CS 232

Transmission Control Protocol (TCP)

Internet Layering

Content (layer 7)
web pages, videos, blogs, RSS feeds, online radio stations, …

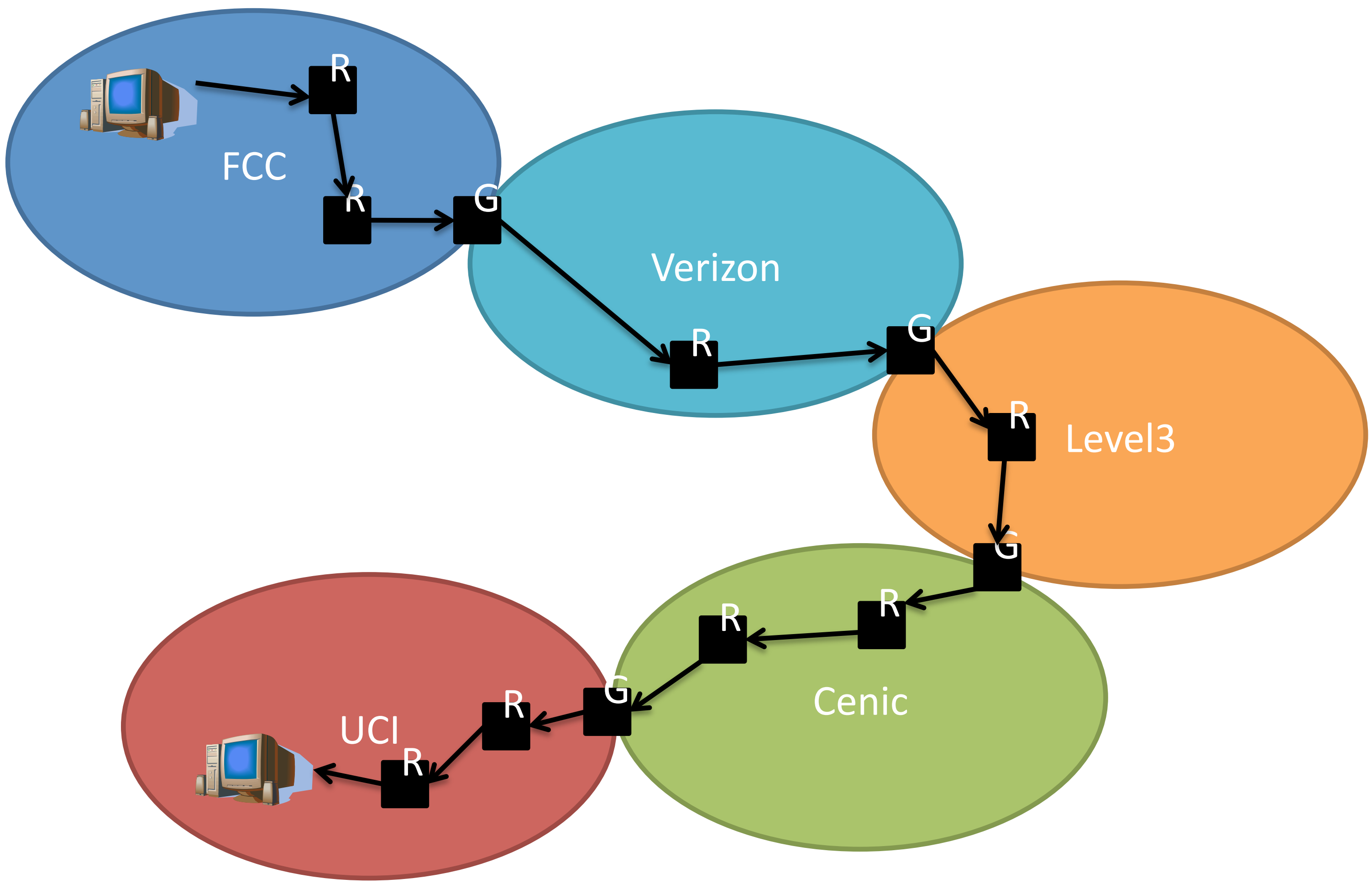
Applications (layers 5-7)
http, email, file transfer, streaming, instant messaging, web hosting, blogging, video conferencing, remote desktop, …

Transport (layer 4)
TCP congestion control

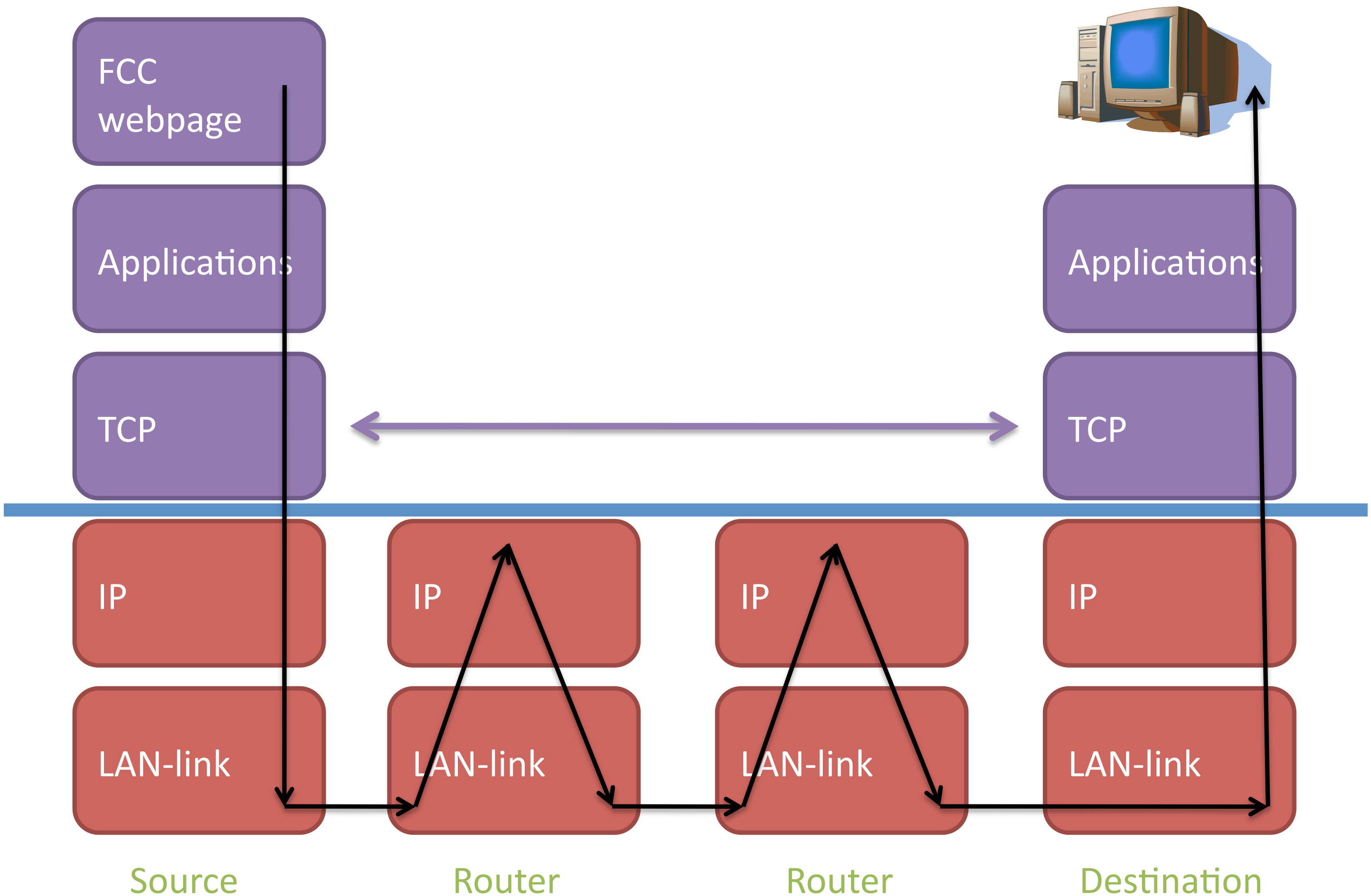
Network (layer 3)
IP addresses, routing

Physical & Link "LAN-link" (layers 1-2)
Ethernet, DSL, cable modems, Wi-Fi, local loop, …

Peering



TCP: end-to-end



Transport layer problems

- Routers may drop packets
- Packets may arrive out of order
- Destination may not be able to process received packets as fast as they arrive
  - Causing destination buffer overflow
- Routers in between source and destination may experience congestion
  - Routers may queue many packets, resulting in queuing delay
  - Routers may drop packets, resulting in packet loss

Reliable service

- Problem: routers may drop packets
- Solution: transport layer retransmits dropped packets
  - *if* want reliable service at transport layer

Connection-oriented service

- Problem: packets may arrive out of order
- Solution: transport layer puts packets in order before handing them to the application layer
  - *if* want connection-oriented service at transport layer

Flow control

- Problem: destination may not be able to process received packets as fast as they arrive
  - Causing destination buffer overflow
- Solution: slow down source rate to match destination rate
  - Called “flow control”

Congestion control

- Problem: routers in between source and destination may experience congestion
  - Routers may queue many packets, resulting in queuing delay
  - Routers may drop packets, resulting in packet loss
- Solution: slow down rates of sources causing congestion
  - Called "congestion control"

Interface to application layer

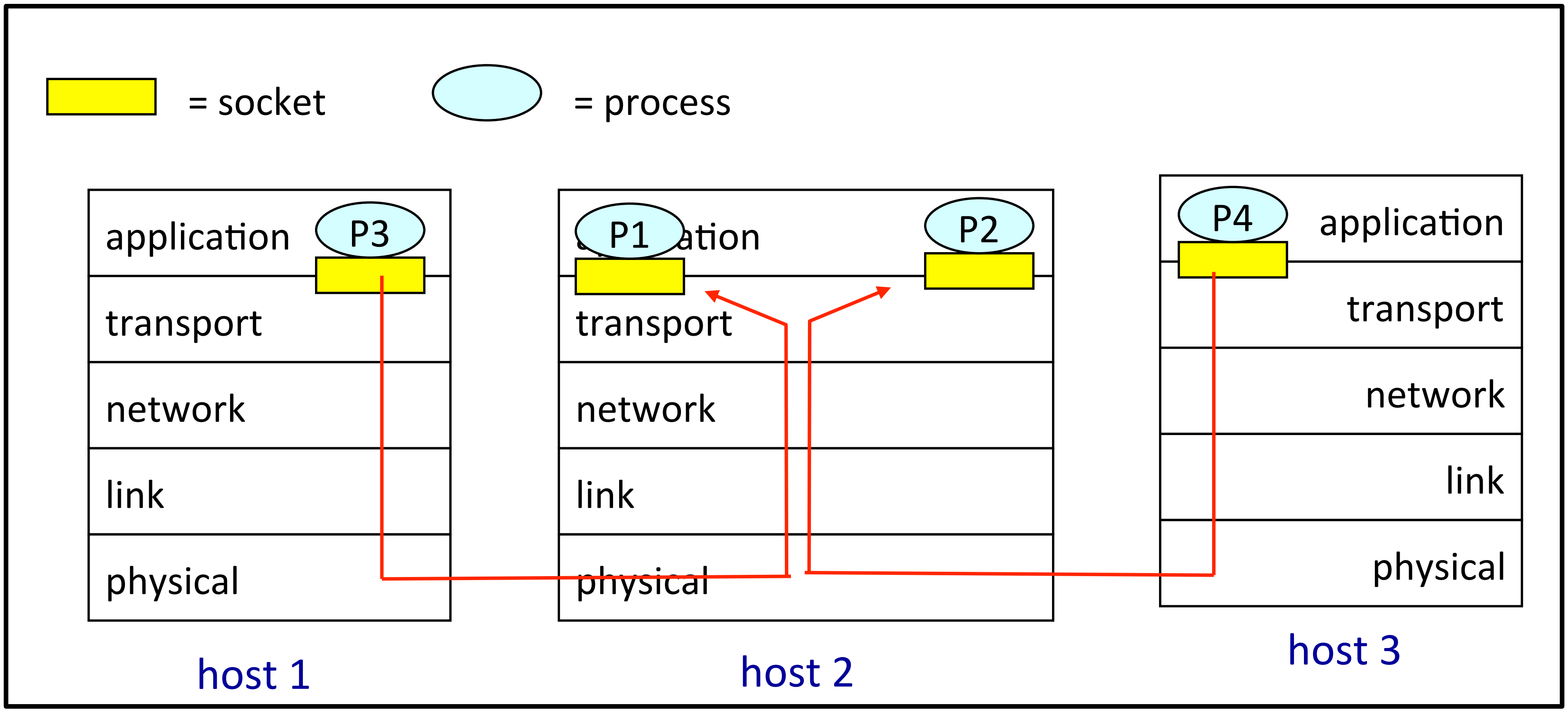
- Source side: accept application-layer messages from various processes
  - Note application choice reliable/unreliable, connection-oriented/connectionless, flow control, congestion control
  - “multiplexing”
- Destination side: feed received segments to various application layer processes
  - “demultiplexing”

Interface: ports

- Part of the interface from application layer to transport layer
- Multiple processes may be talking over Internet at the same time
- Among all packets coming into a machine, separate them by "port"
  - Port roughly corresponds to an application, e.g. port 80 for http, port 25 for smtp, …

Interface: sockets

- Interface to application layer through a "socket"

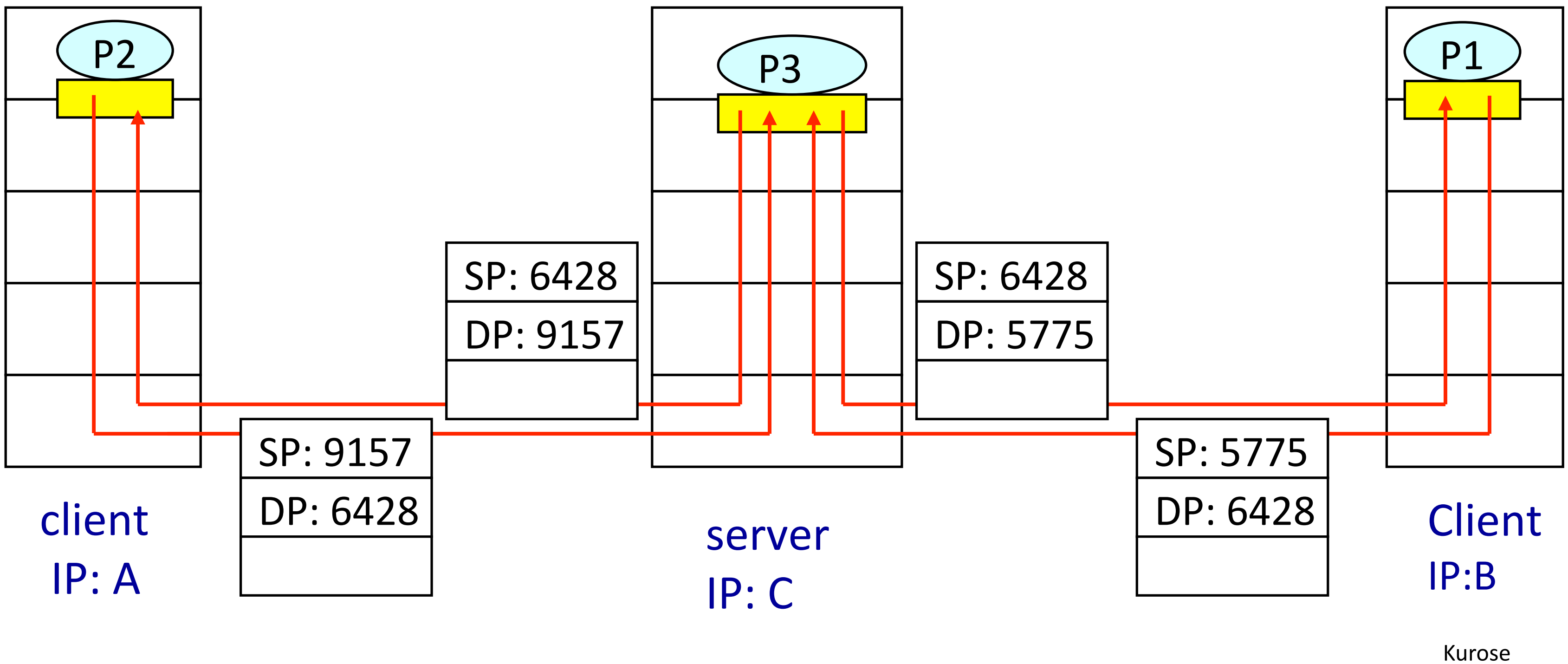


Kurose fig. 3.2

Interface: connectionless sockets

- Source side multiplexing
  - Unique application process for each active source port number
  - Application for a source port passes IP address, source port number, destination port number to transport layer
  - Transport layer puts source & destination port numbers in packet header, and passes destination IP address to network layer
- Destination side demultiplexing
  - Obtain destination port number from header
  - Pass to socket = (destination IP address, destination port)
    - to application process associated with destination port

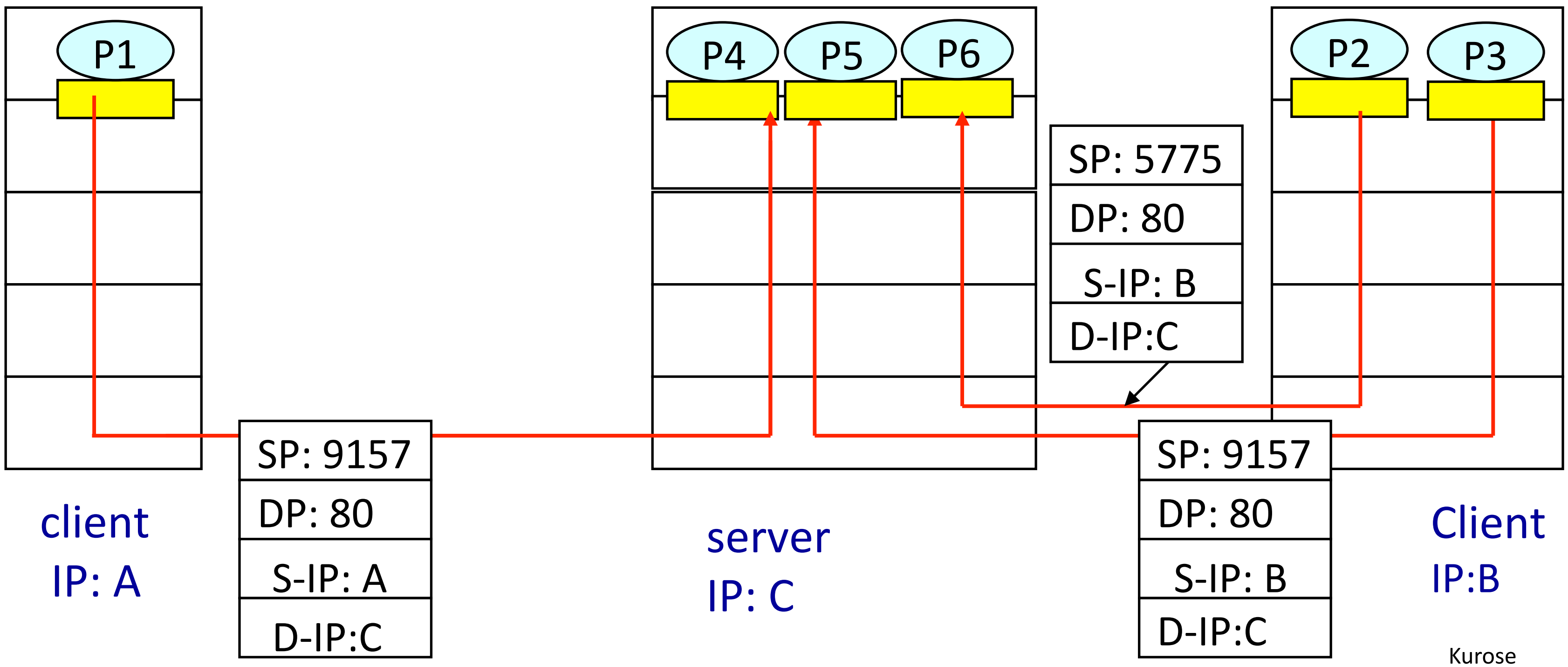
Interface: connectionless sockets



Interface: connection-oriented sockets

- Source side multiplexing
  - Unique application process for each active combination of (source port number, destination port number, destination IP address)
  - Application passes IP address, source port number, destination port number to transport layer
  - Transport layer puts source & destination port numbers in packet header, and passes destination IP address to network layer
- Destination side demultiplexing
  - Obtain source and destination port numbers from TCP header
  - Obtain sequence number from TCP header
  - Request dropped packets
  - Put packets in order
  - Pass to socket = (src IP, src port, dest IP, dest port)
    - to application process associated with (src IP, src port, dest port)

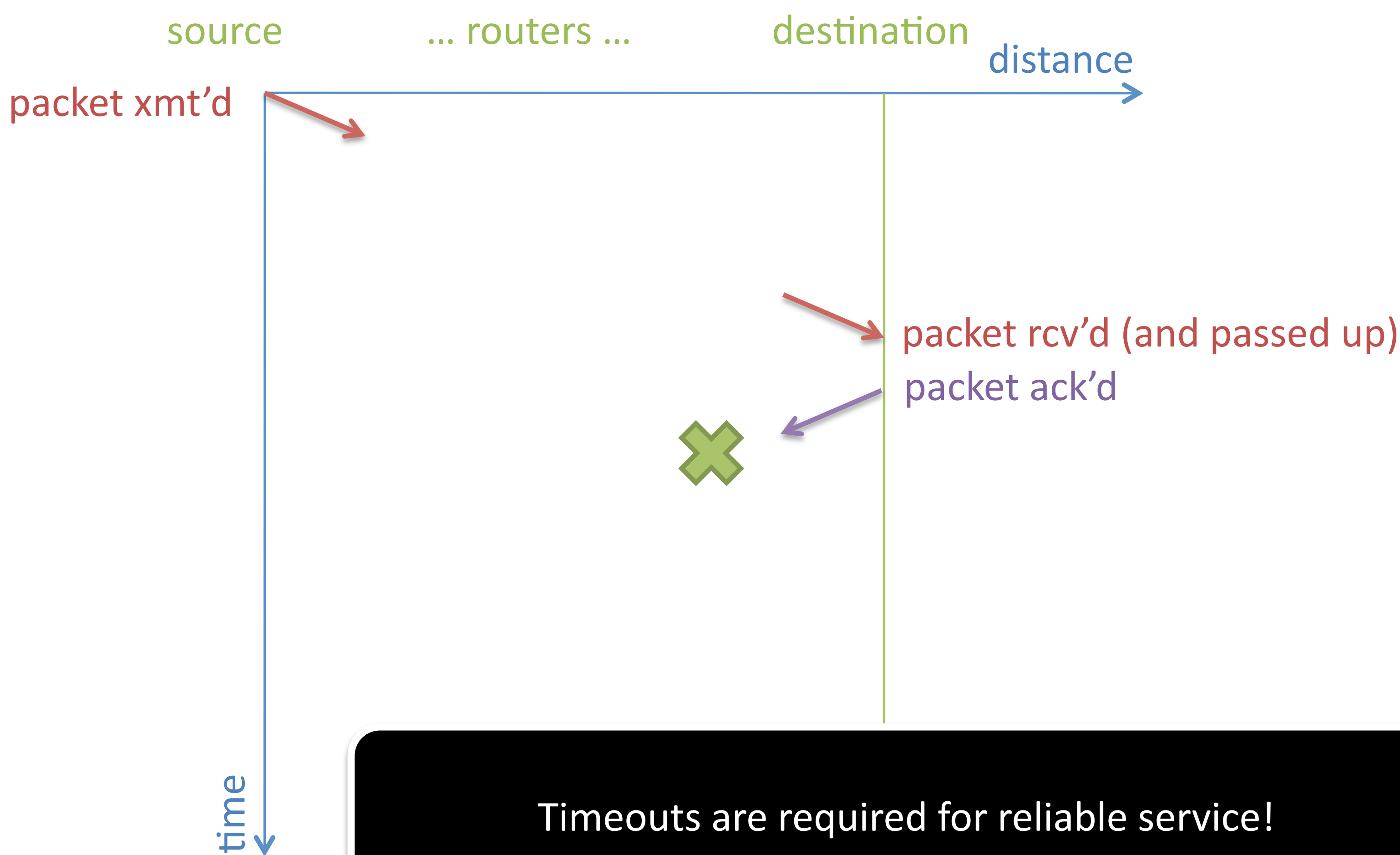
Interface: connection-oriented sockets



essential elements

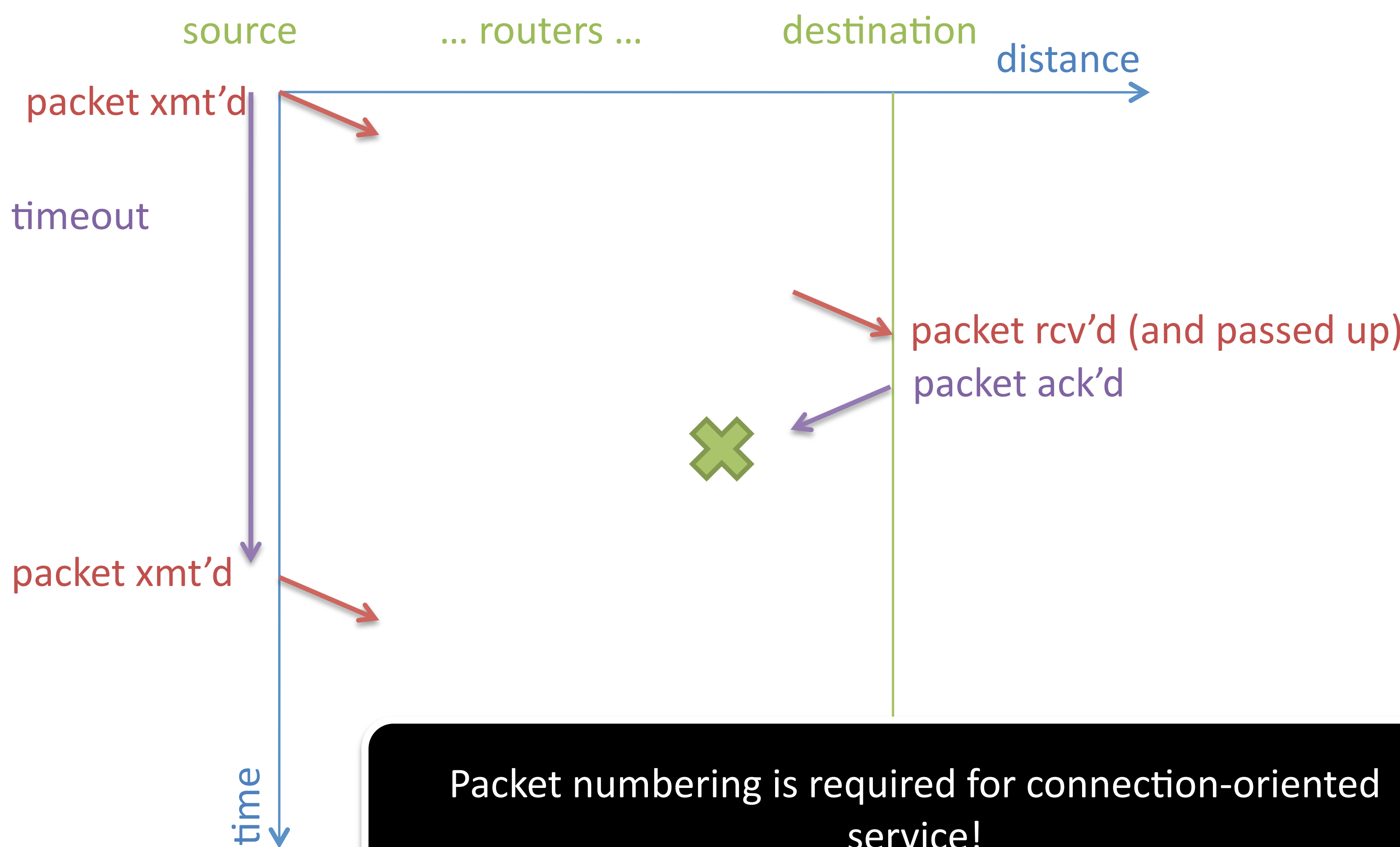
- Goals:
  - Reliable
  - Connection-oriented
- Components:
  - Packet numbering
  - Feedback
  - Retransmission

timeouts



Timeouts are required for reliable service!

packet numbering

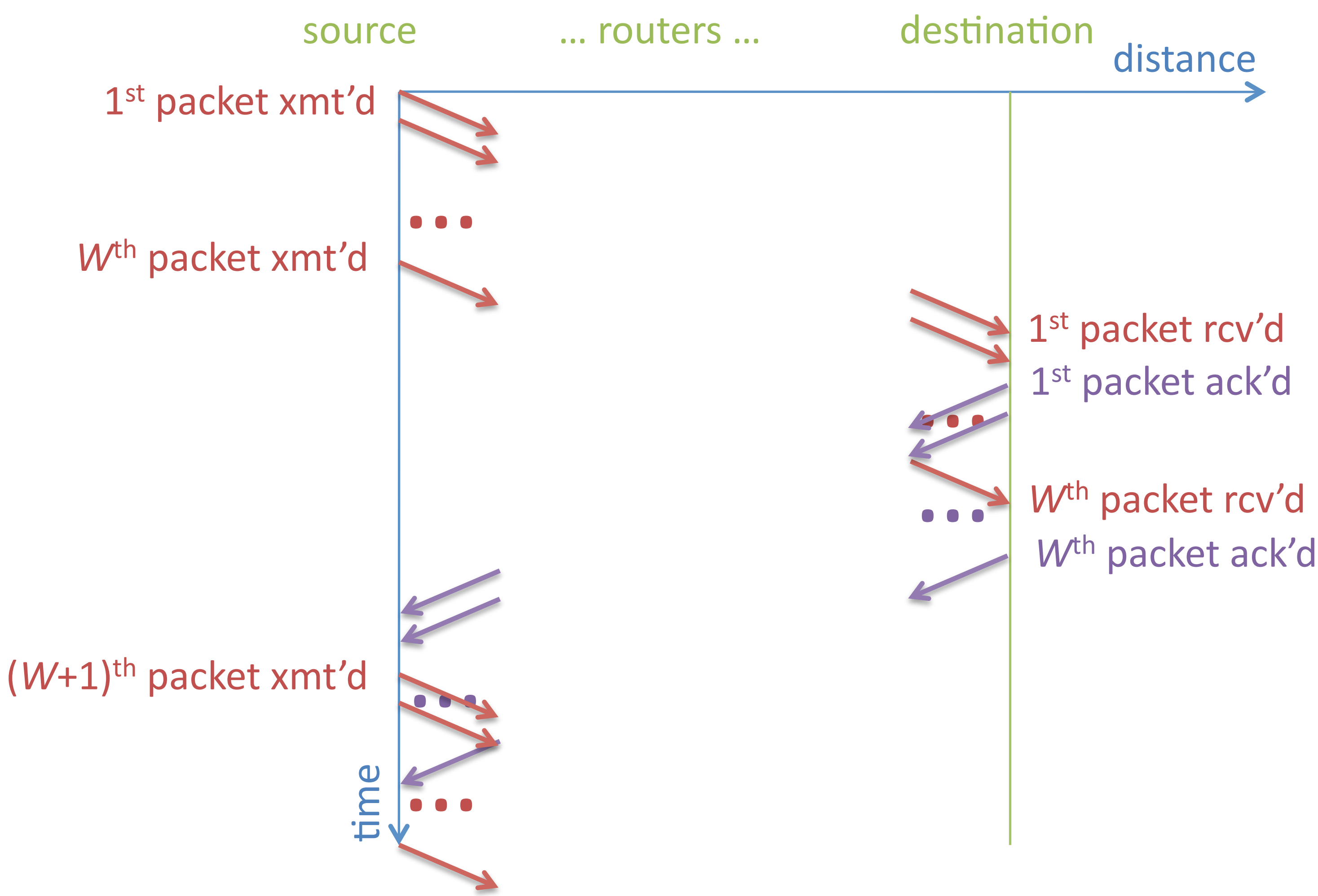


Packet numbering is required for connection-oriented service!

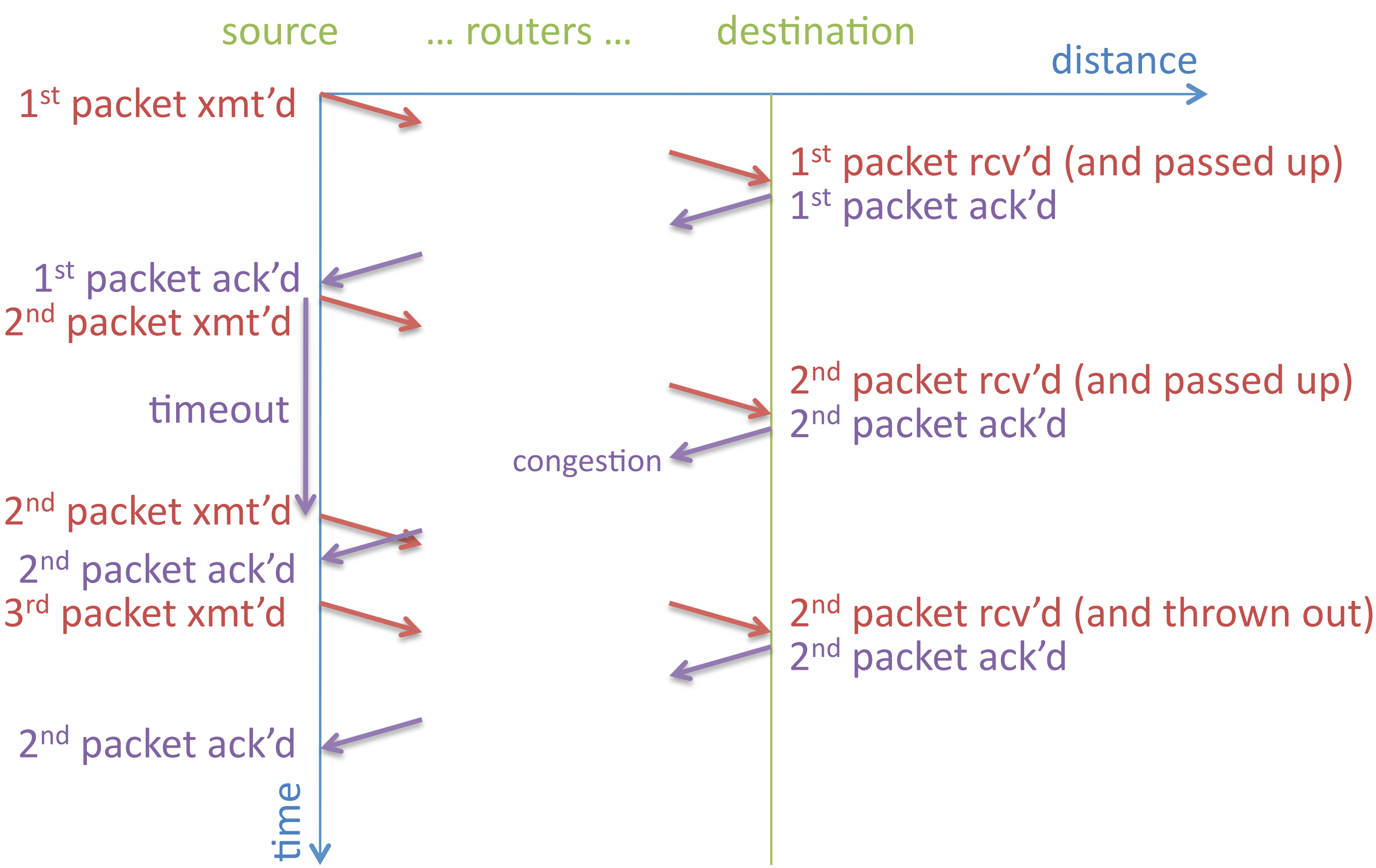
Flow Control Window

- Goals:
  - Reliable
  - Connection-oriented
  - Flow control
- Method:
  - Source allowed to transmit only *W* packets at a time
  - Must wait for ACKs of previous packets before transmitting more packets (flow control)
    - Packets are numbered to detect loss
    - Packets not ACK'd within *timeout* are retransmitted

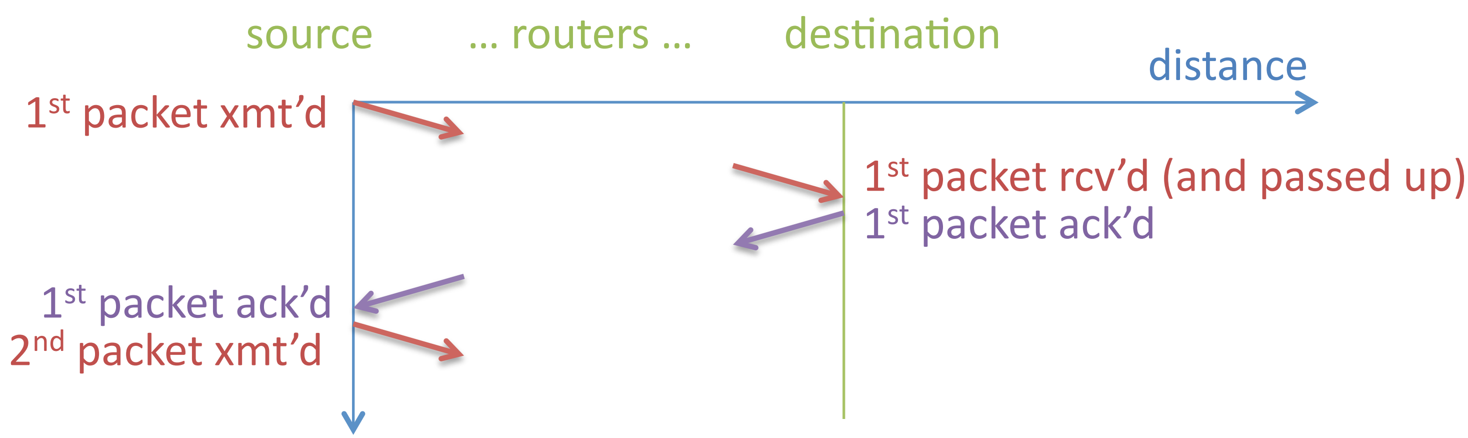
Window flow control



Alternating Bit Protocol (*W*=1)



Efficiency of ABP?



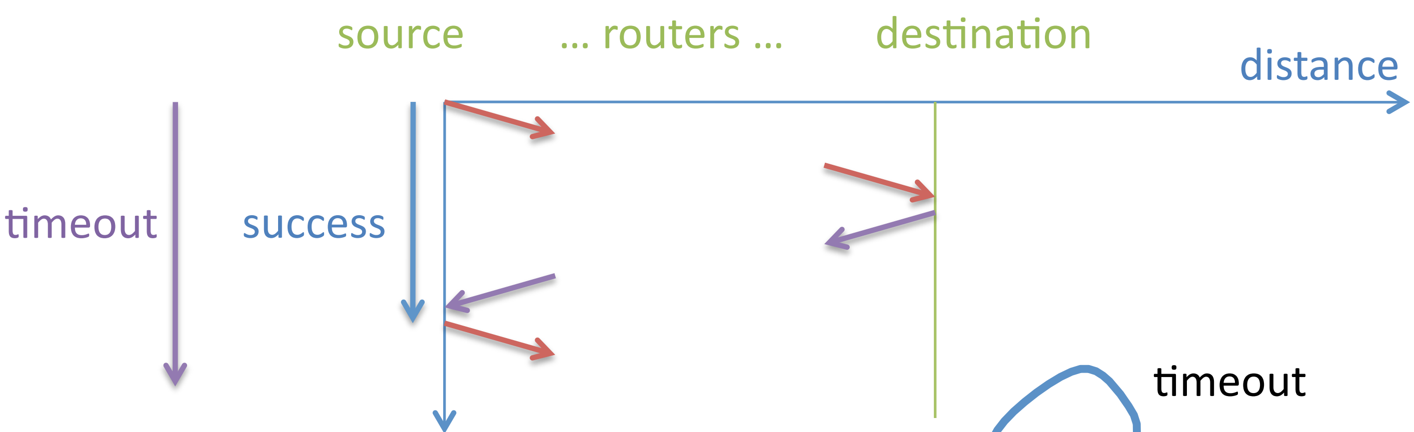
η = proportion of time used for successful transmissions

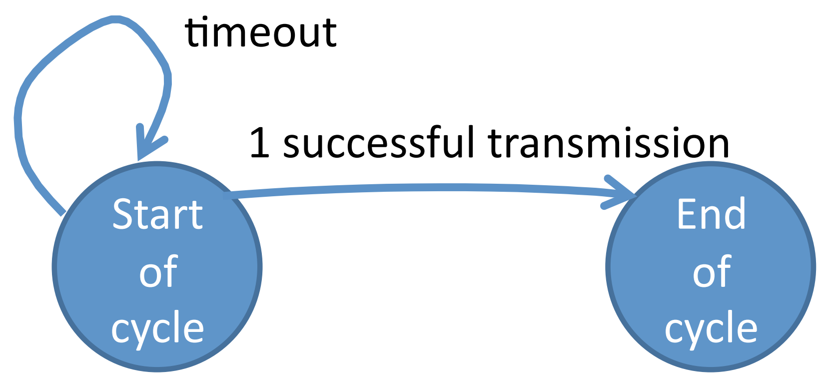
$= T_{trans} / (\mathrm{E}X)$

where EX = average time to successfully transmit a packet

EX = ?

Efficiency of ABP





EX = ?

T = timeout

$S = T_{pck\ trans} + 2T_{prop} + T_{ack\ trans}$

p = P(pck & ack both rcv'd)

$EX = p(?) + (1-p)(?)$

$EX = p(S) + (1-p)(T+EX)$

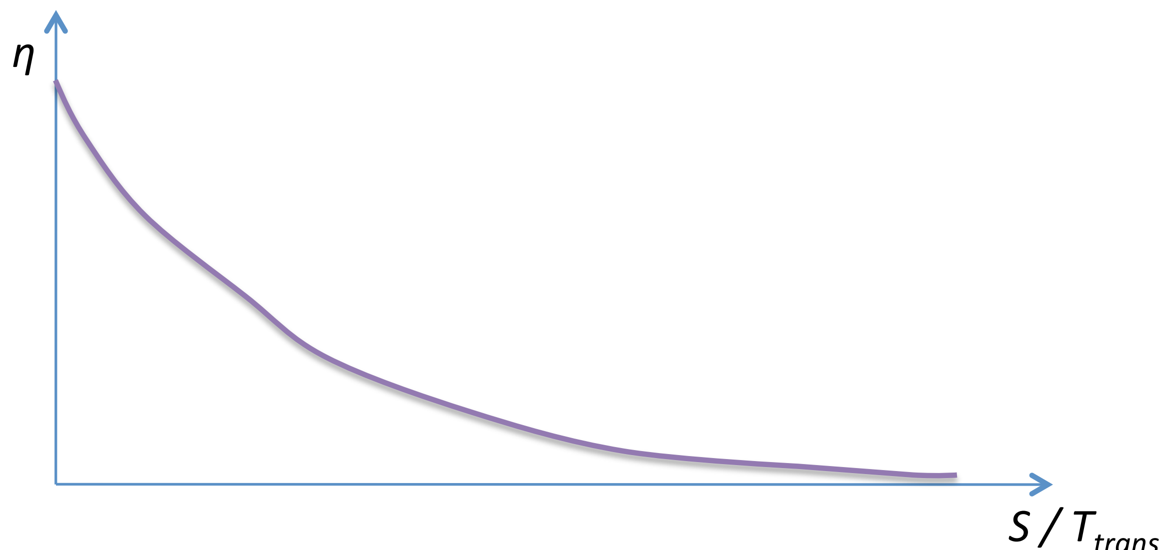
So $EX = S + T(1-p)/p$

Efficiency of ABP

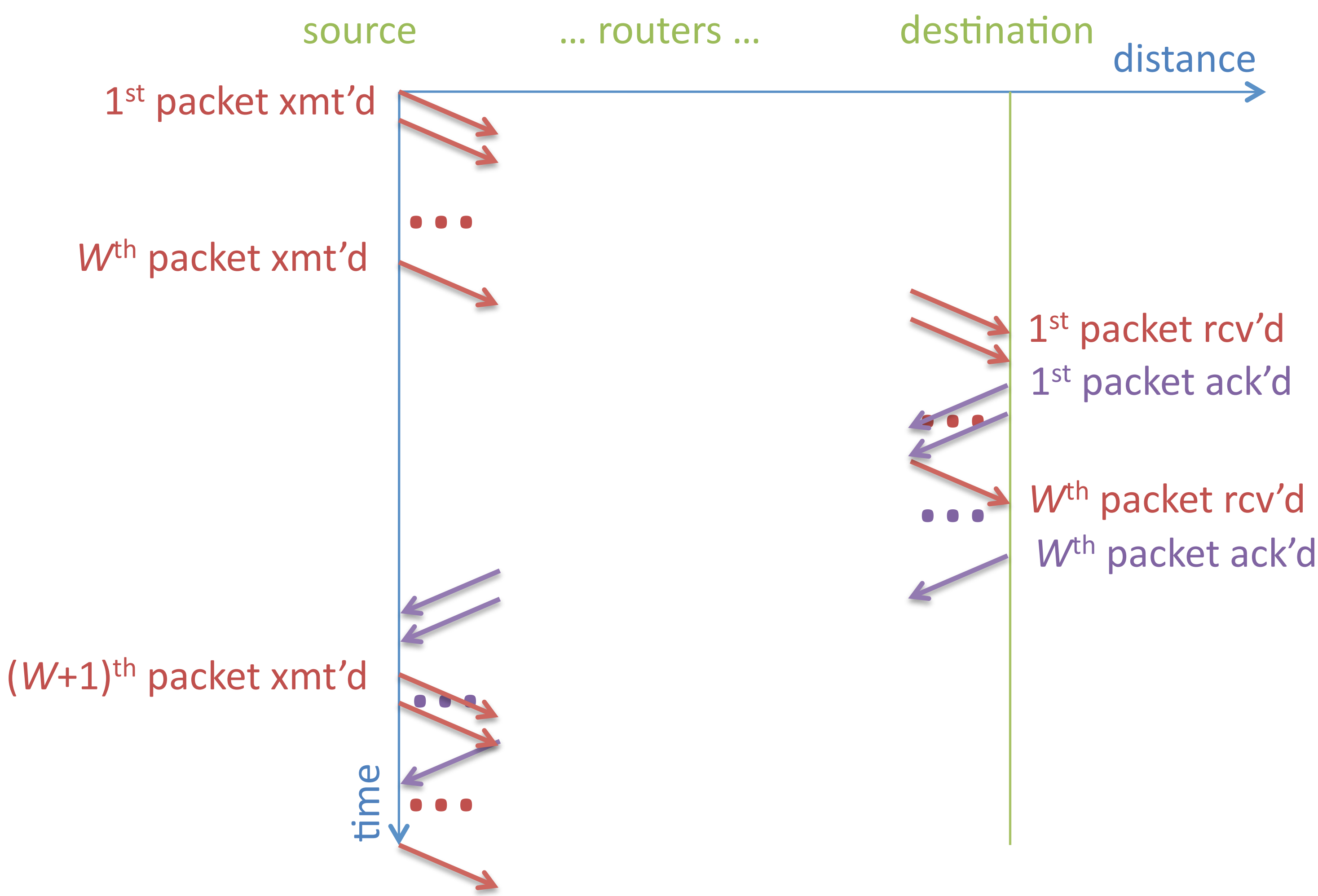
$$EX = S + T(1-p)/p$$

$$\eta = T_{trans} / (EX)$$

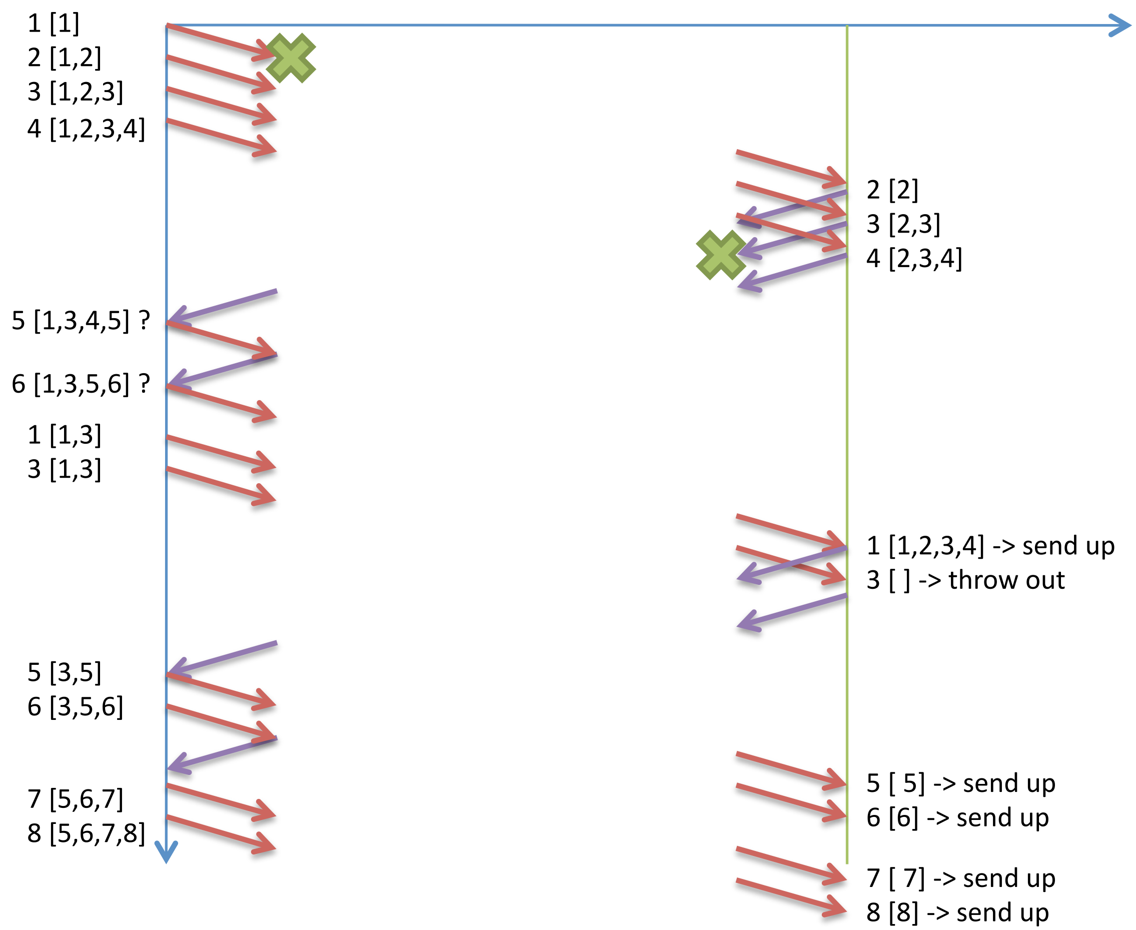
$$= T_{trans} / [S + T(1-p)/p]$$



Selective Repeat Protocol (W>1, fixed)



Selective Repeat Protocol (example with W=4)



Internet transport layer

Two options at Transport Layer:

- User Datagram Protocol (UDP)
  - No flow control (at transport layer)
  - No congestion control (at transport layer)
  - i.e. no transport layer mechanism to slow down source
- Transmission Control Protocol (TCP)
  - Provides flow control at transport layer
  - Provides congestion control at transport layer
  - i.e. source may be slowed down due to destination speed or due to congestion in routers

UDP flow and congestion control

No flow or congestion control at transport layer

- No UDP flow control:
  - Source *application layer* must implement its own flow control
    - using whatever information that the destination application layer sends to it
- No UDP congestion control
  - Source *application layer* should implement its own congestion control
    - *if* it wants to be nice to the network

UDP packet acks

No transport layer packet acknowledgements

- Destination UDP doesn't tell source UDP if it received a packet ("unreliable")
  - Since source UDP doesn't know when packets are dropped, it doesn't retransmit dropped packets
  - Destination *application layer* responsible for acknowledgements or other feedback to source, if desired
- Destination UDP doesn't put packets back into transmitted order ("connectionless")
  - Destination *application layer* responsible for packet reordering

TCP flow and congestion control

Both flow and congestion control at transport layer

- Flow control:
  - Destination TCP signals when it is ready to accept more packets
- Congestion control
  - Source TCP speeds up or slows down based on estimate of congestion

TCP packet acks

Transport layer issues packet acknowledgements

- Destination TCP tells source TCP if it received a packet (and which packet) using acknowledgements (ACKs) ("reliable")
  - Source TCP retransmits packets which are not ACK'd within a *timeout*
- Destination TCP reorders packets into their original order ("connection-oriented")
  - Before handing them to destination application layer

TCP flow and congestion control

Use ACKs from destination TCP to:

- Know when destination is ready to accept more packets (flow control)
  - When the source receives an ACK, it knows that destination has processed that packet
  - Specific field in reverse direction packets' headers
- Estimate congestion (congestion control)
  - The rate at which the source receives ACKs tells it something about the delay between the source and destination

Congestion control

- Problem: routers in between source and destination may experience congestion
  - Routers may queue many packets, resulting in queuing delay
  - Routers may drop packets, resulting in packet loss
- Solution: slow down rates of sources causing congestion
  - Called "congestion control"

Window-based congestion control

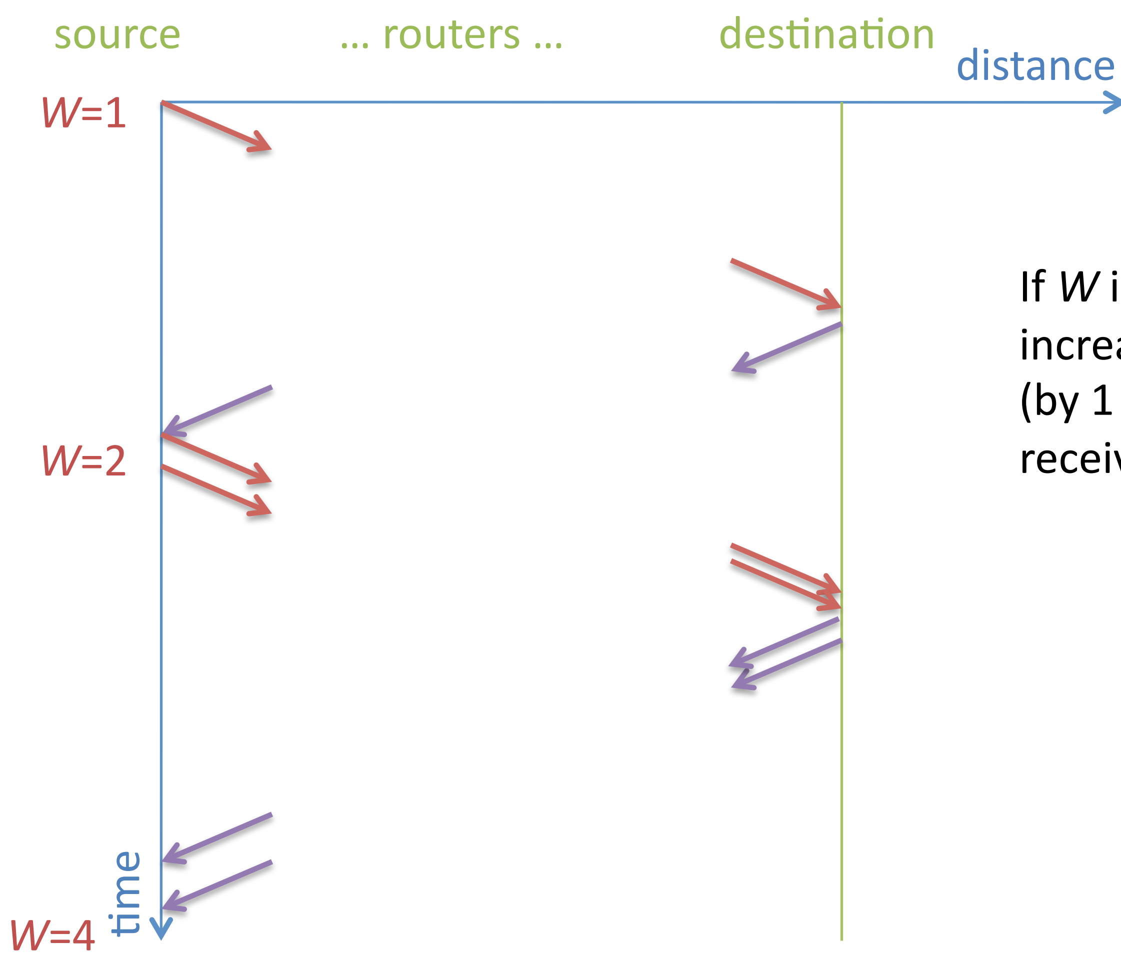
- Goals:
  - Reliable
  - Connection-oriented
  - Flow control
  - Congestion control
- Method:
  - Source allowed to transmit only *W* packets at a time
  - Must wait for ACKs of previous packets before transmitting more packets (flow control)
    - Packets are numbered to detect loss
    - Packets not ACK'd within *timeout* are retransmitted
  - Window size *W* is modified based on estimate of congestion (congestion control)

TCP-Tahoe

Algorithm for *W*:

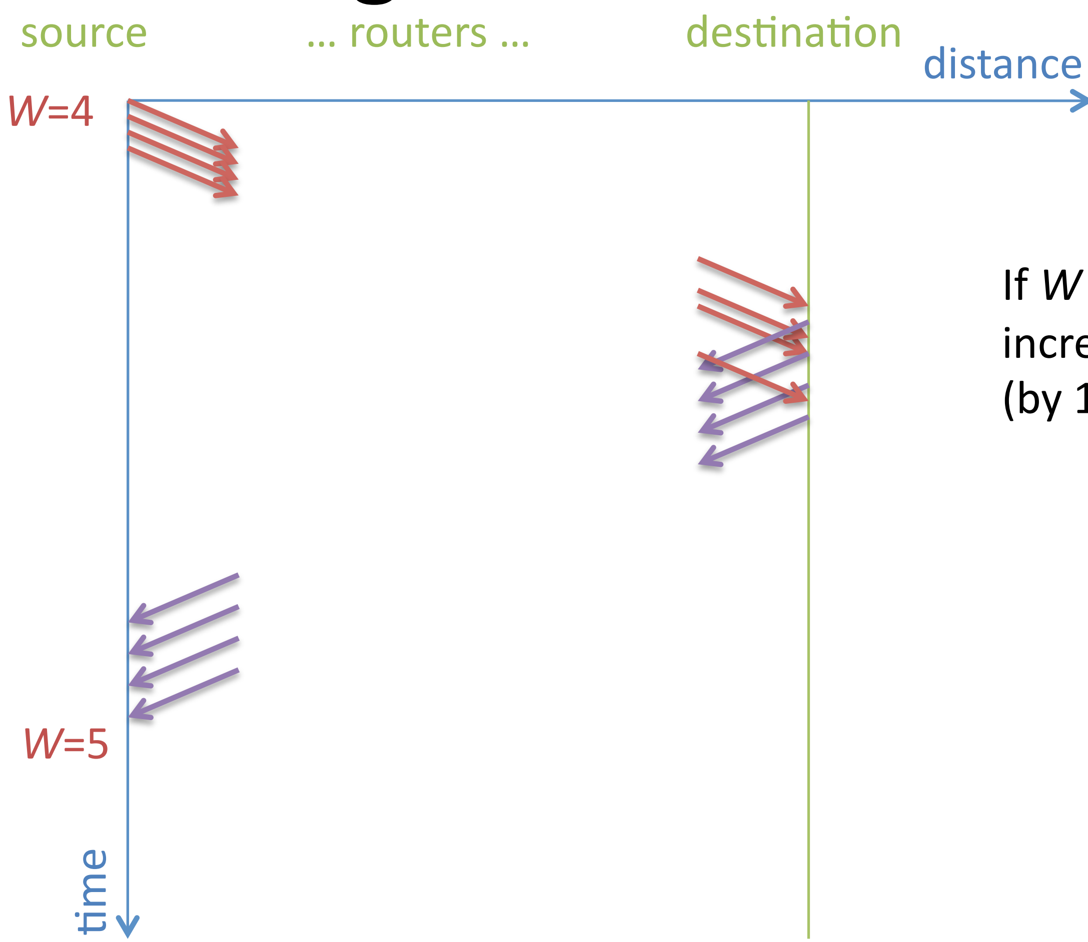
- Start timidly with *W*=1.
- Increase *W* when no congestion
  - "Slow Start": If *W* is small, increase it quickly (by 1 for each received ACK)
  - "Congestion Avoidance": If *W* is large, increase it slowly (by 1 each RTT)
  - There is an upper limit on *W*
- Decrease *W* when congestion
  - If there is a timeout, go back to *W*=1

Increasing *W* when it is small (slow start)



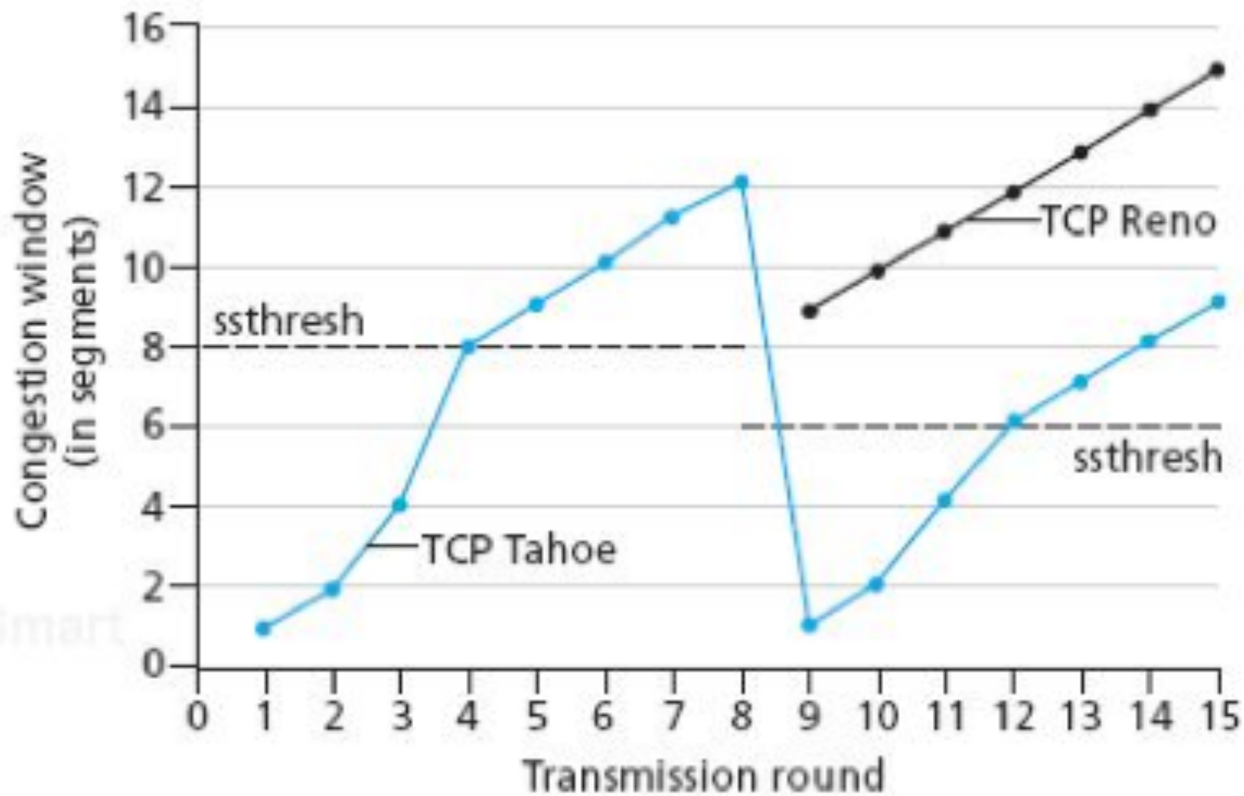
If *W* is small, increase it quickly (by 1 for each received ACK)

Increasing *W* when it is large
Congestion control

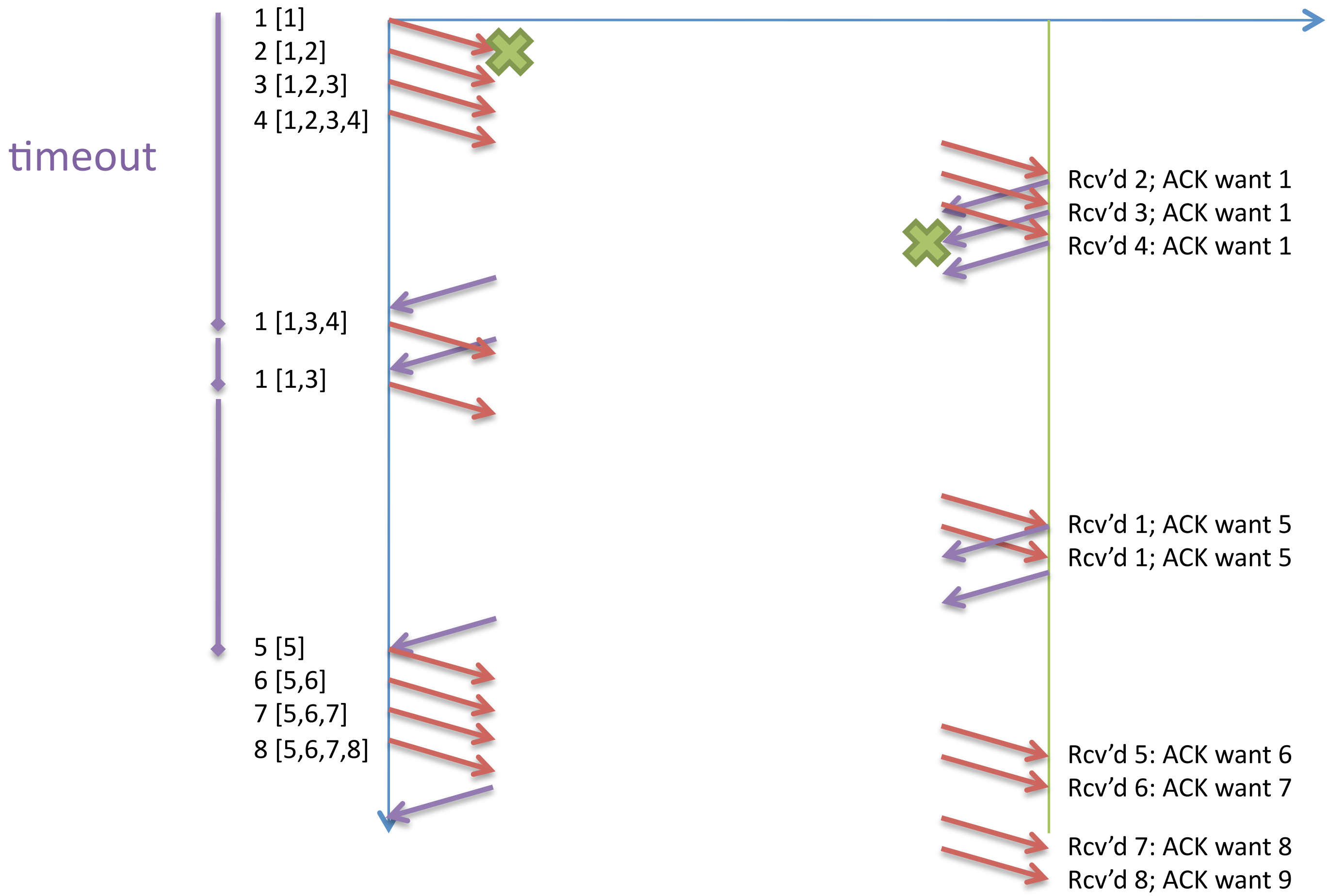


If *W* is large, increase it slowly (by 1 each RTT)

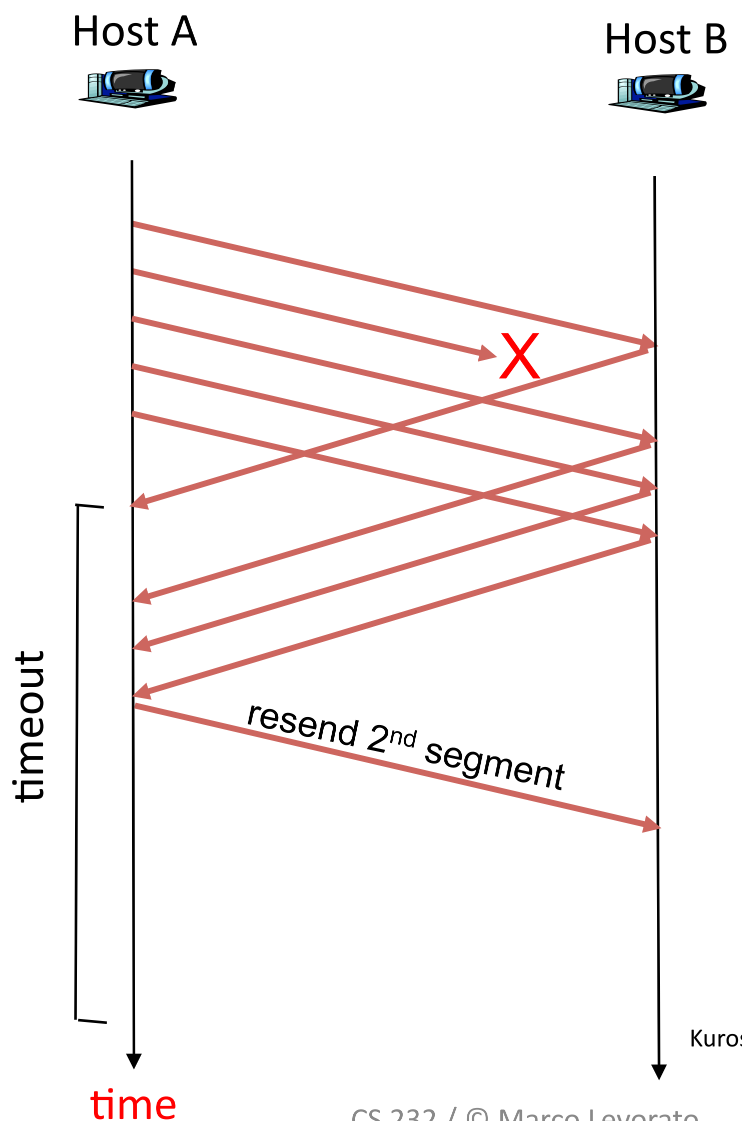
TCP congestion control



Cumulative ACK

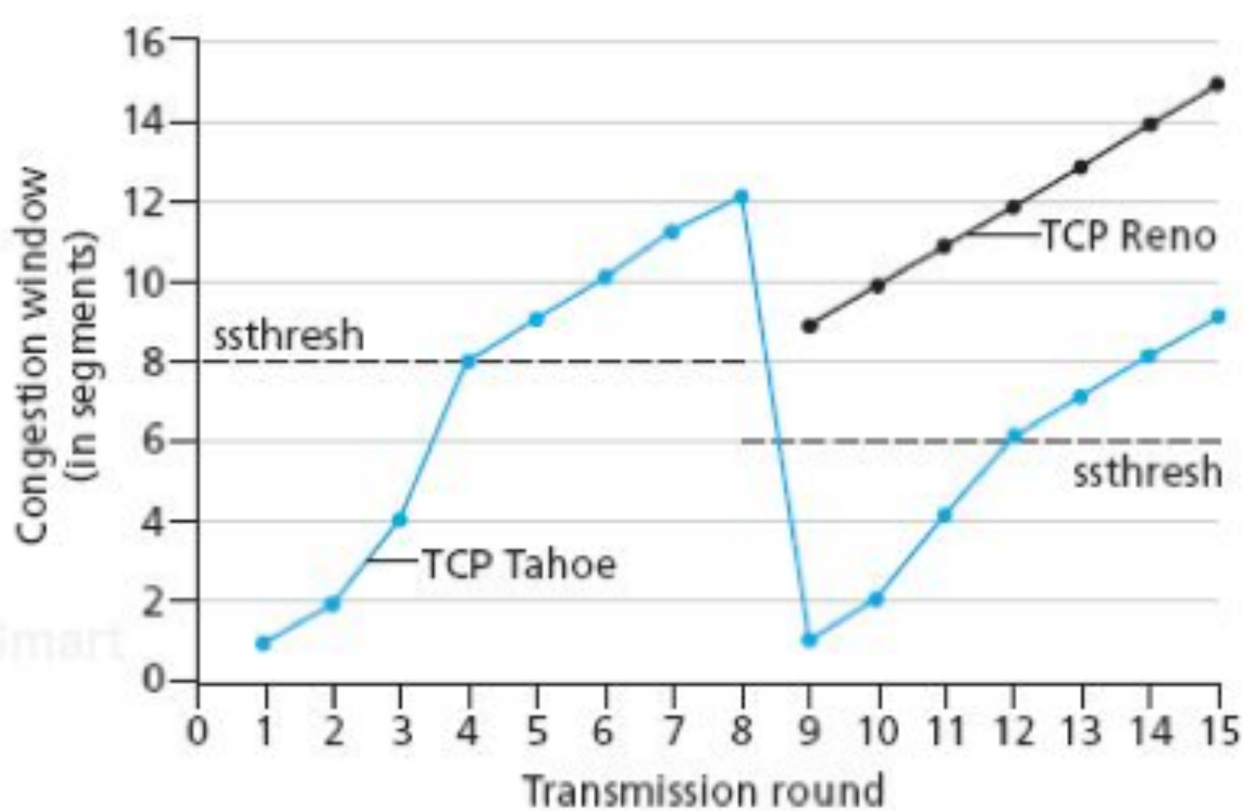


TCP: fast retransmit

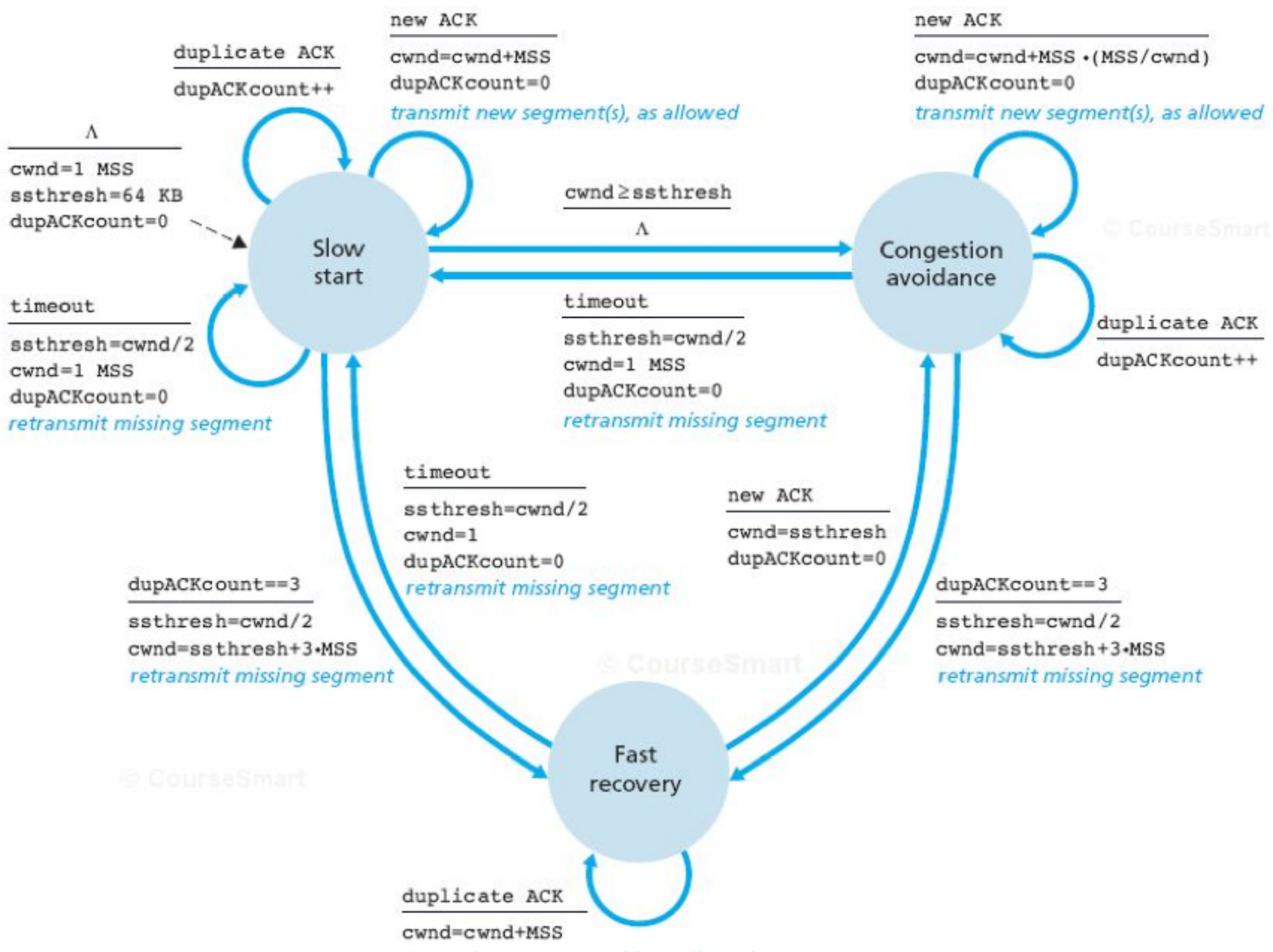


Kurose fig. 3.37

TCP-Reno Fast Recovery



Finite State Machine



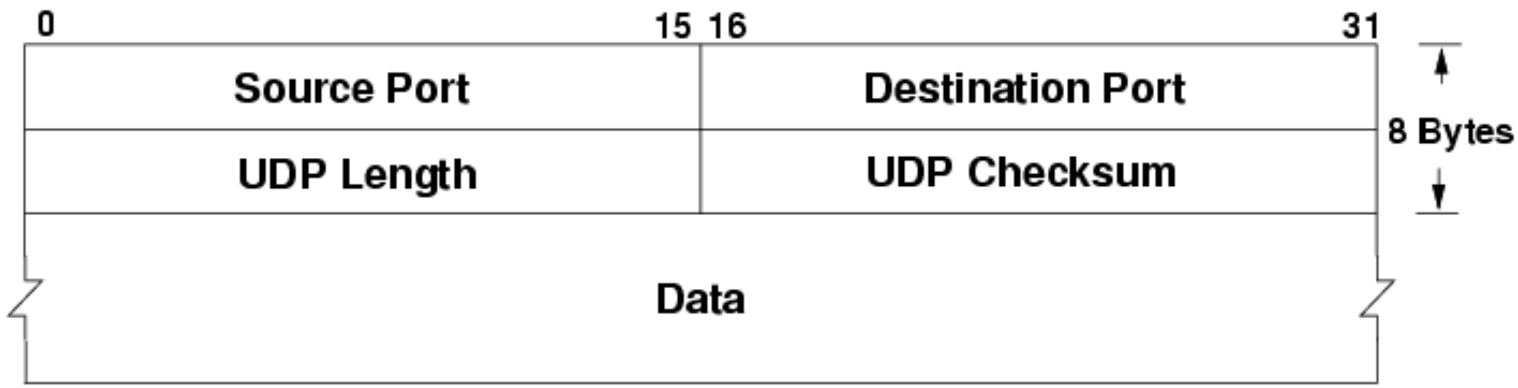
TCP: additional details

- Actual version:
  - uses “segments” and byte #s instead of pkt #s
  - keeps a timer only for the oldest segment
  - dynamically adjusts the threshold between slow start and congestion avoidance
  - dynamically adjusts timeout value
  - has additional rules regarding what to ACK when
  - “TCP-NewReno” has additional tweaks
  - there are many other variants …

UDP vs. TCP

- TCP used by:
  - Applications that want packet retransmission & packet reordering
  - And are willing to put up with congestion control
  - e.g. email, http, ftp
- UDP used by:
  - Applications that are not willing to put up with congestion control
  - And applications that don't need packet retransmission & packet reordering
  - e.g. streaming, voice over IP (VoIP), video conferencing, gaming

UDP header

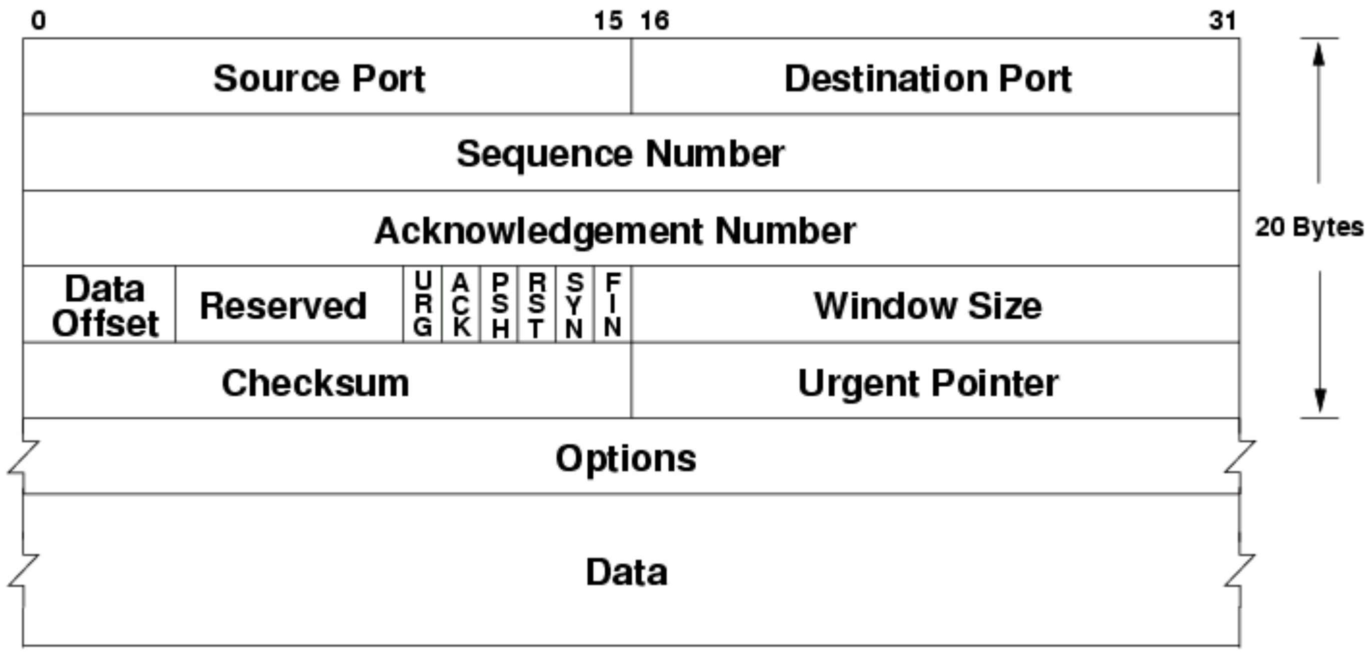


Wikipedia Commons

- ports
- src & dest IP addresses in IP header
- no packet numbers

TCP header

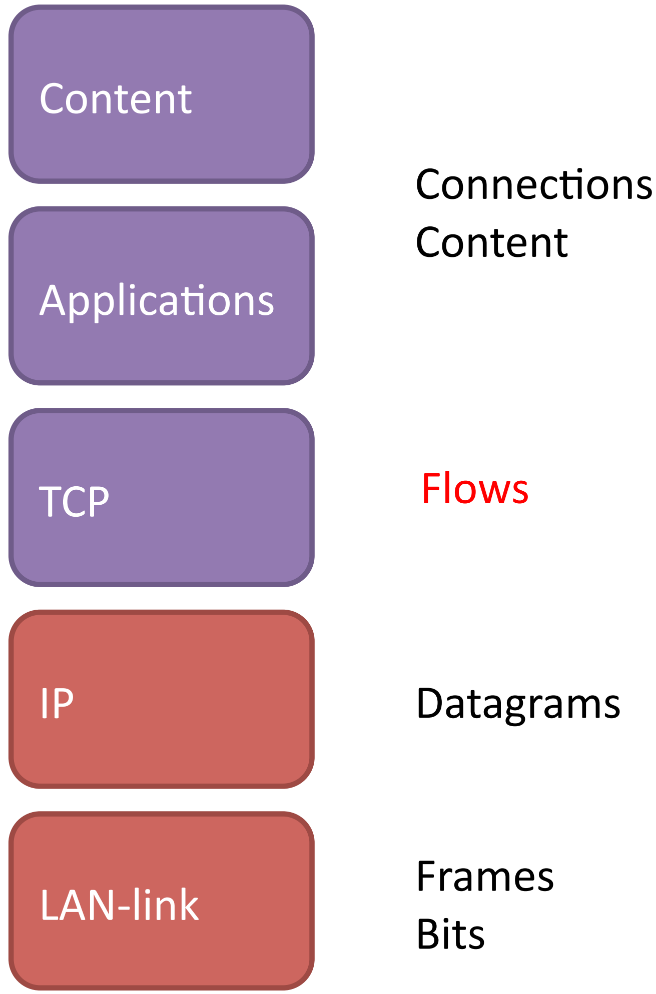
Wikipedia Commons



Wikipedia Commons

- ports
- sequence & ACK numbers for retransmission & ordering
- window size for flow & congestion control

Traffic characterization by layer



Transport layer models

- Flow models
  - Multiple flows
  - Graph of
    - end points
    - links & nodes:
      - all links & nodes
      - abstracted set of links & nodes
      - or just bottleneck links / nodes

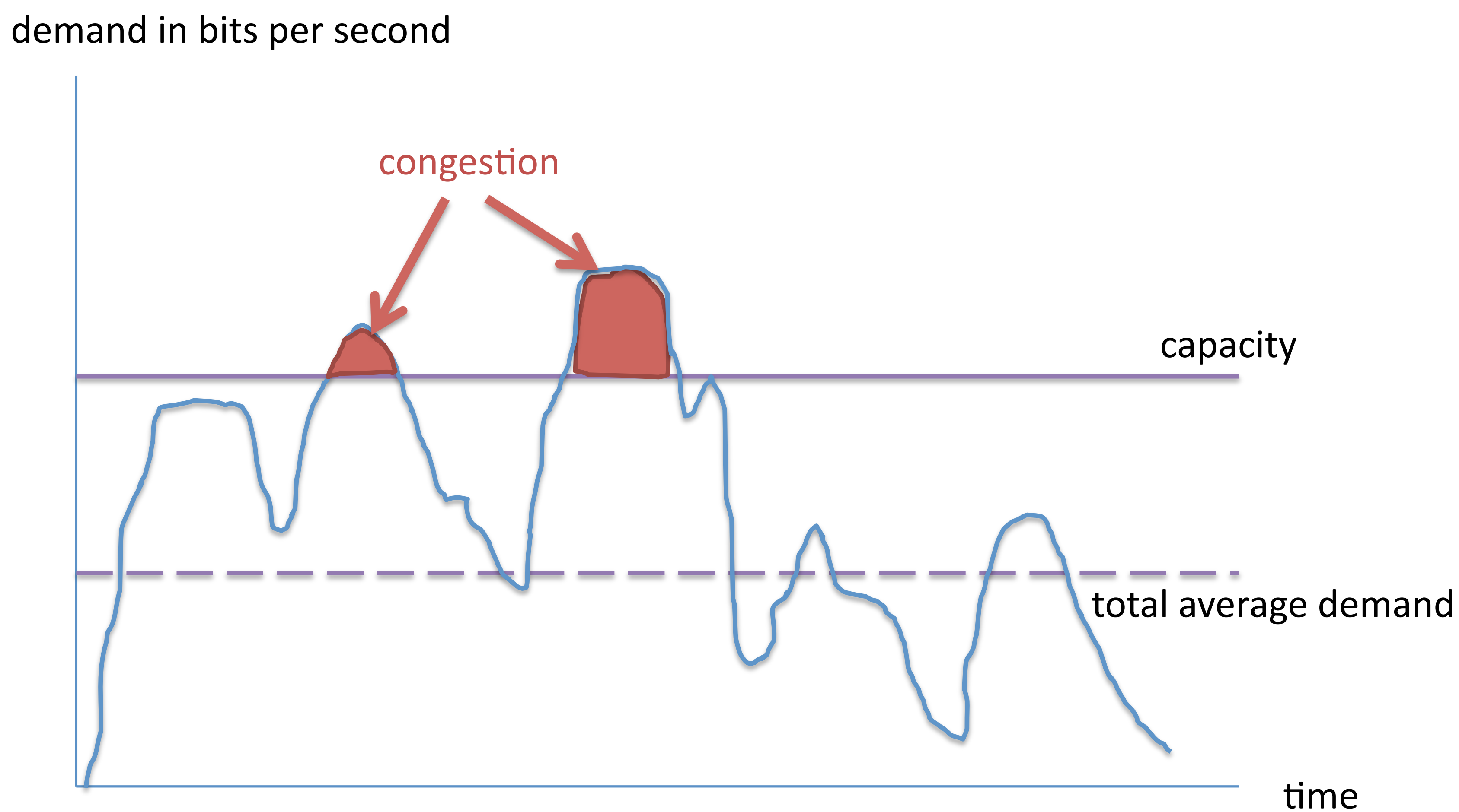
Transport layer models

- Flow initiation & termination
  – by application layer traffic model
  – or considered under a fixed number of connections

Transport layer models

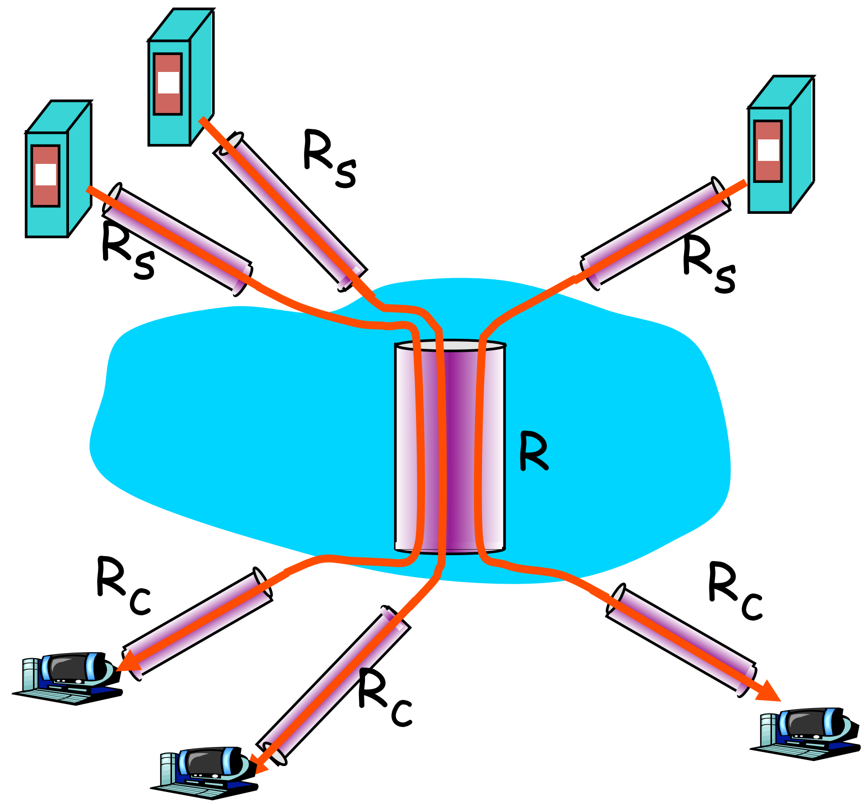
- Flow rate
  - Dynamic capacity of links & nodes given by lower layer models
  - Or may be abstracted by an independent model, e.g. flow restrictions & delay
  - Plus model of window or rate flow & congestion control

Congestion


demand in bits per second
congestion
capacity
total average demand
time

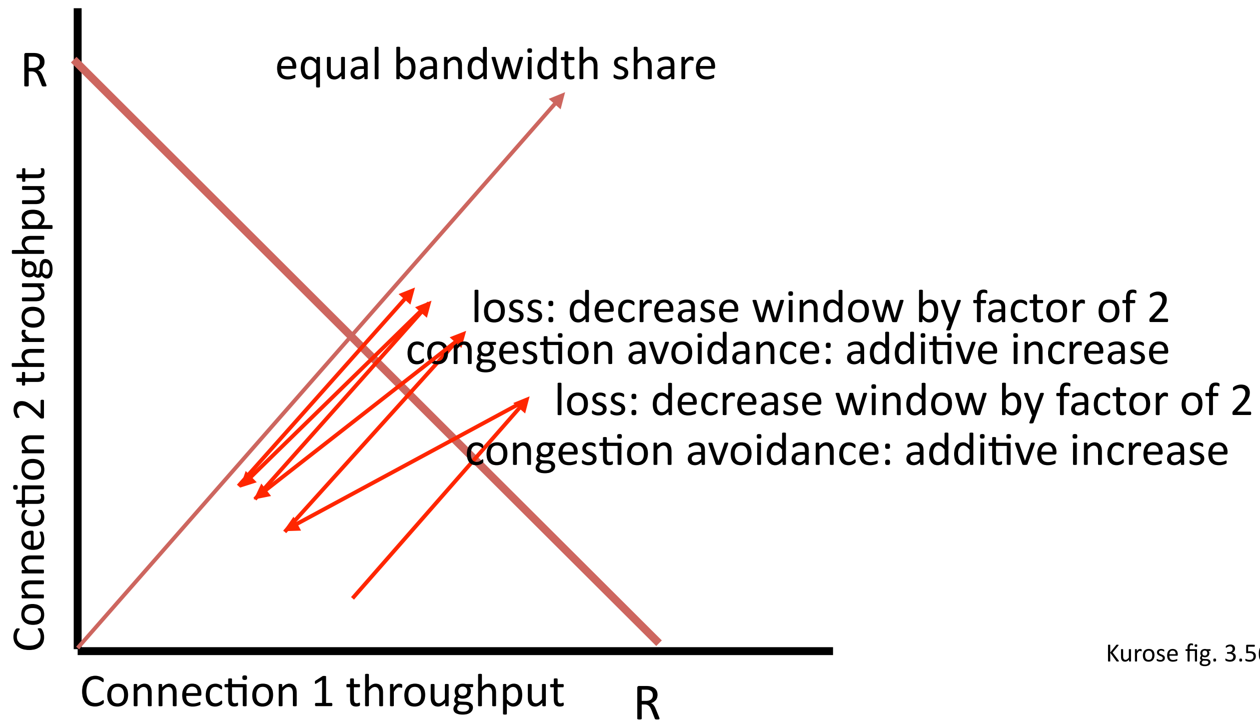
Multiple flow model

- What is the throughput on a single connection?
  - Min(R_s, $R/3$, R_c)



Kurose fig. 1.20

Multiple flow model

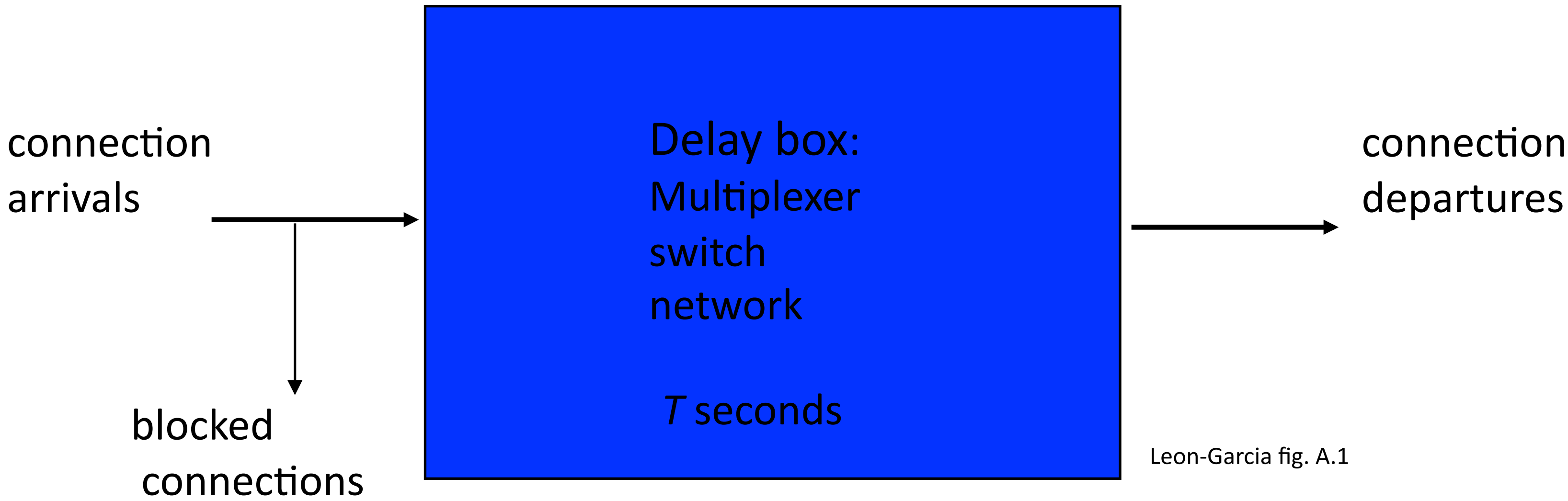


Kurose fig. 3.56

Transport layer model examples

- TCP flow model
  - TCP protocol
  - Model of pertinent events, e.g. timeouts
    - either from lower layer traffic models
    - or abstracted
  - e.g. for http, email, some streaming
- UDP flow model
  - Model of pertinent events, e.g. packet losses
    - either from lower layer traffic models
    - or abstracted
  - e.g. for VoIP, some streaming

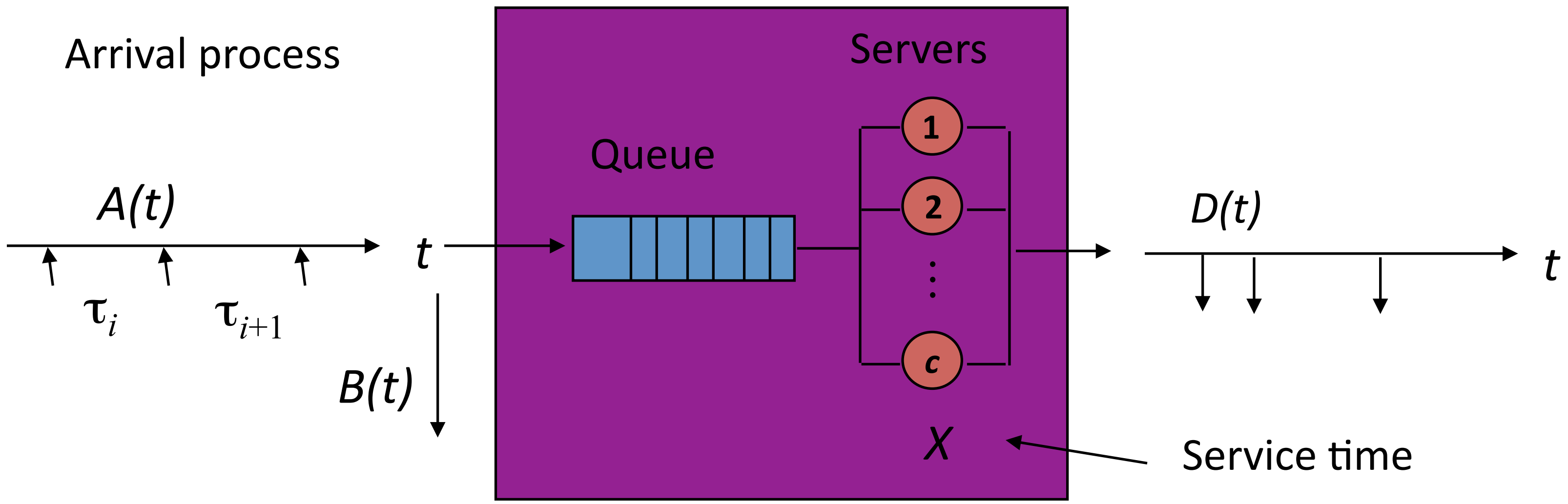
Connection models



Leon-Garcia fig. A.1

- Arrivals
- Duration
- Connection access control

Connection Queuing Model



Leon-Garcia fig. A.6