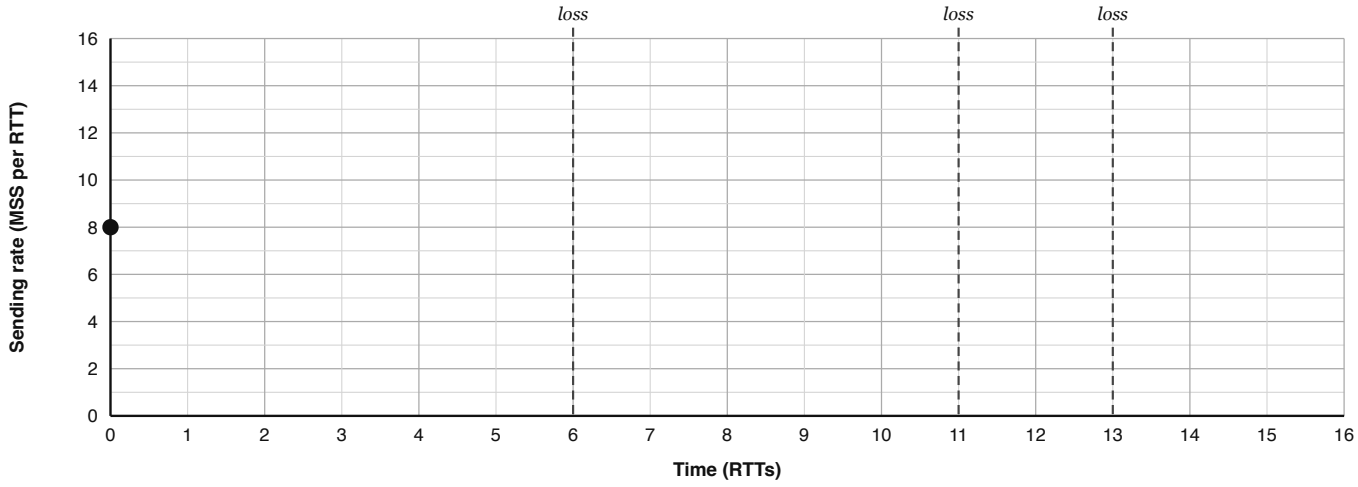
Q1 AIMD and the Congestion Window Ch. 3 slides 122–124

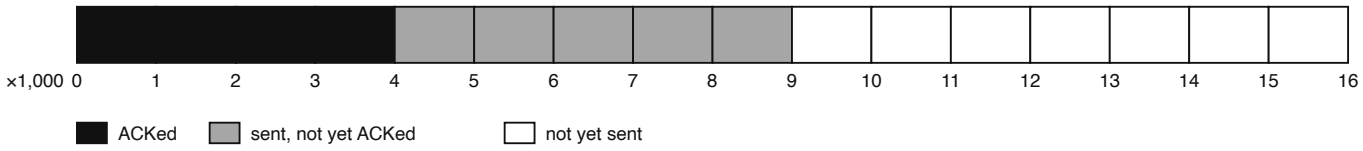
Names: _____

A TCP Reno sender is in congestion avoidance. At **RTT 0** it is sending **8 MSS per RTT**, the dot on the graph. It detects a loss by **three duplicate ACKs** at **RTT 6**, **RTT 11** and **RTT 13**, the dashed lines. No other losses occur.

1. Draw the sender's rate from RTT 0 to RTT 16.



Later, a sender with an **MSS of 1,000 bytes** has **`cwnd` = 8 MSS**. The strip is its sequence-number space: each box is one segment, and the tick labels count bytes in thousands. Bytes **1–4,000** have been ACKed. Bytes **4,001–9,000** have been sent but not ACKed.



2. Write the rule the sender must obey, using `LastByteSent`, `LastByteAcked` and `cwnd`.

3. Draw a bracket above the strip over the bytes `cwnd` covers right now. How many more bytes may the sender send before it must wait for an ACK?

_____ bytes

4. The sender sends bytes that fill the rest of its window. Then an ACK arrives with **ACK number 6,001**; `cwnd` is still 8 MSS. Which byte range can the sender send now?

Bytes _____ to _____

5. The RTT is **100 ms**. Roughly what is this sender's rate, in **kbps**?

_____ kbps

Discussion

Reno halves its rate after three duplicate ACKs but drops to 1 MSS after a timeout. Why is a timeout treated as the worse sign?

v261013-1532

Q2 **Longest Prefix Match** Ch. 4 slides 17–22

A router's forwarding table matches the leading bits of a datagram's 32-bit destination address; each `*` matches either bit. The router uses **longest prefix match**.

Destination address range	Link interface
`11000000 10101000 ******** ********`	**0**
`11000000 10101000 0001**** ********`	**1**
`11000000 10101000 00011*** ********`	**2**
`11000000 10101000 00011010 ********`	**3**
otherwise	**4**

1. For each destination, list the interfaces whose entries match, then give the interface the datagram leaves on. Row a is filled in as an example.

	Destination address	Entries that match	Interface
a	`11000000 10101000 00010110 10100001`	*0, 1*	*1*
b	`11000000 10101000 00011010 01100100`		
c	`11000000 10101000 00011100 00000001`		
d	`11000000 10101000 11110000 00001111`		
e	`11000001 10101000 00011010 01100100`		

2. The operator adds one entry: `11000000 10101000 00011100 0000****` → interface **5**. Which of a–e now leave on a different interface? Why are the others unaffected?

3. Go back to the original table. Suppose the router checked entries top to bottom and used the **first** one that matched. Which of a–e would leave on the wrong interface? How could you reorder the table so the first match is always the longest?

Discussion

The router reads this table for every datagram it forwards. Who writes the table, and how often does it change compared with how often it is read?