Internetworking and Reliable Transmission

CSE 561 Lecture 3, Spring 2002.
David Wetherall

Overview

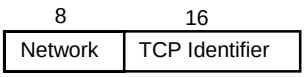
- Internetworking
  - Addressing
  - Packet size
  - Error detection
  - Gateway services
- Reliable Transmission
  - Stop and wait
  - Sliding window
  - Connections

How to connect different networks?

- Cerf&Kahn74, "A Protocol for Packet Network Intercommunication"
  - Foundation for Internetworking and hence, the Internet
- Gateways
  - Connect different networks together
  - One half of gateway is in each network
- Two options
  - Translation: Gateway translates directly between different network formats
  - **Indirection**: Gateway translates between local network format and universal "intermediate" format
- Pros/cons of each approach?

Addressing

- Hierarchical addressing
  - Global inter-network address
  - Local network-specific address



- Why hierarchical?
- Assumptions about networks?

Packet size

- Heterogeneous maximum packet size
  - E.g. Ethernet 1500B, FDDI 4500B, SLIP ~250B
- Options
  - Mandate minimum packet size
    - The Internet did this (576B IPv4, ~1500 IPv6)
  - Fragmentation
    - Gateway splits packets into smaller acceptable packets
    - We'll talk about this next time

Error detection

- **Bit errors**
  - Checksum written at end of packet
  - Checked by receiving hosts. Pros/cons?
  - CRC vs Checksum: performance/detection tradeoff
  - Error distribution assumptions?
    - "When the CRC and TCP Checksum Disagree", Stone& Partridge, SIGCOMM 2000.
- **Packet losses**
  - We'll deal with this shortly

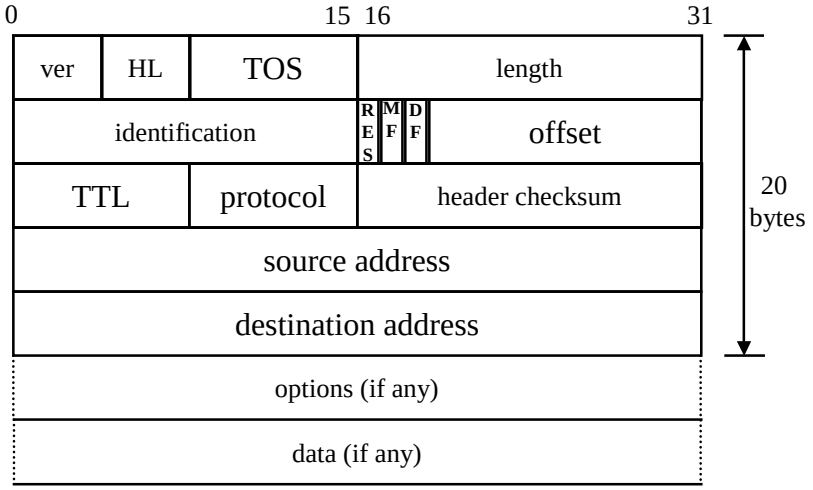
Gateway services

- Things Gateways do
  - Forward packets
  - Fragment (must parse sequence numbers/length)
- Things they do not do
  - Reassembly or duplicate detection. Why?
  - Error detection
  - Loss detection/retransmit requests
  - Failure detection

How IP works today?

- **Packet format**
  - Header checksum
  - Time-to-live
  - Protocol demultiplexing
  - Type-of-Service
  - Options
- **Addressing**
- **Fragmentation**
  - Next lecture

IP Packet Format

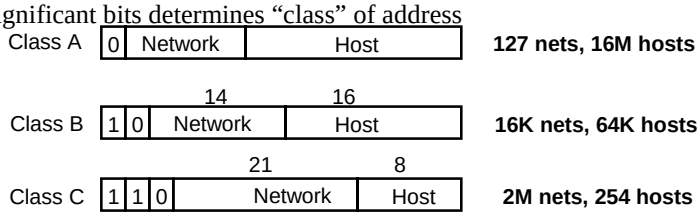


IP addressing

- 32-bits in an IPv4 address
  - Dotted decimal format a.b.c.d
  - Each represent 8 bits of address
- Network part and host part
  - E.g. IP address 132.239.15.3
  - 132.239 refers to the UCSD campus network
  - 15.3 refers to the host gremlin.ucsd.edu
- Which part is network vs host?
  - Pre-1993 class-based addressing, static determination
  - Post-1993 classless addressing, dynamic split
    - This is what the “network mask” is all about

Class-based routing (<1993)

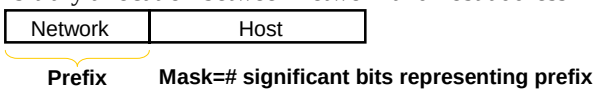
- Most significant bits determines "class" of address



- Pro
  - Single lookup to find address
- Con
  - Fragmentation
  - Can't aggregate

Classless addressing (now)

- In 1993, Classless Inter-Domain Routing introduced
  - Routes represented by tuple (network prefix/mask)
  - Allows arbitrary allocation between network and host address

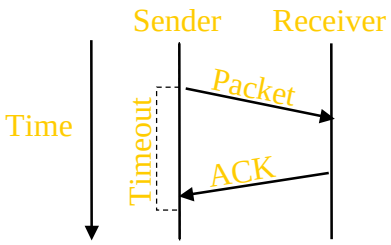


  - e.g. 10.95.1.2/8: 10 is network and remainder (95.1.2) is host
- Pro:
  - Finer grained allocation, aggregation of suballocations
- Con:
  - More expensive lookup: longest prefix match

Reliable Transmission

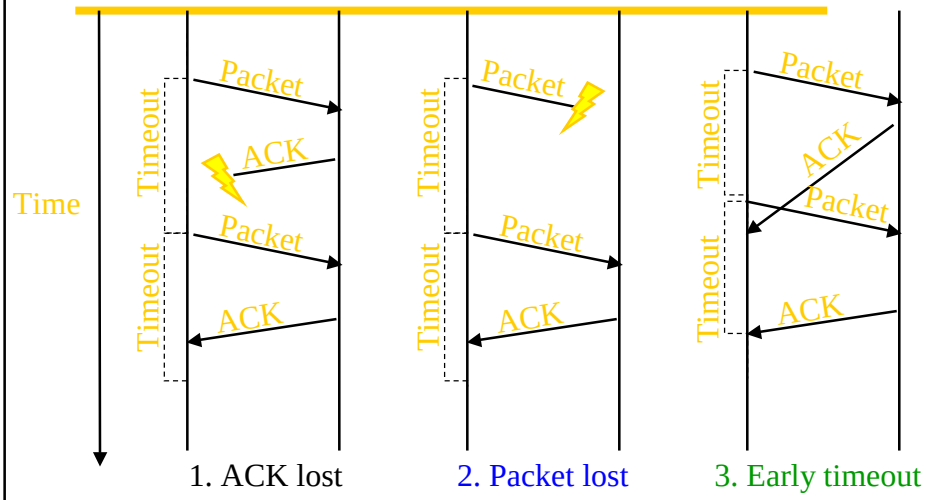
- How do we reliably send a message when packets can be lost in the network?
- Obvious options:
  - Detect a loss and retransmit (Cerf&Kahn74)
  - Send redundantly (Byers et al.98)

Simplest Protocol: Stop and Wait



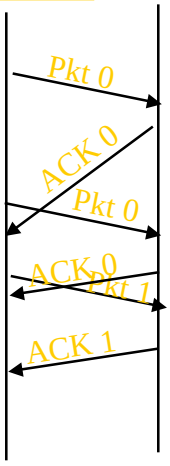
- Send a packet
- Stop and wait until an acknowledgement arrives from receiver
- Retransmit if timeout occurs before ACK arrives

Recovering From Errors



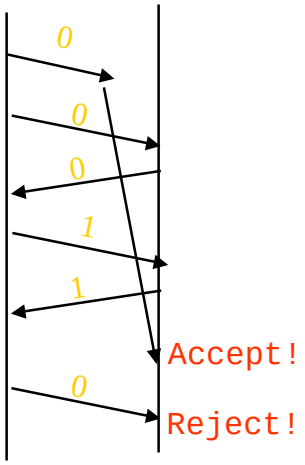
How does receiver recognize duplicate?

- Add sequence numbers to packet
  - Both in data packets and ACKs
- Sequence # in packet is finite, though
- How many bits do we need?
  - One bit for stop and wait
  - Won't send seq#1 until receive ACK for seq#0



What if packets are delayed?

- Never reuse a seq #? Finite…
- Require in order delivery?
- Prevent very late delivery?
  - TTL: Decrement hop count per packet, discard if exceeded
  - Seq #s not reused within delay bound
  - TCP standard: Maximum segment lifetime (MSL) of 120 seconds.

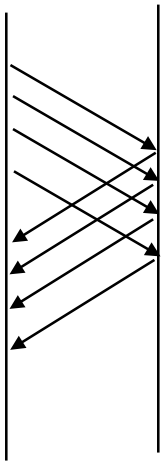


Performance issues

- **Capacity**
  - For a network with bandwidth BW and delay D, a sender can transmit BW*D bytes before the network is "full"
  - Stop-and-**wait** is inefficient… pipe is empty most of the time
- **Delay**
  - Time to detect loss limited by timeout. How to select timeout?
  - How could you do better that a timeout-based scheme?

Pipeline transmission …

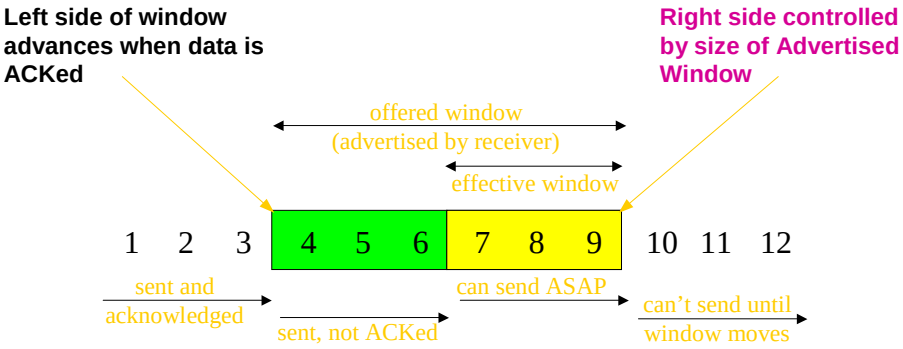
- Send multiple packets without waiting for the first to be ACKed
- Hypothetical, reliable, unordered delivery:
  - Send new packet after each ACK
  - Sender keeps list of unACK'ed packets and resends after timeout
  - Receiver same as stop & wait
- This leads to Sliding Window, where an allowed range of numbered packets can be outstanding at any given time.



Sliding Window

- Sender puts sequence number on packet
- Receiver acknowledges packets by number
  - Replies with highest in-order number received
- Receiver includes the highest number that can be sent
  - It controls the size of the window
- Sliding window is a mechanism. It implements multiple functions:
  - Reliability (retransmit unacknowledged packets)
  - Ordered delivery (use numbering to order)
  - Flow control (match fast sender to slow receiver)

Visualizing a Sliding Window



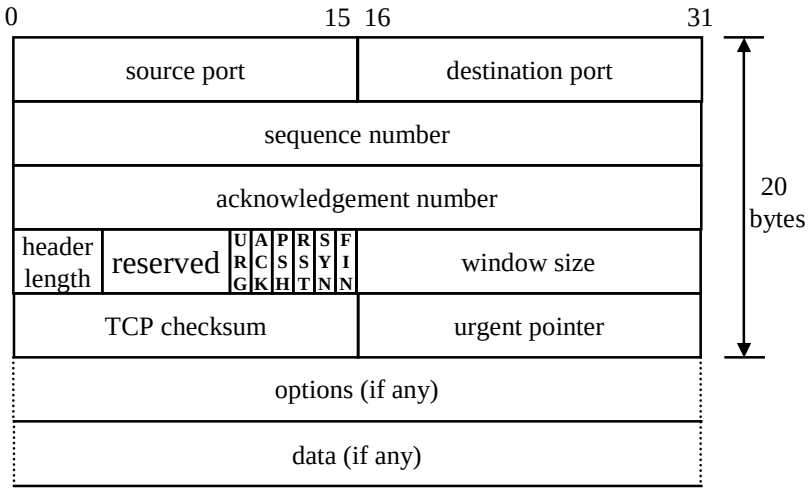
What if we lose a packet?

- **Go back N**
  - Receiver provides cumulative ACKs "got up through packet k"
    - If multiple packets received, only one ACK needed
  - OK for receiver to buffer out of order packets
    - Should you send an ACK for out-of-order packets?
  - On timeout, sender restarts from k+1
- **Selective acknowledgement** (SACK)
  - Receiver sends ACK for each packet in window
  - On timeout, sender resends only the missing packet
- We cover ways to avoid timeouts later
  - Fast Retransmit (TCP Congestion Control)

How TCP works today?

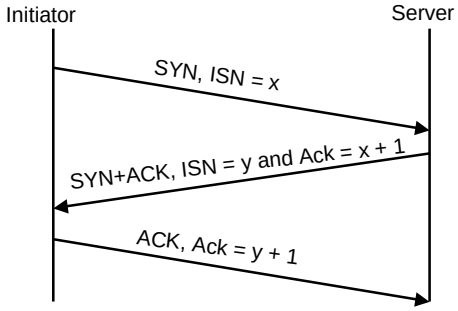
- **Packet format**
  - Process demultiplexing
  - Data checksum
  - Options
- **Connection management**
- **Flow control**

TCP Packet Format



TCP Connection Establishment

- How to synchronize initial sequence numbers on each side?
  - Touched on in Cerf&Kahn, but they didn't know how to do it right
  - Tomlinson invents three-way handshake in 1975



- Why do we need the last ACK?

TCP Flow control

- Sliding window
  - Byte granularity for sequence numbers and advertised window
  - Pro/con of bytes vs packets?
- Go-Back-N where out-of-order packets buffered
  - More efficient alternatives later (congestion control)
  - Fast retransmit (invented for TCP in 1988)
  - SACK option (just becoming widespread)
- Lots of icky details
  - Window probes, Silly Window Syndrome, Nagel algorithm, MTU discovery, PAWS, etc…
  - Steven's "TCP/IP Illustrated" is a great source for these details

Example Icky Detail: Advertised Window Deadlock

- If the receiving process does not empty the buffer (e.g., not scheduled), then the sender fills up the receiver's buffer
  - Advertised Window is 0
  - Effective Window goes to 0 when all data is ACKed
- Problem: When can the sender start sending again?
  - No timeouts because all data is ACKed
  - No packets from receiver with a new Advertised Window because receiver isn't running
- Solution: Ping with a segment of 1 byte of data
  - Eventually receiver responds with a new Advert. Window