题。	报文交换比分组交换带来的转发时延要长很多需要交换节点具有的缓存空间也大很多

三种交换方式的对比

若要连续传送大量的数据，并且数据传送时间远大于建立连接的时间，则使用电路交换可以有较高的传输效率。然而计算机的数据传送往往是突发式的，采用电路交换时通信线路的利用率会很低。报文交换和分组交换都不需要建立连接(即预先分配通信资源)，在传送计算机的突发数据时可以提高通信线路的利用率。

将报文构造成为若干个更小的分组进行分组交换，比将整个报文进行报文交换的时延要小，并且还可以避免太长的报文长时间占用链路，有利于差错控制，同时具有更好的灵活性。

计算机网络早期的一个最简单定义：互连、自治（每台计算机都有独立的软件和硬件，并且可以单独运行）、计算机集合。

计算机网络的分类

交换方式	使用者	传输介质	覆盖范围	拓扑结构
电路交换、报文交换、分组交换	公用网（因特网）、专用网(军队、铁路、电力、银行)	有线网络、无线网络	广域网(WAN)几十到几千千米、城域网(MAN)5到50千米、局域网(LAN)1千米左右、个域网(PAN)10米	总线型、星型、环型、网状型。

计算机网络的性能指标：
计算机网络的性能指标被用来从不同方面度量计算机网络的性能。

常用的八个计算机网络性能指标

速率	带宽	吞吐量	时延	时延带宽积	往返时间	利用率	丢包率
数据的 传送速率(即每秒传送多少个比特)可简记为b/s, 也称为数据率、比特率	用来表示网络的通信线路所能传送数据的能力, 即在单位时间内从网络中的某一点到另一点所能通过的最高速率。 数据传送速率=min[主机接口速率, 线路带宽, 交换机或路由器的接口速率]	吞吐量是指在单位时间内通过某个网络或接口的实际数据量。吞吐量常被用于对实际网络的测量以便获知到底有多少数据量通过了网络。吞吐量受网络带宽的限制。	时延是指数据从网络的一端传送到另一端所耗费的时间, 也称为延迟或迟延。数据可由一个或多个分组、甚至是一个比特构成。	时延带宽积是传播时延和带宽的乘积。链路的时延带宽积也称为以比特为单位的链路长度, 这以后理解以太网的最短帧长是非常有帮助的。	往返时间(Round-Trip Time, RTT)是指从发送端发送数据分组开始, 到发送端收到接收端发来的相应确认分组为止, 总共耗费的时间。		丢包率是指在一定的时间范围内, 传输过程中丢失的分组数量与总分组数量的比率

速率的单位	换算关系	数据量的单位	换算关系
比特/秒(b/s)	基本单位	比特(b)	基本单位
		字节(B)	1B = 8bit
千比特/秒(kb/s)	kb/s =10 ³ b/s	千字节(KB)	KB=2 ¹⁰ B
兆比特/秒(Mbs)	Mb/s = k*kb/s = 10 ⁶ b/s	兆字节(MB)	MB =K*KB=2 ²⁰ B
吉比特/秒(Gb/s)	Gb/s= k*Mb/s = 10 ⁹ b/s	吉字节(GB)	GB=K*MB = 2 ³⁰ B
太比特/秒(Tb/s)	Tb/s = k*Gb/s = 10 ¹² b/s	太字节(TB)	TB=K*GB=2 ⁴⁰ B

- 1

发送时延只涉及数据从发送端推送到网络所需的时间, 不包括数据在网络中的传播时延、处理时延或排队时延。
- 2

对于高带宽网络, 发送时延通常较短, 但当数据量较大时, 发送时延会显著增加。
- 3

发送时延=分组长度**(b)**/发送速率**(b/s)**=数据大小/带宽

1 | 传播时延=信号长度(m)/信号传播速率 (m/s)

排队时延和处理时延不方便计算

1 | 发送速率=min[主机接口速率，线路带宽，交换机或路由器的接口速率]

链路利用率	网络利用率
链路利用率是指某条链路有百分之几的时间是被利用的(即有数据通过) 完全空闲的链路的利用率为零	网络利用率是指网络中所有链路的链路利用率的加权平均。

根据排队论可知，当某链路的利用率增大时，该链路引起的时延就会迅速增加。

当网络的通信量较少时，产生的时延并不大，但在网络通信量不断增大时，分组在交换节点(路由器或交换机)中的排队时延会随之增大，因此网络引起的时延就会增大。

令 D_0 表示网络空闲时的时延， D 表示网络当前的时延，那么在理想的假定条件下，可用下式来表示 D 、 D_0 和网络利用率 U 之间的关系。

1 | $D(\text{网络当前时延}) = D_0(\text{网络空闲时的时延}) / (1 - U)$

分组丢失主要有以下两种情况:

1. 分组在传输过程中出现误码，被传输路径中的节点交换机(例如路由器)或目的主机检测出误码而丢弃。
2. 节点交换机根据丢弃策略主动丢弃分组。

丢包率可以反映网络的拥塞情况:

1. 无拥塞时路径丢包率为0。
2. 轻度拥塞时路径丢包率为1%~4%。
3. 严重拥塞时路径丢包率为5%~15%。

常见的三种计算机网络体系解构:

1.	OSI	TCP/IP	1
	应用层	应用层	
	表示层	运输层	
	会话层	网际层	
	运输层	网络接口层	
	网络层		
	数据链路层		
	物理层		

OSI	TCP/IP	1
OSI标准失败的原因 专家没有实际经验完成标准时没有商业驱动力 协议实现过分复杂运行效率很低 标准的制定周期太长产品无法及时进入市场 层次划分不太合理有些功能在多个层次中重复出现		

误码的相关概念

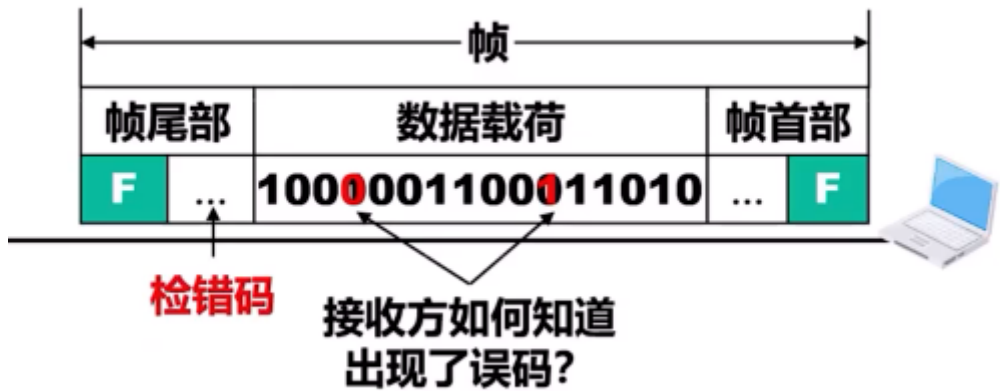
实际的通信链路都不是理想的，比特在传输过程中可能会产生差错(称为比特差错)

比特1可能变成比特0

比特0可能变成比特1

在一段时间内，传输错误的比特数量占所传输比特总数的比率称为误码率(Bit Error Rate，BER)提高链路的信噪比，可以降低误码率。但在实际的通信链路上，不可能使误码率下降为零。

使用差错检测技术来检测数据在传输过程中是否产生了比特差错，是数据链路层所要解决的重要问题之一。



检错码成为FCS

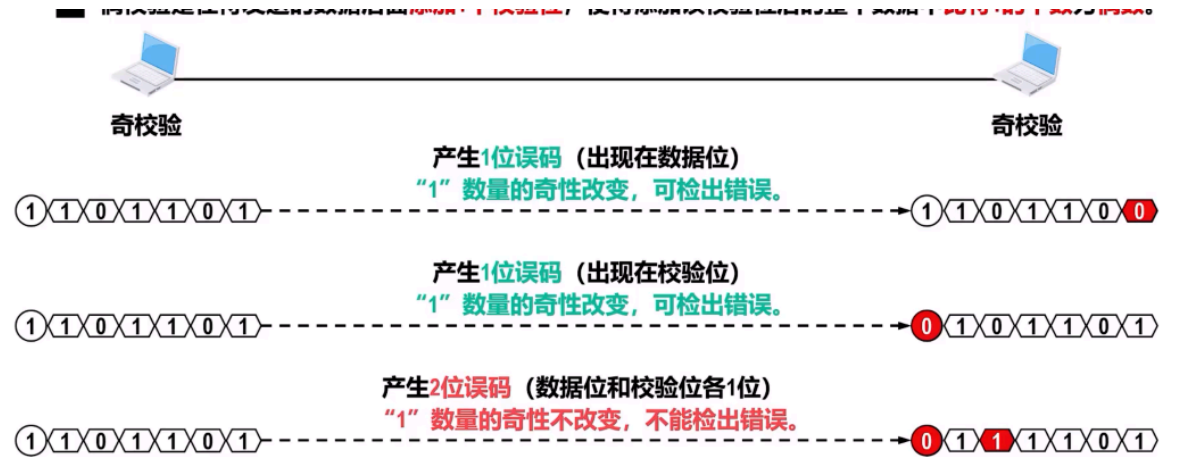
奇校验是在待发送的数据后面添加1个校验位，使得添加该校验位后的整个数据中比特1的个数为奇数。

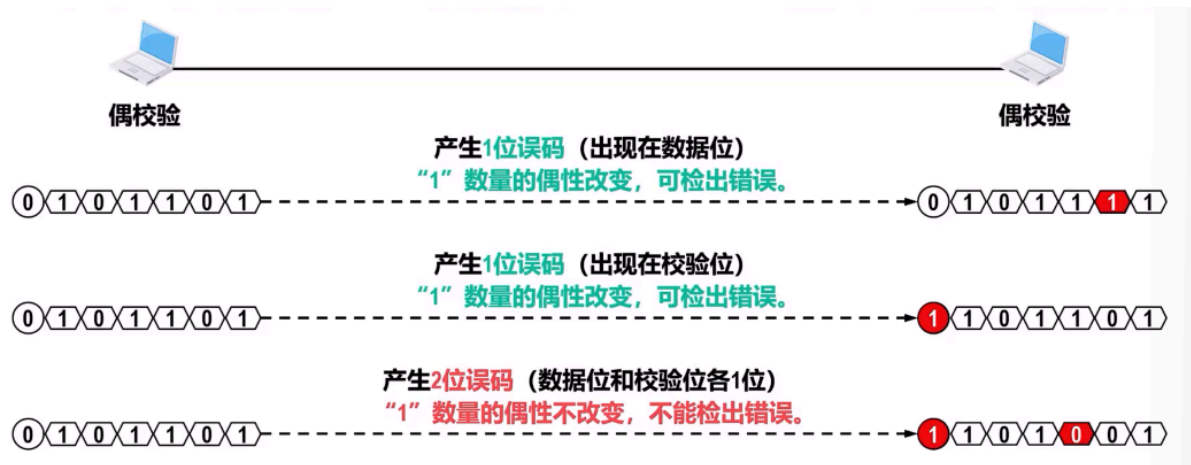
偶校验是在待发送的数据后面添加1个校验位，使得添加该校验位后的整个数据中比特1的个数为偶数。

在所传输的数据中，如果有奇数个位发生误码，则所包含比特1的数量的奇偶性会发生改变，可以检测出误码。

在所传输的数据中，如果有偶数个位发生误码，则所包含比特1的数量的奇偶性不会发生改变，无法检测出误码(漏检)

在实际使用时，奇偶校验又可分为垂直奇偶校验、水平奇偶校验以及水平垂直奇偶校验。





循环冗余校验

数据链路层广泛使用漏检率极低的循环冗余校验:(Cyclic Redundancy Check, CRC)检错技术

循环冗余校验CRC的基本思想:

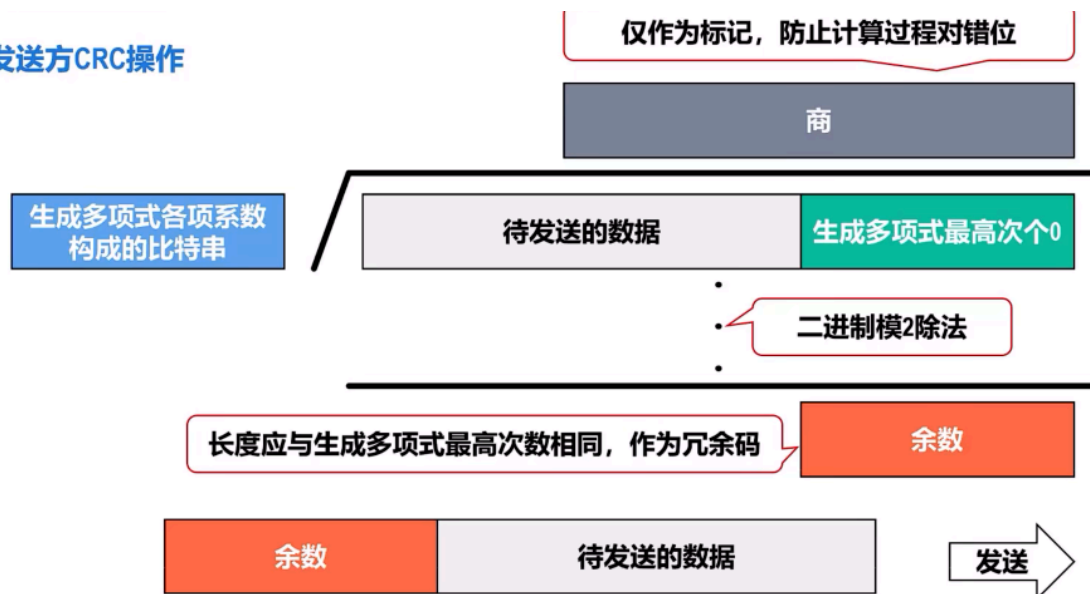
收发双方约定好一个生成多项式 $G(X)$ 。

发送方基于待发送的数据和生成多项式 $G(X)$, 计算出差错检测码(冗余码)

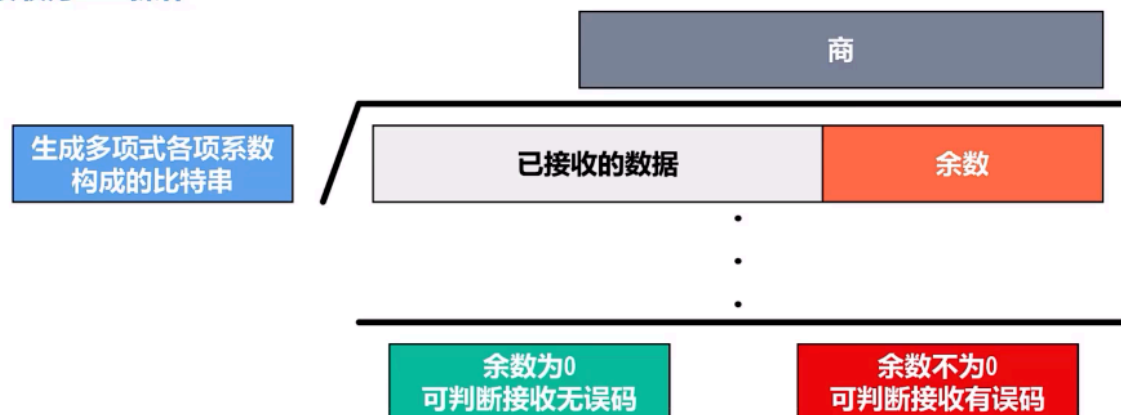
将冗余码添加到待发送数据的后面一起传输。

接收方收到数据和冗余码后, 通过生成多项式 $G(X)$ 来计算收到的数据和冗余码是否产生了误码。

发送方CRC操作



接收方CRC操作



【生成多项式举例】

$$G(X) = X^4 + X^2 + X + 1$$
$$= \boxed{1} \cdot X^4 + \boxed{0} \cdot X^3 + \boxed{1} \cdot X^2 + \boxed{1} \cdot X^1 + \boxed{1} \cdot X^0$$

生成多项式各项系数构成的比特串：10111（计算冗余码时作为除数）

【常用的生成多项式】

$$CRC-16 = x^{16} + x^{15} + x^2 + 1$$

$$CRC-CCITT = x^{16} + x^{12} + x^5 + 1$$

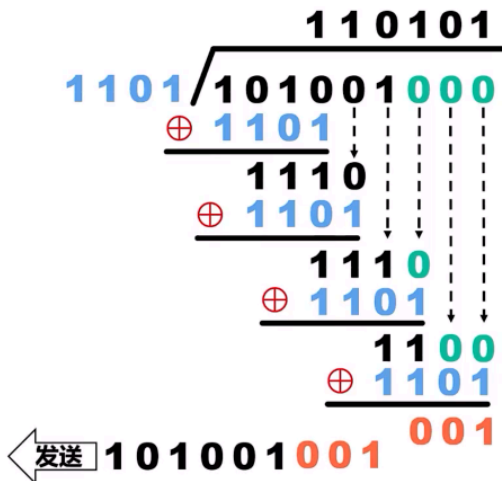
$$CRC-32 = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

算法要求生成多项式必须包含最低次项即x的0次项。

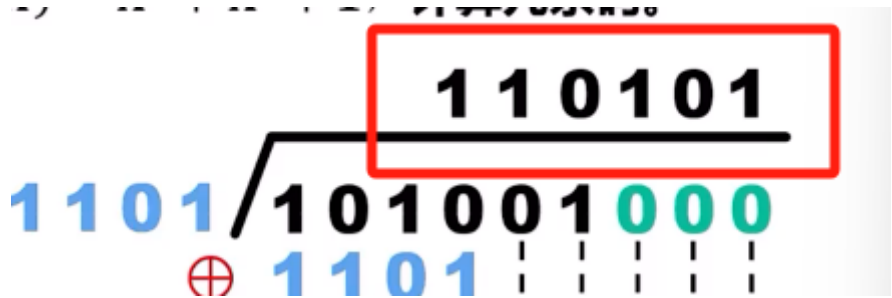
发送方的处理：

【CRC举例】待发送的数据为101001，生成多项式为 $G(X) = X^3 + X^2 + 1$ ，计算冗余码。

1	构造“被除数” 待发送数据后面添加生成多项式最高次数个0
2	构造“除数” 生成多项式各项系数构成的比特串作为除数
3	做“二进制模2除法” 相当于对应位进行逻辑异或运算
4	检查“余数” 余数的位数应与生成多项式最高次数相同， 如果位数不够，则在余数前补0来凑足位数。



生成多项式为 $G(X)=1$ 乘以 X^3+1 乘以 X^2+0 乘以 X^1+1 乘以 X^0



1为够除即被除数和除数位数相同都为4位。

最后一行的1位余数。

接收方的处理

【CRC举例】接收到的信息为101101001，生成多项式为 $G(X) = X^3 + X^2 + 1$ ，判断传输是否误码？

1	构造“被除数” 接收到的信息作为被除数
2	构造“除数” 生成多项式各项系数构成的比特串作为除数
3	做“二进制模2除法” 相当于对应位进行逻辑异或运算
4	检查“余数” 余数为0，可认为传输过程无误码； 余数不为0，可认为传输过程产生了误码。

03 循环冗余校验

【CRC举例】接收到的信息为101101001，生成多项式为 $G(X) = X^3 + X^2 + 1$ ，判断传输是否误码？

1	构造“被除数” 接收到的信息作为被除数	1101
2	构造“除数” 生成多项式各项系数构成的比特串作为除数	1100
3	做“二进制模2除法” 相当于对应位进行逻辑异或运算	
4	检查“余数” 余数为0，可认为传输过程无误码； 余数不为0，可认为传输过程产生了误码。	

110010010
101101000
1100
1100
1100
10

请同学们暂停播放视频

《深入浅出计算机网络》（微课视频版）

最后

接收

写错了，接收方构建被除数时不要将生成多项式的系数添加到后面。

$$\begin{array}{r}
 110010 \\
 1101 \overline{) 101101001} \\
 \underline{\oplus 1101} \\
 1100 \\
 \underline{\oplus 1101} \\
 1100 \\
 \underline{\oplus 1101} \\
 11
 \end{array}$$

余数不为0
可认为传输过程产生了误码!

奇偶校验、循环冗余校验等差错检测技术，只能检测出传输过程中出现了差错，但并不能定位错误因此无法纠正错误。

要想纠正传输中的差错，可以使用冗余信息更多的纠错码(例如海明码)进行前向纠错。但纠错码的开销比较大，在计算机网络中较少使用。

在计算机网络中，通常采用我们后续课程中将要介绍的检错重传方式来纠正传输中的差错，或者仅仅丢弃检测到差错的帧，这取决于数据链路层向其上层提供的是可靠传输服务还是不可靠传输服务。

循环冗余校验CRC具有很好的检错能力(漏检率极低)，虽然计算比较复杂，但非常易于用硬件实现因此被广泛应用于数据链路层。

可靠传输

使用差错检测技术(例如循环冗余校验CRC)，接收方的数据链路层就可检测出帧在传输过程中是否产生了误码(比特差错)。

检测出误码了怎么做呢?

这取决于数据链路层向其上层提供的服务类型。

若为不可靠传输服务:仅仅丢弃有误码的帧，其他什么也不做，

若为可靠传输服务:通过某种机制实现发送方发送什么，接收方最终就能收到什么。（例如：接收方检测到了误码并向发送方发送一个否认帧，发送方收到否认帧后。重新发送）。

一般情况下，有线链路的误码率比较低。为了减小开销，并不要求数据链路层向其上层提供可靠传输服务。即使出现了误码，可靠传输的问题由其上层处理。

无线链路易受干扰误码率比较高，因此要求数据链路层必须向其上层提供可靠传输服务

传输差错:分组重复、分组失序、分组丢失出现在数据链路层的上层。误码(比特差错)出现在数据链路层及其下层

误码(比特差错)

分组丢失:当分组到达路由器R2时,由于R2的排队的队列已经满了,现在又来了一个分组,只能将该分组丢弃了。

分组失序:主机1发送3个有顺序的分组,这3个分组选择了不同的路径,由于路径长度和路由器的繁忙程度不同,所以先发的分组可能最后到达主机2

分组重复:发送的分组到达路由器R5,此时由于R5十分繁忙,导致该分组等待的时间很长,主机1发送超时,所以它重新发送了该分组,该分组选择了一条不繁忙的路径到达了主机2,而在R5的分组也到达了主机2。

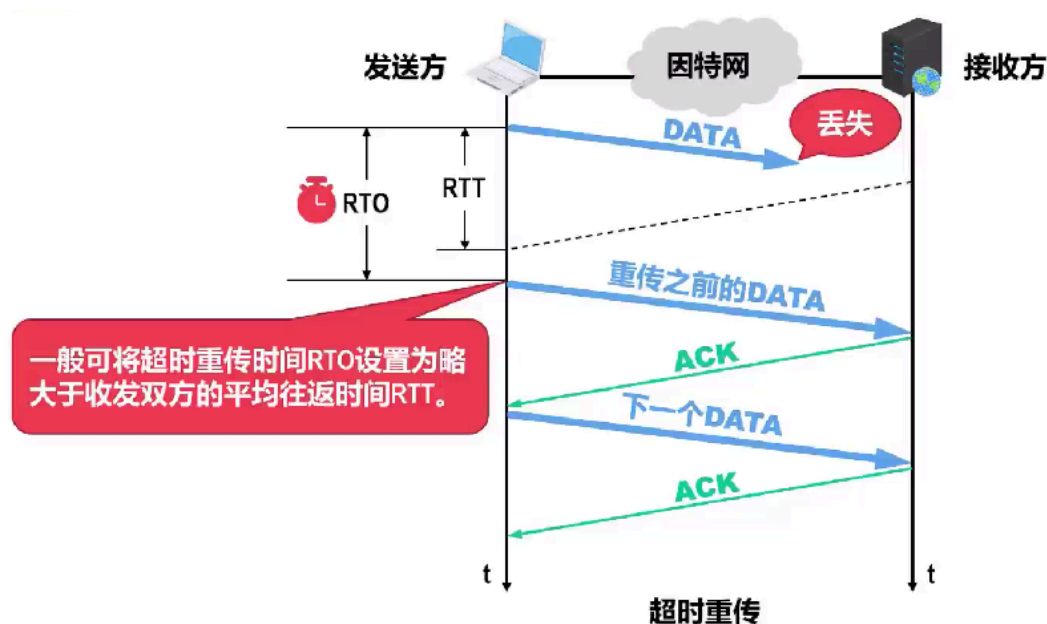
可靠传输服务并不局限于数据链路层,其他各层均可选择实现可靠传输。

实现可靠传输的三种协议

停止-等待(Stop-and-Wait, SW)协议、协议回退N帧(Go-back-N, GBN)协议、选择重传(Selective Repeat, SR)协议

这三种可靠传输实现机制的基本原理并不仅限于数据链路层,可以应用到其上各层。

停止-等待(Stop-and-Wait, SW)协议



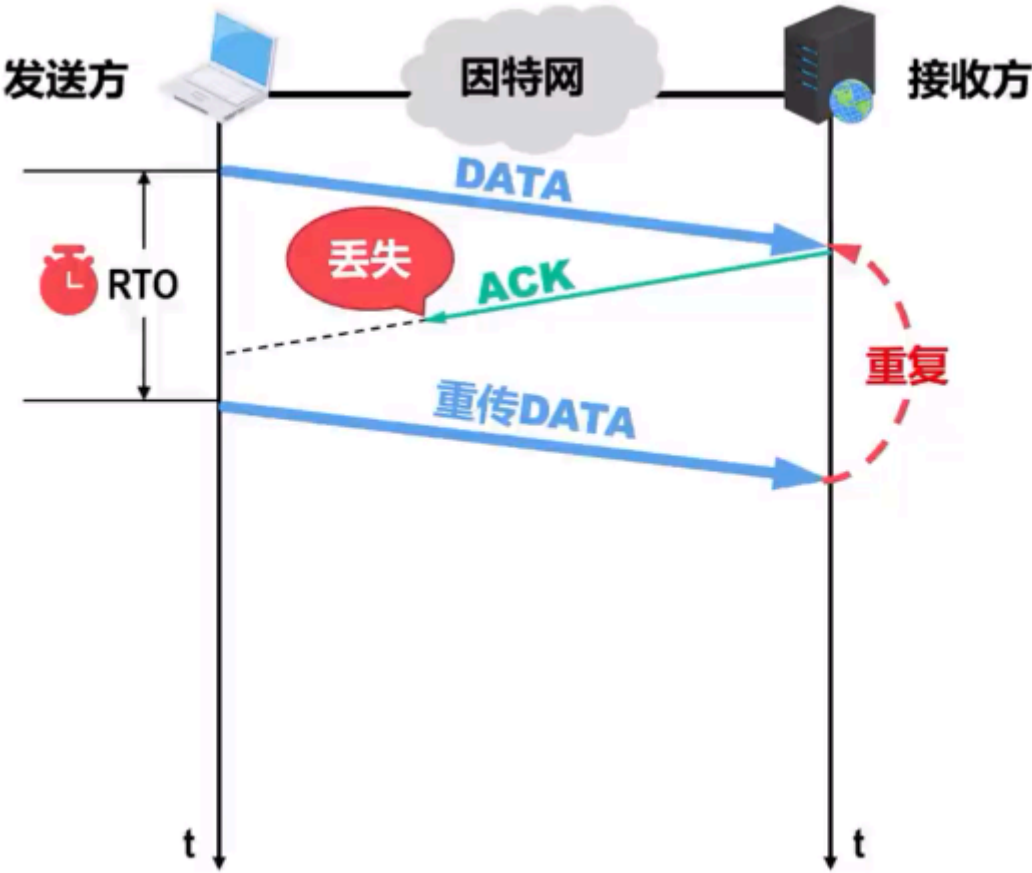
接收方收不到数据分组,就不会发送相应的ACK或NAK。如果不采取措施,发送方就会一直处于等待接收方ACK或NAK的状态。

为了解决上述问题,发送方可在每发送完一个数据分组时就启动一个超时计时器(Timeout Timer)。

若到了超时计时器所设置的超时重传时间(Retransmission Time-Out, RTO),但发送方仍未收到接收方的ACK或NAK,就重传之前已发送过的数据分组。

一般可将超时重传时间RTO设置为略大于收发双方的平均往返时间RTT。超过RTO,重传。

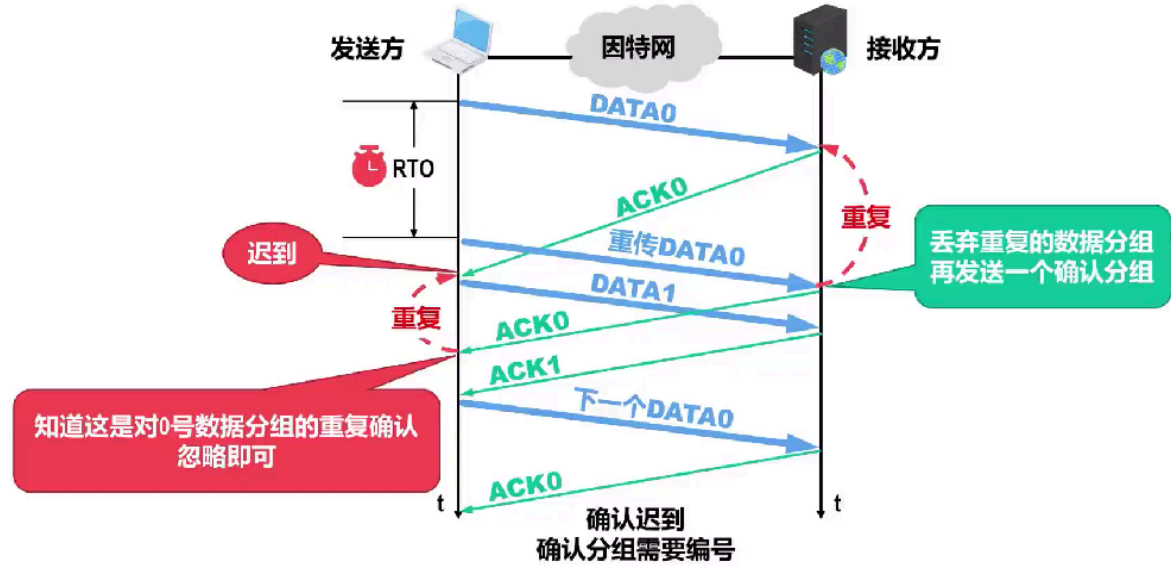
数据分组编号



接收方发送的确认分组丢失，发送方没有收到确认分组，认为发送的数据分组丢失，重新发送数据分组，导致分组重复。

为了避免分组重复这种传输错误，必须给每个分组带上序号。
对于停止-等待协议，由于每发送一个数据分组就停止等待，只要保证每发送一个新的数据分组，其序号与上次发送的数据分组的序号不同就可以了，因此用一个比特来编号就够了，序号有0和1这两个。
接收方接收到了同一数据分组的不同序号时，丢弃该数据分组，并向发送方重新发送一个确认分组。

确认分组编号



发送DATA0，ACK迟到，发送方超时重传DATA0，此时发送方收到了迟到的ACK，发送方发送下一个数据分组DATA1。接收方收到DATA0，根据序号可知，这是一个重复的数据分组，将它丢弃，并发送一个ACK。发送方收到两个对DATA0的ACK，将第二个ACK误认为是对DATA1的。

若对ACK进行比编号，则可知道这是对0号数据分组的重复确认忽略即可。

注意事项

使用超时重传机制后，就可以不使用否认机制了，这样可使协议实现起来更加简单。但是，如果点对点链路的误码率较高，使用否认机制可以使发送方在超时计时器超时前就尽快重传。

为了让接收方能够判断所收到的数据分组是否是重复的，需要给数据分组编号。由于停止-等待协议的特性，只需1个比特编序号即可，即序号0和序号1。

为了让发送方能够判断所收到的确认分组是否是重复的，需要给确认分组编号，所用比特数量与数据分组所用比特数量一样。

数据链路层一般不会出现确认分组迟到的情况，因此在数据链路层实现停止-等待协议可以不用给确认分组编号。

给超时计时器设置的超时重传时间RTO应当仔细选择，一般将RTO设置为略大于收发双方的平均往返时间RTT。

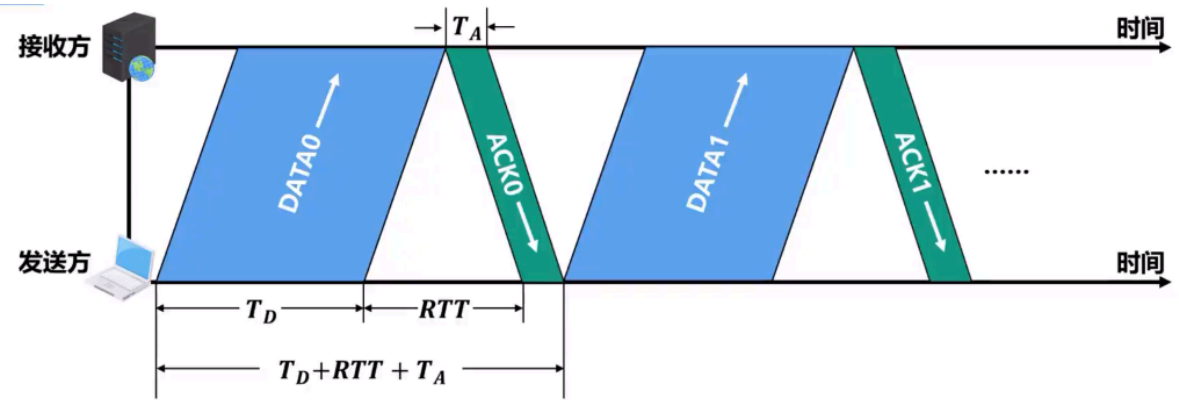
在数据链路层，点对点的往返时间RTT比较固定，RTO就比较好设定。

在运输层，由于端到端往返时间非常不确定，设置合适的超时重传时间RTO有时并不容易。停止-等待协议属于自动请求重传(Automatic Repeat reQuest, ARQ)协议。即重传的请求是发送方自动进行的，而不是接收方请求发送方重传某个误码的数据分组

停止-等待协议的信道利用率

TD是发送方发送数据分组所耗费的发送时延

TA是接收方发送确认分组所耗费的发送时延



因为确认分组长度远小于数据分组长度

例如，卫星链路的RTT很大

信道利用率
$$U \approx \frac{T_D}{T_D + RTT + T_A} \xrightarrow{T_A \ll T_D} U \approx \frac{T_D}{T_D + RTT} \xrightarrow{RTT \gg T_D} U \text{ 很低} \xrightarrow{RTT \ll T_D} U \text{ 比较高}$$

例如，无线局域网的RTT远小于T₀

若出现超时重传，对于传送有用的数据信息来说，信道利用率还要降低。
在往返时间RTT相对较大的情况下，为了提高信道利用率，收发双方不适合采用停止-等待协议，而可以选择使用回退N帧(GBN)协议或选择重传(SR)协议。

【2018年 题36】主机甲采用停-等协议向主机乙发送数据，数据传输速率是3kbps，单向传播延时是200ms，忽略确认帧的传输延时。当信道利用率等于40%时，数据帧的长度为（ ）。

A. 240比特

B. 400比特

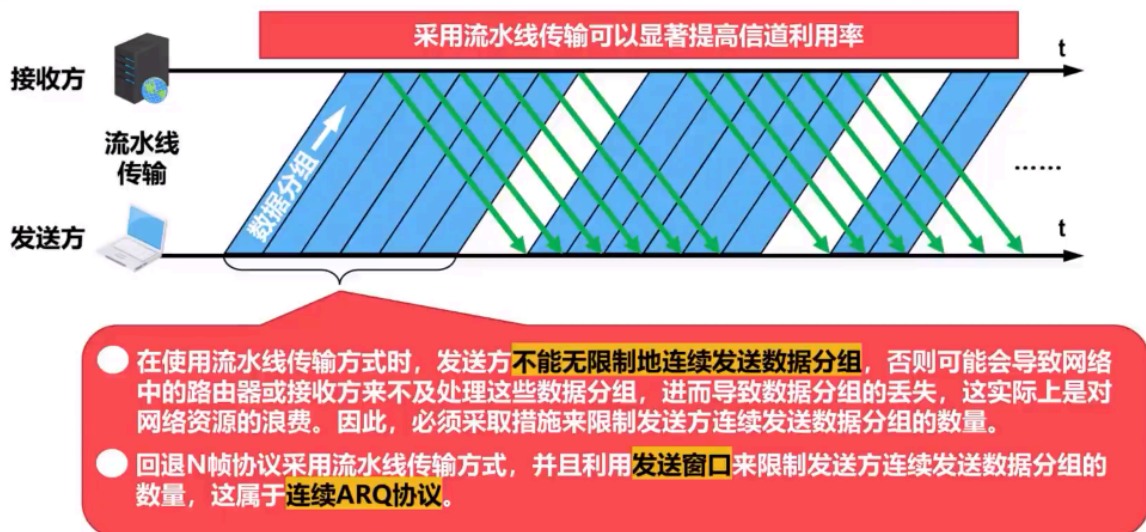
C. 480比特

D. 800比特

$RTT=400ms$ $TD=L/3kbps$

$0.4=L/3kbps$

回退N帧协议

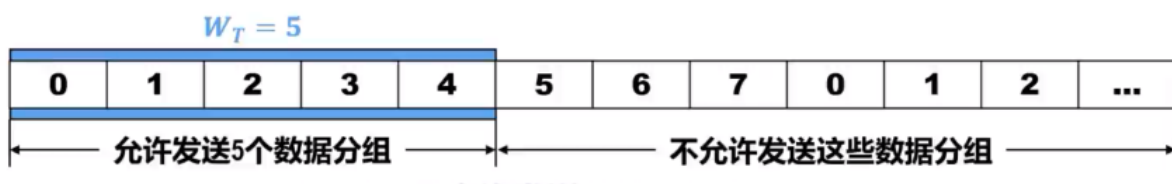


采用n个比特给分组编序号，序号范围是0~(2^n-1)。本例假设采用3个比特给分组编序号则序号范围是0~7。(3个比特可以表示 2^3 个分组，若以0开始则为0~7)

0	1	2	3	4	5	6	7	0	1	2	...
---	---	---	---	---	---	---	---	---	---	---	-----

发送方需要维护一个发送窗口WT，在未收到接收方确认分组的情况下，发送方可将序号落入WT内的所有数据分组连续发送出去。

采用n个比特给分组编序号，则WT的取值范围是 $1 < WT \leq (2^n - 1)$ 。本例假设采用3个比特给分组编序号，则WT的取值范围是2~7，本例取WT=5。(序号不能0/1，0/1是用来给停止-等待协议编号的)



接收方需要维护一个接收窗口WR只有正确到达接收方(无误码)且序号落入WR内的数据分组才被接收方接收。WR的取值只能是1，这一点与停止-等待协议是相同的。

接收方每正确收到一个序号落入接收窗口的数据分组，就将接收窗口向前滑动一个位置这样就有一个新的序号落入接收窗口。与此同时，接收方还要给发送方发送针对该数据分组的确认分组。

发送方每收到一个按序确认的确认分组，就将发送窗口向前滑动一个位置，这样就有一个新的序号落入发送窗口，序号落入发送窗口内的数据分组可继续被发送。

在回退N帧协议的工作过程中，发送方的发送窗口和接收方的接收窗口按上述规则不断向前滑动。因此，这类协议又称为滑动窗口协议。

无传输错误的情况

在无传输差错的情况下，回退N帧协议的信道利用率比停止-等待协议的信道利用率有显著提高。提高的程度取决于发送窗口的大小。

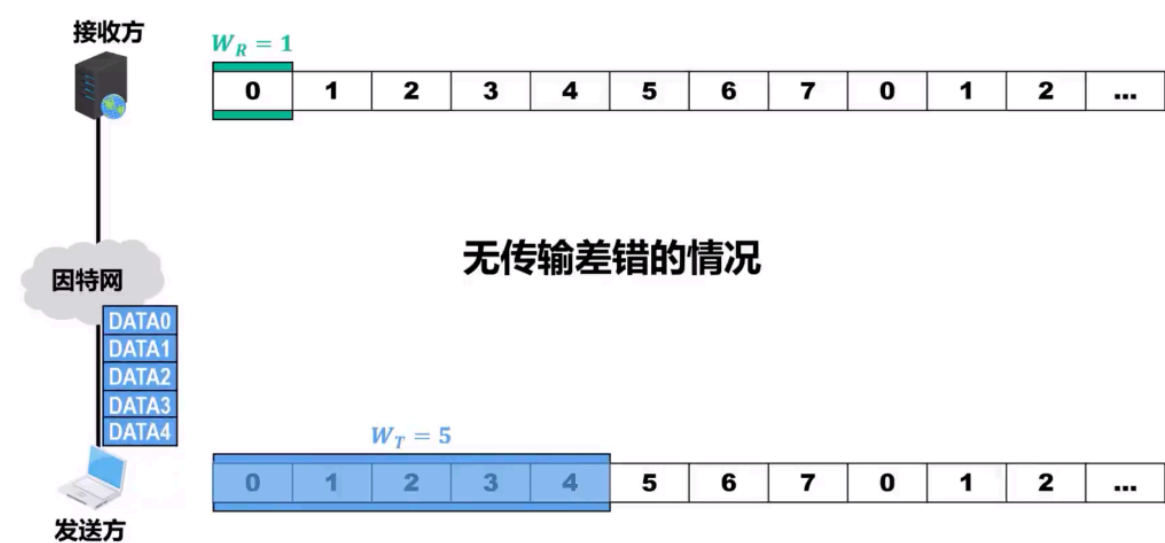


image-20241122204059497

image-20241122204120378

接收方每正确收到一个序号落入接收窗口的数据分组，就将接收窗口向前滑动一个位置这样就有一个新的序号落入接收窗口。与此同时，接收方还要给发送方发送针对该数据分组的确认分组。

image-20241122204149911

发送方每收到一个按序确认的确认分组，就将发送窗口向前滑动一个位置，这样就有一个新的序号落入发送窗口，序号落入发送窗口内的数据分组可继续被发送。

image-20241122204159070

超时重传、回退N帧的情况



image-20241122204707409

通过DATA2中的检错码发现了误码将其丢弃。

image-20241122204845495

将DATA2丢弃后接收窗口不再移动

image-20241122205358541

3、4的序号未落入窗口，接收方将其丢弃。

image-20241122205444882

若数据分组未落入接收窗口，接收方丢弃该分组，并发送最近的数据分组的确认分组。


image-20241123095456724

image-20241123095549884

发送方收到0、1的确认分组，就将发送窗口向前滑动两个位置，这样就有新的序号5和6落入发送窗口。

 image-20241123100123937

发送方现在可将5号和6号数据分组发送出去，发送方收到针对1号数据分组的两个重复确认。发送方就知道了2号数据被丢弃，而之后的两个数据分组也被丢弃。

 image-20241123100549924

5、6未落入接收窗口，接收方将其丢弃，并发送1的确认分组。

 image-20241123100616937

我们假设发送方收到四个重复确认时仍然不会立刻重传，当2号数据分组发生超时，发送方就将序号落入发送窗口内的超时的2号数据分组和其后已发送的3~6号数据分组全部重传，尽管发送方之前已发送的3~6号数据分组到达接收方时并未出现误码，但是接收方只能接收序号落入接收窗口内的数据分组。一旦2号数据分组出现误码被接收方丢弃其后连续发送的3~6号数据分组都要被重传，即一旦出错就需要退回去重传已发送过的N个数据分组。

一个数据分组的差错就可能引起大量数据分组的重传。


在信道质量较差(容易出现误码)的情况下，回退N帧协议的信道利用率并不比停止-等待协议的信道利用率高。

回退N帧协议的接收方采用累积确认方式。

接收方不必对收到的每一个数据分组都发送一个确认分组，而是可以在收到几个序号连续的数据分组后，对按序到达的最后一个数据分组发送确认分组。

接收方何时发送累积确认分组，由具体实现决定。

确认分组ACKn表明序号为n及之前的所有数据分组都已正确接收。

 image-20241123102542050

发送5个数据分组，返回两个累积确认分组。

若ACK1丢失，发送窗口移动5个。

累积确认的优点

减少向网络中注入确认分组的数量。

即使确认分组丢失，也可能不必重传数据分组。


累积确认的缺点

不能向发送方及时准确地反映出接收方已正确接收的所有数据分组的数量。

 image-20241123103022396

发送窗口为一次发送的分组的个数，必然大于2。为什么不能为8呢？当发送窗口为8时，接收方接收，发送累积确认分组ACK7，发送途中丢失。发送方超时重传，接收方收到，但是它分辨不了这是重新发的（分组重复，不可靠传输直接丢弃），还是新发的分组（接收窗口后移），所以发送窗口不能为8。

 image-20241123103729743

 image-20241123104617543

累积确认分组，说明3之前包括3的数据被接收，发送窗口下滑动4个位置。重传4、5、6、7。


 image-20241123105821438

 image-20241123112116024

没看懂，二轮再学。

🔔 导出成功



发送方

0	1	2	
---	---	---	--

接收方



$W_R = 1$

0	1	2	
---	---	---	--