Week3: Transport Layer Pt1

NOS_Lecture_slides_WK03.pdf

Transport Layer Overview

- **Goals:** Understand principles behind transport layer services and learn about Internet transport layer protocols (UDP and TCP).
- **Topics Covered:**
  - Transport-layer services
  - Multiplexing and demultiplexing
  - Connectionless transport (UDP)
  - Principles of reliable data transfer
  - Connection-oriented transport (TCP)
  - Principles of congestion control
  - TCP congestion control
  - Evolution of transport-layer functionality

Key Concepts

- Process-to-Process Communication
- Multiplexing and Demultiplexing
- Connection-oriented vs Connectionless Services
- Flow Control and Error Control

Main Protocols

- **TCP (Transmission Control Protocol)**- Reliable, connection-oriented protocol- Provides flow control and congestion control- Used for applications requiring

guaranteed delivery

- **UDP (User Datagram Protocol)**- Unreliable, connectionless protocol- Lightweight, no overhead for connection setup- Suitable for real-time applications

Transport Layer Services

1. Addressing using ports
2. Segmentation and reassembly
3. Connection establishment (for TCP)
4. Flow control mechanisms
5. Error detection and handling

Transport Services and Protocols

- **Functions:** Provide logical communication between application processes on different hosts.
- **Protocols:** TCP (reliable, in-order delivery, congestion control, flow control, connection setup) and UDP (unreliable, unordered delivery).

Multiplexing and Demultiplexing

- **Multiplexing:** Handling data from multiple sockets, adding transport headers.
- **Demultiplexing:** Using header info to deliver received segments to the correct socket.

Connectionless Transport: UDP - User Datagram Protocol

- **Characteristics:** Unreliable, unordered delivery, no setup needed, used in streaming multimedia apps, DNS, SNMP, and HTTP/3.
- **UDP Segment Format:** Includes source port, destination port, length, and checksum.

Reliable Data Transfer

ACK - Acknowledgment message sent by the receiver to confirm successful receipt of data. This forms a crucial part of reliable data transfer protocols by providing feedback to the sender about transmission status.

NAK - Negative Acknowledgment message sent by the receiver to indicate that a data segment was received with errors or was not received at all. NAKs trigger retransmission of the affected data segment from the sender.

- **Protocols:** RDT 2.0 (handles bit errors with ACKs and NAKs), RDT 2.1 (handles corrupted ACK/NAKs with sequence numbers), RDT 2.2 (NAK-free protocol), and RDT 3.0 (handles packet loss with timers and retransmissions).
- **Techniques:** Stop-and-wait, pipelining, Go-Back-N, and Selective Repeat.

Connection-Oriented Transport: TCP - Transport Control Protocol

- **Features:** Reliable data transfer, flow control, connection management.
- **TCP Segment Structure:** Includes sequence numbers, acknowledgment numbers, and various flags for managing connections.

Principles of Congestion Control

- **TCP Congestion Control:** Mechanisms to avoid network congestion, ensuring efficient data transfer.

Port Numbers

- Well-known ports (0-1023)
- Registered ports (1024-49151)
- Dynamic/Private ports (49152-65535)

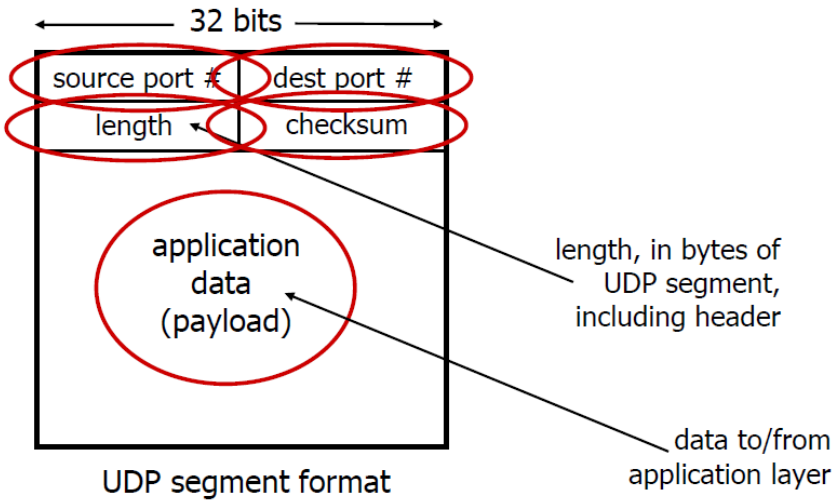
Summary

- **Multiplexing/Demultiplexing:** Based on segment header values.
- **UDP:** Simple, no-frills protocol, suitable for applications that can tolerate loss.
- **TCP:** Reliable, connection-oriented protocol with mechanisms for error recovery and congestion control.

Key Takeaways

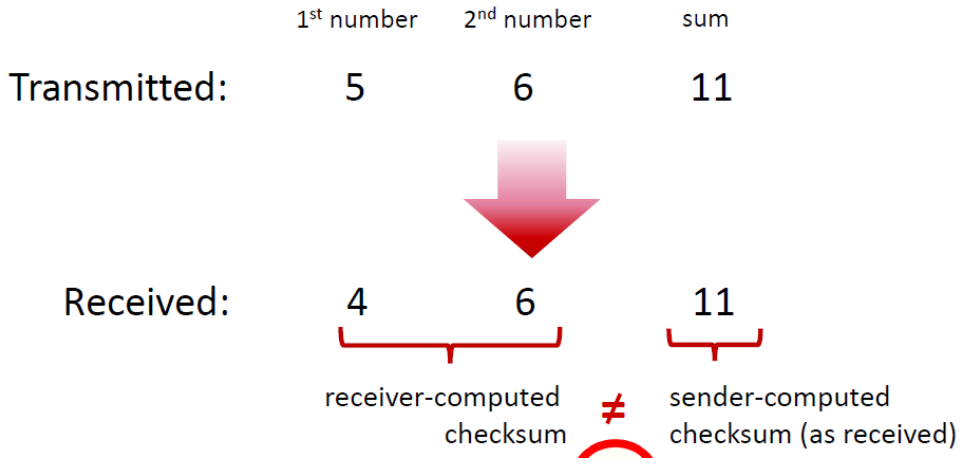
- Transport layer bridges application and network layers
- Provides essential services for reliable data transfer
- Choice between TCP and UDP depends on application requirements
- Port numbers enable multiple applications to communicate simultaneously

UDP segment header



UDP checksum

Goal: detect errors (*i.e.,* flipped bits) in transmitted segment



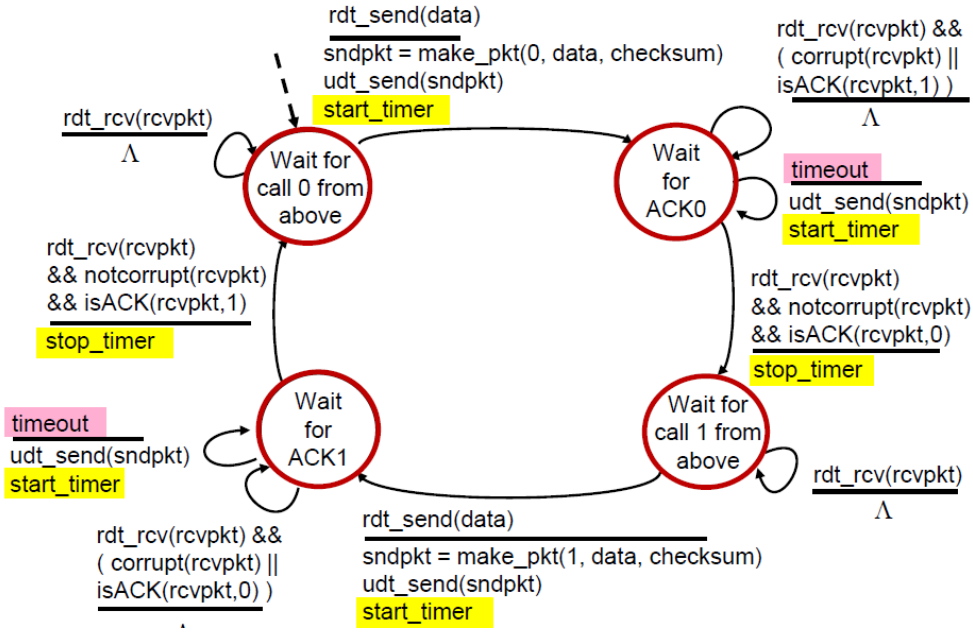
Internet checksum: an example

example: add two 16-bit integers

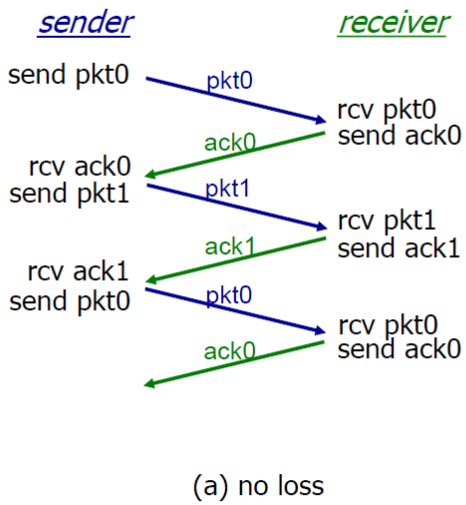
		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

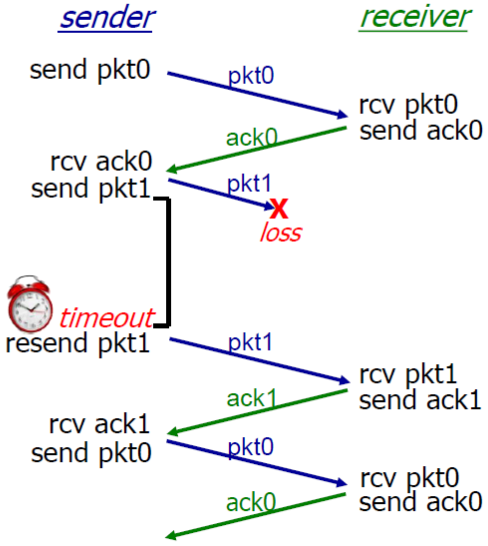
rdt3.0 sender



rdt3.0 in action

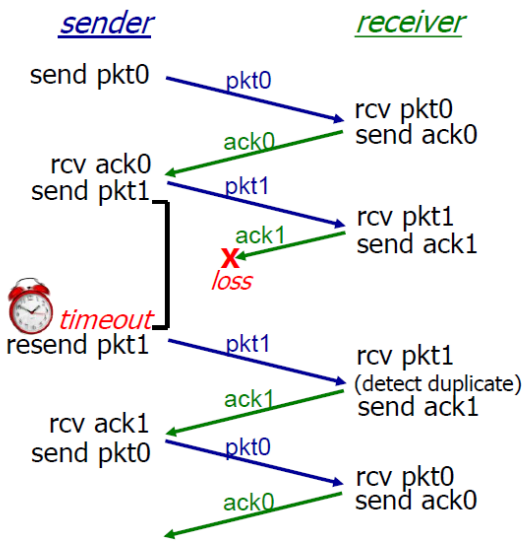


(a) no loss

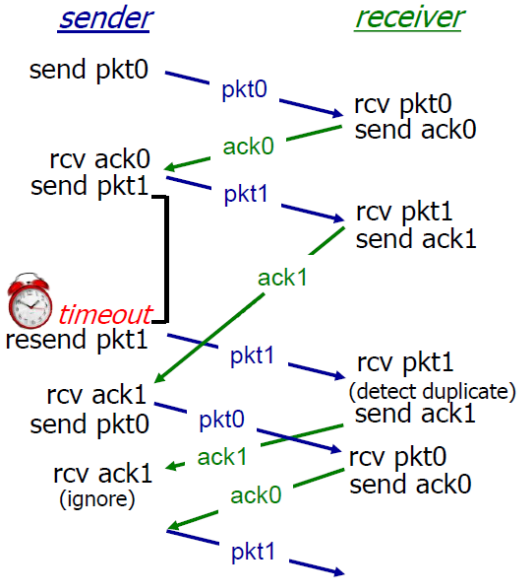


(b) packet loss

rdt3.0 in action



(c) ACK loss



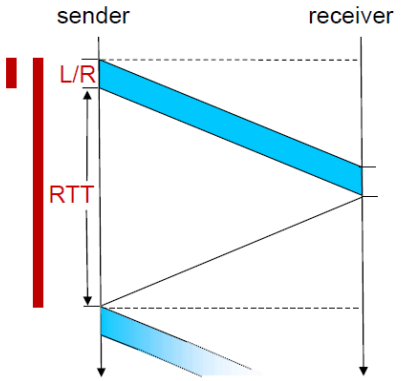
(d) premature timeout/ delayed ACK

rdt3.0: stop-and-wait operation

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

$$= \frac{.008}{30.008}$$

$$= 0.00027$$



- rdt 3.0 protocol performance stinks!
- Protocol limits performance of underlying infrastructure (channel)