CS 372 Lecture #16

Reliable data transfer

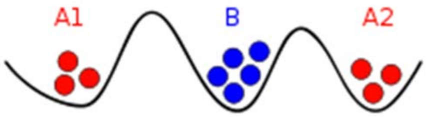
- motivations, concerns, and principles
- error detection

Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach,* 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

Two Generals Problem

- Cannot guarantee that message will be received
- Cannot guarantee that received message has no errors

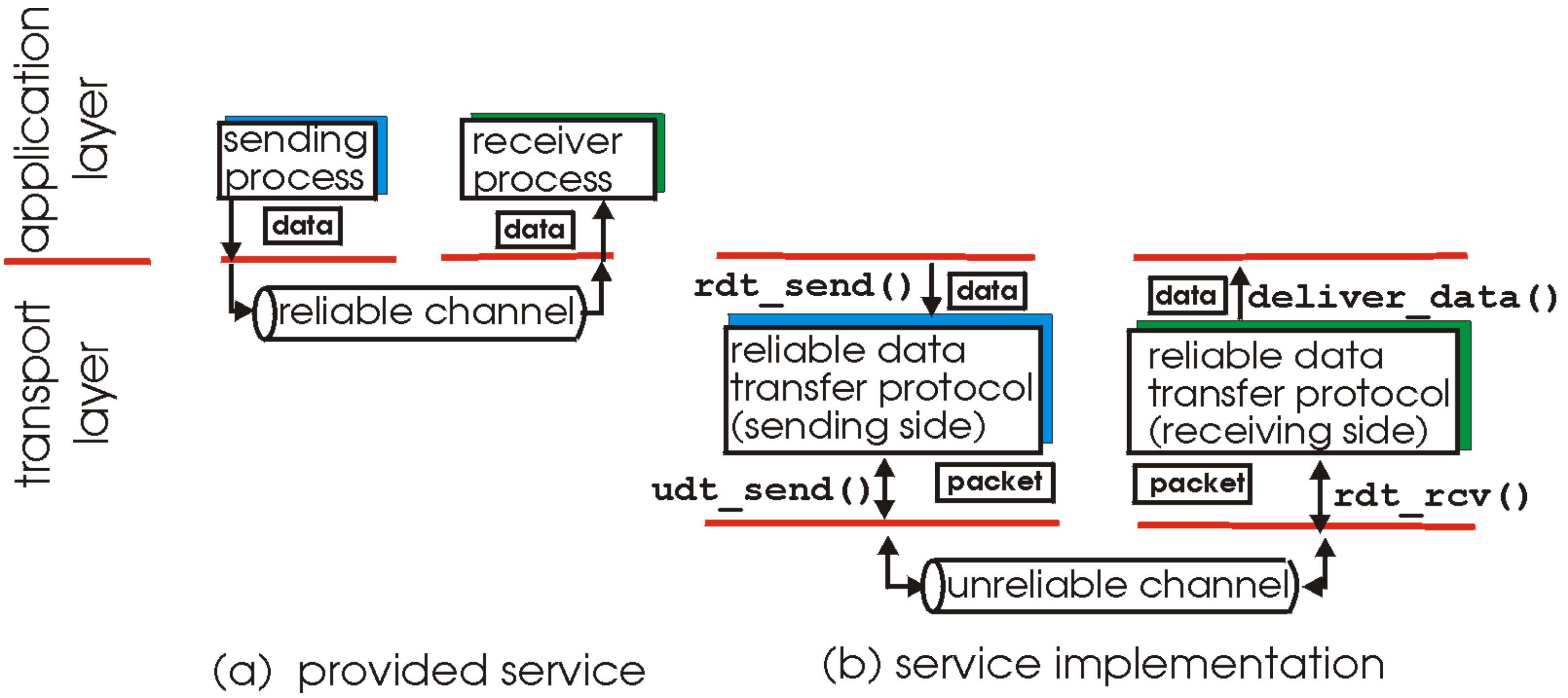
- Is reliable messaging possible?
  - See “Two Generals Problem” on wikipedia
  - See also RFC1149 ☺



Positions of the armies. Armies A1 and A2 need to communicate but their messengers may be captured by army B.

Principles of reliable data transfer

- implemented in application, transport, network, link layers
- **top-10 list of important networking topics!**

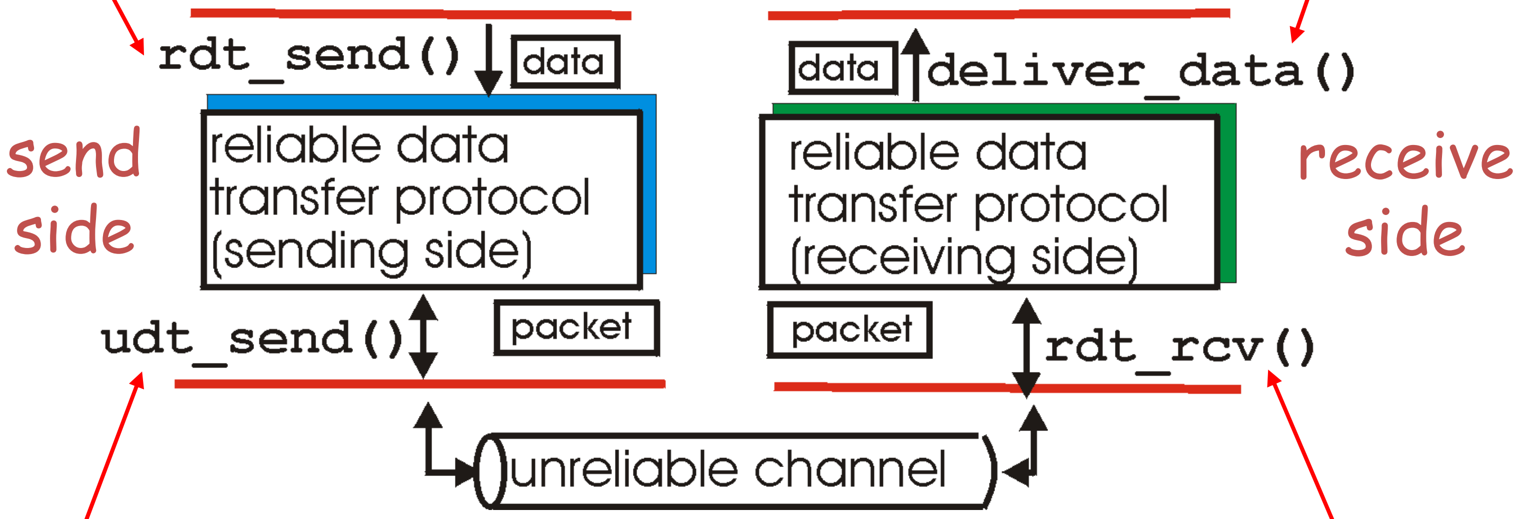


- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Reliable data transfer: getting started

rdt_send(): called from application layer. Data to be delivered to receive-side application layer

deliver_data(): called by **rdt** to deliver data to application layer



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

rdt_rcv(): called from “below” when packet arrives on receive-side of channel

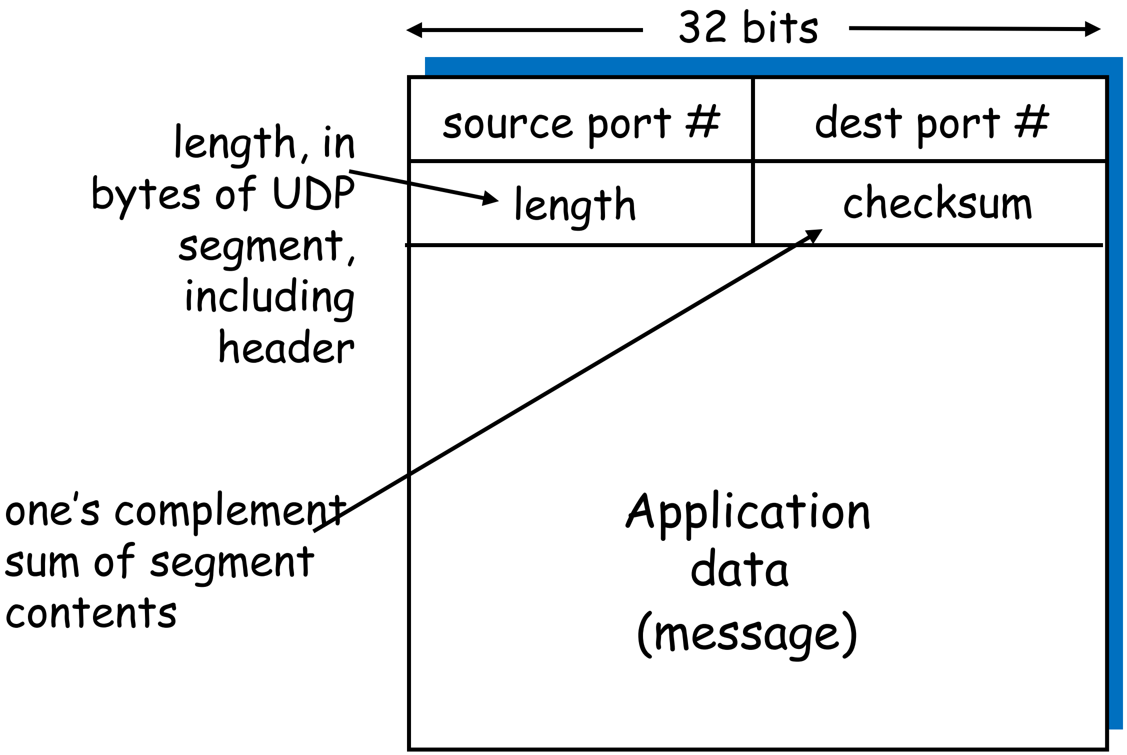
Reliable data transfer: getting started

- "reliable" is a relative term
- Small steps:
  - Error detection
  - Acknowledgement
  - Sequencing
  - Timing (flow/congestion control)
  - Retransmission
  - Fairness

- See textbook for development of reliable data transfer (using finite state diagrams)

Error detection in UDP [RFC 768]

- In addition to port numbers, UDP segment header includes
  - 16-bit *length* field
  - 16-bit *checksum* field



32 bits
source port #
dest port #
length
checksum
Application data (message)
length, in bytes of UDP segment, including header
one's complement sum of segment contents

UDP segment format

UDP checksum

Goal: detect errors (e.g., flipped/lost bits) in transmitted segment

sender:

- start checksum = 0
- compute checksum:
  - ones-complement of sum of segment contents as 16-bit integers
  - see www.netfor2.com/checksum.html

Discussion question: Why use the 1's-complement, instead of just the sum?

receiver:

- compute checksum of received segment
- compare computed checksum to segment checksum field.
  - **Equal** - no error detected
    - *But may be errors anyway*!!
  - **Not equal** - error detected
    - *Discard entire packet*

Internet checksum: 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

UDP: Summary

- “no frills,” “bare bones” transport protocol
- “best effort” service
- basic error detection
- UDP segments may be
  - lost
  - delivered out-of-order
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

- UDP use:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DNS

why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

Summary Lecture #16

- Two generals problem
- Reliable data transfer
- Error detection
- Characteristics of UDP

Discussion topic: When UDP detects an error, the packet is discarded without warning to the sender. Discuss the advantages and disadvantages of implementing error-correction at the transport layer. (e.g., Hamming codes)