

Transport Layer

Flow control - sliding window

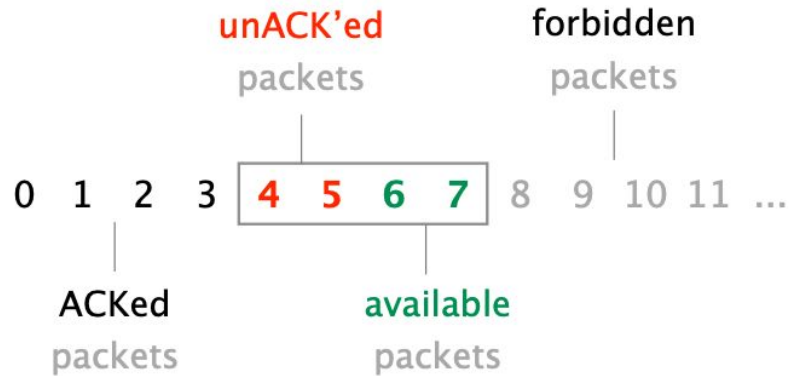
Sender keeps a list of the sequence # it can send
known as the *sending window*

Receiver also keeps a list of the acceptable sequence #
known as the *receiving window*

Sender and receiver negotiate the window size
 $sending\ window \leq receiving\ window$

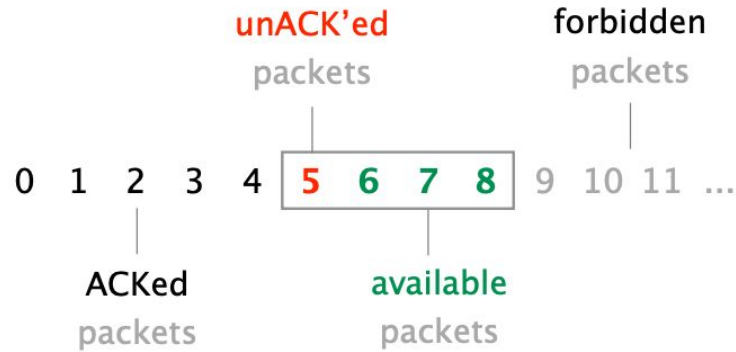
Flow control - sliding window

Example with a window composed of 4 packets



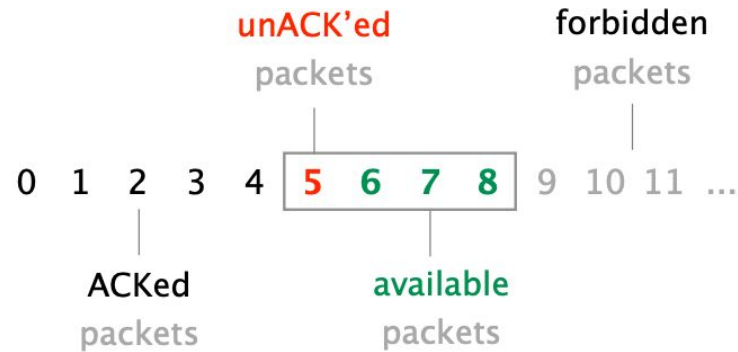
Flow control - sliding window

Window after sender receives **ACK 4**



Flow control - sliding window

Window after sender receives **ACK 4**



Timeliness of the window protocol depends on the size of the sending window

Efficiency of the protocol depends on two factors

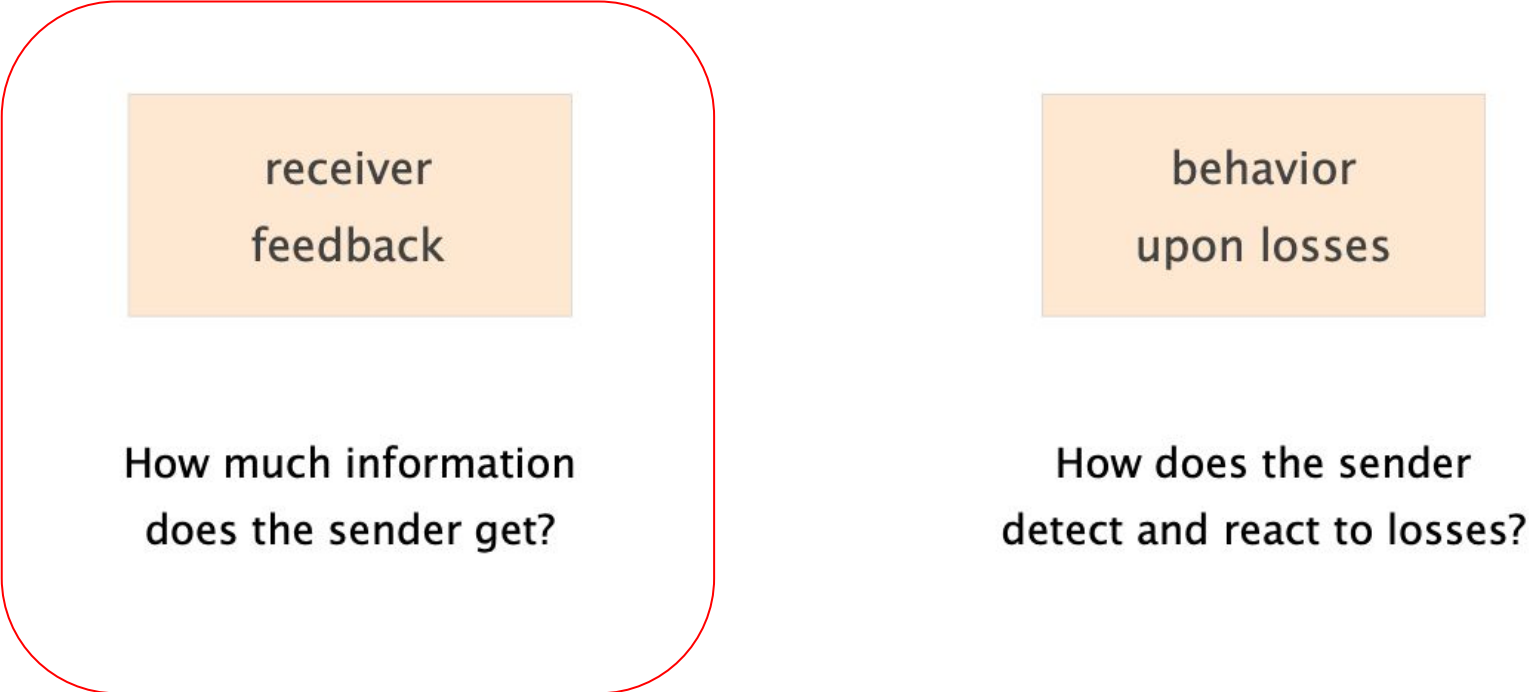
receiver
feedback

How much information
does the sender get?

behavior
upon losses

How does the sender
detect and react to losses?

Efficiency of the protocol depends on two factors



receiver
feedback

How much information
does the sender get?

behavior
upon losses

How does the sender
detect and react to losses?

ACKing individual packets provides detailed feedback, but triggers unnecessary retransmission upon losses

advantages

know fate of each packet

simple window algorithm

W single-packet algorithms

not sensitive to reordering

disadvantages

loss of an ACK packet

requires a retransmission

causes unnecessary retransmission

Cumulative ACKs enables to recover from lost ACKs, but provides coarse-grained information to the sender

approach

ACK the highest sequence number for which all the previous packets have been received

advantages

recover from lost ACKs

disadvantages

confused by reordering
incomplete information about which packets have arrived
causes unnecessary retransmission

Full Information Feedback prevents unnecessary retransmission, but can induce a sizable overhead

approach

List all packets that have been received

highest cumulative ACK, plus any additional packets

advantages

complete information

resilient form of individual ACKs

disadvantages

overhead

(hence lowering efficiency)

e.g., when large gaps between received packets

Full Information Feedback prevents unnecessary retransmission, but can induce a sizable overhead

approach

List all packets that have been received

highest cumulative ACK plus any additional packets

advantages

Once again, Internet design is all about balancing tradeoffs.

disadvantages

overhead

(hence lowering efficiency)

e.g., when large gaps between received packets

Efficiency of the protocol depends on two factors

receiver
feedback

How much information
does the sender get?

behavior
upon losses

How does the sender
detect and react to losses?

We've been talking about detecting loss using timeouts. That's not the only way



ACKS

With individual ACKs, missing packets (gaps) are implicit

Assume packet 5 is lost

but no other

ACK stream

1

2

3

4

6

7

...

sender can infer that 5 is missing
and resend 5 after k subsequent packets

With full information, missing packets (gaps) are explicit

Assume packet 5 is lost

but no other

ACK stream

up to 1

up to 2

up to 3

up to 4

up to 4, plus 6

up to 4, plus 6—7

...

sender learns that 5 is missing

retransmits after k packets

With cumulative ACKs, missing packets are harder to know

Assume packet 5 is lost
but no other

ACK stream

1

2

3

4

4 sent when 6 arrives

4 sent when 7 arrives

...

Duplicate ACKs are a sign of isolated losses. Dealing with them is trickier though.

situation

Lack of ACK progress means that 5 hasn't made it

Stream of ACKs means that (some) packets are delivered

Sender could trigger **resend**
upon receiving k duplicates ACKs

but *what* do you resend?

only 5 or 5 and everything after?

What about fairness?

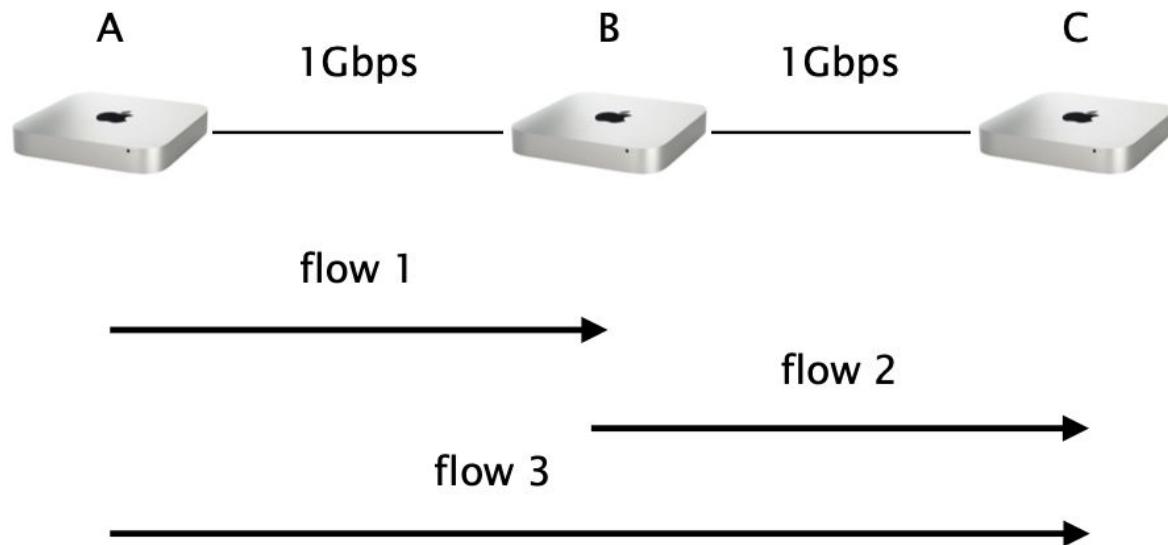
Design a *correct, timely, efficient* and **fair** transport mechanism
knowing that

packets can get

- lost
- corrupted
- reordered
- delayed
- duplicated

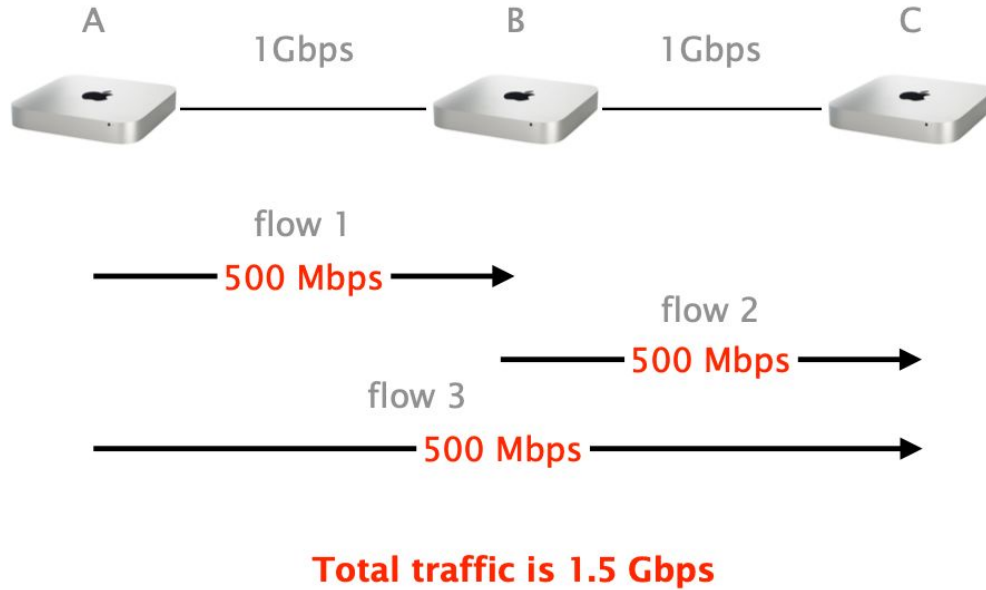
When n entities are using our transport mechanism, we want a fair allocation of the available bandwidth

Consider this simple network in which three hosts are sharing two links

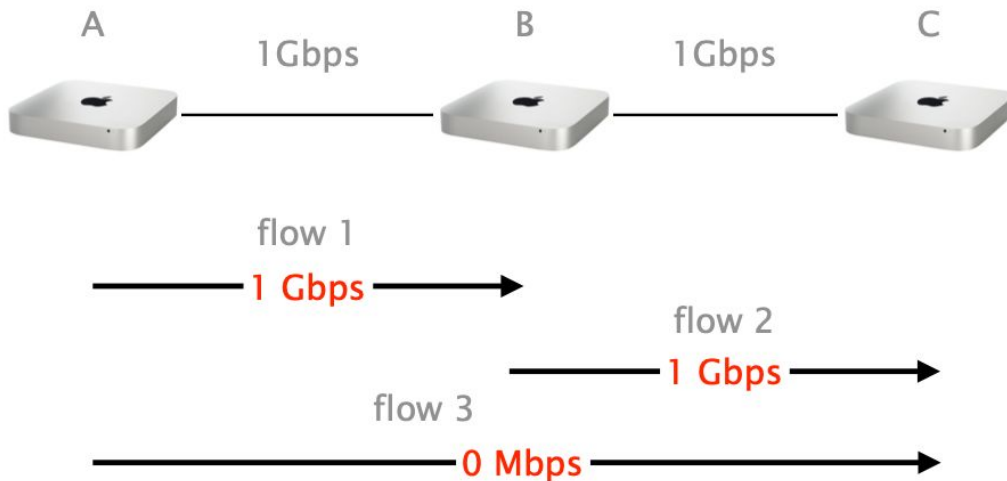


What is a fair allocation for the 3 flows?

An equal allocation is certainly “fair”, but what about the efficiency of the network?

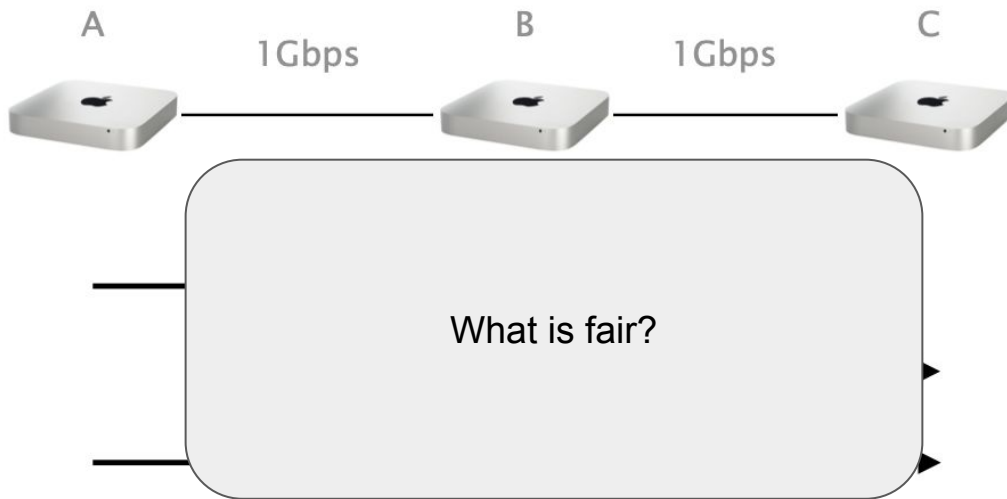


Fairness and efficiency don't always play along, here an unfair allocation ends up more *efficient*



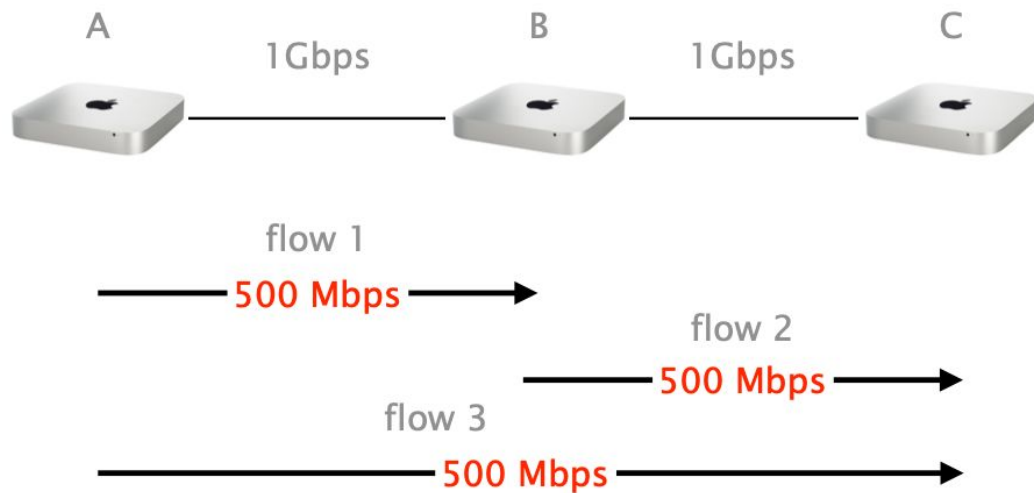
Total traffic is 2 Gbps!

Fairness and efficiency don't always play along, here an unfair allocation ends up more *efficient*



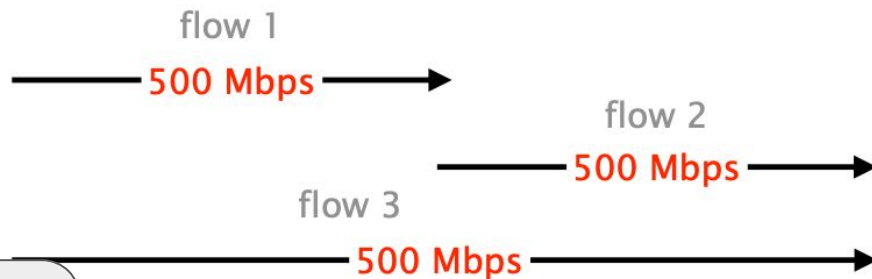
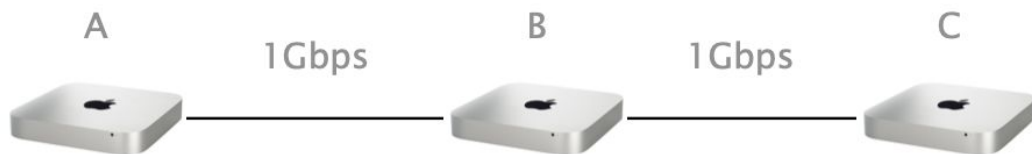
Total traffic is 2 Gbps!

Equal-per-flow isn't really fair as (A,C) crosses two links: **it uses more resources**



Total traffic is 1.5 Gbps

With equal-per-flow, A ends up with 1 Gbps because it sends 2 flows, while B ends up with 500 Mbps



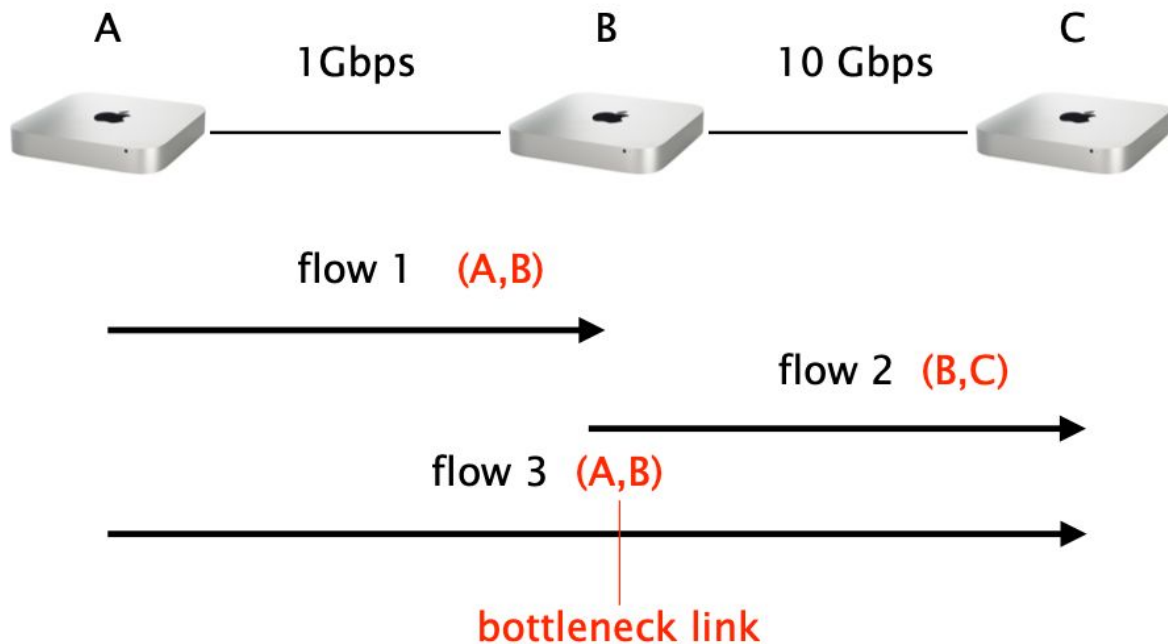
Is that fair?

Total traffic is 1.5 Gbps

Seeking an exact notion of fairness is not productive. What matters is to avoid starvation.

equal-per-flow is good enough for this

Simply dividing the available bandwidth doesn't work in practice since flows can see different bottlenecks



Intuitively, we want to give users with "small" demands what they want, and evenly distribute the rest

Max-min fair allocation is such that

the lowest demand is maximized

after the lowest demand has been satisfied,
the second lowest demand is maximized

after the second lowest demand has been satisfied,
the third lowest demand is maximized

and so on...

Max-min fair allocation can easily be computed

- step 1 Start with all flows at rate 0
- step 2 Increase the flows until there is
 a new bottleneck in the network
- step 3 Hold the fixed rate of the flows
 that are bottlenecked
- step 4 Go to step 2 for the remaining flows

Done!

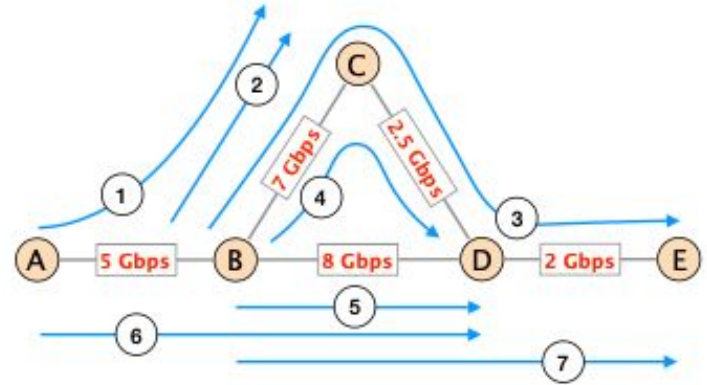
Example

Demands: {2, 2.6, 4, 5}

Capacity: 10

Example

Consider the network on the right consisting of 5 nodes (A to E). Each link has a maximal bandwidth indicated in red. 7 flows (1 to 7) are using the network at the same time. You can assume that they have to send a lot of traffic and will use whatever bandwidth they will get. Apply the max-min fair allocation algorithm to find a fair bandwidth allocation for each flow.



A network with shared links and 7 flows.

For each flow, what is the bottleneck link?

Example

Consider the network on the right consisting of 5 nodes (A to E). Each link has a maximal bandwidth indicated in red. 7 flows (1 to 7) are using the network at the same time. You can assume that they have to send a lot of traffic and will use whatever bandwidth they will get. Apply the max-min fair allocation algorithm to find a fair bandwidth allocation for each flow.

For each flow, what is the bottleneck link?

Bottleneck link	D-E	C-D	B-C	A-B	B-D
Flow 1 A - B - C	1	1.5	2.25		
Flow 2 B - C	1	1.5	2.25		
Flow 3 B - C - D - E	1				
Flow 4 B - C - D	1	1.5			
Flow 5 B - D	1	1.5	2.25	2.75	4.25
Flow 6 A - B - D	1	1.5	2.25	2.75	
Flow 7 B - D - E	1				

Max-min fair allocation can be approximated by slowly increasing W until a loss is detected

Intuition

Progressively increase
the sending window size

max=receiving window

Whenever a loss is detected,
decrease the window size

signal of congestion

Repeat