# ANALYSIS OF TCP WITH SEVERAL BOTTLENECK NODES

Chadi Barakat and Eitan Altman
INRIA
2004, route des Lucioles, 06902 Sophia Antipolis, France

## Abstract

Many works have studied the performance of TCP by modeling the network as a single bottleneck node. In this paper, we present a more general model taking into account all the nodes on the path not only the main bottleneck. We show that, in addition to the main bottleneck, the other nodes can seriously affect the performance of TCP. They may cause an improvement in the performance by decreasing the burstiness of TCP traffic arriving at the main bottleneck. But, if the buffers in these nodes are not well dimensioned, the congestion may be shifted to them which deteriorates the performance even though they are faster than the main bottleneck. We conclude our analysis by guidelines for the dimensioning of network buffers so as to improve the performance of TCP.

## Introduction

Because of its crucial role in the stability of the Internet, the performance of TCP has been extensively studied [1, 4, 6, 7]. These works often model the network as a single bottleneck node, the one having the slowest outgoing rate on the path between the source and the destination (we call it *the main bottleneck* in the sequel). This model is correct if the buffering capacity and the available bandwidth in the other nodes are very large compared to those in the main bottleneck. However, due to the fluctuations in real networks, these quantities can be very close to each other which may result in a different performance. It is known that TCP transmits bursts of packets especially during the Slow Start (SS) phase [1, 4, 7]. These bursts may cause a queue building in many nodes not only in the main bottleneck. This may avoid a buffer overflow predicted by the single bottleneck model which will result in an improvement in the performance. But also, this may cause an unpredicted overflow if the buffers in these nodes are not enough large. The single node model overestimates the real performance in this later case.

In this paper, we study the performance of TCP as a function of the parameters of all the nodes crossed by the connection. As node parameters, we consider *the available bandwidth* and *the buffering capacity*. Drop Tail buffers are considered because they are widely used in the Internet. We present a general model of the network consisting of many nodes, then we simplify it to a model with two nodes without changing the performance of TCP. We use the Network Simulator `ns` [8] to validate our analytical results. Among our results, we show that the other nodes affect the Slow Start (SS) phase much more than the Congestion Avoidance (CA) one. To eliminate the effect of the other nodes, their buffers must scale linearly with that of the main bottleneck. Although the buffers required in these nodes are not as important as in the main bottleneck, they are necessary to absorb TCP bursts when the output rates of these nodes get close to the main bottleneck rate.

## The multiple node model

Let $\mu$ (packets/s) be the main bottleneck rate on the path of a TCP connection. This connection transfers data between a source $Sr$ and a destination $D$ and crosses many nodes. We denote by $N_R$ a node of available bandwidth $R$ (or of rate $R$). In order to consider the burstiest case, we suppose that ACKs are not delayed at $D$. The modification of our model to the Delay ACKs case is straightforward. We suppose also that the return path is not congested. $\mu$ is then the rate at which ACKs return to $Sr$. Because all the nodes between $N_\mu$ and $D$ receive packets at a rate slower than their available bandwidth, their parameters don't affect the performance of TCP. The following analysis focuses on the impact on TCP of nodes between $Sr$ and $N_\mu$.

According to TCP congestion control algorithms [5, 9], if ACKs are not lost nor delayed, an ACK triggers maximum the transmission of a burst of two packets. This happens upon every ACK reception during SS and when the congestion window $W$ increases by one segment during CA. If the nodes between $Sr$ and $N_\mu$ have an available bandwidth $> 2\mu$, the two-packets bursts will cross these upstream nodes as if they don't exist. In this case, we can ignore these nodes and suppose that $N_\mu$ is fed directly by $Sr$. This is what the single node

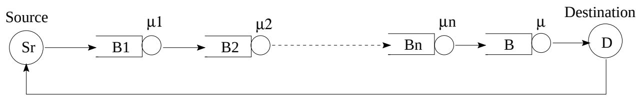

Figure 1: The multiple node network model

model assumes. Now, if one of the upstream nodes has a rate $< 2\mu$, the bursts will be slowed. A queue will build up in this slow node which reduces the queue building rate in the main bottleneck $N_\mu$. We say here that TCP bursts are partially absorbed by this upstream node. This partial absorption may overflow the buffer in this intermediate node and it may avoid a buffer overflow in another node. Thus, to study the performance of TCP, the network model must take account of the nodes preceding $N_\mu$ and having a rate $< 2\mu$. In Figure 1, we show our model where, in addition to $N_\mu$, $n$ nodes of buffers $B_i$ and of bandwidth $\mu_i$ are considered. The $\mu_i$ satisfy

$$\mu < \mu_n < \mu_{n-1} < \cdots < \mu_1 < 2\mu.$$

This assumption is without loss of generality since if a node is faster than its predecessor, then no queueing will occur there, so the related node can be ignored while analyzing the performance of the connection. The restriction $\mu_1 < 2\mu$ is without loss of generality as well, since a node of rate $\mu_1 > 2\mu$ would mean that a queue never builds up in $N_{\mu_1}$, since the input rate to that node, even at bursty periods, is upper-bounded by $2\mu$.

**The behavior of TCP during CA**

We suppose that, during CA, $W$ increases by one segment every Round Trip Time (RTT). The receiver window is set to a high value so that $W$ is only limited by network parameters. After the transmission of a burst of two packets as a result of the window increase from $W-1$ to $W$, the source transmits $W-1$ packets at a rate $\mu$ followed by a new burst of two packets when the last ACK of the window $W$ is received. Here, the window increases to $W+1$. Let $T$ be the constant component of RTT (propagation delay plus service time). If $W$ is smaller than the Bandwidth-Delay Product (BDP) $\mu T$, the two bursts are separated by a time $T$. No queue builds up and the window continues growing up linearly by one segment every $T$.

When the window exceeds $\mu T$, ACKs start to arrive continuously at the source. Thus, there will not be enough time for the nodes of rate slower than $2\mu$ to serve the two packets of a burst. A queue starts to build up in the network. According to the single node model, the queue is seen only in the node of rate $\mu$ and CA ends when $B$ overflows. This happens at a window $W = W_{max} = B + \mu T$. Such assumption is true if the upstream nodes are faster than $2\mu$. However, if one upstream node has a rate slower than $2\mu$, a queue builds up in this node in addition to node $\mu$ which results in a different value of $W_{max}$, thus in an different throughput of the connection.

Consider the two bursts sent when the window increases from $W-1$ to $W$ and from $W$ to $W+1$. In the case $W > \mu T$, the time between these two bursts is equal to $W/\mu$ and the number of packets is equal to $W+1$. We can say that at a window $W > \mu T$, the source transmits packets at an average rate $R_{CA} = (W+1)\mu/W$. This rate is always greater than $\mu$ and then the number of packets waiting in the network always increases during CA. These waiting packets are distributed between $N_\mu$ and the upstream nodes having a rate $< R_{CA}$. Thus, the queue in $N_\mu$ builds up at a rate slower than one packet per RTT as long as there is a $\mu_i$ $(1 \leq i \leq n)$ satisfying $\mu_i < R_{CA}$. Given that $R_{CA}$ is a decreasing function of $W$, the effect of $N_{\mu_i}$ on the performance decreases when $W$ moves away from $\mu T$. Once we reach a window $W$ that results in a $R_{CA}$ smaller than all the $\mu_i$, all the waiting packets move to $B$ which puts us in the case of the single node model.

Because $R_{CA}$ is very close to $\mu$, the most interesting case is when there exists a $\mu_i = \mu$. Let $N_{\mu_i}$ be the closest node to the source having a rate $\mu_i = \mu$ and suppose that all the nodes between $Sr$ and $N_{\mu_i}$ have a rate $> R_{CA}$. Only node $N_{\mu_i}$ in the network is fed at a rate faster than its service rate. TCP bursts during CA are then completely absorbed by this node and a queue doesn't build up in the other nodes. Thus, if many nodes in the network have the same rate $\mu$, the queue builds up in the buffer of the closest node to the source. The capacity of this buffer determines alone the value of $W_{max}$. In the next sections, we suppose that all the $\mu_i$ are larger than $R_{CA}$. $B$ represents the buffer size of the closest node to the source having a rate equal to $\mu$.

To illustrate our conclusions, we simulate a TCP connection across two nodes of rate $\mu_1$ and $\mu$. We take $T$ = 560ms (case of a GEO satellite), $\mu$ = 1.5Mbps (T1 link), packet size=512Bytes, $B_1$ = 100packets and $B$ = 50packets. First we set $\mu_1$ to $\mu$, then we increase it slightly in order to get $R_{CA} < \mu_1$. We plot in Figure 2, $W$ as a function of time for two values of $\mu_1$, 1.5Mbps and 1.6Mbps. Although we increased $\mu_1$, the average window size decreases resulting in a decrease in the average throughput from 1.371Mbps to 1.247Mbps. Indeed, the increase in $\mu_1$ has moved the queue from $B_1$ to $B$. Given that $B < B_1$, $W_{max}$ decreases from $(B1 + \mu T)$ to $(B + \mu T)$ which explains this deterioration in the throughput.

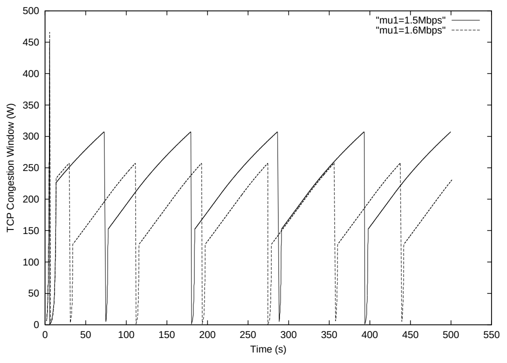

Figure 2: TCP performance as a function of $\mu_1$

## The behavior of TCP during SS

During SS, the source sends long bursts of packets at twice $\mu$ due to a burst of two packets in response to every new ACK [1, 7]. A queue may then build up in any node not only in $N_\mu$. If the buffers in these nodes are not well dimensioned to absorb the bursts sent during SS, they might overflow early before filling the available capacity in the network. Given that TCP considers all losses as a signal of congestion and reacts by reducing its window, an early buffer overflow results in an underestimation of the available resources in the network. This problem occurs when the SS Threshold ($W_{th}$) is very large so that the buffers overflow before getting in CA. It is a typical problem of networks with small buffers compared to their BDP [1, 3, 7]. An example of such networks are satellite networks where the BDP is large and where there are many limitations on the size of buffers on satellite board.

Buffers in the networks must be large enough to absorb the bursts sent during SS. If SS Threshold $W_{th}$ is a good estimation of the network capacity, SS must end without suffering from losses. In contrast, if $W_{th}$ is set to a high default value as at the beginning of a TCP connection, losses during SS are unavoidable. They serve to give $W_{th}$ a more accurate value. However, these losses must not appear early otherwise the available bandwidth will be underestimated. In this paper, we concentrate on the first case where $W_{th}$ is a correct estimation of the network capacity and where the aim of SS is to fill quickly this capacity. The other case will be the subject of future work.

We suppose that $W_{th}$ is set to half $W_{max} = \mu T + B$. This is the value TCP gives to $W_{th}$ after a congestion during the connection lifetime. The same analysis can be applied to other values of $W_{th}$. A condition to not encounter the problem of losses during SS has been calculated in [1, 7]. This condition accounts only for the main bottleneck. In this section, we recalculate this condition with our general model. As in these works, we consider a Tahoe version of TCP [5, 4] where SS is frequently called.

We divide a SS phase into mini-cycles of duration $T$ [7]. $W$ doubles every $T$. During mini-cycle $n$, $Sr$ sends a burst of $2^n$ packets at rate $2\mu$. Mini-cycle $n+1$ starts when the ACK for the first packet of this burst reaches $Sr$. These bursts propagate from node to node and create queues in the different buffers of Figure 1. These queues build up at a rate $(2\mu - \mu_1)$ in $B_1$, at a rate $(\mu_{i-1} - \mu_i)$ in $B_i$ $(i = 2 \dots n)$, and at a rate $(\mu_n - \mu)$ in $B$.

Given these rates, we can calculate the number of packets sent in a burst and required to overflow each buffer

$$\begin{array}{ll} S_1 = 2\mu B_1/(2\mu - \mu_1) & \text{for } B_1, \\ S_i = \mu_{i-1} B_i/(\mu_{i-1} - \mu_i) & \text{for } B_i \ (i = 2 \dots n), \\ S = \mu_n B/(\mu_n - \mu) & \text{for } B. \end{array}$$

We get a loss if in a given mini-cycle, $Sr$ sends a burst of packets at rate $2\mu$ larger than at least one of the $S_i$ $(i = 1 \dots n)$ and $S$. Let $S_B = \min_{i=1\dots n}(S, S_i)$. The loss occurs in mini-cycle $n_B$ given by

$$2^{n_B - 1} < S_B \leq 2^{n_B},$$

and the window size $W_B$ at which this loss happens is

$$W_B = 2^{n_B - 1} + S_B/2 \Longrightarrow 2W_B = 2^{n_B} + S_B.$$

The condition to avoid a buffer overflow during SS is for $W$ to reach $W_{th}$ before $W_B$. Thus, $W_{th} < W_B$. In contrast to that found in [1, 7], we see well that this condition involves the parameters of all the nodes not only those of the main bottleneck. We notice that even if we supply $N_\mu$ with a large buffer so as to avoid the overflow according to the condition found in [1, 7] ($\beta = B/\mu T > 1/3$), the other network parameters can result in a $W_B$ smaller than $W_{th}$, then in an occurrence of losses and a deterioration of the performance. Also, because its input rate is bounded by $\mu_n$ which is smaller than $2\mu$, the queue in buffer $B$ builds up slowly resulting in a higher value of $S$. Depending on the values of $S_i$, this may avoid an overflow predicted by the condition $\beta < 1/3$ which results in a better performance.

Now, if the buffers are large enough so that $W_{th} < W_B$, the bursts sent during SS will not cause an overflow and the source will get in CA with one SS phase. The window then increases slowly from $W_{max}/2$ to $W_{max}$ where a normal loss occurs. We get here the same behavior as with the single node model, of course if $\beta > 1/3$.

## A simplified model with two bottleneck nodes

Any model must consider node $N_\mu$ because it determines TCP behavior during CA, particularly the value of $W_{max}$. However, to predict the behavior during SS, the upstream nodes which have a rate between $\mu$ and $2\mu$ must also be considered. To simplify the analysis without changing the results, the upstream nodes can be replaced by an *equivalent node* which gives a simple model consisting of two bottleneck nodes (Figure 3).

The idea is to replace nodes $N_1$ to $N_n$ by that having the smallest burst size $S_i$, since that node is the candidate to see the first overflow during SS. The buffer size $B'$ and the service rate $\mu'$ of the equivalent node are chosen as follows as a function of $B_i$ and $\mu_i$ $(i = 1 \ldots n)$.

We take $\mu'$ equal to $\mu_n$ and we suppose that the equivalent node is directly fed by the source. The reason for taking $\mu_n$ as a rate is that we don't wish to change the behavior of the main bottleneck whose input rate in the original problem was bounded by $\mu_n$. For $B'$, we choose it in a way that the equivalent node, which has an input rate bounded by $2\mu$ and an output rate $\mu_n$, requires a burst of size $\min_{i=1\ldots n}(S_i)$ to be filled. Thus,

$$S' = 2\mu B'/(2\mu - \mu') = \min_{i=1\ldots n}(S_i).$$

We use `ns` to prove the correctness of the equivalent node approach. We take for $\mu$, $T$ and packet size the same values as in the previous section and we set $B$ to 20packets. The nodes are placed on the path between $Sr$ and $D$ in a way that $T$ is always equal to 560ms. We vary $n$, the number of nodes preceding $N_\mu$, between 1 and 10. For each $n$, we distribute the $\mu_i$ uniformly on the segment $[\mu, 2\mu]$. Thus, $\mu_i = 2\mu - i\mu/(n+1)$ for $i = 1 \ldots n$. A simple calculation shows that $S = B \times (n+2)$ and $S_i = B_i \times [2(n+1) - (i-1)]$, $i = 1 \ldots n$.

If we take all the $B_i$ equal to $B$, then $S_B$ will be always equal to $S$ and the overflow will always occur in node $N_\mu$. To change a little the network behavior while varying $n$, we choose the $B_i$ in a way that the overflow moves to one of the upstream nodes for some values of $n$. By taking the $B_i$ equal to 17packets, we make $S_B$ equal to $S_n$ for $n \geq 4$. For this value of $B_i$ we find,

$$\begin{aligned} \mu' &= \mu_n = \mu + \mu/(n+1), \\ S' &= \min_{i=1\ldots n}(S_i) = S_n \Longrightarrow B' = 8.5n(n+3)/(n+1). \end{aligned}$$

In Figure 4 we plot, for the different values of $n$, TCP throughput with the multiple node model and the simplified model. We plot also the throughput obtained when only the main bottleneck is considered. We see well that the results are very close for our two models.

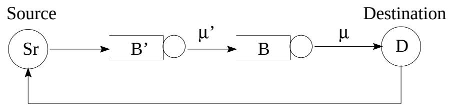

Figure 3: The simplified two-nodes model

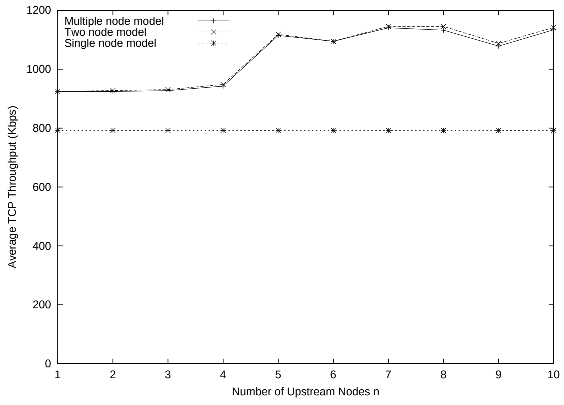

Figure 4: Simulation: Comparison between models

We see also an increase in the throughput when the network contains five nodes and more. Indeed, a large $n$ increases $S_n$ and $S$, then $W_B$, which causes a disappearance of the losses during SS and an improvement in the performance. SS bursts are absorbed by more and more nodes which eliminates the possibility of an overflow observed at small $n$.

The figure shows also that considering only the main bottleneck leads to an underestimation of TCP throughput. A single node model considers only node $N_\mu$ and supposes that it is fed directly by $Sr$. In our case, this reduces $S$ from $20(n+2)$ to $2B = 40$. This new value of $S$ is always smaller than the real value of $S_B$ when the nodes 1 to $n$ are considered. A smaller $S$ means a smaller window at the end of SS and at the beginning of CA. This explains the reduction in the predicted throughput. In fact, the single node model doesn't consider the likelihood of a partial absorption of the bursts by the upstream nodes. It represents the worst case where the bursts are only absorbed by the main bottleneck.

## TCP performance with the simplified model

In Figure 5, we plot $S$ and $S'$ as a function of $\mu'$. $\mu'$ takes its values between $\mu$ and $2\mu$. It is clear that the minimum value of $S$ (resp. $S'$) is $2B$ (resp. $2B'$) and it corresponds to $\mu' = 2\mu$ (resp. $\mu' = \mu$). Thus, to solve the problem of losses during SS for all the values of $\mu'$, we must chose $B$ and $B'$ in a way that

$$\mu T + B < 2^{n_B} + 2B, \qquad 2^{n_B - 1} < 2B \leq 2^{n_B},$$

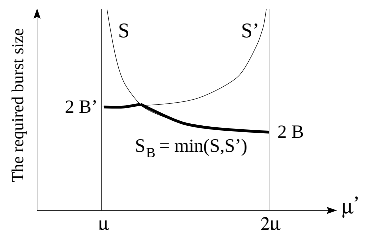

Figure 5: The variation of $S$ and $S'$ as function of $\mu'$

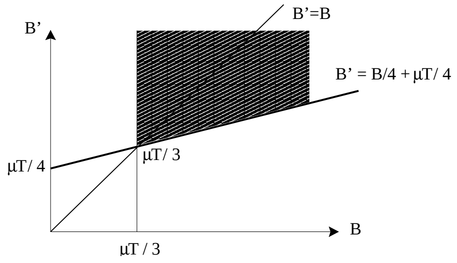

Figure 6: The required $B'$ as a function of $B$

$$\mu T + B < 2^{n'_B} + 2B', \qquad 2^{n'_B - 1} < 2B' \leq 2^{n'_B}.$$

If we approximate $2^{n_B}$ by $2B$ and $2^{n'_B}$ by $2B'$ as in [7], we get the following condition on $B$ and $B'$ to avoid always the losses during SS,

$$B > \mu T/3 \qquad \text{and} \qquad B' > (\mu T + B)/4.$$

The shaded region in Figure 6 represents the appropriate values of $B$ and $B'$ for a given $\mu T$. It is clear that any increase in $B$ beyond $\mu T/3$ doesn't require an increase in the buffer size in node $\mu'$ by the same amount.

Now, if the problem appears at $\mu' = 2\mu$ (resp. at $\mu' = \mu$), it can be avoided by decreasing (resp. increasing) $\mu'$ if $B'$ (resp. $B$) is enough large. A $\mu'$ between $\mu$ and $2\mu$ spreads the queue over the two buffers reducing the likelihood of an overflow during SS.

In Figure 7, we plot simulation results showing how the throughput varies as a function of $\mu'$. We take the network parameters so that an overflow appears in $B$ when $\mu' = 2\mu$. For a large $B'$ (80packets), the throughput jumps up at a certain $\mu' < 2\mu$ due to the disappearance of the overflow during SS but it doesn't decrease if we further decrease $\mu'$. This is because $B'$ is large enough to absorb alone the bursty traffic when $\mu' = \mu$. For a medium $B'$ (45packets), the throughput increases as we decrease $\mu'$ only in some intermediate range. If we further decrease $\mu'$, it then decreases too due to the reappearance of losses during SS. However, for a small $B'$ (20packets), the problem is not only not solved in the middle but also worsened as $\mu'$ decreases.

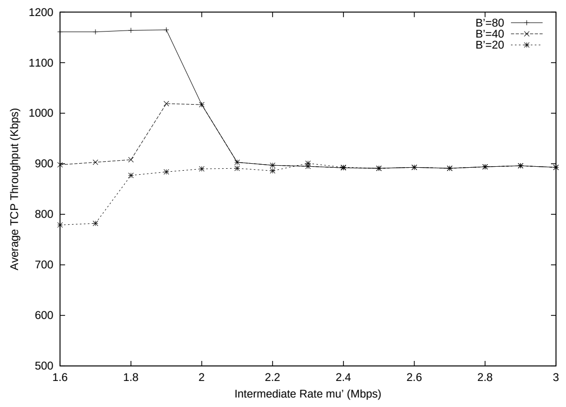

Figure 7: Simulation: TCP Throughput vs. $\mu'$

## Conclusion

In this paper, we developed a model that accounts for all the nodes on the path of a TCP connection. We showed that the nodes upstream the main bottleneck may affect the performance of TCP especially of its Slow Start phase. They may improve the performance by reducing the burstiness of TCP traffic. They may also deteriorate the performance if they are not well dimensioned. At the end, we showed that the required buffer size in these nodes is not as important as in the main bottleneck.

## References

[1] E. Altman, J. Bolot, P. Nain, D. Elouadghiri, M. Erramdani, P. Brown, and D. Collange, "Performance Modeling of TCP/IP in a Wide-Area Network", *34th IEEE Conference on Decision and Control*, Dec. 1995.

[2] C. Barakat and E. Altman, "Analysis of TCP in Networks with Small Buffering Capacity and Large Bandwidth-Delay Product", *INRIA Research Report N=3574*, Dec. 1998.

[3] N. Chaher, C. Barakat, W. Dabbous, and E. Altman, "Improving TCP/IP over Geostationary Satellite Links", *IEEE Globecom*, Dec. 1999.

[4] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communication Review*, Jul. 1996.

[5] V. Jacobson, "Congestion avoidance and control", *ACM Sigcomm*, Aug. 1988.

[6] A. Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link", *IEEE/ACM Transactions on Networking*, Aug. 1998.

[7] T.V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss", *IEEE/ACM Transactions on Networking*, Jun. 1997.

[8] The LBNL Network Simulator, *ns*, http://www-nrg.ee.lbl.gov/ns.

[9] W. Stevens, "TCP Slow-Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001*, Jan. 1997.