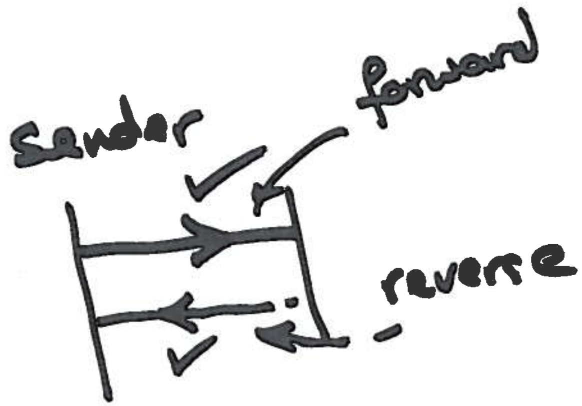
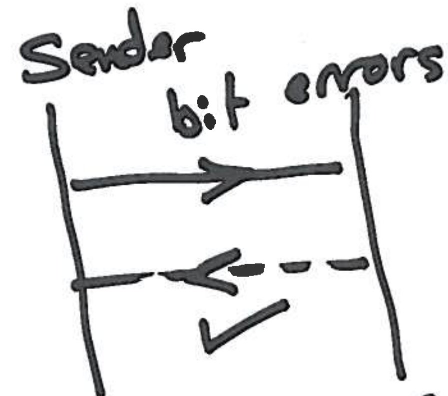
- For simplicity, assume unidirectional data transfer

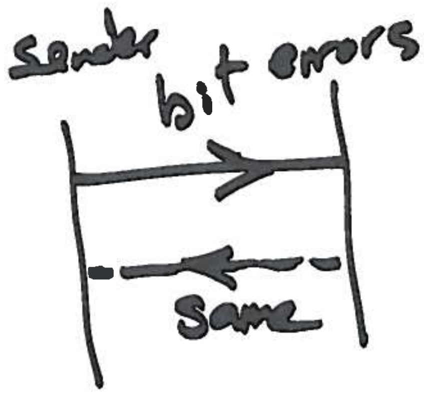
- Rdt 1.0



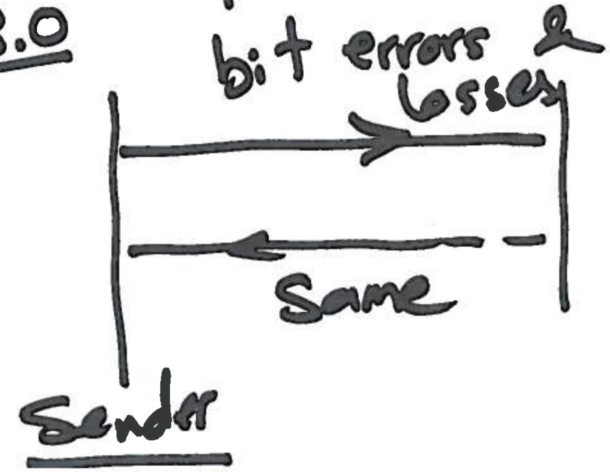
Rdt 2.0



Rdt 2.1
2.2



Rdt 3.0



How to present a protocol?

- Extended Finite State Machine

- Event : Action
  ⋮ } → **CNET**

Rdt 1.0

Q: What goes in pkt = make_pkt (data)?

- Look at "application.c" in CNET

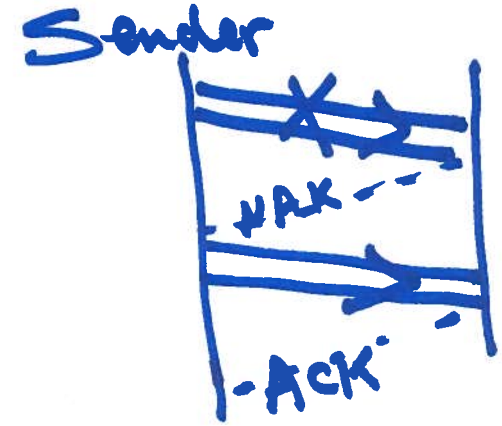
```
struct {
    int     to, from, checksum, session,
            seqno
    size_t  length
    char    data [....]
} MESSAGE
```

- Stop-and-wait.c

```
struct {
        FRAMEKIND    kind;      ← DL_DATA
                                  DL_ACK
        size_t       len;
        int          checksum;
        int          seq;       ← 0 or 1
        MSG          msg;       ← app. layer
                                  message
} FRAME
```

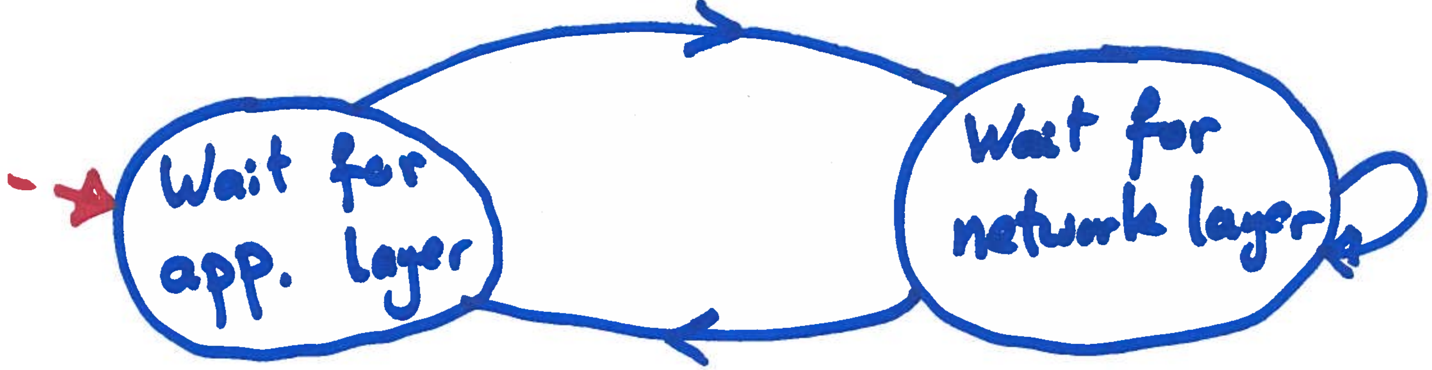
Rdt 2.0

Sender



event: message available

Send



event: pkt = ACK

Λ ← do nothing

event: pkt = NAK

- ~~extract data~~
- ~~deliver data~~

retransmit

Receiver

event: pkt error

- send NAK

event: pkt is OK

- extract data
- deliver
- send ACK

Programming in CNET

function name

- static EVENT_HANDLER (name)

→ name (CnetEvent ev,
CnetTimerID timer,
CnetData data)

```
#include <cnet.h>
static CnetTimerID last_timer;
static EVENT_HANDLER (application_ready)
{ . . . . . . }
```

Visible only within this file.

=

```
static EVENT_HANDLER (reboot_node) {
    ...
    CNET_set_handler (EV_APPLICATION
                      READY,
                      application_ready,
                      0
                     )
    ....
}
```

Packet Error Rate (PER) vs. Bit Error Rate (BER)

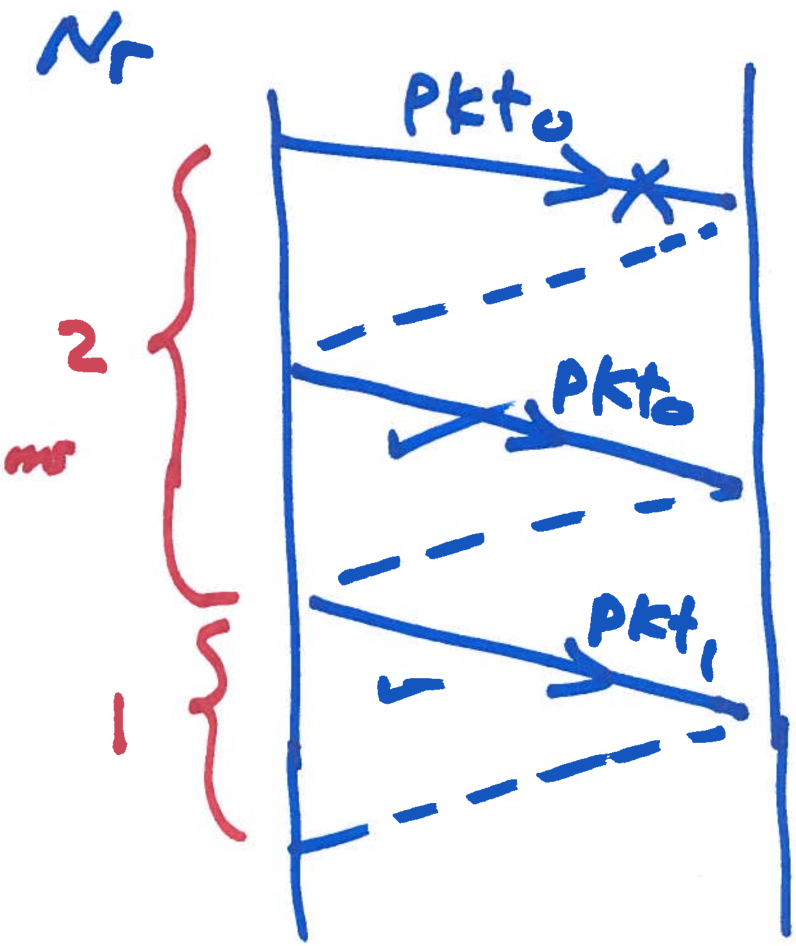
$100 =$ $n =$ pkt length (bits)

$3 =$ $k =$ max. # of bit errors that the error correcting code can handle

$$P[\text{a pkt is transmitted successfully}] = \sum_{i=0,1,2,\underline{\underline{3}}} \binom{n}{i} q^{i} (1-q)^{n-i}$$

$q =$ bit error prob.

BW-delay product = $2aL$



$$\overline{N_r} = \frac{2+1}{2 \text{ pkts}} = 1.5 \quad \underline{\underline{\text{transmissions}}}$$

$$\overline{Thr}_{errors} = \frac{Thr_{error\_free}}{\overline{N_r}}$$

- <u>Simple reasoning</u>?

- $Thr_{error} = \frac{Thr_{error\_free}}{N_r}$

r.v.

$$Y = \frac{\text{Constant}}{X} \overset{?}{\not\Longrightarrow} E[Y] = \frac{\text{Constant}}{E[X]}$$

$$X = \begin{cases} \cdot & - \\ \cdot & - \end{cases}$$

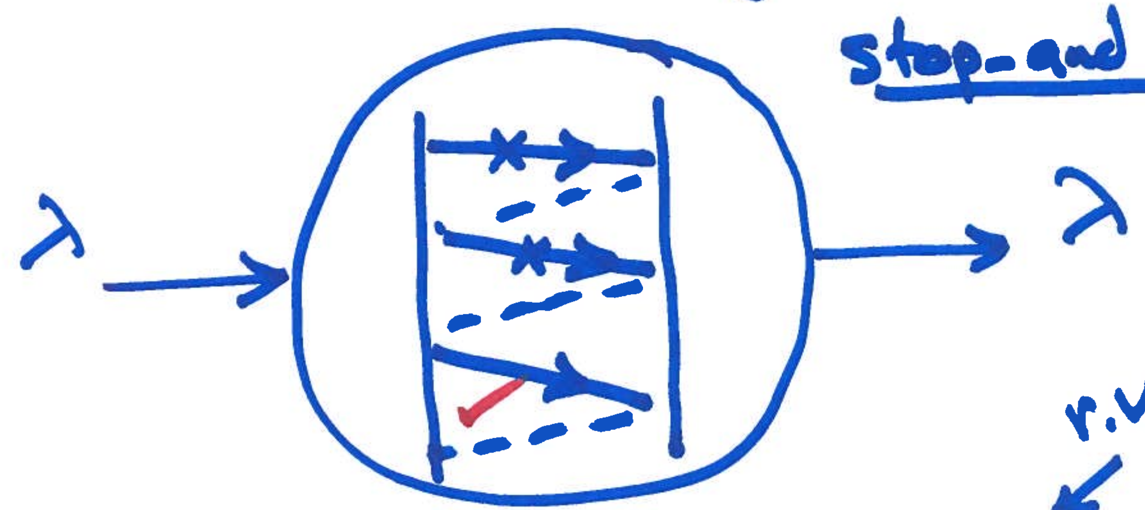
$$\boxed{\begin{array}{l} Y = \text{Const.}\ X \\ E[Y] = \text{Const.}\ E[X] \\ \checkmark \end{array}}$$

- Another <u>argument</u>?

Little's Theorem

Single server

stop_and_wait



$$\bar{N} = \lambda \, \bar{T}$$

If the protocol is continuously busy Then

$\bar{N} = 1$

avg. throughput

$T = N_r * (T_{pkt} + RTT)$

N_r: r.v.

T: r.v.

$\underbrace{\;T_{pkt} + RTT\;}_{\text{Contant}}$

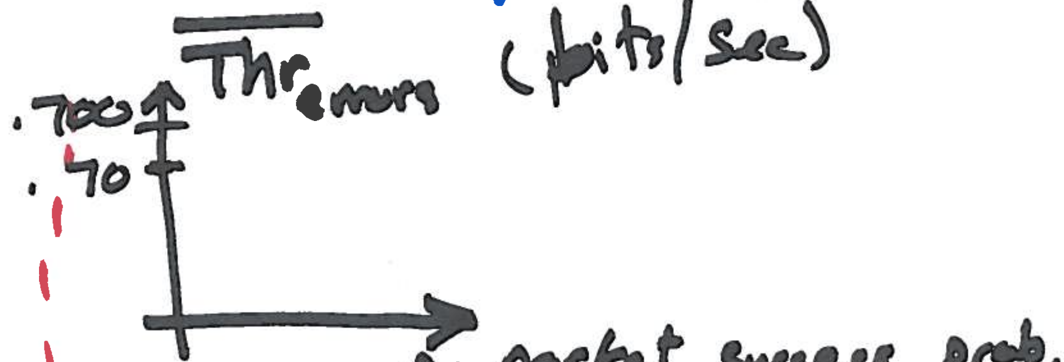
$E[T] = E[N_r] * \cdot$

$= \bar{N}_r * (T_{pkt} + RTT)$

$$\therefore \lambda = \frac{1}{\cdot}$$

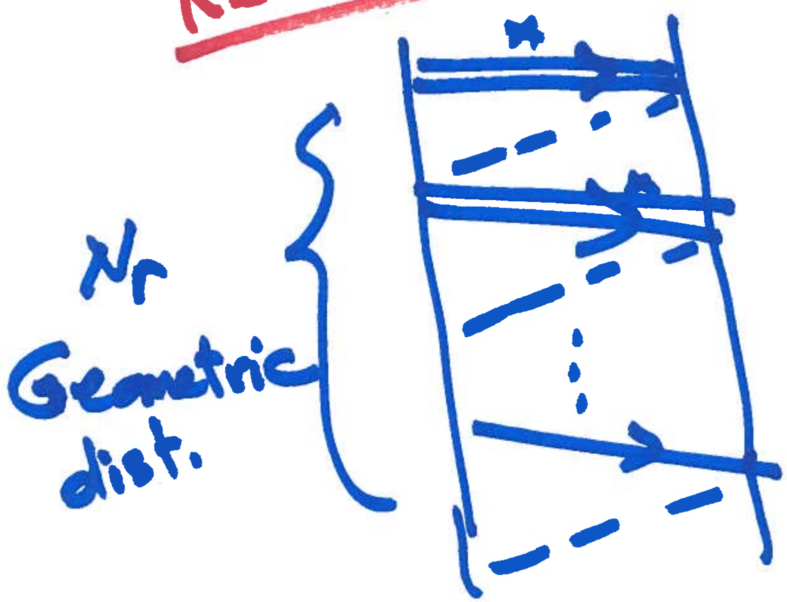
Recap

$$\overline{Thr}_{errors} = \frac{\overline{Thr}_{error\_free}}{\overline{N_r}} = \frac{1}{\overline{N_r}\,(T_{pkt} + RTT)}$$



Today

Remark #1

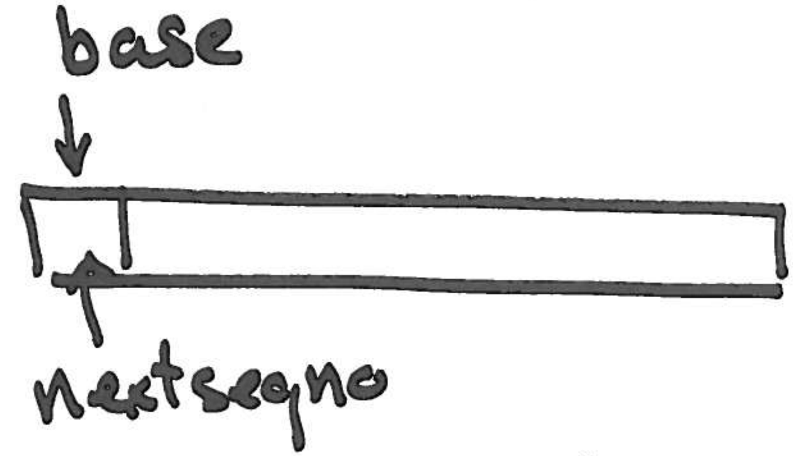


$$\overline{Thr}_{norm} = \frac{\overline{Thr}_{errors}}{Thr_{max}} = \frac{\frac{\overline{Thr}_{no\ errors}}{\overline{N_r}}}{\frac{1}{T_{pkt}}} = \cdots \vdots$$

$$= \frac{1}{\overline{N_r}(1+2a)}$$

Go-Back-N

- Initially:



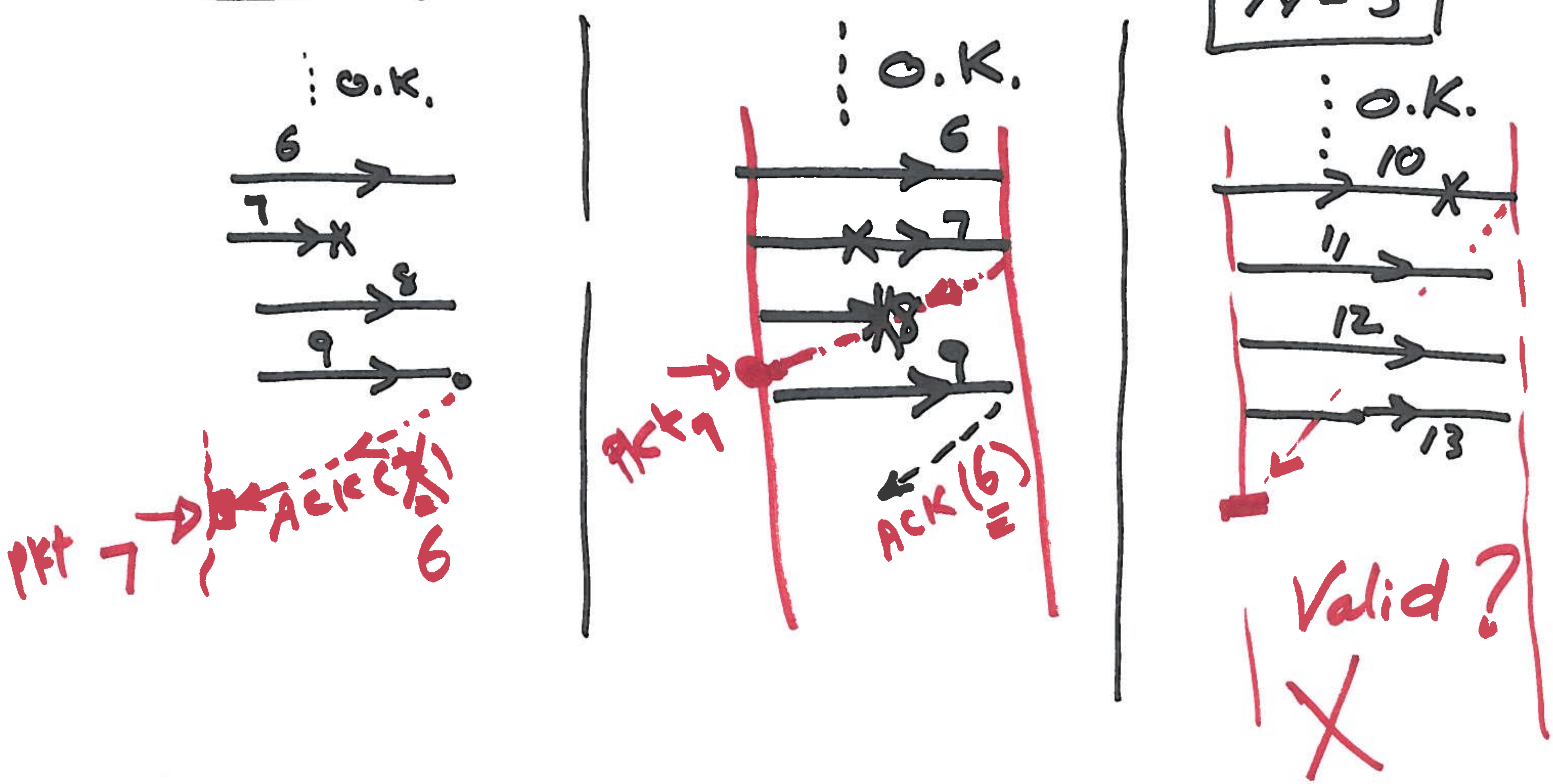
if (base == nextseqno)
$\Rightarrow$ no in-flight pkt

- [base, nextseqno − 1] : range of pkts in-flight

scenarios

- ACK <u>number</u>
- Which pkt is being timed out at a certain point.

$\boxed{N=3}$



<u>event:</u> receive ACK(x)

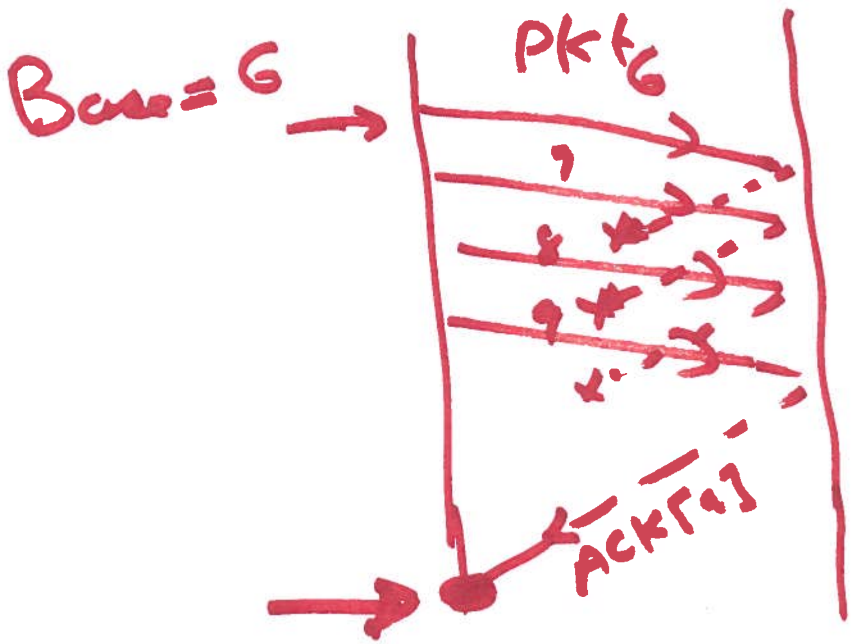
- base = x+1

⋮

<u>event:</u> receive ACK(base)

- base ++

⋮



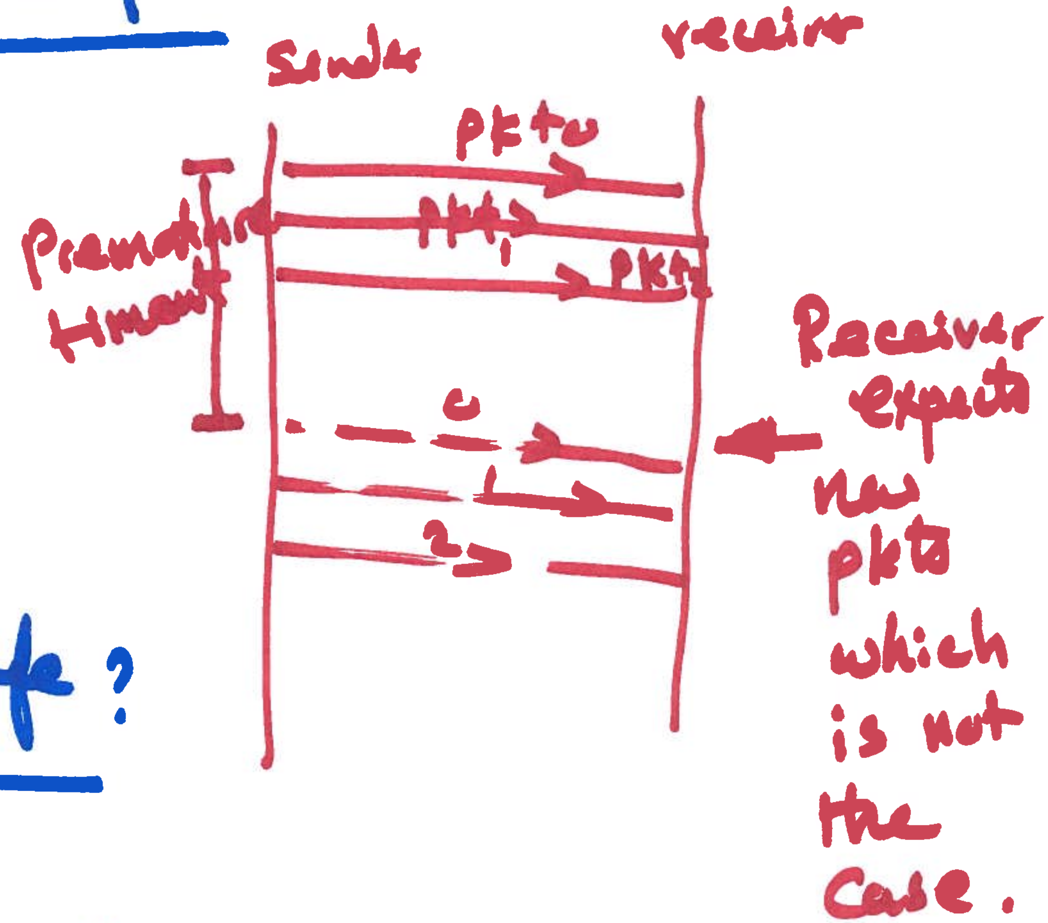
Go-Back-N

Is $M = N$ safe?

$\boxed{N=3}$

So, $M=3 \Rightarrow$

seq no $= \{0,1,2\}$



Is $M = N+1$ safe?

So $M=4 \Rightarrow$

seq no $= \{0,1,2,3\}$

S&W

$$\overline{Thr}_{errors} = \frac{\overline{Thr}_{no\_error}}{\overline{N}_r}$$

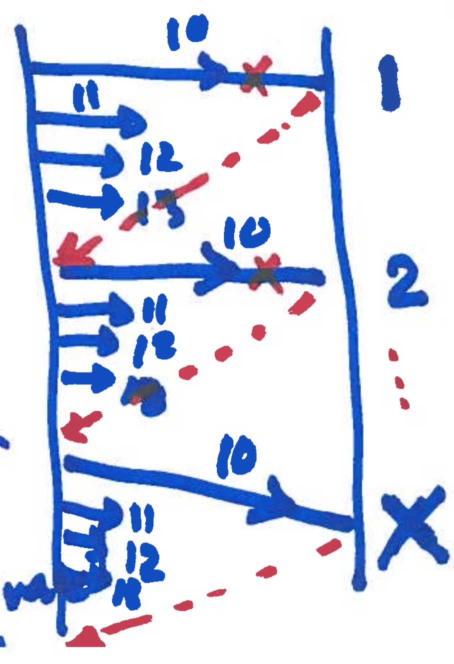
Avg. # of transmissions required to transmit one pkt successfully

GBN

$$\overline{Thr}_{error} = \frac{\overline{Thr}_{no\_errors}}{\overline{N}_x}$$

Avg. # of pkts transmitted to deliver one pkt successfully.

$N=4$



Scenario #1

pkt 10 is delivered correctly

Scenario #2

Pkts 10,11,12,13 are delivered correctly

Scenario #1: Only pkt_{10} is delivered

$$\overline{Thr}_{error\text{-}free} = \frac{4 \text{ pkts}}{1+2a}$$

- **By inspection**

$$\overline{Thr} = \frac{1 \text{ pkt}}{X(1+2a)}$$

- **Using Formula**

pkt	N_x
10	$4(X-1)+1$

$$\overline{Thr} = \frac{\overline{Thr}_{error\text{-}free}}{\overline{N}_x} = \frac{\frac{4}{1+2a}}{4(X-1)} \approx \frac{1}{(1+2a)(X-1)}$$

Scenario #2 pkts 10, 11, 12, 13 are delivered

- **By inspection**

$$\overline{Thr} = \frac{4 \text{ pkts}}{X(1+2a)}$$

- **Using Formula**

pkt	N_x
10	$4(X-1)+1$
11	1
12	1
13	1

$$\overline{N}_x = \frac{4X \text{ pkts}}{4} = X$$

So, $\overline{Thr} = \frac{\frac{4}{1+2a}}{X} = \frac{4}{X(1+2a)}$

CNET timers

10 timer events : $\underbrace{\text{EV\_TIMER1}}_{}$ .... EV_TIMER10

10 queues

- reboot_node:

  ⋮

  CHECK(CNET_set_handler(
  EV_TIMER5, on_timer5, 0))

  ⋮

- physical_ready:

  ⋮

  100 → t1 = CNET_start_timer
  (EV_TIMER5, 1000, 0)

  ⋮

  105 → t2 = CNET_start_timer
  (EV_TIMER5, 3000, 0)

  ⋮

- EVENT_HANDLER (on_timer5)

  ↓

  void on_timer5(CnetEvent event,
  CnetTimerID → timer,
  CnetData data)

  ⋮

  if (timer == t1)
  Cnet_stop_timer (t1)

  ⋮

  if (timer == t2)
  ....

Correctness of a sliding window protocol with N_{sender} & $N_{recv.}$

$N_{send} = N_{recv} = \underline{N}$

Is $M = N+1$ safe?

Case: $N = 3$
$M = 4$ (i.e., $\{0,1,2,3\}$)

No!

$\dots \boxed{3 \ 4 \ 0}$

Is $M = N+2$ safe?
$N=3$
$M=5$ (i.e., $\{0,1,2,3,4\}$)

$\boxed{\text{So, } N > \frac{M}{2} \text{ is unsafe}}$

Is $M = N+3$ safe?
$N=3$
$M=6$ $\{0,1,2,3,4,5\}$

$\dots \boxed{3 \ 4 \ 5}$

$M \geq 2N$ is safe

Searching for an expression of the form: if $M \geq f(N_{send}, N_{recv})$ then operation is safe.

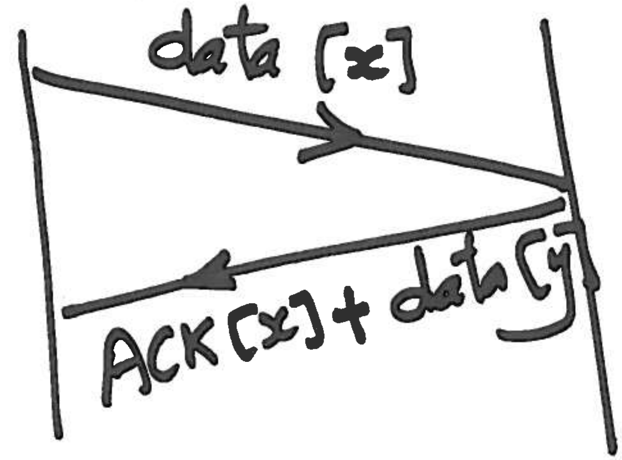
Guess: $2\min(N_{send}, N_{recv})$

$= N \quad 1$

What if: $M \geq N_{sender} + N_{recv.}$?

	$M \geq$	N_{sender}	$N_{recv.}$
S&W:	2	1	1
GBN:	$N+1$	N	1
SR:	$2N$	N	N

Piggybacking



Dealing with the Delayed Duplicate Problem:

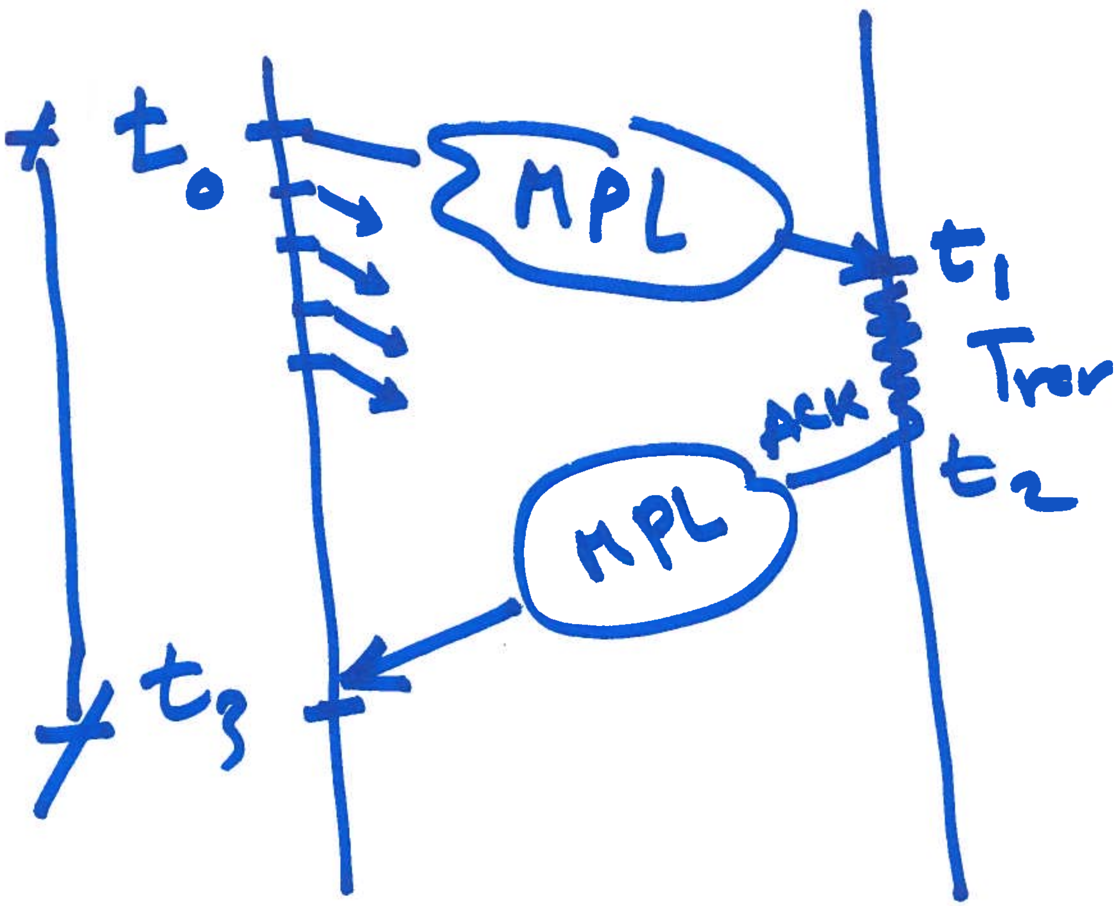
Objective: Ensure that no two pkts with the same <seqno, connection> are alive at the same time in the Internet.

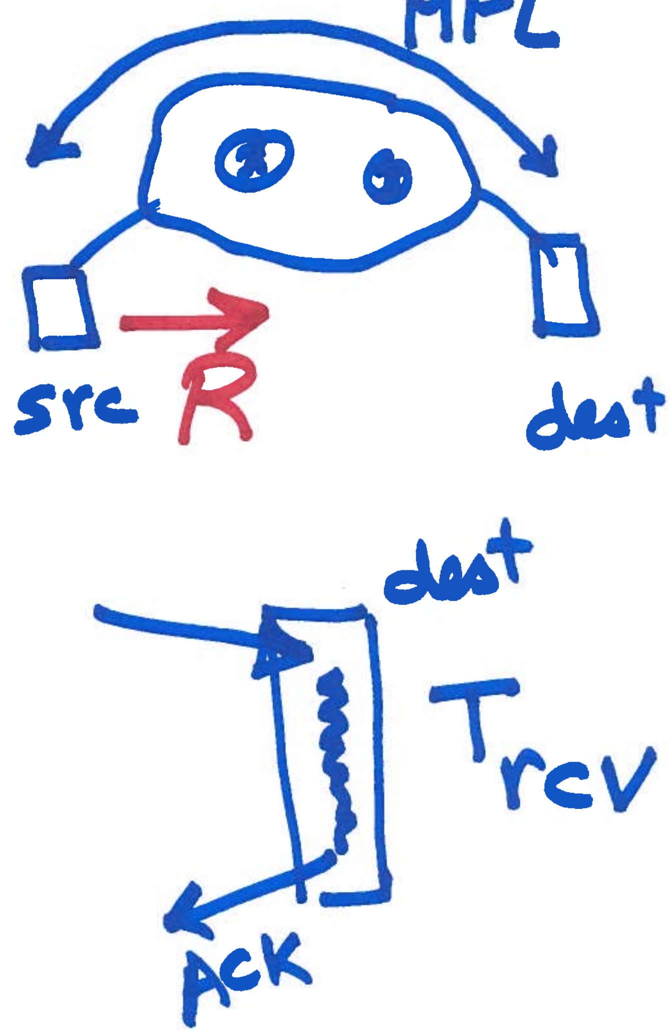
TCP

<src IP#, src PORT#, dst IP#, dst PORT#>

possible

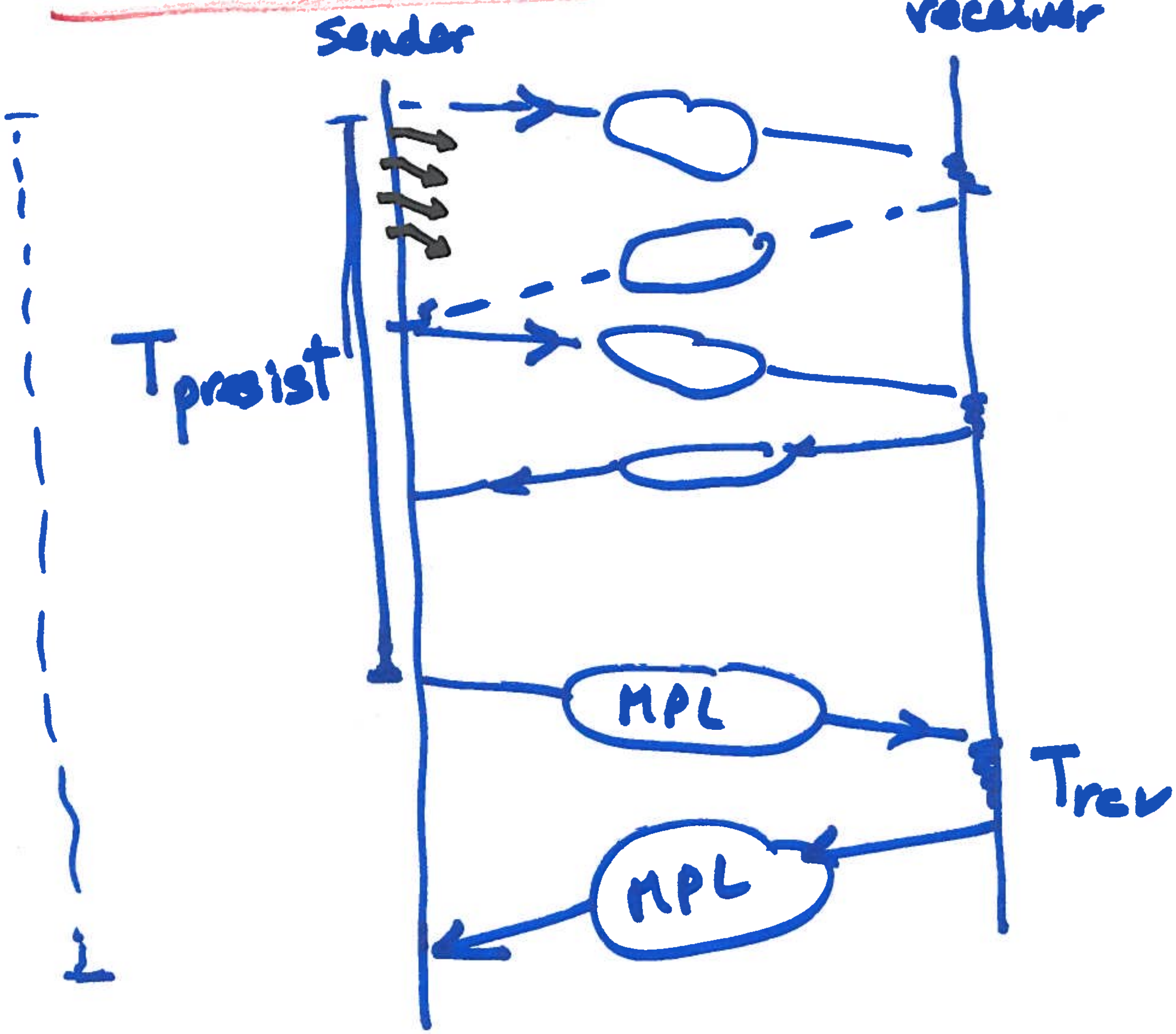
<u>Framework:</u> How large should the seq. space be?





$(2\,MPL + T_{rcv}) \cdot R$

Revised framework



$$\left(\underline{\underline{T_{presist}}} + 2MPL + T_{rcv} \right) \cdot R$$

Reliable Release

Objective: All data & ACKs are delivered to both sides

Circuit Switching: Not a problem

App. layer

write() ← Get my balance

timeout ⋮

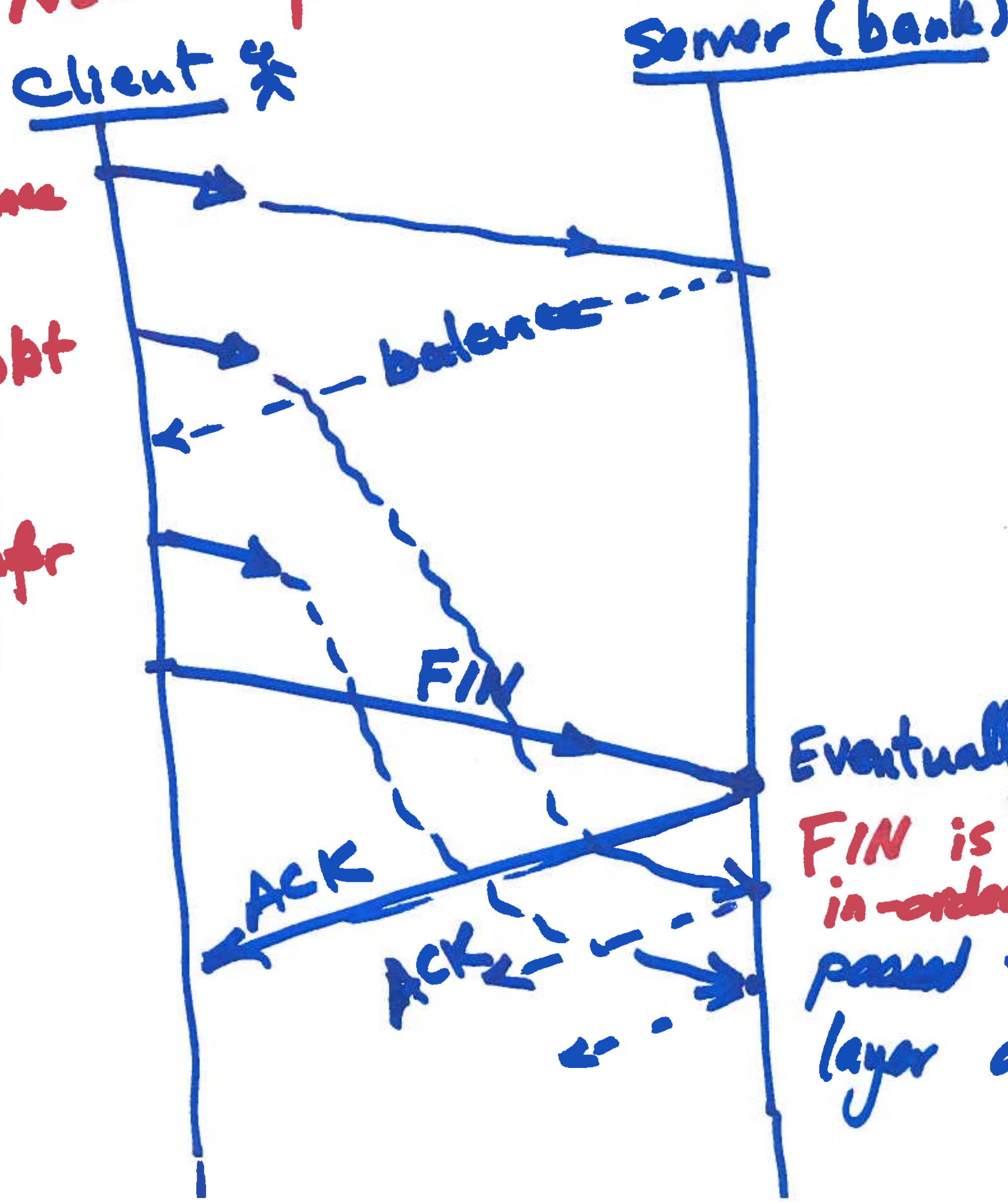
Resend the pkt

read(): balance

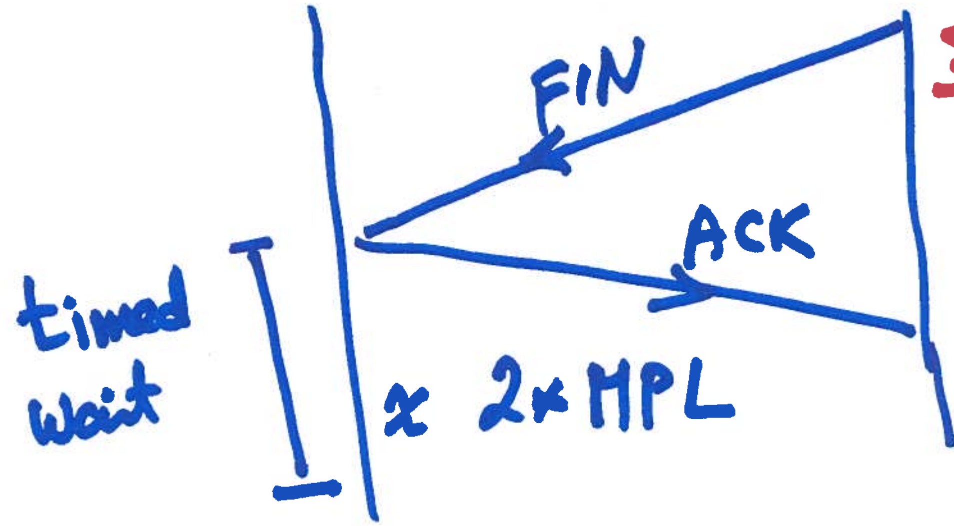
write() ← fund transfer

close() ← Socket

state: Active Close



Eventually (when FIN is processed in-order), FIN is passed to the app. layer as end-of-file



State: passive close

<u>During timed-wait</u>

- Client maintains state in order to resend the last ACK
- Cannot start "<u>incarnation</u>" of the connection

  ≡ Connection with the same port #

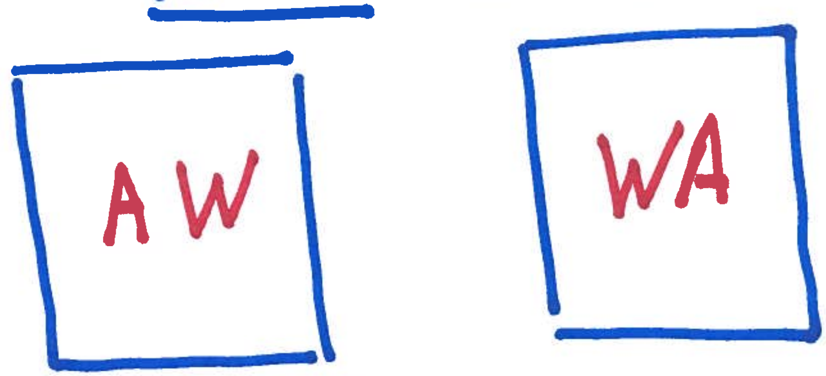
Crash Recovery

- Not in text book
- Not on any exam.

Context

Sender: Client

Receiver: Server



- 

ACK: A
Write: W
to upper layer

- Can be in either

SO: Received ACK to last transmitted pkt

- Crashes: C
- After rebooting: Server broadcasts a "rescue me" message asking for help

S1: Waiting for ACK to the last transmitted pkt.

- C(AW)
- AC(W)
- AWC

- C(WA)
- WC(A)
- WAC