Transport Services and Protocols

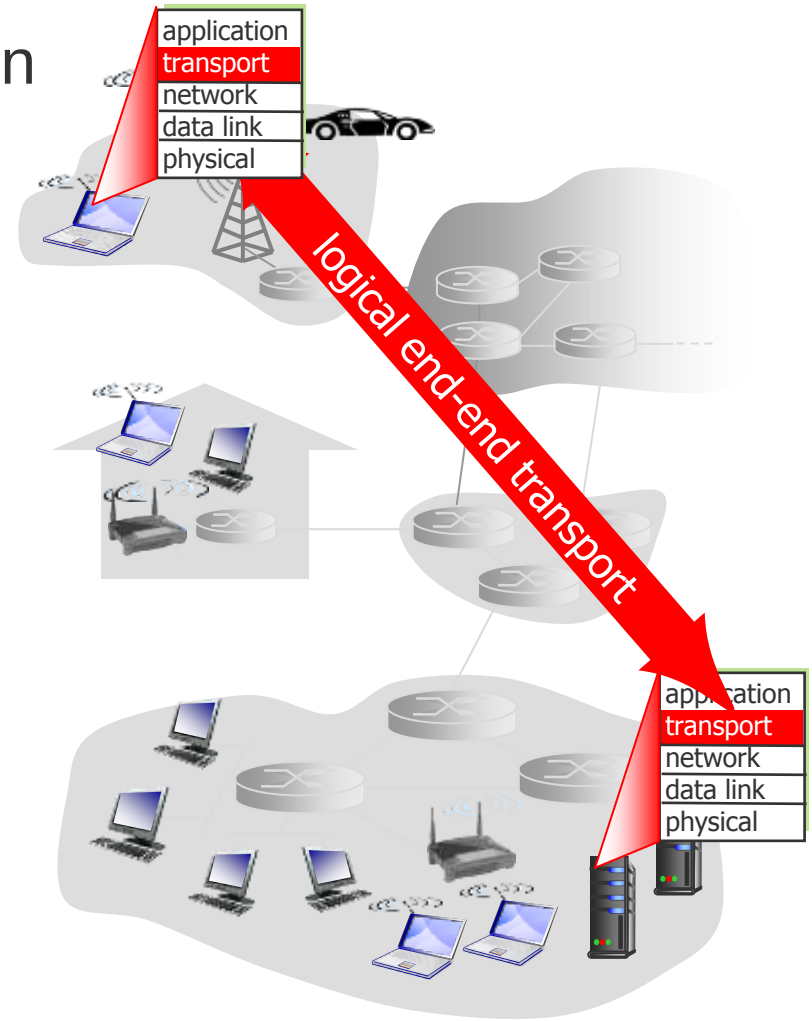
Andy Carpenter

(Andy.Carpenter@manchester.ac.uk)

Elements these slides come from Kurose and Ross, authors of "Computer Networking: A Top-down Approach", and are copyright Kurose and Ross

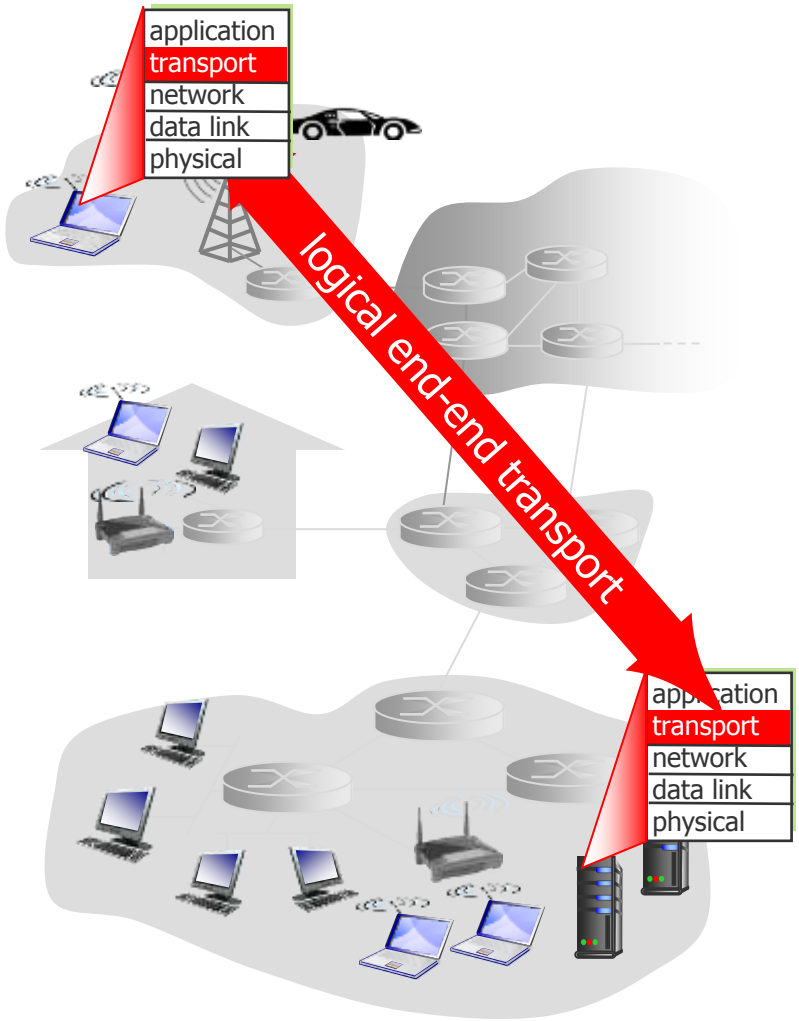
Transport Services and Protocols

- Provides logical communication between applications
  - process-to-process
- Runs in end systems
  - send side: breaks app messages into segments, passes to network layer
  - receiver side: reassembles segments into messages, passes to app layer
- Multiplex/ id end-points
- Protocols available:
  - Internet: TCP, UDP



Transport Service QoS Params

- Provides QoS to applications
- Options
  - Connectionless/connection-orientated
  - reliability
  - flow control
  - congestion control
- Alters underlying network QoS
- Services not available:
  - delay guarantees
  - bandwidth guarantees



End-points (Ports/Sockets)

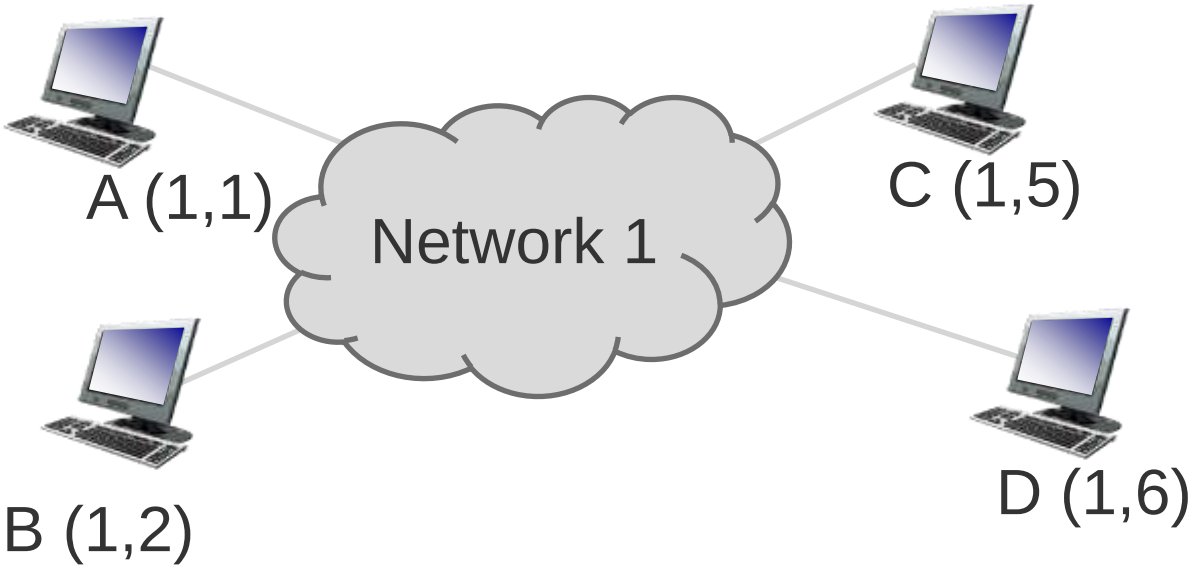
- Support multiple destinations
- Identify process via 16-bit identifier; port
- Applications use sockets which have associated port
- Well know services allocated ports 0-1024

Port	Mnemonic	Service
7	ECHO	Echo
20	FTP-DATA	FTP (default data)
21	FTP	FTP (control)
53	DOMAIN	Domain name service
80	HTTP	Hypertext Transfer protocol

Centrally allocated

- Client ports are dynamically allocated by o/s

End-Point: Connecting to a Service

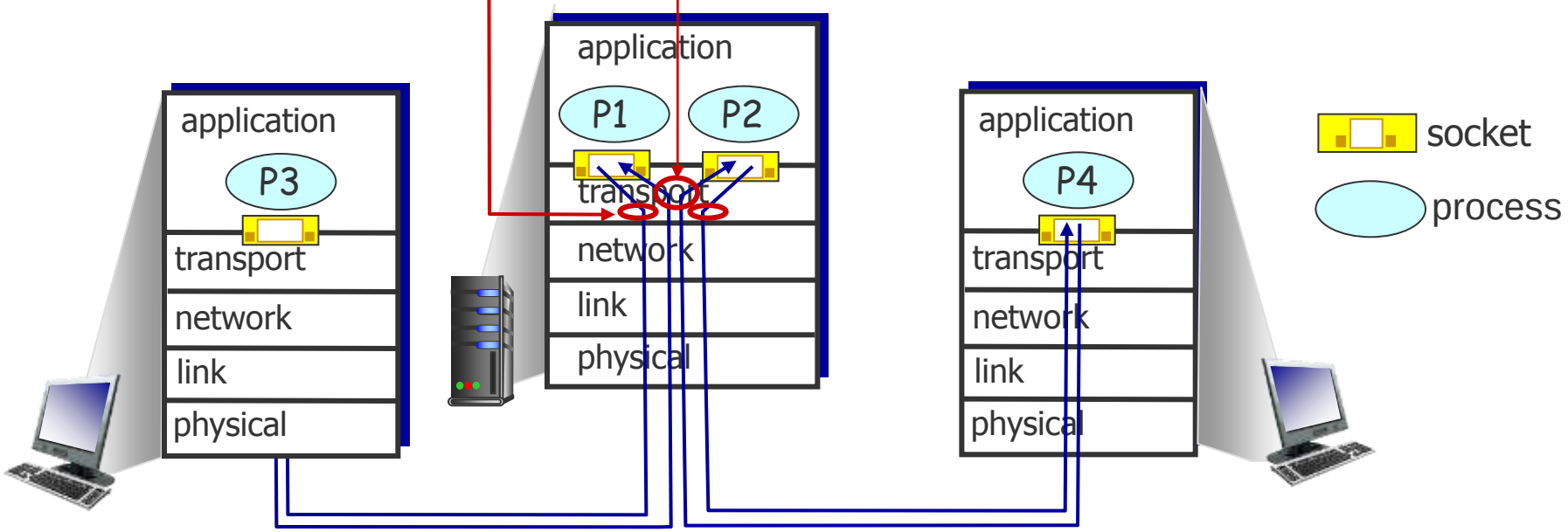


- Packets are sent to ports with attached applications
- Data to other ports will generate an error message

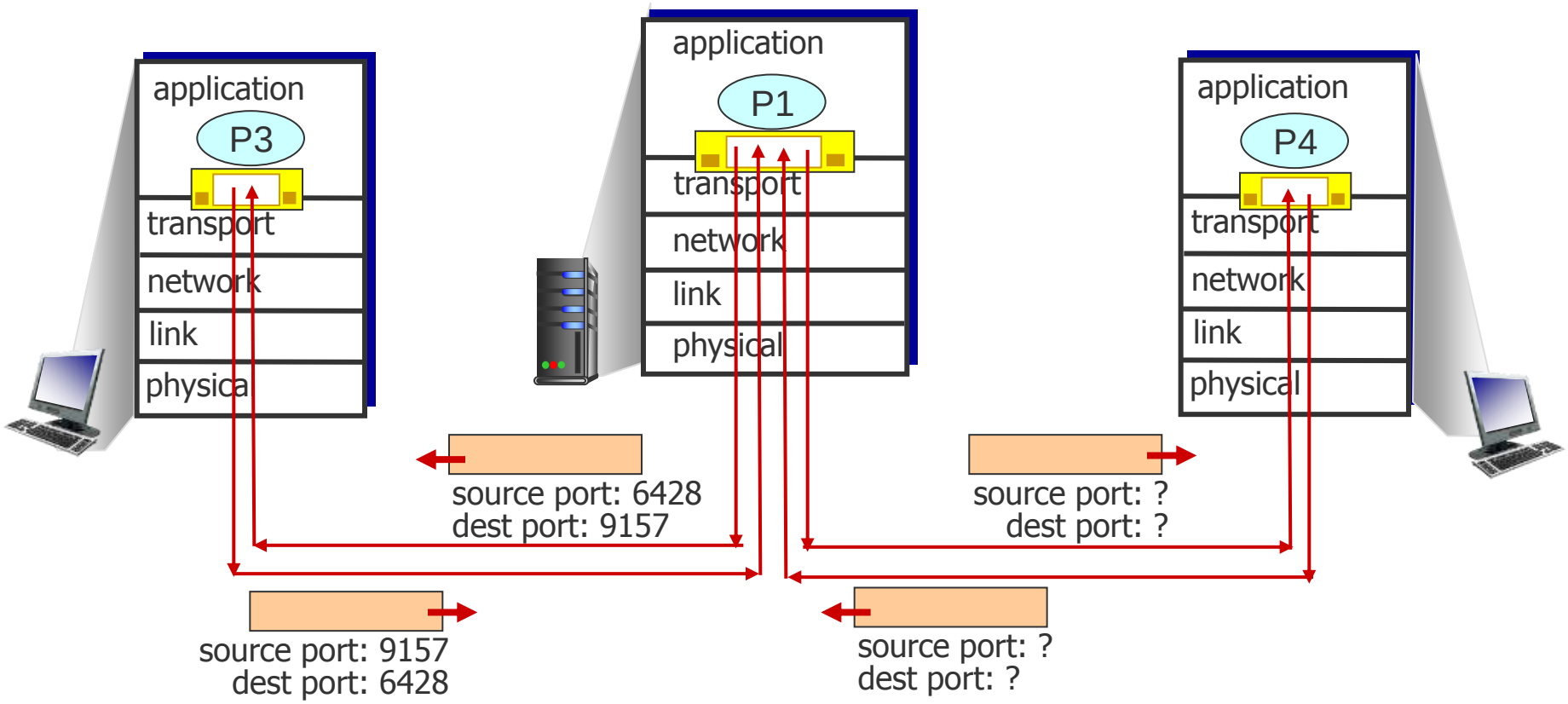
End-Point: Multiplexing

multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

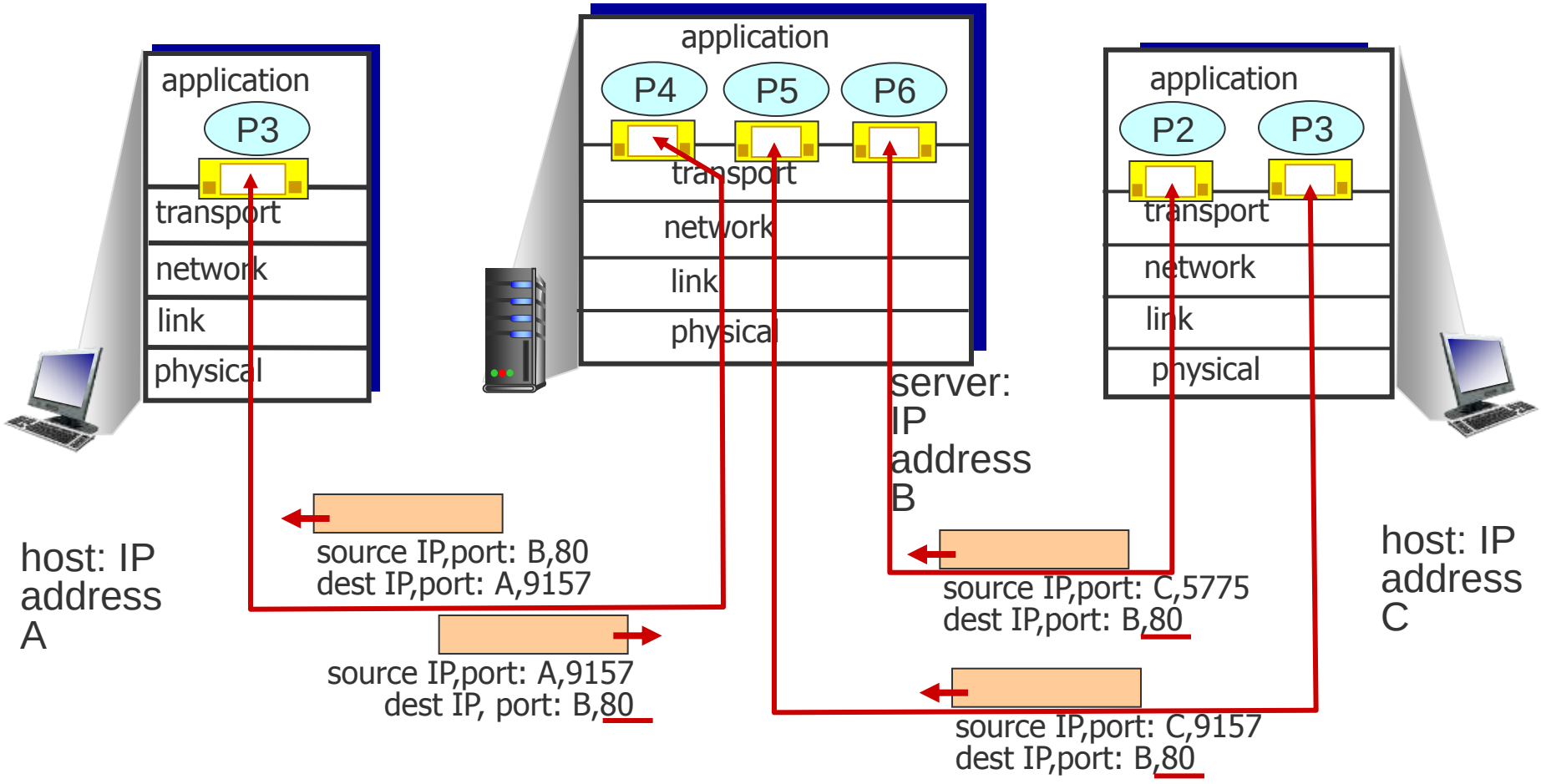
demultiplexing at receiver:
use header info to deliver received segments to correct socket



End-Point: Id (Connectionless)



End-Point: Id (Connection-Orient)



User Datagram Protocol (UDP)

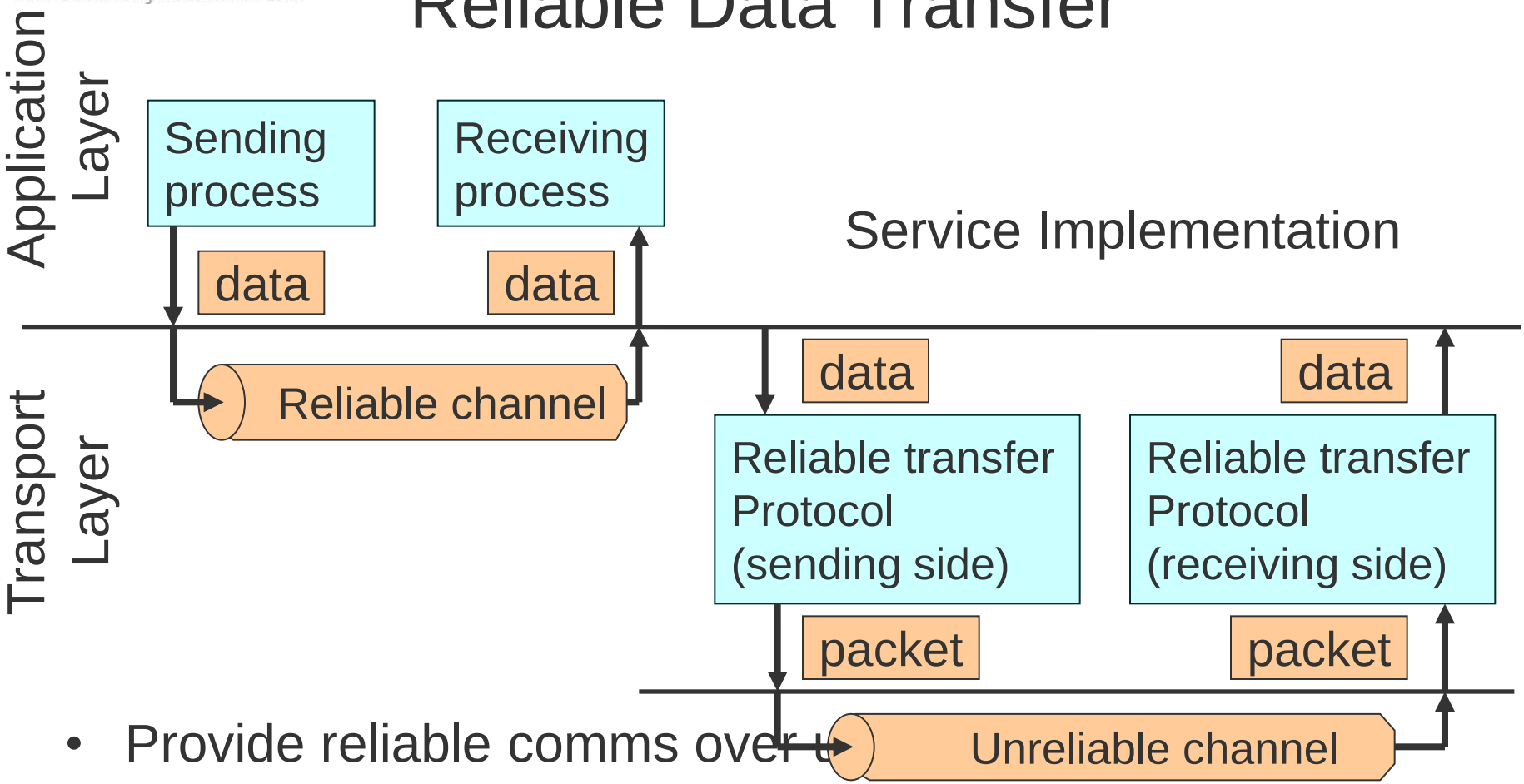
- "no frills", "best effort" connectionless service, segments may be:
  - lost, reordered
- Segments independent
- Often used for streaming multimedia applications
  - loss tolerant
  - rate sensitive
- Other uses:
  - DNS, SNMP
  - NFS
  - o/s applications

Why is there a UDP?

- no connection setup (adds delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

[RFC 768]

Reliable Data Transfer

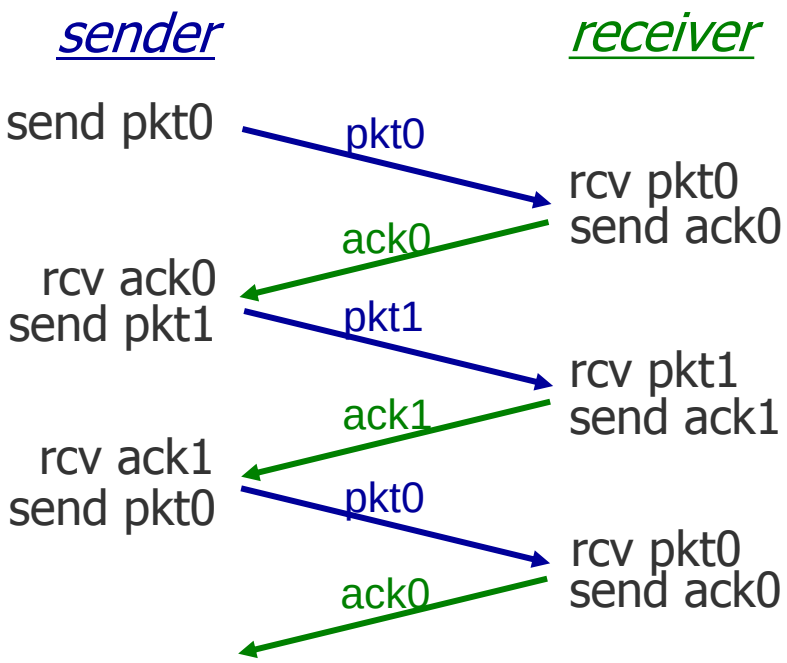


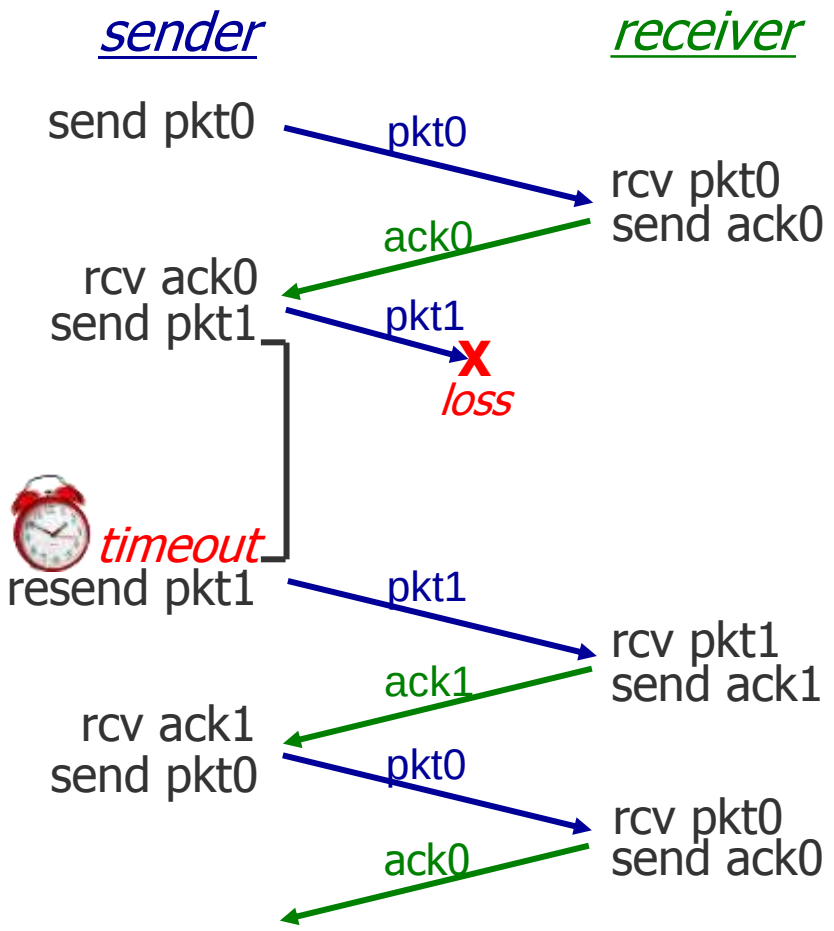
- Provide reliable comms over u
- Complexity depends on characteristics of unreliable channel

Recovering from Errors

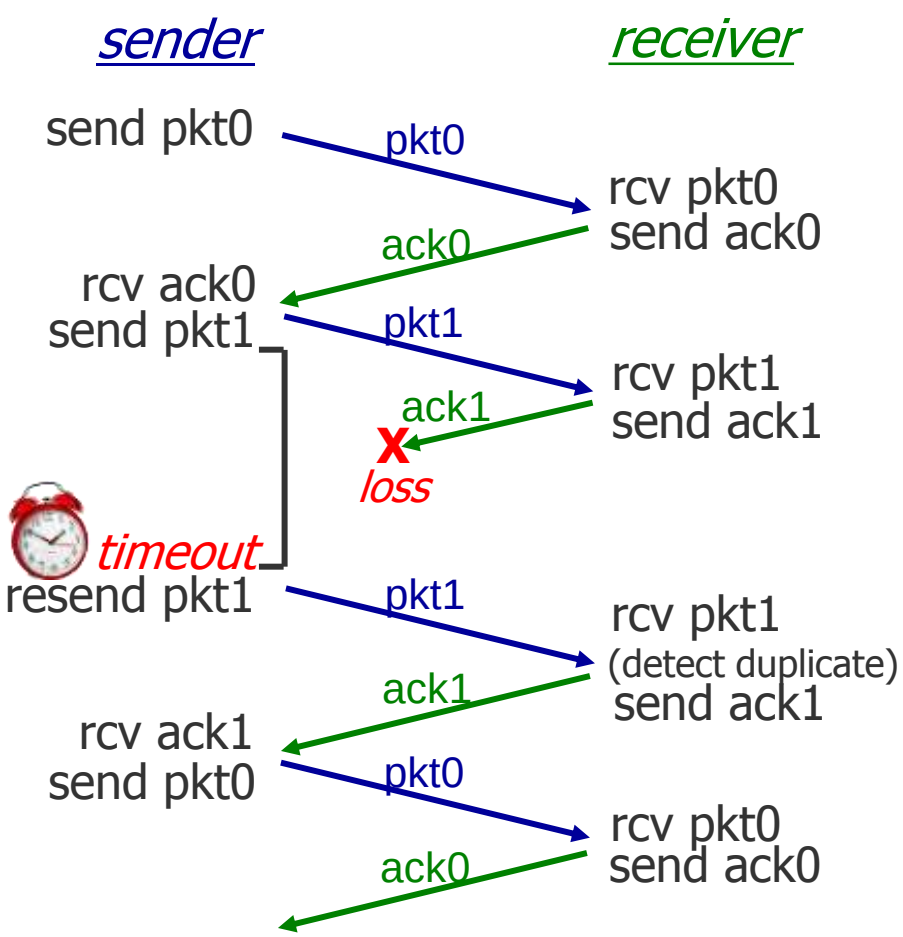
- Recovery uses two mechanisms:
  - acknowledgements and timeouts
- Acknowledgement (**ACK**) is control packet from
  - receiver to transmitter of data packet being ACKed
- Receipt of ACK confirms delivery of data
- If ACK not receiver within **timeout**:
  - transmitter of data retransmits data; needs copy
- Process called automated repeat request (ARQ)
- ARQ mechanisms: **stop-and-wait**, **sliding window**

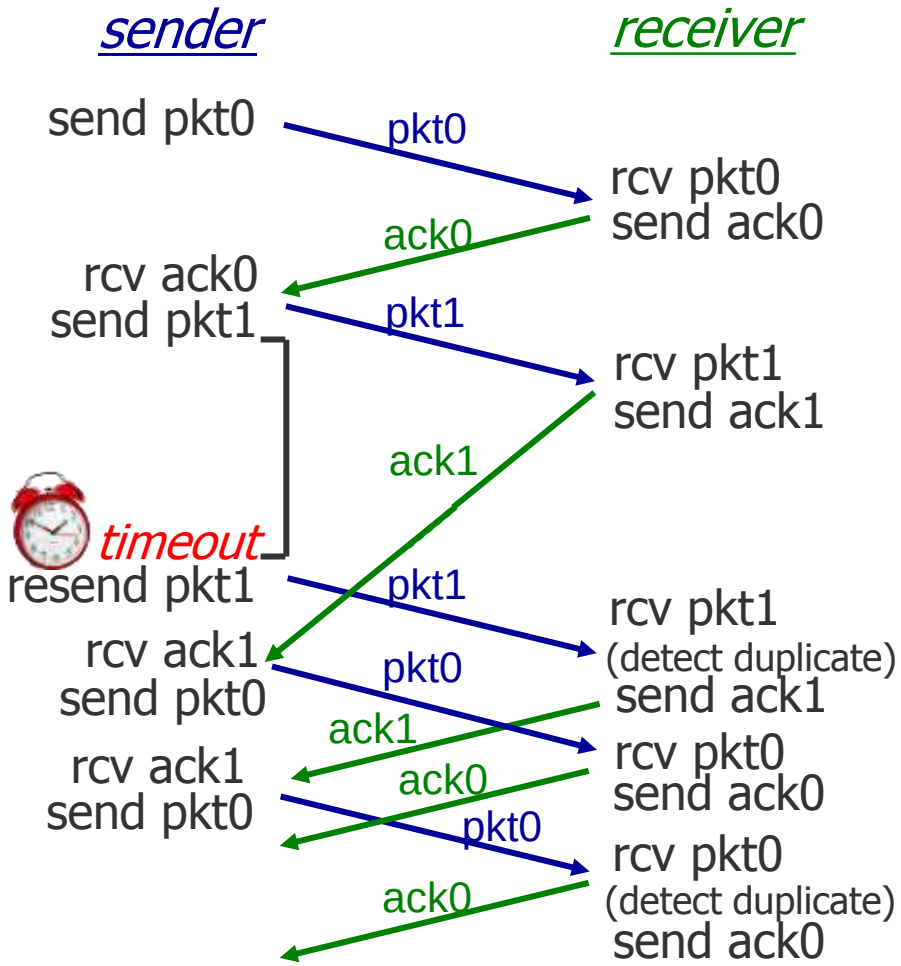
Reliability: Stop-and-Wait



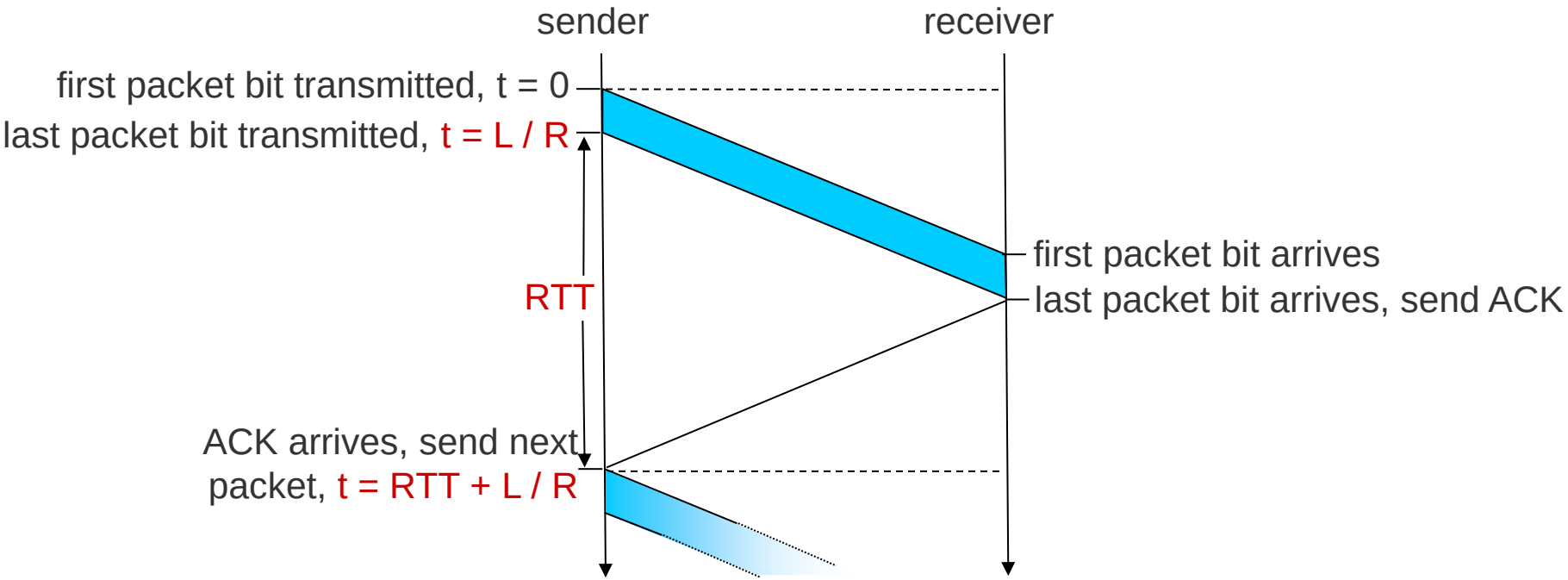


Reliability: Stop-and-Wait





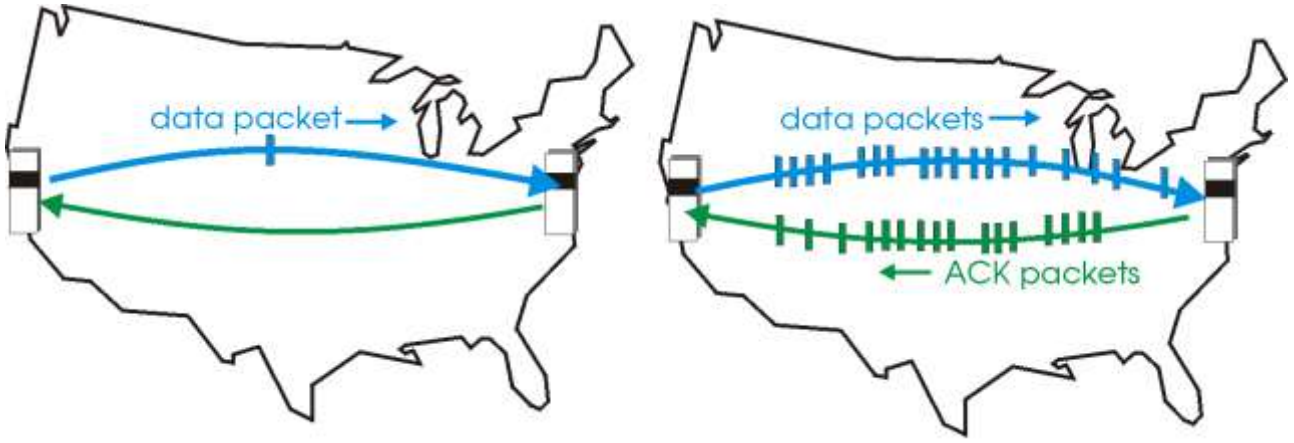
Reliability: S-a-Wait - Utilisation



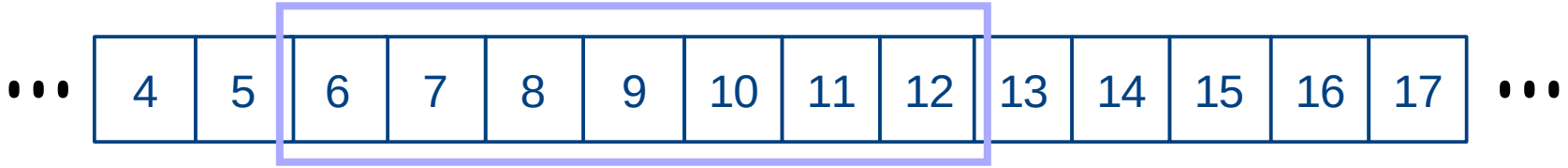
$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Network protocols limit use of physical resources

Reliability: Pipelined Protocols

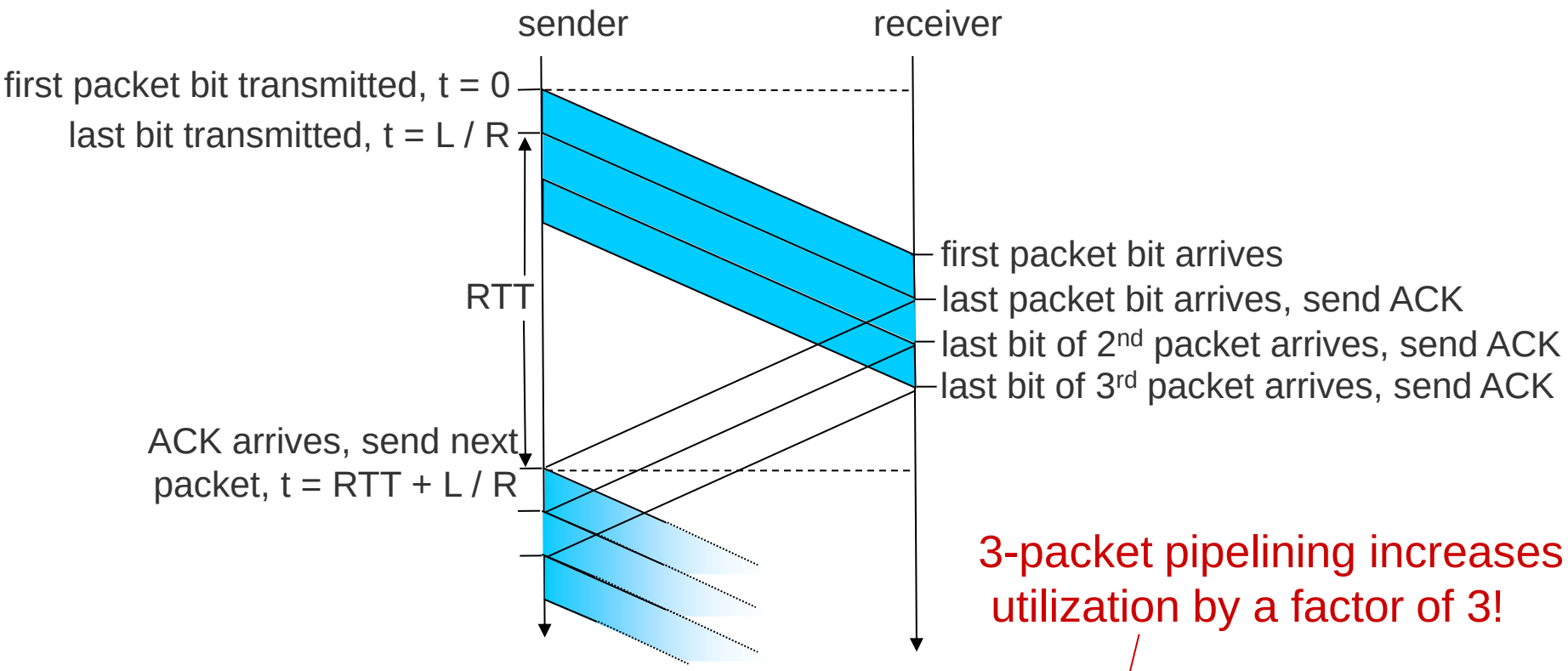


- Allow multiple, “in-flight”, yet-to-be-acknowledged pkts
- Example: received ACK 5, window 6



- Often referred to as sliding window

Reliability: Pipelined Utilisation



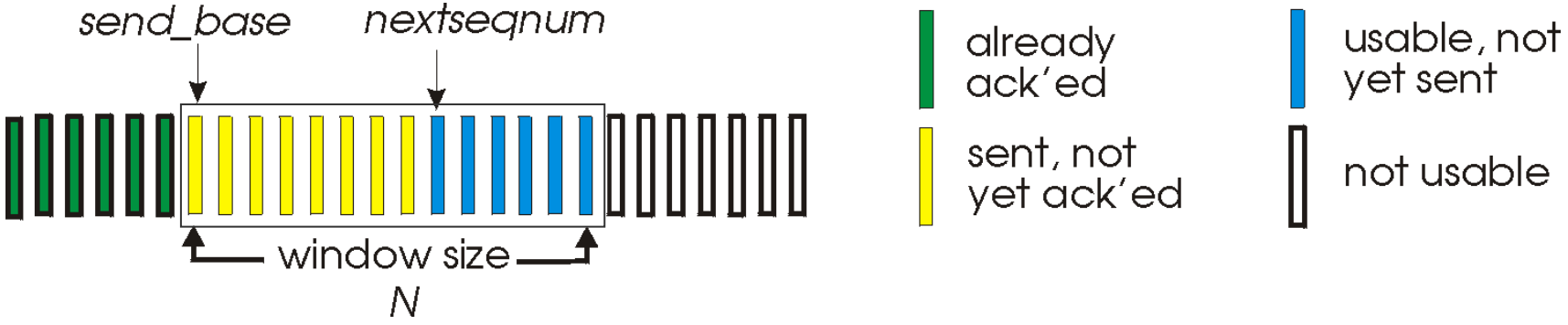
3-packet pipelining increases utilization by a factor of 3!

$$U_{sender} = \frac{3L / R}{RTT + L / R} = \frac{.0024}{30.008} = 0.00081$$

Reliability: Pipelined Approaches

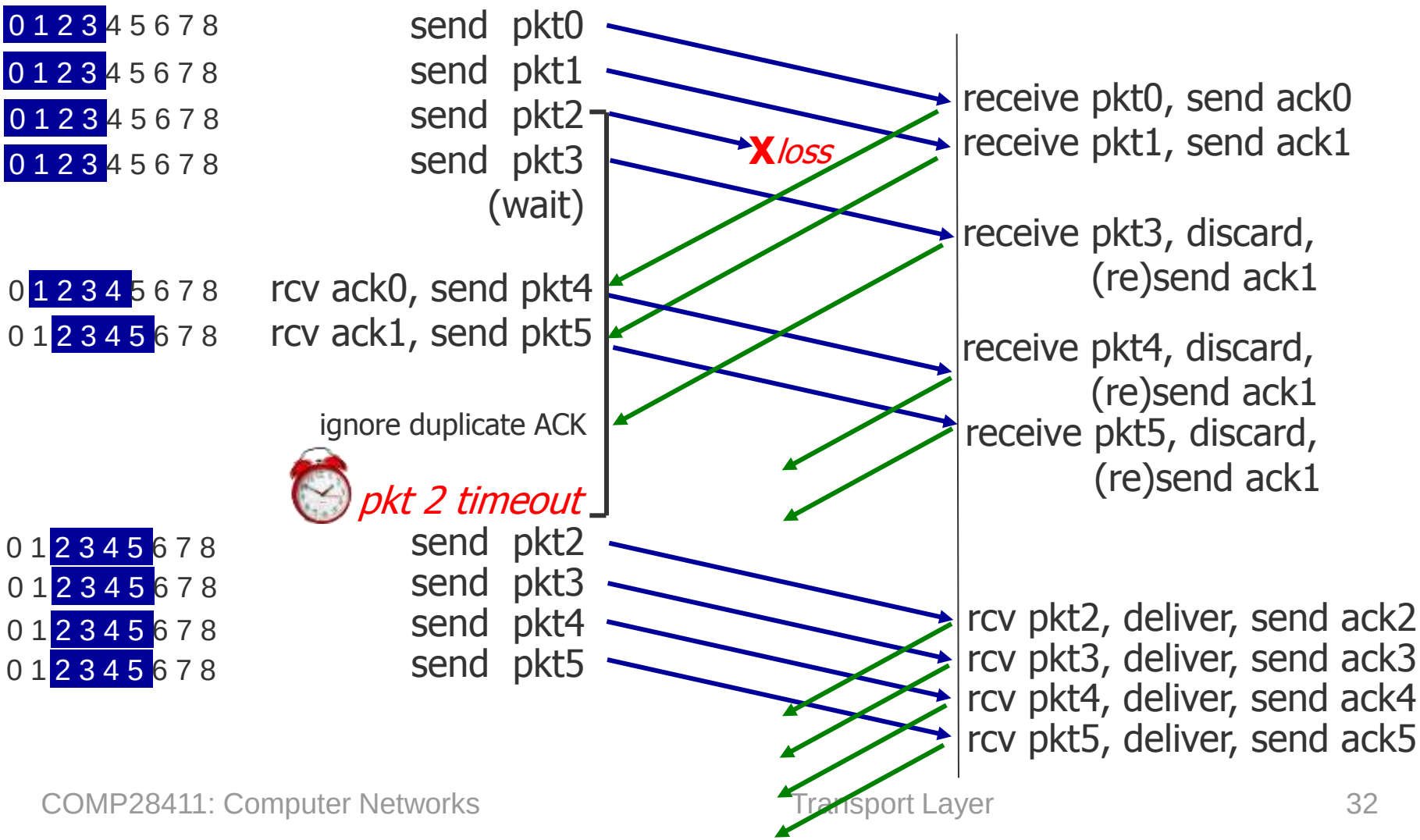
- What if expecting packet 8 and get packet 9?
- If packet 8 delayed, when arrives can send ACK(9)
- If lost, timeout will expire and will be resent
- **Go-Back-N**, send cumulative ACKs:
  - ACK(n) acknowledges all packets upto n
  - likely that will also get packets 9, 10, … resent
- **Selective repeat**:
  - explicitly acknowledges all packet
  - only unsuccessfully received packets are resent
- Could **negatively acknowledge** (NACK) packet 8,
  - requests retransmission without timeout expiring

Reliability: Go-Back-N

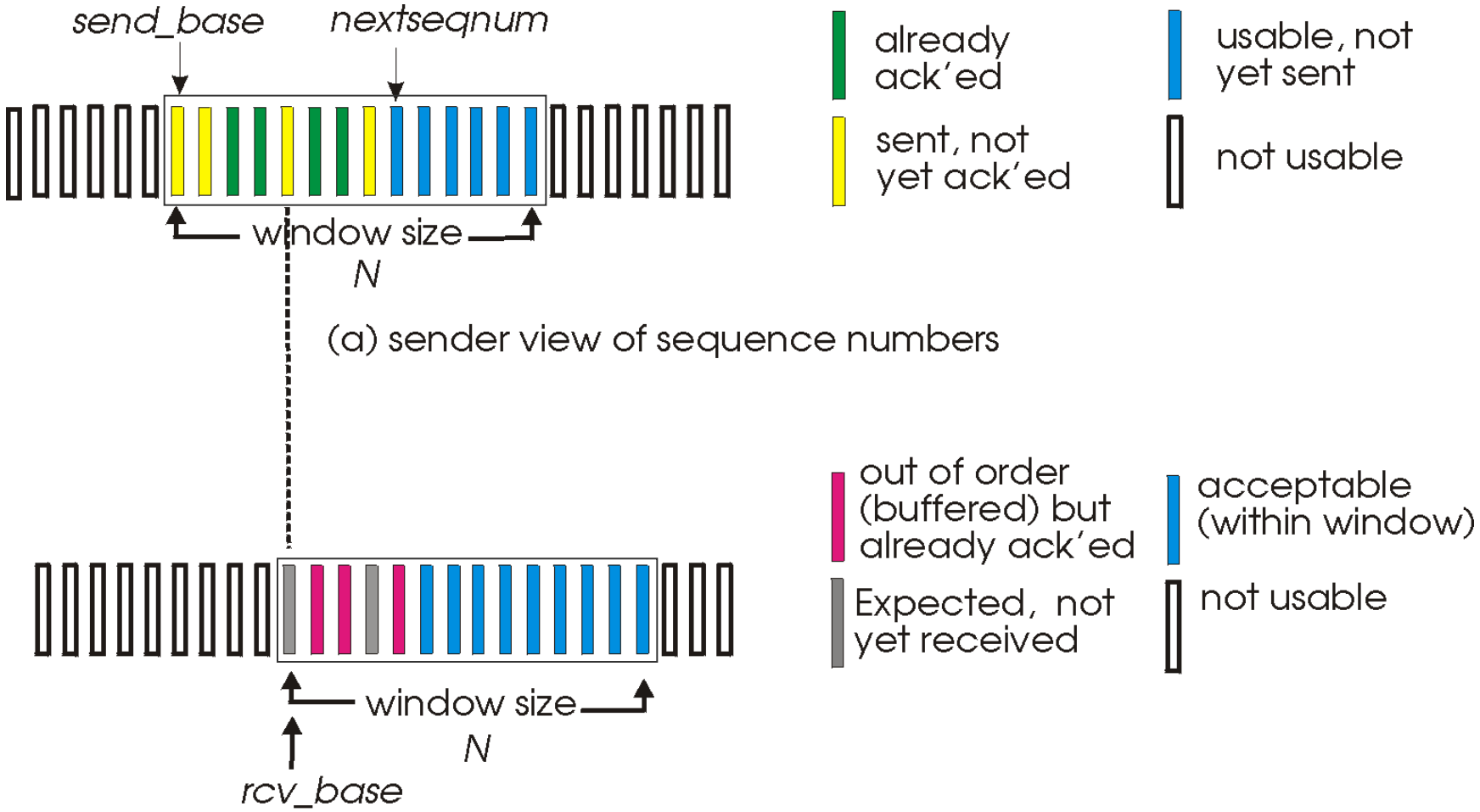


- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed
- ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"
- may receive duplicate ACKs (see receiver)
- timer for oldest in-flight pkt
- timeout(n): retransmit packet n and all higher seq # pkts in window

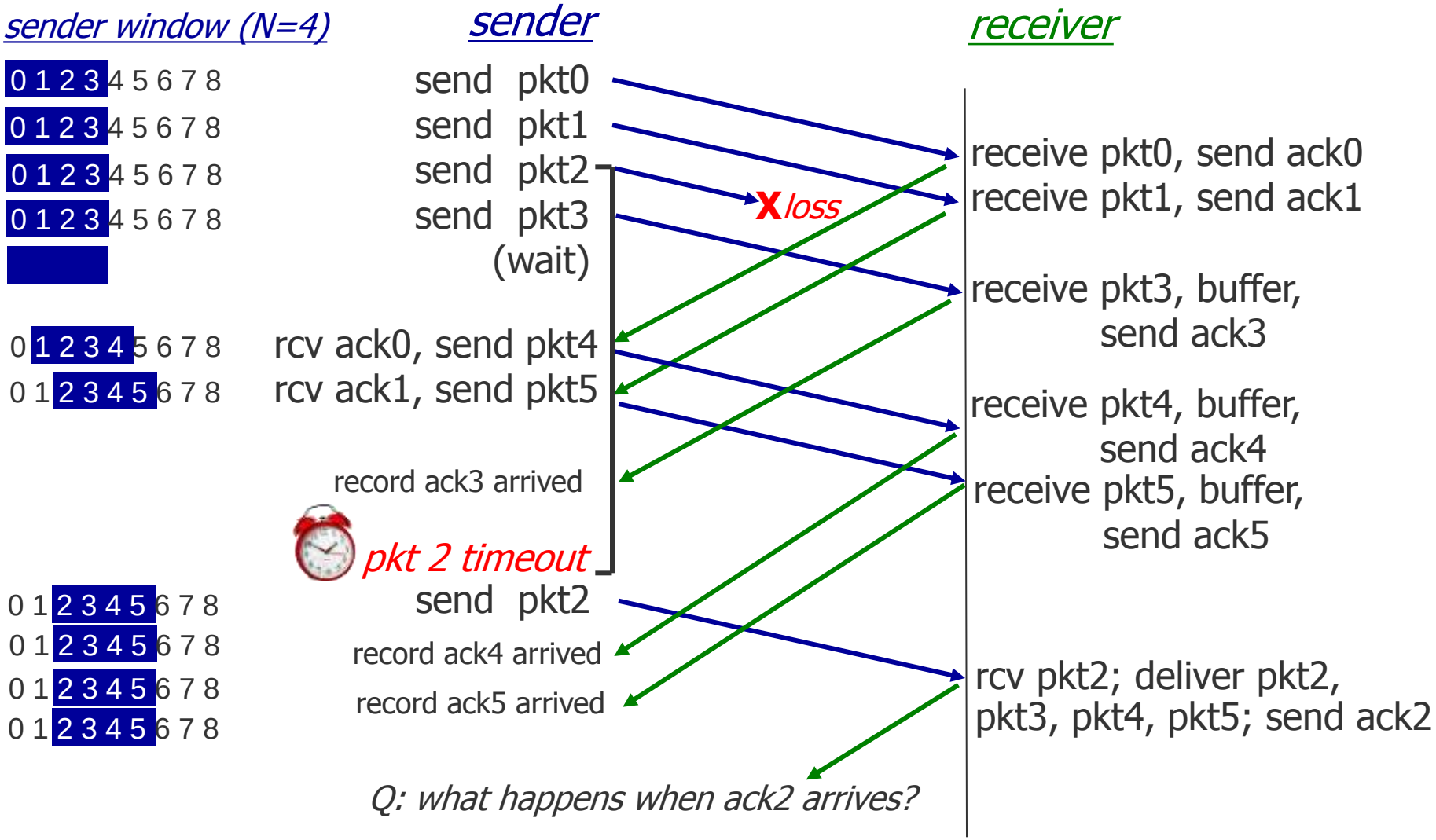
Reliability: Go-Back-N Example (H



Reliability: Selective Repeat



Reliability: Selective Repeat



Reliability: Window Issue