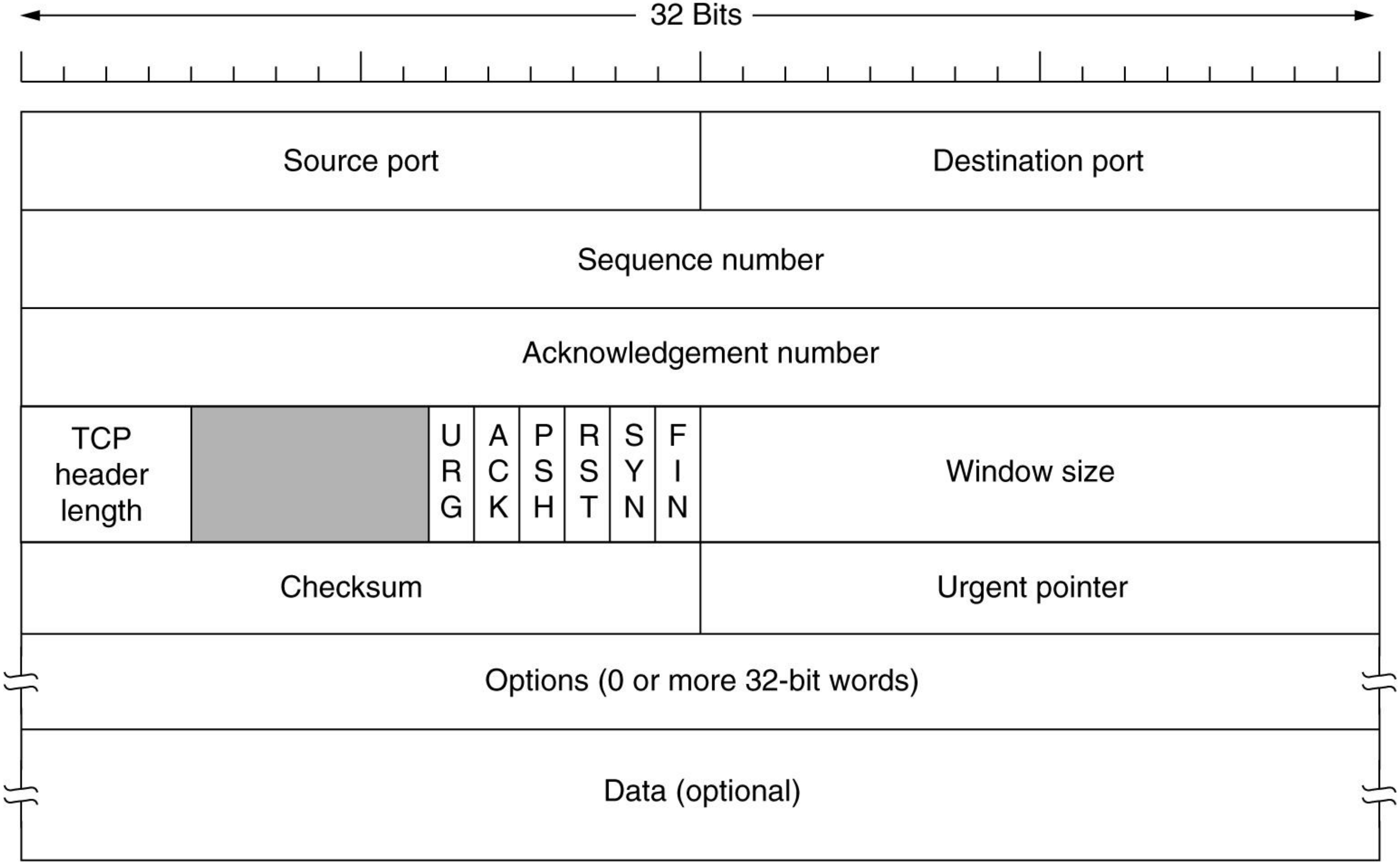
Computer Communications Networks

TCP Error Control

Review: TCP flow control

- Purpose
  - to avoid overflow
- Mechanism
  - sliding window
  - variable window

TCP packet header



32 Bits

Source port

Destination port

Sequence number

Acknowledgement number

TCP header length

URG

ACK

PSH

RST

SYN

FIN

Window size

Checksum

Urgent pointer

Options (0 or more 32-bit words)

Data (optional)

What can go wrong?

- IP packet delivery
  - duplicated
    - deleted by referring to sequence number; done
  - reordered
    - rearranged by referring to sequence number; done
  - corrupted
    - arrived but in “bad shape”
  - lost
    - *transmission error* or network congestion

Error detection

- Corrupted packets
  - detected by TCP checksum
    - action: drop!
- Lost packets
  - how do you tell if something ***is*** already lost?
  - TCP sender
    - timer
  - TCP receiver (cumulative acknowledgment)
    - duplicate acknowledgment

TCP/IP checksum

- Algorithm: 16-bit one complement of one’s complement sum with carry
  - 16-bit: padding when necessary
    - cover: TCP header, payload, pseudo header
  - calculate: pad, sum, with carry, complement
  - verify: sum == 0?
- IP pseudo header
  - to detect mis-delivered packets by IP layer
    - include: IP addresses, protocol ID, segment length

TCP/IP checksum (example)

- Sender:

```
    0000 0000 0000 0001
    1111 0010 0000 0011
    1111 0100 1111 0101
+   1111 0110 1111 0111

--------------------------------
10 1101 1101 1111 0000
⇒ 1101 1101 1111 0010
⇒ 0010 0010 0000 1101
   (checksum)
```

Receiver:

```
    0000 0000 0000 0001
    1111 0010 0000 0011
    1111 0100 1111 0101
    1111 0110 1111 0111
+   0010 0010 0000 1101

--------------------------------
10 1111 1111 1111 1101
 => 1111 1111 1111 1111
 => 0000 0000 0000 0000
    Success!
```

TCP sender timer

- TCP sender
  - start a timer when sending out a packet
    - in reality: one timer per a window of packets
  - on acknowledgment “covering” this packet
    - cancel the timer and setup another one
  - if timer timeouts: *indicate* packet may be lost
- Timeout value
  - too soon: unnecessary transmission
  - too late: “slow response”

TCP timeout value

- RTO: based on round-trip time (RTT)
  - RTT: from sending a packet to receiving its ack
  - smoothed RTT (SRTT)
    - exponentially weighted moving average (EWMA)
    - $\mathrm{SRTT}_{i+1} = \mathrm{SRTT}_i + a\,(\mathrm{RTT}\text{-}\mathrm{SRTT}_i)$; $a = 1/8$
  - RTT variance (RTTV)
    - $\mathrm{RTTV}_{i+1} = \mathrm{RTTV}_i + b(|\mathrm{RTT}\text{-}\mathrm{SRTT}_i|\text{-}\mathrm{RTTV}_i)$; $b = 1/4$
  - RTO
    - RTO = d (SRTT + c RTTV); c: initially 2 now 4
    - d: backoff factor; normally 1, doubled when timeout

Duplicate acknowledgment

- TCP acknowledgment
  - cumulative acknowledgment
  - example
    - rcv: [0, 500),[500, 1000),[1500, 2000),[2000, 2500)
    - ack: 500,1000,1000 (1st dupack),1000 (2nd dupack)
- Enough duplicate acknowledgments
  - *indicate* packet loss may have occurred
    - ack: 500, 1000, 1000, 1000, 1000 (3rd dupack)
    - packet [1000,1500) is considered lost

Error recovery

- End-to-end retransmission
  - go-back-N (GBN)
    - retransmit from ackno and upward
  - selective retransmission
    - only retransmit those “known” to be lost
- TCP’s error recovery
  - mostly GBN
    - also regulated by congestion control
  - explore further: TCP selective acknowledgment

Summary

- TCP error control
  - purpose
  - mechanisms
    - detection
    - recovery
- Explore further
  - TCP selective acknowledgment (SACK)
  - http://www.icir.org/floyd/

Next

- TCP congestion control