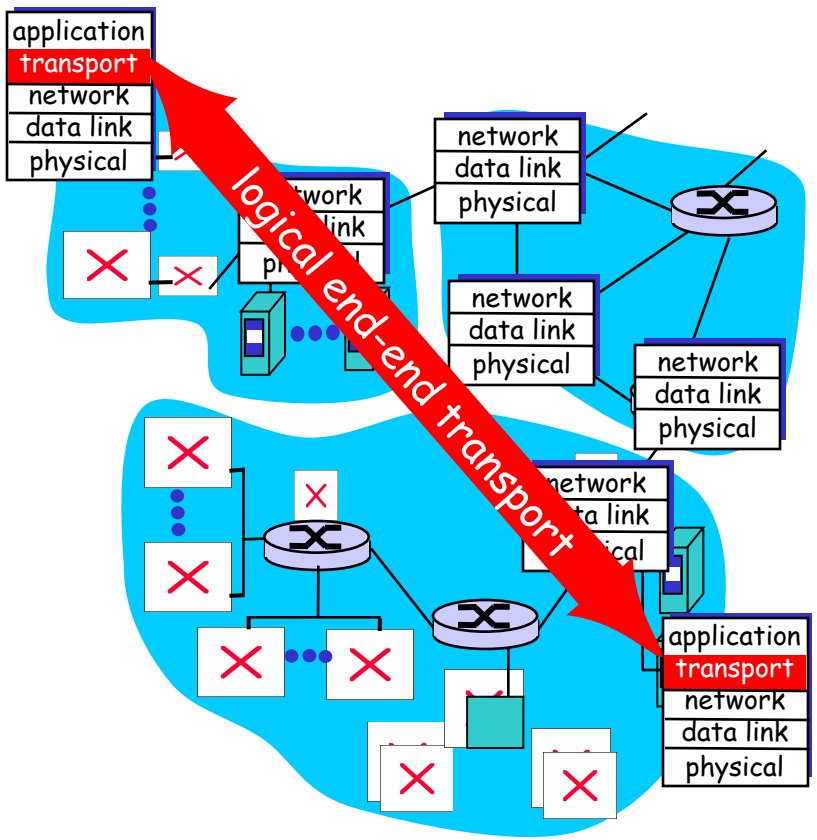
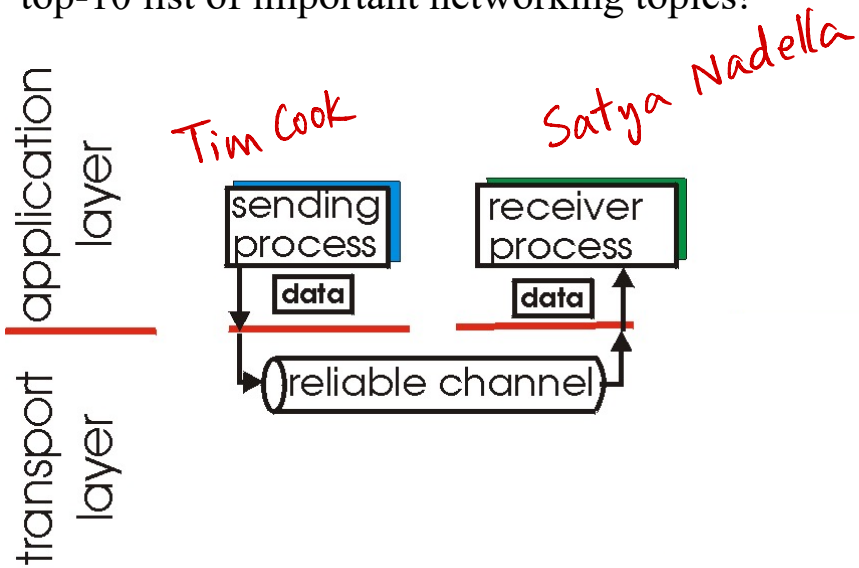
Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- services not available:
  - delay guarantees
  - bandwidth guarantees



Principles of Reliable data transfer

- important in app., transport, link layers
- top-10 list of important networking topics!

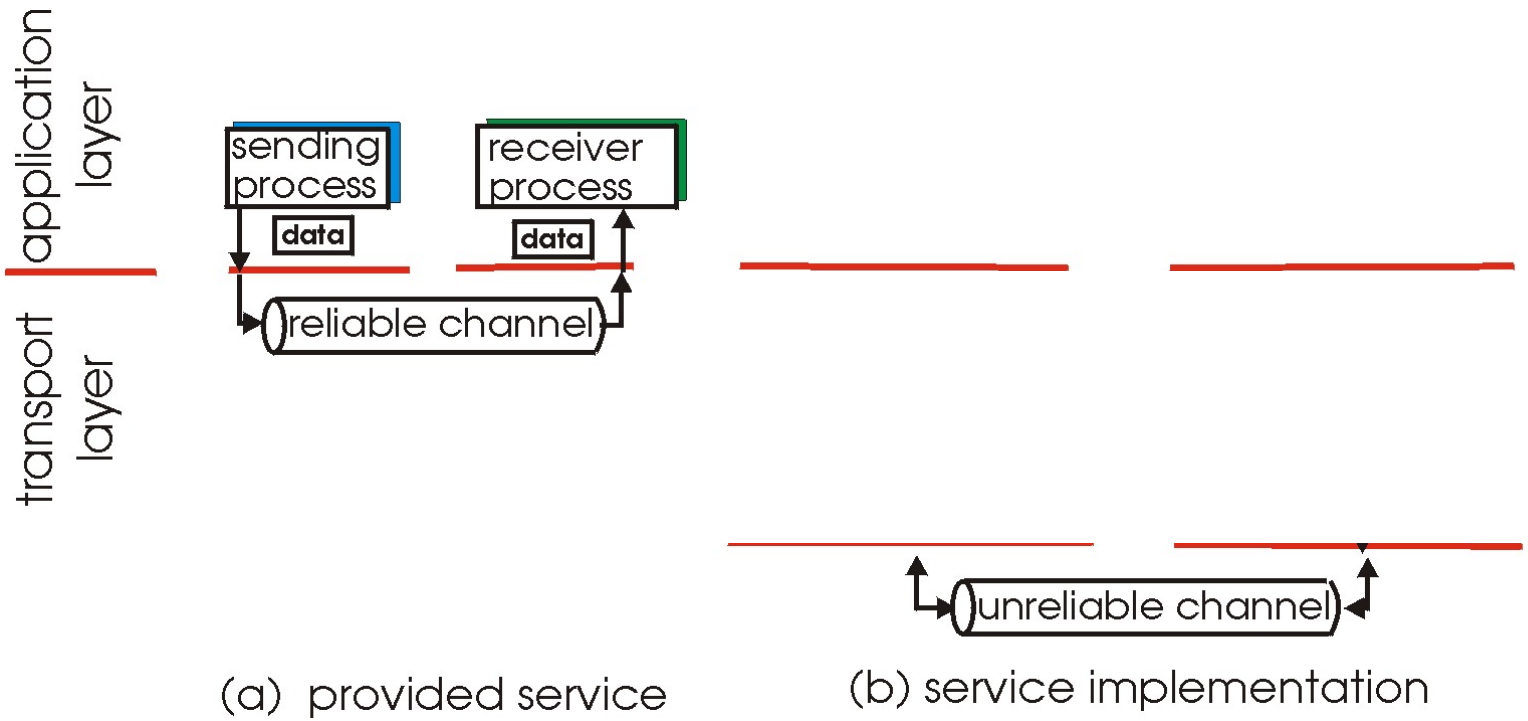


(a) provided service

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of Reliable data transfer

- important in app., transport, link layers
- top-10 list of important networking topics!



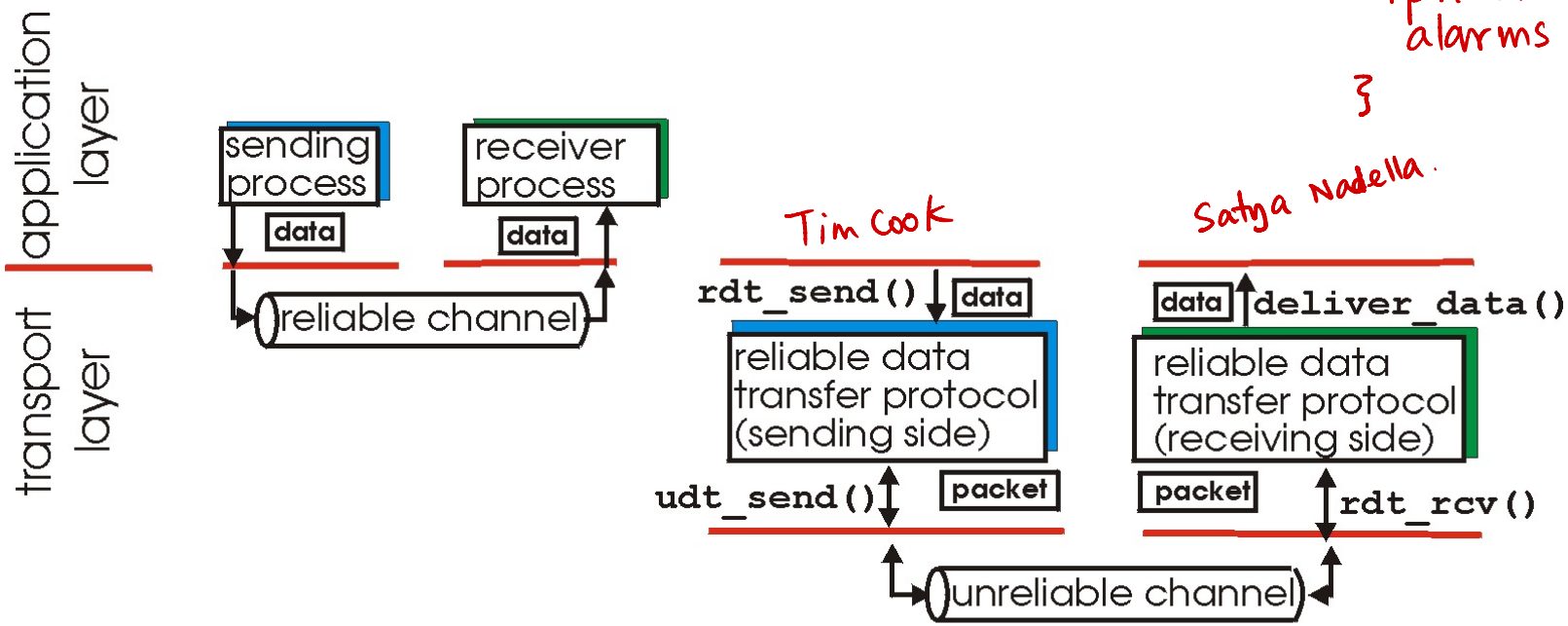
- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of Reliable data transfer

Tim cook
{
reliable_alarm_clock (4:30am)
}

Reliable_Alarm_clk(t)
{
Set up ∞ iphones with alarms (4:30am)
}

- important in app., transport, link layers
- top-10 list of important networking topics!



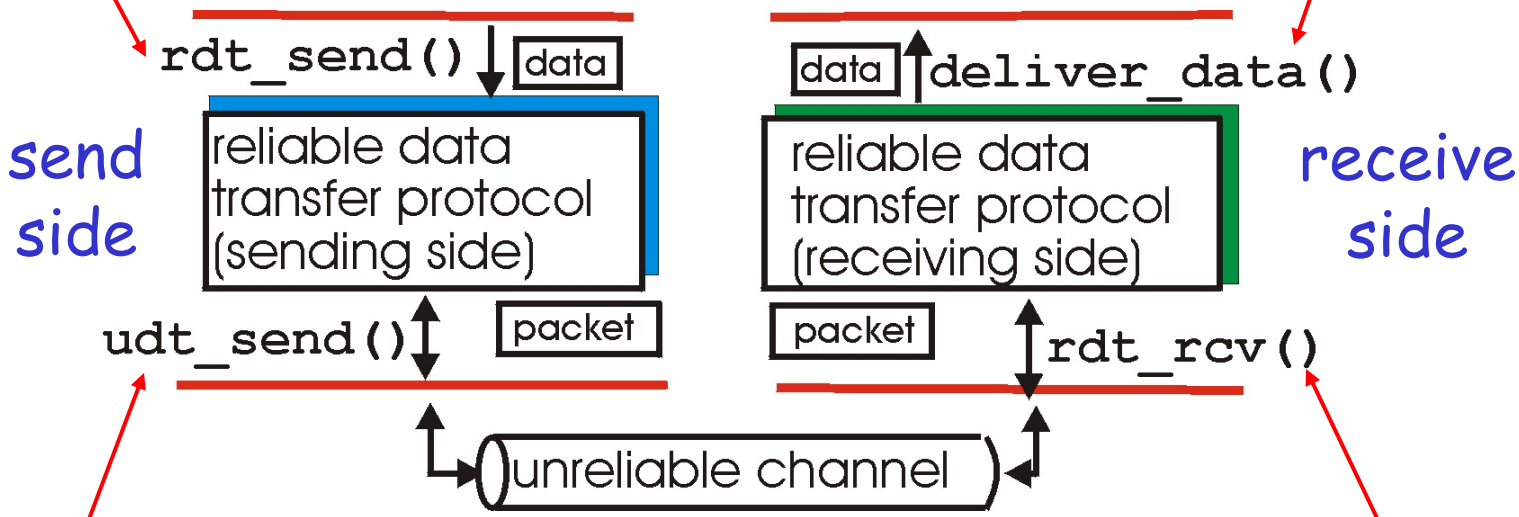
(a) provided service (b) service implementation

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Reliable data transfer: 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



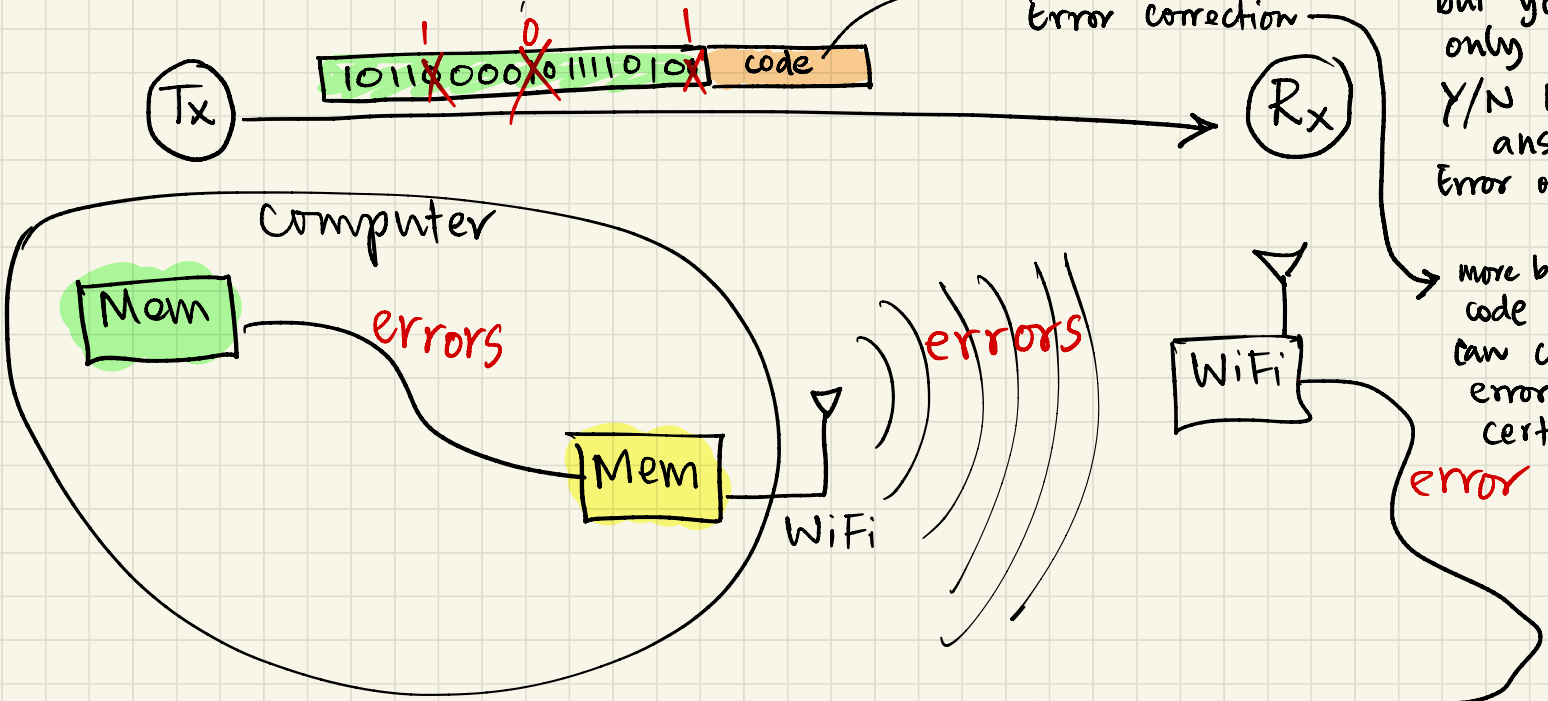
udt_send(): called by rdt, to transfer packet over unreliable channel to receiver

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

Error Detection / Correction

Error detection → less bits of code but you can only get a Y/N binary answer. Error or not.

Error correction → more bits of code but you can correct the error to a certain extent.



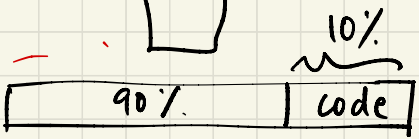
$$\text{code} = f(\text{data})$$

with high prob.

$$f(\text{data}' \neq \text{data}) \neq \text{code}$$

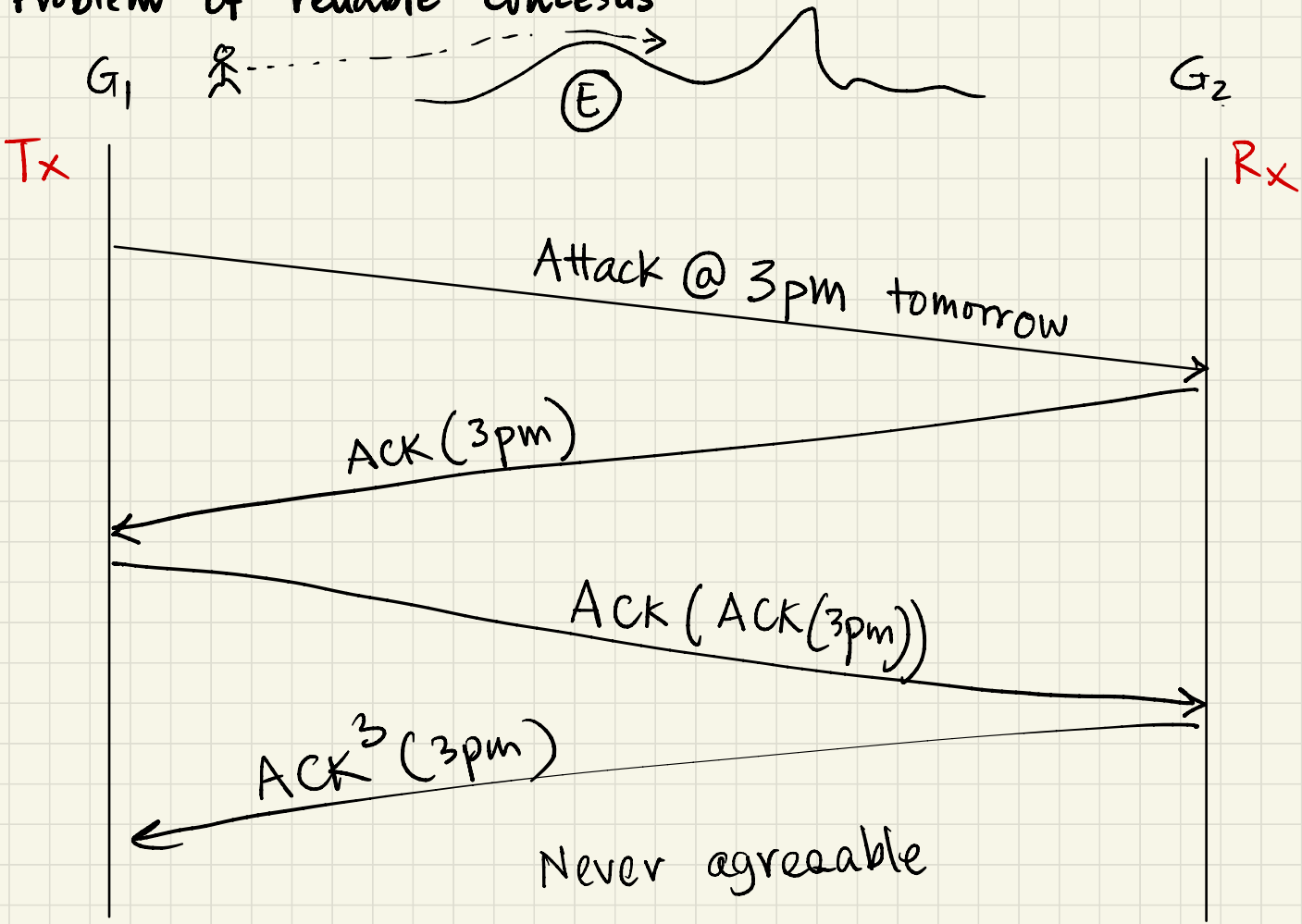
Rx

$$f(\text{data}') \sim \text{code}'$$

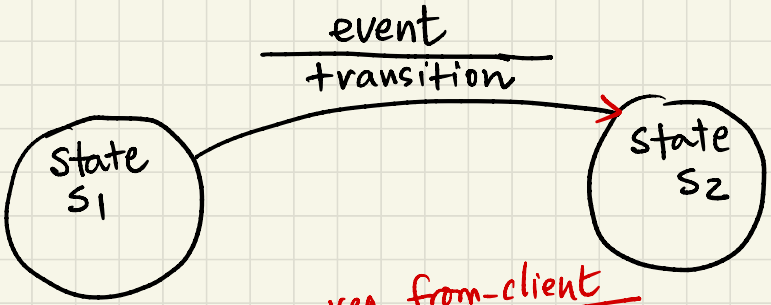


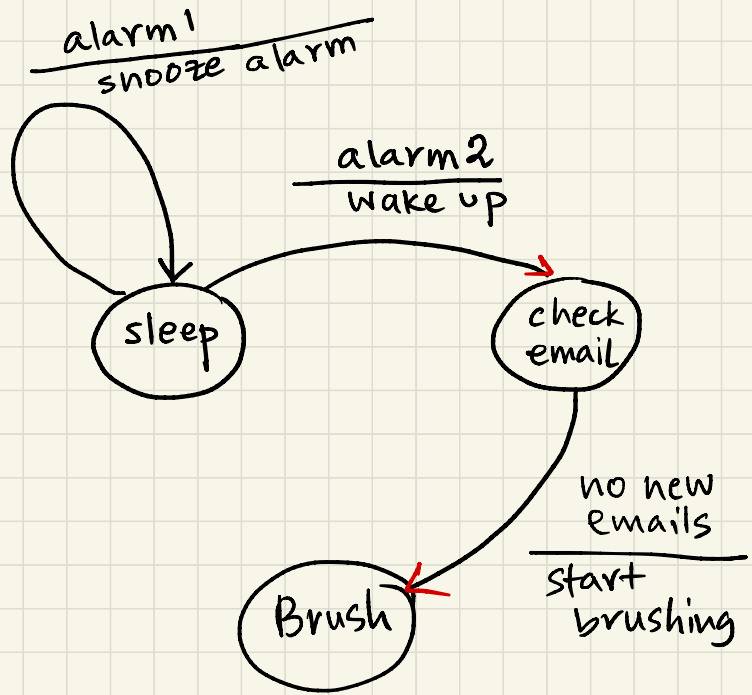
$$\text{HASH}(90\% \text{ of pkt}) \sim \text{code}(10\% \text{ of pkt})$$

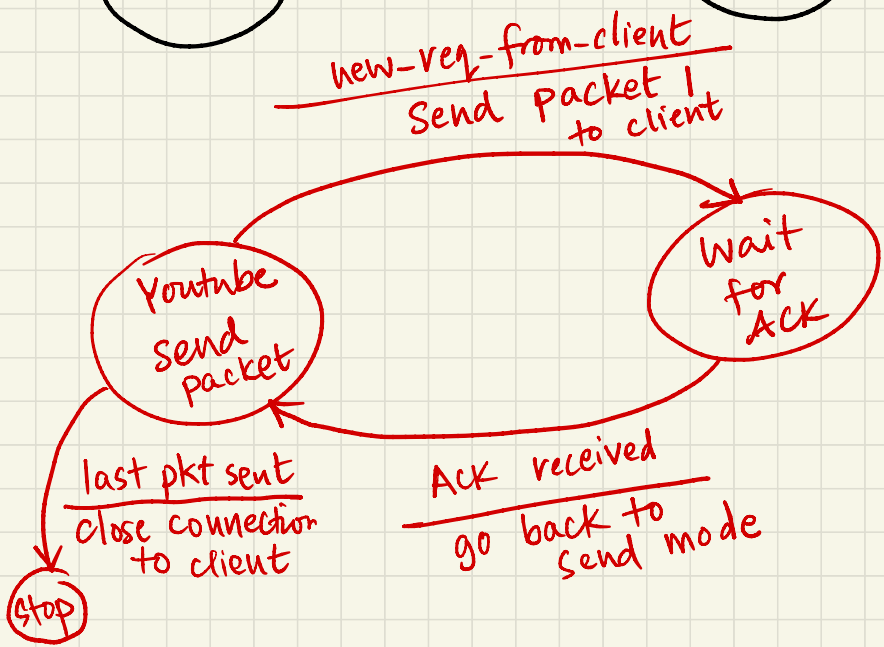
Problem of reliable concesus



State Machines → ways to formally capture the operation of a protocol.

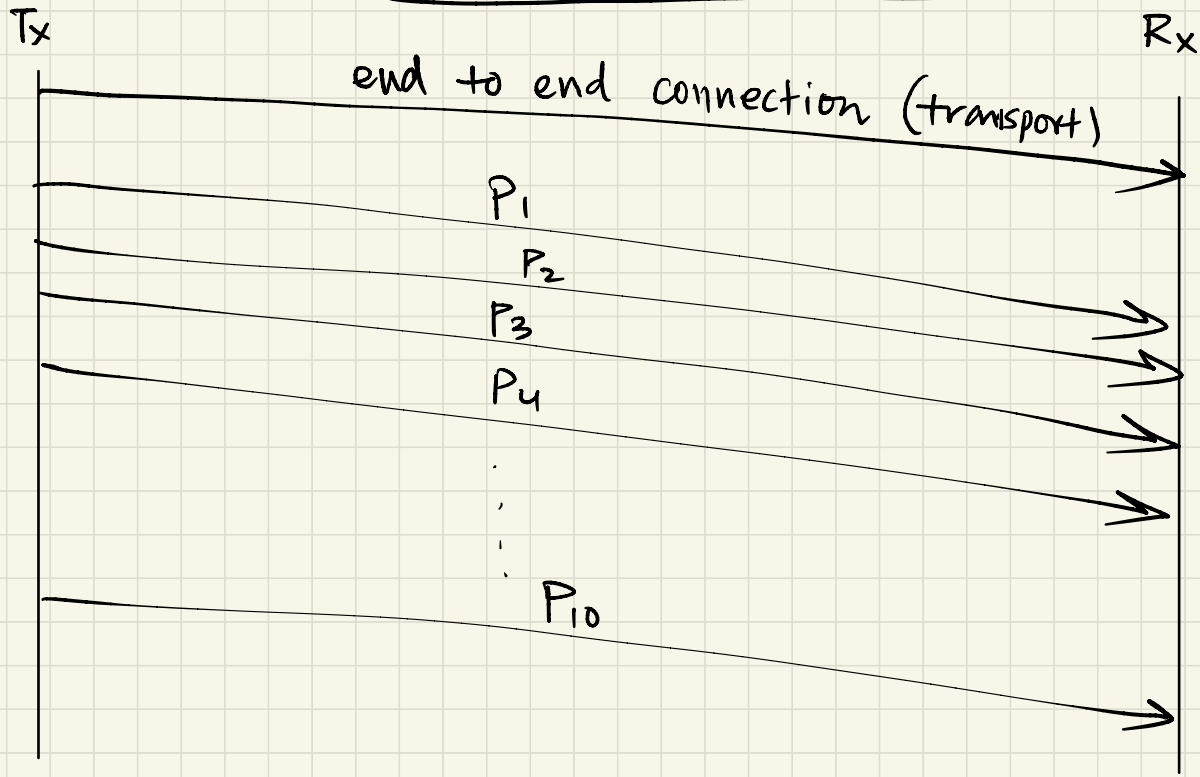






② Building Reliable Protocols from First Principles.

① Error model : completely error-free connection



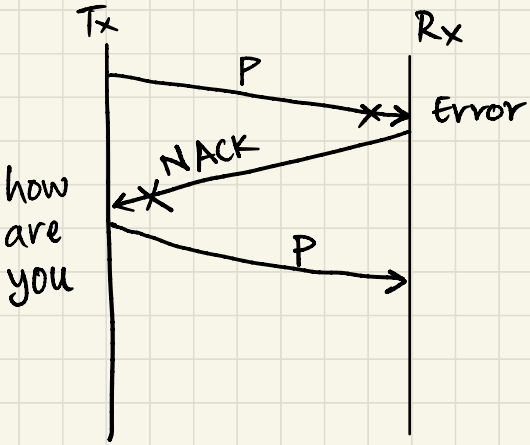
correctly deliver all 10 pkts.

② Error model : Bit error only

1010~~0~~1011100111~~1~~10001

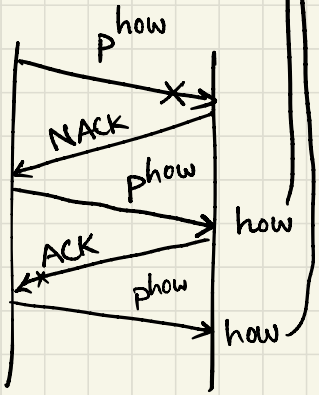
1 0

app

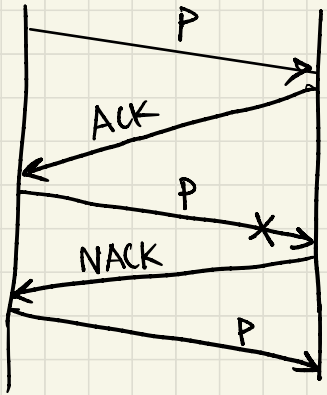


$f(\text{data}') \neq \text{code}$

$f(\text{NACK} \rightarrow \text{ACK}) = \text{code}$



Either case : Tx should resend packet.

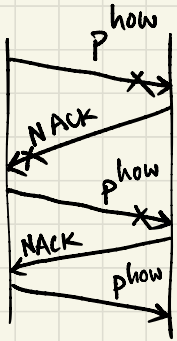


ACK correct } Send next pkt.

ACK corrupt

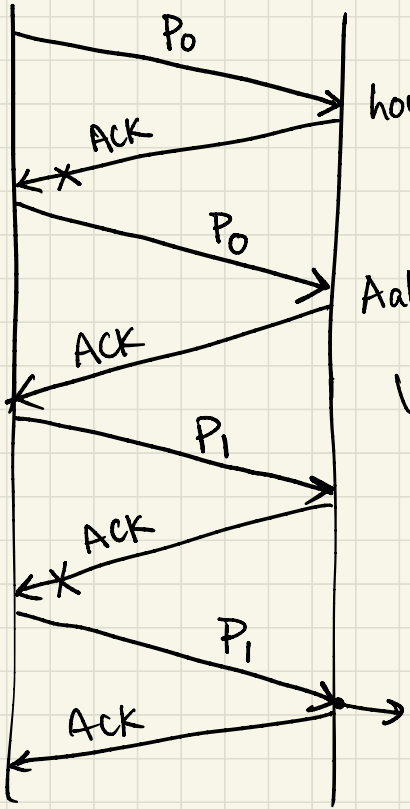
NACK corrupt } Resend pkt

NACK correct



Problem is duplicate packets.

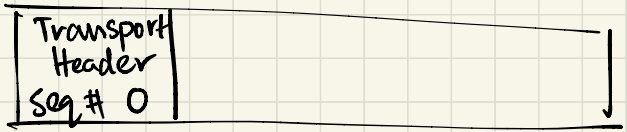
③ Transport Layer Sequence Numbers ... and how many bits?



how

Aah! same sequence number, hence drop the packet.

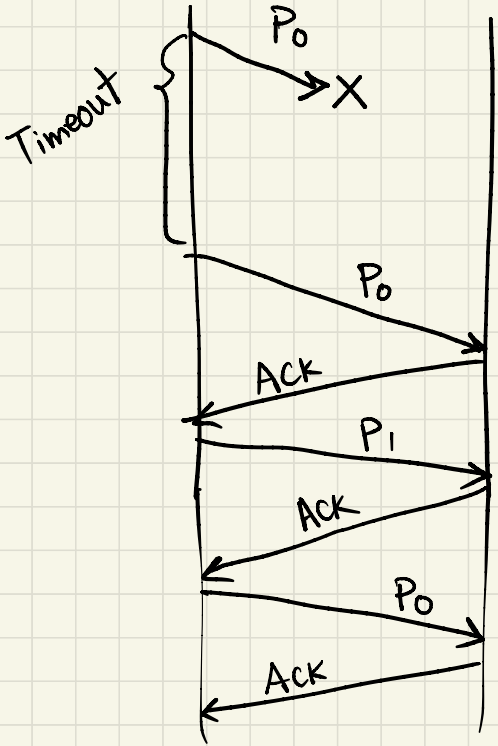
drop it because now I am expecting Packet P_0.

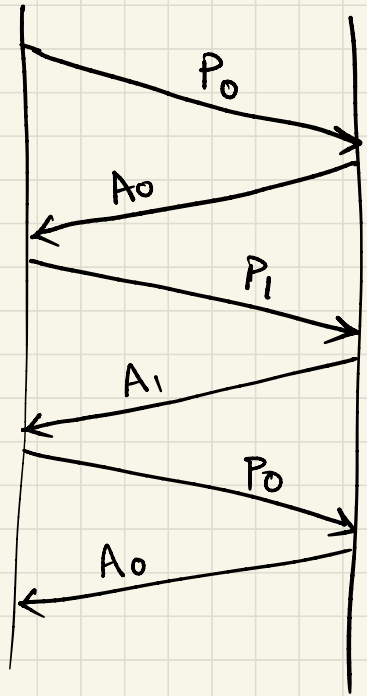


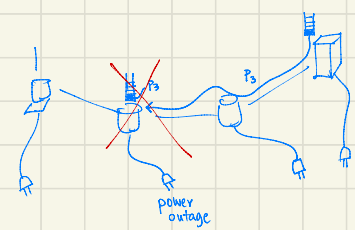
How many bits should be allocated for Seq. # in the header?

What about seq # for ACKs

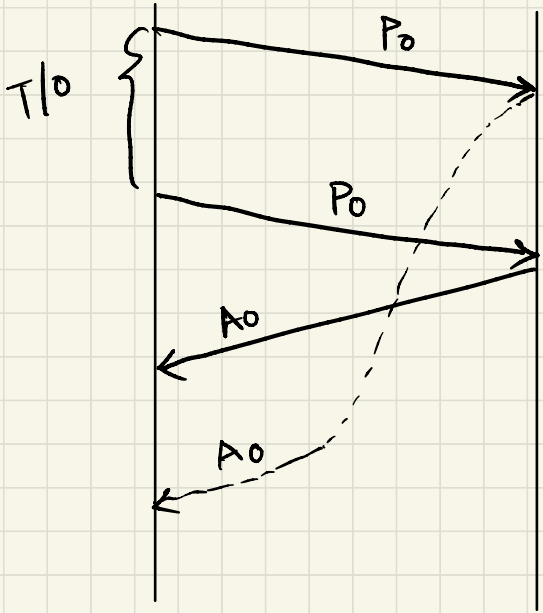
④ Error Model : Bit error + pkt loss (congestion and disasters)

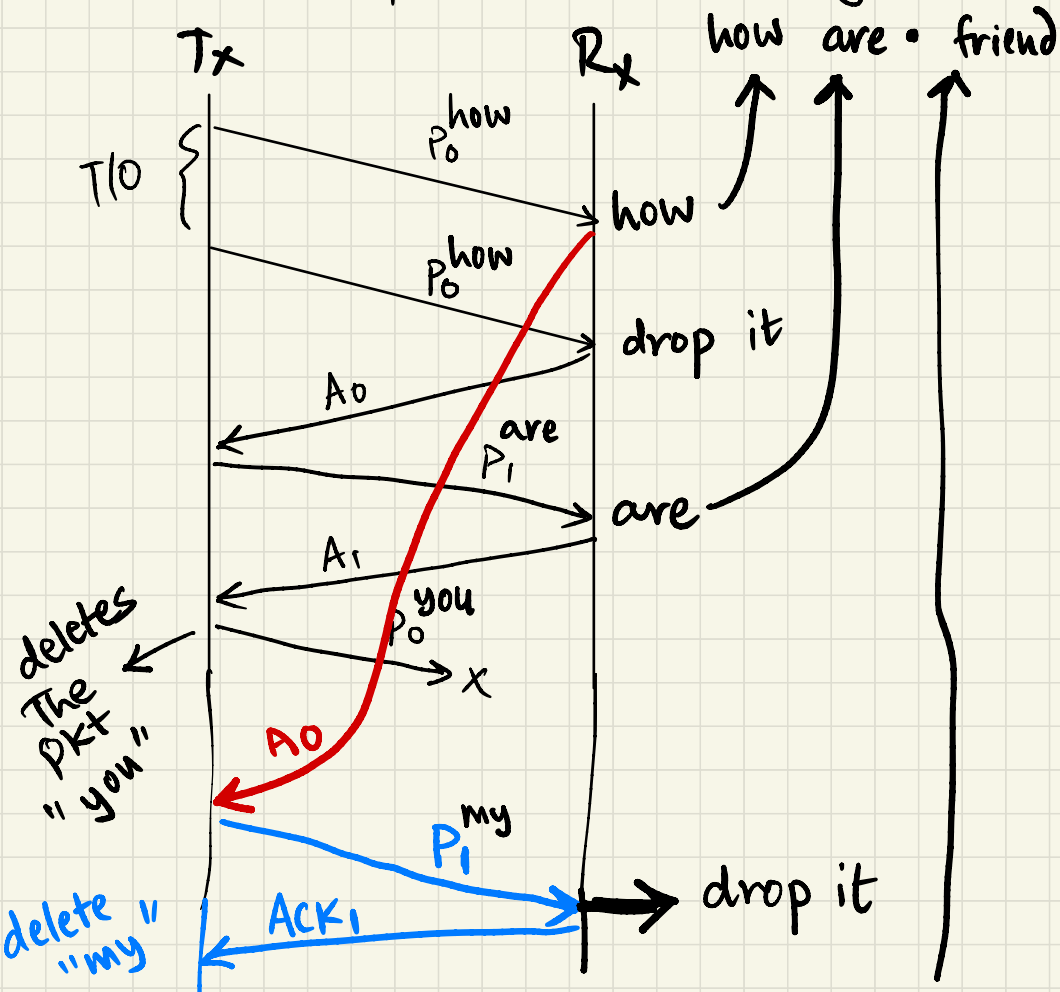


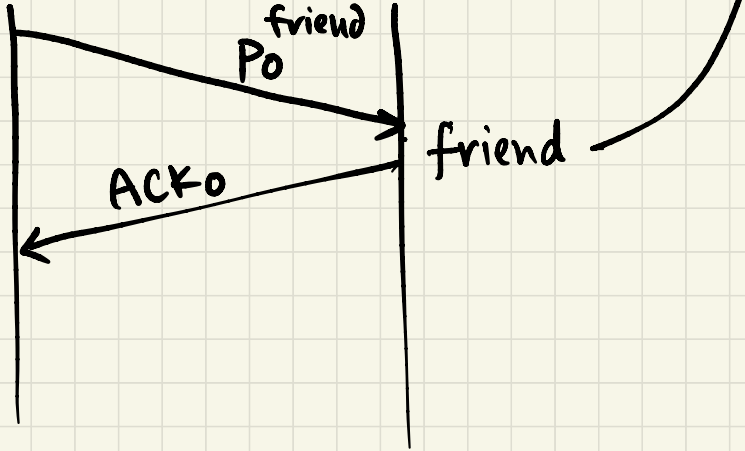




⑤ Error Model : Bit error + pkt loss + Delay






friend
Po
friend
ACKo