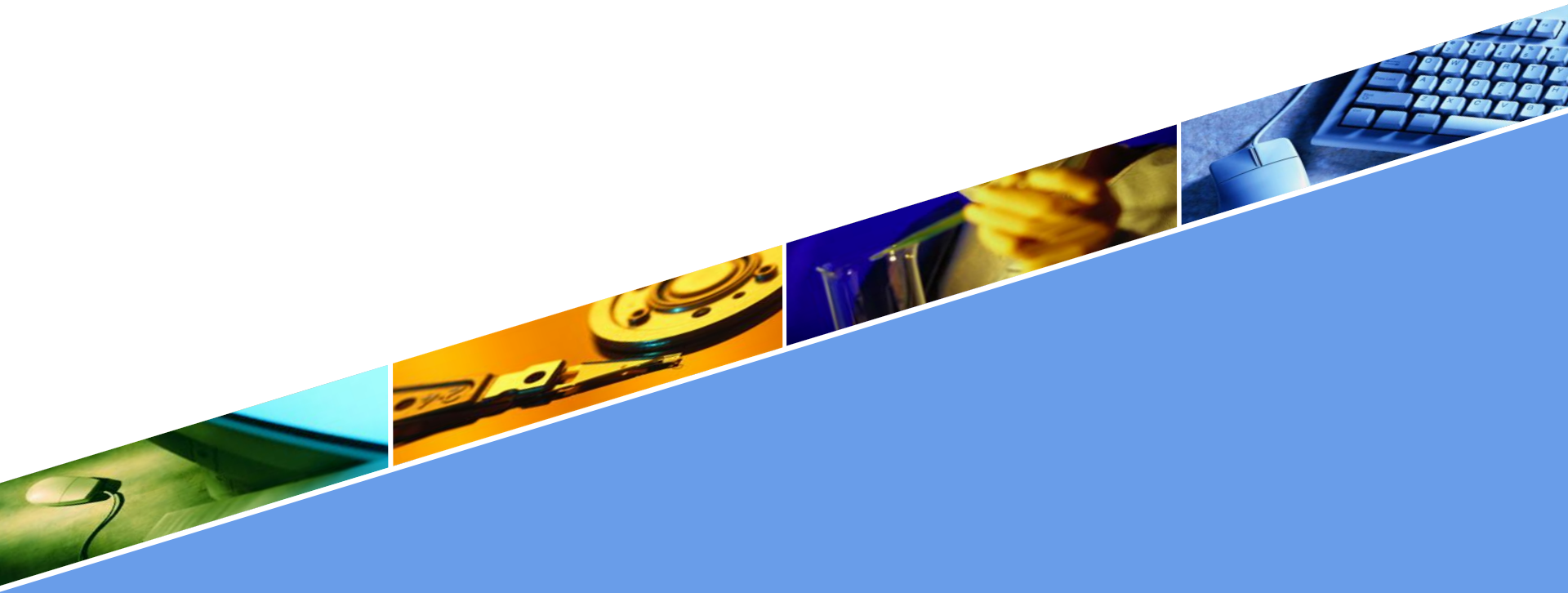
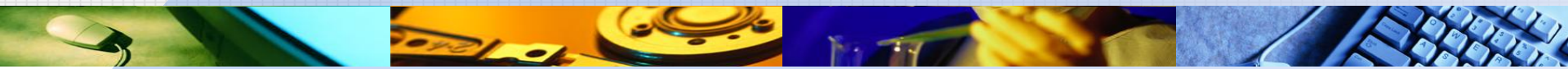
运输层协议与技术



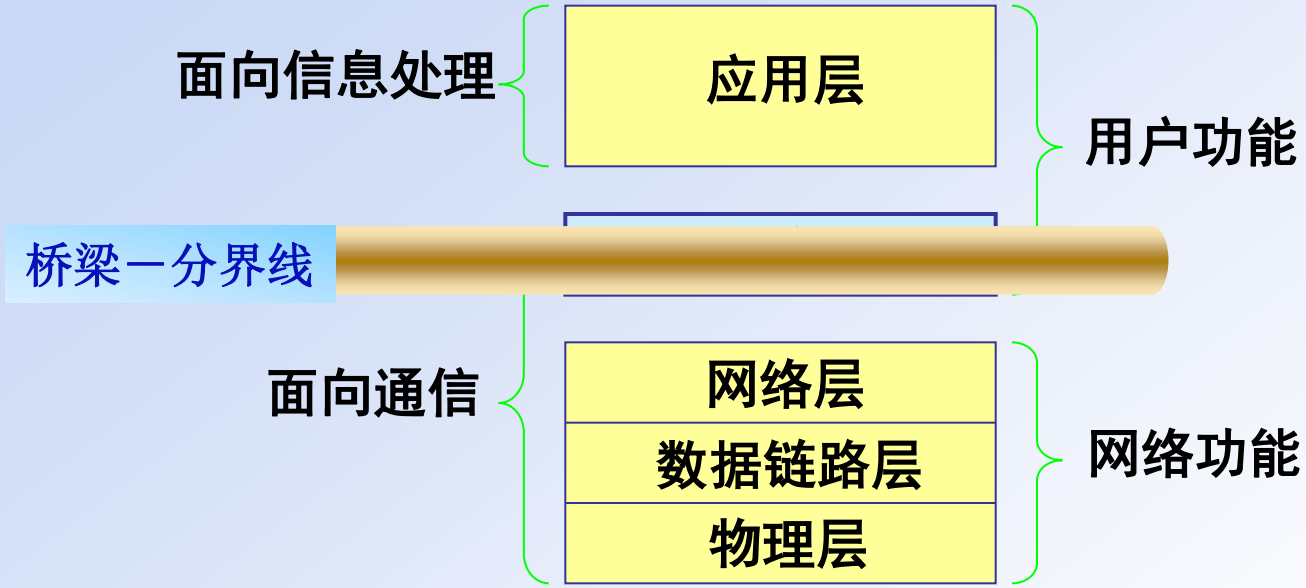
主题 1



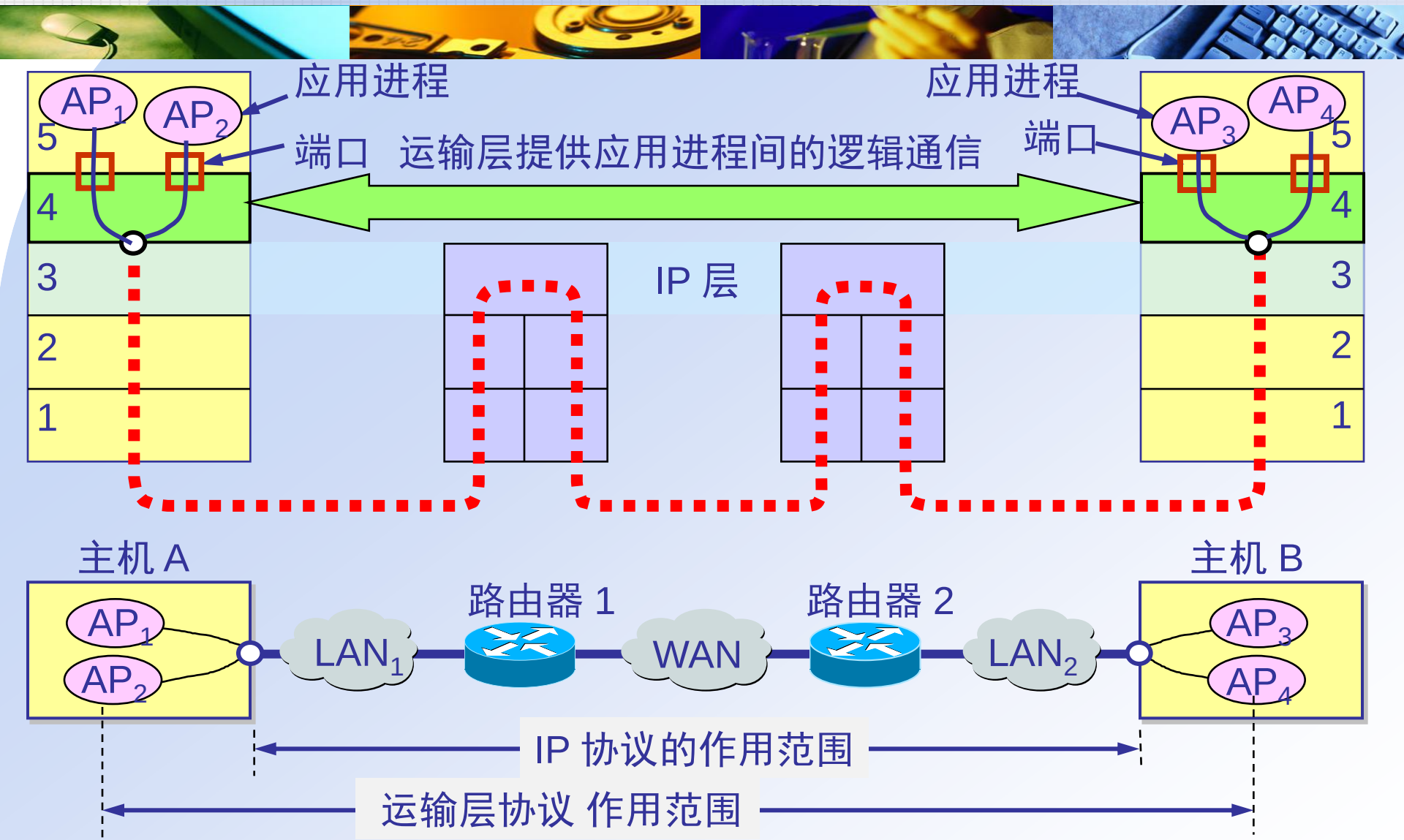
1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. TCP的流量控制与拥塞控制概述

基本概念

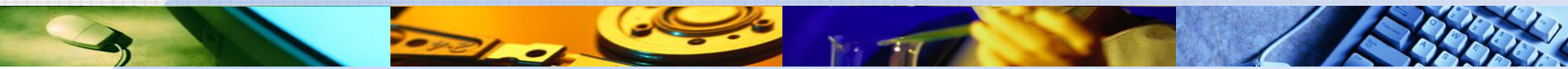
从通信和信息处理的角度看，运输层向它上面的应用层提供通信服务，它属于面向通信部分的最高层，同时也是用户功能中的最低层。

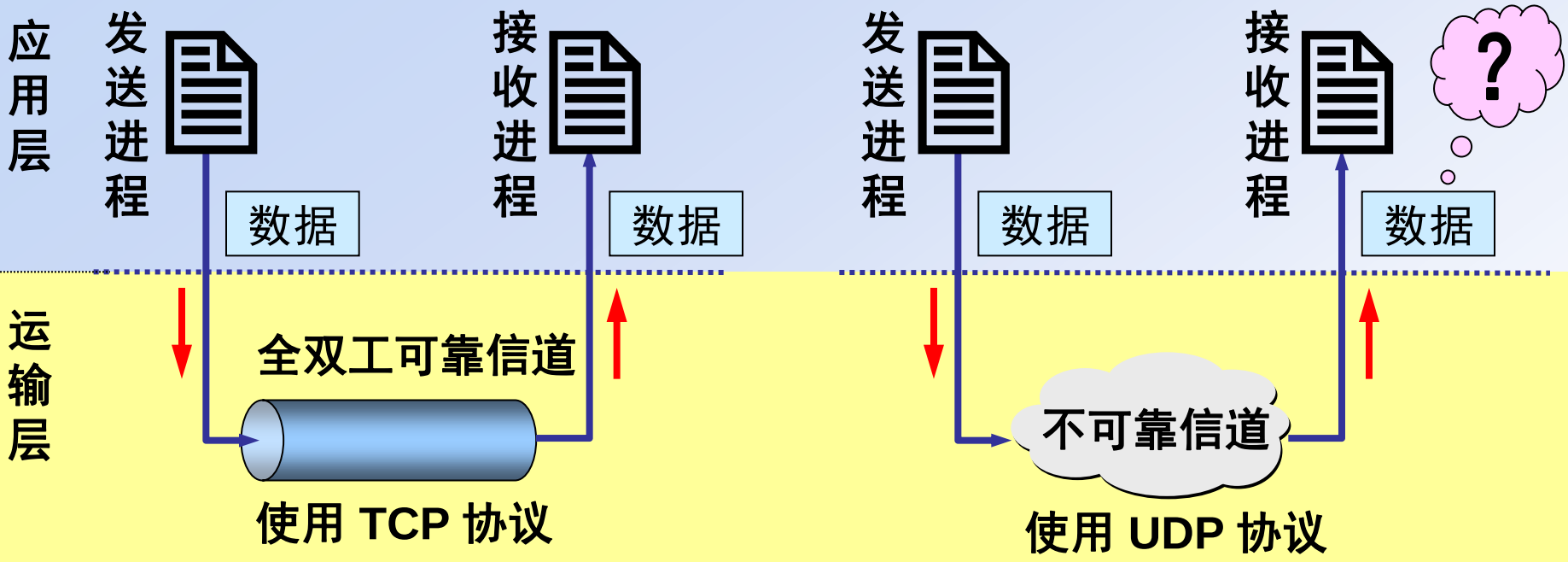


运输层为相互通信的应用进程提供了逻辑通信



基本概念—两种协议




应用层
发送进程
数据
接收进程
数据
运输层
全双工可靠信道
使用 TCP 协议
发送进程
数据
接收进程
?
数据
不可靠信道
使用 UDP 协议

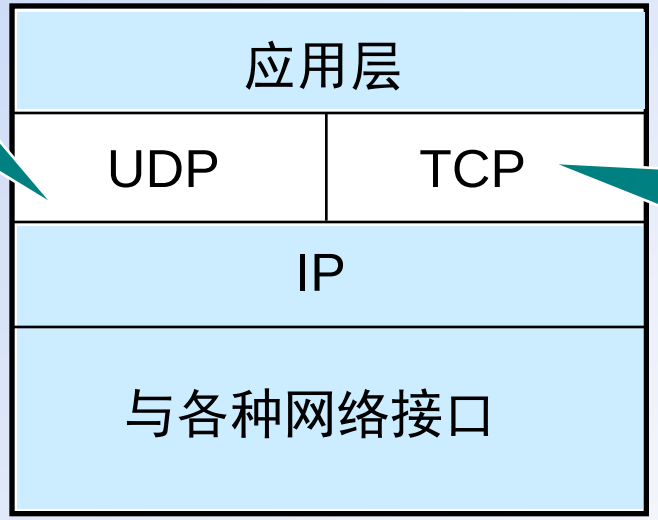
基本概念—两种协议

TCP/IP的运输层有两个不同的协议：

(1) 用户数据报协议 UDP (User Datagram Protocol)

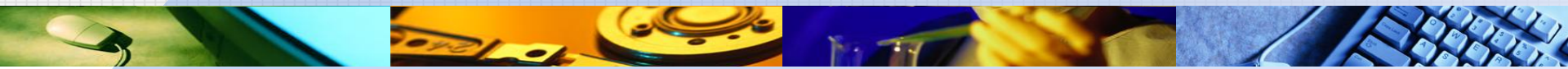
(2) 传输控制协议 TCP (Transmission Control Protocol)

① 无连接—不可靠
② 小数据-广播
③ 管控应用协议
④ 资源消耗小
⑤ 数据包简单



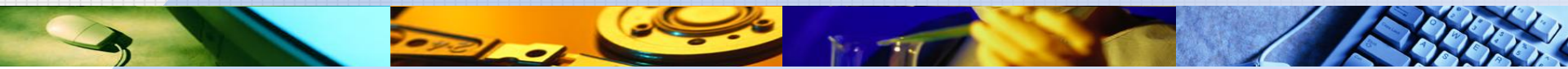
① 连接—可靠
② 非广播类
③ 资源消耗大
④ 数据包复杂

TCP 与 UDP基本概念



- 两个对等运输实体在通信时传送的数据单位叫作运输协议数据单元 TPDU (Transport Protocol Data Unit)。
- TCP 传送的协议数据单元是 TCP 报文段 (segment)
- UDP 传送的协议数据单元是 UDP 报文或用户数据报。

运输层两种协议与IP协议对比



	连接	传输通道	服务对象	寻址	寻址承担者
IP寻址	无	网络中的路由器	UDP或TCP	IP地址	路由器：网络
UDP	无	端到端抽象通道：路由器透明	应用层	端口	端设备
TCP	有	端到端虚电路：路由器透明	应用层	端口	端设备

需要UDP why？TCP不好吗？

① 应用层协议需要，如网络管理协议、DNS等

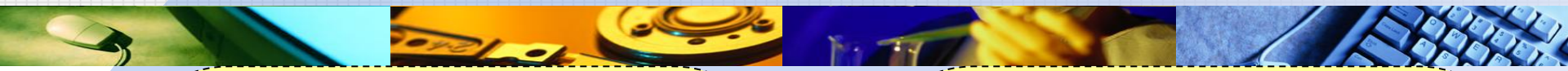
② 数据量少

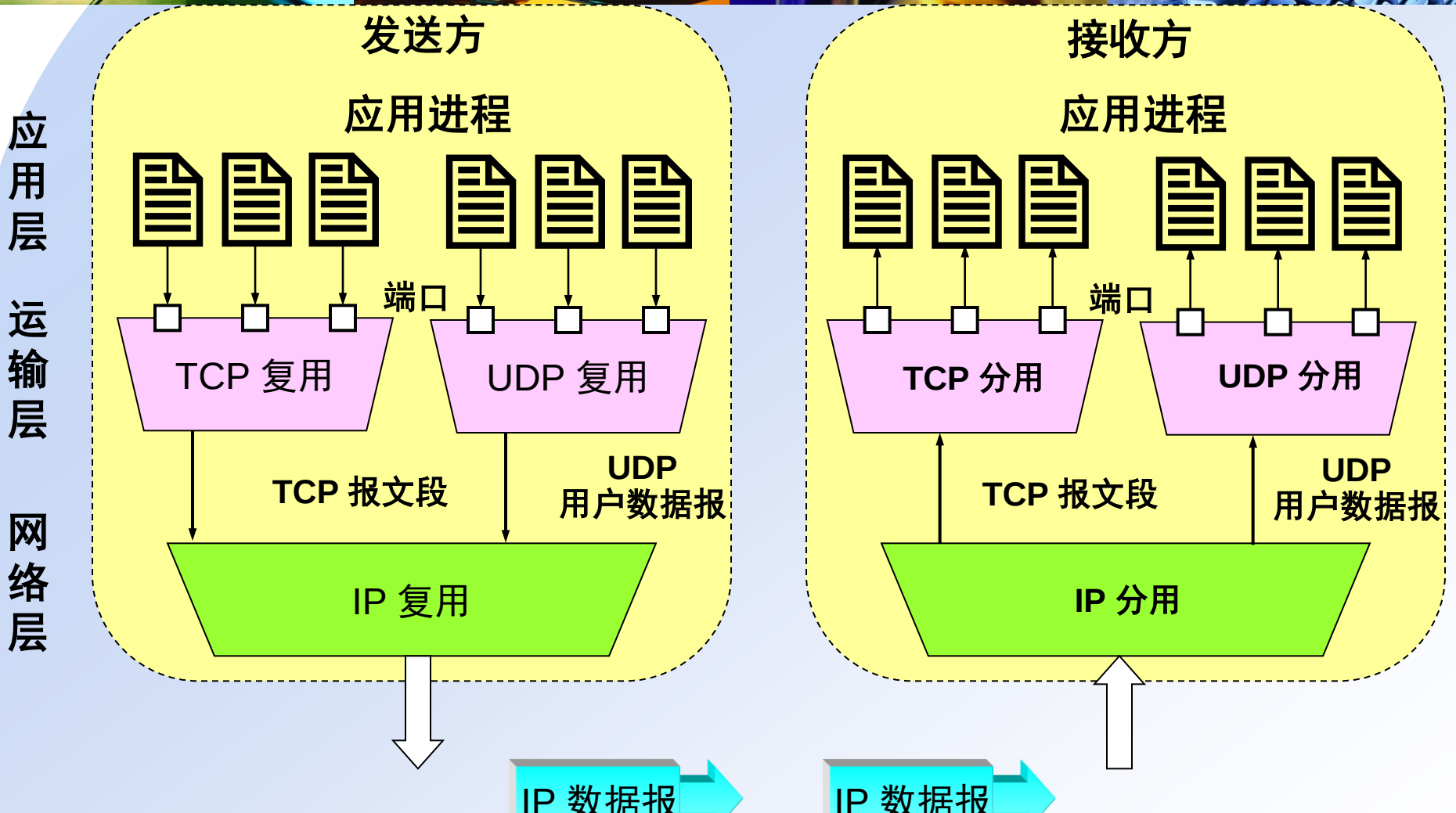
③ 控制或管理协议，不应占用过多网络资源

运输层的端口

- 端口就是运输层服务访问点 TSAP。
- 端口的作用就是让应用层的各种应用进程都能将其数据通过端口向下交付给运输层，以及让运输层知道应当将其报文段中的数据向上通过端口交付给应用层相应的进程。
- 计算机“端口”是计算机与外界通讯交流的出口，分为软件端口和硬件端口：
  - 硬件领域的端口又称接口，如：USB端口、串行端口等。
  - 软件端口一般指协议栈层间的抽象的协议端口，是一种抽象的软件结构，包括一些数据结构和I/O（基本输入输出）缓冲区。

基本概念—端口



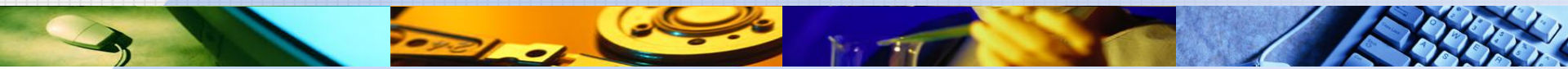


应用层

运输层

网络层

端口标识



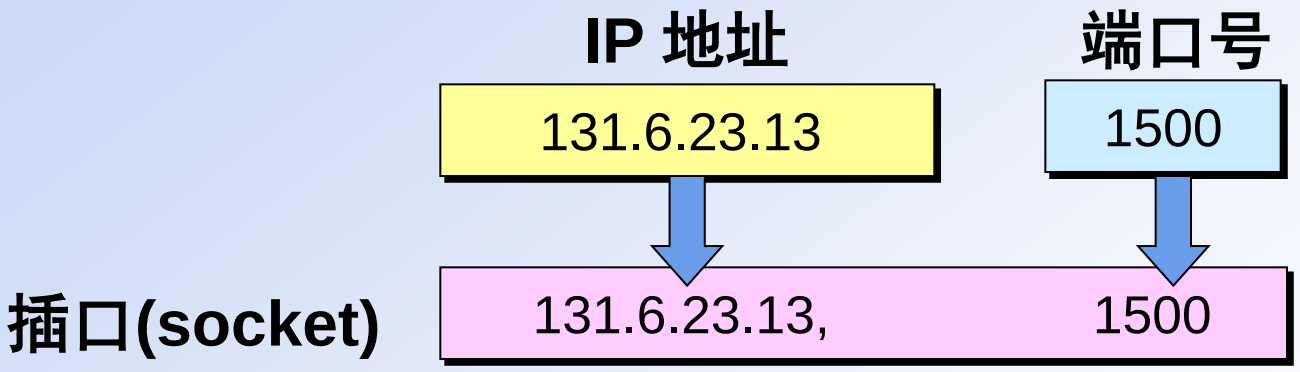
- 端口用一个 16 bit 端口号进行标志。
- 端口号只具有本地意义，即端口号只是为了标志本计算机应用层中的各进程。在因特网中不同计算机的相同端口号是没有联系的。

三类端口

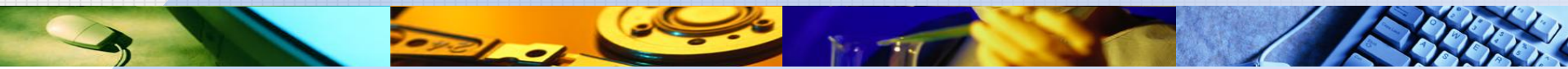
- **熟知端口**，服务器端使用，数值一般为 0~1023。
- **登记端口号**，服务器端使用，数值为 1024~49151，为没有熟知端口号的应用程序使用的。使用这个范围的端口号必须在 IANA 登记，以防止重复。
- **客户端口号**或**短暂端口号**，数值为 49152~65535，留给客户进程选择暂时使用。当服务器进程收到客户进程的报文时，就知道了客户进程所使用的动态端口号。通信结束后，这个端口号可供其他客户进程以后使用。

基本概念—插口Socket（套接字）

- TCP 使用“连接”(而不仅仅是“端口”)作为最基本的抽象，同时将 TCP 连接的端点称为插口(socket)，或套接字。
- 插口和端口、IP 地址的关系是：



主题 2

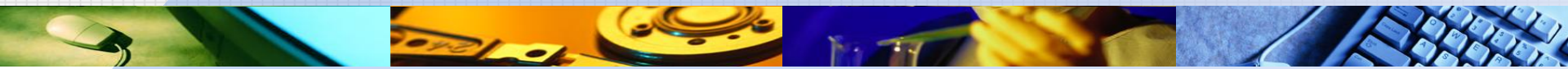


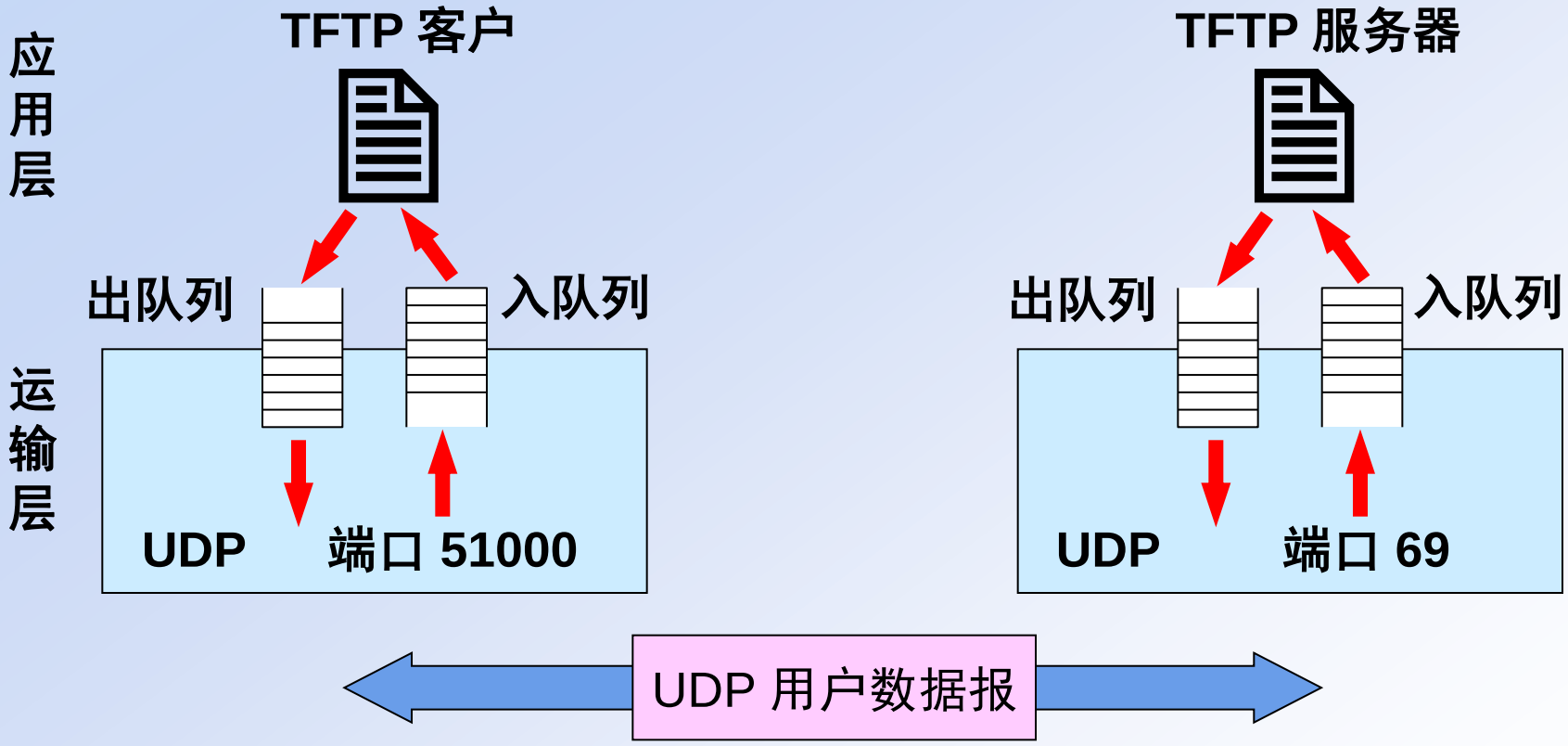
1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. TCP的流量控制与拥塞控制

UDP协议

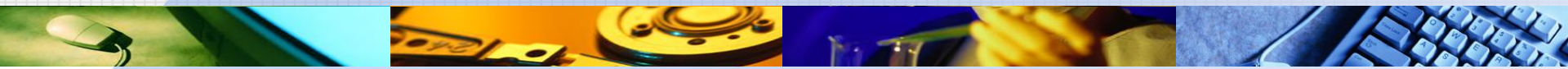
- UDP 只在 IP 的数据报服务之上增加了很少一点的功能，即端口的功能和差错检测的功能。
- 虽然UDP用户数据报只能提供不可靠的交付，但 UDP 在某些方面有其特殊的优点。
  - ① 发送数据之前不需要建立连接；
  - ② UDP 的主机不需要维持复杂的连接状态表；
  - ③ UDP 用户数据报只有8个字节的首部开销；
  - ④ 网络出现的拥塞不会使源主机的发送速率降低；这对某些实时应用是很重要的。

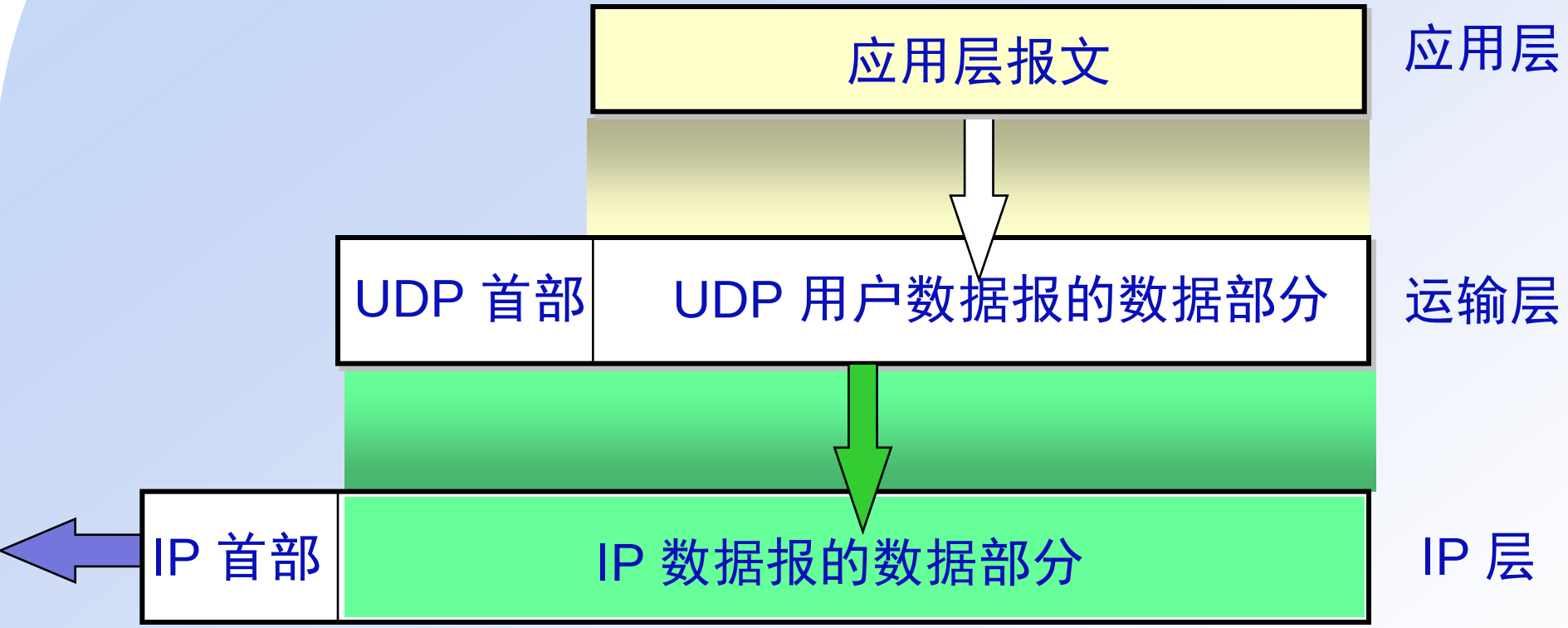
UDP协议—端口与报文队列



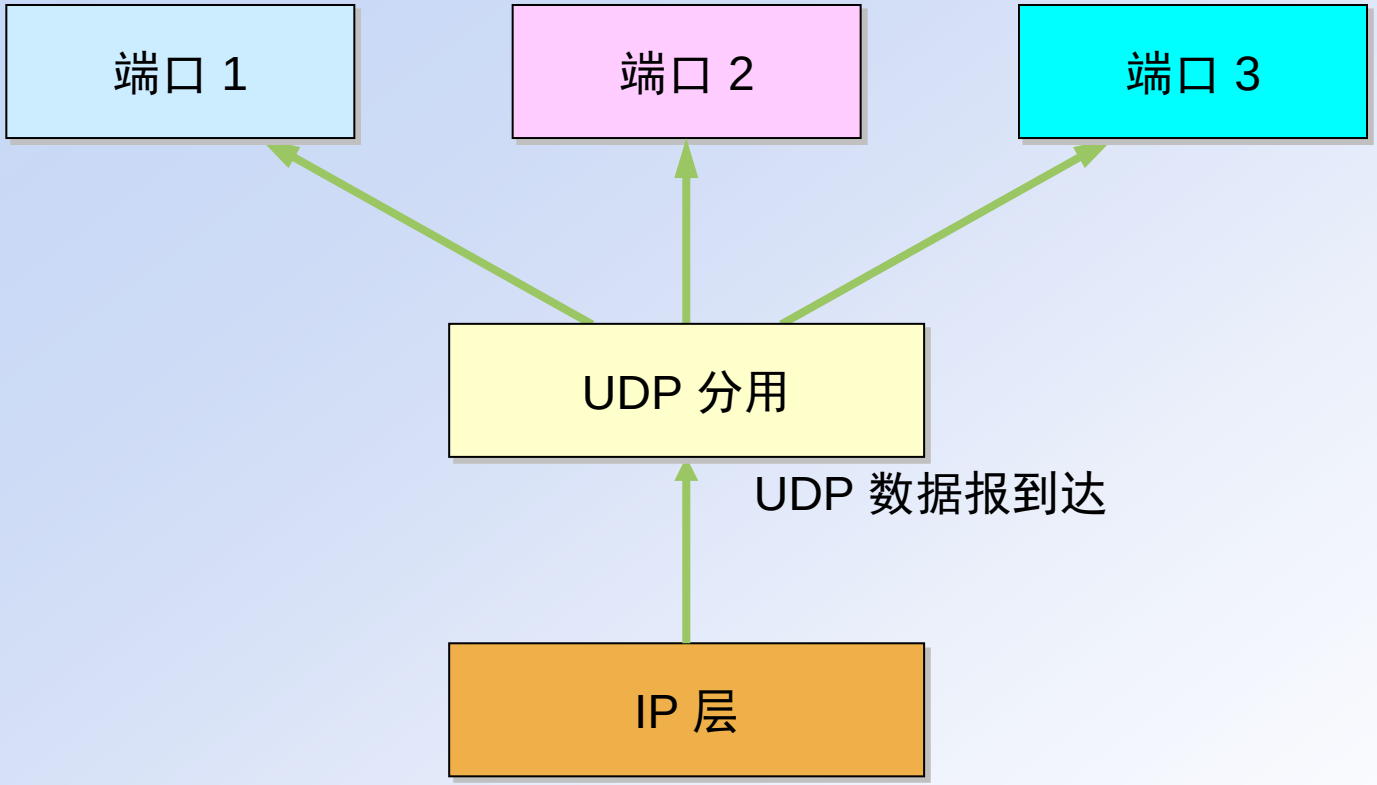


UDP 是面向报文的

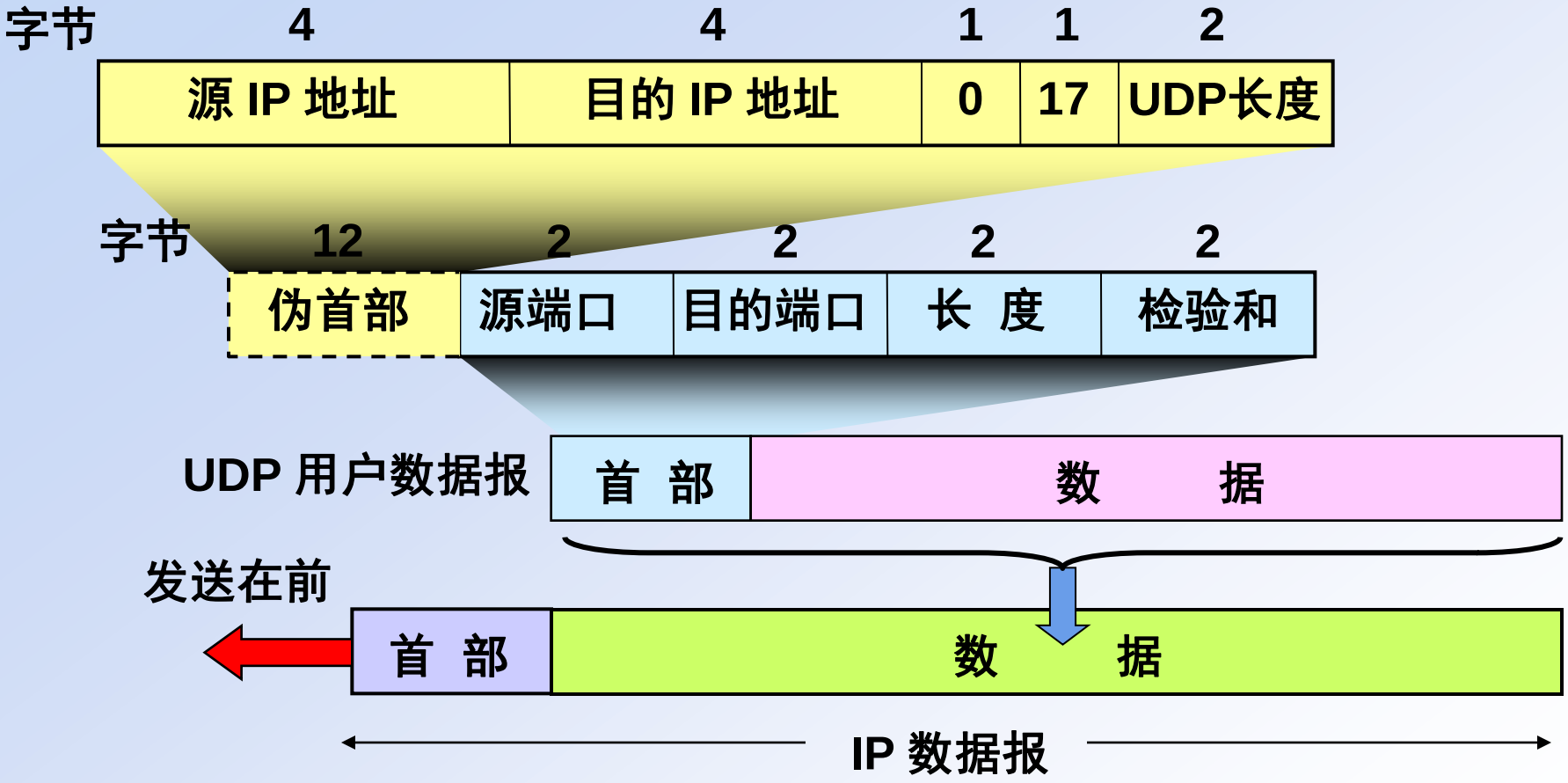




UDP 基于端口的分用

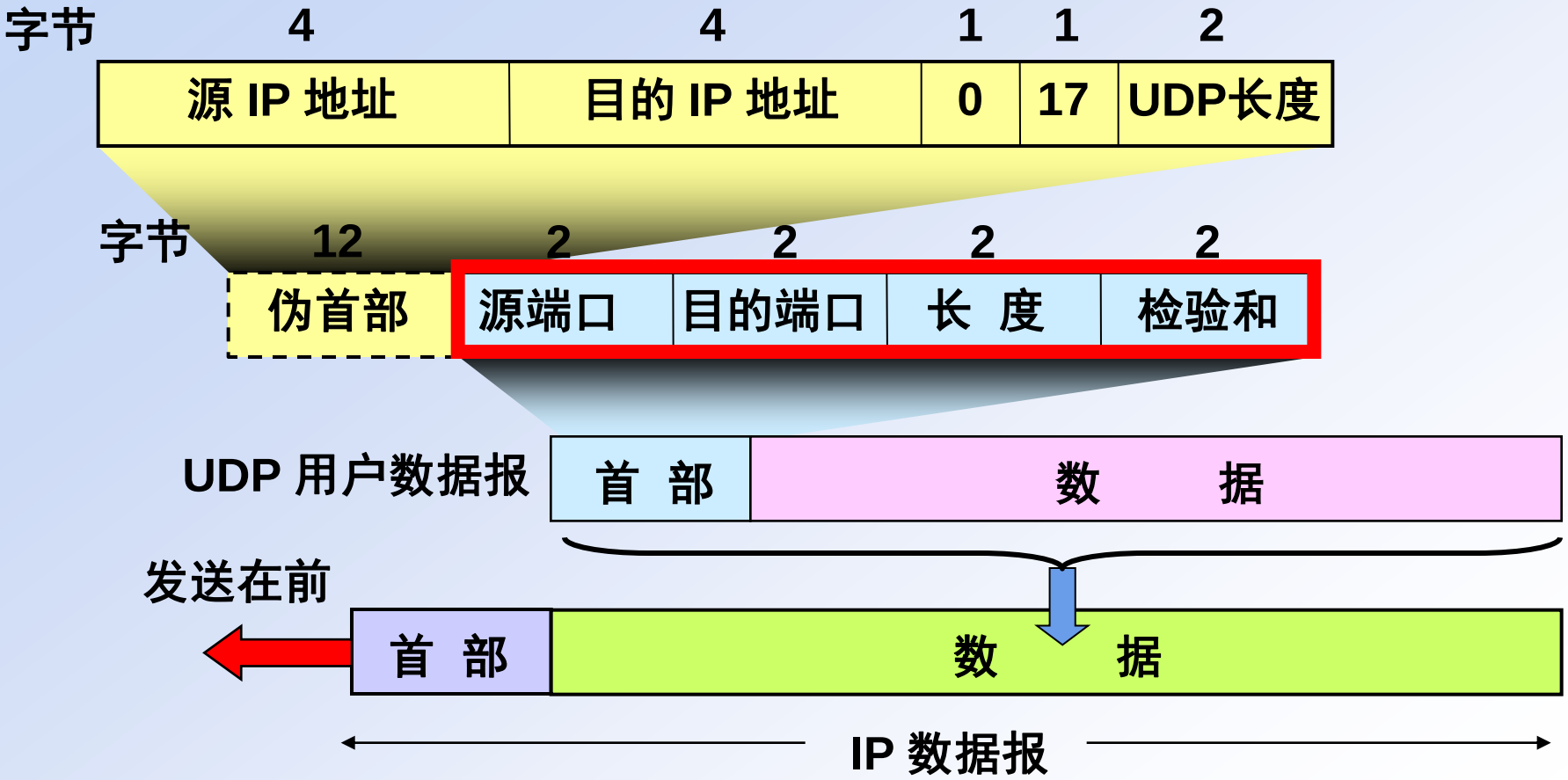


UDP协议—报文结构


字节
4
4
1
1
2
源 IP 地址
目的 IP 地址
0
17
UDP长度
字节
12
2
2
2
2
伪首部
源端口
目的端口
长 度
检验和
UDP 用户数据报
首 部
数 据
发送在前
首 部
数 据
IP 数据报

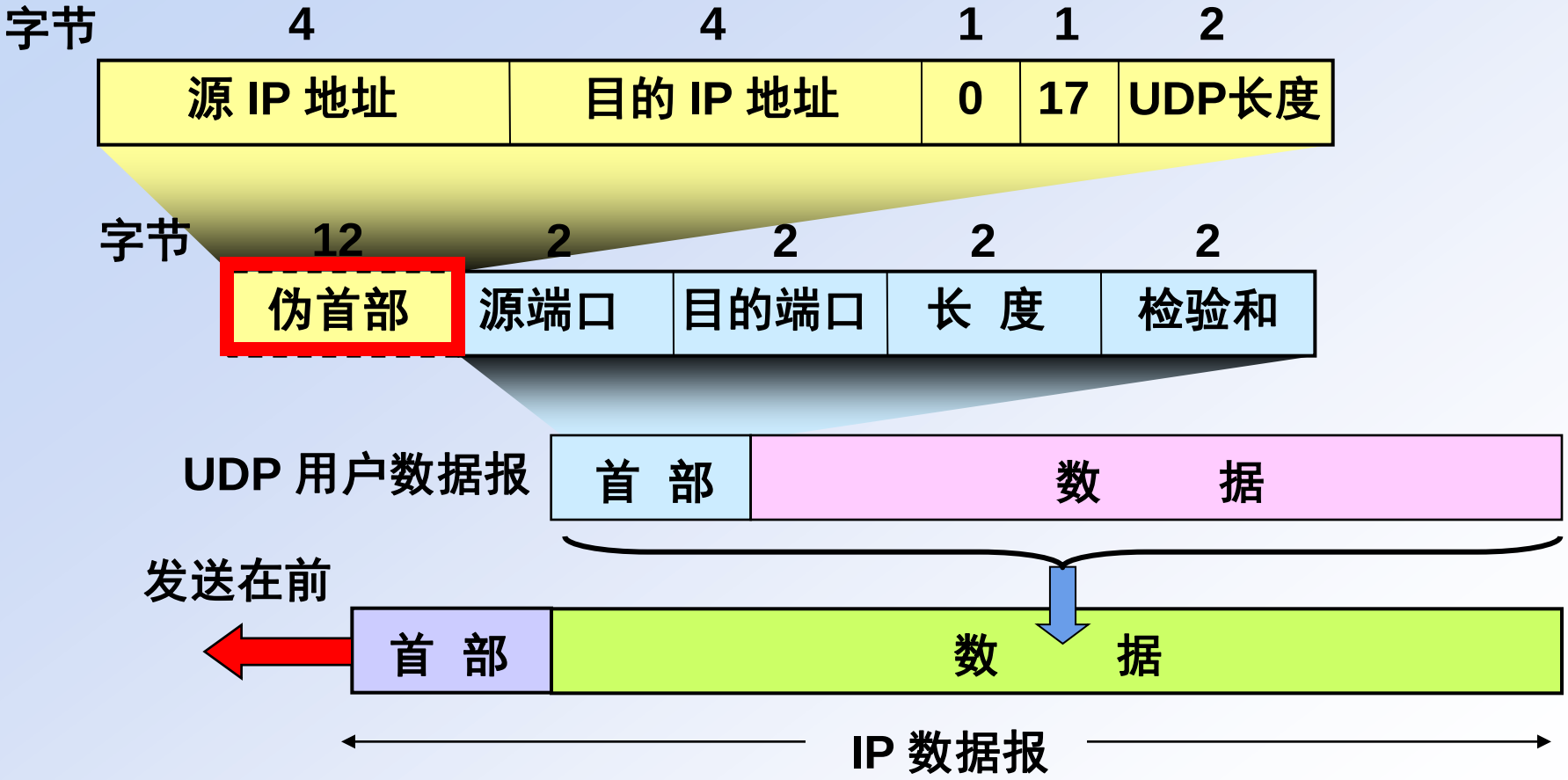
UDP协议—首部格式

用户数据报 UDP 有两个字段：数据字段和首部字段。首部字段有 8 个字节，由 4 个字段组成，每个字段都是两个字节。

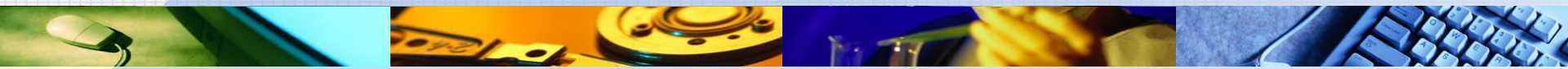


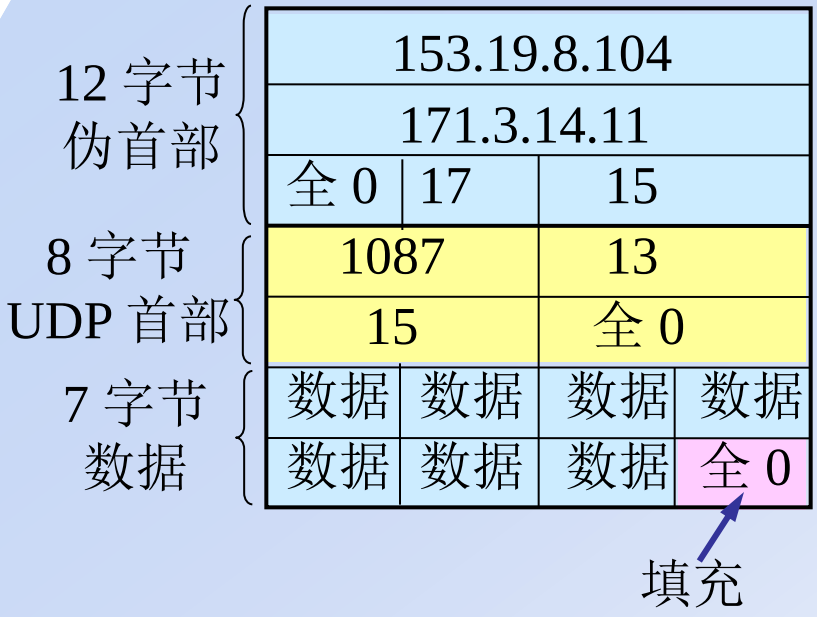
UDP协议—首部格式

- 在计算检验和时，临时把“伪首部”和 UDP 用户数据报连接在一起。伪首部仅仅是为了计算检验和。



计算 UDP 检验和的例子



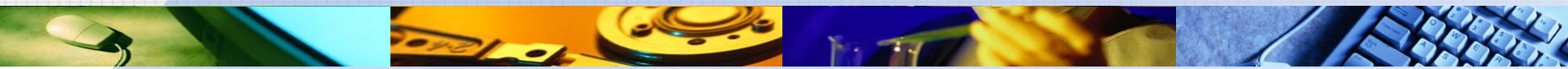


10011001 00010011 → 153.19
00001000 01101000 → 8.104
10101011 00000011 → 171.3
00001110 00001011 → 14.11
00000000 00010001 → 0 和 17
00000000 00001111 → 15
00000100 00111111 → 1087
00000000 00001101 → 13
00000000 00001111 → 15
00000000 00000000 → 0（检验和）
01010100 01000101 → 数据
01010011 01010100 → 数据
01001001 01001110 → 数据
01000111 00000000 → 数据和 0（填充）

按二进制反码运算求和 10010110 11101101 → 求和得出的结果

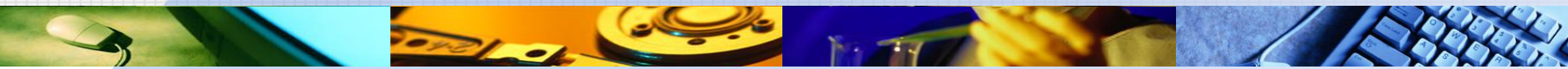
将得出的结果求反码 01101001 00010010 → 检验和

主题 3



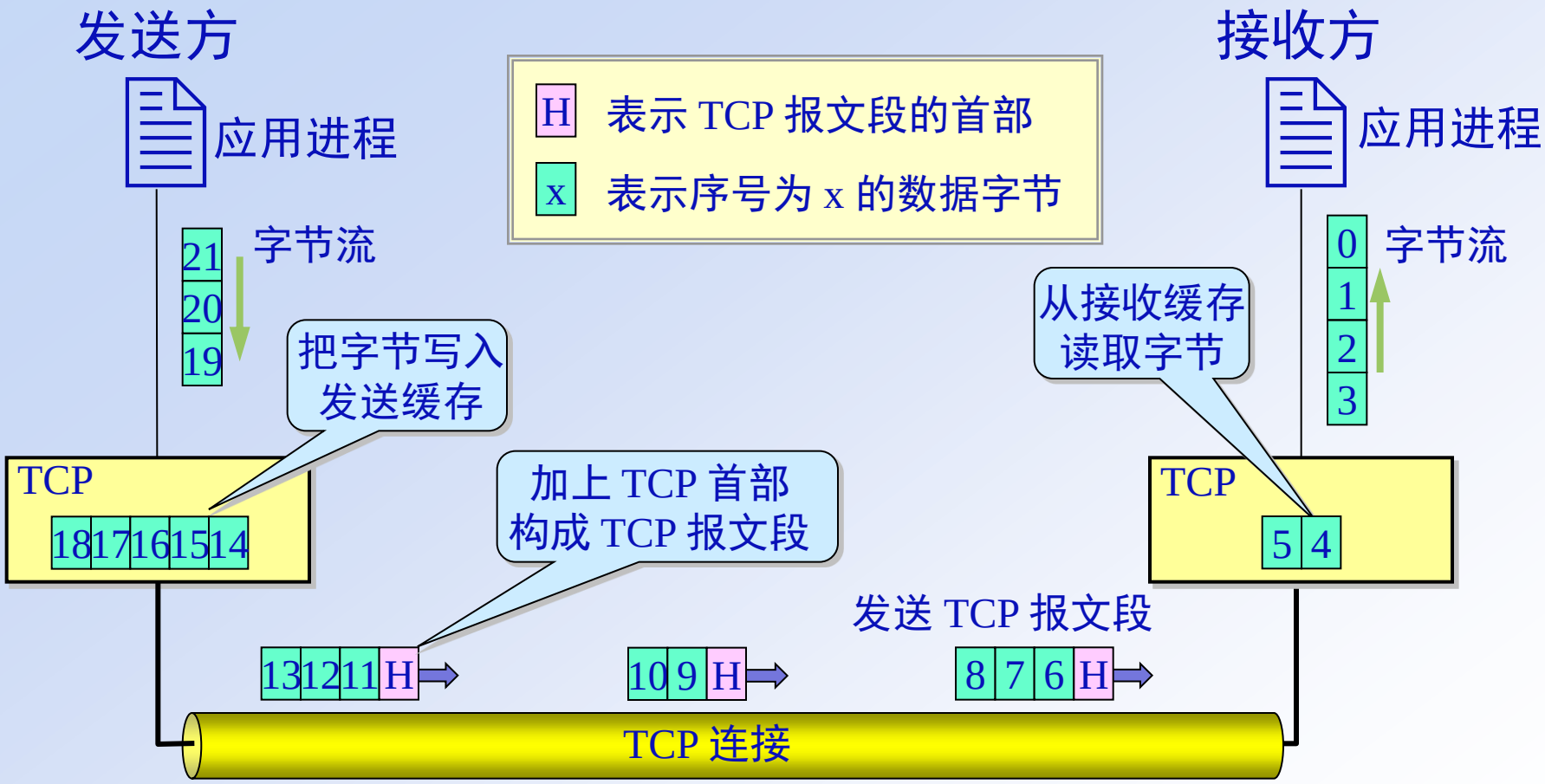
1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. TCP的流量控制与拥塞控制

TCP 最主要的特点

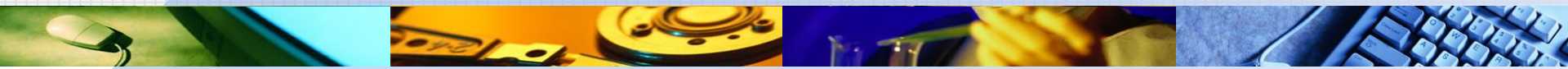


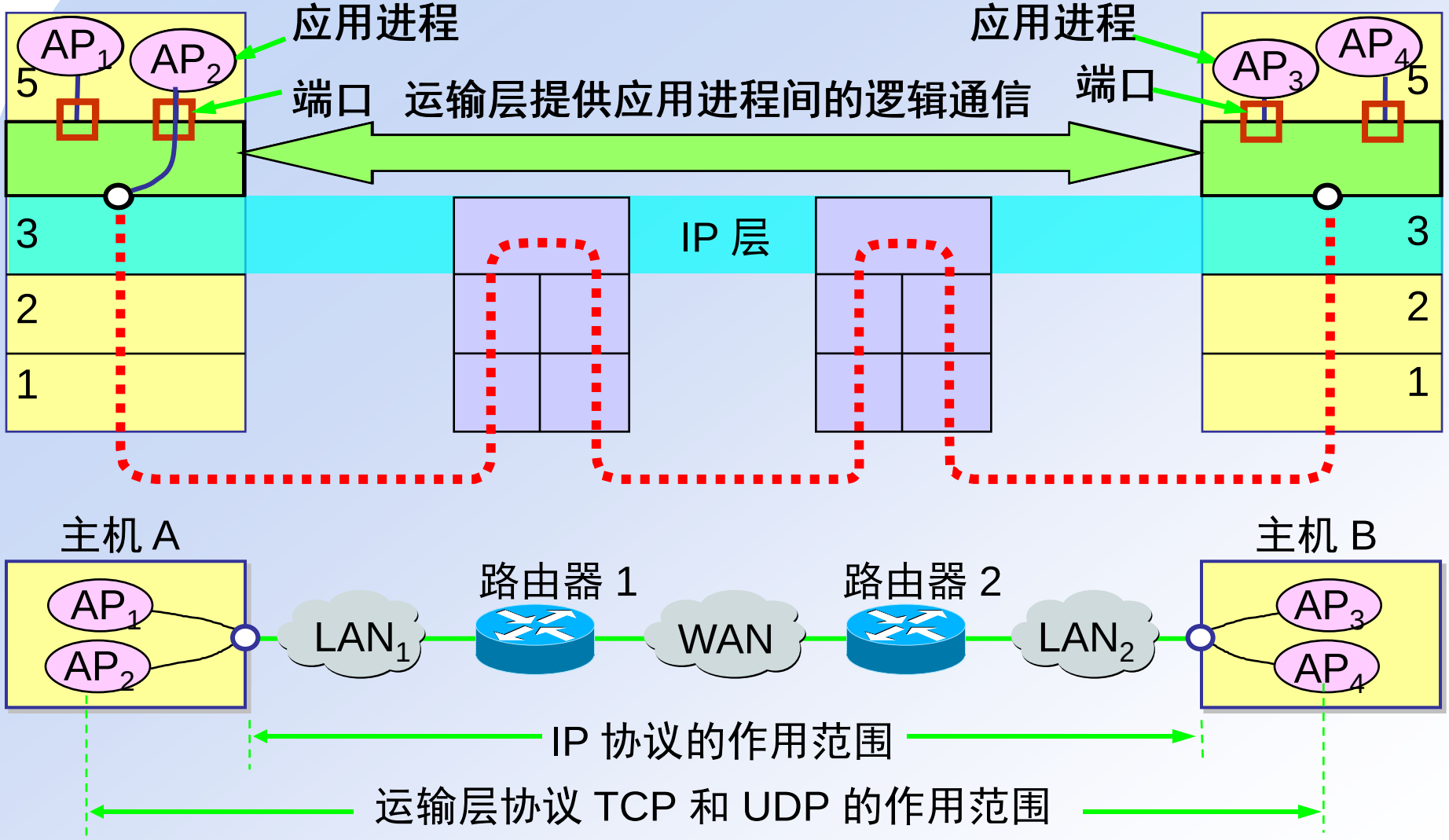
- TCP 是面向连接的运输层协议。
- 每一条 TCP 连接只能有两个端点(endpoint)，每一条 TCP 连接只能是点对点的（一对一）。
- TCP 提供可靠交付的服务。
- TCP 提供全双工通信。
- 面向字节流。

TCP 面向流的概念

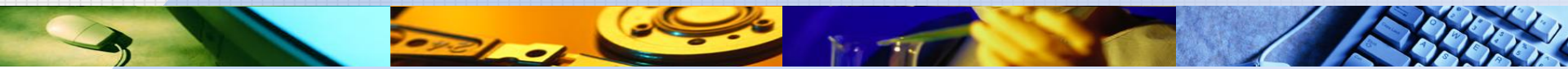

发送方
应用进程
字节流
21
20
19
把字节写入
发送缓存
TCP
18 17 16 15 14
H 表示 TCP 报文段的首部
x 表示序号为 x 的数据字节
加上 TCP 首部
构成 TCP 报文段
13 12 11 H
10 9 H
8 7 6 H
发送 TCP 报文段
TCP 连接
接收方
应用进程
字节流
0
1
2
3
从接收缓存
读取字节
TCP
5 4

TCP工作方式



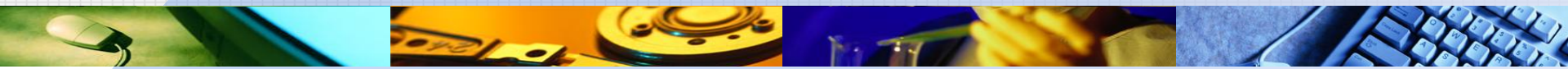

应用进程
应用进程
AP1
AP2
AP3
AP4
端口
端口
运输层提供应用进程间的逻辑通信
5
5
3
2
1
IP 层
3
2
1
主机 A
主机 B
路由器 1
路由器 2
AP1
AP2
LAN1
WAN
LAN2
AP3
AP4
IP 协议的作用范围
运输层协议 TCP 和 UDP 的作用范围

应当注意



- TCP 连接是一条虚连接而不是一条真正的物理连接。
- TCP 对应用进程一次把多长的报文发送到TCP 的缓存中是不关心的。
- TCP 根据对方给出的窗口值和当前网络拥塞的程度来决定一个报文段应包含多少个字节（UDP 发送的报文长度是应用进程给出的）。
- TCP 可把太长的数据块划分短一些再传送。TCP 也可等待积累有足够多的字节后再构成报文段发送出去。

TCP 的连接



- TCP 把连接作为最基本的抽象。
- 每一条 TCP 连接有两个端点。
- TCP 连接的端点不是主机，不是主机的IP 地址，不是应用进程，也不是运输层的协议端口。TCP 连接的端点叫做**套接字**(socket)或**插口**。
- 端口号**拼接到**(contatenated with) IP 地址即构成了套接字。

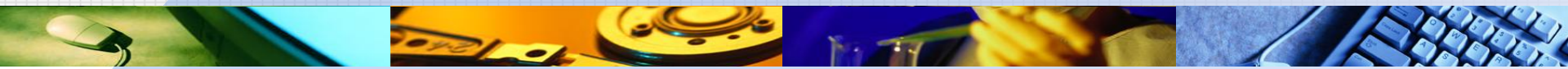
套接字（socket）

套接字 socket = (IP地址: 端口号)　　(5-1)

- 每一条 TCP 连接唯一地被通信两端的两个端点（即两个套接字）所确定。即：

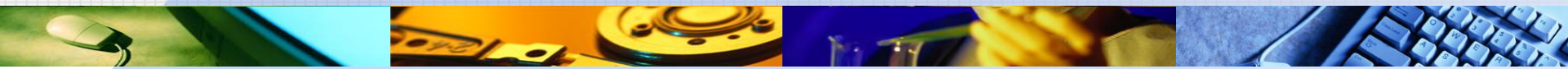
TCP 连接 ::= {socket1, socket2}
= {(IP1: port1), (IP2: port2)}　　(5-2)

同一个名词 socket有多种不同的意思



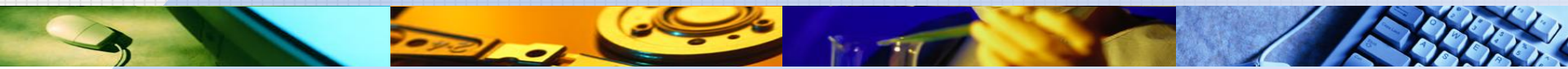
- 应用编程接口API称为 socket API，简称为 socket。
- socket API 中使用的一个函数名也叫作 socket。
- 调用 socket 函数的端点称为 socket。
- 调用 socket 函数时其返回值称为 socket 描述符，可简称为 socket。
- 在操作系统内核中连网协议的 Berkeley 实现，称为 socket 实现。

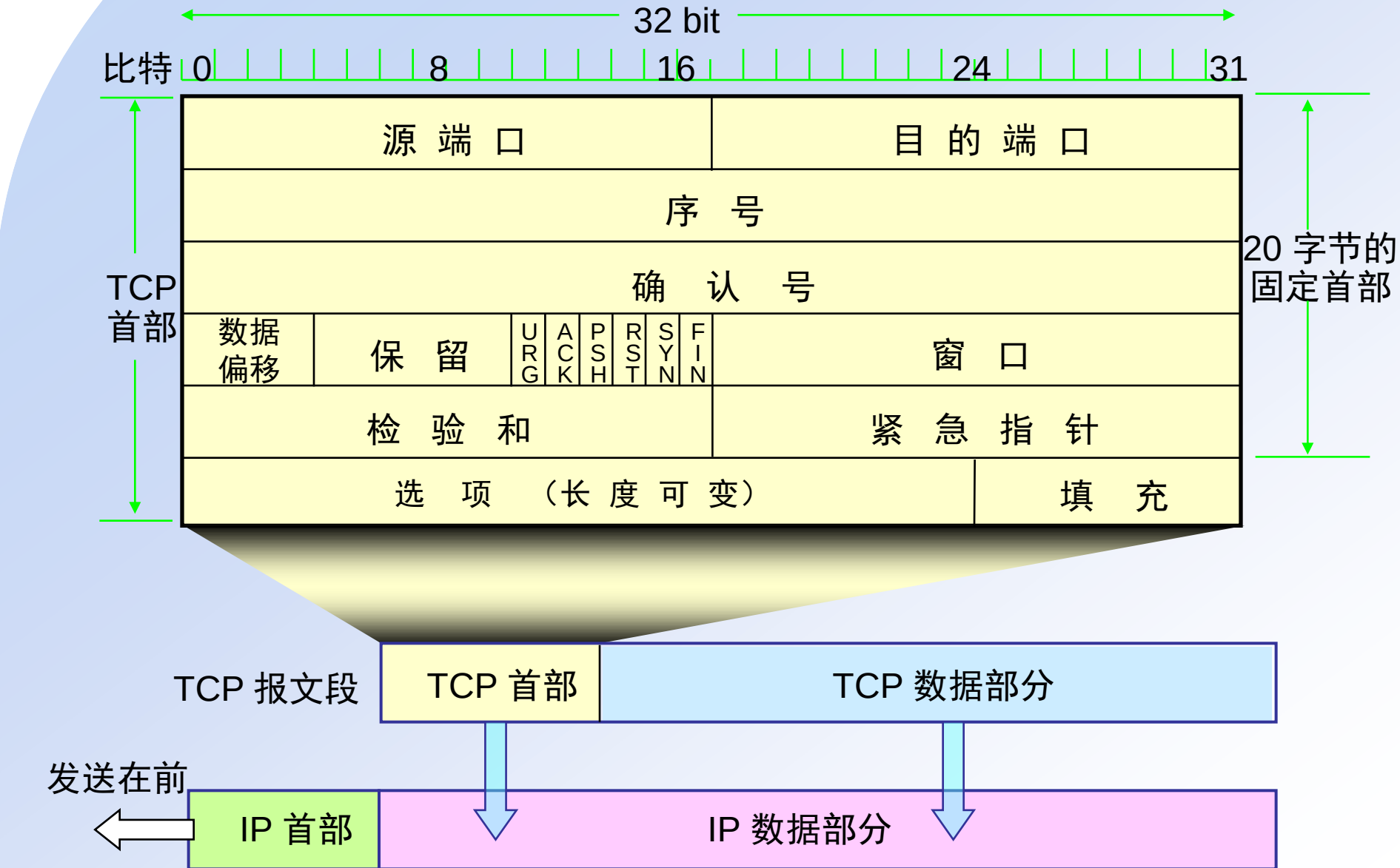
主题 4



1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. TCP的流量控制与拥塞控制

TCP数据报文结构

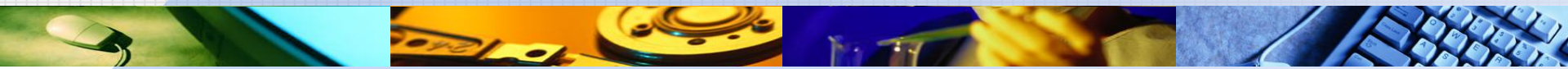


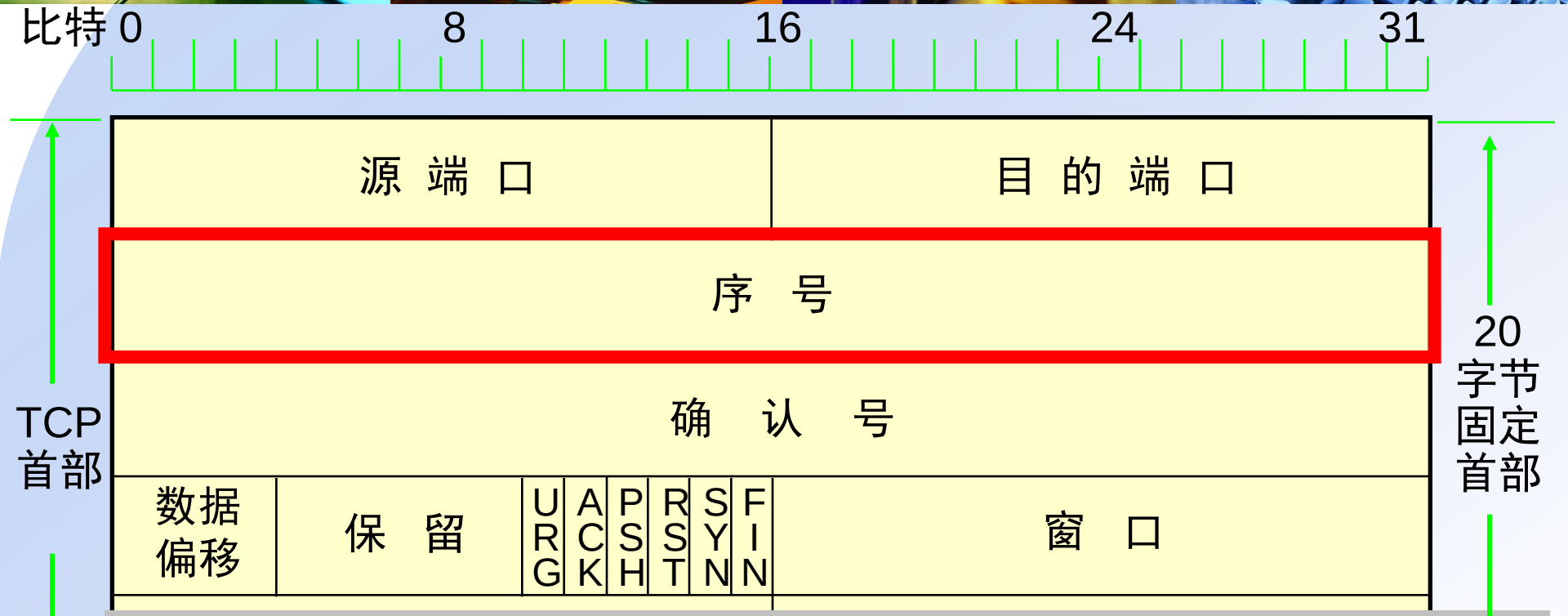

32 bit
比特
0
8
16
24
31
源 端 口
目 的 端 口
序 号
确 认 号
数据偏移
保 留
URG
ACK
PSH
RST
SYN
FIN
窗 口
检 验 和
紧 急 指 针
选 项 （长 度 可 变）
填 充
TCP 首部
20 字节的固定首部
TCP 报文段
TCP 首部
TCP 数据部分
发送在前
IP 首部
IP 数据部分

TCP数据报文结构

- **源端口和目的端口字段——各占 2 字节。端口是运输层与应用层的服务接口。运输层的复用和分用功能都要通过端口才能实现。**

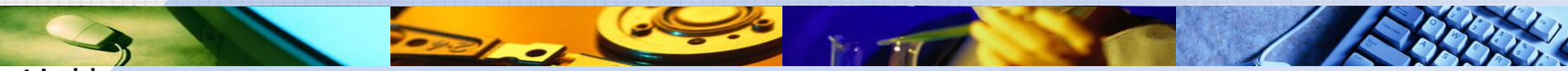
TCP数据报文结构

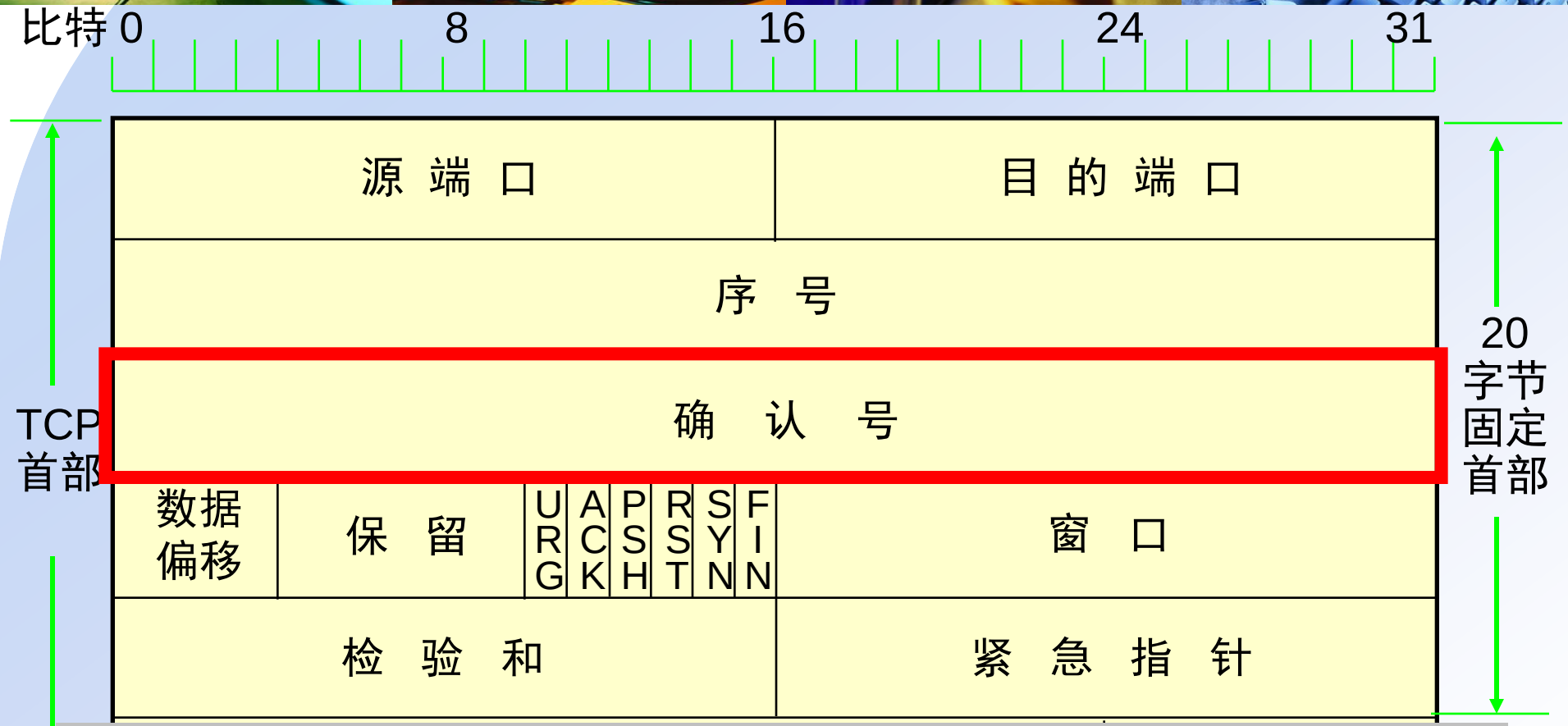




- **序号字段——占 4 字节。TCP 连接中传送的数据流中的每一个字节都编上一个序号。序号字段的值则指的是本报文段所发送的数据的第一个字节的序号。**

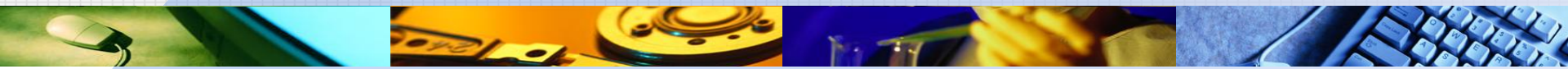
TCP数据报文结构





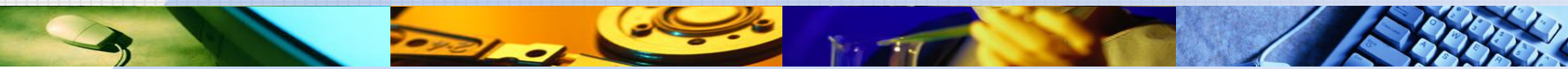
- **确认号字段——占 4 字节，是期望收到对方的下一个报文段的数据的第一个字节的序号。**

TCP数据报文结构



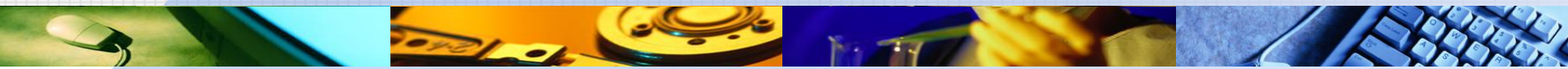
- **数据偏移——占 4 bit，它指出 TCP 报文段的数据开始处距离 TCP 报文段的起始处有多远，实际是首部长度。“数据偏移”的单位不是字节而是 32 bit 字（4 字节为计算单位）。**

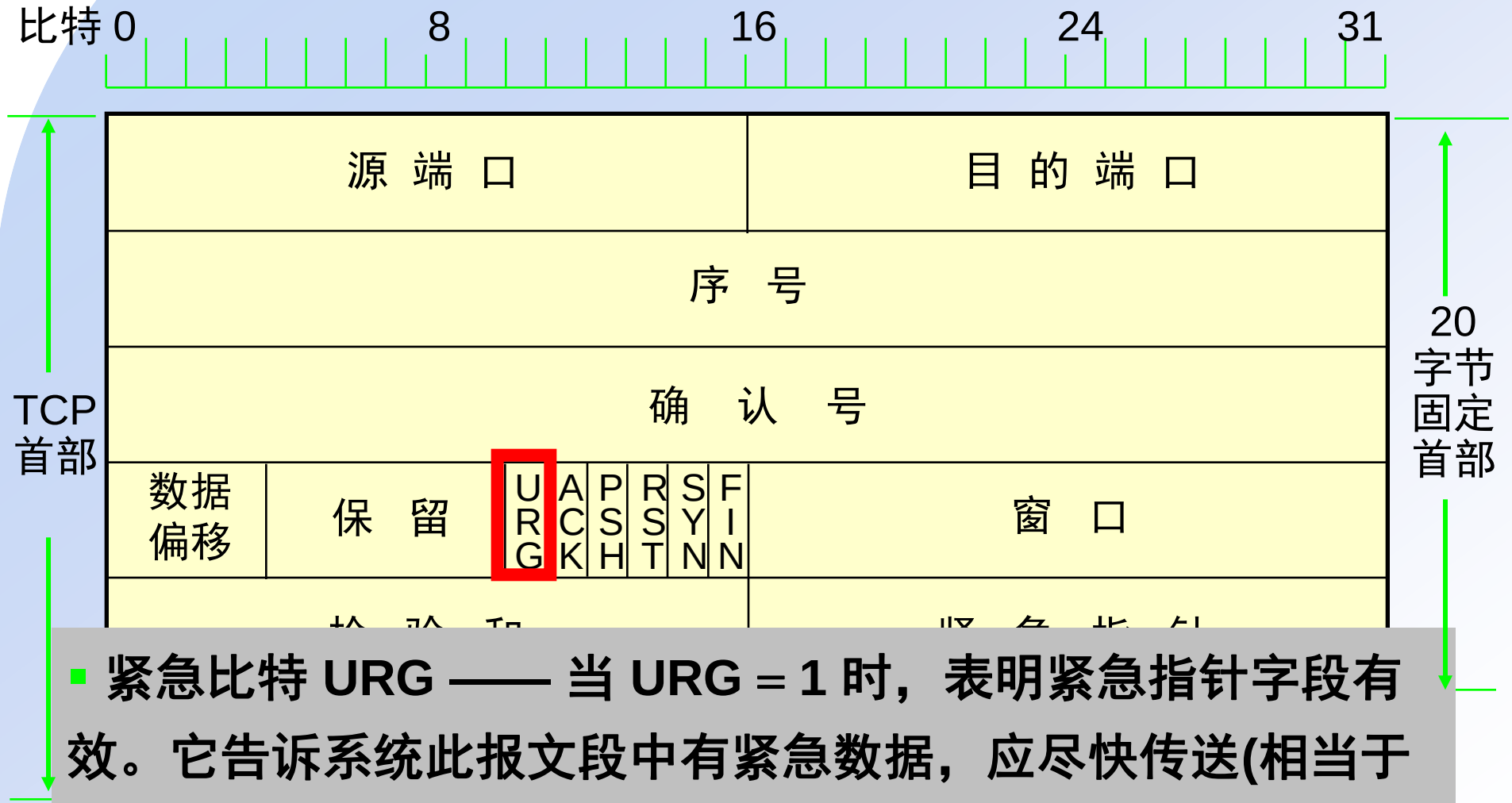
TCP数据报文结构



- **保留字段——占 6 bit，保留为今后使用，但目前应置为 0。**

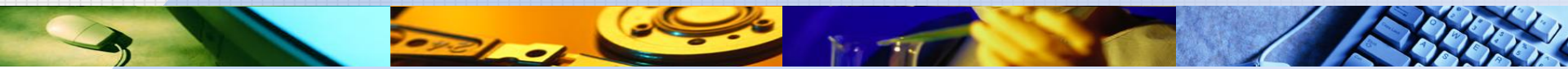
TCP数据报文结构





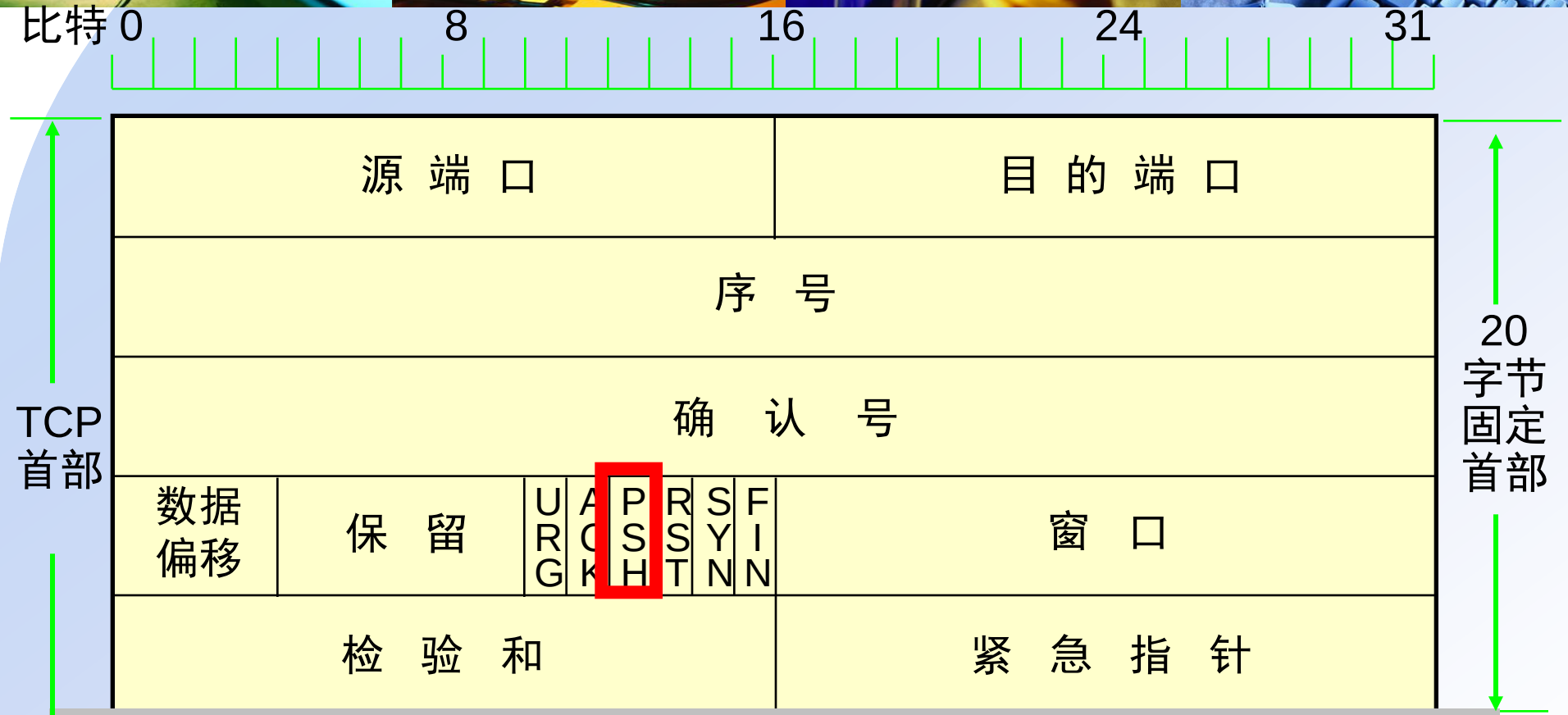
- **紧急比特 URG —— 当 URG = 1 时，表明紧急指针字段有效。它告诉系统此报文段中有紧急数据，应尽快传送(相当于高优先级的数据)。**

TCP数据报文结构



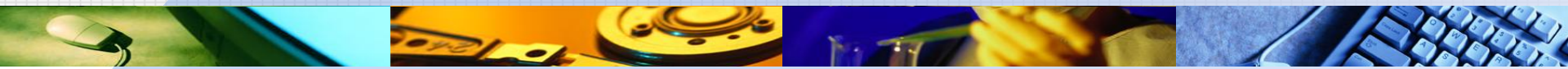
- **确认比特 ACK —— 只有当 ACK = 1 时确认号字段才有效。当 ACK = 0 时，确认号无效。**

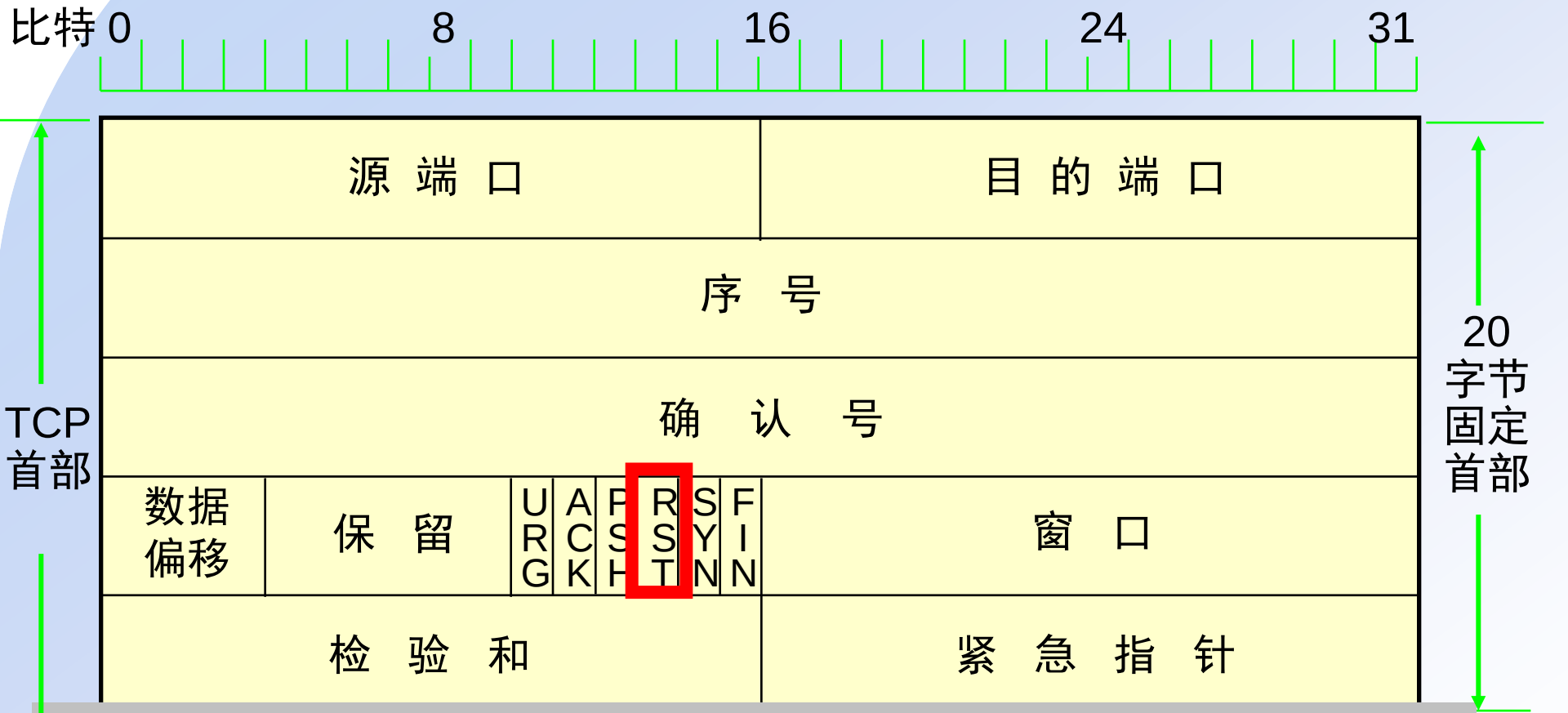
TCP数据报文结构



- **推送比特 PSH (PuSH) —— 接收 TCP 收到推送比特置 1 的报文段，就尽快地交付给接收应用进程，而不再等到整个缓存都填满了后再向上交付。**

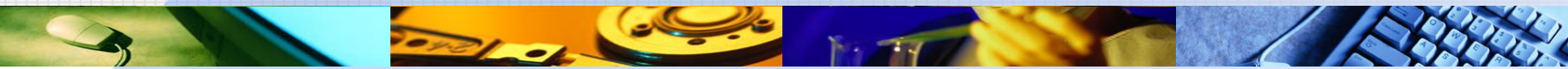
TCP数据报文结构





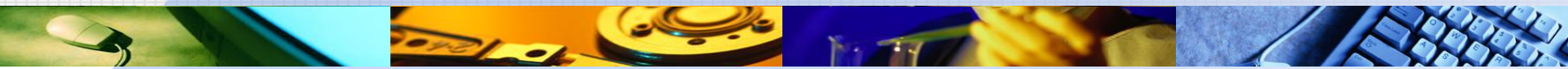
- **复位比特 RST (ReSeT) —— 当 RST = 1 时，表明 TCP 连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。**

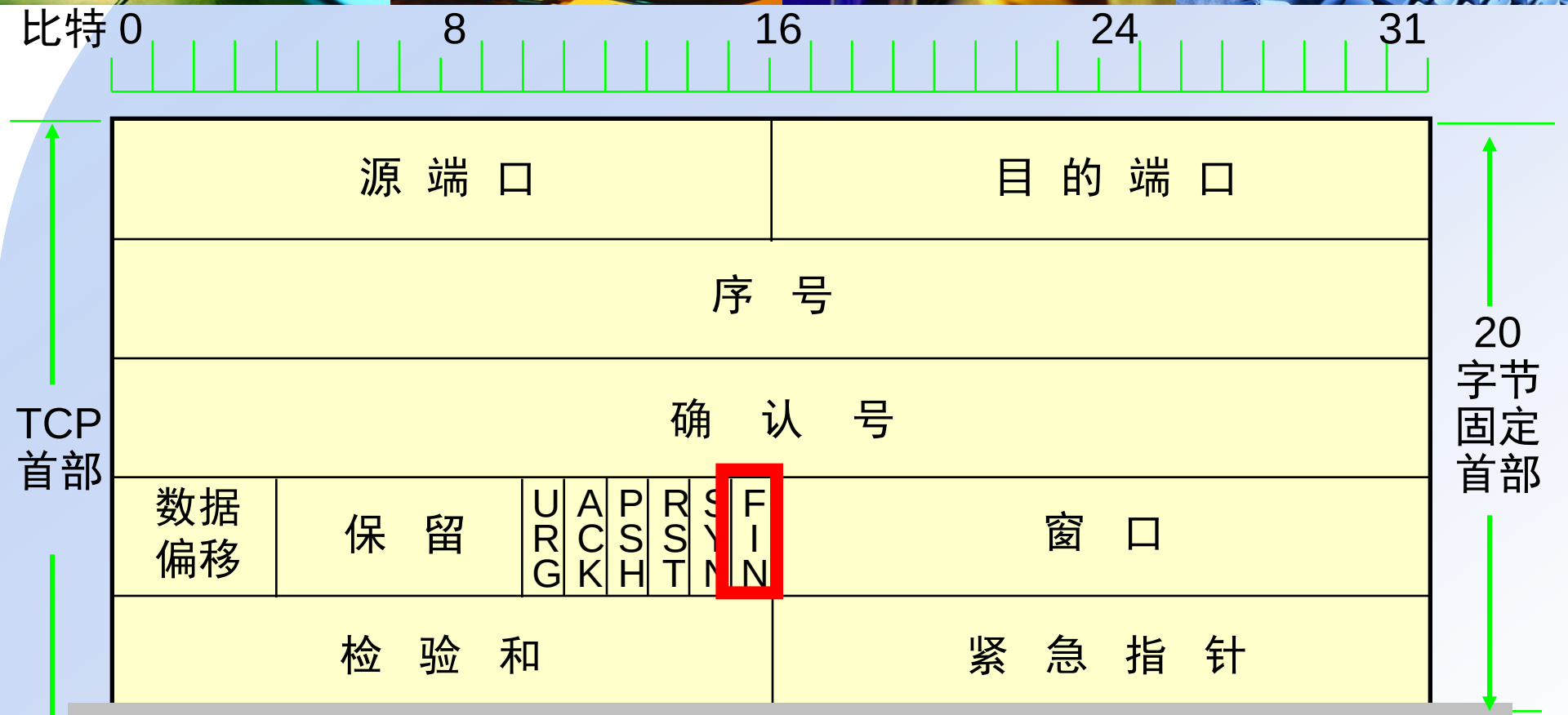
TCP数据报文结构



- **同步比特 SYN —— 同步比特 SYN 置为 1，就表示这是一个连接请求或连接接受报文。**

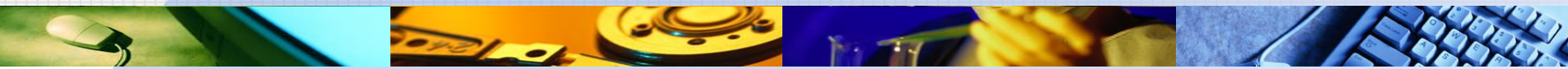
TCP数据报文结构

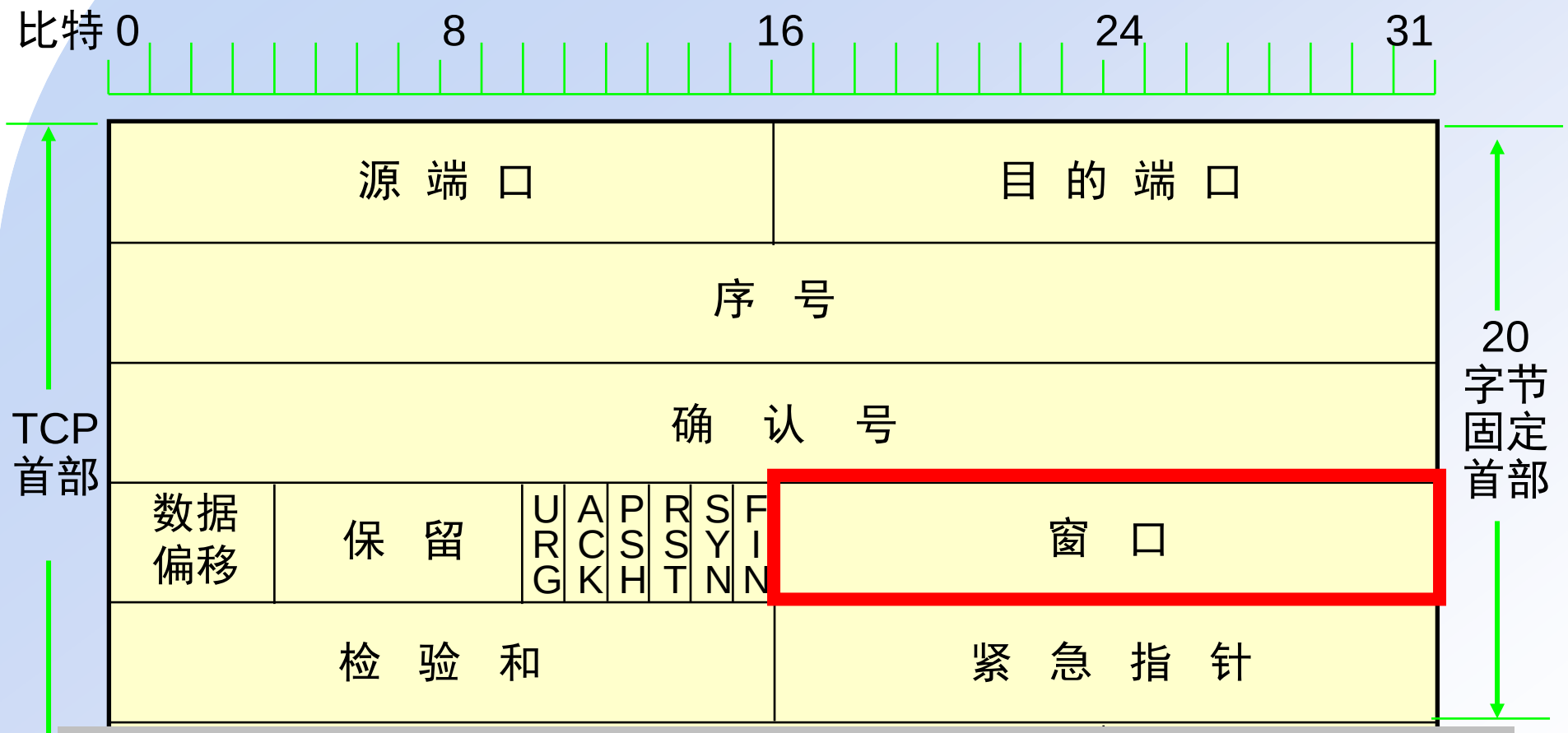




- **终止比特 FIN (FINal) —— 用来释放一个连接。当FIN = 1 时，表明此报文段的发送端的数据已发送完毕，并要求释放运输连接。**

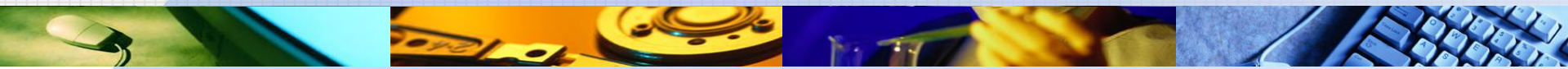
TCP数据报文结构

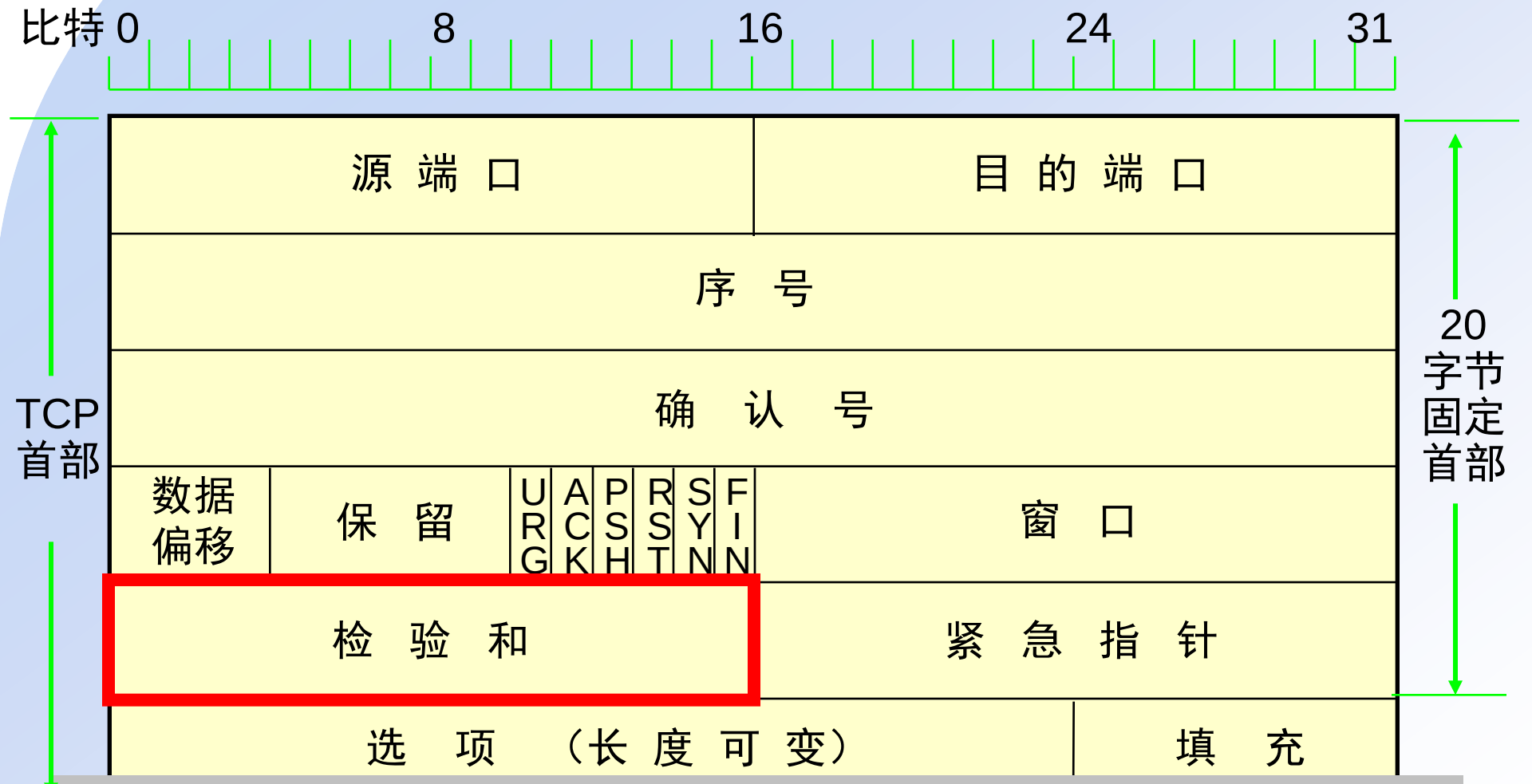




- **窗口字段 —— 占 2 字节。窗口字段用来控制对方发送的数据量，单位为字节。TCP 连接的一端根据设置的缓存空间大小确定自己的接收窗口大小，然后通知对方以确定对方的发送窗口的上限。**

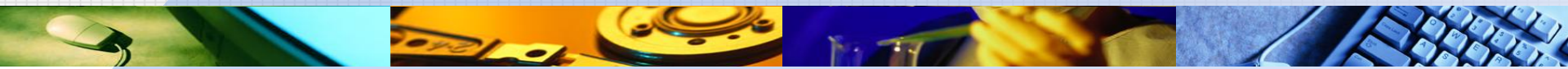
TCP数据报文结构

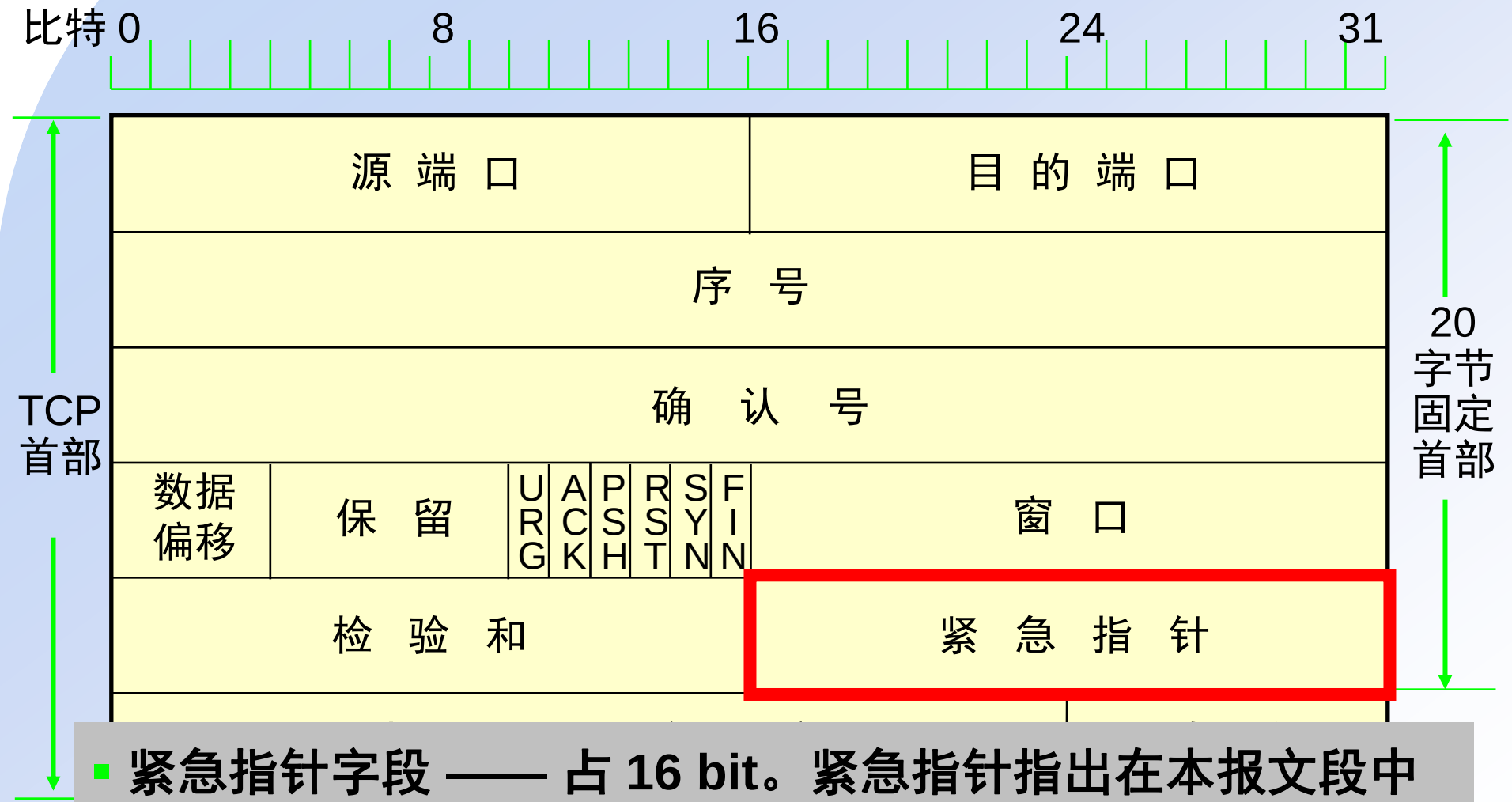




- **检验和 —— 占 2 字节。检验和字段检验的范围包括首部和数据这两部分。在计算检验和时，要在 TCP 报文段的前面加上 12 字节的伪首部。**

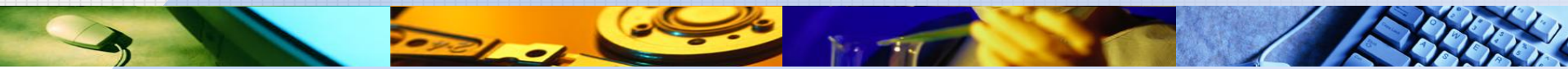
TCP数据报文结构

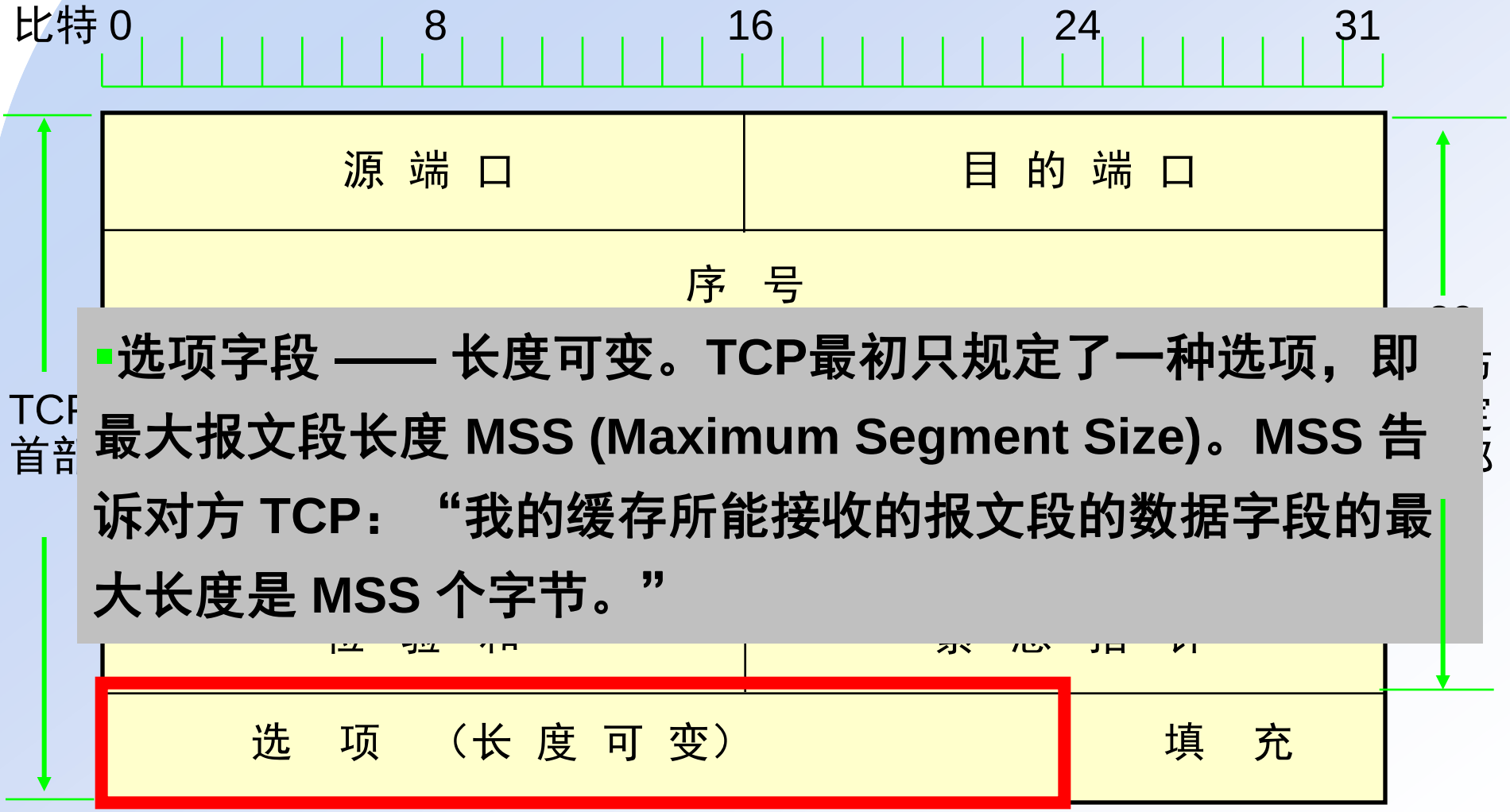




- **紧急指针字段 —— 占 16 bit。紧急指针指出在本报文段中的紧急数据的最后一个字节的序号。**

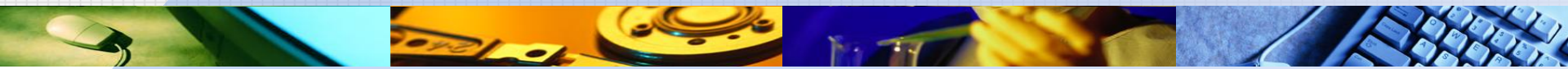
TCP数据报文结构

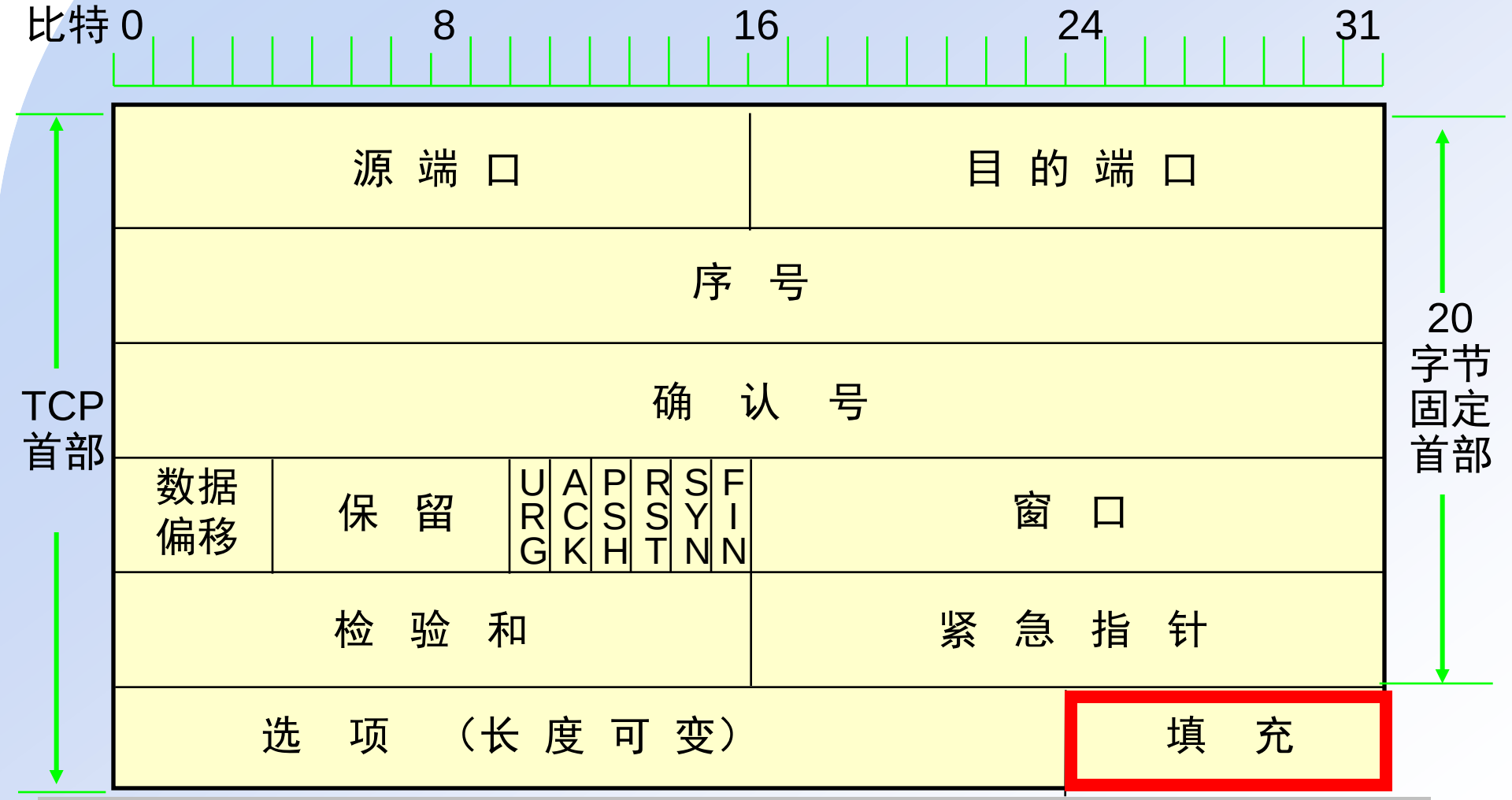




- **选项字段 —— 长度可变。TCP最初只规定了一种选项，即最大报文段长度 MSS (Maximum Segment Size)。MSS 告诉对方 TCP：“我的缓存所能接收的报文段的数据字段的最大长度是 MSS 个字节。”**

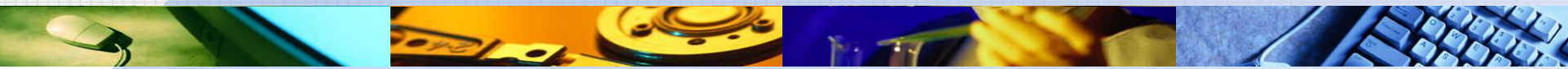
TCP数据报文结构





填充字段 —— 这是为了使整个首部长度是 4 字节的整数倍。

主题 5



1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. 利用滑动窗口实现流量控制

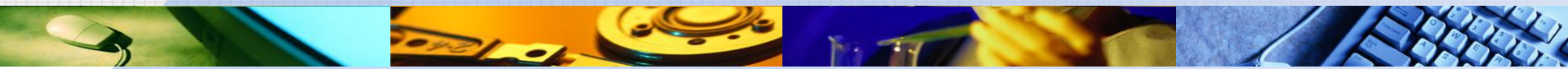
数据编号与确定

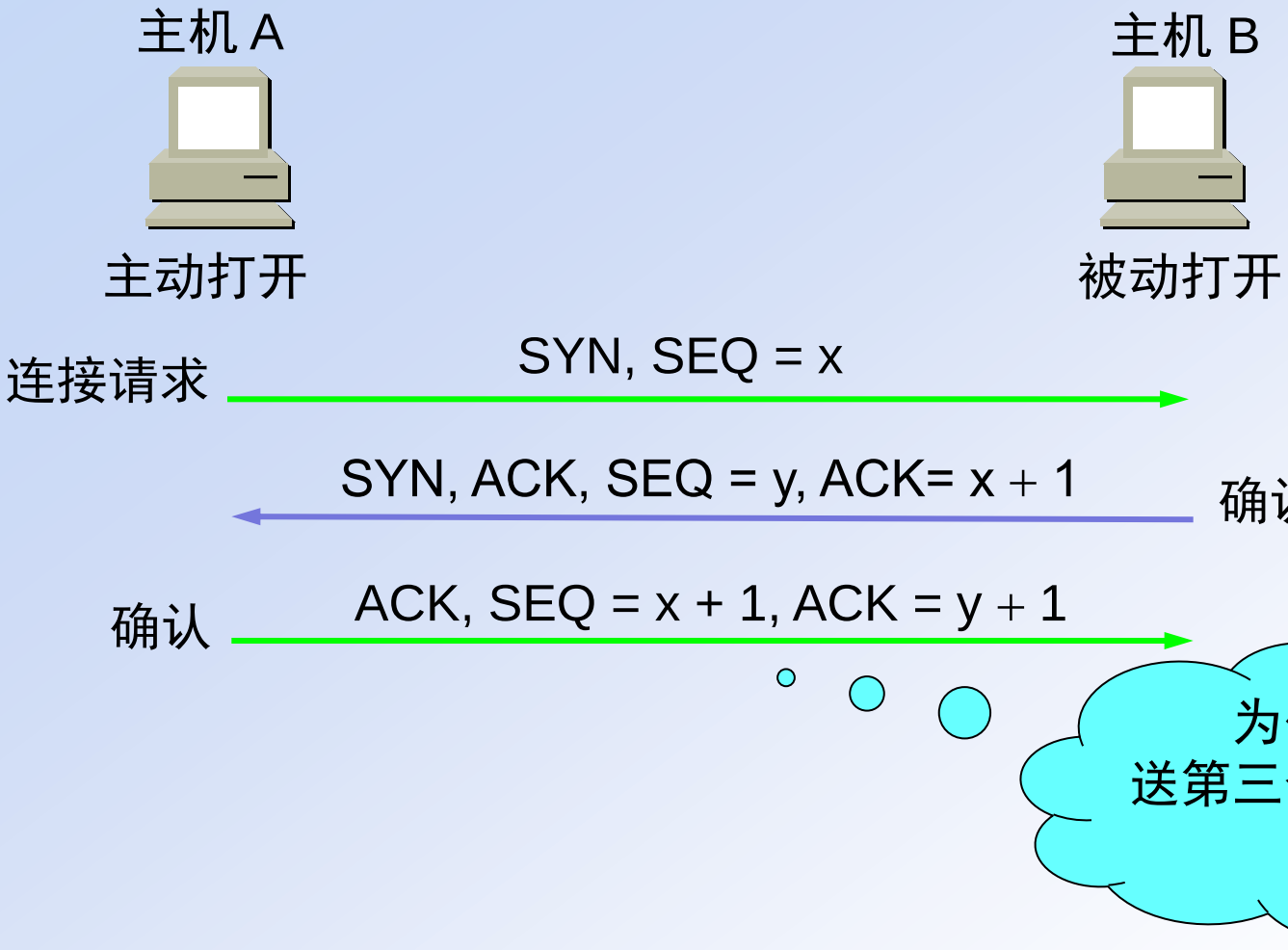
- TCP 协议是面向字节的。TCP 将所要传送的报文看成是字节组成的数据流，并使**每一个字节对应于一个序号**。
- 在**连接建立时，双方要商定初始序号**。
- TCP 每次发送的报文段的首部中的序号字段数值表示该**报文段中的数据部分的第一个字节的序**号。
- TCP 的确认是对接收到的**数据的最高序号**表示确认。
- 接收端返回的确认号是已**收到的数据的最高序号加 1**。因此确认号表示接收端期望下次收到的数据中的第一个数据字节的序号。

TCP 的运输连接管理

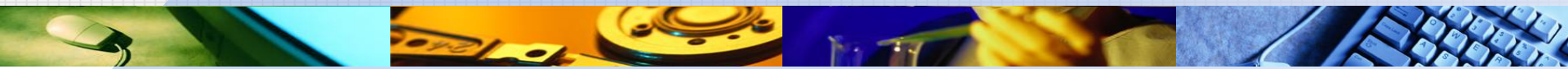
- 运输层连接就有三个阶段：连接建立、数据传送和连接释放。
- 连接建立过程中要解决以下三个问题：
  - 要使每一方能够确知对方的存在。
  - 要允许双方协商一些参数（如最大报文段长度，最大窗口大小，服务质量等）。
  - 能够对运输实体资源（如缓存大小）进行分配。
- TCP连接采用客户-服务器方式：客户主动发出连接请求，服务器被动等待连接的建立。

用三次握手建立 TCP 连接



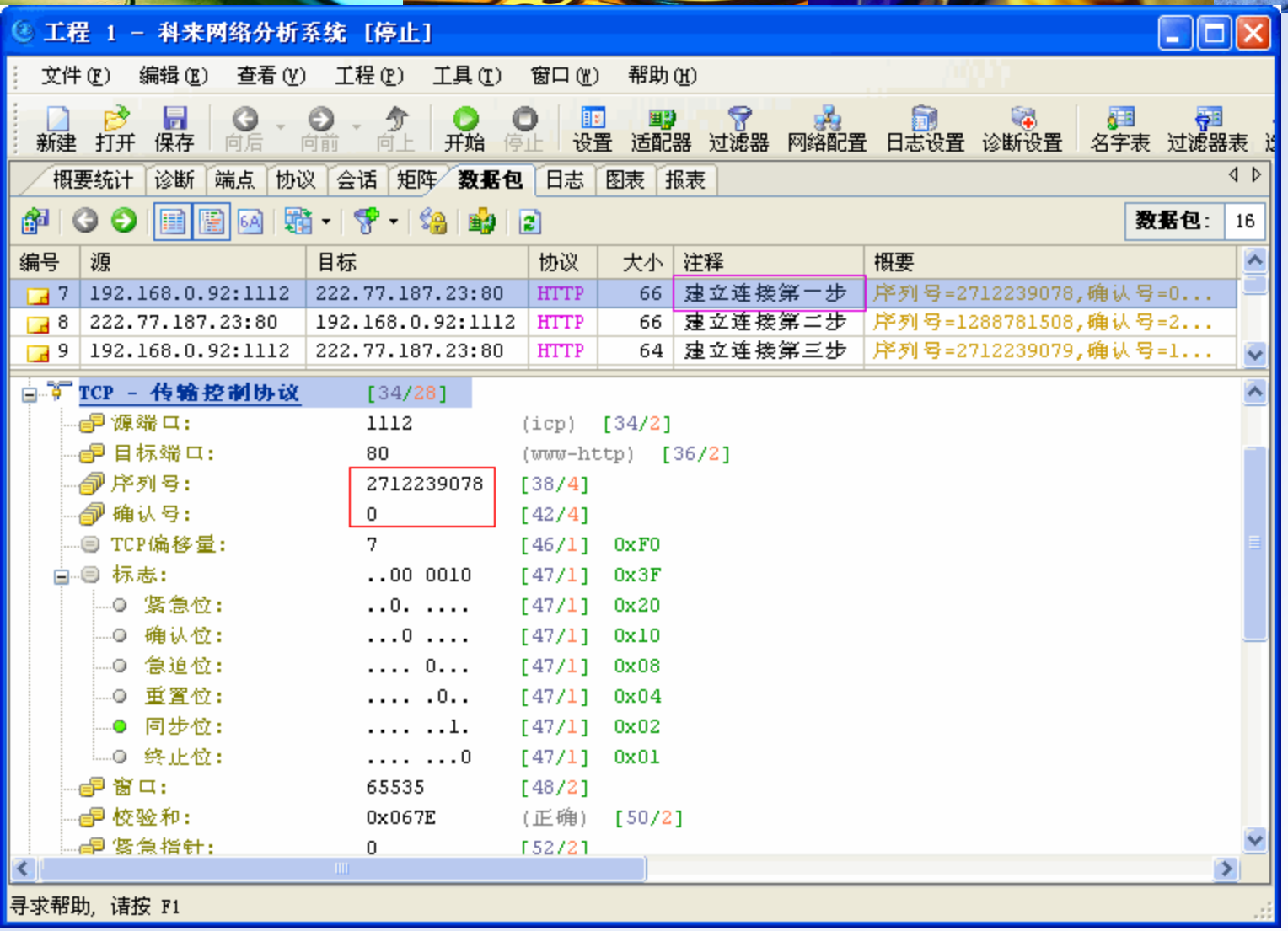


建立 TCP 连接



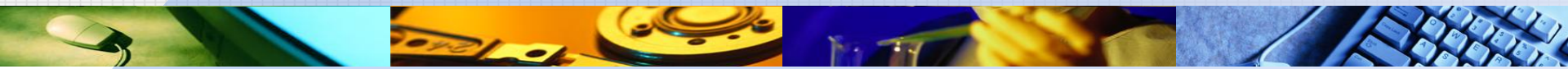
- A 的 TCP 向 B 发出连接请求报文段，其首部中的同步比特 SYN 应置为 1，并选择序号 x，表明传送数据时的第一个数据字节的序号是 x。
- B 的 TCP 收到连接请求报文段后，如同意，则发回确认。
- B 在确认报文段中应将 SYN 置为 1，其确认号应为 x + 1，同时也为自己选择序号 y。
- A 收到此报文段后，向 B 给出确认，其确认号应为 y + 1。
- A 的 TCP 通知上层应用进程，连接已经建立。
- 当运行服务器进程的主机 B 的 TCP 收到主机 A 的确认后，也通知其上层应用进程，连接已经建立。

TCP连接建立示例：捕获TCP协议的数据
客户端为192.168.0.92，服务器为：222.77.187.23



客户端向服务器发起一个同步请求数据包，请求连接服务器的80端口，客户端随机产生一个初始序列号（ISN）为2712239078，确认号为0

TCP连接示例：服务器确认



工程 1 – 科来网络分析系统 [停止]

文件(F) 编辑(E) 查看(V) 工程(P) 工具(T) 窗口(W) 帮助(H)

新建 打开 保存 向后 向前 向上 开始 停止 设置 适配器 过滤器 网络配置 日志设置 诊断设置 名字表 过滤器表

概要统计 诊断 端点 协议 会话 矩阵 数据包 日志 图表 报表

数据包: 16

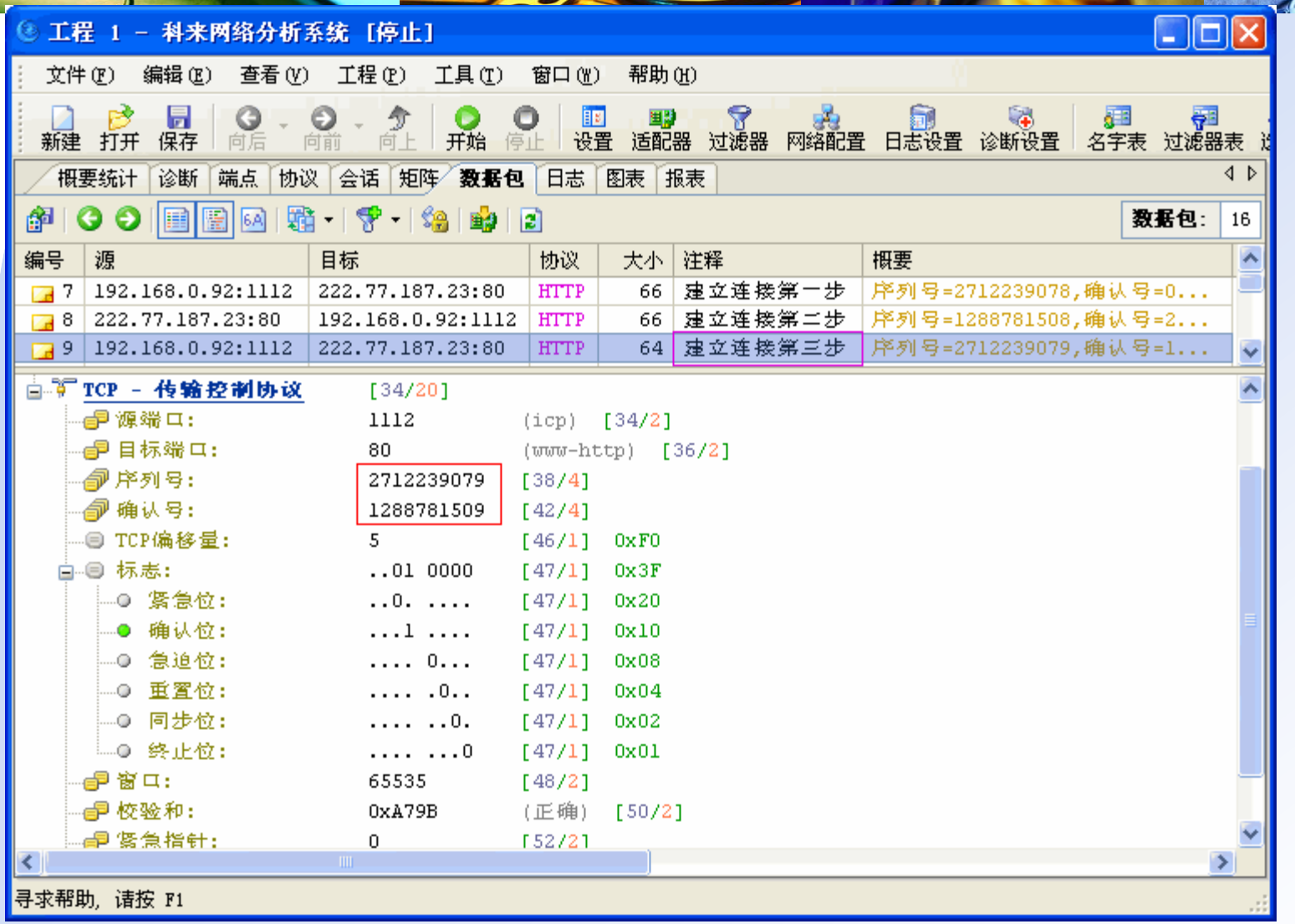
编号	源	目标	协议	大小	注释	概要
7	192.168.0.92:1112	222.77.187.23:80	HTTP	66	建立连接第一步	序列号=2712239078,确认号=0...
8	222.77.187.23:80	192.168.0.92:1112	HTTP	66	建立连接第二步	序列号=1288781508,确认号=2...
9	192.168.0.92:1112	222.77.187.23:80	HTTP	64	建立连接第三步	序列号=2712239079,确认号=1...

TCP – 传输控制协议	[34/28]			
源端口:	80	(www-http)	[34/2]	
目标端口:	1112	(icp)	[36/2]	
序列号:	1288781508	[38/4]		
确认号:	2712239079	[42/4]		
TCP偏移量:	7	[46/1]	0xF0	
标志:	..01 0010	[47/1]	0x3F	
紧急位:	..0.	[47/1]	0x20	
确认位:	...1	[47/1]	0x10	
急迫位:	.... 0...	[47/1]	0x08	
重置位:	.... .0..	[47/1]	0x04	
同步位:	.... ..1.	[47/1]	0x02	
终止位:	0	[47/1]	0x01	
窗口:	5840	[48/2]		
校验和:	0x6423	(正确)	[50/2]	
紧急指针:	0	[52/2]		

寻求帮助，请按 F1

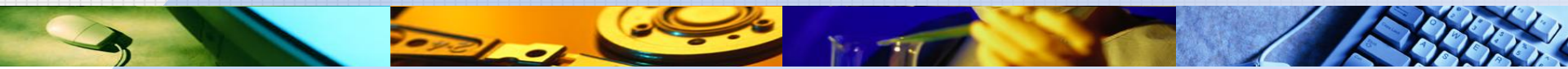
服务器收到客户的同步请求数据包后，并向客户端发送一个同步确认数据。这个数据包中，服务器随机产生一个初始序列号（1288781508），同时，将客户端发送的初始序列号（ISN）加1（2712239078+1=2712239079）以作为确认号发回给客户段进行确认。

TCP连接示例：客户端确认



客户端收到这个同步确认数据包后，再次对服务器进行一次确认。在这个数据包中，序列号为上一个数据包的确认号（2712239079），确认号为服务 器的初始序列号（ISN）加1（1288781508+1=1288781509），以对服务器的同步确认数据包进行确认，这样TCP连接就建立了

序列号接续：初次传输数据



工程 1 – 科来网络分析系统 [停止]

文件(F) 编辑(E) 查看(V) 工程(P) 工具(T) 窗口(W) 帮助(H)

新建 打开 保存 向后 向前 向上 开始 停止 设置 适配器 过滤器 网络配置 日志设置 诊断设置 名字表 过滤器表

概要统计 诊断 端点 协议 会话 矩阵 数据包 日志 图表 报表

数据包: 16

编号	源	目标	协议	大小	注释	概要
9	192.168.0.92:1112	222.77.187.23:80	HTTP	64	建立连接第三步	序列号=2712239079,确认号=1...
10	192.168.0.92:1112	222.77.187.23:80	HTTP	1,018	传输数据	C: GET /stat.htm?id=230826...
11	222.77.187.23:80	192.168.0.92:1112	HTTP	64	确认收到	序列号=1288781509,确认号=2...

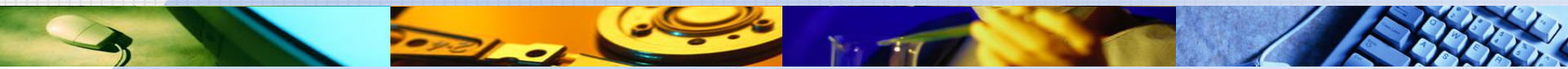
TCP – 传输控制协议 [34/20]

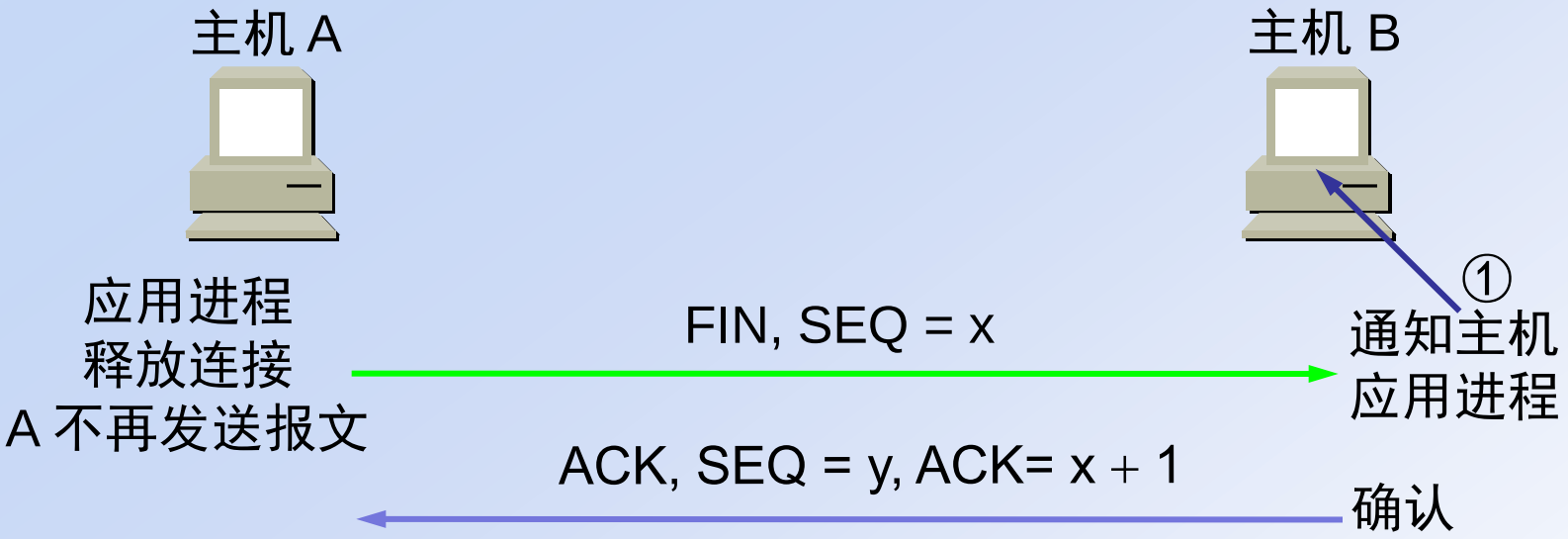
源端口:	80	(www-http)	[34/2]
目标端口:	1112	(icp)	[36/2]
序列号:	1288781509	[38/4]	
确认号:	2712240039	[42/4]	
TCP偏移量:	5	[46/1]	0xF0
标志:	..01 0000	[47/1]	0x3F
紧急位:	..0.	[47/1]	0x20
确认位:	...1	[47/1]	0x10
急迫位:	.... 0...	[47/1]	0x08
重置位:	.... .0..	[47/1]	0x04
同步位:	.... ..0.	[47/1]	0x02
终止位:	0	[47/1]	0x01
窗口:	7680	[48/2]	
校验和:	0x85DB	(正确)	[50/2]

寻求帮助，请按 F1

序列号为，确认号为，这和三次的数据包中认号相同。这个数据包字节，其Ethernet的IP报头，报头和4字18-14-），得到小为960。包中的序列大小（即+960 9），发现列号”的值就是下一个器发送给客中的确认号

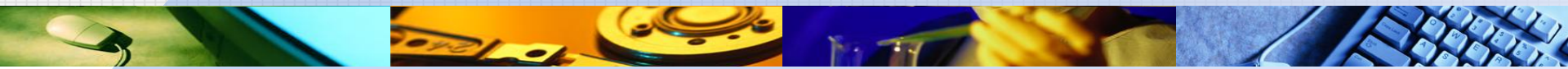
TCP 连接释放的过程





从 A 到 B 的连接就释放了，连接处于半关闭状态。相当于 A 向 B 说：“我已经没有数据要发送了。但你如果还发送数据，我仍接收。”

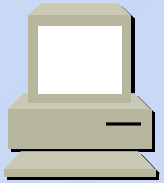
TCP 连接释放的过程

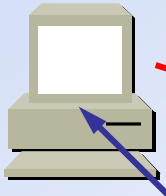


建立一个连接需要3个步骤，但是关闭一个连接需要经过4个步骤。因为TCP连接是全双工的工作模式，所以每个方向上需要单独关闭。在TCP关 闭连接时，首先关闭的一方（即发送第一个终止数据包的）将执行主动关闭，而另一方（收到这个终止数据包的）再执行被动关闭

主机 A

主机 B





应用进程
释放连接
A 不再发送报文

FIN, SEQ = x

①
通知主机
应用进程

②

ACK, SEQ = y, ACK= x + 1

确认

应用进程
释放连接
B 不再发送报文

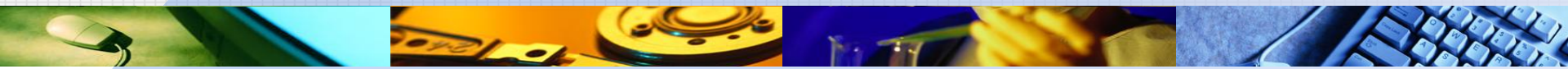
FIN, ACK, SEQ = y, ACK = x + 1

确认

ACK, SEQ = x + 1, ACK = y + 1

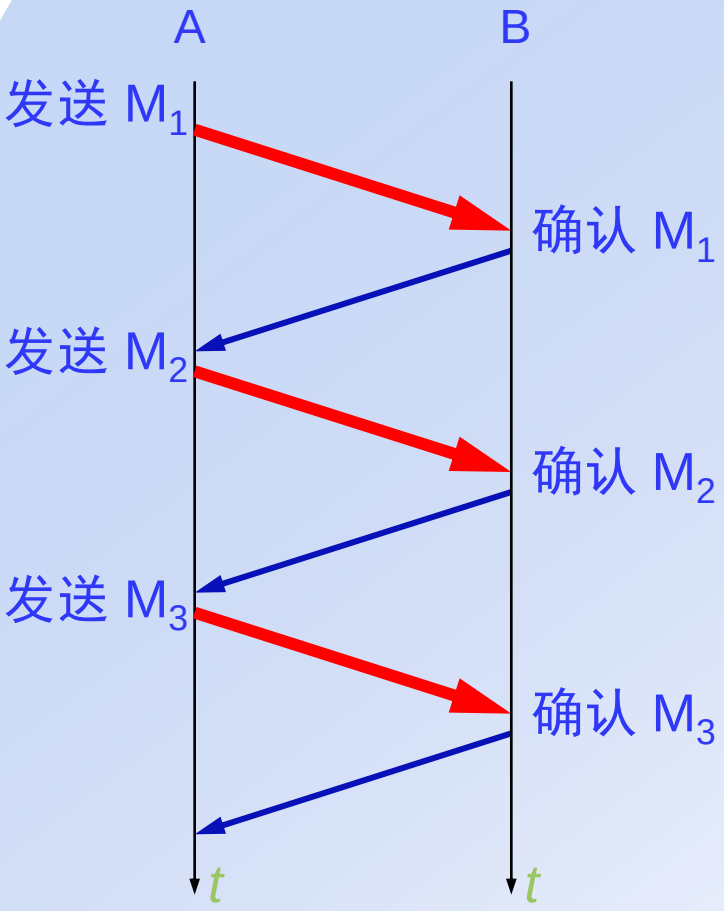
至此，整个连接已经全部释放。

主题 6

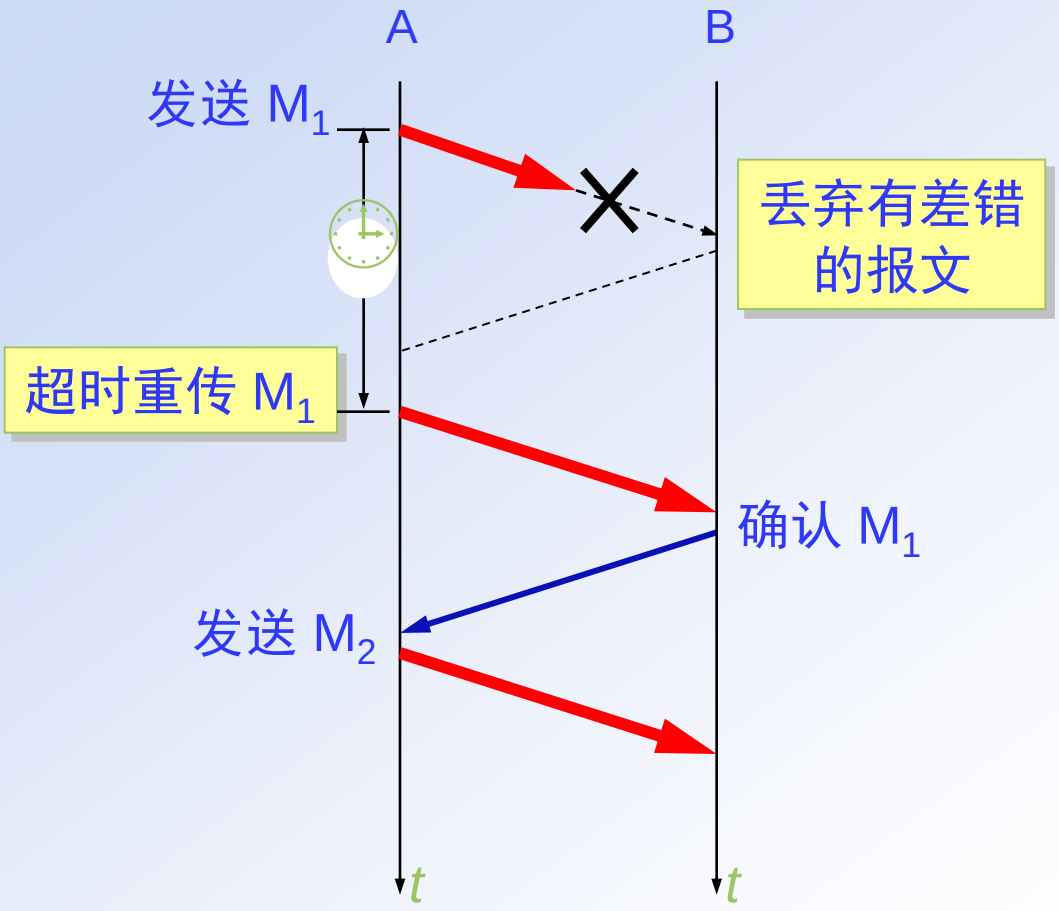


1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. 利用滑动窗口实现流量控制

停止等待协议

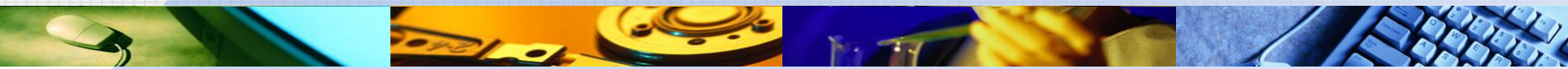


(a) 无差错情况



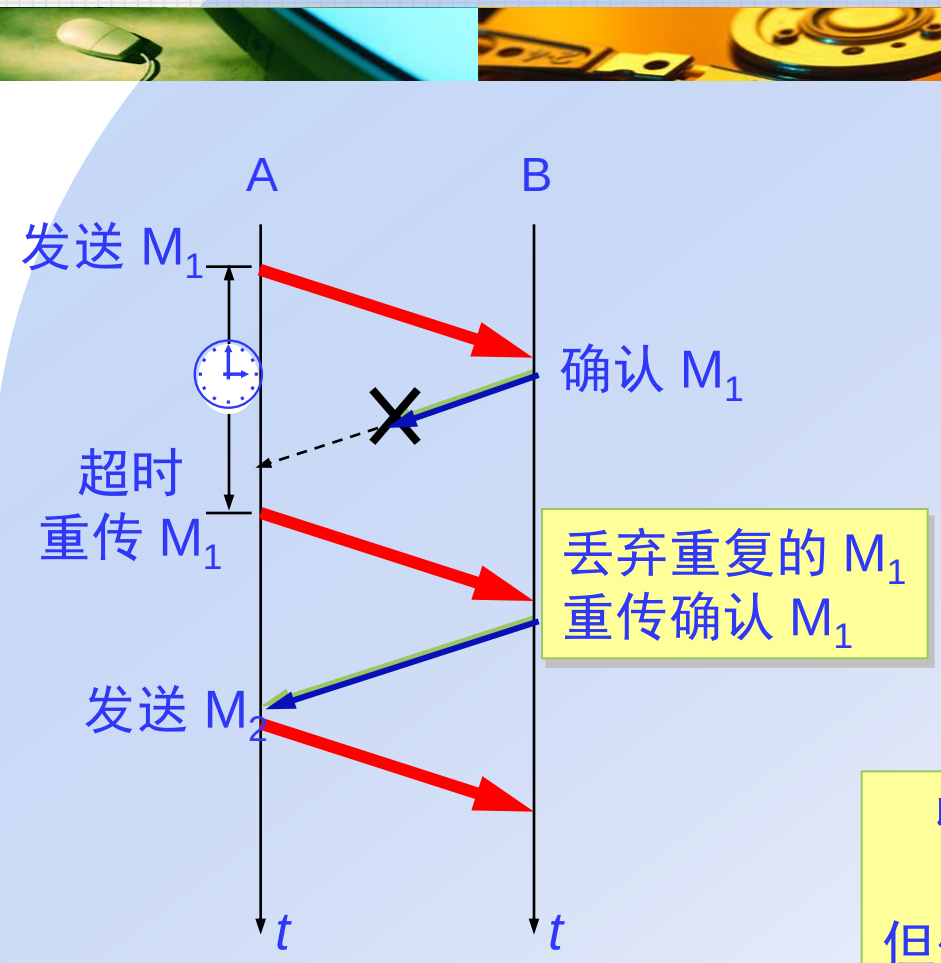
(b) 超时重传

请注意

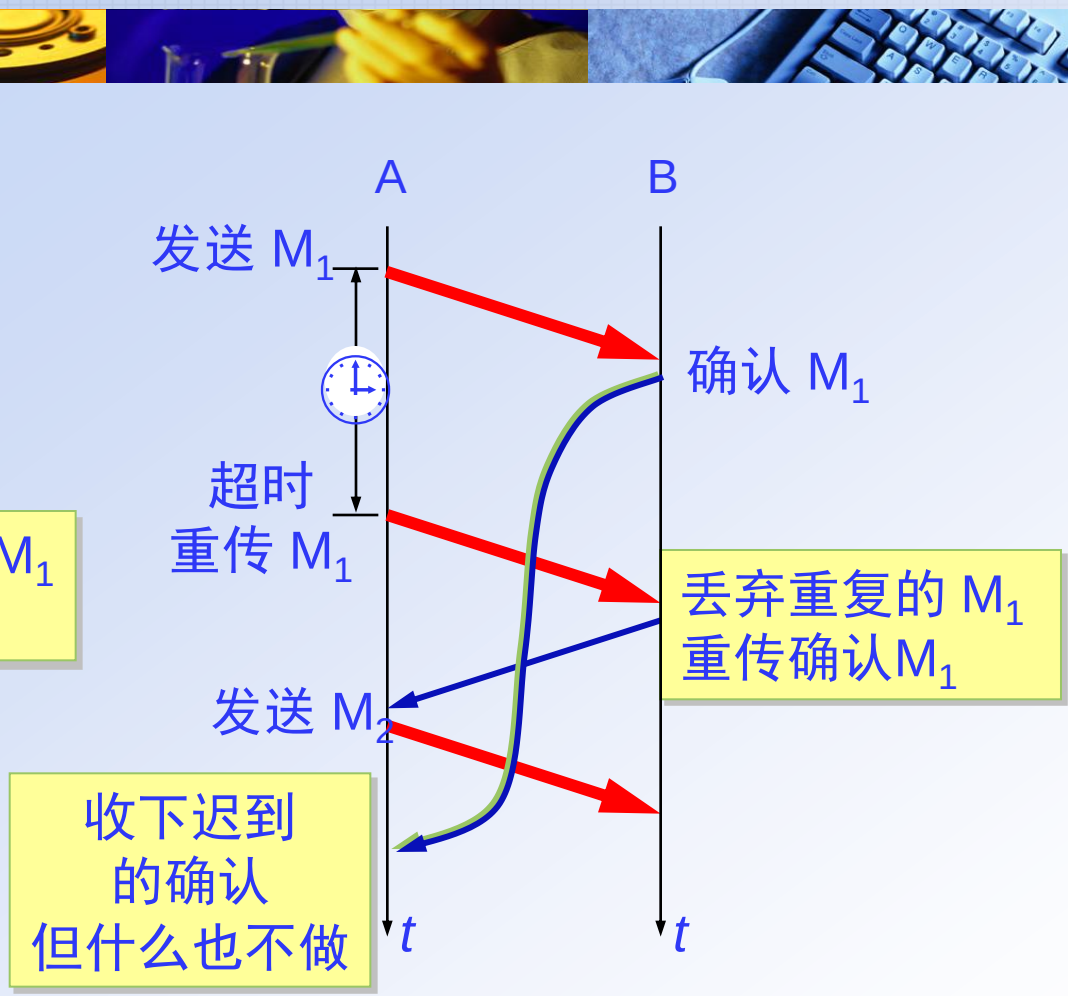


- 在发送完一个分组后，必须暂时保留已发送的分组的副本。
- 分组和确认分组都必须进行编号。
- 超时计时器的重传时间应当比数据在分组传输的平均往返时间更长一些。

确认丢失和确认迟到

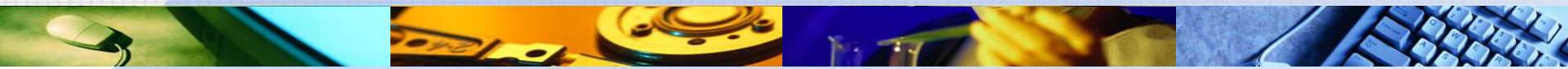


(a) 确认丢失



(b) 确认迟到

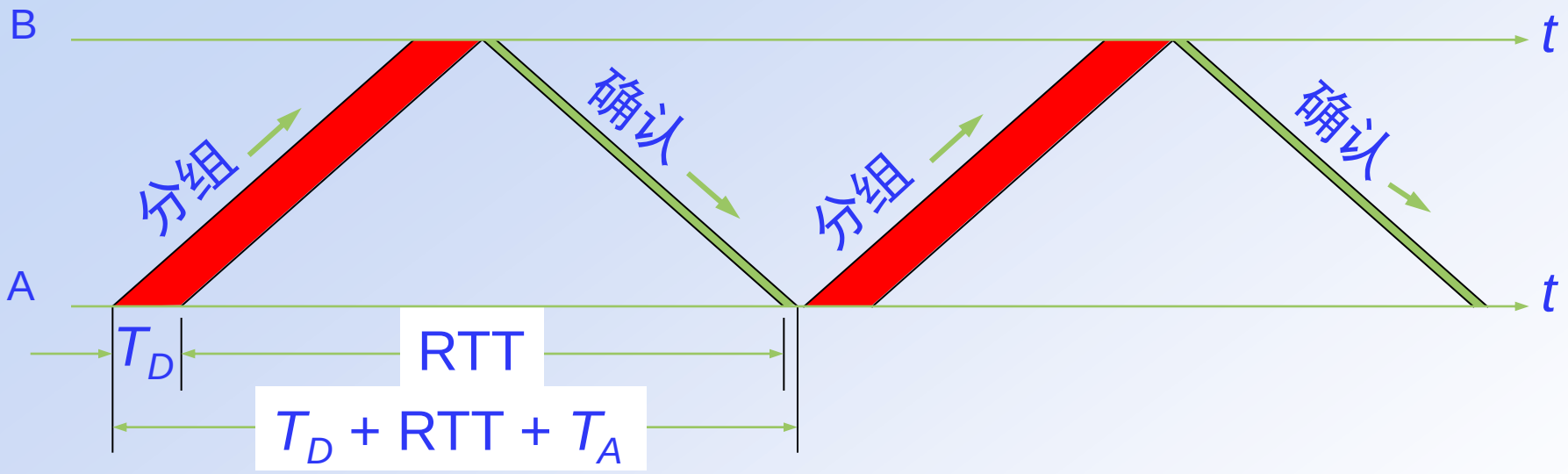
可靠通信的实现



- 使用上述的确认和重传机制，我们就可以在不可靠的传输网络上实现可靠的通信。
- 这种可靠传输协议常称为自动重传请求ARQ (Automatic Repeat reQuest)。
- 接收方只对正确接收的报文段进行确认。
- ARQ 表明重传的请求是自动进行的。接收方不需要请求发送方重传某个出错的分组 。

信道利用率

- 停止等待协议的优点是简单，但缺点是信道利用率太低。

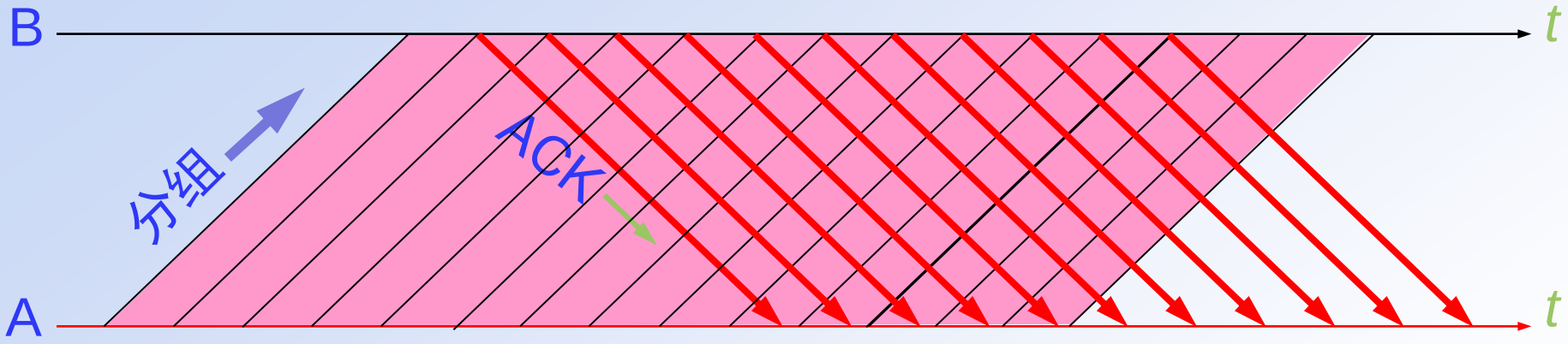


信道的利用率 U

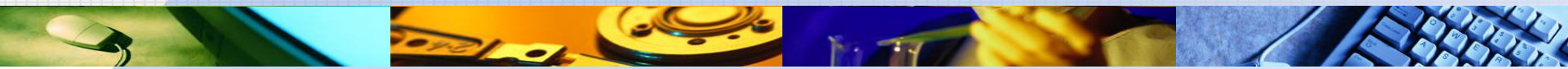
$$U = \frac{T_D}{T_D + \mathrm{RTT} + T_A}$$

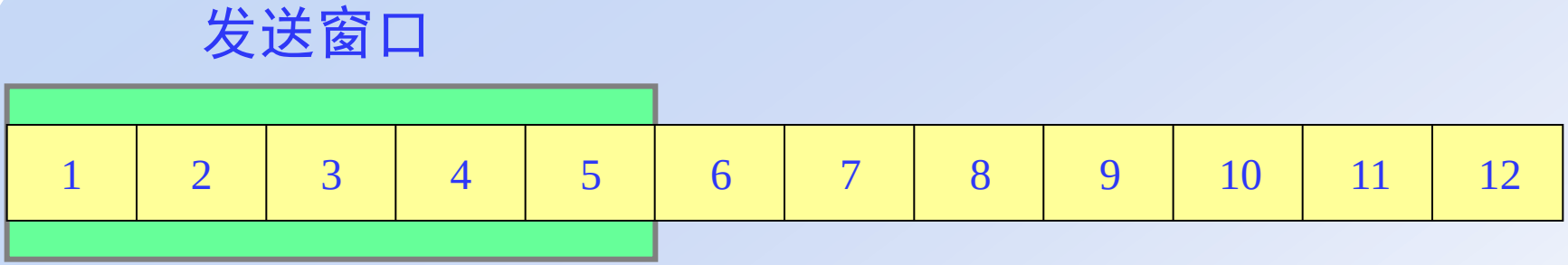
流水线传输

- 发送方可连续发送多个分组，不必每发完一个分组就停顿下来等待对方的确认。
- 由于信道上一直有数据不间断地传送，这种传输方式可获得很高的信道利用率。

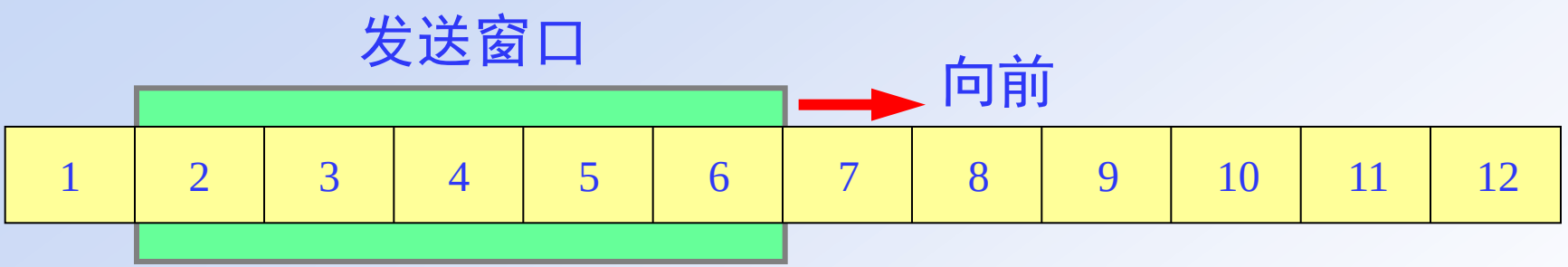


连续 ARQ 协议



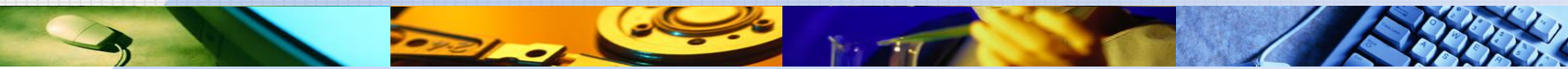


(a) 发送方维持发送窗口（发送窗口是 5）



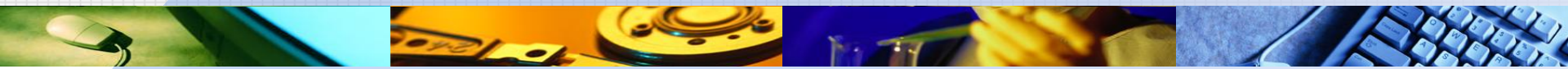
(b) 收到一个确认后发送窗口向前滑动

累积确认



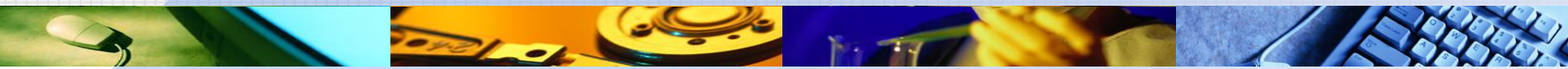
- 接收方一般采用累积确认的方式。即不必对收到的分组逐个发送确认，而是对按序到达的最后一个分组发送确认，这样就表示：到这个分组为止的所有分组都已正确收到了。
- 累积确认有的优点是：容易实现，即使确认丢失也不必重传。缺点是：不能向发送方反映接收方已经正确收到的所有分组的信息。

Go-back-N（回退 N）



- 如果发送方发送了前 5 个分组，而中间的第 3 个分组丢失了。这时接收方只能对前两个分组发出确认。发送方无法知道后面三个分组的下落，而只好把后面的三个分组都再重传一次。
- 这就叫做 Go-back-N（回退 N），表示需要再退回来重传已发送过的 N 个分组。
- 可见当通信线路质量不好时，连续 ARQ 协议会带来负面的影响。

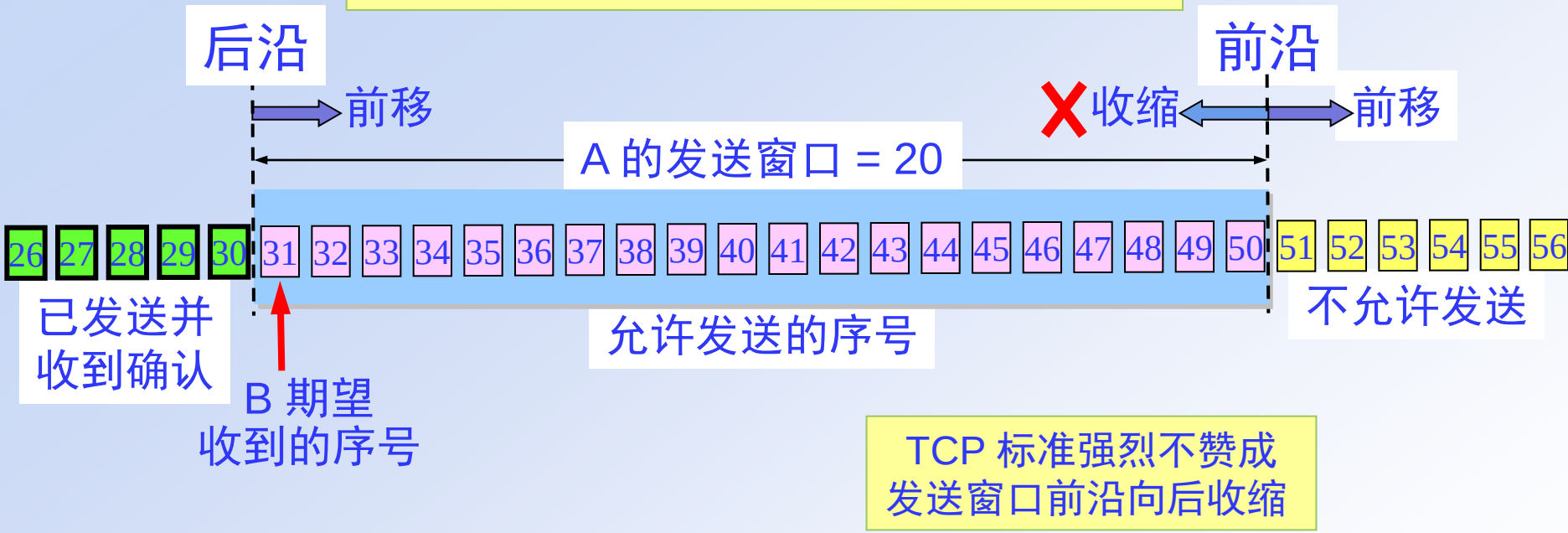
TCP 可靠通信的具体实现



- TCP 连接的每一端都必须设有两个窗口——一个发送窗口和一个接收窗口。
- TCP 的可靠传输机制用字节的序号进行控制。TCP 所有的确认都是基于序号而不是基于报文段。
- TCP 两端的四个窗口经常处于动态变化之中。
- TCP连接的往返时间 RTT 也不是固定不变的。需要使用特定的算法估算较为合理的重传时间。

以字节为单位的滑动窗口

根据 B 给出的窗口值
A 构造出自己的发送窗口



TCP 标准强烈不赞成
发送窗口前沿向后收缩

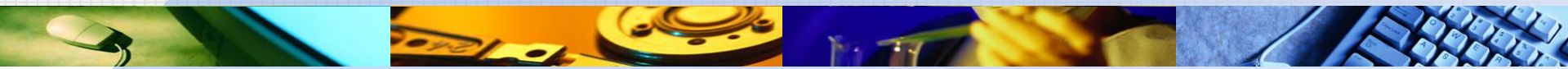
A 发送了 11 个字节的数据

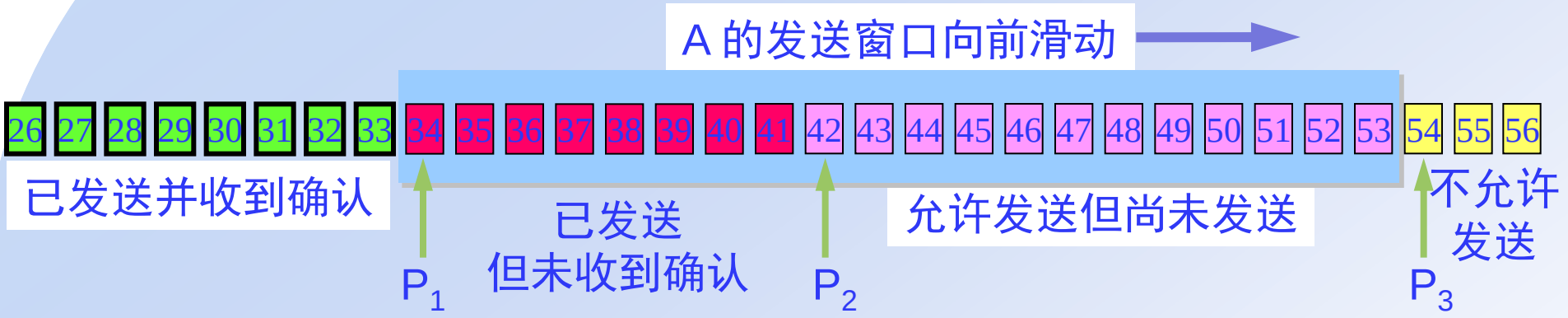
$P_3 - P_1$ = A 的发送窗口（又称为通知窗口）

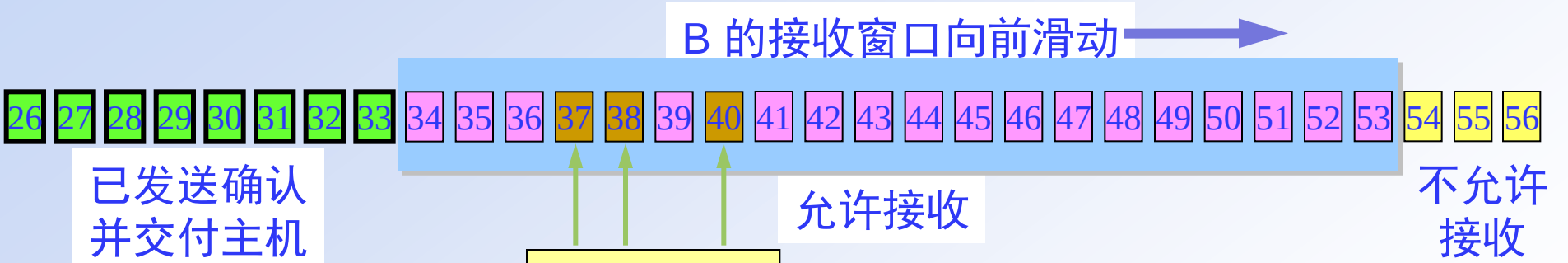
$P_2 - P_1$ = 已发送但尚未收到确认的字节数

$P_3 - P_2$ = 允许发送但尚未发送的字节数（又称为可用窗口）

A 收到新的确认号，发送窗口向前滑动

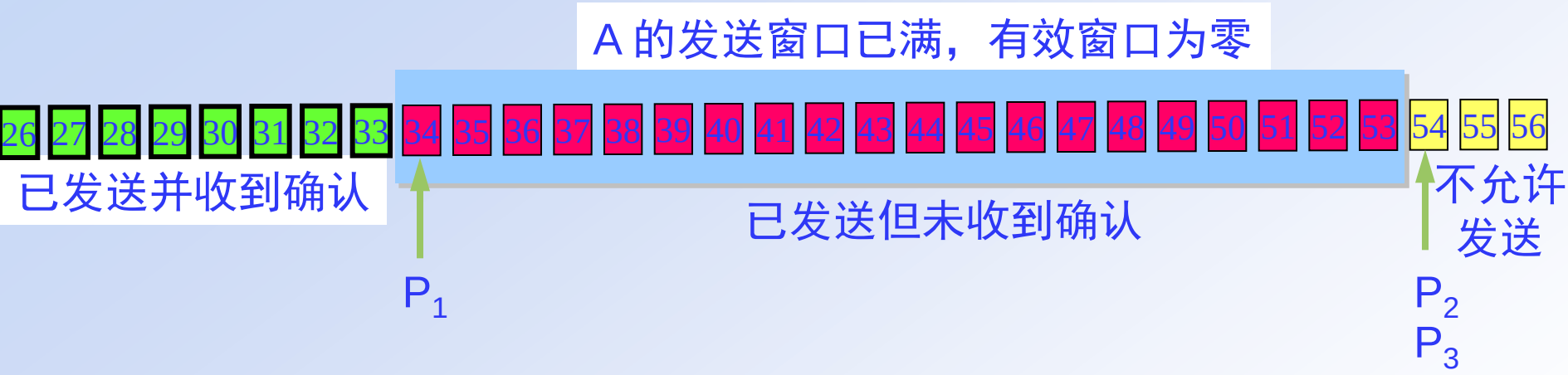




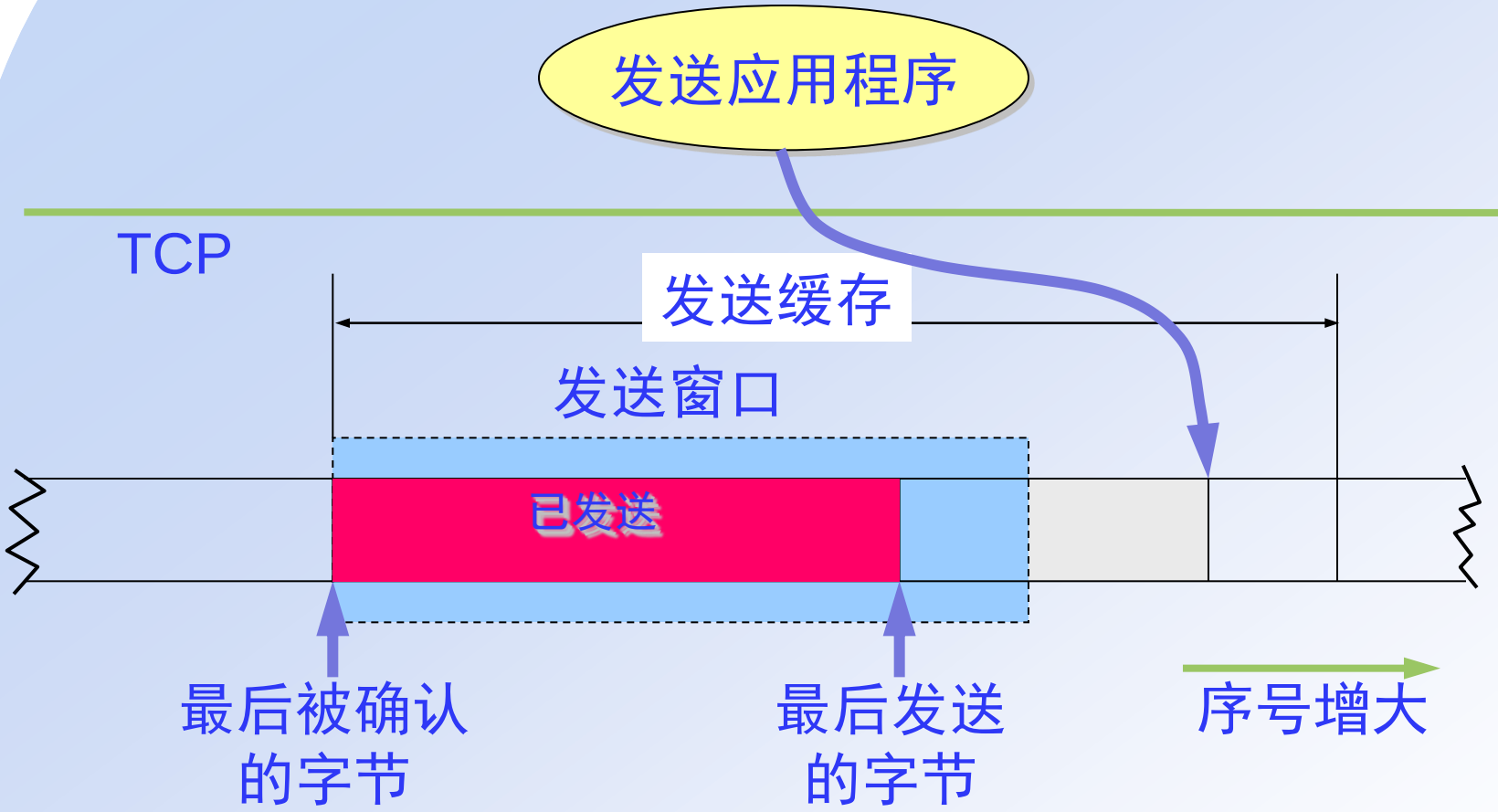


先存下，等待缺少的
数据的到达

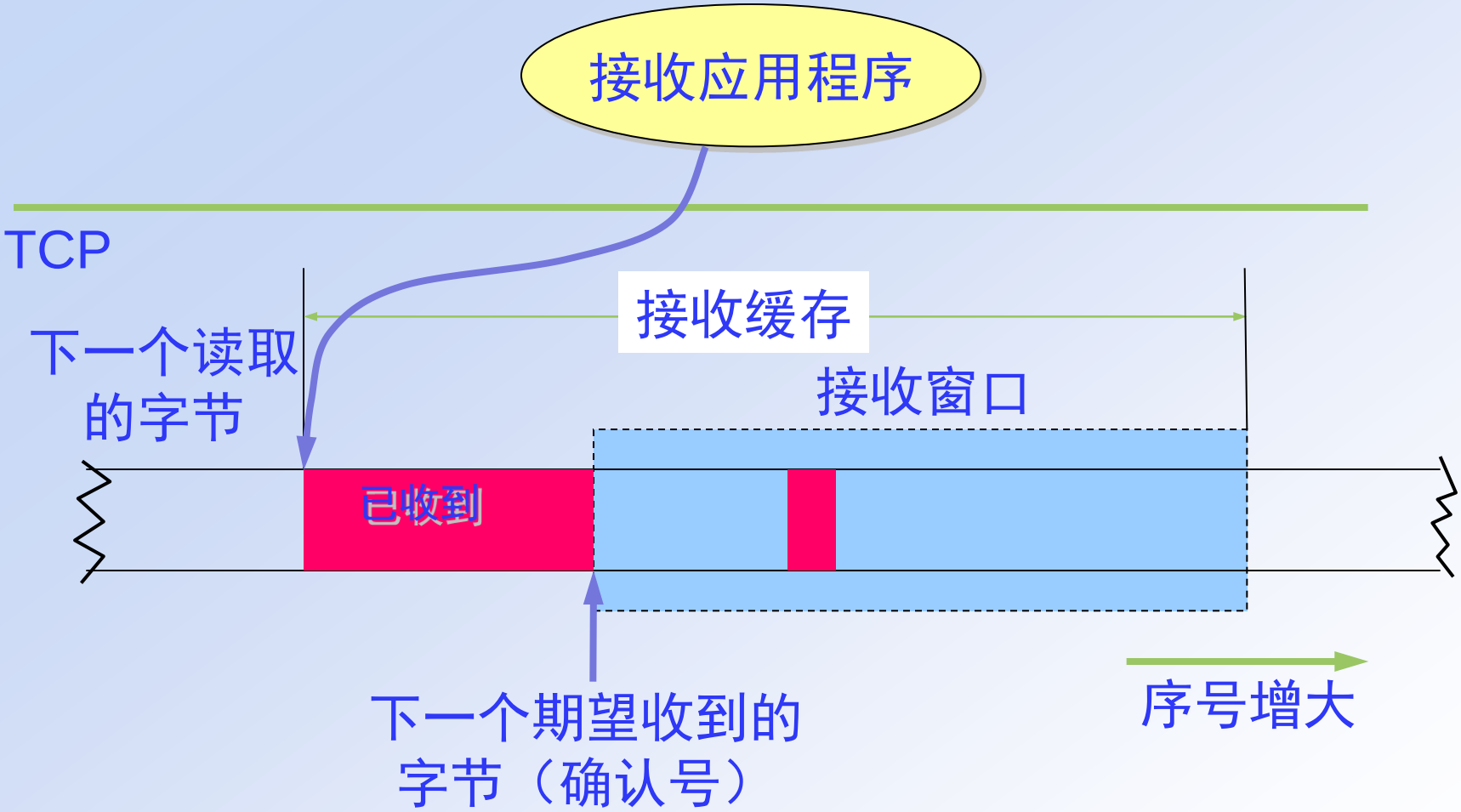
A 的发送窗口内的序号都已用完，
但还没有再收到确认，必须停止发送。



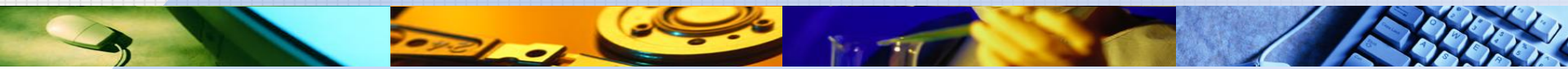
发送缓存



接收缓存

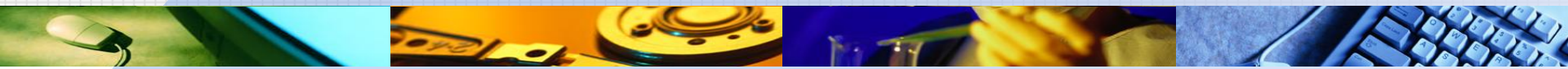


发送缓存与接收缓存的作用



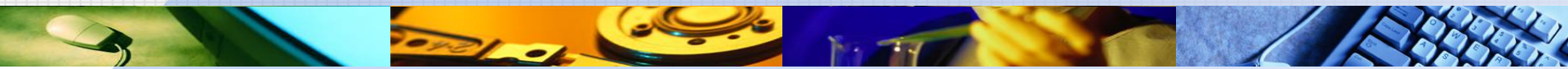
- 发送缓存用来暂时存放：
  - 发送应用程序传送给发送方 TCP 准备发送的数据；
  - TCP 已发送出但尚未收到确认的数据。
- 接收缓存用来暂时存放：
  - 按序到达的、但尚未被接收应用程序读取的数据；
  - 不按序到达的数据。

需要强调三点



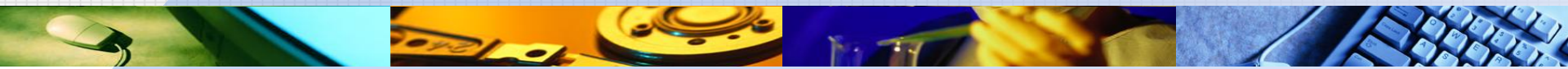
- A 的发送窗口并不总是和 B 的接收窗口一样大
  - **因为有一定的时间滞后**
  - **接收方可以调整窗口大小**
- TCP 标准没有规定对不按序到达的数据应如何处理。通常是先临时存放在接收窗口中，等到字节流中所缺少的字节收到后，再按序交付上层的应用进程。
- TCP 要求接收方必须有累积确认的功能，这样可以减小传输开销。

主题 7



1. 运输层协议概述
2. 用户数据报协议UDP
3. 传输控制协议TCP概述
4. TCP报文格式
5. TCP的运输连接管理
6. TCP可靠传输工作原理
7. TCP的流量控制与拥塞控制

利用滑动窗口实现流量控制

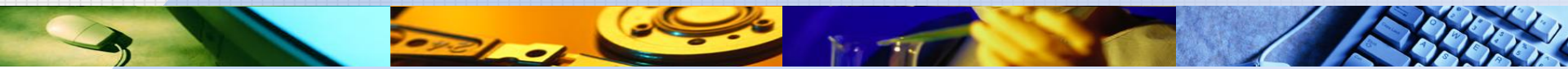


- 一般说来，我们总是希望数据传输得更快一些。但如果发送方把数据发送得过快，接收方就可能来不及接收，这就会造成数据的丢失。
- 流量控制(flow control)就是让发送方的发送速率不要太快，既要让接收方来得及接收，也不要使网络发生拥塞。
- 利用滑动窗口机制可以很方便地在 TCP 连接上实现流量控制。

流量控制举例

例：利用可变窗口大小进行流量控制，双方确定的窗口值是 400。假定报文段数据大小为100字节

网络拥塞



- 在某段时间，若对网络中某资源的需求超过了该资源所能提供的可用部分，网络的性能就要变坏——产生拥塞(congestion)。
- 出现资源拥塞的条件：

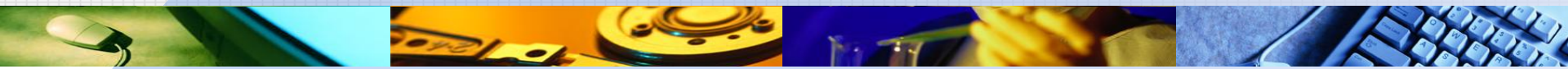
  对资源需求的总和 > 可用资源

- 若网络中有许多资源同时产生拥塞，网络的性能就要明显变坏，整个网络的吞吐量将随输入负荷的增大而下降。

拥塞控制与流量控制的关系

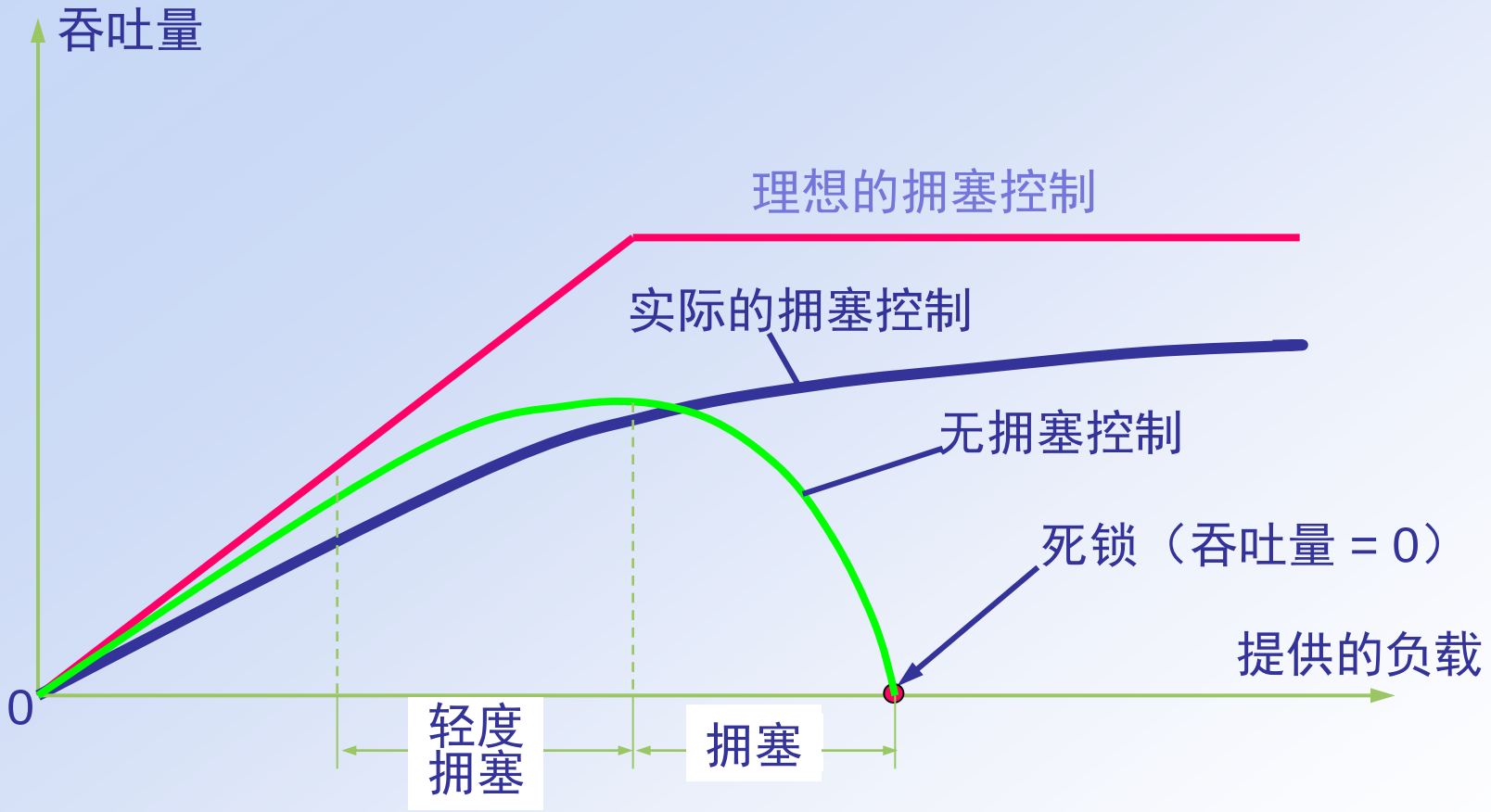
- 拥塞控制是一个全局性的过程，涉及到所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素。
- 流量控制往往指在给定的发送端和接收端之间的点对点通信量的控制。
- 流量控制所要做的就是抑制发送端发送数据的速率，以便使接收端来得及接收。

TCP 的拥塞控制作用



- 产生拥塞的原因
  - 接收方的处理能力不足（接收容器小）
  - 网络不够通畅（传输管道细）
- TCP解决拥塞办法
  - 通过端到端的点对点流量控制，降低发送方的数据传输速率，从而解决网络拥塞。

拥塞控制所起的作用



TCP 的拥塞控制—在TCP的发送端引入拥塞窗口

- 发送端的主机在确定发送报文段的速率时，既要根据接收端的接收能力，又要从全局考虑不要使网络发生拥塞。
- 因此，每一个 TCP 连接需要有以下两个状态变量：
  - 接收端窗口 rwnd (receiver window) 又称为通知窗口(advertised window)。
  - 拥塞窗口 cwnd (congestion window)。

(1) 接收端窗口 rwnd：这是接收端根据其目前的接收缓存大小所许诺的最新的窗口值，是来自接收端的流量控制。接收端将此窗口值放在 TCP 报文的首部中的窗口字段，传送给发送端。

(2) 拥塞窗口 cwnd：是发送端根据自己估计的网络拥塞程度而设置的窗口值，是来自发送端的流量控制。

对网络拥塞能力的预期

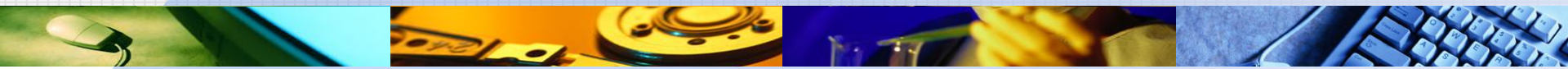
TCP 的拥塞控制—动态调整拥塞窗口

- 只要网络没有出现拥塞，发送端就使得拥塞窗口再增大一点。以便将更多的分组发送出去。但只要网络出现拥塞，发送端就使拥塞窗口减少一些，以减少注入到网络中的分组数。

发送端如何判断网络出现拥塞呢？

是否及时收到ACK报文

TCP 的拥塞控制—发送端的发送窗口上限值

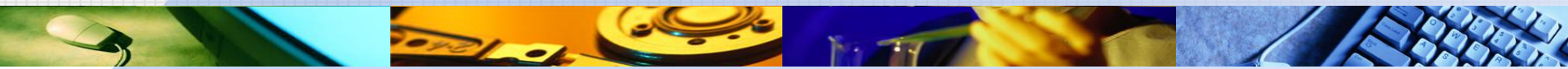


- 发送端的发送窗口的上限值应当取为接收端窗口 rwnd 和拥塞窗口 cwnd 这两个变量中较小的一个，即应按以下公式确定：

  发送窗口的上限值 = Min [rwnd, cwnd]

- 当 rwnd < cwnd 时，是接收端的接收能力限制发送窗口的最大值。
- 当 cwnd < rwnd 时，则是网络的拥塞限制发送窗口的最大值。

几种拥塞控制方法



- RFC2581定义了拥塞控制的4种方法
  - 慢开始 (slow - start)
  - 拥塞避免 (congestion avoidance)
  - 快重传 (fast retransmit)
  - 快恢复 (fast recovery)
- RFC2582又对这些算法进行了一些修改