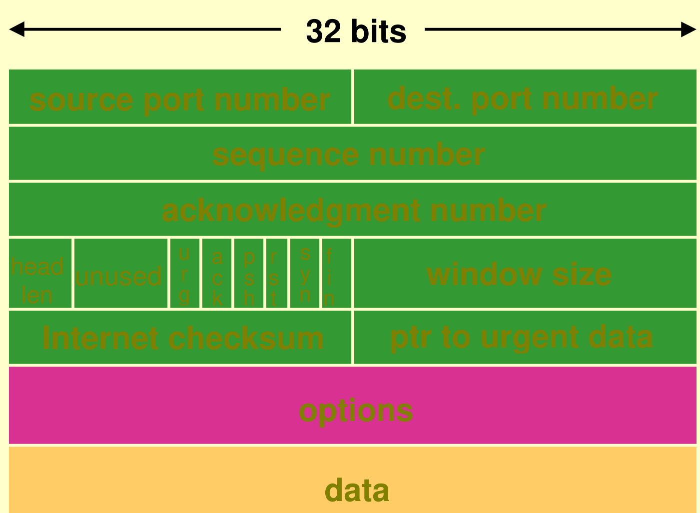
Data Transfer Case Study: TCP

- Go-back N ARQ
  - 32-bit sequence # indicates byte number in stream
  - transfers a byte stream, not fixed size user blocks
  - full duplex (bi-rectional) data transfer
  - sends upper level data "at its convenience" (RFC793!), trying to accumulate 512 bytes of data
- cumulative ACK: ACK(*n*) ack's all bytes up through *n*
  - ACK for received A-to-B data piggybacked on B-to-A data packet
- Internet checksum: add up data, take 1's complement
  - covers both header and data

TCP Packet Format



Data Transfer: XTP

- XTP: Xpress transfer protocol
- designed for high-speed, high-performance networks
- 32-bit sequence numbers with transition to 64-bit seq. numbers
- 32-bit priority field for different priority data
- user-selectable: reliable or unreliable data transfer
- Go-Back-N ARQ but receiver can also indicate spans of packets received
  - sender only retransmits gaps

Data Transfer: XTP

- checksum:
  - form of two-dimensional parity
  - header and data checksummed separately
  - data checksumming can be disabled
  - data checksumming at end (send data while computing checksum (need only touch data once)

Flow and Congestion Control

sometimes sender shouldn't send a ready packet:

- receiver not ready (e.g., buffers full)
- react to congestion
  - many unACK'ed packets may mean long end-end delays, congested networks
  - network itself may provide sender with congestion indication
- avoid congestion:
  - sender transmits smoothly to avoid temporary network overloads

Flow and Congestion Control

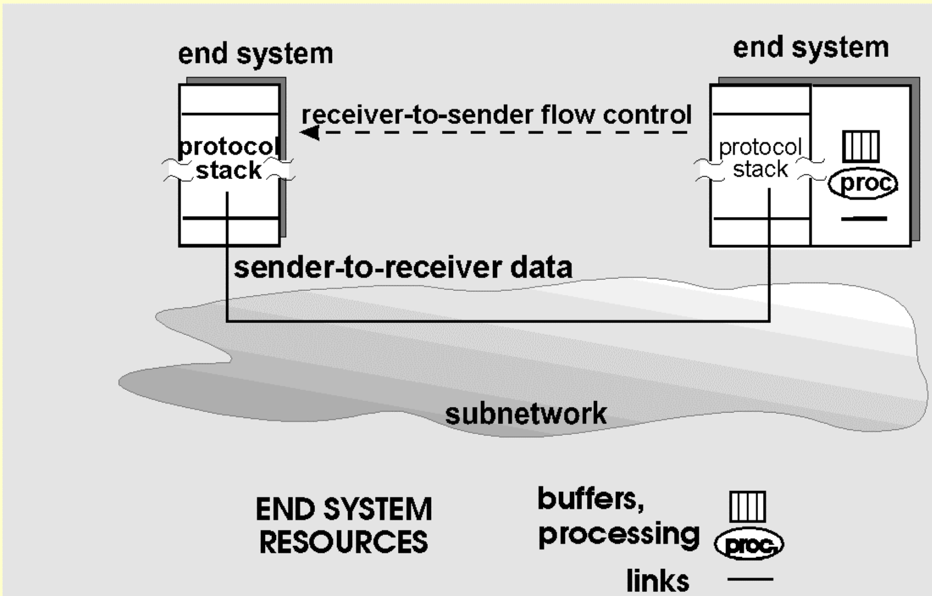
flow control: speed and resource matching of sender and receiver

- sender should not overwhelm receiver

congestion control: action taken in response to network layer (and below) congestion

- throttling sender is but one solution to congestion

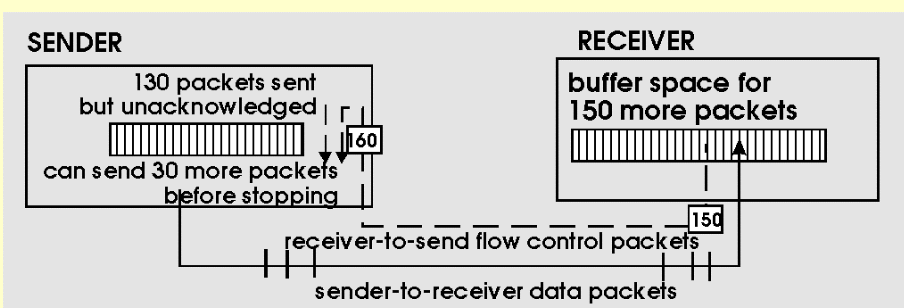
The flow control scenario



Two approaches towards flow control

Explicit flow control

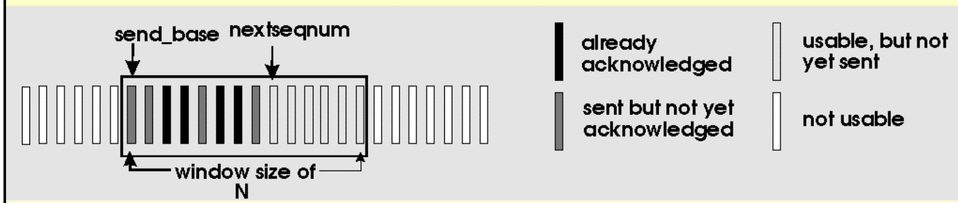
- receiver tells sender how much to send



Explicit flow control (cont)

Useful abstraction: sender maintains sliding window over sequence number, indicating what it can send

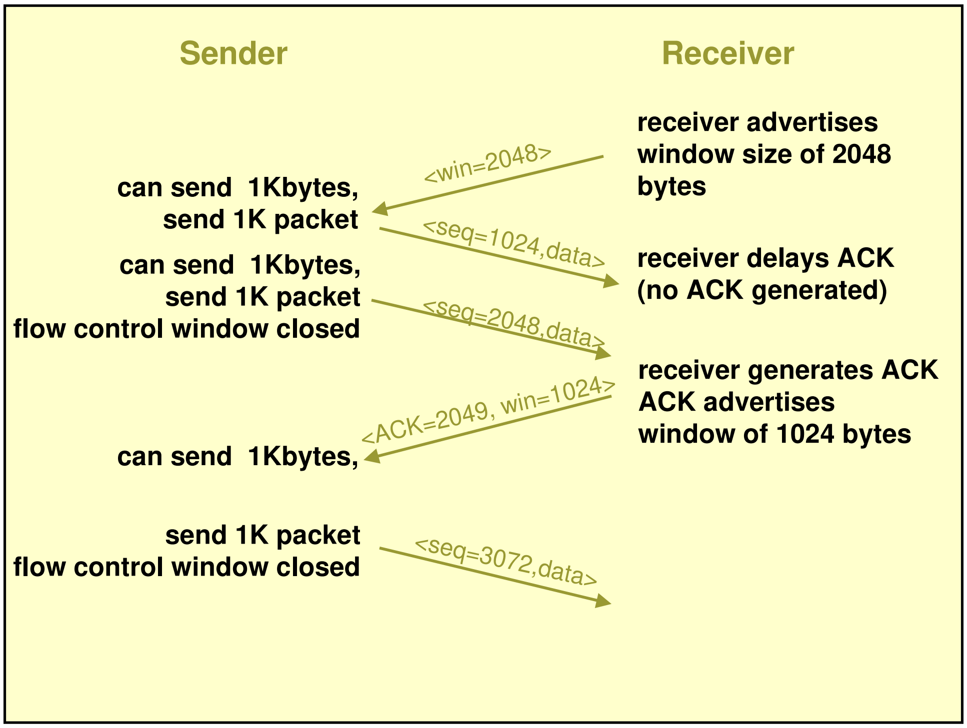
- done in TCP and TP4
- congestion (as opposed to flow) control window may further restrict sender



Flow Control in TCP

Receiver explicitly advertises available buffer space to sender:

- 16-bit "advertised window" specifies number of bytes (starting from ACKnum) receiver willing to receive
- max window size is 64K
- recent "scaling" option for larger windows



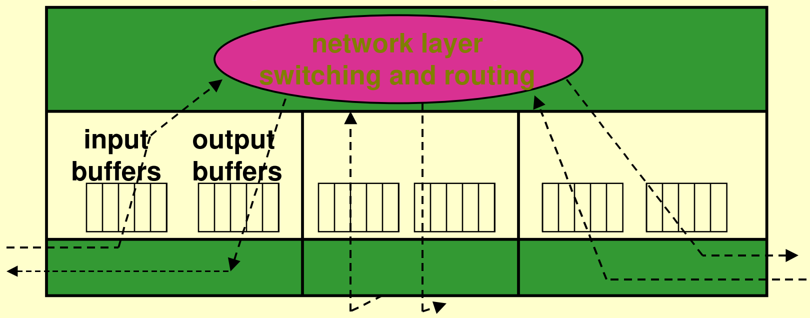
Implicit flow control

- receipt of ACK's triggers more sending
- delayed ACK (for whatever reason) slows down sender
- IBM virtual route pacing:
  - initially send window of *N* packets
  - ACK of first packet in this (and subsequent) windows of *N* allows sender to send *N* more
  - jumping", non-sliding window
  - max. number of unACK'ed packets?
  - decrease in traffic for explicit flow control

Congestion Control

Temporary demand for shared resources (links, processing, buffers) in *network* layer and below may exceed demand:

- packets buffered until resources available
- buffers full: packet lost (discarded)
  - which to discard?
- lots of buffering: excessive delays



Congestion Control: retransmission effects

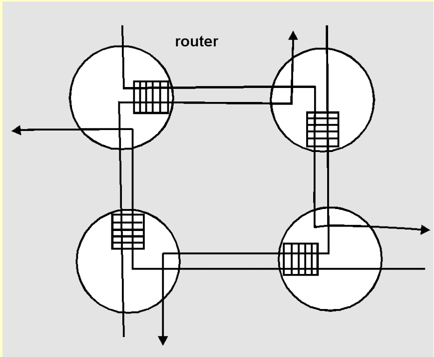
Ideal case:

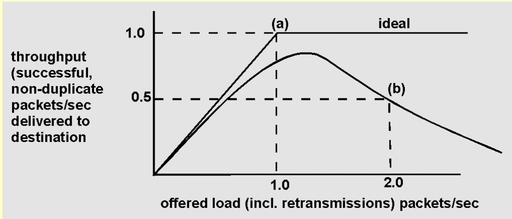
- every packet delivered successfully until subnet reaches capacity
- beyond capacity: deliver packets at capacity rate

In face of loss or long end-end delay, retransmissions can make things worse

- inject more (not less) traffic into net
- throwing fuel on fire!

Congestion Control: retransmission effects





Congestion Control (cont)

Realistically:

- as offered load increases, more packets lost, causing more retransmission, causing more traffic, causing more losses, …
  - at (b), each original packet sent four times on average
  - decreasing rate of transmission (e.g., a larger timeout value) ***increases*** overall throughput

Moral:

- when losses occur: ***backoff***, don't aggressively retransmit

However:

- ***social versus individual good:*** if most back off (and suffer) how to handle greedy sender who does not back off (and benefits)

Three Basic Approaches towards Congestion Control

End-end congestion control

- sender-observed congestion (e.g., delays, losses) used to control sender
- closed loop control

Network-indicated congestion control

- network layer provides feedback to sender

Rate-based control

- sender behavior fixed (bounded) over time
- open-loop control

Real-world protocols sometimes mix/combine these