1 TB-TCP XFSM IMPLEMENTATION

#	Event	Conditions	State	Next	Actions, Updates
1	connect	any	C	SyS	setField(tcp.flags,1,SYN); sendPacket(1,1); setTimer(1,0,0.5s);
2	timeout	any	SyS	SyS	*same actions as table entry 1*
3	pktRcvd	tcp.flags = ACK	SyS	SS	removeTimer(1); setField(tcp.flags,2,ACK); setTimer(nextTx,0,1µs);
4	any	socketClosed = true, bytesToTx ≤ 0, lastAck ≥ highTx	SS,CA,R,PR	FW1	setField(tcp.flags,1,FIN); sendPacket(1,1); setTimer(0,0,2µs);
5	timeout	timeoutSeq ≥ lastAck	FW1	FW1	*same actions as table entry 4*
6	pktRcvd	tcp.flags = ACK	FW1	FW2	*No actions*
7	pktRcvd	tcp.flags = FINACK	FW1	TW	setField(tcp.flags,1,ACK); setTimer(120s); sendPacket(1,1);
8	timeout	any	TW	C	closeSocket();

9	timeout	timeout.data2 ≤ 0, availWin >0	SS	SS	rttDoubled = rtt*2; setTimer(nextTxSeq, 1, rttDoubled);
					setField(tcp.seqNo, 2, nextTxSeq); setField(tcp.ackNo, 2, remoteSeq);
					setField(tcp.timestamp, 2, currentTime);
					sendPacket(2, 1); nextTxSeq = nextTxSeq + 1448; highTxMark = nextTxSeq + 1448;
					highTxMark = nextTxSeq + 1448;
					setTimer(nextTxSeq, 0, 1); availWin = highTxMark - lastAckedSeqedSeq;
					availWin = cwnd - availWin;

10	timeout	timeout.data2 >0, timeout.data1 ≥ lastAckedSeq,
		currentRetxRound <timeout.data1
			SS	PR	currentRetxRound=currentRetxRound+1; cwnd = cwnd/2;
					pktsToPace = cwnd/1448; pacingTime = rtt/pktsToPace;
					nextTxSeq = lastAckedSeqedSeq;
					setTimer(nextTxSeq, 0, pacingTime);

11	pktRcvd	tcp.flags >2, lastAckedSeq ≠ tcp.AckNo	SS	SS	cwnd = cwnd + 1448; lastAckedSeq = tcp.ackNo;
					setTimer(nextTxSeq, 0, 1);
					rtt = max(rtt, tcp.timestampEchoReply);
					availWin = highTxMark - lastAckedSeq;
					availWin = cwnd - availWin;

12	timeout	timeout.reTxCount ≤ 0, availWin >0,timeout.seqNo = nextTxSeq	PR	PR	setField(tcp.seqNo, 2, nextTxSeq); setField(tcp.ackNo, 2, remoteSeq);
					setField(tcp.timestamp, 2, currentTime);
					nextTxSeq = nextTxSeq + 1448; highTxMark = nextTxSeq + 1448;
					pktsToPace = pktsToPace + 1; pacingTime = rtt/pktsToPace;
					availWin = highTxMark - lastAckedSeqedSeq;
					availWin = cwnd - availWin;

13	timeout	timeout.reTxCount ≤ 0, availWin ≤ 0 timeout.seqNo = nextTxSeq	PR	CA	*No actions*

14	pktRcvd	tcp.flags>2, lastAckedSeq ≠ tcp.ackNo	PR	PR	lastAckedSeq=tcp.ackNo; availWin = highTxMark - lastAckedSeq;
					availWin=cwnd-availWin;

15	pktRcvd	tcp.flags = ACK, lastAckedSeq ≠ tcp.ackNo	CA	CA	cwndIncr = MSS*MSS; cwndIncr = cwndIncr/cwnd;
					cwnd = cwnd + cwndIncr;
					lastAckedSeq = tcp.ackNo; rttDoubled = rtt*2;
					setTimer(nextTxSeq, 1, rttDoubled);
					setField(tcp.seqNo, 2, nextTxSeq);
					setField(tcp.ackNo, 2, remoteSeq); setField(tcp.timestamp, 2, currentTime);
					sendPacket(2,1); nextTxSeq = nextTxSeq + 1448;
					highTxMark = nextTxSeq + 1448; rtt = max(rtt, ack.timeStamp);
					setTimer(nextTxSeq, ZERO, ONE);
					availWin = highTxMark - lastAckedSeqedSeq; availWin = cwnd - availWin;

16	timeout	timeout.reTxCount ≤ 0, availWin ≥ MSS	CA	CA	rttDoubled = rtt*2; setTimer(nextTxSeq, 1, rttDoubled);
					setField(tcp.seqNo, 2, nextTxSeq);
					setField(tcp.ackNo, 2, remoteSeq);
					setField(tcp.timestamp, 2, currentTime); sendPacket(2,1);
					setTimer(nextTxSeq,ZERO,ONE);
					availWin = highTxMark - lastAckedSeqedSeq;
					availWin = cwnd - availWin;

17	timeout	timeout.reTxCount >0, timeout.seqNo ≥ lastAckedSeq,
		currentReTxRound <timeout.reTxCount
			CA	PR	currentReTxRound = currentReTxRound + 1;
					cwnd = cwnd/2; pktsToPace = cwnd/1448;
					pacingTime = rtt/pktsToPace;
					nextTxSeq = lastAckedSeq;

18	timeout	timeout.data2 ≤ 0, nextTxSeq <highTxMark, timeout.data1 = nextTxSeq	R	R	setField(tcp.seqNo, 2, nextTxSeq); setField(tcp.ackNo, 2, remoteSeq);
					setField(tcp.timestamp, 2, currentTime); sendPacket(2, 1);
					nextTxSeq = nextTxSeq + 1448; setTimer(nextTxSeq, 0, pacingTime);

19	pktRcvd	tcp.flags >2, nextTxSeq ≥ highTxMark	R	PR	currentRetxRound = currentRetxRound - 1; pktsToPace = pktsToPace + 1;
					pacingTime = rtt/pktsToPace; setTimer(nextTxSeq, 0, pacingTime);
					lastAckedSeq = tcp.ackNo;
					availWin = highTxMark - lastAckedSeqedSeq; availWin = cwnd - availWin;

Table 1: The complete TB-TCP XFSM implementation. States: C = Closed; SyS = SYN Sent; FW(1/2) = FIN-WAIT(1/2); TW = Time Wait; SS = Slow Start; CA = Congestion Avoidance; R = Recovery; PR = Post Recovery