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!

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

(a) provided service
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

- 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)

(a) provided service
(b) service implementation

Transport Layer  3-9
Reliable data transfer: getting started

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

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

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

**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.

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

with high prob. \( f(\text{data} \neq \text{data}) \neq \text{code} \)

- \( R_x \)

Hash (90% of pkt) \( \Rightarrow \) code (10% of pkt)
Problem of reliable consensus

\[ G_1 \xrightarrow{\text{\textbullet}} G_2 \]

Tx \rightarrow Rx

Attack @ 3pm tomorrow

\[ \text{Ack}(3\text{pm}) \]

\[ \text{Ack}(\text{Ack}(3\text{pm})) \]

\[ \text{Ack}^3(3\text{pm}) \]

Never agreeable
State Machines → ways to formally capture the operation of a protocol.

```
- State S1
- State S2

event
transition

new-reg-from-client
  Send packet 1
to client

Wait for ACK

Alarm 1
  Snooze alarm

Alarm 2
  Wake up

sleep

Youtube
Send packet

Wait for ACK

Last pkt sent
  Close connection
to client

Ack received
  Go back to
send mode

Stop

Brush

Check email

No new emails
  Start brushing
```
Building Reliable Protocols from First Principles.

Error model: completely error-free connection

end to end connection (transport)

correctly deliver all 10 pkts.
Error model: Bit error only

\[ f(\text{data}) \neq \text{code} \]
\[ f(\text{NACK} \rightarrow \text{ACK}) = \text{code} \]

Either case: Tx should resend packet.

- **ACK correct**: Send next pkt.
- **ACK corrupt**: Resend pkt
- **NACK correct**: Resend pkt
- **NACK corrupt**: Resend pkt

Problem is duplicate packets.
Transport layer sequence numbers ... and how many bits?

Aah! Same sequence number, hence drop the packet.

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

drop it because now I am expecting Packet P0.

What about Seq. # for ACKs?
Error Model: Bit error + pkt loss (congestion and disasters)
Error Model: Bit error + pkt loss + Delay

\( \text{TK} \rightarrow \text{yo} \rightarrow \text{Poh} \)

how

\( \text{show} / \text{you} \),

drop

\( \text{ave} \)

\( \leftarrow \)

are

\( \div \).

are

pit

good

delites

"you"

\( \leftarrow \)

how

\( \text{my} \)

\( \leftarrow \)

drop it

\( \leftarrow \)

Are

\( \leftarrow \)

how are friend

\( \rightarrow \)

drop it

\( \rightarrow \)

how