Q1.

Persistent HTTP	pipelining		=	3	RTT	+	58	L/R	+	18	Q
Persistent HTTP	no pipelining		=	11	RTT	+	58	L/R	+	18	Q
non-persistent HTTP	no parallel		=	20	RTT	+	58	L/R	+	18	Q
non-persistent HTTP	parallel TCP max=?	2	=	12	RTT	+	34	L/R	+	9	Q
non-persistent HTTP	parallel TCP max=?	4	=	8	RTT	+	22	L/R	+	5	Q

Q2

	access link T	=	375000	/	6000000					=	0.0625	s
acc lnk	access link TB	=	120	/	60	*	0.0625			=	0.125	
	avg access delay	=	0.0625	/	(1 - 0.125)					=	0.071429	s
	Total avg res t	=	0.071429	+	0	+	6			=	6.071429	s
	avg access delay	=	0.0625	/	(1 - 0.4	*	0.125)			=	0.065789	s
	Total avg res t	=	0.065789	+	0	+	6			=	6.065789	s
	Total avg res t	=	0.6	*	0	+	0.4	*	6.065789	=	2.426316	s
	The average response time is reduced from						6.071429	to	2.426316			
	access link T	=	375000	/	5000000					=	0.075	s
lnk 1	access link TB	=	120	/	60	*	30%	*	0.075	=	0.045	
	avg access delay	=	0.075	/	(1 - 0.045)					=	0.078534	s
	access link T	=	375000	/	10000000					=	0.0375	s
lnk 2	access link TB	=	120	/	60	*	70%	*	0.0375	=	0.0525	
	avg access delay	=	0.0375	/	(1 - 0.0525)					=	0.039578	s
	Total avg res t	=	0.3*0.079 + 0.7*0.04			+	0	+	6	=	6.051265	s

Q3

	12	S/R	>	RTT	>	4	S/R
=	2	RTT	+	4	S/R	+
	1	RTT	+	4	S/R	+
	1	RTT	+	16	S/R	+
	0	RTT	+	28	S/R	+
	0	RTT	+	0	S/R	+
	0	RTT	+	0	S/R	+
=	4	RTT	+	52	S/R	

	?	S/R	>	RTT	>	12	S/R
=	2	RTT	+	4	S/R	+	
	1	RTT	+	4	S/R	+	
	1	RTT	+	4	S/R	+	
	1	RTT	+	28	S/R	+	
	0	RTT	+	0	S/R	+	
	0	RTT	+	0	S/R	+	
=	5	RTT	+	40	S/R		

Q4

i

Sender Events:

Send data octets 1-1000

Send data octets 1001-2000

Send data octets 2001-2300

Receive ack for 1000

Receive ack for 2000

Receive ack for 2300

Receiver Events:

advertise window=2300

ack up to 1000, window=1300

ack up to 2000, window=300

ack up to 2300, window=0

ii

Sender Events:

Send data octets 2501-3500

Send data octets 3501-4500

Receive ack for 3500

Receive ack for 4500

Receive ack for 4500

Receiver Events:

application reads 1800 octets

ack up to 2500, window=1800

ack up to 3500, window=800

ack up to 4300, window=0

application reads 1000 octets

ack up to 4500, window=1000

Q5

i. TCP slow start is operating in the interval [4, 7]: double the previous window size

ii. TCP congestion avoidance is operating in the intervals [1, 3] and [8, 16]: linearly increase the window size

iii. After the 3rd transmission round, packet loss is detected due to timeout, and hence the congestion window size is set to 1.

iv. After the 10th transmission round, segment loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.

v. The maximum possible initial value of the threshold at the first transmission round is 30 since when the congestion window size is 30, TCP congestion avoidance is operating.

vi. The threshold is 16 during the 4th transmission round since packet loss is detected. When loss is detected during transmission round 3, the congestion windows size is 32. Hence the threshold is 16 during the 4th transmission round.

vii. The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 10, the congestion windows size is 18. Hence the threshold is 9 during the 11st transmission round.

viii. The congestion window size is 6 and the threshold value is 3 since there is a loss at the 15th transmission round in which the congestion window size is 6.