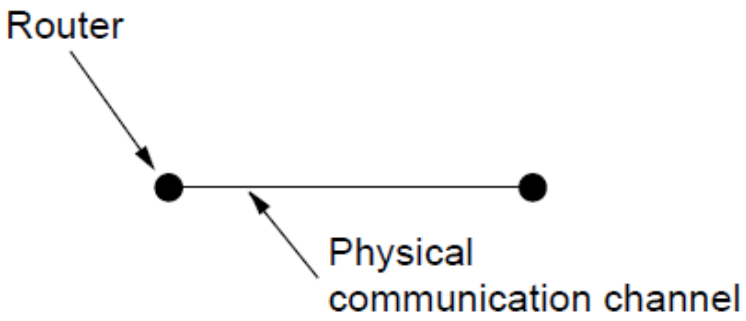
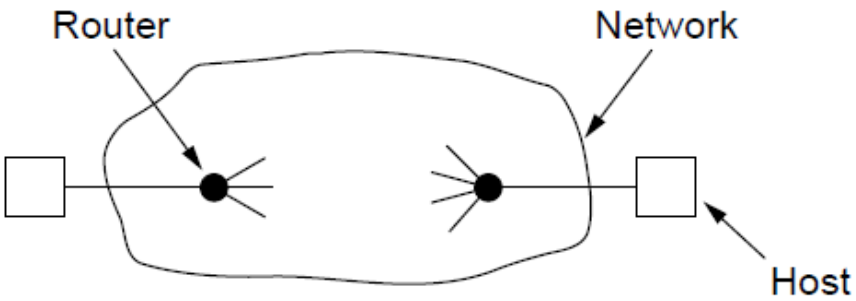
Error control & flow control over a layer 4 connection



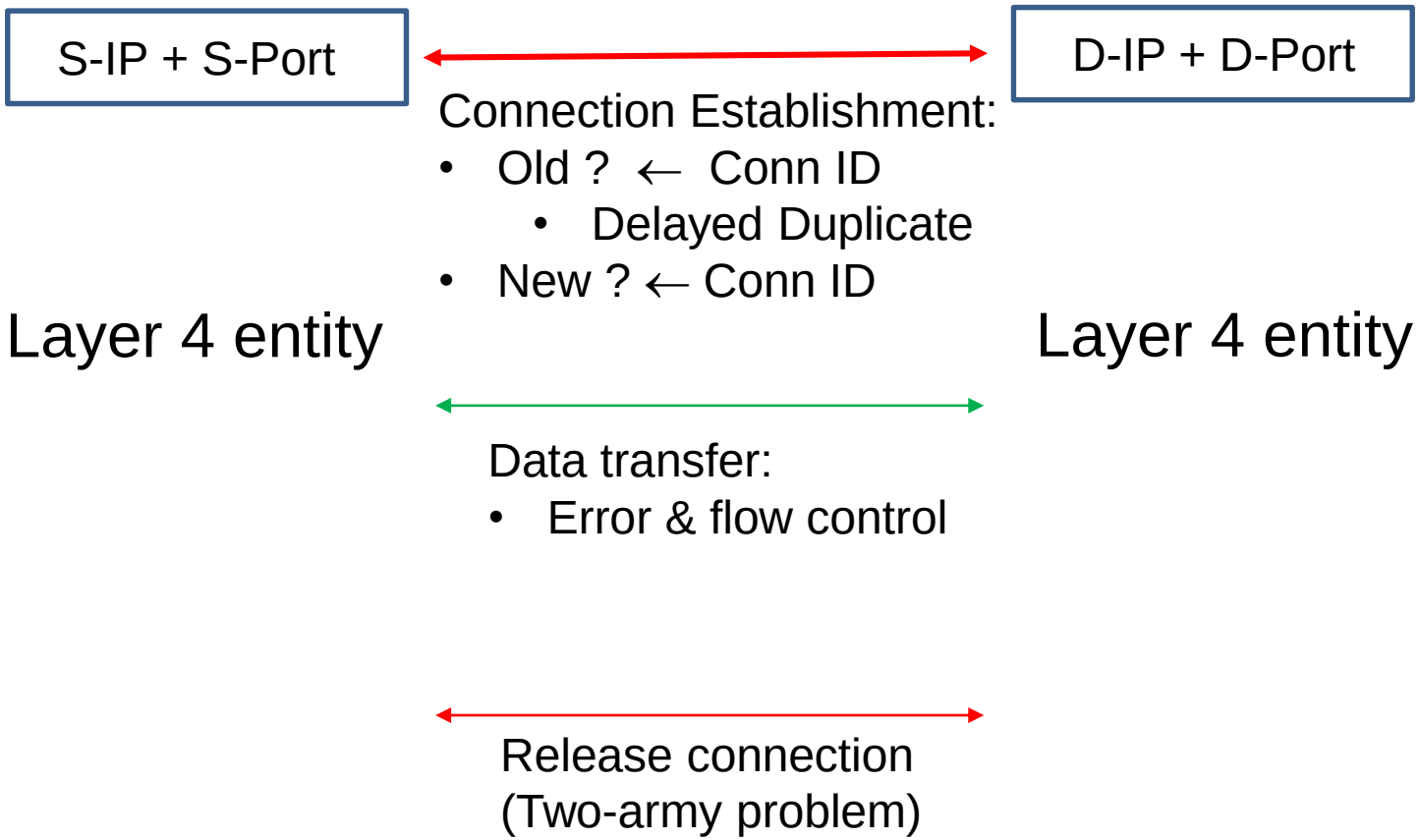
(a)



(b)

(a) Environment of the data link layer.

(b) Environment of the transport layer.

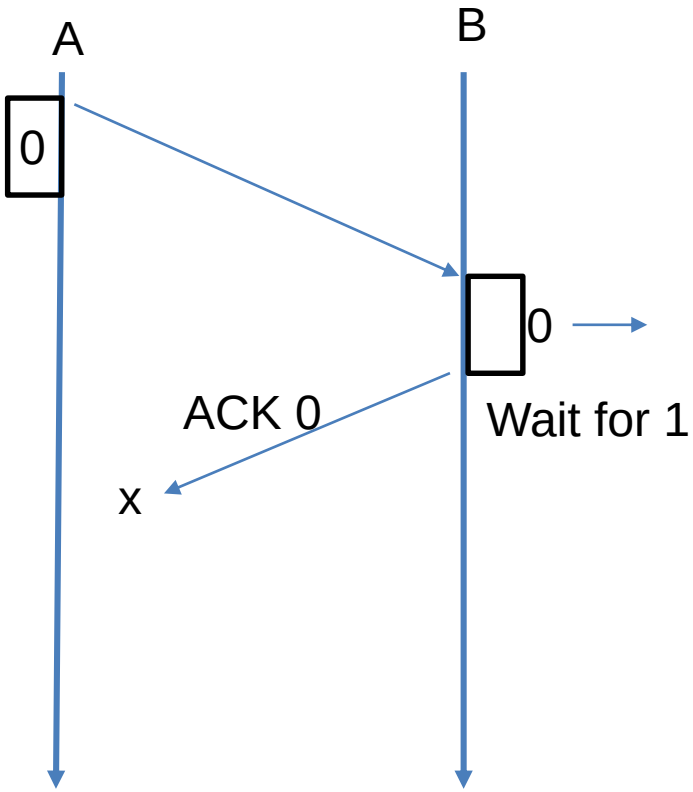

S-IP + S-Port
D-IP + D-Port
Connection Establishment:
• Old ? ← Conn ID
• Delayed Duplicate
• New ? ← Conn ID
Layer 4 entity
Layer 4 entity
Data transfer:
• Error & flow control
Release connection
(Two-army problem)

Error and flow control at layer II

1. A frame carries an error detecting code (e.g., a CRC or checksum) that is used to check if the information was correctly received.
2. A frame carries a sequence number to identify itself and is retransmitted by the sender until it receives an acknowledgement of successful receipt from the receiver, called ARQ(Automatic Repeat reQuest).
3. There is a maximum number of frames that the sender will allow to be outstanding at any time, pausing if the receiver is not acknowledging frames quickly enough.
   - Stop-and-wait
   - Go-back-N
   - Select-repeat
4. The sliding window protocol combines these features and is also used to support bidirectional data tranfer.

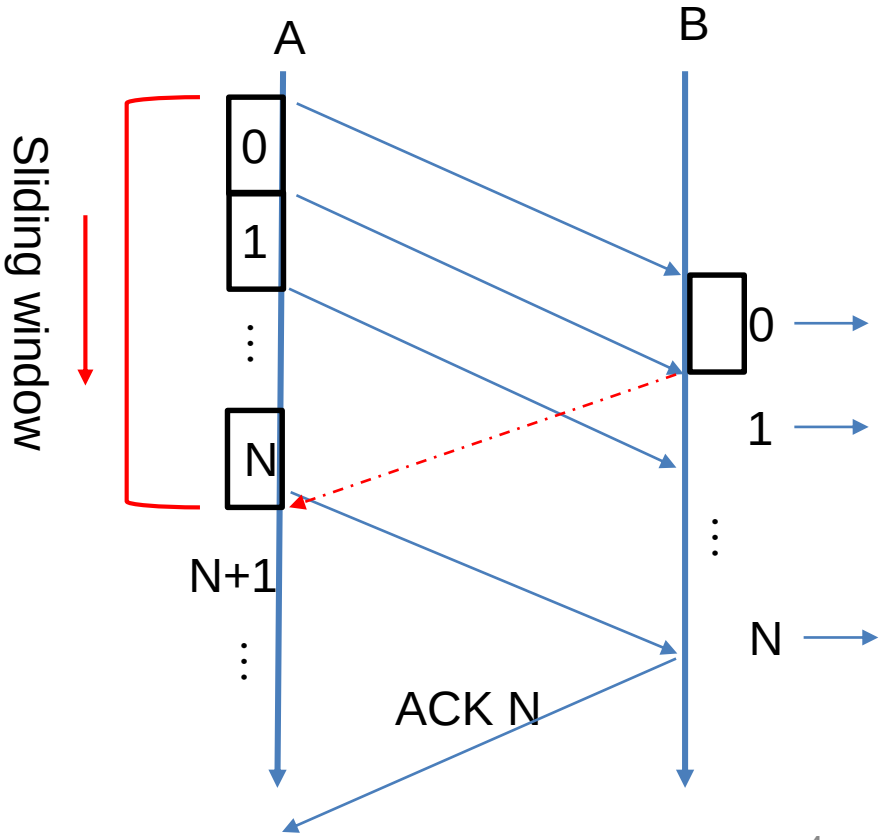
Stop-and-Wait ARQ,

- Require buffer space for one frame only (Receive Window size=1).
- Need only one bit, 0 or 1, for sequence number.



Go-back-N ARQ, with N smaller than the number of sequence numbers

- Receiver requires buffer space for one frame (Receive window size=1).



Cont.

- The basic similarity in the data link layer and the transport layer is that a sliding window or other scheme is needed on each connection to keep a fast transmitter from overrunning a slow receiver.
  - Window size ~ Bandwidth-delay product
- The main difference is that a router usually has a relatively few lines whereas a host may have numerous connections.

  This difference makes it impractical to implement the data link buffering strategy in the transport layer.

Cont.

- If the receiver knows that the sender buffers all segments until they are acknowledged, the receiver may or may not dedicate specific buffers to specific connections.
  - The receiver may maintain a single buffer pool shared by all connections.
  - When a segment comes in, an attempt is made to dynamically acquire a new buffer.
    - If one is available, the segment is accepted; otherwise, it is discarded.
- If the network service is unreliable, or if the receiver can not guarantee that every incoming segment will be accepted, the sender must buffer all segments sent.
  - In the latter case, the sender can not trust the network layer's acknowledgement, because *the acknowledgement means only that the segment arrived, not that it was accepted.*

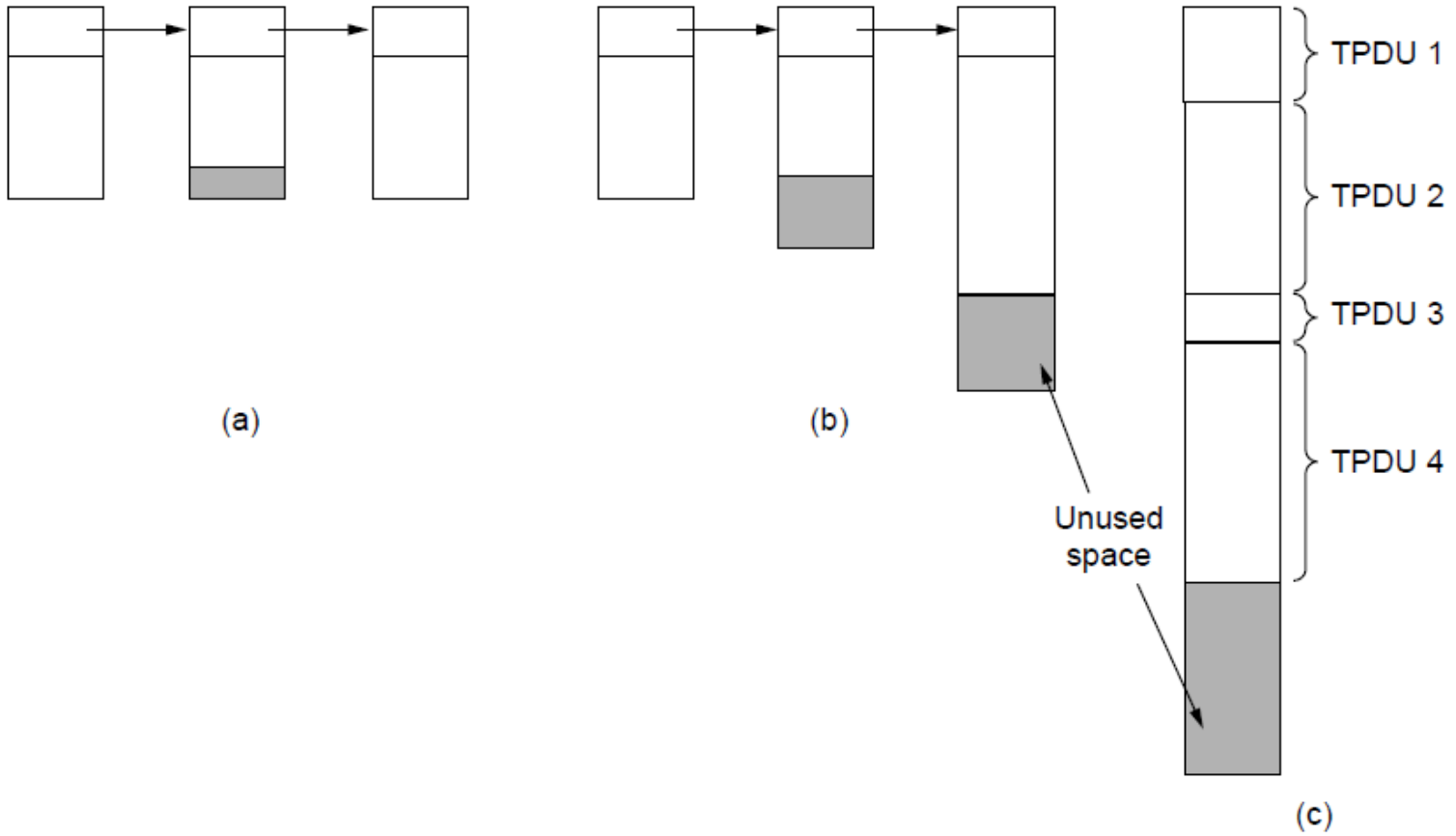
Cont.

- The optimal trade-off between source-buffering and destination-buffering depends on the type of traffic carried by the connection.
  - For low-bandwidth bursty traffic,
    - It is better not to dedicate any buffers, but rather to acquire them dynamically at both end.
    - It is better to buffer at the sender.
      - The sender must retain a copy of the segment until it is acknowledged.
  - For high-bandwidth smooth traffic, it is better to buffer at the receiver.
    - To allow the data to flow at maximum speed.

Cont.

- With reliable network service, other trade-offs become possible.
  - If the sender knows that the receiver always have buffer space, it need not retain copies of the segments it sends.
- On the buffer size at the receiver.
  - Fixed size buffers:
    - If most segments are nearly the same size, it is natural to organize the buffers as a pool of identical size buffers, with one segment per buffer.
    - If there is a wide variation in segment size: If the buffer is chosen equal to the largest possible segment, space will be wasted whenever a short segment arrives. If the buffer size is chosen less than the maximum segment size, multiple buffers will be needed for long segments, with the attendant complexity.
  - Variable-size buffers:
    - It gets better memory utilization, at the price of more complicated buffer management.
  - A single large circular buffer per connection:
    - It makes good use of memory, provided that all connections are heavily loaded but is poor if some connections are lightly loaded.

Buffer size problems



(a) Chained fixed-size buffers. (b) Chained variable-sized buffers. (c) One large circular buffer per connection.

Cont.

- Dynamic buffer allocation:
  - As connections are opened and closed, and as traffic pattern changes, the sender and receiver need to dynamically adjust their buffer allocations.
  - Dynamic buffer management means a variable-sized window:
    - Initially, the sender requests a certain number of buffers, based on its perceived need.
    - The receiver then grants as many of these as it can afford.
    - Every time the sender transmits a segment, it must decrement its allocation, stopping altogether when the allocation reaches zero.
    - The receiver then piggybacks both acknowledgements and buffer allocations onto the reverse traffic.

Error Control and Flow Control (2)

	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Dynamic buffer allocation. The arrows show the direction of transmission. An ellipsis (...) indicates a lost TPDU

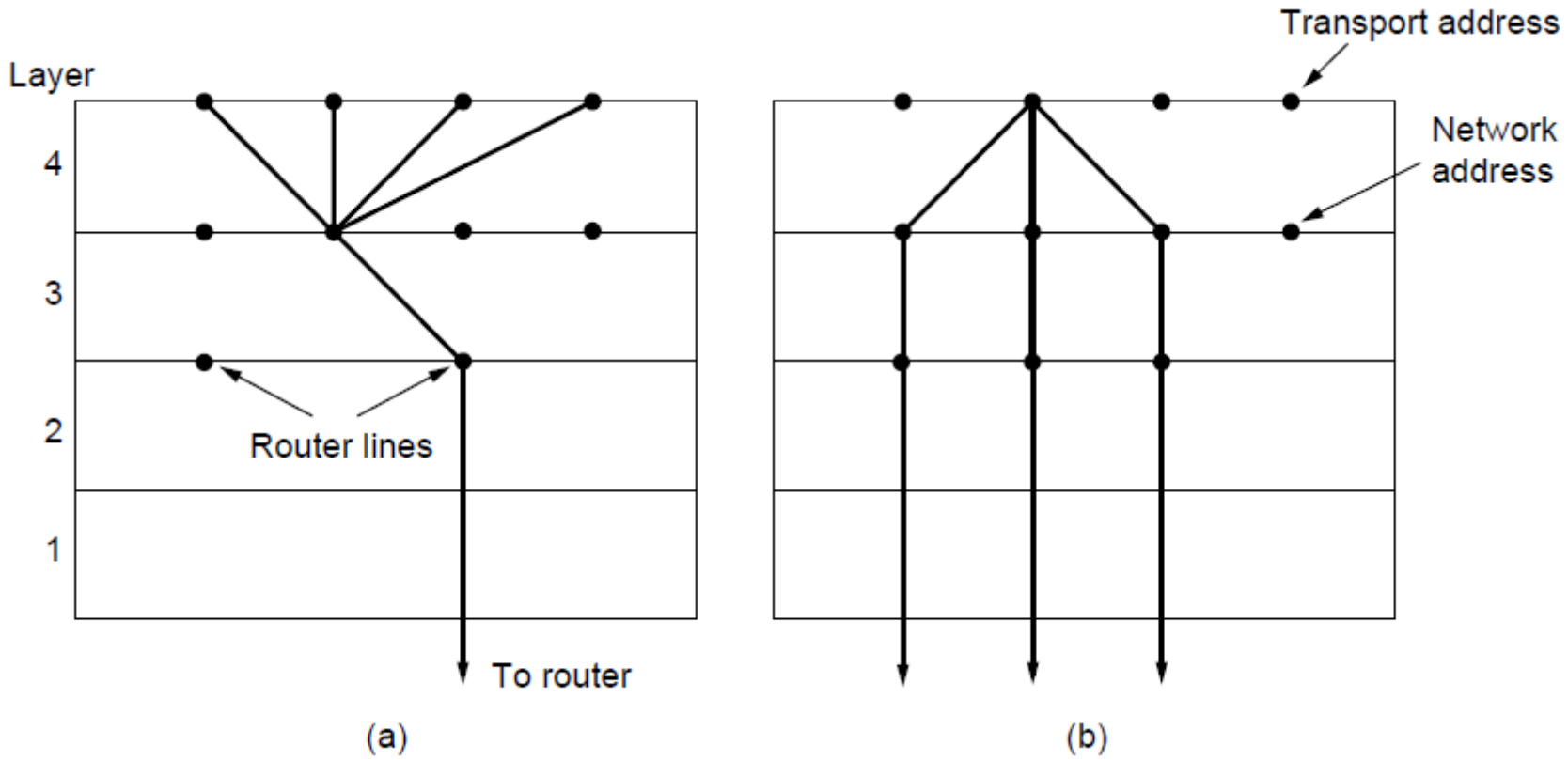
Cont.

- The previous figure shows an example of dynamic buffer allocation with 4-bit sequence numbers.
  - Assume that buffer allocation information travels in separate segments, and is not piggybacked onto reverse traffic.
  - Potential problems can arise in datagram networks if control segments can get lost. Check line 16 in the figure.
    - To prevent the deadlock situation, each host should periodically send control segments giving the acknowledgement and buffer status on each connection.

Cont.

- When buffer space no longer limits the maximum flow, another bottleneck will appear: *the carrying capacity of the subnet.*
  - The flow control mechanism must be applied at the sender to prevent it from having too many unacknowledged segments outstanding at once.
  - A dynamic sliding window implements both flow control and congestion control (by Belsnes):
    - If the network can handle *c* segments/sec and the cycle time (including transmission, propagation, queueing, processing at the receiver, and return of the acknowledgement) is *r*, then the sender window should be *cr*.
    - In order to adjust the window size periodically, the sender could monitor both parameters and then compute the desired window size.

Multiplexing



(a) Multiplexing. (b) Inverse multiplexing.

Cont.

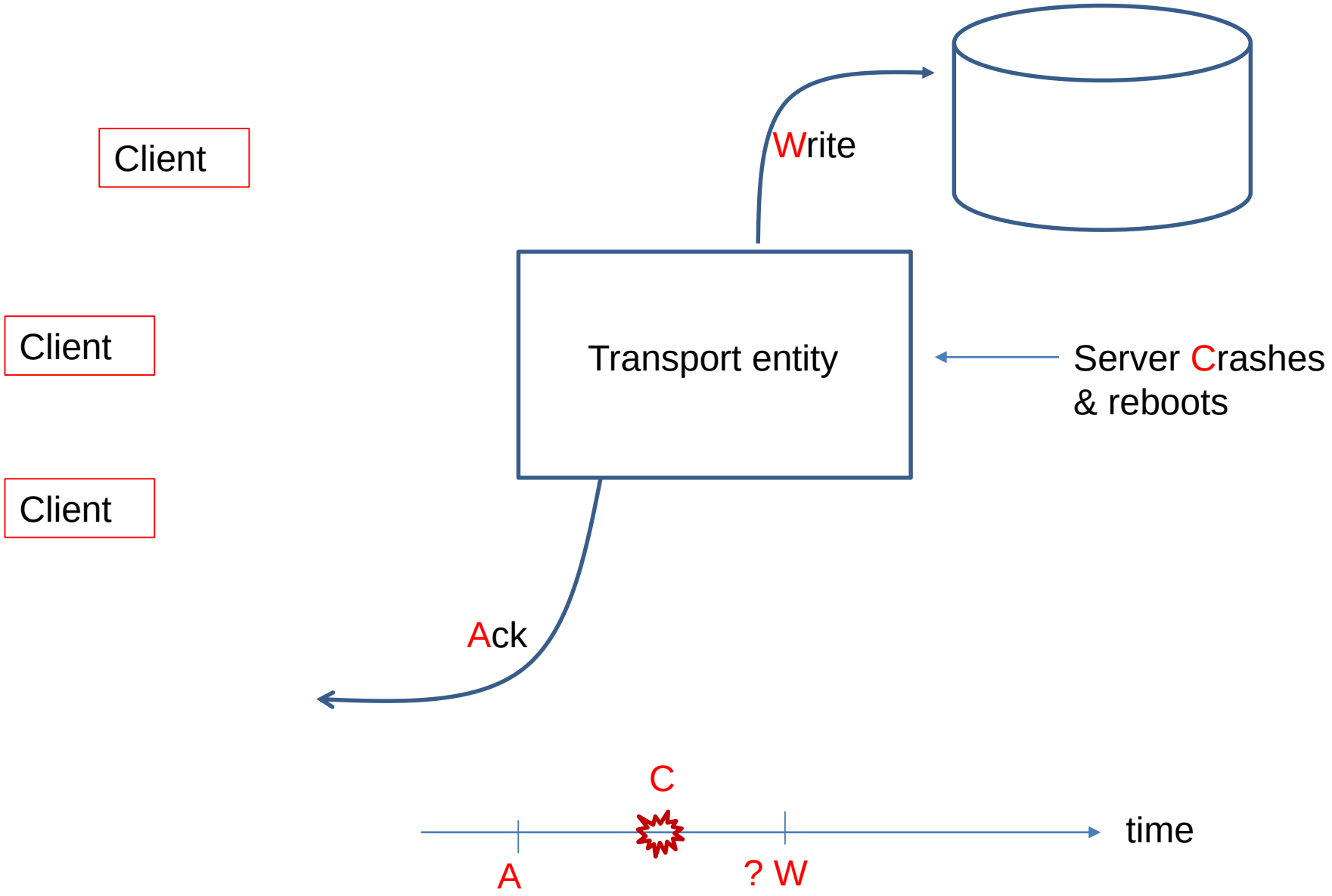
- **Multiplexing** : *different transport connections use the same network connection* .
- **Inverse multiplexing**: The form is to have the transport layer open multiple network connections and distribute the traffic among them on a round-robin basis.
  - With *k* network connections open, the effective bandwidth is increased by a factor of *k*.
  - SCTP (Stream Control Transmission Protocol), an example of inverse multiplexing, can run a connection using multiple network interfaces.

Crash Recovery

- If the transport entity is entirely within hosts, *recovery from network and router crashes* can be easily done as follows:
  - If the network layer provides datagram service, the transport entities expect lost segments all the time and know how to cope with them.
  - If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which segments it has received and which ones it has not received. The latter one can be retransmitted.

Cont.

- A more troublesome problem is how to recover from host recovery.
  - Writing a segment onto the output stream (to the application process) and sending an acknowledgement are two distinct events that cannot be done simultaneously. A host may crash between two events.
  - The previous figure shows all eight combinations of client and server strategy and the valid event sequences for each one.
    - A, sending an acknowledgement; W, write to the output process; C, crashing.
    - Notice that for each strategy there is some sequence of events that causes the protocol to fail.
- Without simultaneous events, host crash and recovery cannot be made transparent to the higher layers.
  - Recovery from a layer *N* crash can only be done by layer *N*+1, and then only if the higher layer retains enough status information.


Client
Client
Client
Write
Transport entity
Server Crashes
& reboots
Ack
C
A
? W
time

Crash Recovery

Strategy used by sending host	Strategy used by receiving host: First ACK, then write			Strategy used by receiving host: First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

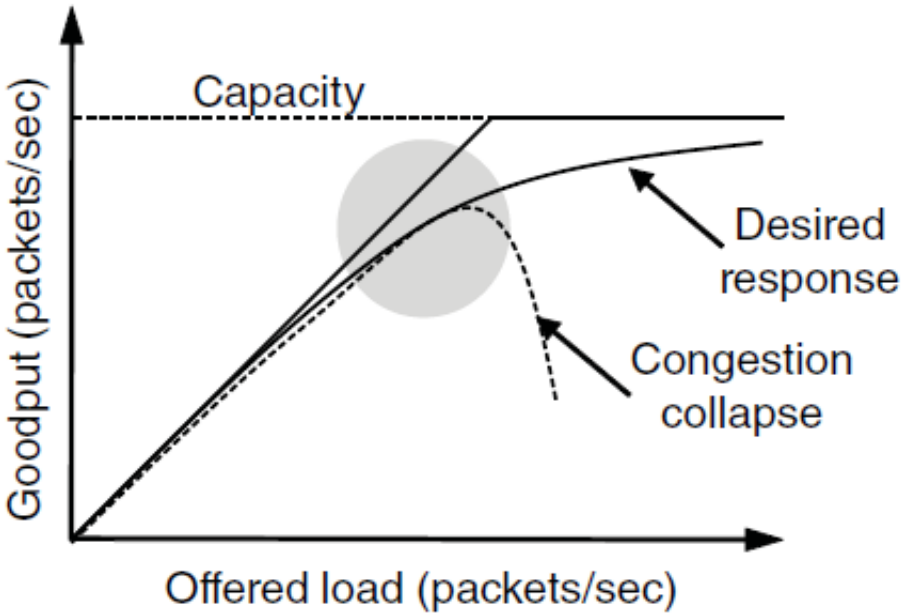
OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

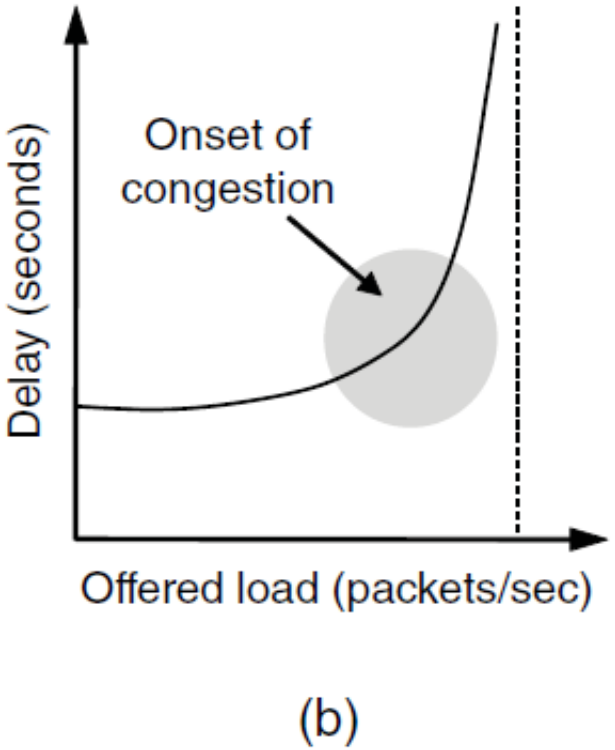
Different combinations of client and server strategy

Congestion Control

- Desirable bandwidth allocation
- Regulating the sending rate

Desirable Bandwidth Allocation (1)





(a) Goodput and (b) delay as a function of offered load

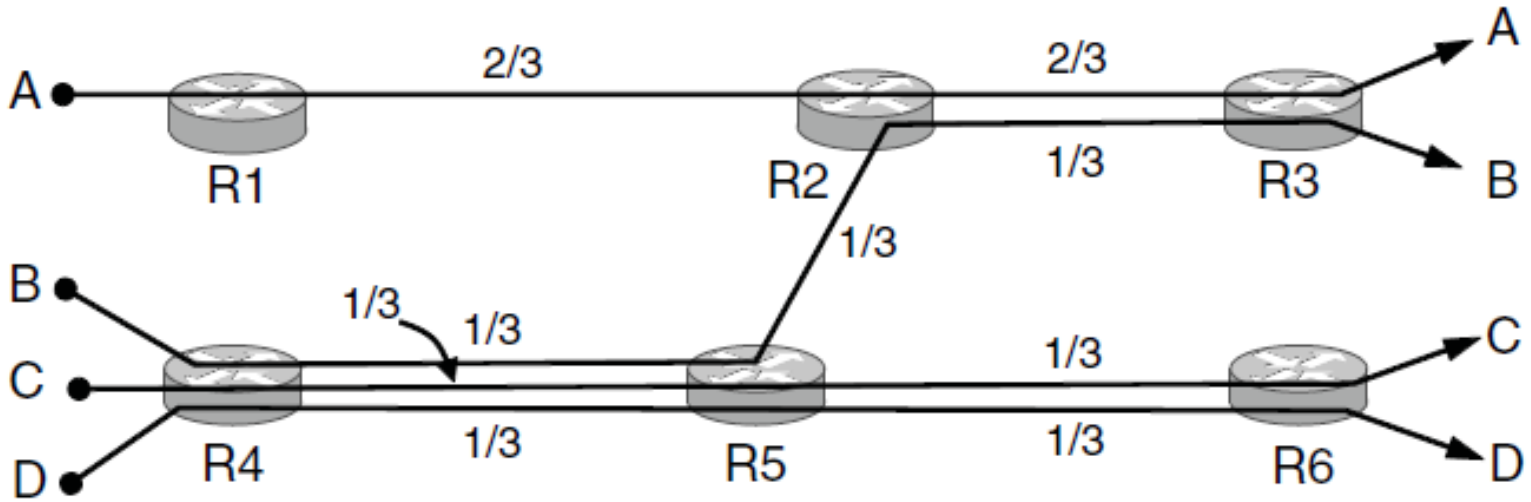
Desirable bandwidth allocation

- Efficiency and Power
  - For both goodput and delay, performance begins to degrade at the onset of congestion
  - Bandwidth should be allocated up until the delay starts to climb rapidly. This point can be identified by Kleinrock's ***power***:

    $$Power = load/delay$$

  - Power will initially rise with offered load, but will reach a maximum and fall as delay grows rapidly.
- Max-min fairness
  - Divide bandwidth between different transport senders.
  - Fairness versus efficiency
  - *An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation that is no larger.*

Desirable Bandwidth Allocation (2)



$$\max \min\{x_A, x_B, \cdots, x_D\}$$

subject to the constraint of link flow capacicity :

$$x_B + x_C + x_D \le f_{45}$$

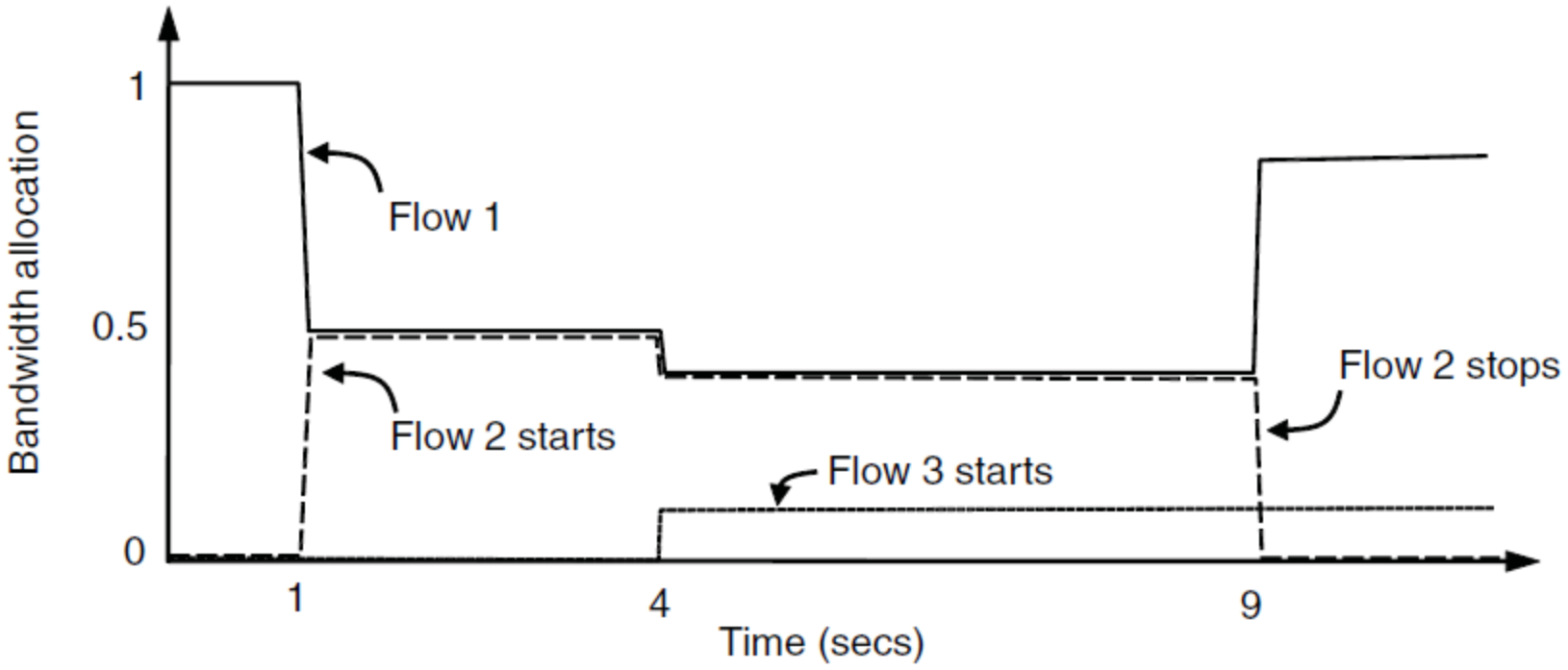
$$x_C + x_D \le f_{56}$$

$$x_A + x_B \le f_{23}$$

$$x_A \le f_{12}$$

Max-min bandwidth allocation for four flows

Desirable Bandwidth Allocation (3)

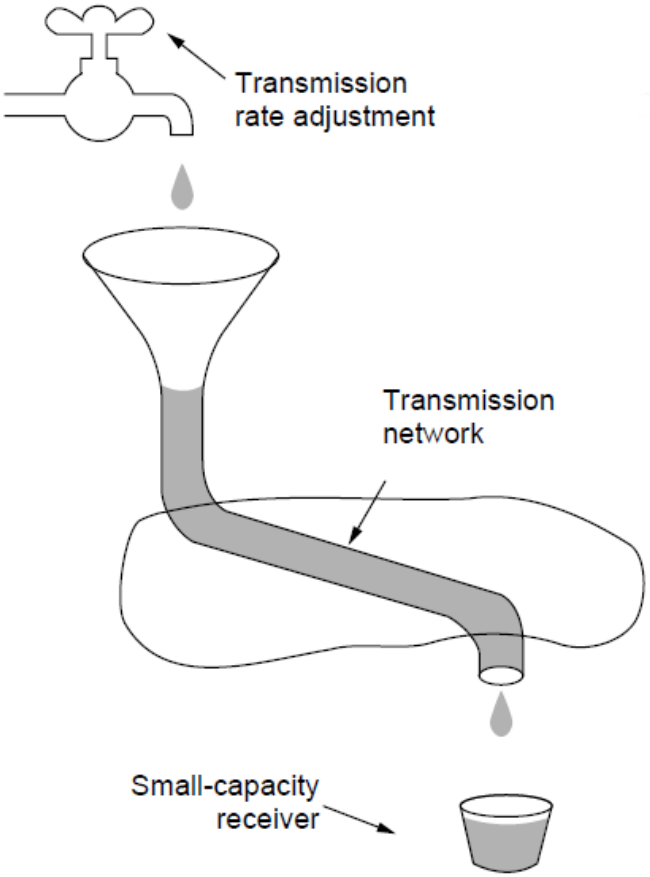


Changing bandwidth allocation over time

Regulating the sending rate

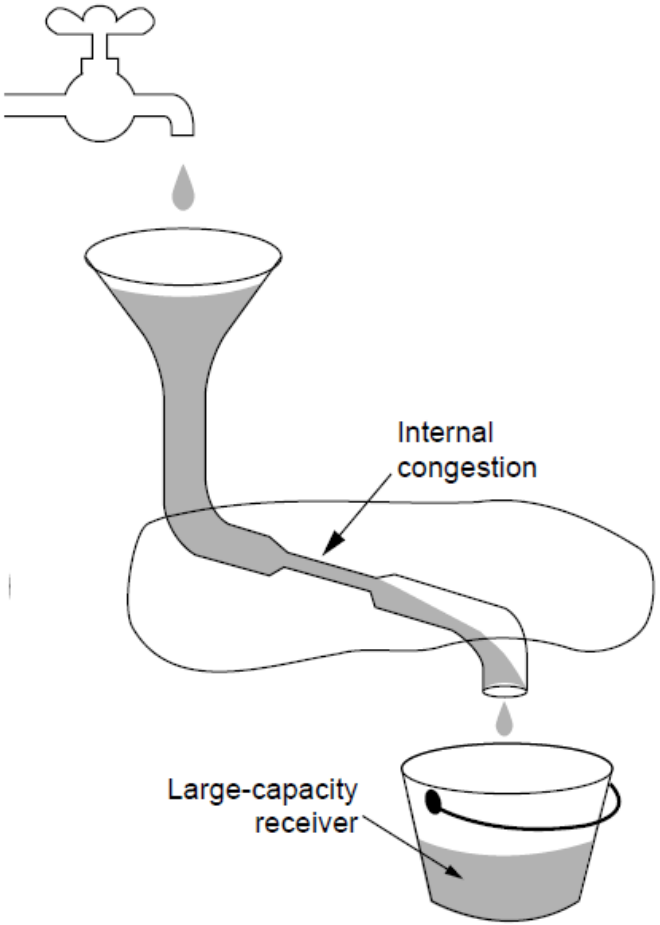
- Two factors limit the sending rate:
  - Flow control
  - Congestion
- Depend on the form of the feedback returned by the network:
  - Explicit or implicit
  - Precise or imprecise
- Control law for rate change(increase or decrease)
  - **AIMD** (Additive Increase Multiplicative Decrease) is an appropriate control law for approaching the *efficient* and *fair* operating point.

Regulating the Sending Rate (1)



A fast network feeding a low-capacity receiver

Regulating the Sending Rate (2)



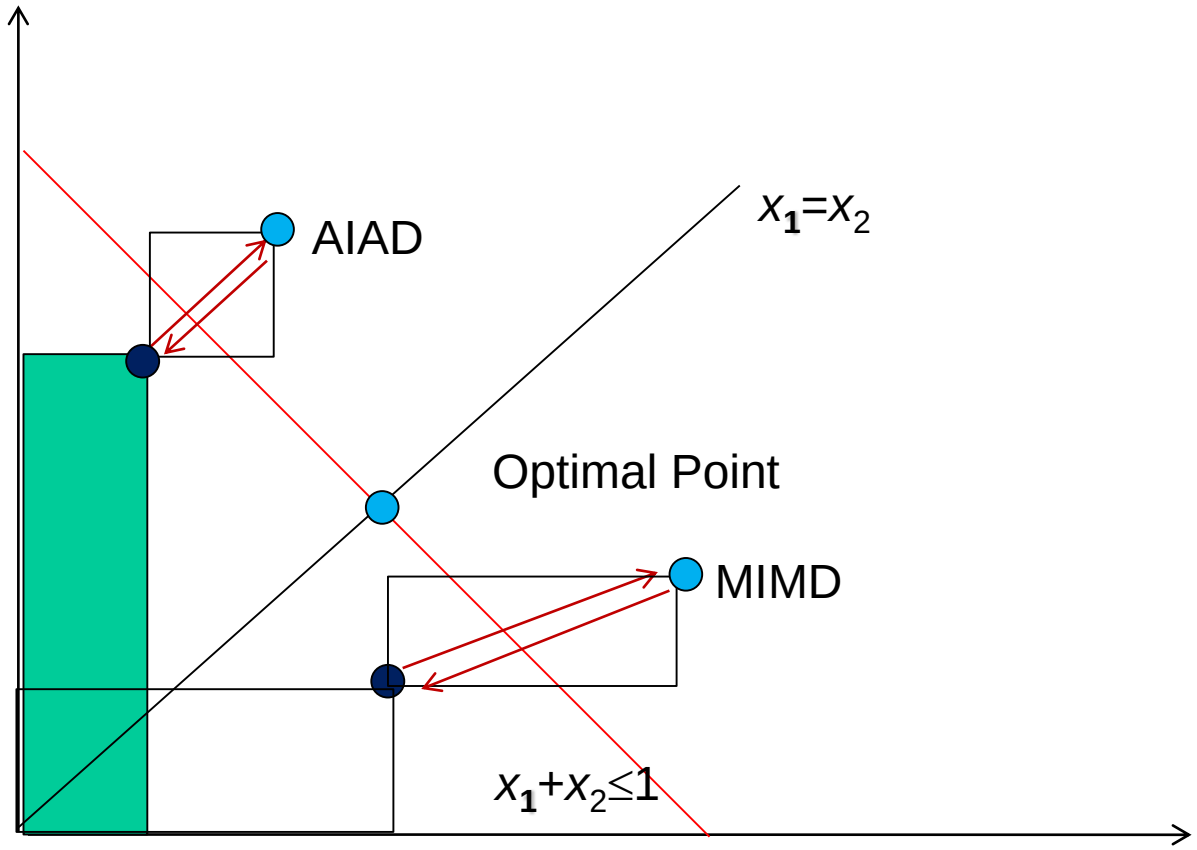
A slow network feeding a high-capacity receiver

Regulating the Sending Rate (3)

Protocol	Signal	Explicit?	Precise?
XCP	Rate to use	Yes	Yes
TCP with ECN	Congestion warning	Yes	No
FAST TCP	End-to-end delay	No	Yes
CUBIC TCP	Packet loss	No	No
TCP	Packet loss	No	No

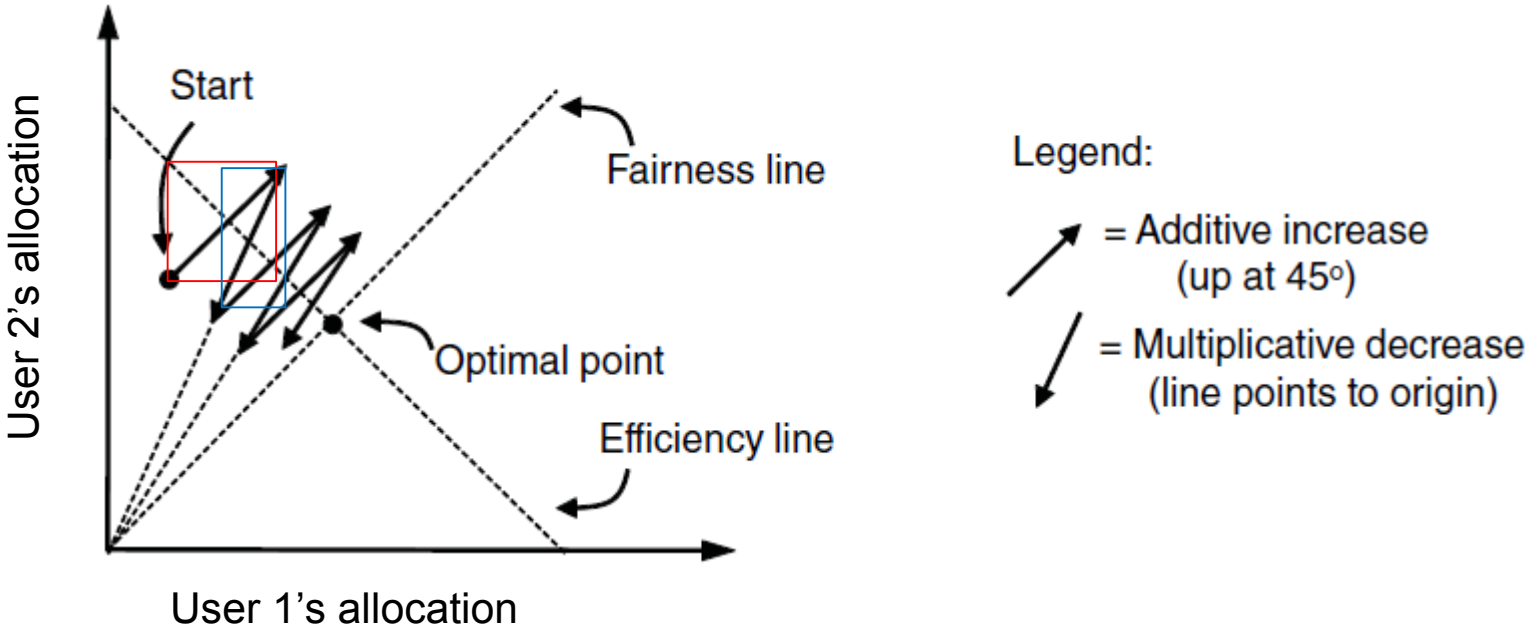
Some congestion control protocols

Control laws



Both AIAD and MIMD are close to efficient but not necessarily fair

Regulating the Sending Rate (4)



Additive Increase Multiplicative Decrease (AIMD) control law.

AIMD Converges to the optimal point VS MIAD diverges from the point.

Wireless Issue

- Wireless network lose packets all the time due to transmission errors, while packet loss is often used as a congestion signal, including TCP
  - The AIMD control law requires very small levels of packet loss.
  - One solution: Mask the wireless losses by using retransmission.
- The capacity of a wireless link changes over time.
  - Causes: mobility, interfering links, …

Wireless Issues