数据链路层的简单模型

1、使用点对点信道的数据链路层：数据链路和帧；链路层的服务（封装成帧）；透明传输（3.1.22）；差错检测（3.1.23）

2、点对点协议PPP

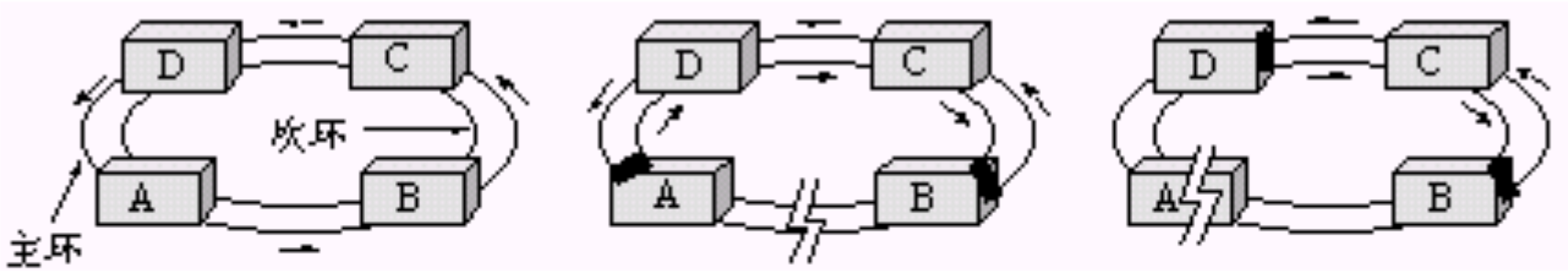
3、使用广播信道的数据链路层（3.3）

4、使用广播信道的以太网（3.4）：以太网的信道利用率；以太网的MAC层

5、扩展的以太网：如何用集线器组网？如何用交换机组网？交换机的工作原理，交换机的例子，交换机的优点一隔离网络流量，交换机的优点二实现独享式访问，交换机组网例子，交换机与路由器的区别，环形网-FDDi

分布式光纤数据接口 FDDI *

- FDDI (Fiber Distributed Data Interface)
- MAC
  - Token Ring (IEEE 802.5)
- 物理层
  - 多模光纤
  - 双环, 更加可靠
  - 100Mbps
- 最大节点数量
  - 1000
- 距离
  - Whole ring: 200km
  - Inter nodes: 2Km
- 帧长度
  - Maximum: 4500 Byte



(a)正常情况　(b)链路出故障　(c)站点出故障

具有双环的 FDDI

数据链路层的简单模型

1、使用点对点信道的数据链路层：数据链路和帧；链路层的服务（封装成帧）；透明传输（3.1.22）；差错检测（3.1.23）

2、点对点协议PPP

3、使用广播信道的数据链路层（3.3）

4、使用广播信道的以太网（3.4）：以太网的信道利用率；以太网的MAC层

5、扩展的以太网：如何用集线器组网？如何用交换机组网？如何用网桥组网：网桥的内部结构，两种类型的网桥，多接口网桥(3.5.2 4)

6、高速以太网(3.6)

读书

知识点
P221 可靠传输的工作原理

第五章 问题

1、为什么要有传输层，它都能干什么?

核心：建立进程间的逻辑通信

2、如何是建立进程间的逻辑通信?

- 通过复用和分用：利用套接字(socket)
- 以UDP协议为例说明如何利用套接字实现复用和分用?
- 以TCP协议为例说明如何利用套接字实现复用和分用？使用四元组将报文段引导到正确地套接字

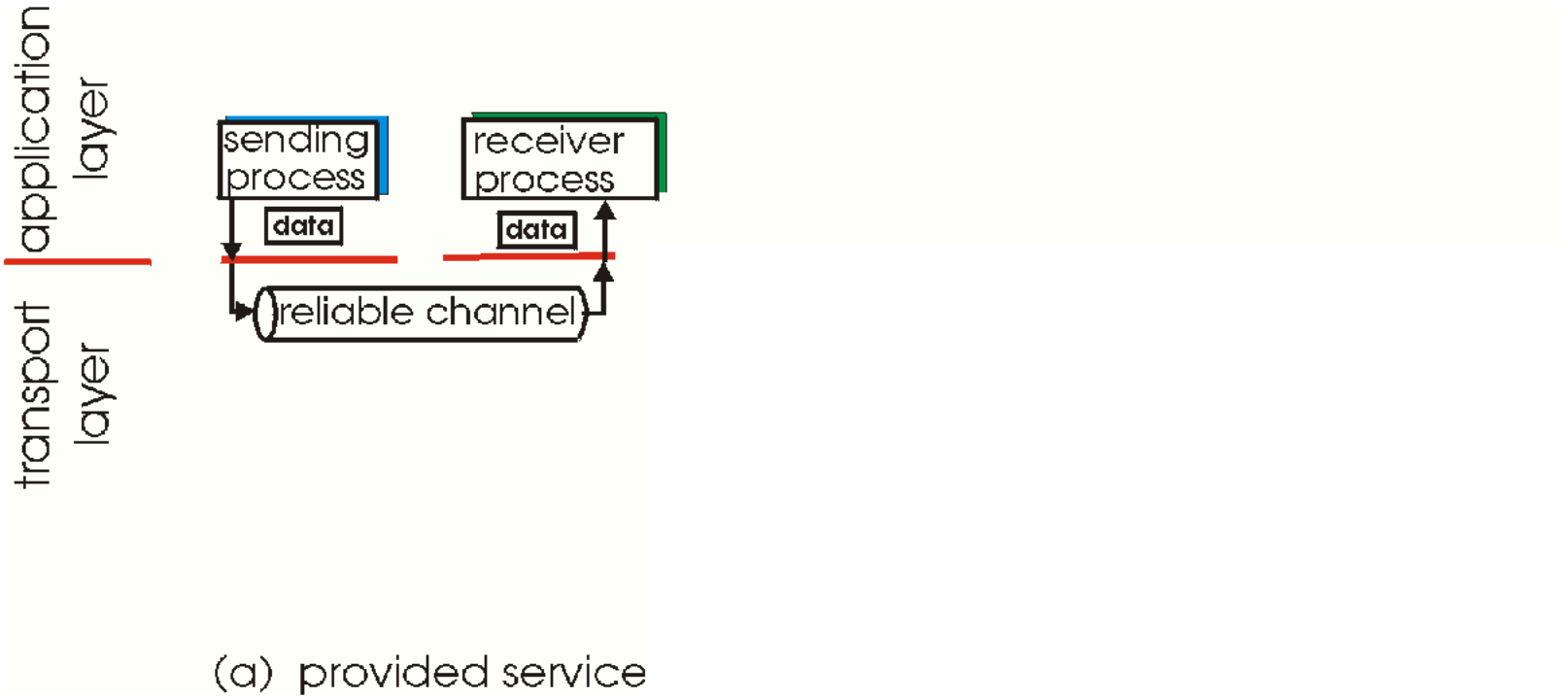
3、UDP的特点？为什么要使用UDP? UDP的数据形式是什么?

4、为什么要有可靠传输？重要性(5.4)

可靠数据传输原理

无论在应用层、传输层还是链路层都是非常重要的

Top-10 list of important networking topics!



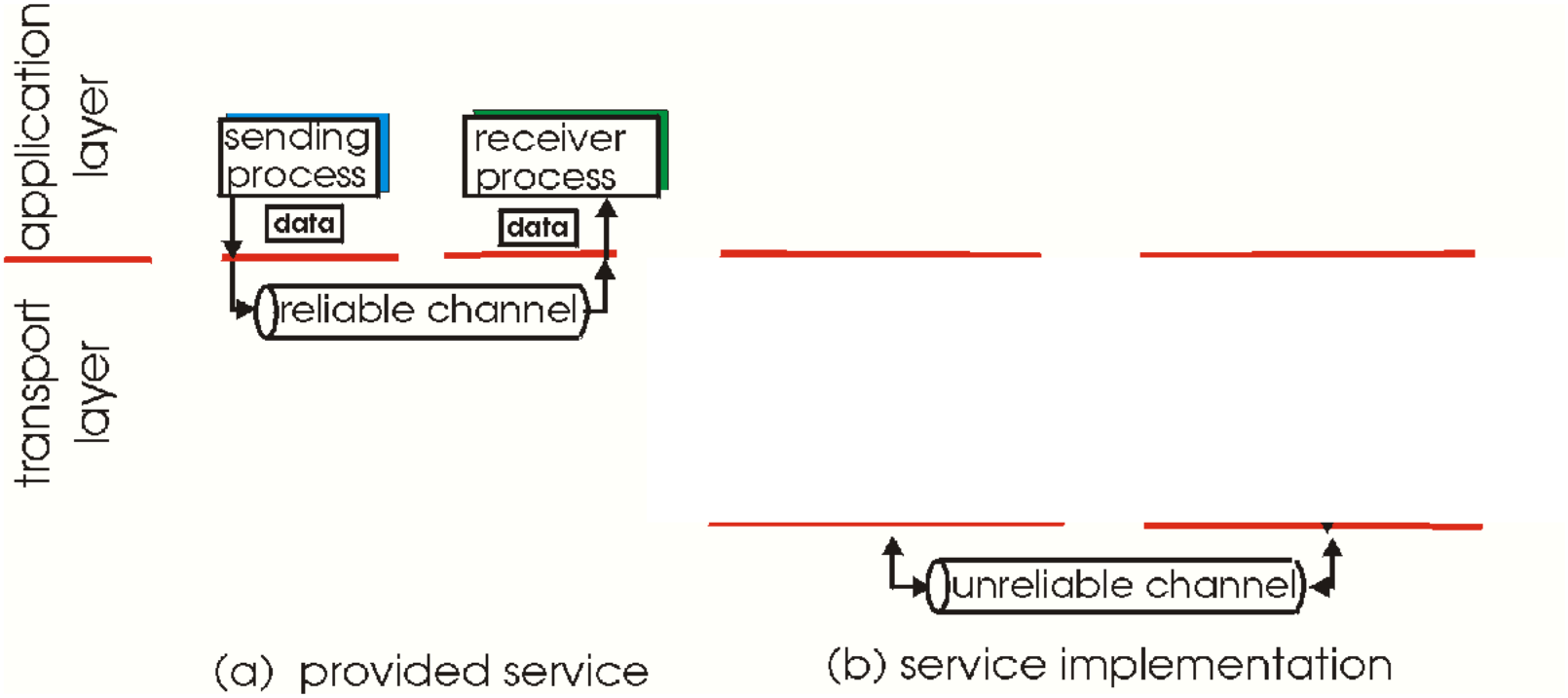
(a) provided service

不可靠信道的特点将决定可靠数据传输协议(rdt)的复杂性

可靠数据传输原理

无论在应用层、传输层还是链路层都是非常重要的

Top-10 list of important networking topics!

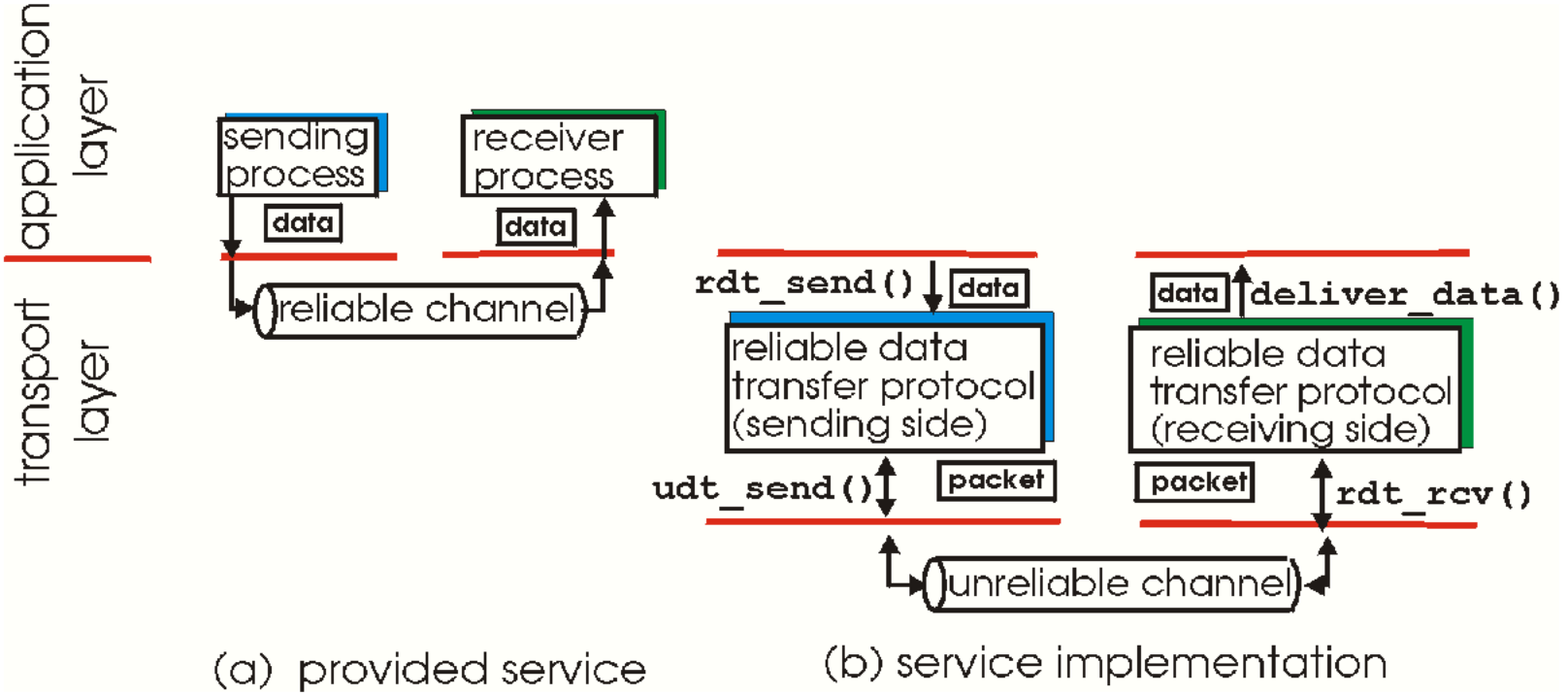


不可靠信道的特点将决定可靠数据传输协议(rdt)的复杂性

可靠数据传输原理

无论在应用层、传输层还是链路层都是非常重要的

Top-10 list of important networking topics!



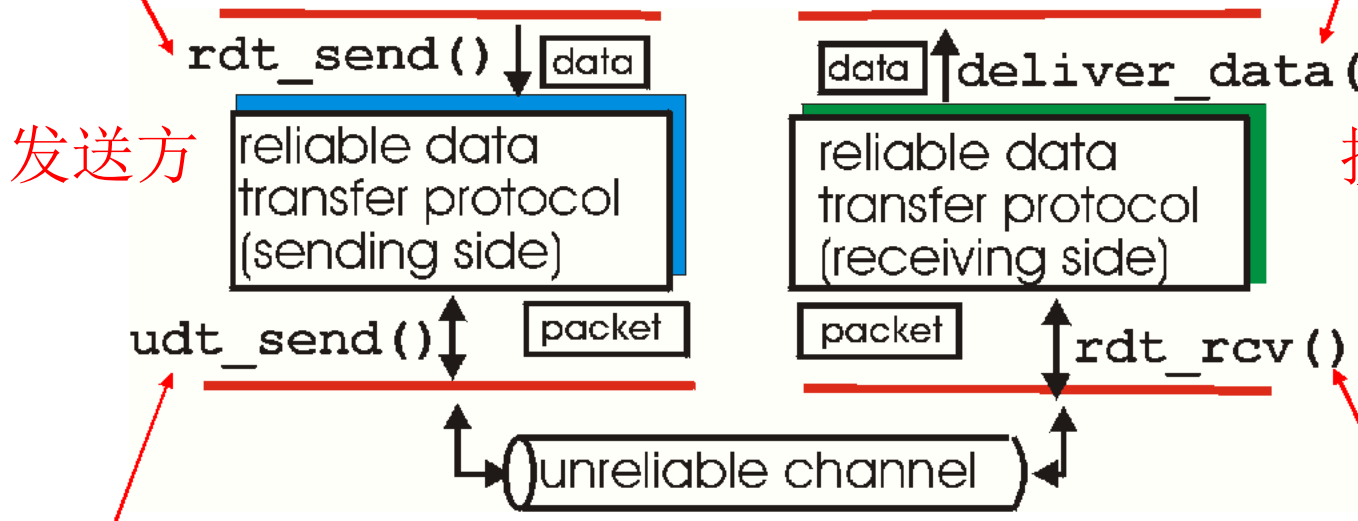
不可靠信道的特点将决定可靠数据传输协议(rdt)的复杂性

rdt:reliable data transfer; udt: unreliable data transfer

可靠的数据传输: getting started

rdt_send(): called from above, (e.g., by app.). Passed data to deliver to receiver upper layer

deliver_data(): called by rdt to deliver data to upper



udt_send(): called by rdt, to transfer packet over unreliable channel to receiver

rdt_rcv(): called when packet arrives on rcv-side of channel

第五章 问题

1、为什么要有传输层，它都能干什么?

核心：建立进程间的逻辑通信

2、如何是建立进程间的逻辑通信?

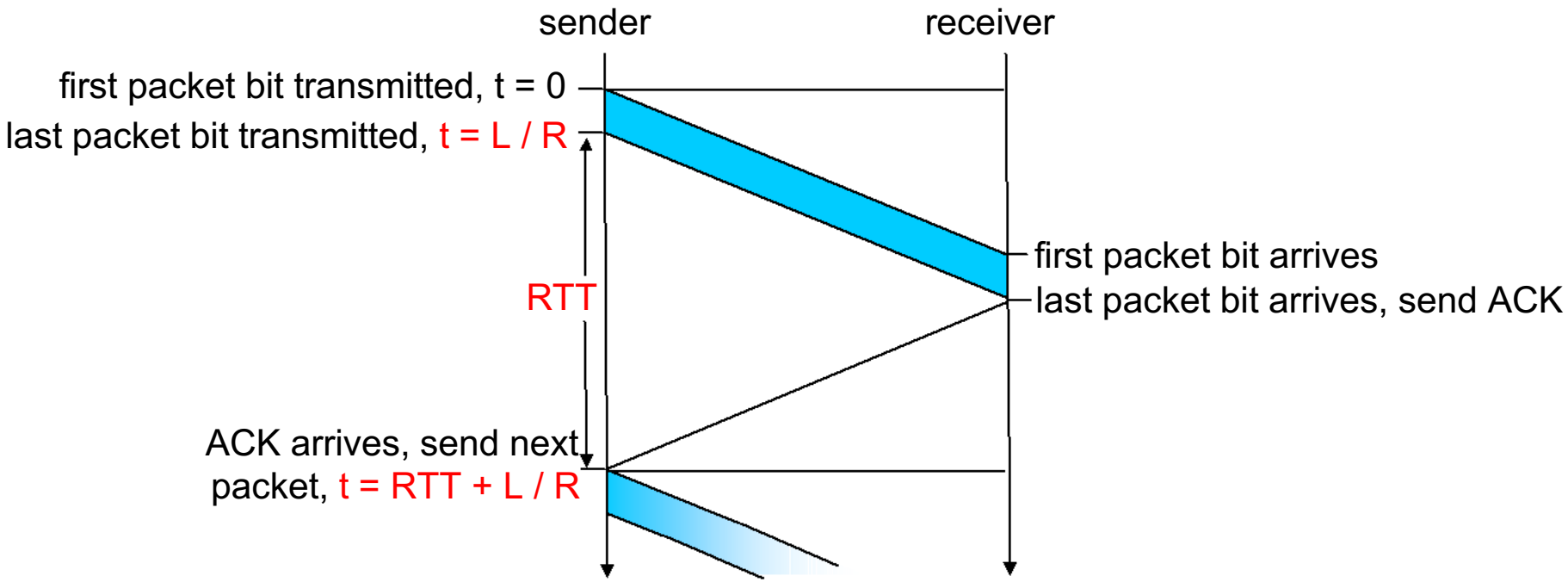
- 通过复用和分用：利用套接字(socket)
- 以UDP协议为例说明如何利用套接字实现复用和分用?
- 以TCP协议为例说明如何利用套接字实现复用和分用？使用四元组将报文段引导到正确地套接字

3、UDP的特点？为什么要使用UDP? UDP的数据形式是什么?

4、为什么要有可靠传输？重要性

5、可靠传输协议（5.4）：停等协议(5.4.1)

rdt3.0: 停-等 stop-and-wait 操作



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

信道利用率非常低！

第五章 问题

1、为什么要有传输层，它都能干什么?

核心：建立进程间的逻辑通信

2、如何是建立进程间的逻辑通信?

- 通过复用和分用：利用套接字(socket)
- 以UDP协议为例说明如何利用套接字实现复用和分用?
- 以TCP协议为例说明如何利用套接字实现复用和分用？使用四元组将报文段引导到正确地套接字

3、UDP的特点？为什么要使用UDP? UDP的数据形式是什么?

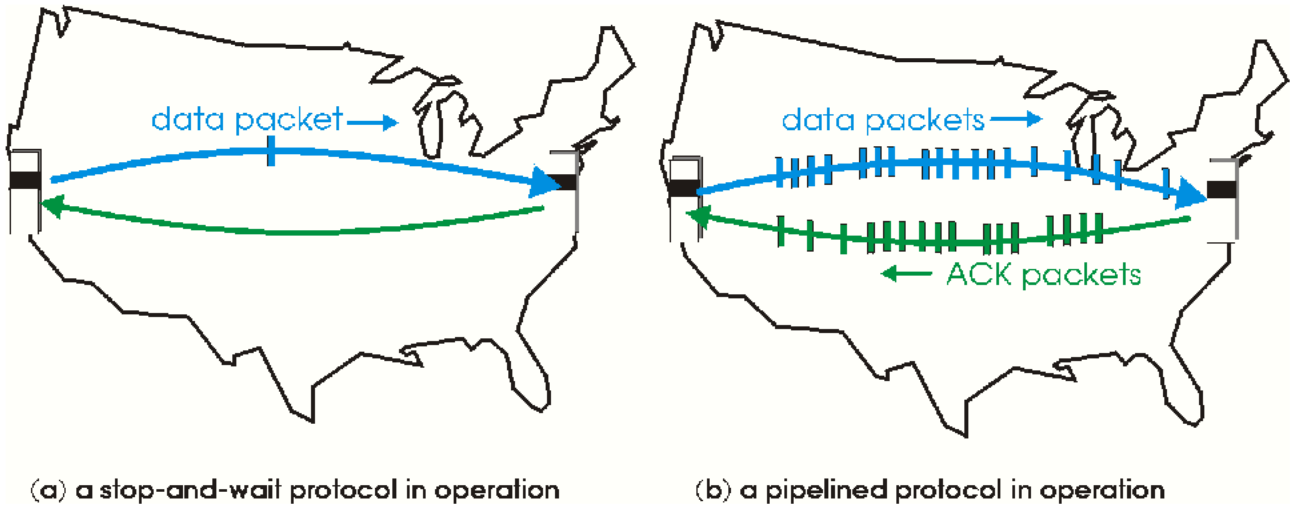
4、为什么要有可靠传输？重要性

5、可靠传输协议回顾：停等协议；流水线协议；

流水线协议 Pipelined protocols

流水线 Pipelining: 发送方允许发送出多个 “in-flight在飞行中”, 还没有得到确认的分组

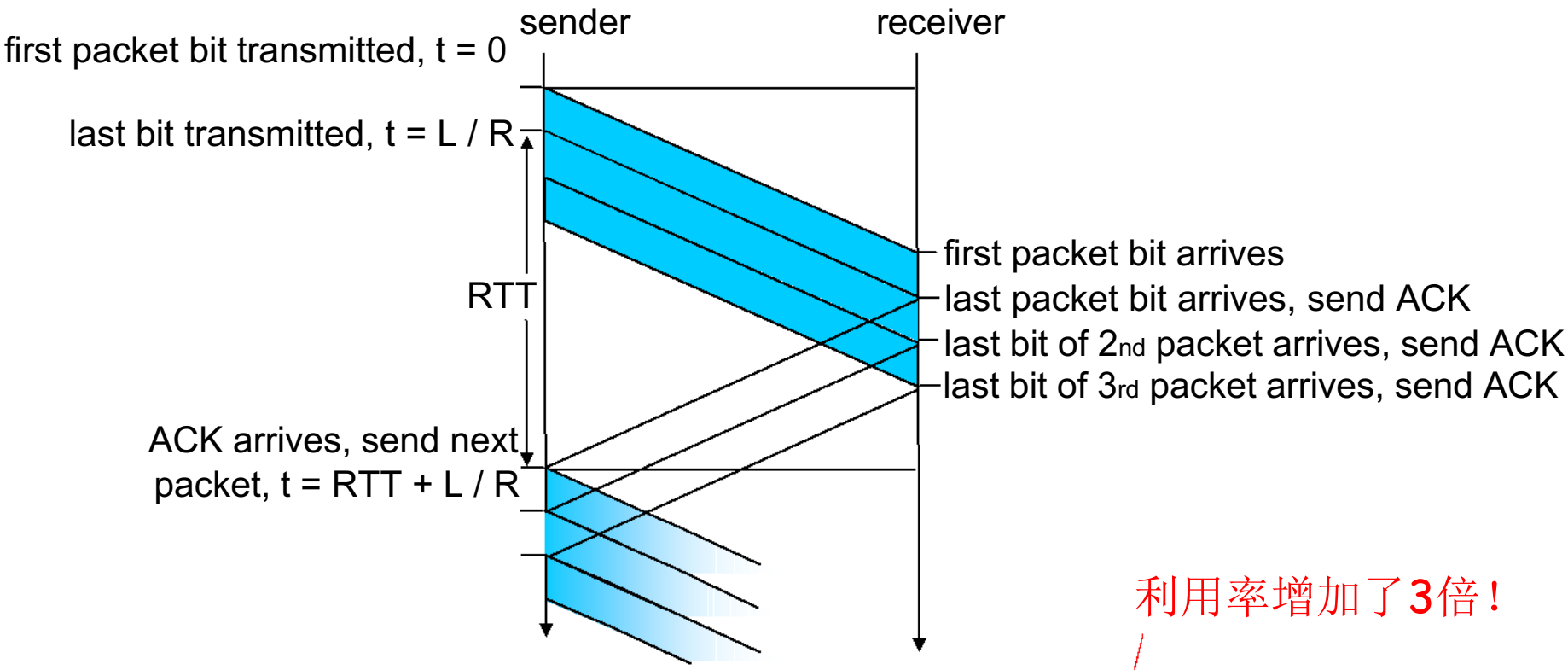
- 顺序号的范围必须扩大
- 在发送方和/或接收方要建立缓存区



(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

流水线协议有两种一般形式: 回退*N*步 *go-Back-N*, 选择性重传 *selective repeat*

流水线：利用率增加了



利用率增加了3倍！

$$U_{sender} = \frac{3*L/R}{RTT + L / R}$$

两类流水线协议比较

Go-Back-N (GBN协议，又称滑动窗口协议):

接收方仅发送累计确认

- 只有当所有序号n之前的分组已经正确接收，才发送n序号分组的ACK

发送方为最早的未确认分组设定计时器

- 一旦定时器超时，重传所有的未确认分组

仅设置1个超时重传定时器

Selective Repeat (SR协议):

接收方逐一分组发送确认信息

发送方为每个未确认分组维持一个计时器

- 当计时器超时，仅重传未确认分组

设置多个超时重传定时器

第五章 问题

1、为什么要有传输层，它都能干什么?

核心：建立进程间的逻辑通信

2、如何是建立进程间的逻辑通信?

- 通过复用和分用：利用套接字(socket)
- 以UDP协议为例说明如何利用套接字实现复用和分用?
- 以TCP协议为例说明如何利用套接字实现复用和分用？使用四元组将报文段引导到正确地套接字

3、UDP的特点？为什么要使用UDP? UDP的数据形式是什么?

4、为什么要有可靠传输？重要性

5、有限状态机?

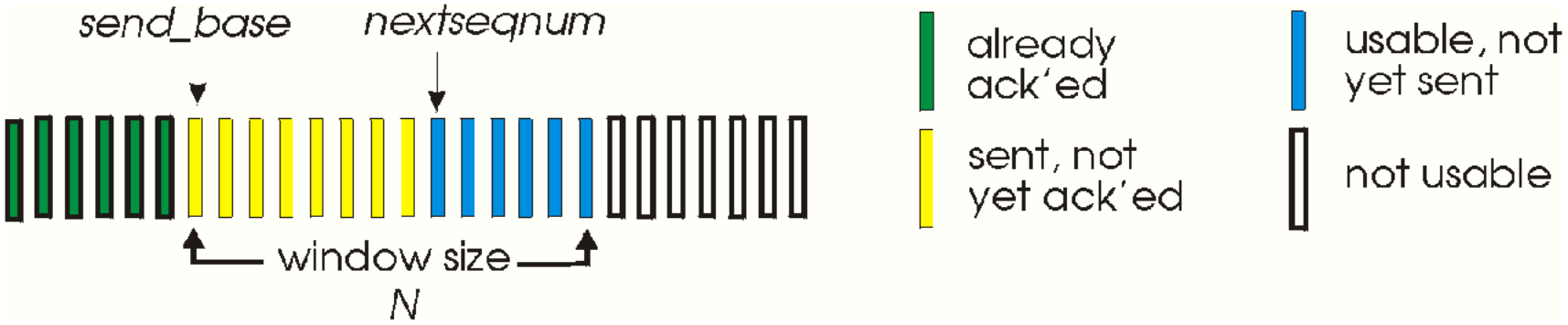
6、可靠传输协议回顾：停等协议；流水线协议；GBN协议（滑动窗口协议）的实现(5.4.2)；

GBN协议（滑动窗口协议）

发送方

在分组的报头部分有k位的顺序号，可连续发送窗口内N个未收到确认的分组

"window" of up to N, consecutive unack'ed pkts allowed



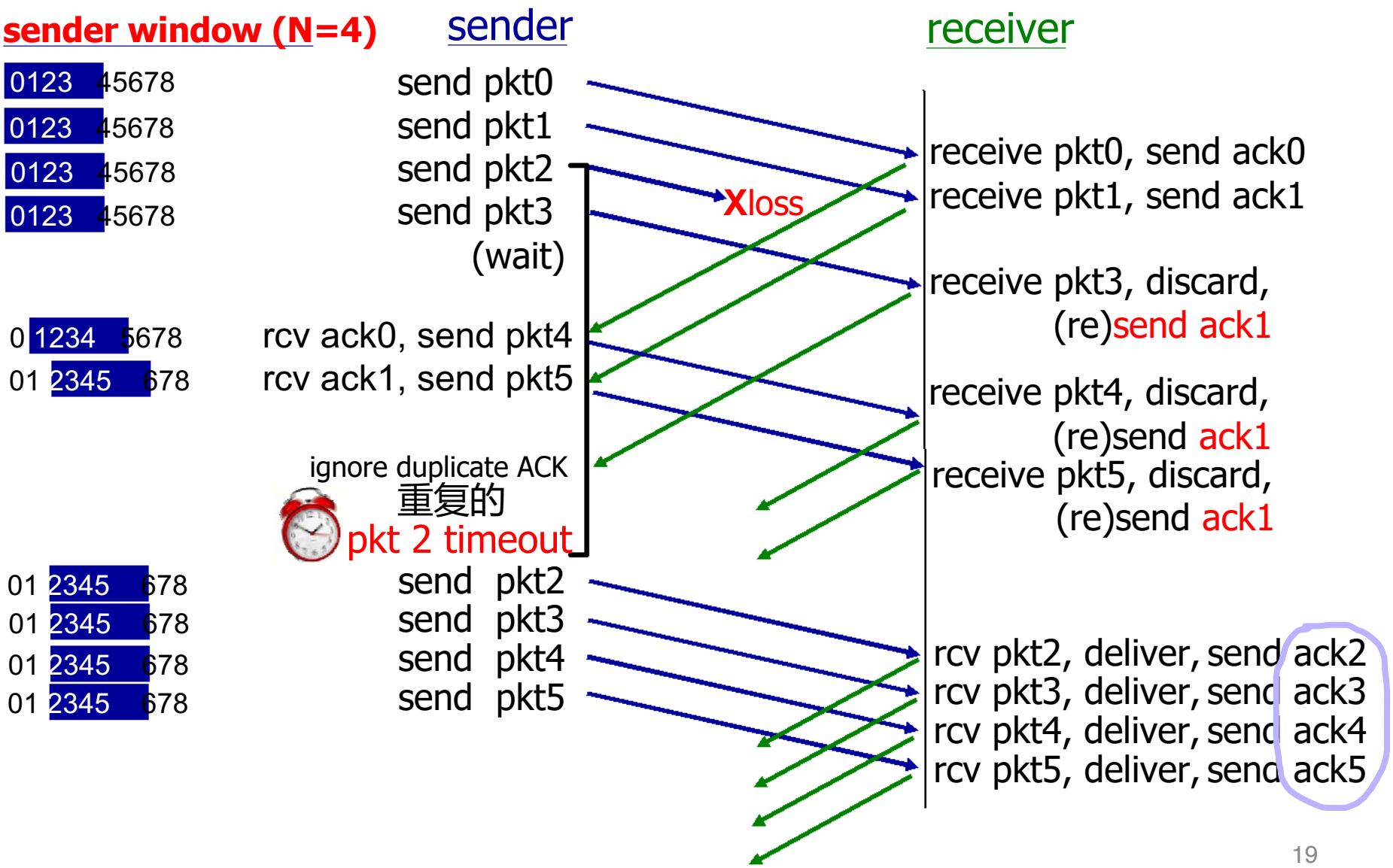
ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"累积确认

- 只有当所有序号n之前的分组已正确接收，才发送n序号分组的ACK
- 为最早已发送的但还未被确认的分组设置一个定时器(图中如何设置?)

timeout(n): retransmit pkt n and all higher seq # pkts in window

定时器触发，重传窗口内所有的包括分组n在内的分组

GBN 实例


sender window (N=4)
sender
receiver
0123 45678
0123 45678
0123 45678
0123 45678
send pkt0
send pkt1
send pkt2
send pkt3
(wait)
Xloss
receive pkt0, send ack0
receive pkt1, send ack1
receive pkt3, discard,
(re)send ack1
0 1234 5678
01 2345 678
rcv ack0, send pkt4
rcv ack1, send pkt5
receive pkt4, discard,
(re)send ack1
receive pkt5, discard,
(re)send ack1
ignore duplicate ACK
重复的
pkt 2 timeout
01 2345 678
01 2345 678
01 2345 678
01 2345 678
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

第五章 问题

1、为什么要有传输层，它都能干什么?

核心：建立进程间的逻辑通信

2、如何是建立进程间的逻辑通信?

- 通过复用和分用：利用套接字(socket)
- 以UDP协议为例说明如何利用套接字实现复用和分用?
- 以TCP协议为例说明如何利用套接字实现复用和分用？使用四元组将报文段引导到正确地套接字

3、UDP的特点？为什么要使用UDP? UDP的数据形式是什么?

4、为什么要有可靠传输？重要性

5、有限状态机?

6、可靠传输协议回顾：停等协议；流水线协议；GBN协议（滑动窗口协议）；选择重传协议（SR协议）

选择性重传 Selective Repeat (SR协议)

接收方个别确认所有正确接收的分组

- 为了最终按序递送到上一层，需要时将对分组进行缓存 buffers pkts, as needed, for eventual in-order delivery to upper layer

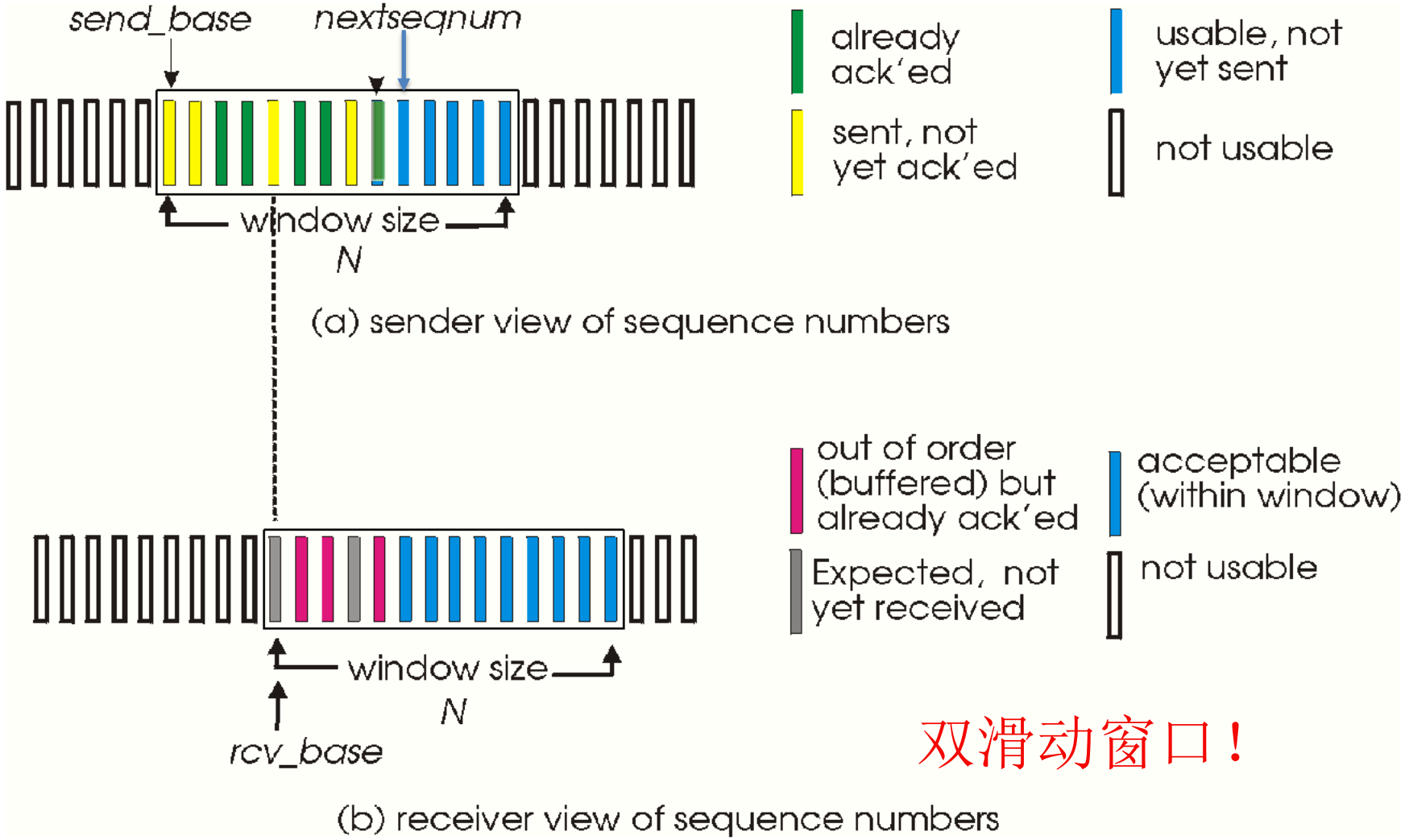
发送方仅重传那些没有按序接收到 ACK 的分组

- 为每个未确认的分组设置一个定时器，（而GBN协议只需为第一个没有收到ACK的分组设定一个超时定时器）

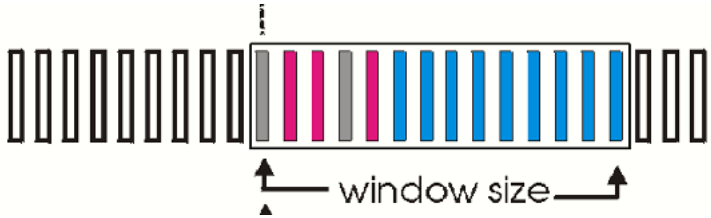
发送窗口

- 可发送最大N个连续序号区间内的分组
- 已发送但未确认的分组数量 ≤ N

选择性重传：发送窗口和接收窗口



选择性重传(自学)



sender

上层调用发送数据data from above :

如果滑动窗口中有可用序号，则发送
if next available seq # in window, send pkt

分组n超时 timeout(n):

重发分组n，并重启其定时器 resend pkt n, restart timer

分组n确认 ACK(n) in [sendbase,sendbase+N]:

将分组n标识为已确认 mark pkt n as received

如果n是最小的未确认分组序号，将其序号推进至下一最小未确认分组序号 (向后滑动发送窗口)

if n is the smallest unACKed pkt, advance window base to next unACKed seq #

receiver

收到序号n的分组

pkt n in [rcvbase, rcvbase+N-1]

- 发送分组n的确认 send ACK(n)
- 乱序out-of-order: 缓存 buffer
- 按序 in-order: 组装数据传给上层应用，且将窗口基序号推进至下一未接收分组序号

收到序号n的分组

pkt n in [rcvbase-N,rcvbase-1]

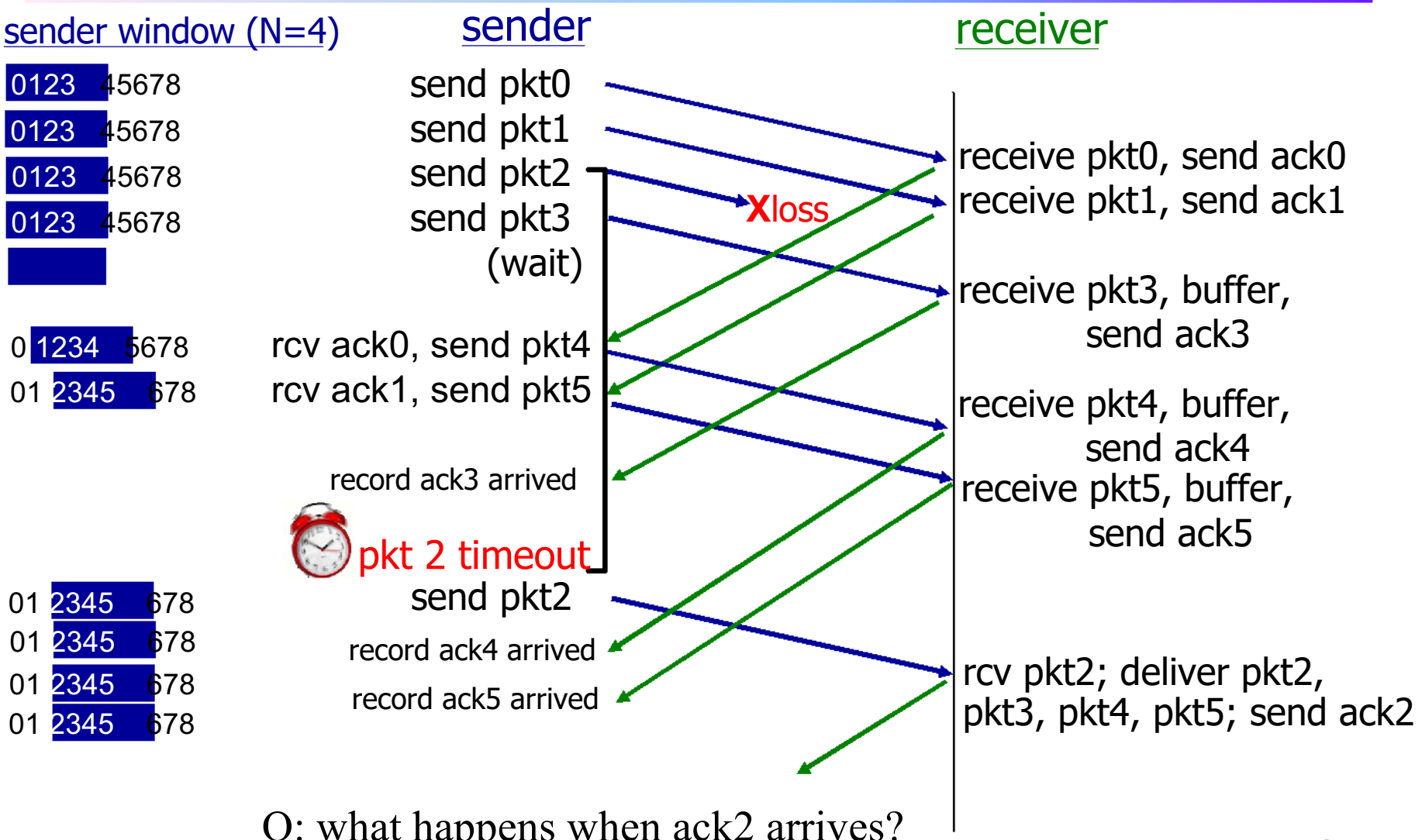
发送分组n的确认 ACK(n)

otherwise:

忽略 ignore

可能ACK丢包，导致发送方重传小于当前基序号的分组

选择性重传实例（自学）



Q: what happens when ack2 arrives?