Analysis of TCP CUBIC: Deterministic Loss Model and Steady State Throughput Model using Markov Chain

Course: ECE GY 7353 Network Modeling and Analysis

Instructor: Prof. S. Panwar

Report By: Vinita Parasrampuria

Net ID: vp2238

1. Abstract:

TCP CUBIC is a new variant of Congestion control in networks which has been adopted in Linux operating systems. In this study, I have mainly focused on understanding how TCP CUBIC works [1], throughput analysis of CUBIC using a deterministic loss model [2],[3] and steady state throughput of TCP Cubic using a Markov chain model in wireless environment where losses are random [4]. My main motivation to take up this topic was to study how the multiplicative decrease factor β can be varied to achieve better throughput. I thought one possible way to implement variability in β is to count the number of packets lost when congestion happens and change β accordingly. If the number of packets lost is more, then the congestion is more severe and hence β should be increased. However, before delving into this analysis, I spent some time reading and understanding the works of other authors and this remains a topic of future study. While doing the analysis I did find few minor differences in some of the published papers. I also tried to simulate analytical models given in the papers.

2. Introduction:

With the evolution of Internet when the speed and distance of transmission has increased manifolds, there is a need to improve the utilization of bandwidth. These systems have high bandwidth delay product. In congestion control protocols like TCP Reno, TCP New Reno, TCP-SACK window grows per round trip time (RTT). For instance, if a network has bandwidth 10Gbps, RTT of 100 ms and packets of size 1250 bytes each, then the BDP can be calculated as:

$10X10^9$ bits/sec X (100 /1000) sec /1250X8 bits=100,000 packets.

Suppose TCP grows its window from 50,000 packets, to reach full utilization, the time required to grow the window size is given as:

50,000X100/1000 sec=5,000 sec=1.38 hours.

High speed TCP like FAST, HSTCP, STCP, BIC-TCP, Westwood, HTCP etc. were introduced to overcome this problem. It was found that BIC-TCP was highly stable algorithm and hence implemented by Linux. BIC-TCP is a precursor to CUBIC.

BIC-TCP:

It is based on binary search algorithm. It consists of two parts: binary search increase and additive increase.

a. Binary search increase:

BIC TCP works as follows:

1. The last window size where there was a packet loss is defined as W_{max}.
2. The last window size where there is no loss for one RTT is W_{min}. The starting point is the window size after reduction.
3. The midpoint of these two windows is calculated and window size is set to this value.
4. After growing the window, if there is no packet loss then the network can handle more traffic. At this point W_{min} is updated and set to the current window size. W_{max} remains the same.
5. New midpoint is now calculated using W_{min} and W_{max} and window grows further.
6. This type of window growth makes sense because if the network conditions are quite similar to the last congestion signal when there was a packet loss, then the current path must be somewhere in between these two windows.

   With this algorithm the window size grows really fast when the current window size is very small as compared to W_{max}. This increment slows down when window size approaches W_{max}. Hence at the saturation point where there was a packet loss, the window growth is much slower which provides it stability. This type of increase function is characteristic of logarithmic function. This is the concave region of operation.

b. Additive Increase:

Binary search increase is combined with additive increase to ensure fast convergence and RTT fairness.

1. If the midpoint is greater than a fixed constant S_{max}, that means there will be too much increment in one RTT and may cause transmission jitter. Hence the window increases by S_{max}. This is the linear increment phase.
2. Once the calculated midpoint is less than $S_{max,}$ window increases to this calculated midpoint.
3. When there is a packet loss, the window is reduced by a multiplicative factor and W_{max} is updated.

Therefore, when the maximum window size is large, initially the growth is additive increase because multiplicative increase causes larger reduction in window when there is a packet loss. Hence the additive increase period is more. On the other hand, when the maximum window size is less, binary search period is more.

c. Slow start:

If there is no further loss when the window size is equal to W_{max}, the algorithm enters the convex region. Initially the growth is very slow which further provides stability but increases exponentially after probing for some time near the saturation point. In order to find the new maximum, the protocol enters slow start phase. The window increases each RTT in steps of $W_{max}+S_{min}$, $W_{max}+2*S_{min}$, $W_{max}+4*S_{min}$,….., $W_{max}+S_{max}$. After safely crossing this point, window growth enters the exponential phase.

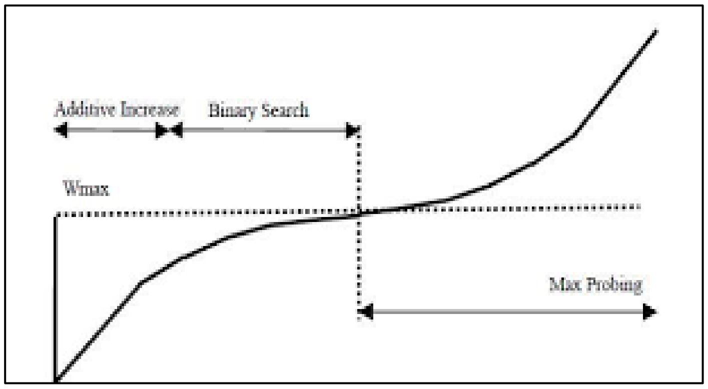

FIG 1: WINDOW GROWTH FUNCTION OF BIC-TCP (TAKEN FROM [1])

TCP CUBIC:

In TCP CUBIC the concave and convex portions of BIC-TCP is simply replaced by a cubic function. CUBIC makes use of the time between two consecutive packet loss events to determine the growth function. This makes it independent of RTT. Since BIC-TCP is very aggressive for TCP under short RTT or low speed networks. Moreover, the implementation and performance analysis of binary search increase, max probing, S_{max} and S_{min} gets complex. To overcome these issues, CUBIC was introduced which uses a cubic function. This function has shape similar to the BIC-TCP. It works as follows:

1. W_{max} is the window size at which the congestion happens.
2. After congestion the window size reduces to (1-β)W_{max}, where β is multiplicative decrease factor.
3. Once entering the period of fast recovery, it starts increasing window size using the concave profile.
4. The plateau of the curve is achieved at W_{max}.
5. Upon reaching W_{max}, the convex region begins.

Window growth in the concave region followed by convex region, gives stability to the protocol. The window size stays near the W_{max} for some time before entering the convex phase. If there is no packet loss in this time, then the maximum is far beyond and therefore growth is exponential. The window growth of the cubic function is given as :

$$\mathrm{W(t)} = \mathrm{C(t-K)3} + W_{max} \qquad (1)$$

C=Cubic growth factor

t= Time elapsed from the last window reduction

K=the time required to increase W to W_{max} if there is no further loss event

Calculation of W_{max}:

At time t=0; window size is (1-β) W_{max}. Putting these values in eq. (1)

$$(1-\beta)\, W_{max} = -CK^3 + W_{max}$$

$$\beta\, W_{max} = CK^3$$

$$K=\sqrt[3]{\frac{\beta\ \mathrm{Wmax}}{C}} \qquad (2)$$

After receiving ACK during congestion avoidance, the window size is calculated using the equation (1) during the next RTT. TCP CUBIC operates in three regions:

a. If the calculated window size using equation (1) is less than the window size it should have achieved if it were using the standard TCP protocol, then the window growth follows TCP growth function and is in TCP mode.
b. If the calculated congestion window is less than W_{max}, then it is in concave region.
c. If the calculated congestion window is more than W_{max}, then it is in convex region.

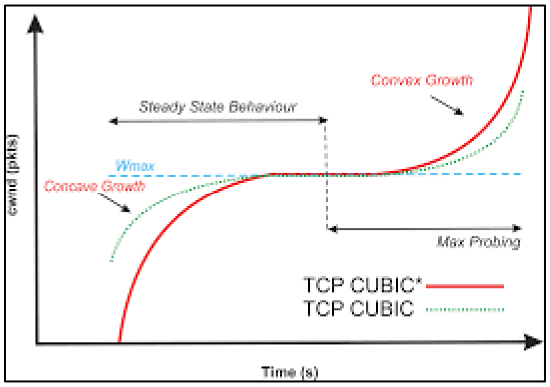

FIG 2 : WINDOW GROWTH FUNCTION OF TCP CUBIC (TAKEN FROM [10])

a. TCP-friendly region:

Upon receiving an ACK, it is first checked whether the protocol is in the TCP region. This is done by checking the window growth for standard TCP.

The AIMD response function T is defined as the AIMD's flow steady state sending rate T in packets per second for s deterministic loss model.

The approximate value of the response function given by[5]:

$$\frac{\sqrt{\alpha(2-\beta)/2\beta p}}{\mathrm{RTT}} \qquad (3)$$

where α= growth factor
β= multiplicative decrease factor
p=probability of loss
RTT= Round trip time

When α=1, β=0.5 for standard TCP, the response function is given as:

$$\frac{\sqrt{1(2-0.5)/2*0.5*p}}{\mathrm{RTT}}=\frac{\sqrt{1.5/p}}{\mathrm{RTT}}$$

For equation (3) to be the same for standard TCP,

$$\frac{\sqrt{\alpha(2-\beta)/2\beta p}}{\mathrm{RTT}} = \frac{\sqrt{1.5/p}}{\mathrm{RTT}}$$
$$\alpha = 3\beta/2-\beta$$

If the growth of TCP is α per RTT, then the growth function is given as:

$$W_{TCP}(t) = W_{max}(1-\beta) + \frac{3\beta}{2-\beta}\frac{t}{RTT} \tag{4}$$

If the congestion window cwnd is less than $W_{TCP}(t)$, then upon reception of ACK, the cwnd is set to WTCP(t) and the protocol operates in TCP mode.

b. Concave Region:

In case the protocol is not in the TCP mode and cwnd is less than Wmax then the protocol operates in concave region. For each ACK received, the window grows as W(t+RTT)-cwnd/cwnd.

c. Convex Region:

Once the congestion window is more than Wmax, the protocol operates in convex region. As the congestion window has increased past the plateau at Wmax, this implies there is more available bandwidth may be due to reduction in flows. Hence there is room for more increase in congestion window. This profile ensures stability because the window grows very slowly in the beginning before increasing the window exponentially. This phase is also called max probing since the protocol is searching for new Wmax. In this phase too, for each ACK received, the window grows as W(t+RTT)-cwnd/cwnd.

Multiplicative decrease:

Upon a packet loss, the window size is reduced to (1-β) Wmax. Generally, β is set to 0.2 or 0.3.

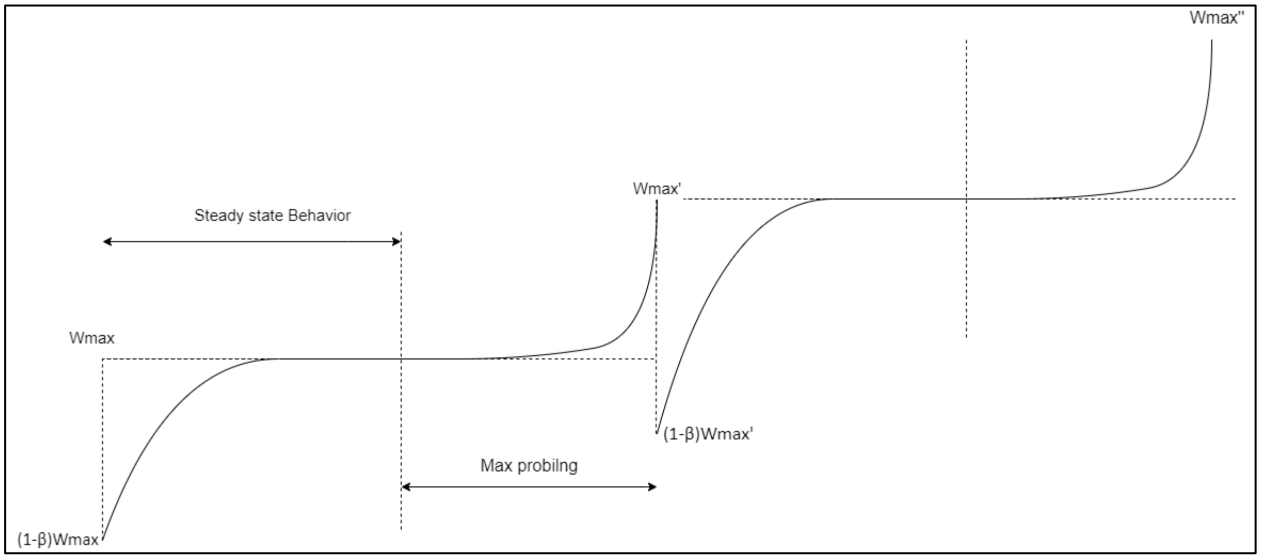

FIG 3 : WINDOW GROWTH AND REDUCTION FOR TCP CUBIC (REFERENCE TAKEN FROM [1])

Fast Convergence:

To ensure fast convergence, following is included in the protocol:

1. Wlast_max is updated to be the last maximum window size where the congestion happened.
2. The current Wmax is the maximum window size when the congestion happened.
   For instance in the figure above, Wlast_max = Wmax' and Wmax= Wmax''.
3. If Wlast_max< Wmax, then the plateau is achieved at Wmax.
4. If Wlast_max> Wmax, then the Wmax is reduced further to the value= Wmax(2- β)/2. This is done to release more bandwidth for competing flows and ensure fairness. The plateau by this flow is achieved earlier and time taken by it to increase the window size is increased thereby providing opportunity to other flows to catch up the window size.

Deterministic Loss Model for TCP CUBIC Analysis:

Assumptions:

1. The packet loss probability is p and packet losses are random and independent to each other.
2. The flow operates only in the concave region.
3. Packet loss happens due to buffer overflow or other reasons which prevents the receiver from acknowledging the packets received.
4. Congestion happens due to only one packet loss at one time and action is taken for that loss event.
5. Probability of number of packets lost follows a geometric law.

Metrics used:

With w as the initial window size, following metrics are used:

1. W(r,w) - window size upon acknowledging r packets
2. T(r,w) – time taken by r packets to be sent and acknowledged
3. Rmax(w)- number of packets acknowledged before a packet loss happens

Let the probability of acknowledging a packet successfully is:
q=1-p
If (r-1) packets have been acknowledged, then the probability mass function R is an independent and identically distributed random variable defined as:

$$P(R=r)= q^{r-1}p = (1-p)^{r-1}p \qquad (5)$$

where r varies from 1 to Rmax+1.
Time period of a cycle= T(R,w) and window size when congestion happens=W(R,w).
Considering a Markov chain consisting of the states w_n where n is the index for each RTT. The evolution of w_n and T_n is given by:

$$w_{n+1} = W(R_n, w_n) \qquad (6)$$
$$T_{n+1} = T(R_n, w_n) \qquad (7)$$

The transition probabilities for the Markov chain are derived from equations (5), (6) and (7). W(r,w) and T(r,w) are independent and identically distributed random variables. The expectation of T(r,w) is given as E[T] and expectation of random vatiable R is given as E[R]. The throughput x is defined as :

$$x=\frac{E[R]}{E[T]} \tag{8}$$

Let Wmax=ω, therefore in one cycle the window increases from w= (1-β) ω to ω. Upon completion of one cycle, i.e., on reaching ω, the state changes from w_i to w_{i+1}.
Since expectation of a geometric distribution is given as 1/p, hence

$$E[R]=1/p \tag{9}$$

Alternatively, we can derive it using the definition of expectation:

$$E[R]=\sum rP(R=r)$$

$$E[R]=\sum_{k=1}^{\infty} r(1-p)^{r-1}p$$

$$\sum_{k=1}^{\infty}(1-p)^{r}=\frac{1}{p}$$

$$\frac{d}{dp}\sum_{k=1}^{\infty}(1-p)^{r}=\frac{d}{dp}\frac{1}{p}$$

$$\sum_{k=1}^{\infty} r(1-p)^{r-1}(-1)=-\frac{1}{p^2}$$

$$\sum_{k=1}^{\infty} r(1-p)^{r-1}p=\frac{1}{p}$$

At (i+1)th cycle, the number of packets transmitted before loss happens is R_{i+1}:

$$E[R]=1/p$$

From equation (2) we know that

$$K=\sqrt[3]{\frac{\beta\, Wmax}{C}}$$

The window size after 1/p packets have been transmitted when window grows from w is given as:

$$W(1/p, w)=W_{max} \tag{10}$$

Time taken to reach W_{max} is given by using eqn (1):

$$W_{max} = C(t-K)^3 + W_{max}$$

$$t=K$$

Therefore

$$E[T]= K \tag{11}$$

We can re-write equation (1) in terms of RTT as follows:

$$W(t)= C(t\text{RTT}-\sqrt[3]{\frac{\beta\,\text{Wmax}}{C}})^{3} + \omega$$

To calculate the amount of transmitted packets in each cycle, we find the area under the curve as in Fig.4 This value is equal to E[R]. At time t, W(t)= ω. Hence t= $\frac{K}{RTT}$.

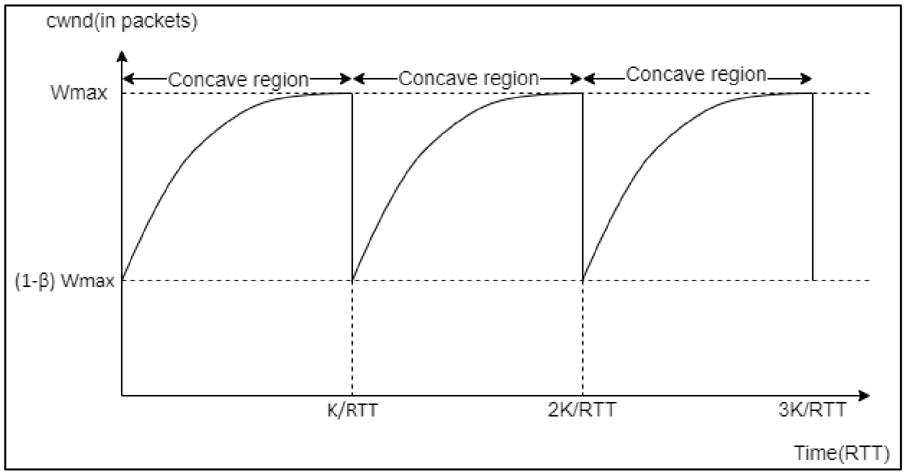

FIG 4: CUBIC GROWTH FUNCTION UNDER PERIODIC LOSS EVENTS (REFERENCE TAKEN FROM [2])

$$\frac{1}{p} = \int_{0}^{\frac{K}{RTT}} W(t)dt \qquad (12)$$

$$\frac{1}{p} = \int_{0}^{\frac{K}{RTT}} [\text{C}(\text{tRTT} - \sqrt[3]{\frac{\beta\,\omega}{C}})^{3} \ + \ \omega]dt$$

$$\frac{1}{p} = \frac{\text{C(tRTT−K)}^{4}}{\text{4 RTT}} \ + \ \omega \text{t}\Big|_{0}^{K/RTT}$$

$$\frac{1}{p} = \frac{C}{\text{4 RTT}}(\frac{K}{RTT}\text{RTT} - \text{K})^{4} \ + \ \omega\frac{K}{RTT} - \frac{C}{\text{4 RTT}}(0 - \text{K})^{4} - \omega.0$$

$$\frac{1}{p} = 0+ \omega\frac{K}{RTT} - \frac{C}{\text{4 RTT}}\text{K}^{4}$$

$$\frac{RTT}{p} = \ \omega\text{K} - \frac{C}{4}\text{K}^{4}$$

$$\frac{RTT}{p} = \ \omega\sqrt[3]{\frac{\beta\,\omega}{C}} - \frac{C}{4}\left(\sqrt[3]{\frac{\beta\,\omega}{C}}\right)^{4}$$

$$\frac{RTT}{p} = \ \omega\sqrt[3]{\frac{\beta\,\omega}{C}} - \frac{C}{4}\left(\sqrt[3]{\frac{\beta\,\omega}{C}}\right)^{4}$$

$$\frac{RTT}{p} = \ \frac{\omega^{\frac{4}{3}}\beta^{\frac{1}{3}}}{C^{\frac{1}{3}}} - \frac{\omega^{\frac{4}{3}}\beta^{\frac{4}{3}}}{4C^{\frac{1}{3}}}$$

$$\frac{RTT}{p} = \frac{\omega^{\frac{4}{3}}\beta^{\frac{1}{3}}}{C^{\frac{1}{3}}} \left[1-\frac{\beta}{4}\right]$$

$$\frac{RTT}{p}\frac{C^{\frac{1}{3}}}{\beta^{\frac{1}{3}}}\frac{4}{4-\beta} = \omega^{\frac{4}{3}}$$

$$\left(\frac{RTT}{p}\right)^3 \frac{C}{\beta}\left(\frac{4}{4-\beta}\right)^3 = \omega^4$$

$$\omega = \sqrt[4]{\left(\frac{RTT}{p}\right)^3 \frac{C}{\beta}\left(\frac{4}{4-\beta}\right)^3} \tag{13}$$

Expected size of window can now be found as:
$E[W_{CUBIC}]$=Total number of packets transmitted in one cycle/ Cycle time

$$E[W_{CUBIC}]=\frac{RTT}{pK}$$

$$=\frac{RTT}{p\sqrt[3]{\frac{\beta\,\omega}{C}}}$$

$$=\frac{RTT}{p}\frac{C^{\frac{1}{3}}}{\omega^{\frac{1}{3}}\beta^{\frac{1}{3}}}$$

$$=\frac{RTT}{p}\frac{C^{\frac{1}{3}}}{\left(\sqrt[4]{\left(\frac{RTT}{p}\right)^3\frac{C}{\beta}\left(\frac{4}{4-\beta}\right)^3}\right)^{\frac{1}{3}}\beta^{\frac{1}{3}}}$$

$$=\frac{RTT}{p}\frac{C^{\frac{1}{3}}}{\left(\frac{RTT}{p}\right)^{\frac{3}{12}}\left(\frac{C}{\beta}\right)^{\frac{1}{12}}\left(\frac{4}{4-\beta}\right)^{\frac{3}{12}}\beta^{\frac{1}{3}}}$$

$$=\frac{RTT}{p}\frac{C^{\frac{1}{3}}p^{\frac{1}{4}}\beta^{\frac{1}{12}}(4-\beta)^{\frac{1}{4}}}{RTT^{\frac{1}{4}}C^{\frac{1}{12}}4^{\frac{1}{4}}\beta^{\frac{1}{3}}}$$

$$=\frac{C^{\frac{1}{4}}\,RTT^{\frac{3}{4}}(4-\beta)^{\frac{1}{4}}}{4^{\frac{1}{4}}\beta^{\frac{1}{4}}p^{\frac{3}{4}}}$$

$$=\left(\frac{RTT}{p}\right)^3\frac{C}{\beta}\left(\frac{4-\beta}{4}\right)$$

$$=\sqrt[4]{\left(\frac{RTT}{p}\right)^{3}\frac{C}{\beta}\left(\frac{4-\beta}{4}\right)}$$

$$E[W_{CUBIC}]=\sqrt[4]{\left(\frac{RTT}{p}\right)^{3}\frac{C}{\beta}\left(\frac{4-\beta}{4}\right)} \tag{14}$$

Equation (14) is different from paper [2] by a factor of 4 in the denominator. However, it is same as given in papers [1] and [3].

Throughput is then given as:

$$x=\frac{E[R]}{E[T]}=\frac{1}{p*K}=\frac{E[WCUBIC]\,K}{RTT*K}=\frac{E[WCUBIC]}{RTT}$$

Validation of Analytical Model:

The validation of deterministic loss model is carried out using following code in MATLAB:

```
function cubic_reno(RTT,B,C)
Wcubic=zeros(8,1);
Wreno=zeros(8,1);
W=zeros(8,1);
p=[10.^-8, 10.^-7, 10.^-6, 10.^-5, 10.^-4, 10.^-3, 10.^-2, 10.^-1];
for i=1:length(p)
   Wreno(i)=1.31/(p(i))^0.5;
   Wcubic(i)=(((RTT/(p(i)))^3)*(C*(4-B)/(4*B)))^0.25;
   % If C=0.4, B=0.2, the WCubic is given by following
       %Wcubic(i)=1.17.*((RTT/p(i))^0.75);
   if(Wreno(i)<Wcubic(i))
      W(i)=Wcubic(i);
   else
      W(i)=Wreno(i);
   end
end
semilogx(p,W);
hold on;
semilogx(p,Wreno);
set(gca, 'YScale', 'log');

end
```

In the above code, the average window size of TCP reno is calculated using the value of C=1.31 from [6] and stored in the variable Wreno:

$$E[Wreno]=1.31/\sqrt[2]{p} \tag{15}$$

We calculate the average window size using equation (14) and store it in Wcubic. Since in TCP CUBIC, the protocol operates in TCP reno mode if Cubic calculated window size is less than that attained by TCP reno in that time, we put the condition to check both the windows and set greater of the two as the final window size of TCP CUBIC in variable W.

a) Response function of TCP CUBIC and TCP reno for different RTT:

The evolution of average window size is compared with loss rate p for TCP CUBIC and TCP Reno by varying RTT=10, 50, 100, 200 ms.

Simulation Results: We attain Fig. 5, 6, 7 and 8 upon simulation when calling the function cubic_reno(RTT,B,C) with RTT=10ms, β=0.2 and C=0.4.

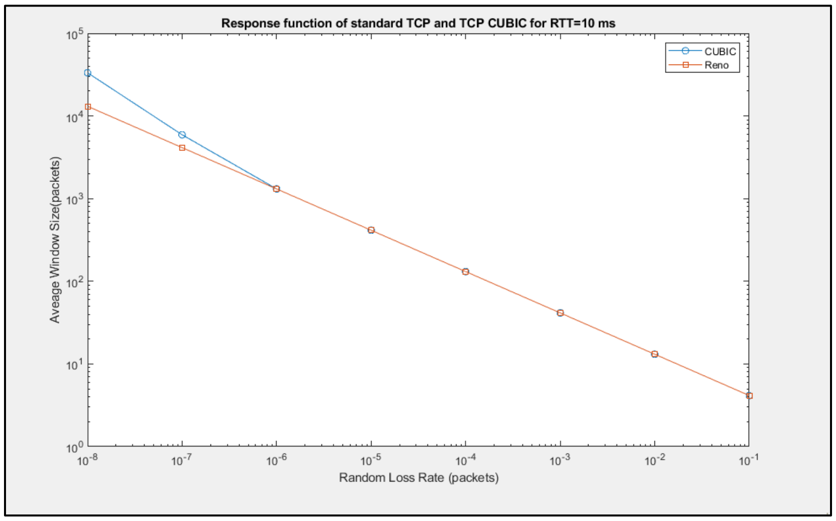

FIG 5: RESPONSE FUNCTION OF STANDARD TCP AND TCP CUBIC FOR RTT=10MS

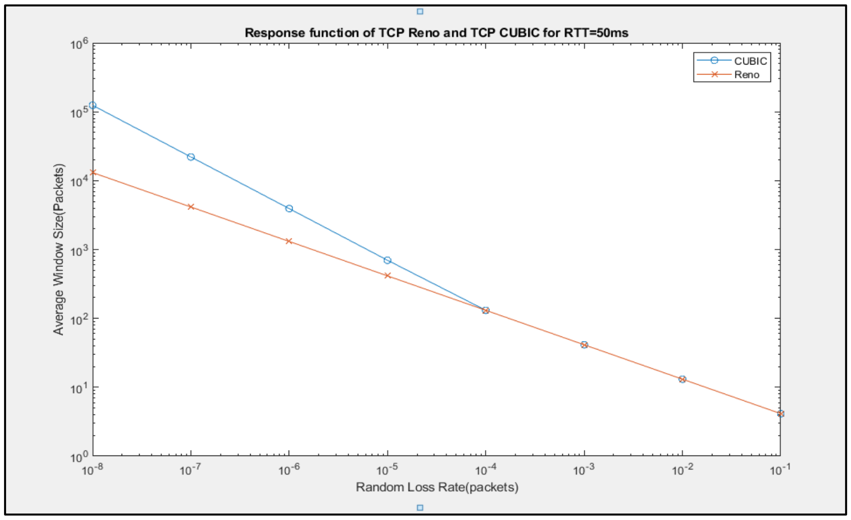

FIG 6 : RESPONSE FUNCTION OF STANDARD TCP AND TCP CUBIC FOR RTT=50MS

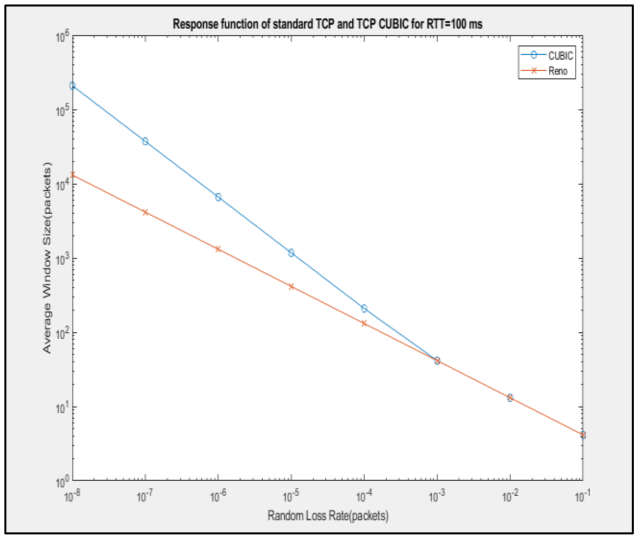

FIG 7: RESPONSE FUNCTION OF STANDARD TCP AND TCP CUBIC FOR RTT=100MS

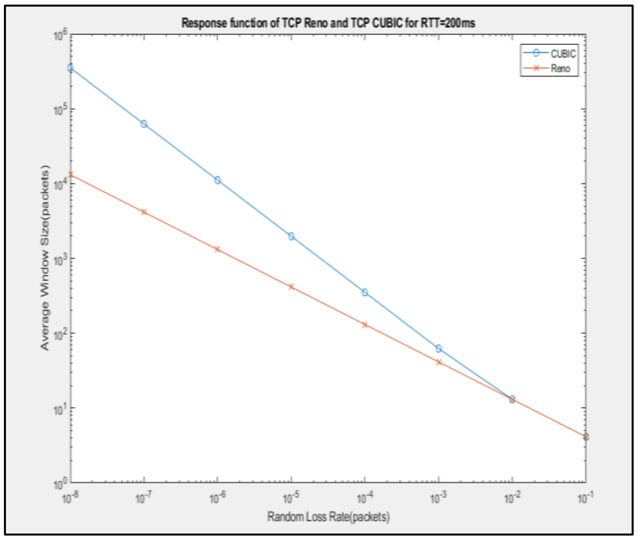

FIG 8: RESPONSE FUNCTION OF STANDARD TCP AND TCP CUBIC FOR RTT=200MS

Fig. 9 has been taken from [2].

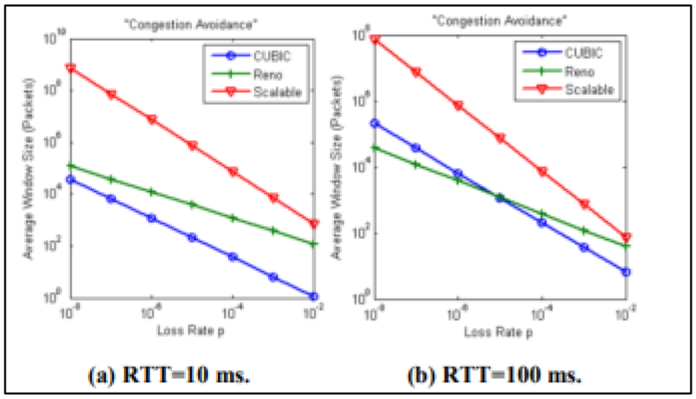

FIG 9: RESPONSE FUNCTION OF STANDARD TCP AND TCP CUBIC FOR RTT=10MS AND 100 MS (FROM [2])

Fig. 10 have been taken from [1].

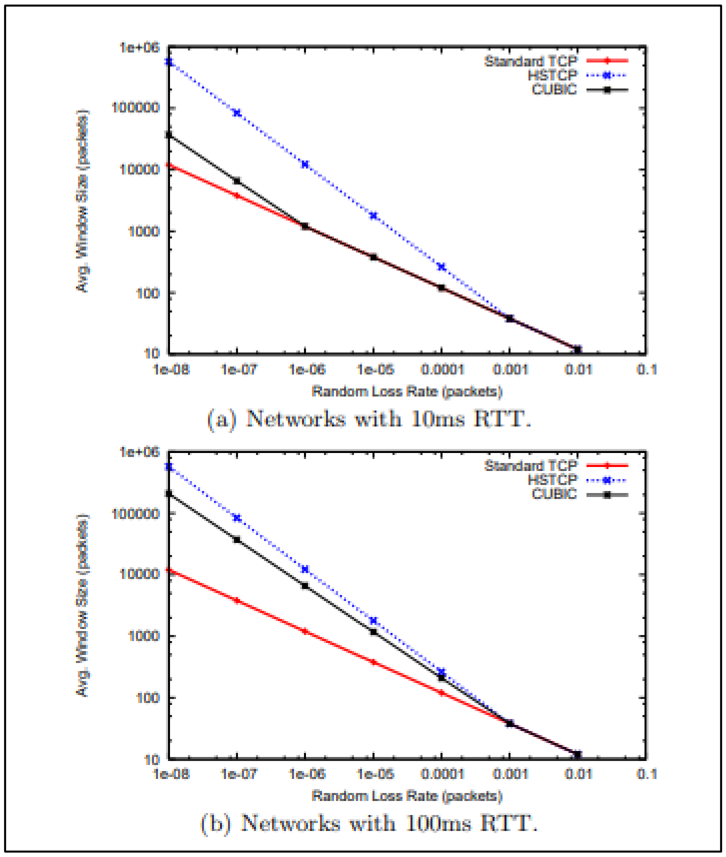

FIG 10 : RESPONSE FUNCTION OF STANDARD TCP AND TCP CUBIC FOR RTT=10MS AND 100MS (FROM [1])

Explanation: There is similarity between the results of this paper and [1] whereas its different from [2]. This is because TCP cubic follows TCP reno if its calculated window size is less than TCP reno. Once the window growth is more, the window grows as per TCP CUBIC growth function. At this point, TCP CUBIC gets more aggressive than TCP reno. Hence there is a deviation from the straight curve. With less RTT, TCP CUBIC behaves similar to TCP Reno as it is designed for networks with high bandwidth delay product. TCP Reno average window size is independent of RTT, hence we get same curve in all the four graphs when RTT=10,50,100 and 200ms. However for TCP CUBIC, as RTT increases the average window size starts deviating from TCP reno for more value of p. Since E[W_{CUBIC}] is inversely proportional to packet loss rate, we achieve higher window size for lesser p.

b) Response function of TCP CUBIC for different C:

The same model is simulated for different values of C (=0.01,0.04,0.4 and 4) and window growth is observed.

Simulation Results: Fig.11 and 12 are obtained when simulating the model with different values of C, RTT=10ms and 100 ms and β=0.2:

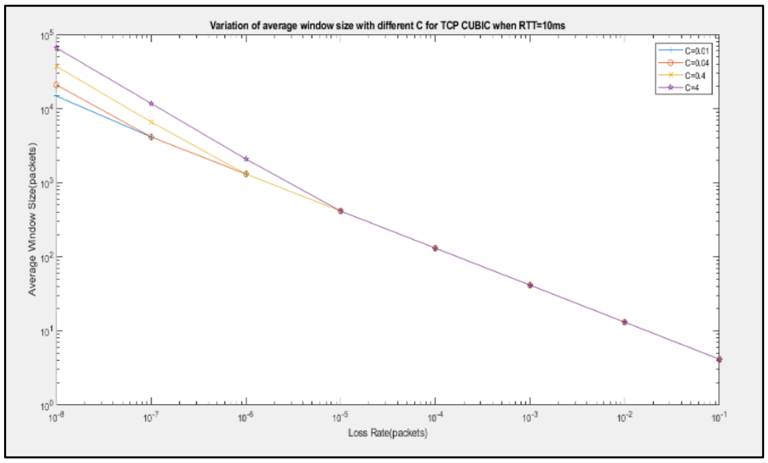

FIG 11 :RESPONSE FUNCTION OF TCP CUBIC FOR DIFFERENT C WHEN RTT=10MS

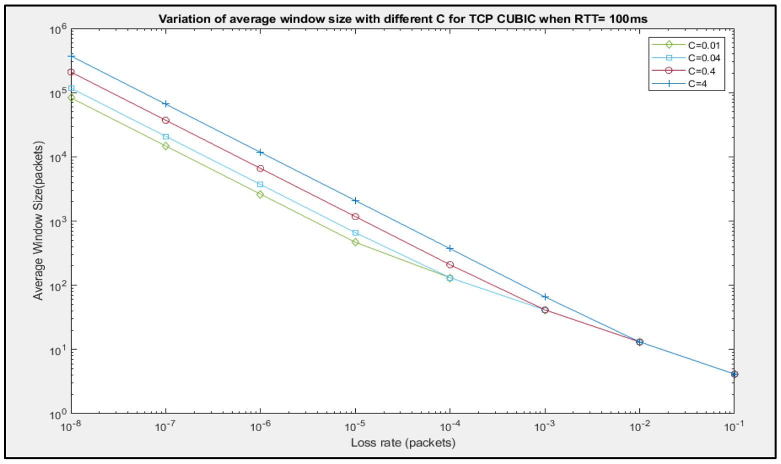

FIG 12: RESPONSE FUNCTION OF TCP CUBIC FOR DIFFERENT C WHEN RTT=100MS

Fig. 13 is taken from [2]:

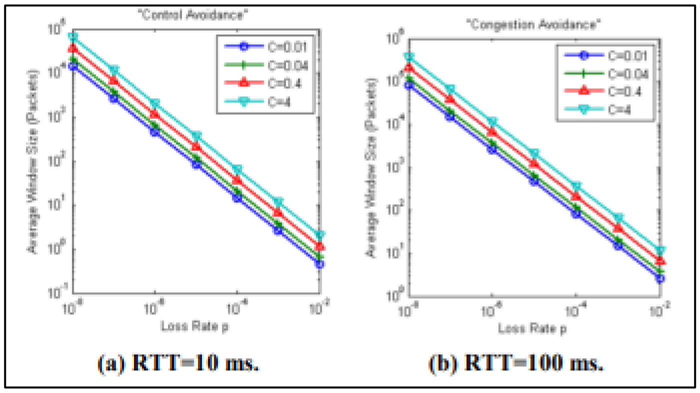

FIG 13: RESPONSE FUNCTION OF TCP CUBIC FOR DIFFERENT C WHEN RTT=10MS AND 100 MS (FROM [2])

Explanation:

The nature of the graph in this paper and [2] is different as TCP friendly region is considered while simulation in this paper. However, the result "upon increasing C, average window size also increases" is same in both the papers. Upon varying C, that is the growth factor, the average window size increases since they are directly proportional as can be seen in equation(14). This is quite intuitive because if we increase the growth factor then the window size attained should be more. This result is valid for both RTT=10 ms and 100 ms. Only for RTT=100ms, the average window size is more as established from previous result.

c) **Response function of TCP CUBIC for different RTT:**

RTT is set at 0.01,0.05,0.1,0.2 and 0.5 sec and average window size for various packet loss rate is observed.

Simulation Result: Upon varying RTT we get Fig. 14 for average window size vs packet loss:

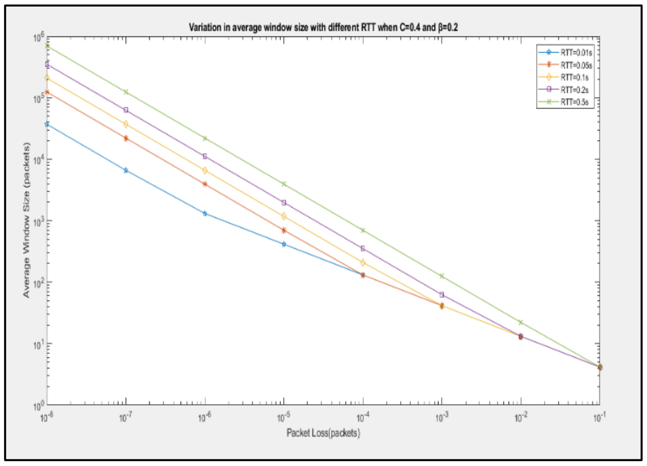

FIG 14: RESPONSE FUNCTION OF TCP CUBIC FOR DIFFERENT RTT WHEN C=0.4 AND B=0.2

Explanation:

As can be seen from equation (14), as RTT increases the average window size increases since TCP CUBIC performs much better in high BDP networks. Upon increasing RTT from 10 ms to 100 ms the average window size increases by 6 times at packet loss rate=10^{-8} and increases by 19 times when RTT=500ms.

d) Response function of TCP CUBIC for different β:

RTT =100 ms, C=0.4 and β is set at 0.2,0.5 and 0.9 and average window size for various packet loss rate is observed.

Simulation Result: Upon varying β Fig. 15 is obtained:

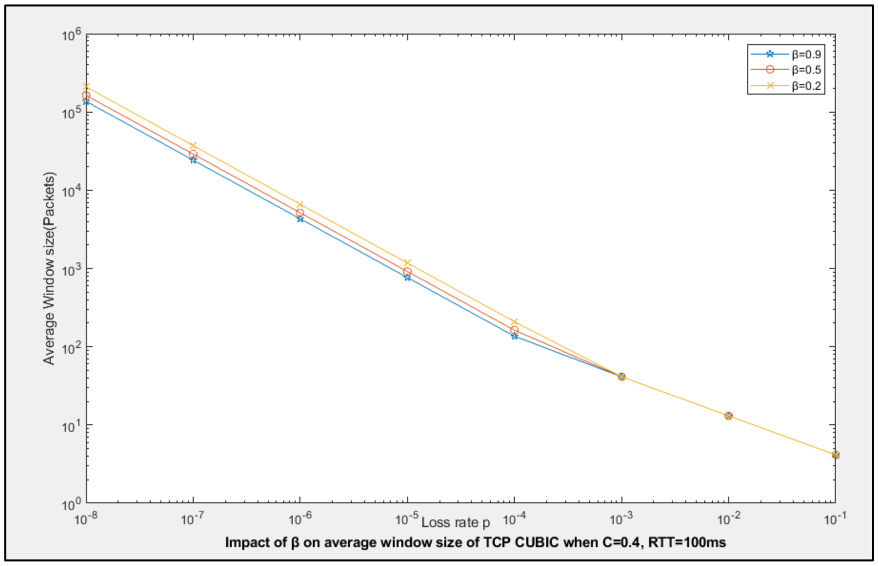

FIG 15: RESPONSE FUNCTION OF TCP CUBIC FOR DIFFERENT Β WHEN C=0.4 AND RTT=100 MS

Explanation: As β is decreased, then upon congestion loss, the window size is reduced by a lesser factor and hence average window size is more. The window size increases by a factor of 1.5 when β is reduced from 0.9 to 0.2 at loss rate of 10^{-8}.

Steady State Throughput Model of TCP CUBIC:

This model determines the steady state throughput of TCP CUBIC in wireless networks where packet losses can be random using a Markov chain model. Both congestion and random losses are considered to arrive at the expression of throughput. Transition probabilities, steady state probabilities and average throughput are calculated by quantization of the window size.

This model is basically developed in following steps:

1. A Bottleneck capacity of C is considered where congestion happens
2. Probability density function of time duration between two losses is defined.
3. Markov chain is formulated.
4. Transition probabilities are calculated using pdf in (2)
5. Steady state probability of Markov chain states is evaluated.
6. Average throughput is then calculated using above results.

1) **Losses:** Loss in a network may happen due to congestion or random losses which are more common in wireless networks.
Let the capacity of the bottleneck link be C bits/sec. Congestion loss will occur when the transmission rate exceeds the bottleneck capacity C. We assume that RTT is a constant. Therefore the maximum window size W (bits) that can be achieved is :

$$W=C.RTT \tag{16}$$

When window size reaches the value of W, congestion loss will happen.
For random losses, we assume that it follows Poisson distribution with rate λ. The time duration between two losses be given by $\tau_{loss.}$ It is a random variable with exponential distribution. This is because, if the number of arrivals are Poisson distributed then the interarrival times are exponentially distributed. The pdf of τ_{loss} is given as:

$$f(\tau_{loss})= \lambda \exp(-\lambda\ \tau_{loss}) \tag{17}$$

where $\tau_{loss}>0$
The cumulative distribution function is given by:

$$P(\tau_{loss}\leq T)=1- \exp(-\lambda T) \tag{18}$$

The probability of loss happening between time intervals T_1 and T_2 is given by:

$$\begin{aligned} P(T_1 \leq\tau_{loss}\leq T_2) &= P(\tau_{loss}\leq T_2)- P(\tau_{loss}\leq T_1) \\ &=1- \exp(-\lambda T_2)- (1- \exp(-\lambda T_1) \\ &=\exp(-\lambda T_1)- \exp(-\lambda T_2) \end{aligned} \tag{19}$$

2) **Metrics Used for model formulation:**
τ= Elapsed time from last congestion or random loss occurrence
x=window size when the loss happens
α= window growth factor
β=multiplicative decrease factor
In this analysis we assume window reduces to βx upon congestion instead of (1-β)x in the previous analysis.
w(x, τ)= window size as a function of x and τ.
The growth function is similar to the previous analysis (eqn (1)) and is given as:

$$w(x, \tau)= \alpha\ (\tau - K)^3 + x \tag{20}$$

$$K=\sqrt[3]{\frac{(1-\beta)x}{\alpha}} \tag{21}$$

D(x,y)=Time duration in which window grows from βx to y without any further loss since the window reduced from the value of x.

$$y = \alpha\left(t - \sqrt[3]{\frac{(1-\beta)x}{\alpha}}\right)^3 + x$$

$$t = D(x,y) = \sqrt[3]{\frac{y-x}{\alpha}} + \sqrt[3]{\frac{(1-\beta)x}{\alpha}} \tag{22}$$

3) Markov Chain formulation:

Congestion window can vary from 0 to W. We use the concept of quantization to find the states of the Markov chain. In quantization, continuous range of values are mapped to discrete values.

We take a value N for the number of states. Th range (0,W] is equally divided into N intervals. If the value of N is large, the model becomes more precise.

For instance, if W=1000 and N=100, then we get 100 equal sized intervals of length 1000/100=10.

From Fig. 16**,** we see that the ith interval starts from (i-1)W/N and ends at iW/N. If the congestion window size happens to lie in the ith interval, the midpoint of that interval is considered to be its value.

$$\text{Midpoint} = \frac{1}{2}\left[(i-1)W/N + iW/N\right]$$

$$= (i-0.5)W/N \tag{23}$$

This midpoint is denoted as a_i. In this way the entire continuous range is mapped to discrete values. The values of a_i ($=a_1, a_2, a_3, \ldots, a_N$) forms the value of the ith state of the Markov chain.

In Fig. 16, we have considered N=100 and W=1000, therefore the Markov states 1,2, 3,…100 have values : 5,15, 25,….995. Suppose congestion occurs at 814. As it lies between 810 and 820, it is in the 82nd interval which is also its state and has value of 815.

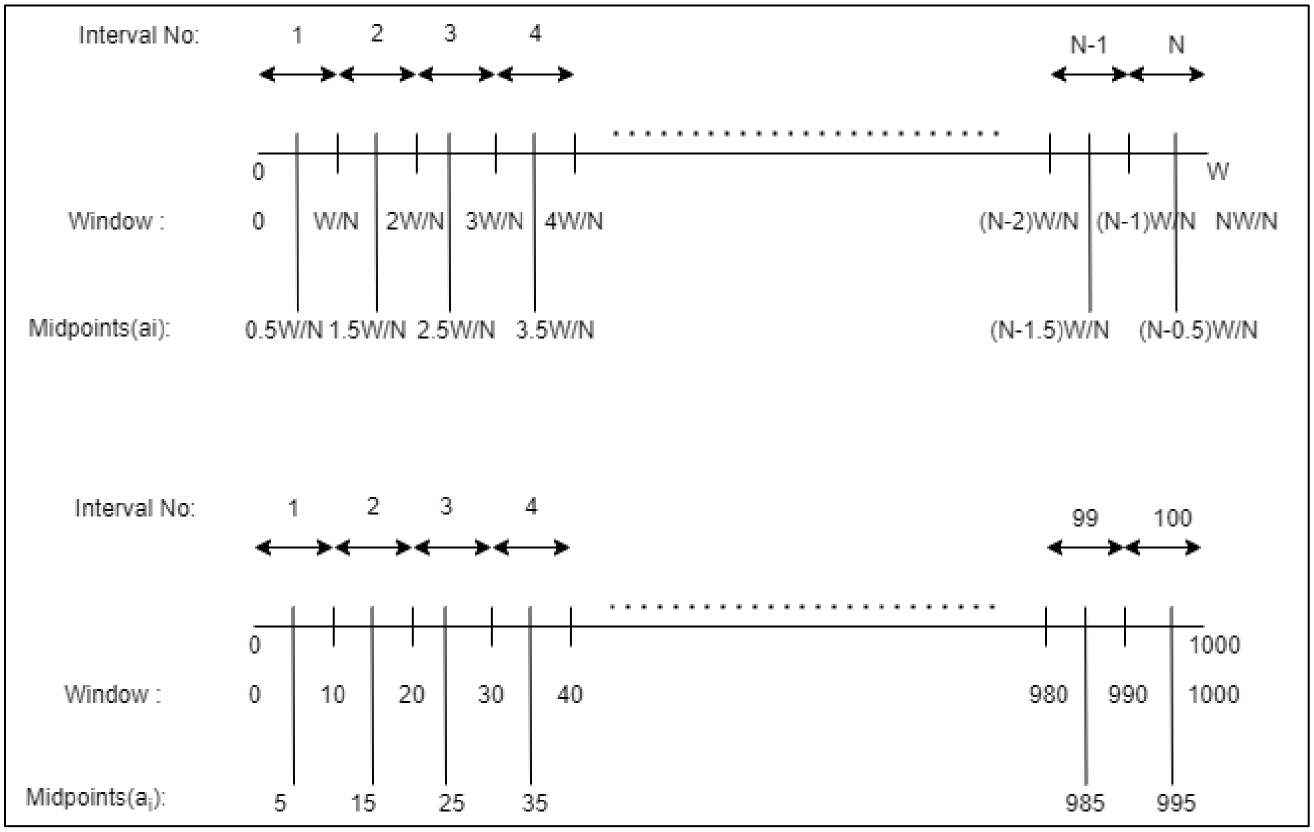

FIG 16: QUANTIZATION OF WINDOW SIZE TO FORMULATE MARKOV CHAIN

Whenever there is a loss, window size is reduced and Markov chain is observed as the state transits from i^{th} to j^{th} state.

k=1,2,3…… denote the kth reduction.

x_k= Window size just before kth reduction. It will obviously lie in one of the N intervals.

That interval is its state and is denoted by X_k.

The midpoint which denotes the value of that state is given by $\tilde{x}_k$ and belongs to the set $\{a_1, a_2, a_3, \ldots, a_N\}$.

Let x_k lie in the ith interval, hence is ranges between ((i-1)W/N, iW/N].

$$\tilde{x}_k = (i-0.5)W/N$$

$$X_k = i$$

τ_k = Time duration between kth and (k+1)th window reduction

From any particular interval, window can grow to maximum W. Hence τ_k can have maximum value of $D(x_k, W)$.

The state sequences are $X_1, X_2, X_3, X_4, X_5, \ldots X_k$

The congestion window at (k+1)th loss event depends only on the congestion window at kth loss event and τ_k. Hence x_{k+1} is given as:

$$x_{k+1} \qquad = w(x_k, \tau_k)$$

$$=\alpha\left(\tau_k - \sqrt[3]{\frac{(1-\beta)x}{\alpha}}\right)^3 + x_k \text{ where } \tau_k \leq D(x_k, W) \tag{24}$$

Hence we see that x_{k+1} is independent of the previous window size and depends only on the present window size.

$$P(x_{k+1} \mid x_k, x_{k-1}, x_{k-2}, \ldots, x_1) = P(x_{k+1} \mid x_k) \tag{25}$$

Similarly the mapped values $\tilde{x}_{k+1}$ and the states X_{k+1} are independent of the past and given by:

$$P(\tilde{x}_{k+1} \mid \tilde{x}_k, \tilde{x}_{k-1}, \tilde{x}_{k-2}, \ldots, \tilde{x}_1) = P(\tilde{x}_{k+1} \mid \tilde{x}_k) \tag{26}$$

$$P(X_{k+1} \mid X_k, X_{k-1}, X_{k-2}, \ldots, X_1) = P(X_{k+1} \mid X_k) \tag{27}$$

The states $X_1, X_2, X_3, X_4, X_5, \ldots, X_k, X_{k+1} \ldots$ form a Markov chain.

4) **State Transition Probabilities:**

We now calculate state transition probabilities P_{ij}, where $j \neq N$. There is a transition from ith state to jth state. This means kth reduction happens when window size lies within $((i-1)W/N, iW/N]$ range. Hence $\tilde{x}_k$ is mapped to value of a_i. Upon reduction window size is βa_i. Window grows to x_{k+1} when the (k+1)th loss happens. This implies (k+1)th reduction happens when window size lies within $((j-1)W/N, jW/N]$. Hence $\tilde{x}_{k+1}$ is mapped to value of a_j.

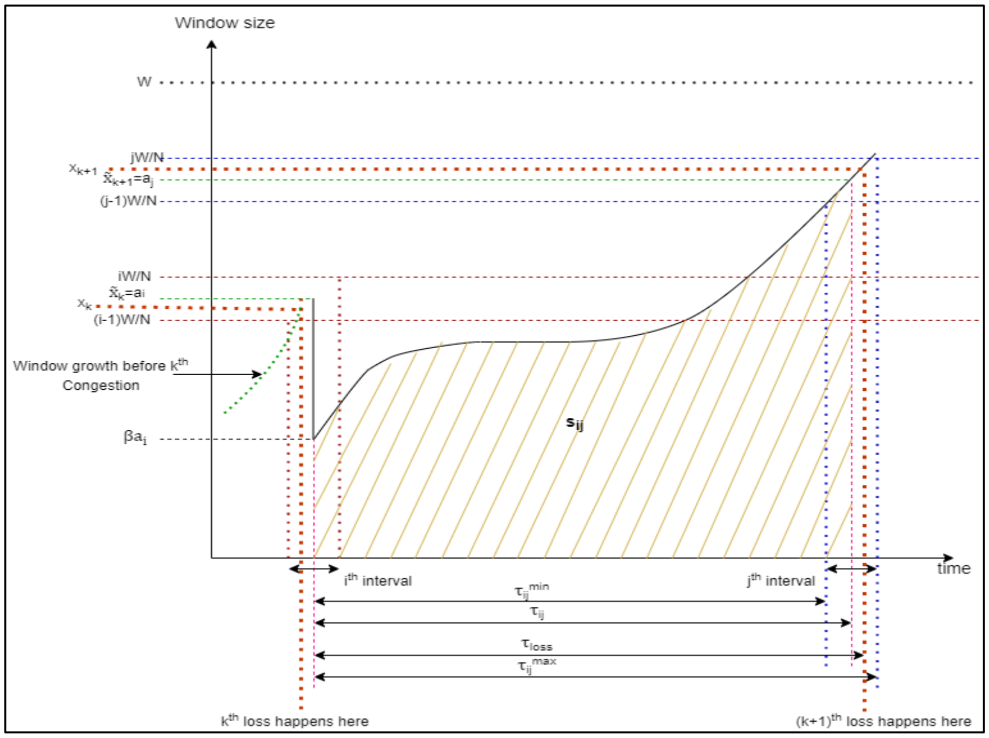

FIG 17: CONGESTION EPOCH AND ASSOCIATED VARIABLES (REFERENCE TAKEN FROM [4])

When (k+1)th loss happens, the window size cannot be less than βa_i. Hence transition probability to any states less than βa_i is not possible

$$P_{ij}=0, \text{ when } jW/N< \beta a_i \tag{28}$$

We can substitute the value of a_i as (i-0.5)W/N in equation (28)

$$j<\beta(i-0.5) \tag{29}$$

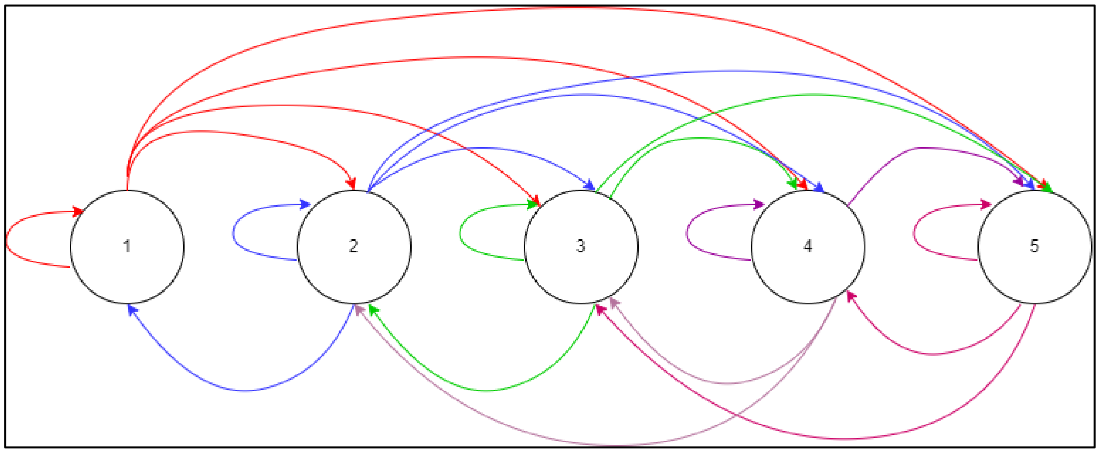

FIG 18: MARKOV CHAIN STATES FOR N=5 AND B=0.5

Also, as jth interval lies between ((j-1)W/N, jW/N], the random variable τ_{loss} lies between τ_{ij}^{min} and τ_{ij}^{max}.

$$\tau_{ij}^{min}< \tau_{loss} \le \tau_{ij}^{max} \tag{30}$$

τ_{ij}^{min} is the minimum time taken to reach jth state which is equal to the time taken for window to grow from βa_i to (j-1)W/N.

$$\tau_{ij}^{min} \quad =D(a_i, (j-1)W/N)^+ \tag{31}$$

$$= \left(\sqrt[3]{\frac{(j-1)W/N-a_i}{\alpha}} + \sqrt[3]{\frac{(1-\beta)a_i}{\alpha}}\right)^+$$

where $(a)^+= \max(a,0)$

Similarly, τ_{ij}^{max} is the maximum time taken to reach jth state which is equal to the time taken for window to grow from βa_i to jW/N.

$$\tau_{ij}^{max} \quad =D(a_i, jW/N) \tag{32}$$

$$= \left(\sqrt[3]{\frac{jW/N-a_i}{\alpha}} + \sqrt[3]{\frac{(1-\beta)a_i}{\alpha}}\right)$$

Now we can calculate state transition probabilities for $jW/N \geq \beta a_i$,i.e., $j \geq \beta(i-0.5)$ and $j \neq N$ using the cumulative distribution function of τ_{loss} as given in equation (19).

$$P_{ij} = P(\tau_{ij}^{min} < \tau_{loss} \leq \tau_{ij}^{max}) \quad (33)$$

$$= \exp(-\lambda\, D(a_i, (j-1)W/N)^+) - \exp(-\lambda\, D(a_i, jW/N))$$

$$= \exp\left(-\lambda \left(\sqrt[3]{\frac{(j-1)W/N-a_i}{\alpha}} + \sqrt[3]{\frac{(1-\beta)a_i}{\alpha}}\right)^+\right) - \exp\left(-\lambda \left(\sqrt[3]{\frac{jW/N-a_i}{\alpha}} + \sqrt[3]{\frac{(1-\beta)a_i}{\alpha}}\right)\right) \quad (33a)$$

For j=N, we find state transition probability by subtracting the sum of all transition probabilities from state i to j where j=1,2,3,…, N-1 from 1. This is done to ensure the sum of transition probability from state i to state j is 1.

$$P_{iN} = 1-\sum_{j=1}^{N-1} P_{ij} \quad (33b)$$

Fig. 18 shows Markov chain for N=5 and β=0.5. In the original paper[4], transition to its own state is not considered. The simulation results show that there exists a non zero probability of transitioning to its own state after a loss happens.

5) Stationary Distribution:

The stationary distribution of Markov chain can be denoted as $\pi_1, \pi_2, \pi_3, \ldots \pi_N$, where π_i is the stationary probability of the ith state. Stationary distribution can be found out by solving following equations:

$$\sum_{i=1}^{N} \pi_i P_{ij} = \pi_j \quad (34)$$

and

$$\sum_{i=1}^{N} \pi_i = 1 \quad (35)$$

We have already found the time taken to transit from i state to j state in equation (30). If N is large we can find the average time τ_{ij} to transit from state i to state j by considering the time taken to transit from window size of a_i to a_j. This means the time taken to reach the midpoint of the interval ((j-1)W/N, jW/N] from window size of βa_i is given by:

$$\tau_{ij} = D(a_i, (j-0.5)W/N)^+ \quad (36)$$

6) Throughput:

As shown in Fig. 17, let s_{ij} denote the area under congestion epoch shown by the shaded region.

$$s_{ij} = \int_0^{\tau_{ij}} w(a_i, t)\, d\tau \tag{37}$$

$$= \int_0^{\tau_{ij}} \left(\alpha \left(t - \sqrt[3]{\frac{(1-\beta)a_i}{\alpha}} \right)^3 + a_i \right) d\tau$$

$$= \frac{\alpha(t-L)^4}{4} + a_i t \Big|_0^{\tau_{ij}}$$

$$= a_i \tau_{ij} + \frac{\alpha}{4}\left((\tau_{ij} - L)^4 - L^4 \right) \tag{38}$$

$$\text{where } L = \sqrt[3]{\frac{(1-\beta)a_i}{\alpha}}$$

The total number of transmissions r_{ij} between state i and state j is given by:

$$r_{ij} = \frac{1}{RTT} \int_0^{\tau_{ij}} w(a_i, t) d\tau \tag{39}$$

$$= \frac{sij}{RTT}$$

To calculate the average normalized throughput, we let r(t) denote the total number of transmissions in time t. Hence the normalized throughput is given as:

$$\bar{x} = \frac{\lim_{t\to\infty} \frac{r(t)}{t}}{C} \tag{40}$$

Let $M_{ij}(t)$ denote the occurrence time of the ith state transition to jth state during time t.
If t is very large, then by the law of large numbers: $\lim_{t\to\infty} \frac{M_{ij}(t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} M_{ij}(t)}$ is equal to the occurrence probability of ith state transition to jth state. Using equation (16), $\bar{x}$ can be written as:

$$\bar{x} = \frac{\lim_{t\to\infty} \frac{r(t)}{t}}{\frac{W}{RTT}}$$

$M_{ij}(t)\, r_{ij}$ is the total number of transmissions when the ith state moves to jth state. Summing it over all i and j from 1 to N, gives the total number of transmissions between all the states. Similarly, $M_{ij}(t)\ \tau_{ij}$ is the total time taken for all the transitions from ith to jth state. Summing it over all i and j from 1 to N, gives the total time taken for all the transitions.

Hence $\lim_{t\to\infty} \frac{r(t)}{t}$ can be written as:

$$\lim_{t\to\infty}\frac{r(t)}{t} = \lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t) r_{ij}(t)}{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)\tau_{ij}} \tag{41}$$

$$\bar{x} = \frac{RTT}{W}\lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)\, r_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)\,\tau_{ij}} = \frac{1}{W}\lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)(r_{ij}.RTT)}{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)\,\tau_{ij}} = \frac{1}{W}\lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)\, s_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)\,\tau_{ij}}$$

The numerator and denominator can be divided by $\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}(t)$ to write it in terms of state transition probabilities

$$\bar{x} = \frac{1}{W}\lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\left(M_{ij}(t)/\sum_{p=1}^{N}\sum_{q=1}^{N} M_{pq}(t)\right)s_{ij}\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\left(M_{ij}(t)/\sum_{p=1}^{N}\sum_{q=1}^{N} M_{pq}(t)\right)\tau_{ij}\right)} = \frac{1}{W}\lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(P(X_{k+1}=j, X_k=i)\, s_{ij}\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(P(X_{k+1}=j, X_k=i)\tau_{ij}\right)} = \frac{1}{W}\lim_{t\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\pi_i P_{ij}\, s_{ij}\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\pi_i P_{ij}\tau_{ij}\right)} \tag{42}$$

Validation of Analytical model:

The above analytical model was simulated in MATLAB using below code:

```
function tp=analytical(lambda,C,RTT,a,b)

N=100;
num=0;
den=0;
P=zeros(N,N);
T=zeros(N,N);
S=zeros(N,N);
A=zeros(1,N);
row_sum=zeros(1,N);
tp=0;
W=C*RTT;

for j=1:N
   A(j)=(j-0.5).*W/N;
end
```

```
for i=1:N
   for j=1:N
      if j<b*(i-0.5)
         P(i,j)=0;
      elseif (j>=b*(i-0.5) && j~=N)
         P(i,j)=(exp(-lambda*(max(0,Time_elapsed(A(i),((j-1)*W/N),a,b))))-exp(-
lambda*Time_elapsed(A(i),(j*W/N),a,b)));
         row_sum(i)=row_sum(i)+P(i,j);
      else
         P(i,N)=1-row_sum(i);
   end
   end
end

for i=1:N
   for j=1:N
      T(i,j)= max(0,Time_elapsed(A(i),((j-0.5)*W/N),a,b));
   end
end

for i=1:N
   for j=1:N
      L=((1-b)*A(i)/a)^(1/3);
      S(i,j)= A(i)*T(i,j)+(a/4)*(((T(i,j)-L)^4)-L^4);
   end
end

[V,E] = eig(P');
M = V(:,1)';
M = M./sum(M);

for i=1:N
   for j=1:N
      num=num+M(i)*P(i,j)*S(i,j);
   end
end

for i=1:N
   for j=1:N
      den=den+M(i)*P(i,j)*T(i,j);
   end
end

tp=(num/(den*W));



function D=Time_elapsed(x,y,a,b)
  if(y-x<0)
   D=(((1-b)*x)/a)^(1/3)-(((x-y)/a)^(1/3));
  else
    D=(((y-x)/a)^(1/3))+((((1-b)*x)/a)^(1/3));
  end



function result
lambda1=[0.001,0.002,0.003,0.004,0.005,0.006,0.007,0.008,0.009,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1];
%a=[0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,2,3,4,5,6,7,8,9,10];
a=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,2,3,4,5,6,7,8,9,10];
b=[0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9]
tpp=zeros(1,length(b));

for k=1:length(b)
   tpp(k)=analytical(1,100000000,0.1,1000000,b(k));
end
semilogx(b,tpp);
```

Discussions on Code:

Three functions: analytical, Time_elapsed and results were made.

The function Time_elapsed is used to calculate the time taken for window size to grow from x to y. It takes x, y, a and b as its arguments where x is the window size just before congestion, y is the window size after some time t, a is the cubic growth factor and b is the multiplicative decrease factor.

There can be two cases y<x or y>x.

a. y<x

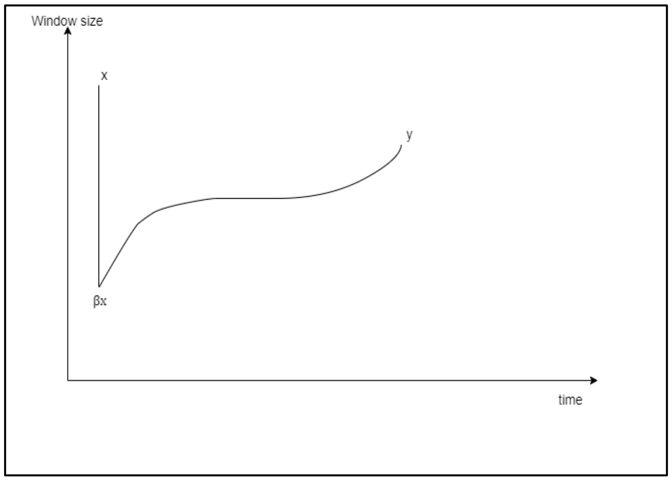

FIG 19: TCP CUBIC WINDOW GROWTH WHEN Y<X

In this case use equation:

$D(x,y)= \sqrt[3]{\frac{(1-\beta)x}{\alpha}} - \sqrt[3]{\frac{y-x}{\alpha}}$

b. y>x

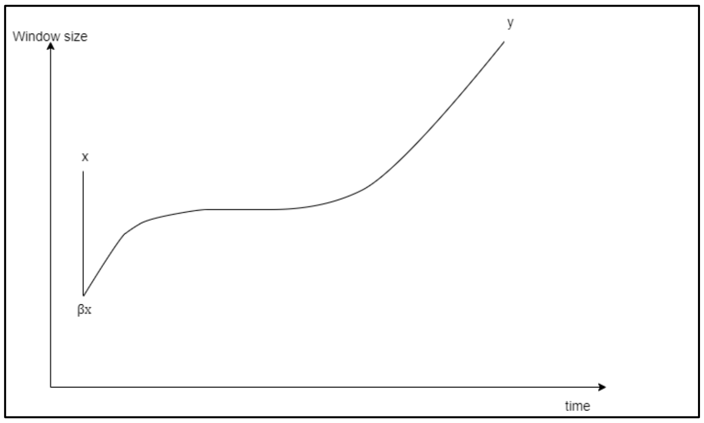

FIG 20: TCP CUBIC WINDOW GROWTH WHEN Y>X

In this case use equation: $D(x,y)= \sqrt[3]{\frac{y-x}{\alpha}} + \sqrt[3]{\frac{(1-\beta)x}{\alpha}}$

The function analytical takes lambda, C, RTT, a and b as its arguments where lambda is the loss rate, C is the bottleneck capacity, RTT is the round trip time. N is the number of intervals. We define N by N matrix P for storing state transition probabilities, N by N matrix T for storing average time duration of transition from state i to sate j, N by N matrix S for storing the value of area under the congestion epoch s_{ij} and 1 by N matrix A for storing the midpoints of intervals, i.e, the value of ai. First Pij is calculated. If j<b*(i-0.5) then Pij=0, else if j is not equal to N use equation (33a) to calculate Pij. Also store the combined row sum in a variable row_sum to calculate PiN. PiN is assigned the value of 1-row_sum to ensure the row sum is 1.

Now calculate the average time of transmission from state i to j using equation (36).

For calculating area under the congestion epoch s_{ij} use equation (38).

Finally calculate the stationary state distribution by finding the eigen vector of P' and store it in M matrix.

Find the sum of numerator and denominator in equation (42) to arrive at the normalized average throughput.

The function result is only used to find different trends in the throughput by varying lambda, a, b, RTT. etc.

Simulation Results:

a) Normalized average throughput under different bandwidth-delay product C.RTT and loss rate λ.
α= 1Mbps
β=0.5
N=100

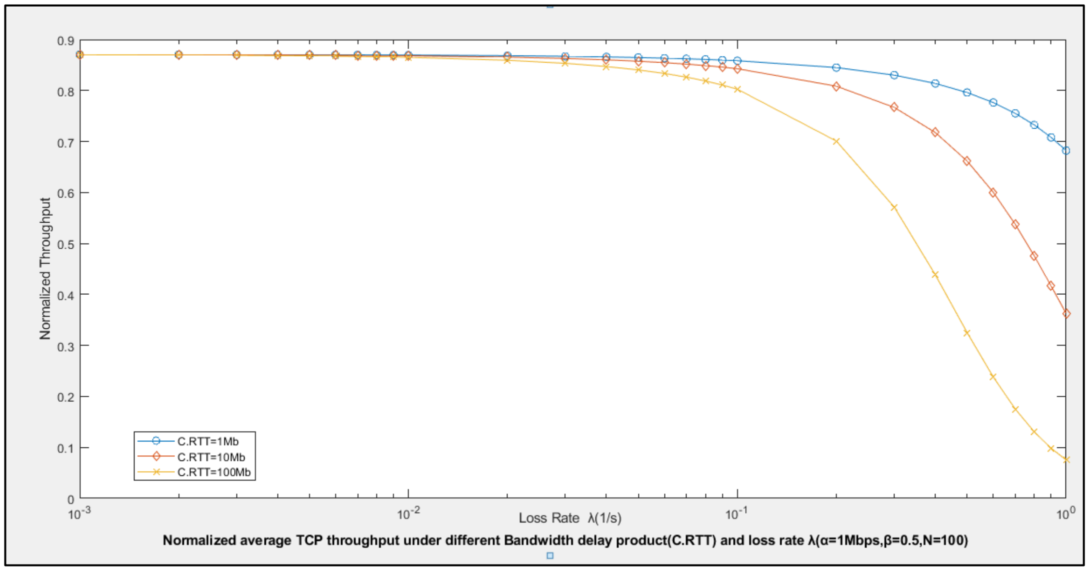

FIG 21: NORMALIZED AVERAGE TCP CUBIC THROUGHPUT UNDER DIFFERENT BDP AND LOSS RATE (A= 1MBPS, B=0.5, N=100)

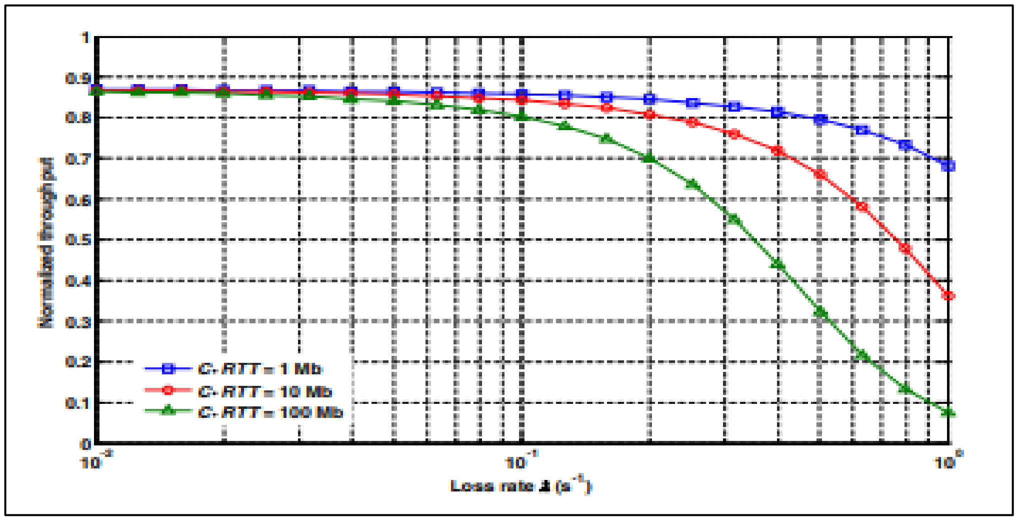

FIG 22: NORMALIZED AVERAGE TCP CUBIC THROUGHPUT UNDER DIFFERENT BDP AND LOSS RATE (A= 1 MBPS, B=0.5, N=100) (FROM [4])

b) **Normalized average throughput under different window growth factor α**

C= 100 Mbps

RTT=100ms

β=0.5

N=100

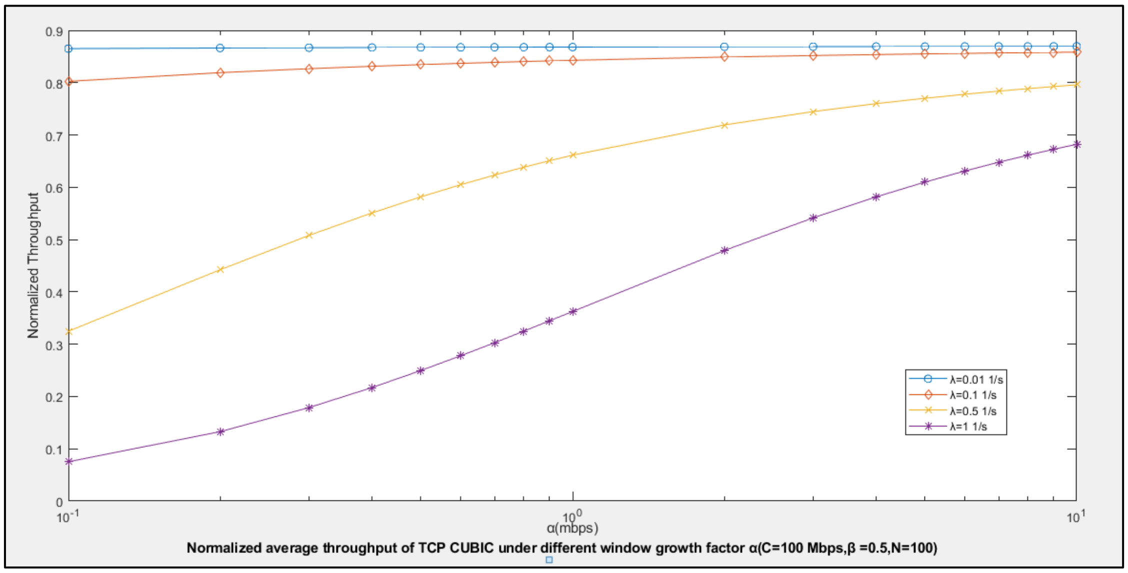

FIG 23: NORMALIZED AVERAGE TCP CUBIC THROUGHPUT UNDER DIFFERENT GROWTH RATE AND LOSS RATE (C= 100 MBPS, Β=0.5, N=100)

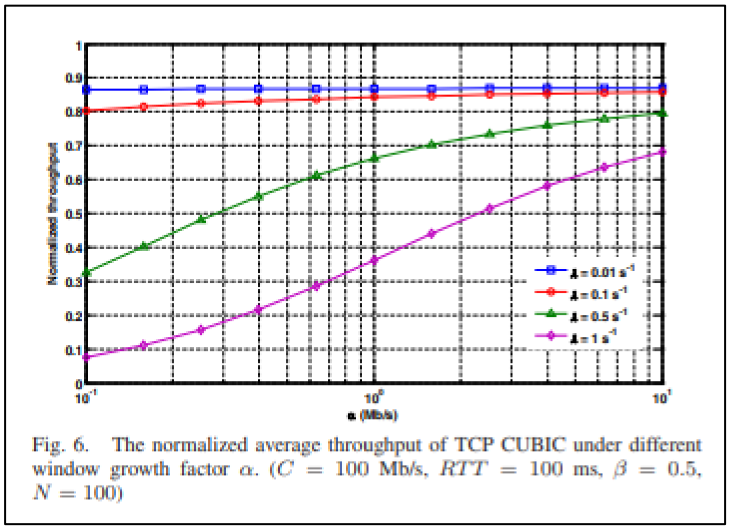

Fig. 6. The normalized average throughput of TCP CUBIC under different window growth factor α. ($C = 100$ Mb/s, $RTT = 100$ ms, $\beta = 0.5$, $N = 100$)

FIG 24: NORMALIZED AVERAGE TCP CUBIC THROUGHPUT UNDER DIFFERENT GROWTH RATE AND LOSS RATE (C= 100 MBPS, Β=0.5, N=100) (TAKEN FROM [4])

c) Normalized average throughput under different β.

α= 1Mbps

C=100 Mbps

RTT=100ms

N=100

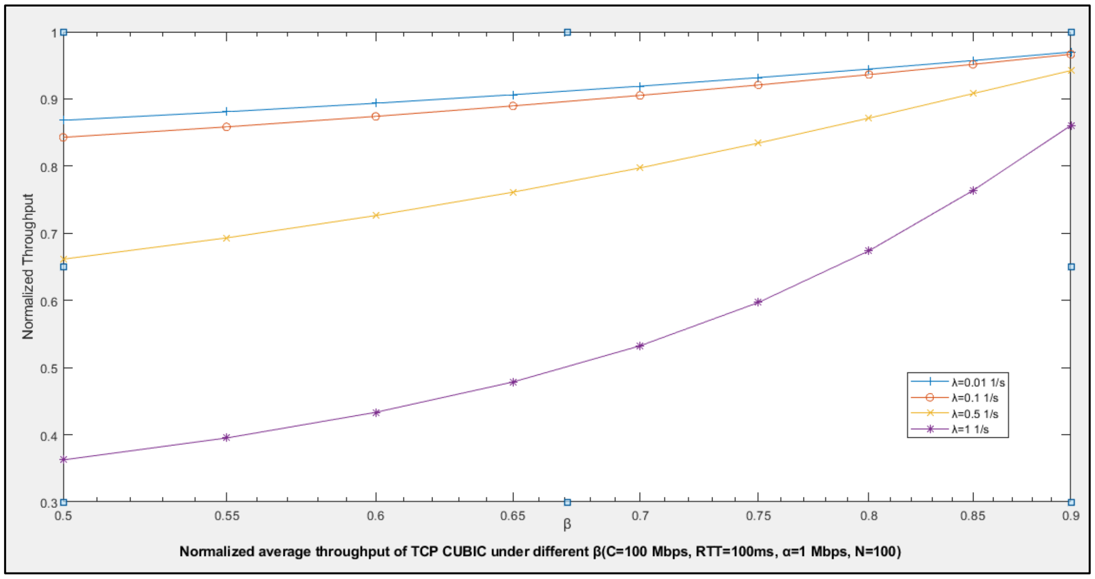

FIG 25: NORMALIZED AVERAGE TCP CUBIC THROUGHPUT UNDER DIFFERENT Β AND LOSS RATE (C= 100 MBPS, Α=1 MBPS, RTT=100MS, N=100)

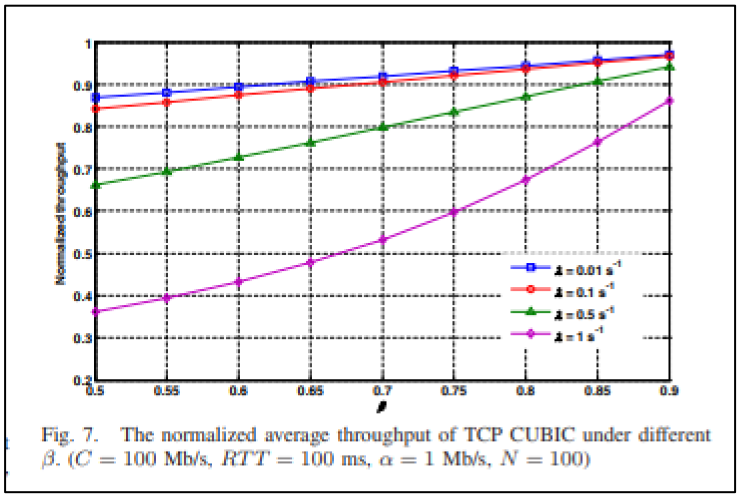

Fig. 7. The normalized average throughput of TCP CUBIC under different β. ($C = 100$ Mb/s, $RTT = 100$ ms, $\alpha = 1$ Mb/s, $N = 100$)

FIG 26: NORMALIZED AVERAGE TCP CUBIC THROUGHPUT UNDER DIFFERENT Β AND LOSS RATE (C= 100 MBPS, Α=1 MBPS, RTT=100MS, N=100) (TAKEN FROM [4])

Discussions on simulated results:

The results of simulation in this paper are the same in paper [4]. If we consider throughput of TCP CUBIC for particular bandwidth delay product C.RTT, we see that the performance goes down with increasing loss rate. In the simulation represented by Fig 21, we have set α= 1Mbps, β=0.5, N=100. The model is simulated for BDP of 1Mb, 10Mb and 100Mb. Average normalized throughput is observed for various loss rates ranging from 10^{-3} to 1 per second. As loss rate increases, average throughput decreases as expected. Moreover, as BDP increases, performance degrades since random losses are more. The normalized throughput never achieves a value of 1 even for very low loss rate. This is because of congestion losses which occurs at window size of W. Also, for loss rates less than 0.1, the variation in performance with bandwidth delay product is very less. Further reducing loss rates, does not affect the performance much.

Fig 23. shows the normalized average throughput under different window growth factor α when, C= 100 Mbps, RTT=100ms, β=0.5 and N=100. Variation in performance is observed for different loss rates. λ is set to 0.01, 0.1, 0.5 and 1 per second. α ranges from 0.1 to 10Mbps. For very high loss rate λ =1 per second, the performance varies from 0.1 to 0.7 as α changes from 0.1to 10 Mbps. On the other hand, for very low loss rate, λ =0.01 per second, variation in performance is not perceptible and stays approximately at 0.85 as α changes from 0.1to 10 Mbps. This change is quite large for high loss rate than for low loss rate. This is because, for low loss rate higher performance is achieved even for very low growth factor whereas, when loss rate is high performance is very low for lesser growth factor and increases gradually as α increases. With low growth factor, the window size grows very slowly. This when combined with high loss rate, does not give sufficient time for the window to grow and hence, bandwidth is not optimally utilized. In contrast to this, for low growth factor and low loss rate, the flow gets sufficient time to grow its window before a loss occurs which results in better throughput. Hence it can be concluded that for networks with high loss rate, increasing growth factor helps in improving the performance whereas for networks with low loss rate, α does not affects the throughput much.

Fig 25. shows normalized average throughput under different β when α= 1Mbps, C=100 Mbps, RTT=100ms and N=100. β is varied from 0.5 to 0.9, λ is set to 0.01, 0.1, 0.5 and 1 per second and variation in throughput is observed. This graph shows trends similar to Fig 23. For very high loss rate λ =1 per second, the performance varies from 0.37 to 0.85 as β changes from 0.5 to 0.9. On the other hand, for very low loss rate, λ =0.01 per second, variation in performance is very small and varies from 0.88 to 0.96 as α changes from 0.1to 10 Mbps. This change is quite large for high loss rate than for low loss rate. This is because, for low loss rate higher performance is achieved even for very low β whereas, when loss rate is high performance is very low for lesser β and increases gradually as β increases. With low β, the window size is reduced by a larger factor upon a loss. This when combined with high loss rate, does not give sufficient time for the window to grow from a smaller value and hence, bandwidth is not optimally utilized. In contrast to this, for low β and low loss rate, the window reduction after loss is less and flow gets sufficient time to grow its window before a loss occurs which results in better throughput. Hence it can be concluded that for networks with high loss rate, increasing β helps in improving the performance whereas for networks with low loss rate, β does not affects the throughput much.

Simulation Model:

A simulation model is developed to validate the analytical model. It is a discrete event simulator. It works on below algorithm:

a. Starting and end of an interval is created using the window size and Number of intervals. This is stored in array “interval”.
b. The midpoint of each interval is created and stored in an array “A”.
c. An array M is declared to store the number of transition from state i to state j.
d. An array P is declared to store the steady state probability.
e. We define another variable k to iterate from k=1 to k=K for the number of times the state transitions has to be done. In other words at every k a loss event is generated. At any time instant k, τ_{loss} is randomly generated using the exponential distribution with loss rate lambda. τ_k is the time between two loss events and is set to be the minimum of τ_{loss}, and D(x_k, W).

f. x_1 is randomly generated between 0 and W. This is mapped to one of the intervals and its midpoint $\tilde{x}_1$.The value of M(i) is updated.
g. Window size at next window reduction x_k (for k=2,3,….K) is calculated using the cubic growth function, $\tilde{x}_k$ and τ_k.
h. Area under the two window reduction “s” is calculated to find the total number of packets transmitted in time τ_k .
i. Again in the next iteration $\tilde{x}_{k+1}$ is calculated.
j. Finally the throughput can be calculated by summing all the area under two window reduction divided by RTT and summing all the τ_k.
k. Steady state probability can be found by dividing M(i) with the sum of all the number of occurrences.

Simulation Model Code:

```
function res=discrete_model(lambda)
C=100000000;
RTT=0.1;
a=1000000;
b=0.5;
N=100;
sum_t=0;
sum_r=0;
A=zeros(1,N);
interval=zeros(1,N+1);
K=10000;
W=C*RTT;
M=zeros(1,N);
P=zeros(5,N);

for j=1:N+1
    interval(j)=(j-1)*W/N;
end
for j=1:N
    A(j)=(j-0.5).*W/N;
end
for k=1:5
x=W*rand(1,1);
for i=1:K
    for j=1:N
        if(interval(j+1)>=x && interval(j)<x)
            xk=A(j);
            M(j)=M(j)+1;
            break;
        end
    end
        Tloss =exprnd(1/lambda);
        D=((W-xk)/a)^(1/3)+((1-b)*xk/a)^(1/3);
        Tk=min(Tloss,D);
        sum_t=sum_t+Tk;

        L=(((1-b)*xk)/a)^(1/3);
        s=(xk*Tk)+(a/4)*((Tk-L)^4-L^4);
        r=s/RTT;
        sum_r=sum_r+r;
        x=a*((Tk-L)^3) + xk;

end

res=sum_r/(sum_t*C);

for i=1:N
    P(k,i)=M(i)/sum(M,'all');
end
end
```

Simulation result of Discrete event model vs Analytical model:

1. **Analytical and simulated average normalized throughput for different loss rate (C=100 Mbps, RTT= 100ms, α= 1Mbps, β=0.5, N=100, K=10000)**
   Discrete event model is plotted by running the dicrete_model for 5 times and taking the average.

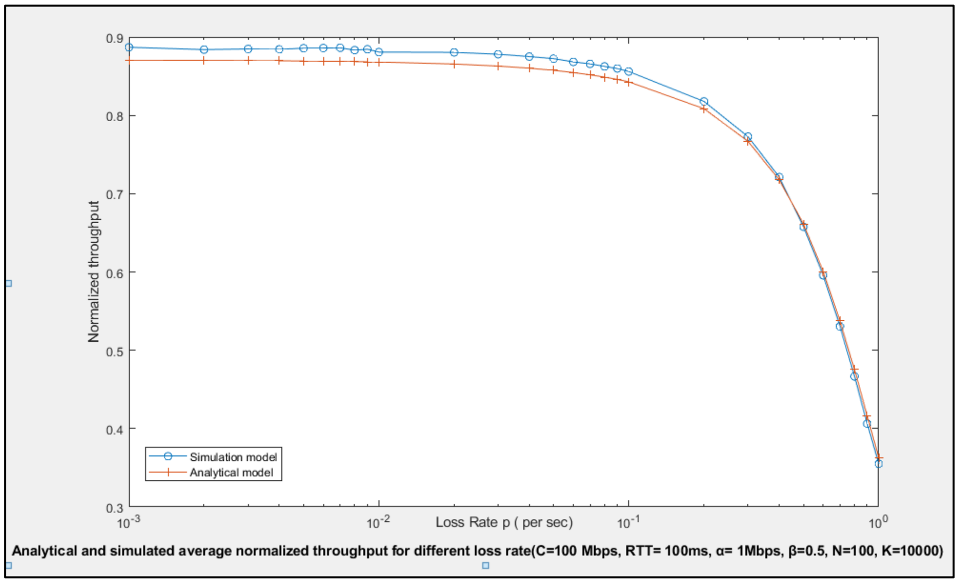

FIG. 27 : NORMALIZED AVERAGE TCP CUBIC THROUGHPUT FOR ANALYTICAL AND SIMULATION MODEL FOR VARIOUS LOSS RATE (C= 100 MBPS, A=1 MBPS, RTT=100MS, B=0.5, N=100, K=10000)

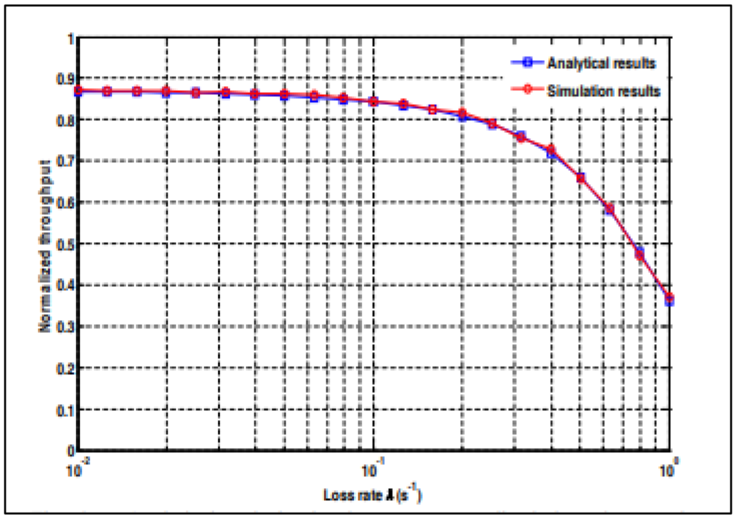

FIG. 28 : NORMALIZED AVERAGE TCP CUBIC THROUGHPUT FOR ANALYTICAL AND SIMULATION MODEL FOR VARIOUS LOSS RATE (C= 100 MBPS, Α=1 MBPS, RTT=100MS, Β=0.5, N=100, K=10000) (TAKEN FROM 4)

It is observed that the normalized throughput obtained from the analytical model is similar to the discrete event model. The simulation in MATLAB is also similar to the result in paper.

2. **Analytical and simulated steady state probabilities for loss rate=1 per sec (C=100 Mbps, RTT= 100ms, α= 1Mbps, β=0.5, N=100, K=10000)**

   Discrete event model steady state probability is plotted by running the dicrete_model for 5 times and taking the average.

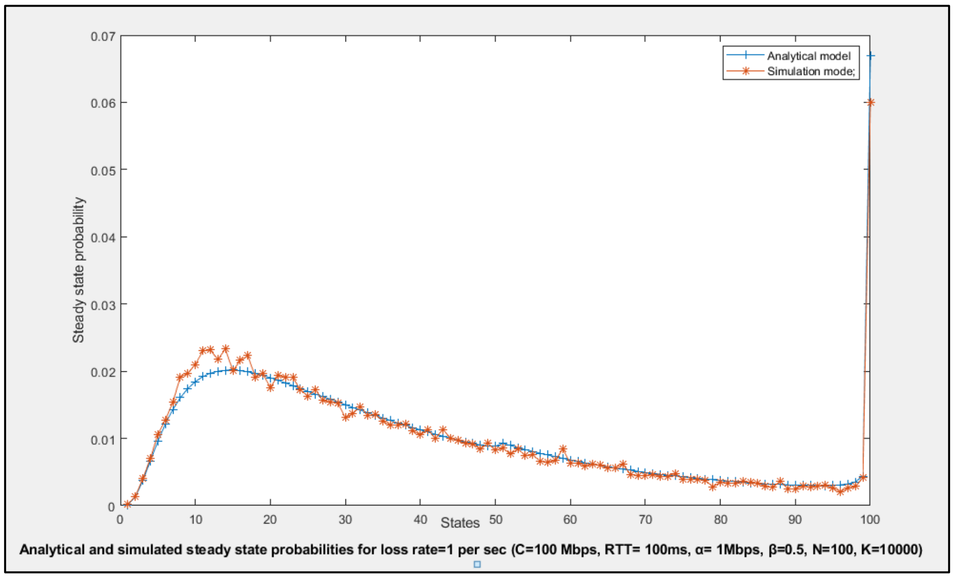

FIG. 29: ANALYTICAL AND SIMULATED STEADY STATE PROBABILITIES FOR LOSS RATE=1 PER SEC (C=100 MBPS, RTT= 100MS, Α= 1MBPS, Β=0.5, N=100, K=10000)

The steady state probability for analytical model also follows similar trend to simulation model.

3. Discrete event model normalized throughput for different number of states (N). (C=100Mbps, RTT= 100 ms, α= 1 Mbps, β=0.5, K=10000)

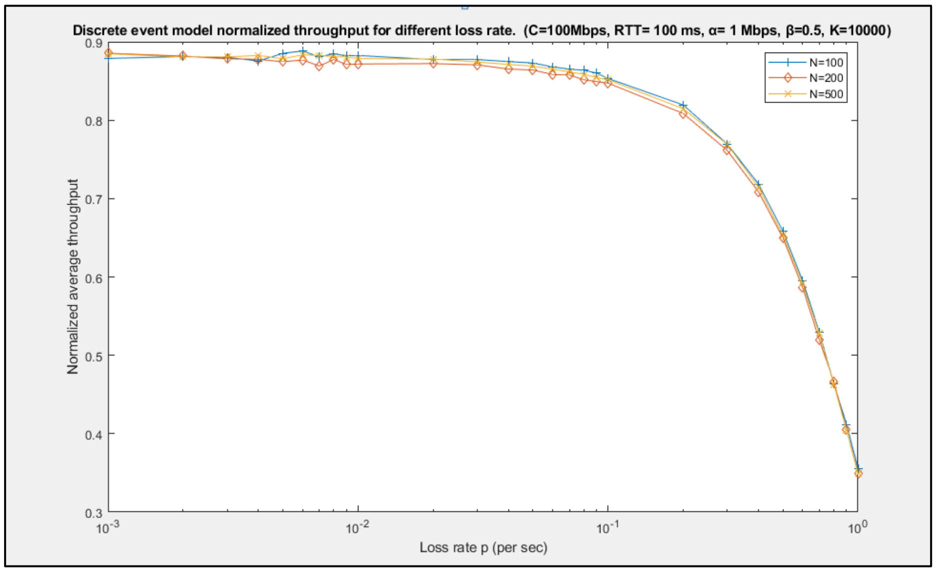

FIG. 30: DISCRETE EVENT MODEL NORMALIZED THROUGHPUT FOR DIFFERENT N. (C=100MBPS, RTT= 100 MS, A= 1 MBPS, B=0.5, K=10000)

The normalized throughput from discrete event model does not vary much on changing the number of intervals.

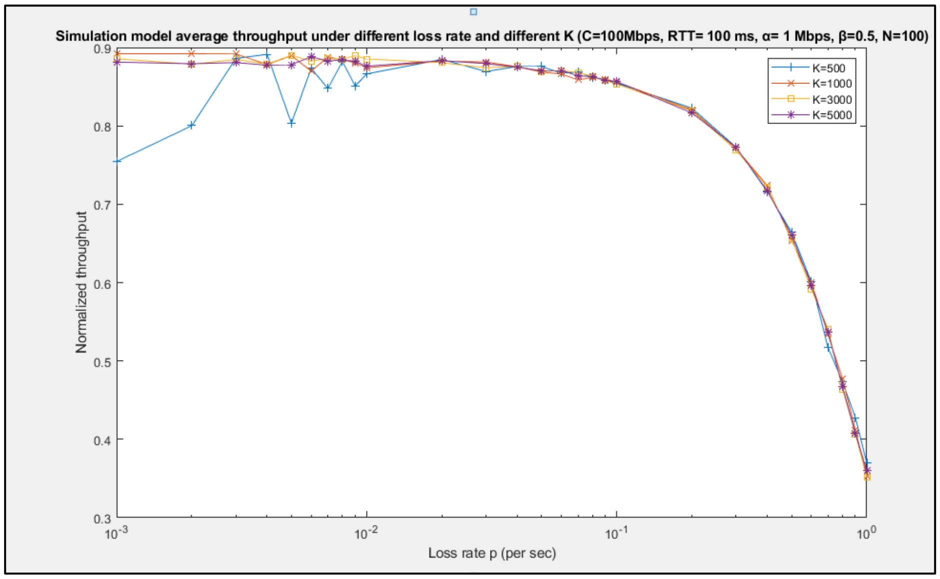

FIG. 31: SIMULATION MODEL AVERAGE THROUGHPUT UNDER DIFFERENT LOSS RATE AND DIFFERENT K (C=100MBPS, RTT= 100 MS, A= 1 MBPS, B=0.5, N=100)

The normalized throughput on varying K also does not change the trend. Only if the value of K is very small we get irregular trend. This is because the number of transitions are quite less when we simulate it for less number and hence error is more.

Conclusion:

A. **Deterministic Loss model: The model in [2] is validated in this paper and there is a minor difference of a factor of 0.707.** The behavior of window growth in this paper is like that found in [1] but different from [2] because TCP friendly mode of operation is considered to simulate the model. High RTT results in better performance, similarly, increasing the growth rate improves the average window size. In this paper, simulation for various multiplicative decrease factor β is also carried out and it was observed that upon increasing β, average window size reduces because in this model, the window size is reduced "by" a factor of β upon loss.

B. **Steady state throughput model for random losses:** The model in [4] is validated in this paper and the outcomes are same. The analytical model was simulated and found to follow the trends shown in [4]. In this model, random losses are also taken into account along with congestion losses due to a bottleneck link in the network. A Markov chain model was established, state transition probabilities were calculated, stationary distribution of states were derived and finally the normalized average throughput was found. The results of this paper are:

   i. As bandwidth delay product increases throughput is reduced due to more random losses for high loss rate networks.

   ii. Upon increasing the growth factor, throughput is improved for high loss rate networks whereas its effect is negligible in low loss rate networks.

   iii. As multiplicative decrease factor is increased, performance improves for high loss rate networks whereas this improvement is not large for low loss rate networks. This is different from deterministic loss model because here the window size reduces "to" β times upon loss.

In order to validate the analytical model developed in paper [4], a discrete event simulator was developed where losses are randomly generated following an exponential distribution. The normalized throughput obtained from this simulator is found to be similar to the analytical model. In addition to this plot, I also simulated the model to find the steady state distribution, the variation in throughput with number of states N and the variation in throughput with the number of transitions K.

References

[1] Sangtae Ha, Injong Rhee, Lisong Xu: CUBIC: A new TCP-Friendly High Speed TCP Variant
[2] Goyzueta R., Yu Chen: A Deterministic Loss model based Analysis of CUBIC
[3] Sudheer Poojary, Vinod Sharma: An asymptotic approximation for TCP CUBIC
[4] Wei Bao, Vincent W.S. Wong, and Victor C.M. Leung: A Model for Steady State Throughput of TCP CUBIC
[5] Sally Floyd, Mark Handley, Jitendra Padhye : A Comparison of Equation-Based and AIMD Congestion Control
[6] Matthew Mathis, Jerey Semke, Jamshid Mahdavi : The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm
[7] Lisong Xu, Khaled Harfoush, and Injong Rhee: Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks
[8] Jitendra Padhye Victor Firoiu Don Towsley Jim Kurose : Modeling TCP Throughput: A Simple Model and its Empirical Validation
[9] Sudheer Poojary and Vinod Sharma : Analytical Model for Congestion Control and Throughput with TCP CUBIC connections

[10] Mudassar Ahmad, Awais Ahmad, Sohail Jabbar, M. Asif, Muhammad Asif Habib, Syed Hassan Ahmed, S. C. Shah: TCP CUBIC: A Transport Protocol for Improving the Performance of TCP in Long Distance High Bandwidth Cyber-Physical Systems

Figures 1 and 2 were directly taken from [1] and [10], figures 3, 4,17,18 were modified for this paper and figures 16,19 and 20 are generated for this paper. The rest are either the simulation figures from MATLAB or the papers [1] [2] and [4].