# Steady State Analysis of BBR

Nitin Shyamkumar (nhs9171)

## ABSTRACT

Recently, Cardwell et. al. proposed the BBR algorithm to improve on TCP variants for congestion control. In contrast to packet loss based congestion control algorithms, BBR attempts to operate at the optimal operating point of a bandwidth constrained network where the delay is minimized and throughput is maximized. However, BBR is in its early stage and has numerous shortcomings. We examine poor performance on throughput in long latency networks in detail. We survey three different techniques from hybrid systems literature, including identifying Piecewise Affine systems, Max Min Plus Scaling systems, and the Min Plus service curve framework from the network calculus literature. We develop an analytical explanation of the performance degradation for longer latencies and design a steady state model for BBR in the network calculus framework. Our analytical explanation also suggests two possible fixes to improve BBR throughput in long latency networks.

## 1. INTRODUCTION

In the last few decades as memory has become cheaper, network buffers have increased in size. The increased buffer size means that packet loss is not an effective signal of congestion. Yet most traditional TCP congestion control algorithms use packet loss as a signal nonetheless. When large buffers absorb excess packets, packet loss can occur well after the network becomes congested and packet loss-based congestion control algorithms can have significant performance degradation.

In order to address this shortcoming of packet loss based congestion control algorithms, Cardwell et. al. propose a congestion algorithm based on accurately measuring the Bottleneck Bandwidth and Round-trip propagation time, or BBR [4]. BBR is a simple state machine that alternates between measuring the bottleneck bandwidth and the round trip propagation time. Packets are sent at a rate informed by these two quantities. In contrast to TCP Cubic, BBR has higher throughput even at large packet loss rates, and lower latencies [4]. While BBR achieves its goal of keeping buffers minimum, TCP Cubic fills buffers resulting in a delay that increases linearly with buffer size [4].

However, BBR's initially promising results invited scrutiny. Recent studies demonstrated critical shortcomings in BBR. [10] showed that multiple BBR flows can each overestimate their fair share of the bottleneck bandwidth, resulting in overload, which can lead to RTT unfairness, queue buildup, increased latency, and massive packet loss. Other works have reported similar phenomena through detailed empirical analysis [2, 11, 17]. [2] specifically studies BBR's performance in cellular networks where link delays can be much longer than in standard networks.

These papers largely fall into the category of empirical analysis. While [10] pays more attention to the BBR state and measurement dynamics, other works treat BBR as a black box and study relevant metrics like external throughput, delay, and queue sizes. To the best of the author's knowledge, no work has attempted to construct a model of the BBR state dynamics, despite Cardwell et. al's contention that "BBR is a simple instance of a Max-plus control system" [4].

In this paper, we take a first step to constructing an analytical model. We investigate three different approaches: a data driven Piecewise Affine (PWA) model (Section 3), an analytically constructed Max Min Plus Scaling (MMPS) model (Section 3), and another analytically constructed model using the Network Calculus framework (Section 4). We use the Network Calculus framework to construct a closed loop model of BBR steady state dynamics (Section 5). Following model construction, we use the model to characterize BBR performance and tradeoffs (Section 6). Finally, we compare the model predictions to empirical observations (Section 6).

## 2. PROBLEM STATEMENT

One of BBR's shortcomings is significant performance degradation on networks with longer latencies. We present a simple experiment to show this. Consider a three node network connected as

$$\text{Node } 0 \Longleftrightarrow \text{Node } 1 \Longleftrightarrow \text{Node } 2$$

where a source server (Node 0) is connected to a destination (Node 2) via a middle router (Node 1). The 0-1 link is fixed to be a 15Mbps bandwidth link with 10ms of delay. The 1-2 link has varied bandwidth $B$ and delay $D$, but $B$ is always less than 15Mbps so that it is guaranteed to be the bottleneck. $B$ is varied from 0.1 to 1 Mbps while $D$ is varied from 1ms to 1.5 seconds. The results are shown in Figure 2, which shows

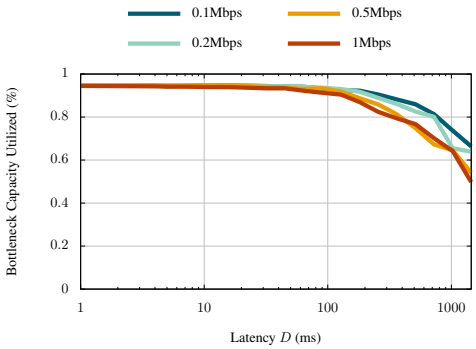


Figure 1: Latency ($D$) vs Throughput (Mbps)

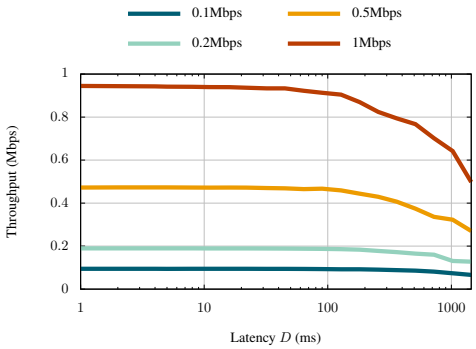


Figure 2: Latency ($D$) vs Bottleneck Utilization ($\%$)

throughput degradation to 60% of the bottleneck utilization when latencies are on the order of 1 second.

An intuitive explanation is easy to find after a brief examination of the BBR dynamics. In the steady state, BBR alternates between spending 10 seconds in the ProbeBW state sending at the estimated full capacity, and spending 0.2 + $\text{RTT}_{\text{true}}$ seconds in the ProbeRT state, where the window is restricted to just 4 packets. As latencies increase, $\text{RTT}_{\text{true}}$ increases, and so BBR spends a much higher proportion of its running time with a window of just 4 packets.

While this specific problem may be easily resolved, the lack of a precise analytical model makes answering other questions more difficult. How are queue sizes affected? Do longer latencies amplify measurement errors? To answer these questions as well as our originally formulated problem, we seek a more definitive model.

# 3. HYBRID SYSTEMS APPROACHES

## 3.1 Overview

Hybrid systems are a class of systems that include continuous and discrete dynamics, or intuitively, systems that can both *flow* and *jump*. Typically, the continuous dynamics are described using differential equations while the discrete dynamics are modeled by a finite state machine. Researchers have studied a huge variety of hybrid systems over the years, though many are equivalent to one another under some minimal assumptions [9, 19].

The benefit of using hybrid systems over more general nonlinear models (such as recurrent neural networks) is that we can apply traditional control theoretic concepts for steady state analysis. Over the past few decades, traditional control theoretic methods and concepts like Lyapunov stability, reachability, and Poincaré maps have been mapped to hybrid systems [19]. Hybrid systems are especially compelling for modeling BBR since BBR includes both a discrete state machine and approximately continuous dynamics for packet throughput.

We explore two hybrid systems. The Max Min Plus Scaling (MMPS) framework is a superset of the Max Plus framework. Thus, any system realizable in the more restrictive Max-Plus framework (consisting of just the maximum and addition operations) is also a valid MMPS system. Cardwell et. al declare that BBR is a "simple instance of a Max-plus control system", so this framework is a reasonable first choice. The second approach is to learn a data driven Piecewise Affine system model. Such an approach would eliminate the costly exercise of manually constructing a model while still yielding a structured model for control theoretic analysis. Unfortunately, as we will show in the subsequent sections, both methods face significant shortcomings.

## 3.2 Max Min Plus Scaling (MMPS) Systems

A Max Min Plus Scaling (MMPS) system is a discrete time dynamical system

$$\mathbf{x}_{t-1} = g(\mathbf{x}_t)$$
$$f := x_i | \alpha | \max(f_k, f_l) | \min(f_k, f_l) | f_k + f_l \mid \beta f_k$$

where $f$ is a recursively defined MMPS expression using the operations of scalar multiplication, maximum, minimum, and addition. $g$ is a vector valued function where $g_j$ is an MMPS expression of the form $f$.

We were unable to satisfactorily construct such a model using the most common definition of Max-Plus systems [4]. Our approach followed that of Baccelli et. al., which explicitly models the arrival times of the last $W$ packets in a state vector [3]. $W$ is set to be the maximum possible size of the congestion window. This results in a max plus linear system of the form $Z(n) = A_w(n)Z(n-1)$ where $Z$ is a concatenated vector of packet arrival times and $A_w$ is a matrix describing the system evolution for the packet $n$ with a congestion window of size $w$. The details of this model are described in Appendix A.1.

Unfortunately, there are three major shortcomings with the approach. Firstly, BBR's bandwidth estimation process is not realizable as an MMPS expression, let alone as a Max-Plus operation. This makes dynamically modeling window size challenging. Baccelli et al. avoid this challenge because they study TCP Reno and Tahoe, which both have simpler window dynamics that can be precomputed for an $n$ packet sequence [3]. It may be possible, though certainly trickier, to compute the window sequence for BBR. However, we will

show in Section 4 that a service curve framework provides a mathematically concise approach for modeling window dynamics as a better alternative.

The second shortcoming is that the scale of our analysis is theoretically limited by the complexity of algorithms for studying Max Plus (and MMPS) systems. Although max-plus systems can be interpreted as timed processes on graphs, the relevant graph algorithms are too slow. Computation of max-plus eigenvalues, which is useful for throughput analysis, has time complexity $\mathcal{O}(W^2 \log W)$ in our setting [18]. At the bandwidth limits we are interested in, this time complexity pushes the limits. In contrast, the Network Calculus framework presented in Section 4 is mathematically concise and can be parameterized efficiently.

Finally, the max-plus framework lacks high quality software tooling. The MATLAB toolbox I used is prone to both bugs and large constant factors in the algorithms' implementation [18]. On a few different systems, computing the eigenvalue of the max-plus system entered an infinite loop. A tremendous benefit for practitioners could be achieved with the creation of a high performance, well tested software library for modeling and manipulating Max Plus systems.

### 3.3 Piecewise Affine (PWA) Systems

PieceWise Affine (PWA) systems are dynamical systems with a fixed number of $n$ modes, where for each mode $i \in [n]$, the dynamics are specified by a Linear Time Invariant system, with the mode determined by which portion of the state and input space the system is in. Formally,

$$\mathbf{x}(t) = A_i\mathbf{x}(t) + B_i\mathbf{u}(t)$$
$$\mathbf{y}(t) = C_i\mathbf{x}(t) + D_i\mathbf{u}(t)$$

where the system is in mode $i$ if $\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix} \in \Omega_i$; $\Omega_i$ is a convex polyhedra and the set $\{\Omega_i\}$ partitions the entire state/input space. In our experiments, we use discrete timesteps and the state space $\mathbf{x}$ consists of observations at previous timesteps, forming an autoregressive system. This special case is known in the literature as PWARX, although this distinction is not especially important for our work here.

The PWA systems identification problem using global optimization is NP-hard. Some approaches, such as in [16], approximate the global optimization problem resulting in a polynomial complexity mixed integer quadratic programming problem, but this is still too costly for the scale we need.

In practice, heuristic approximations that result in suboptimal solutions are good enough. I examined two of these methods. Lauer proposes k-LinReg which uses the k-means clustering algorithm to develop a heuristic approximation of the partition $\{\Omega_i\}$ followed by regression to identify the dynamical system [13]. Lauer et al. also suggest using the Multilevel Coordinate Search (MLCS) algorithm combined with sequential quadratic programming to minimize an error objective [14].

However, both methods struggle to recover the true underlying modes using solely packet throughput data. An example of this failure is in Figure 3 where one can see that k-LinReg with $k = 2$ identifies the ProbeBW states beginning at time $t = 25$ and $t = 45$ as two different modes. In 5 trials across modes ranging from 2 to 5, there was not a single instance where k-LinReg or MLCS could assign ProbeBW states the same mode and correctly distinguish the ProbeBW and ProbeRTT states as two different modes. Both methods were tested from $k = 2$ to $k = 5$. While ideally only two modes would be identified, extra modes were added for flexibility to capture transition dynamics.

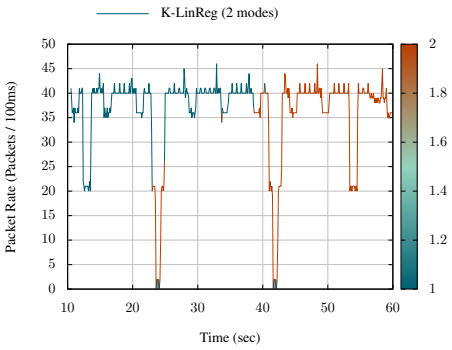


Figure 3: Latency ($D$) vs Throughput (Mbps)

### 3.4 Conclusion

Despite their initial failures, hybrid systems seem promising for modeling network traffic flow, which often consist of discrete state transitions along with approximately continuous dynamics for packet throughput. The MMPS framework may not be flexible enough for our work, but the PWA approach seems usable. The main problem in the data driven approach presented here is that learning a dynamical system becomes much more challenging without also using internal state measurements.

In real systems, state transitions can be driven by internal state machines whose operations are not immediately visible from external metrics like packet throughput. In traditional control parlance, we do not have fully observable systems but we still want to recover a full system model. In this setting, identifying an accurate hybrid systems model would require both careful data collection and a deliberate effort to appropriately configure the state automaton when possible. This may be a worthwhile exploration for future research.

## 4. NETWORK CALCULUS IN THE MIN PLUS FRAMEWORK

The Network Calculus framework, first introduced in [6] and [7], provides a method for studying network flows mathematically. In comparison to the MMPS model studied earlier, we can concisely represent packet flow dynamics on large networks through the use of service curves, the key element in the Network Calculus framework.

## 4.1 Service Curves

The service curve is at the heart of the Network Calculus framework. We define a service curve $x$ is a non-decreasing function $x : \mathbb{R}^+ \to \mathbb{R}^+ \cup \{+\infty\}$. $x(t)$ is defined as the number of bits that have arrived at (or left) a network element in the time interval $[0, t)$ (the definition varies across the literature, but the concept is the same). In theory, a service curve can be any arbitrary nondecreasing function, but in practice, we need just a few different types of piecewise linear functions. We will also adopt the convention that we only study causal service curves $x$, a service curve that satisfies $x(t) = 0$ for all $t < 0$. We now introduce two classes of service curves.

### 4.1.1 Delay service curve $\delta_d(t)$

Define $\delta_d(t)$ as the piecewise function

$$\delta_d(t) := \begin{cases} t \leq d & 0 \\ t > d & +\infty \end{cases}$$

$\delta_d(t)$ models a delay element of $d$ seconds. Of course in practice, a network element cannot forward infinite data in arbitrary time as $\delta_d$ implies; we will show how to combine $\delta_d$ with other service curves to realistically model a network link.

### 4.1.2 Bandwidth curve $\lambda_b(t)$

Define $\lambda_b(t)$ as the linear function $\lambda_b(t) := bt$ where $b$ is a bandwidth rate of the number of bits per second. The definition is quite intuitive, clearly the amount of data we can send through a bandwidth constrained link over total time $t$ is precisely the product of the bandwidth rate $b$ and the total time $t$.

## 4.2 Min Plus Algebra

The most general form of the Min Plus Algebra (analogous to the Max Plus algebra introduced earlier) is over an arbitrary set $\mathcal{D}$ equipped with the $\oplus$ and $\otimes$ operations, forming a tuple called a dioid. The tuple $(\mathcal{D}, \oplus, \otimes)$ is required to meet properties of a Min Plus algebra. Although out of scope for this report, a full description of the Min-Plus Algebra can be seen in [8].

We focus on defining the Min Plus algebra over service curves equipped with the binary $\otimes$ and $\oplus$ operations. These operations can be used to model the interactions of different network elements. The two operations $\oplus$ and $\otimes$ on service curves can be defined as follows:

$$(x \oplus y)(t) := x \wedge y = \min\{x(t), y(t)\}$$
$$(x \otimes y)(t) := x * y = \inf_{s:0\leq s\leq t} \{x(t-s) + y(s)\}$$

The $\oplus$ and $\otimes$ are the point wise minimum and convolution with infimum respectively. For clarity, instead of using the notation for the Min-Plus framework $\otimes$ and $\oplus$, we will use the operations in $\mathbb{R}$. That is, we will use $x \wedge y$ to describe $x \oplus y$, the pointwise minimum, and $x * y$ to denote $\otimes$, the convolution using $\inf$.

LEMMA 1. *Suppose $\beta_1$ and $\beta_2$ are the service curves of network elements $\mathcal{S}_1$ and $\mathcal{S}_2$ in sequence. Then $\beta_1 * \beta_2$ is the service curve of the combined concatenated elements.*

PROOF. Lemma 1 restates Theorem 1.4.6 from Le Boudec and Thiran. See [15] for the theorem and proof. □

### 4.2.1 A delay and bandwidth constrained network link

Now that we have defined the convolution operation $*$, we can define the service curve of a network link constrained by both a delay $d$ and a bandwidth $b$. We denote this service curve $L_{d,b}(t)$. $L_{d,B}(t) = (\delta_d * \lambda_b)(t)$ by application of Lemma 1. Observe that $L_{d,b}(t)$ is defined as

$$L_{d,b}(t) := \begin{cases} t \leq d & 0 \\ t > d & b(t-d) \end{cases}$$

The definition is intuitively clear from how we'd expect a delay and bandwidth constrained service curve to function. We leave a more detailed explanation of how $L_{d,b}(t)$ arises from the convolution $\delta_d * \lambda_b$ due to Lemma 1 in Appendix B.1.

### 4.2.2 A sequence of delay and bandwidth constrained network elements

Given a sequence of network elements $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, ...\mathcal{S}_n$ with service curves $L_{d_i,B_i}$ for the network element $\mathcal{S}_i$, the concatenated service curve is $L_{D,B}$ where $D = \sum_i d_i$ and $B = \min_i\{b_i\}$. The proof follows from applying Lemma 1 to a sequence of delay and bandwidth constrained network links. This brief observation highlights an important feature of service curves (and Lemma 1) - we can abstract an entire sequence of $n$ network elements $\{\mathcal{S}_i\}$ just by performing an $n$-fold convolution to get a service curve for the combined element $\mathcal{S}$.

## 4.3 Quality of Service (QoS) Metrics

Since we can abstract concatenated network elements as a single element, we can study the properties of the overall network by studying the service curve of the combined element $\mathcal{S}$. Let $\mathcal{S}$ have an input (arrival) service curve $\alpha(t)$ and output (departure) service curve $\beta(t)$. I now present three different QoS metrics on $\mathcal{S}$.

### 4.3.1 Windowed Average Throughput

Suppose we have a window of $W$ seconds. Define the windowed averaged throughput as

$$\text{Tpt}_W(t) := \frac{\beta(t) - \beta(t-W)}{W}$$

$\text{Tpt}_W(t)$ represents the average throughput over the $W$ seconds preceding $t$. Additionally, we denote $\text{Tpt}(t) := \text{Tpt}_t(t)$ or the throughput averaged over all time.

### 4.3.2 Input-Output Delay

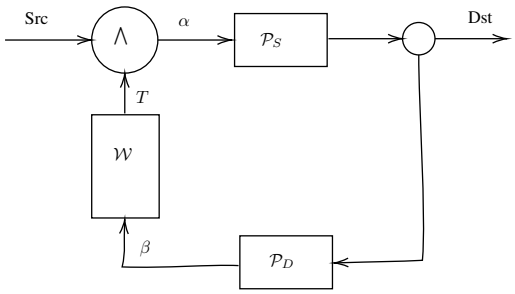


Figure 4: Model of BBR window flow controller in closed loop system

The delay as a function of time is defined as

$$D_{\alpha,\beta}(t) := \inf\{s \geq 0 : \alpha(t) \leq \beta(t+s)\}$$

Intuitively, if $\beta$ achieves the same value as $\alpha$ some $s$ seconds later, we can be guaranteed that any bits sent from the input at time $t$ experienced a delay of $s$ seconds.

#### *4.3.3 Queue Size*

Sometimes called the backlog in the literature, the queue size is the amount of built up data that has not yet exited $\mathcal{S}$. Define the queue size $Q_{\alpha,\beta}(t)$ as

$$Q_{\alpha,\beta}(t) := \alpha(t) - \beta(t)$$

which is simply the amount of data that has entered but not exited $\mathcal{S}$.

## 5. MODEL

We consider the original three-node network modeled as a closed loop system in Figure 4. We make the simplifying assumption that the acknowledgments sent back to the destination follow an identical sequence of links. Although in practice acknowledgements are significantly smaller, we will model the acknowledgment process by assuming that the originally sent data is passed through back to the source. By Lemma 1, we can concatenate any arbitrary sequence of network elements and represent the process as a single element with a single service curve.

We denote $\mathcal{P}_S$ as the service curve representing the path from the Source (Src) to the Destination (Dst) and $\mathcal{P}_D$ as the service curve representing the path back from the Destination to the Source. The joint pathway is represented as $\mathcal{P} = \mathcal{P}_S * \mathcal{P}_D$. Throughout our analysis we will assume that the source process is an infinite data stream with data instantaneously available ($\text{Src}(t) = +\infty$ for all $t$).

As Figure 4 illustrates, $\alpha$ is the input service curve describing incoming traffic leaving the source and entering the network. $\beta = \alpha * \mathcal{P}$ is the output network service curve, which the window process $\mathcal{W} : \mathbb{R}^+ \to \mathbb{R}^+$ uses to estimate a window size.

LEMMA 2. *The network input process $\alpha$ in the closed loop with the window controller is $\alpha = \beta + \mathcal{W}$. The input process $\alpha$ is recursively specified $\alpha = \alpha * \mathcal{P} + \mathcal{W}$.*

PROOF. By the definition of a window process $\mathcal{W}$, we require that the inflight data is upper bounded by $\mathcal{W}$. Assume that the process is optimal, and we can achieve equality: $Q_{\alpha,\beta} = \mathcal{W}$. $T$ is the throttling process that is influenced by $\mathcal{W}$. Since $\alpha = \text{Src} \wedge T$, we have $\text{Src} \wedge T - \beta = \mathcal{W}$. Lastly, since Src is assumed to be an infinite data stream with instantaneous availability, we can simplify $\text{Src} \wedge T - \beta = \mathcal{W}$ to get $T = \beta + \mathcal{W}$. Substitute this definition into $\alpha$, again applying the definition of the Src data process, and we achieve the stated result for $\alpha$. To get $\alpha = \alpha * \mathcal{P} + \mathcal{W}$, we just substitute $\beta = \alpha * \mathcal{P}$. This result is similar to the result presented in Section B. of [1] □

### 5.1 BBR Estimation Processes

We now define the BBR estimation process using service curves with a few simplifying assumptions. We approximate the BBR Bandwidth estimation process, assuming that BWEst is windowed over time rather than packets. We also assume that the valid range for both RTT and Bandwidth estimates are the last 10 seconds. This is partially consistent with BBR. BBR RTT estimates are valid for 10 seconds, while no such constraint is placed on the bandwidth estimate, which updates more frequently and uses a window over packets.

Let $\text{Window}_{\text{BW}}$ be the window over which BBR computes the bandwidth estimate. Then define the bandwidth estimation process $\text{BWEst}(t)$ and the round trip time estimation process $\text{RTTEst}(t)$ as follows:

$$\text{BWEst}(t) := \sup_{0 \leq \delta \leq 10} \text{Tpt}_{\text{Window}_{\text{BW}}}(t) \tag{1}$$

$$\text{RTTEst}(t) := \inf_{0 \leq \delta \leq 10} D_{\alpha,\beta}(t - \delta) \tag{2}$$

$$\mathcal{W}(t) := g \cdot \text{RTTEst}(t) \cdot \text{BWEst}(t) \tag{3}$$

Each of these quantities can be estimated using just $\alpha$ and $\beta$, the input and output service curves. Technically, $\mathcal{W}(t) := \min\{g \cdot \text{RTTEst}(t) \cdot \text{BWEst}(t), \mathcal{W}_{\text{max}}\}$ where $\mathcal{W}_{\text{max}}$ is the maximum possible window size, but the simplification in (3) does not affect our analysis.

### 5.2 BBR State Dynamics

We are specifically interested in studying the steady state dynamics of BBR. In the steady state, BBR transitions through just the ProbeBW and ProbeRT cycles. In these two states, the dynamics are easier to study because the only change in behavior is in the process $\mathcal{W}$. For the ProbeBW cycle, the gain $g$ switches from 1, 1.25, and 0.75 with each gain phase lasting for a single RTT. ProbeBW continuously switches through the 8-gain cycle for 10 seconds. For the ProbeRT cycle, the $\mathcal{W}$ process is fixed to be just 4 packets (32k bits) for a duration of 200ms and a single RTT. Thus, our model is sufficiently detailed to study the steady state behavior of BBR.

### 5.3 Network Parameters

Finally, we define our notation for the network parameters.

We let $L^0_{d_0,b_0}$ denote the service curve for the link from the source to the middle router, and $L^1_{d_1,b_1}$ denote the service curve for the link from the router to the final destination. We assume that the service curves are identical for the reverse path. Thus, $\mathcal{P}_S = L^0_{d_0,b_0} * L^1_{d_1,b_1}$, and $\mathcal{P}_D = L^1_{d_1,b_1} * L^0_{d_0,b_0}$. We also make a significant assumption that routers have infinite queues that do not drop packets; an admittedly large assumption.

# 6. ANALYSIS

The full model of BBR's window process using service curves, presented in equations (1), (2), and (3), is complicated to analyze. We walk through simplifications of $\mathcal{W}$ before arriving at a final result. In order to make these conclusions, we first state two critical results from [1] that we will make use of in our analysis.

LEMMA 3. *Suppose $\mathcal{W}(t) = \bar{W}$ where $\bar{W}$ is an arbitrary constant. Then there is a unique process $\bar{\alpha}$ satisfying $\alpha \leq \alpha * \mathcal{P} + \bar{W}$ where $\bar{\alpha}$ is guaranteed a maximum service curve $\bar{\mathcal{S}}_{\mathcal{W}} := \wedge_{m=0}^{\infty} \left(\mathcal{P} + \bar{W}\right)^{(m)}$ where $\left(\mathcal{P} + \bar{W}\right)^{(m)}$ denotes the $m$-fold convolution of $\left(\mathcal{P} + \bar{W}\right)$ with itself. That is, $\bar{\alpha} \leq$ Src $* \bar{\mathcal{S}}_{\mathcal{W}}$*

LEMMA 4. *Suppose $\mathcal{W}(t) = W$ where $W$ is an arbitrary constant. Then there is a unique process $\alpha^*$ satisfying $\alpha \geq \alpha * \mathcal{P} + W$ where $\alpha^*$ is guaranteed a minimum service curve $\mathcal{S}_{\mathcal{W}} := \wedge_{m=0}^{\infty} (\mathcal{P} + W)^{(m)}$ where $(\mathcal{P} + W)^{(m)}$ denotes the $m$-fold convolution of $(\mathcal{P} + W)$ with itself. That is, $\alpha^* \geq$ Src $* \mathcal{S}_{\mathcal{W}}$*

PROOF. Both Lemma 3 and Lemma 4 are the results of applying Corollary 18 and Corollary 19 in [1] to our setting. □

## 6.1 Constant BWEst, Constant RTTEst, Constant gain $g = 1$

LEMMA 5. *Suppose the following conditions hold: (1) BWEst is fixed as BWEst$(t) = C$, (2) RTTEst is fixed as RTTEst$(t) = RTT_{true}$, (3) the gain $g$ is fixed to 1, (4) the bottleneck bandwidth is $B$, and (5), the ProbeRT window size is $W_{RT} = 3.2 \cdot 10^4$. Then for the network with service curve $\mathcal{P}$, the input service curve $\alpha$ is:*

$$\alpha_{BW,C\geq B}(t) = \begin{cases} t \leq R \ : & CR \\ t > R \ : & (C-B)R + Bt \end{cases}$$

$$\alpha_{BW,C\leq B}(t) = \begin{cases} t \leq R \ : & CR \\ nR < t \leq nR + \frac{CR}{B} \ : & nR(C-B) + Bt \\ nR + \frac{CR}{B} < t \leq (n+1)R \ : & (n+1)CR \end{cases}$$

*where R is an abbreviation of $RTT_{true}$, $n$ is an arbitrary integer in $\mathbb{N}$, and $C \geq B$ and $C \leq B$ specify the conditions for the appropriate $\alpha$. For the ProbeRT cycle, the input service curve $\alpha$ is given by substituting the constant $C = W_{RT}/R$ into $\alpha_{BW,C\geq B}(t)$ and $\alpha_{BW,C<B}(t)$ to get the service curves $\alpha_{RT,W_{RT}\geq BR}(t)$ and $\alpha_{RT,W_{RT}<BR}(t)$.*

PROOF. With assumptions (1) and (2) on BWEst and RTTEst respectively, $\mathcal{W}$ is a fixed constant during the ProbeBW cycle; $\mathcal{W}(t) = C \cdot \text{RTT}_{\text{true}}$. The BBR dynamics also fix $\mathcal{W}(t) = W_{\text{RT}}$ for the ProbeRT cycle. As stated in Lemma 2, the input service curve $\alpha$ satisfies the equation $\alpha = \alpha * \mathcal{P} + \mathcal{W}$. Secondly, since $\alpha = \alpha * \mathcal{P} + \mathcal{W}$, the conditions of both Lemma 3 and Lemma 4 hold with $\bar{\mathcal{S}}_{\mathcal{W}} = \mathcal{S}_{\mathcal{W}}$. Thus, $\alpha = \text{Src} * \mathcal{S}_{\mathcal{W}}$ and since Src is defined as an infinite data stream instantaneously available, $\alpha = \mathcal{S}_{\mathcal{W}}$.

We first consider the ProbeBW cycle, considering the two cases $C \geq B$ and $C \leq B$ separately (Note that both service curves are the same when $C = B$). We denote these service curves as $\alpha_{\text{BW},C\geq B}$ and $\alpha_{\text{BW},C\leq B}$. We compute $\mathcal{S}_{\mathcal{W}}$ (and therefore $\alpha$) in the two cases by applying the $m$-fold convolution and taking the pointwise minimum to obtain the specified service curves. Now we can consider the ProbeRT state. Note that our results for $\alpha_{\text{BW},C\geq B}$ and $\alpha_{\text{BW},C\leq B}$ hold for an arbitrary constant $C$ with fixed window size $C\text{RTT}_{\text{true}}$, so we can substitute the value $C = W_{\text{RT}}/\text{R}$ into the results above. This lets us apply the results for $\alpha_{\text{BW},C\geq B}$ and $\alpha_{\text{BW},C\leq B}$ by substituting $C = W_{\text{RT}}/\text{R}$. □

THEOREM 1. *Suppose that conditions (1-5) from Lemma 5 hold. Additionally, assume (6) the duration of the ProbeRT state is $T_{RT}$, (7) the duration of the ProbeBW state is $T_{BW}$, (8) $C \leq B$, and (9) $W_{RT} \leq B \cdot RTT_{true}$.*

*Then the average throughput is a constant given by*

$$Tpt = \frac{T_{BW}C + T_{RT}W_{RT}/RTT_{true}}{T_{BW} + T_{RT}}$$

*The time varying source to destination delay is*

$$D_{\alpha,Dst}(t) := \frac{RTT_{true}}{2}$$

*for both the ProbeBW and ProbeRT cycles. The queue size at the middle router is 0 for both the ProbeBW and ProbeRT cycles.*

PROOF. Since the conditions of Lemma 5 hold, we can apply the stated results of Lemma 5 to obtain the input service curve $\alpha$ for any cycle and for the case $C \leq B$. Apply the definition $\beta = \alpha * \mathcal{P}$ (or Dst $= \alpha * \mathcal{P}_1$) and we have the corresponding output service curve. Then we can recover the stated results by applying each of the three definitions from Section 4.3 on the Quality of Service metrics. While the overall approach is straightforward, there are many details in the full proof, so we leave it to Appendix B.2. □

Before we proceed to more complicated settings, we first discuss the preliminary results here. A reader may notice that we ignore the case $C > B$ in Theorem 1. As we will show in Section 6.3, the assumption of fixed dynamics is not compatible with the assumption $C > B$. $C > B$ results in an oscillatory estimate process RTTEst$(t)$; the dynamics are specified in Lemma 6.

Theorem 1 shows that conservatively underestimating the bandwidth as $C \leq B$ results in a slight throughput loss during the ProbeBW state while guaranteeing optimal latency

and queue sizes throughout BBR's execution. Finally, and most importantly, Theorem 1 shows that even in this very simple setting, we already see the effect of long latencies on reducing throughput. With $T_{\text{BW}}$ fixed to 10 seconds in BBR, and $T_{\text{RT}}$ typically set to $T_{\text{RT}} = 0.2 + \text{RTTEst}(t)$, the throughput $\text{Tpt}(t)$ will be interpolated between $C$ and $W_{\text{RT}}/\text{RTT}_{\text{true}}$, decreasing as $\text{RTTEst}(t)$ increases.

## 6.2 Constant Inaccurate Estimates

The analysis in Lemma 5 and Theorem 1 is more general than initially implied because the results only depend on the window process $\mathcal{W}$. Instead of restricting possible error to the bandwidth estimate BWEst, we assume multiplicative error and arbitrary nonzero gain $g$, showing that a result similar to Theorem 1 holds.

COROLLARY 1. *Suppose the following conditions hold: (1) the bottleneck bandwidth is $B$, (2) BWEst is constant as $\mathit{BWEst}(t) = C_1 B$, (3) RTTEst is constant as $\mathit{RTTEst}(t) = C_2 \mathit{RTT}_{true}$, (4) the gain $g$ is nonzero, and (5) the ProbeRT window size is $W_{RT}$ satisfying $W_{RT} < B \cdot \mathit{RTT}_{true}$. Define $C = C_1 C_2 g$ and assume (6) $C \leq 1$.*

*Then the average throughput is a constant given by*

$$\mathit{Tpt} = \frac{T_{BW} C B + T_{RT} W_{RT} / \mathit{RTT}_{true}}{T_{BW} + T_{RT}}$$

*The time varying source to destination delay is*

$$D_{\alpha,Dst}(t) := \frac{\mathit{RTT}_{true}}{2}$$

*for both the ProbeBW and ProbeRT cycles. The queue size at the middle router is $0$ for both ProbeBW and ProbeRT cycles.*

PROOF. The full proof is in Appendix B.3. The general idea is that we can reframe conditions (1-4, 6) as the conditions of Theorem 1 and then apply Theorem 1. □

## 6.3 Full Dynamics with Fixed Network

LEMMA 6. *Suppose that the following conditions hold: (1) the bottleneck bandwidth is $B$, (2) BWEst is a constant $\mathit{BWEst}(t) = C_1 B$, (3) the initial RTTEst estimate is $C_2 \mathit{RTT}_{true}$, (4) the gain $g$ is nonzero, (5) the ProbeRT window size $W_{RT}$ satisfies $W_{RT} < B \cdot \mathit{RTT}_{true}$, and (6) $C > 1$ where $C = C_1 C_2 g$.*

*Define $T_{RT}^{(k)}$ as the duration of the ProbeRT state the $k^{th}$ time BBR enters the ProbeRT state. Define $\mathit{RTTEst}^{(k)}$ as the RTT estimate after the $k^{th}$ occurrence of the ProbeRT cycle.*

*Then $\mathit{RTTEst}^{(k)}$ has initial estimate*

$$\mathit{RTTEst}^{(0)} = C_2 \mathit{RTT}_{true}$$

*and evolves as*

$$\mathit{RTTEst}^{(k)} = \max\left\{ C_1 g \cdot \mathit{RTTEst}^{(k-1)} - T_{RT}^{(k-1)} \left(1 - \frac{W_{RT}}{\mathit{RTT}_{true} B}\right), \mathit{RTT}_{true} \right\}$$

*with $T_{RT}^{(k)}$ defined as $T_{RT}^{(k)} = 0.2 + \mathit{RTTEst}^{(k)}$.*

PROOF. Proof in Appendix B.4. □

Lemma 6 shows that when considering the full dynamics, overestimating the bottleneck bandwidth or RTT has a doubly damaging effect. Not only does overestimating increase $C$ resulting in throughput reduction as already showed in Theorem 1, but it also increases the overestimation of $\text{RTT}_{\text{true}}$. These effects can cascade and result in secondary increases on delay and queue sizes while decreasing throughput through a cycle of incorrect RTT estimates leading to longer times for $T_{\text{RT}}$.

Further analysis beyond the specification in Lemma 6 is difficult. Unlike Theorem 1, finding a closed form expression is challenging because Lemma 6 includes the $\max$ operator. Nonetheless, we can still derive upper and lower bounds for the relevant results with some reasonable assumptions to obtain Theorem 2.

THEOREM 2. *Suppose that the following conditions hold: conditions (1-5) from Lemma 6 and let $C = C_1 C_2 g$. Denote bounds for the average throughput, delay, and queue size as follows: Tpt is bounded as $\underline{\mathit{Tpt}} \leq \mathit{Tpt} \leq \overline{\mathit{Tpt}}$, the delay $D_{\alpha,Dst}(t)$ is similarly bounded as $\underline{D_{\alpha,Dst}} \leq D_{\alpha,Dst}(t) \leq \overline{D_{\alpha,Dst}}$, and the queue size at the middle router $Q_1(t)$ is bounded as $\underline{Q_1} \leq Q_1(t) \leq \overline{Q_1}$. Then the bounds are:*

$$\underline{\mathit{Tpt}} = \frac{T_{BW} \min\{CB, B\} + (0.2 + \mathit{RTT}_{true}) W_{RT} / \mathit{RTT}_{true}}{T_{BW} + 0.2 + \mathit{RTT}_{true}}$$

$$\overline{\mathit{Tpt}} = \frac{T_{BW} \min\{CB, B\} + (0.2 + \max\{C, 1\} \mathit{RTT}_{true}) W_{RT} / \mathit{RTT}_{true}}{T_{BW} + 0.2 + \max\{C, 1\} \mathit{RTT}_{true}}$$

$$\underline{D_{\alpha,Dst}}(t) = \frac{\mathit{RTT}_{true}}{2}$$

$$\overline{D_{\alpha,Dst}}(t) = \mathit{RTT}_{true} \max\left\{ C - \frac{1}{2}, \frac{1}{2} \right\}$$

$$\underline{Q_1} = 0$$

$$\overline{Q_1} = \max\{C - 1, 0\} \cdot B \cdot \mathit{RTT}_{true}$$

PROOF. The full proof is in Appendix B.5. To prove Theorem 2 we consider both $C \leq 1$ and $C > 1$, using the results of Lemma 5, Corollary 1, and Lemma 6. Then we apply the Quality of Service metrics from Section 4.3. □

## 6.4 Empirical Comparison

We can check how the theoretical results derived in Section 6.3 hold on a real instance of BBR. Revisiting our first experiment in Section 2, we compare our theoretical bounds in Theorem 2 to the empirical results presented in Figure 2 in Section 2. Figure 5 matches Figure 2 qualitatively in key aspects. The theoretical bounds predict bottleneck utilization dropping off at 100ms latency on the bottleneck link. And in both figures, we see that lower bandwidth links have slightly higher utilization then higher bandwidth links. Yet closer examination reveals that the computed results differ significantly - Figure 5 predicts a little over 80% utilization of the bottleneck with 1 second latency on the bottleneck link. In contrast, the actual results show that BBR only achieved around 60-70% utilization in an identical setting. It is likely that the additional underutilization comes from the slow start which takes a much higher proportion of the experiment as $\text{RTT}_{\text{true}}$ increases.

In a separate experiment, we fix the bottleneck delay to 5ms and the bottleneck bandwidth to 1Mbps (results shown in Figures 6 and 7). We now consider the RTTEst$^{(k)}$ dynamics predicted by Lemma 6. In the first stage, although $C_2 \approx 1$, $C_1 > 1$, so this creates an oversized congestion window with $C > 1$. It is immediately clear from Figure 6 that the dynamics predicted by Lemma 6 are present qualitatively. After the first ProbeBW cycle, RTTEst is never measured accurately and continues to oscillate away from the true value. This results in an oversized window $\mathcal{W}$, and causes both delays and queue size (shown in Figure 7) to grow. However, while Lemma 6 estimates the first RTTEst jump correctly (estimate of $1.6\text{RTT}_{\text{true}}$), it does not estimate future evolutions correctly: RTTEst$^{(k)}$ stabilizes in our empirical results while Lemma 6 predicts continued growth.

Like throughput, the upper bound for Queue Size $\overline{Q_1}$ appears to have the right relationship (increasing with $C$), but the wrong numbers. Using recorded measurements, we can take $C \approx 1.75$ after $t = 20$ ($C_1 = 1.05$, $C_2 = 1.66$). Applying the bound $\overline{Q_1}$ from Theorem 2 yields an estimated queue size upper bound of 22500 bits, or roughly 3 packets. This is significantly different from the observed queue sizes in Figure 7 (although curiously, the difference between the initial queue size when $\text{RTT}_{\text{true}}$ is correctly estimated and the queue size after $t = 20$ seconds is in fact about 3 packets).

## 6.5 Discussion

In summary, the analytical results were consistent qualitatively with empirical evidence, but not consistent numerically. Qualitative results are not entirely pointless. The results here suggest two opportunities to improve BBR's throughput decay over long latencies. One is to keep a constant ratio of $T_{\text{BW}} : T_{\text{RT}}$ so that the overhead of the ProbeRT state is constant even during long latencies. Another alternative is the approach taken by the original authors of the BBR team in the follow up work [5], where $W_{\text{RT}}$ is set to $0.5\text{BWEst}(t) \cdot \text{RTTEst}(t)$ instead of $3.2 \cdot 10^4$. One drawback of this approach is that it is unclear how robust such an algorithm would be to mismeasurement. While our empirical studies suggest tolerance to a congestion window oversized by a factor of 2, our theoretical results show the possibility that such a window would create continued queue build up. It is possible that the other modifications of BBR v2 in [5] remedy these issues, but this would need further analysis.

# 7. CONCLUSION

We have presented three different methods for modeling and analyzing BBR. The Network Calculus was the most tractable framework, and led to a number of theoretical characterizations of BBR regarding throughput, queue size, and delays, that were shown to hold qualitatively. However, closer inspection revealed that many of these results were numerically inaccurate, highlighting a few shortcomings of the Network Calculus approach.

Working in the Network Calculus framework was tedious.

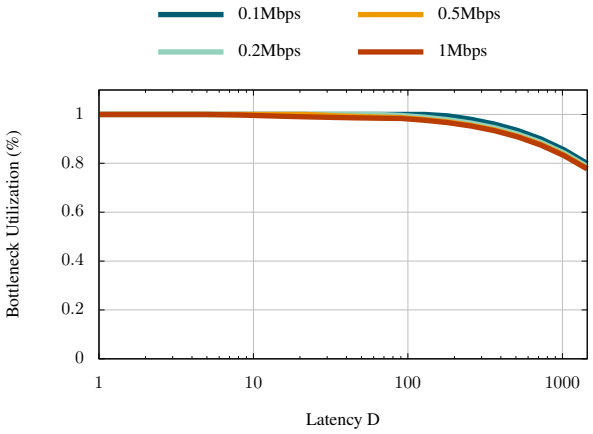


Figure 5: Predicted throughput from Theorem 2 using network parameters

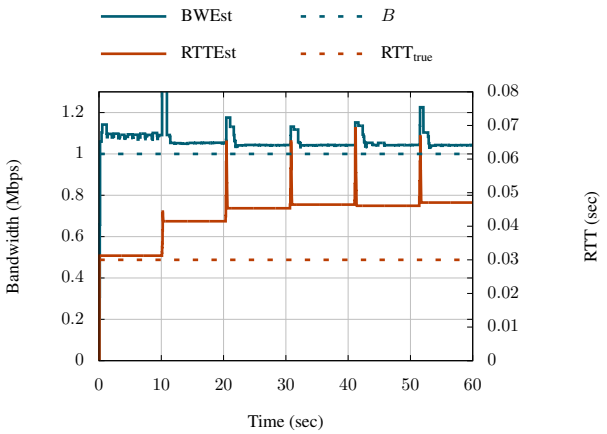


Figure 6: Empirical BWEst and RTTEst evolution compared with true values

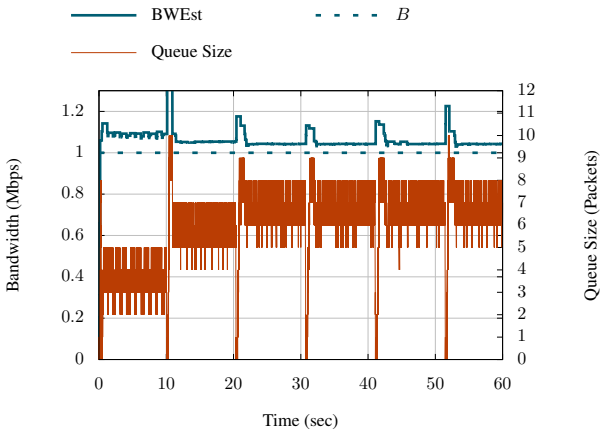


Figure 7: Queue size evolution of three-node network

Deriving precise bounds depends on one's willingness to rigorously model and convolve service curves, which is often a manual approach for the types of piecewise functions that are useful. Added effort does not guarantee model accuracy either, as this precision comes at the price of model bias. The stochastic network calculus outlined in [12] may be a promising alternative that adds some needed flexibility to the network calculus framework.

A more general issue that arises from using the Network Calculus is the inability to use data driven methods. While the hybrid systems approach did not work appropriately in our setting, it is possible that more carefully recording internal state data along with a few well chosen inductive model biases would help data driven methods for hybrid systems yield useful models.

Data driven approaches using hybrid systems are especially desirable because of the large arsenal of control theoretic analysis tools for these types of systems. The type of analysis done for the RTTEst dynamics in Lemma 6 is functionally similar to a control theoretic analysis. We studied the long term trajectory of the RTTEst process, with the aim to find parameters and check whether the BBR algorithm would stabilize RTTEst to the true value $\text{RTT}_{\text{true}}$. Unfortunately, while the initial aim of this project was to achieve a control theoretic analysis of BBR, the Network Calculus framework lacked these tools. A data driven control theoretic analyzer would be an invaluable tool for studying congestion control algorithms.


## Acknowledgments

Thanks to Anirudh Sivaraman for advice and guidance.


## 8. REFERENCES


[1] R. Agrawal, R. L. Cruz, C. Okino, and R. Rajan. Performance bounds for flow control protocols. *IEEE/ACM Transactions on Networking*, 7(3):310–323, 1999.

[2] E. Atxutegi, F. Liberal, H. K. Haile, K.-J. Grinnemo, A. Brunstrom, and A. Arvidsson. On the use of tcp bbr in cellular networks. *IEEE Communications Magazine*, 56(3):172–179, 2018.

[3] F. Baccelli and D. Hong. TCP is max-plus linear and what it tells us on its throughput. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 219–230, 2000.

[4] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. BBR: Congestion-based congestion control. *ACM Queue*, 14, September-October:20 – 53, 2016.

[5] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, and V. Jacobson. Bbrv2: A model-based congestion control.

[6] R. Cruz. A calculus for network delay. i. network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[7] R. Cruz. A calculus for network delay. ii. network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[8] M. Daniel-Cavalcante, M. Magalhaes, and R. Santos-Mendes. The max-plus algebra and the network calculus. In *2006 8th International Workshop on Discrete Event Systems*, pages 433–438, 2006.

[9] W. P. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[10] M. Hock, R. Bless, and M. Zitterbart. Experimental evaluation of bbr congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10, 2017.

[11] B. Jaeger, D. Scholz, D. Raumer, F. Geyer, and G. Carle. Reproducible measurements of TCP BBR congestion control. *Computer Communications*, 144:31–43, 2019.

[12] Y. Jiang, Y. Liu, et al. *Stochastic network calculus*, volume 1. Springer, 2008.

[13] F. Lauer. Estimating the probability of success of a simple algorithm for switched linear regression. *Nonlinear Analysis: Hybrid Systems*, 8:31–47, 2013.

[14] F. Lauer, G. Bloch, and R. Vidal. A continuous optimization framework for hybrid system identification. *Automatica*, 47(3):608–613, Jan. 2011.

[15] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.

[16] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

[17] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, and G. Carle. Towards a deeper understanding of tcp bbr congestion control. In *2018 IFIP networking conference (IFIP networking) and workshops*, pages 1–9. IEEE, 2018.

[18] J. Stanczyk. *Max-Plus Algebra Toolbox for Matlab version 1.7*. 2016.

[19] A. J. Van Der Schaft and J. M. Schumacher. *An introduction to hybrid dynamical systems*, volume 251. Springer London, 2000.


# APPENDIX

## A. HYBRID SYSTEMS

### A.1 A detailed Max Plus model

As we showed in Section 4, there are alternative definitions of the max/min plus framework where the Max Plus algebra is defined over service curves rather than the reals ($\mathbb{R}$). A few papers, such as [3] successfully use the Max Plus framework instead of service curves.

We experiment with the model proposed by [3]. Let $i$ vary from 0 to 2 (for the source, midpoint router, and destination). Let $y_i(n)$ be the departure time of the $n^{\text{th}}$ packet from $i$. Let $Y(n) = (y_0(n), y_1(n), y_2(n))^T$, or a vector representing the departure times at all routers. Finally, let $Z(n) = [Y(n)^T, Y(n-1)^T, ...Y(n-\mathcal{W}_{\max}+1)^T]^T$, a vector representing all the departure times of the last $\mathcal{W}_{\max}$ packets where $\mathcal{W}_{\max}$ is the maximum possible window size.

Finally, let $M(n)$ be a matrix describing induced delays from $Y(n-1)$ to $Y(n)$. For any two routers $i, j$, the departure time of packet $n$ from router $j$ affects router $i$ only if $j \leq i$. This leads to the definition $[M(n)]_{ij} = \sum_{k=j}^{i} \sigma_k(n) + \sum_{k=j}^{i-1} d_{k,k+1}$. Here, $d_{i,i+1}$ is the delay (ex 1 ms) while $\sigma_k(n)$ is the service time, inversely proportional to the available bandwidth. We define $A_W(n)$ identically as [3].

Finally, the window process $\mathcal{W}(n)$ is determined by the gain and rolling estimates of the RTT (denoted RTTEst) and bandwidth (denoted BWEst). Combining these estimates. The model generalizes to $K$ routers in sequence.

$$
\begin{aligned}
Z(n) &= A_W(n)Z(n-1) \\
\mathcal{W}(n) &= \min\{gain \cdot RTTEst(n) \cdot BWEst(n), \mathcal{W}_{\max}\} \\
RTTEst(n) &= \min_{1 \leq j \leq RttEstWindow} \{Y_2(n-j) - Y_0(n-j)\} \\
BWEst(n) &= \max_{1 \leq j \leq BWEstWindow} \left\{ \frac{n-j-W(n-j)+1}{Y_0(n-j) - Y_2(n-\mathcal{W}(n-j))} \right\}
\end{aligned}
$$

# B. NETWORK CALCULUS AND SERVICE CURVES

## B.1 Proof of $_{d,b}(t)$ definition

To see why $L_{d,b}(t)$ is defined as

$$L_{d,b}(t) := \begin{cases} t \leq d & 0 \\ t > d & b(t-d) \end{cases}$$

and how this definition arises from the convolution $\delta_d * \lambda_b$, we can proceed as follows. First, suppose $t \leq d$. $\delta_d$ is 0 everywhere on the domain $[0, t]$, so when we consider the convolution $(\delta_d * \lambda_b)(t) := \inf_{s:0\leq s\leq t}\{\delta_d(t-s) + \lambda_b(s)$, the term $\delta_d(t-s)$ is 0 everywhere for all $s$. On the other hand, the quantity $\lambda_b(s)$ is clearly minimized at $s = 0$. This gives us the first condition. Now consider when $t > d$. In this setting, $t - s \leq d$, or the term $\delta_d(t-s)$ will be $+\infty$. So $s \geq t - d$. For $s \geq t - d$, $\delta_d(t-s)$ is 0 everywhere, so we can pick the $s$ that minimizes $bs$, which is when we take equality of the bound $s \geq t - d$. This yields $b(t-d)$, giving the second part of the convolution.

## B.2 Proof of Theorem 1

First we examine the throughput. Since we have found the input service curves $\alpha$ that satisfy the desired closed loop system equation $\alpha = \alpha * \mathcal{P} + \mathcal{W}$, and the output service curve of the network element is $\beta = \alpha * \mathcal{P}$, we can now compute the average throughput of the two phases. Since assumption (8) states $C \leq B$ and (9) states $W_{\text{RT}} \leq B\text{R}$, we only need to consider the service curves $\alpha_{\text{BW},C\leq B}$ and $\alpha_{\text{RT},W_{\text{RT}}\leq B\text{R}}$. Evaluating the output service curve $\beta = \alpha * \mathcal{P}$, observe that $\beta_{\text{BW},C\leq B} = \alpha_{\text{BW},C\geq B} - C\text{R}$ (except at $t = 0$) and similarly $\beta_{\text{RT},W_{\text{RT}}\leq B\cdot\text{R}} = \alpha_{\text{RT},W_{\text{RT}}\leq B\cdot\text{R}} - W_{\text{RT}}$.

Since $C < B$ (assumption (8)), the average throughput during the ProbeBW cycle over a single R is $C$, which can be verified by examining $\beta_{\text{BW},C\leq B}$. Similarly, the average throughput during the ProbeRT cycle can be verified as $W_{\text{RT}}/\text{R}$. To compute the average throughput in the limit, we take the time weighted average throughput of a complete state cycle (which corresponds to concatenating the input service curves and studying the result) using assumptions (6) and (7) for the durations of the ProbeRT and ProbeBW cycles. This yields the stated result for Tpt. Note that we can ignore the initial R seconds where $\beta$ is 0 since this is negligible in the limit.

Next, we can evaluate the delay $D_{\alpha,\text{Dst}}$. Since for any time $t$ such that $t \geq \text{R}$, both $\alpha$ and $\text{Dst} = \alpha * \mathcal{P}_1$ have identical slopes, it suffices to compare the horizontal distance at any time $t \geq \text{R}$. We can solve for $t^*$ such that $\alpha(\text{R}) = \alpha * \mathcal{P}_1(t^*)$, then subtract R from $t^*$ to get the stated result. Note that this estimation process holds for both $\alpha_{\text{BW}}$ and $\alpha_{\text{RT}}$. Since $C < B$, the delay is at least $\text{R}/2$.

Lastly, the queue size can be estimated by recognizing that at the middle router, the output service curve has a larger slope than the input. I.e., $L^1_{d_1,b_1}$ has a larger slope than $\alpha * L^0_{d_0,b_0}$. This implies that any queue build up at the middle router will be transient and drained rapidly. This yields the estimated queue size of 0.

## B.3 Proof of Corollary 1

Consider the window (a constant): $\mathcal{W} = C_1C_2gB\text{RTT}_{\text{true}}$. We let the $C$ from Theorem 1 be the quantity $C_1C_2gB$. By assumption (6), $C_1C_2g \leq 1$ so $C_1C_2gB \leq B$. Thus, the window $\mathcal{W}$ with the conditions assumed in Corollary 1 is the same as the window $\mathcal{W}$ with the conditions assumed in Lemma 1. The only difference is that $T_{\text{RT}}$ is now $0.2 + C_2\text{RTT}_{\text{true}}$ instead of $C_2\text{RTT}_{\text{true}}$. Thus, we've verified that all conditions of Theorem 1 hold with any differences reflected in the quantity $T_{\text{RT}}$. Apply the stated reframing of conditions (1-4, 6) ($C = C_1C_2gB$), and we obtain Corollary 1.

## B.4 Proof of Lemma 6

The result can be inductively proven, applying the BBR estimation process to the transitions between ProbeBW and ProbeRT cycles. The initial estimate $\text{RTTEst}^{(0)}$ is $C_2\text{RTT}_{\text{true}}$ by assumption (3). Observe that the congestion window before entering the $\text{k}^{\text{th}}$ ProbeRT cycle is $C_1g\text{RTTEst}^{(k)}$.

Now consider the $\text{k}^{\text{th}}$ time that BBR enters the ProbeRT cycle. By definition, the current estimate for RTT is $\text{RTTEst}^{(k-1)}$. We consider the RTTEst process, defined in (2) as $\inf_{0\leq\delta\leq 10} D_{\alpha,\beta}(t-\delta)$, applied during the ProbeRT sequence. The duration of the ProbeRT sequence is $T_{\text{RT}}^{(k-1)}$, at which point the input rate increases due to a new window size $C_1g\text{RTTEst}^{(k)}$.

Thus, at the end of the ProbeRT cycle, after $T_{\text{RT}}^{(k-1)}$ seconds have elapsed, is when the RTT estimate will achieve its minimum. The total bits put into the system by $\alpha$ is $h = \frac{W_{\text{RT}}}{\text{RTT}_{\text{true}}} \cdot T_{\text{RT}}^{(k-1)}$. The output service curve will be no faster than $B$, so the time taken to output the additional $h$ bits is $\frac{W_{\text{RT}}}{B\text{RTT}_{\text{true}}} \cdot T_{\text{RT}}^{(k-1)}$. The initial delay when the output service curve receives the first bit sent by the ProbeRT cycle is $C_1g\text{RTTEst}^{(k-1)}$. Thus, the minimum estimate will be $C_1g\text{RTTEst}^{(k-1)} - T_{\text{RT}}^{(k-1)} + \frac{W_{\text{RT}}}{B\text{RTT}_{\text{true}}} \cdot T_{\text{RT}}^{(k-1)}$, the first part of the expression for $\text{RTTEst}^{(k)}$.

If the output service curve is restricted by the input $\alpha$ slope before reaching $T_{\text{RT}}^{(k-1)}$ seconds, then the delay must achieve the minimum possible time of $\text{RTT}_{\text{true}}$ seconds. Combining the two possible minimum estimates yields the stated result.

## B.5 Proof of Theorem 2

First, we look at throughput. Consider the case when $C \leq 1$. In this setting, the conditions of Lemma 5 and Corollary 1 are met and we apply those results directly. The RTTEst does not fluctuate, so the bounds are tight, and $\underline{\text{Tpt}} = \overline{\text{Tpt}}$. Now suppose that $C > 1$, so all assumptions of Lemma 6 are met. It's clear that a lower bound of $\text{RTTEst}^{(k)}$ is $\text{RTT}_{\text{true}}$ from the definition. But it's less clear how to extract an upper bound. If we ignore the minimum, observe that $\text{RTTEst}^{(k)}$ is

a recurrent sequence with closed form expression

$$\begin{aligned}\text{RTTEst}^{(k)} \leq &\text{RTTEst}^{(0)} \left(C_1 g - 1 + \frac{W_{\text{RT}}}{B\text{RTT}_{\text{true}}}\right)^k \\ &- 0.2 \left(\frac{1 - \left(C_1 g - 1 + \frac{W_{\text{RT}}}{B\text{RTT}_{\text{true}}}\right)^k}{\frac{W_{\text{RT}}}{B\text{RTT}_{\text{true}}} - C_1 g}\right)\end{aligned}$$

Assuming the quantity $C_1 g - 1 + \frac{W_{\text{RT}}}{\text{RTT}_{\text{true}} B} \leq 1$, we can upper bound $\text{RTTEst}^{(k)}$ as $C_1 g C_2 \text{RTT}_{\text{true}} = C\text{RTT}_{\text{true}}$. Hence, using upper and lower bounds for $\text{RTTEst}^{(k)}$, we can derive upper and lower bounds for $T_{\text{RT}}$.

Finally, we observe that the only change to our proof for throughput bounds in Theorem 1 is that the throughput during ProbeBW is restricted to $B$. Thus, combinining the upper and lower bounds for $T_{\text{RT}}$, the restriction of $B$ for throughput during the ProbeBW cycle, we obtain the stated bounds $\overline{\text{Tpt}}$ and $\underline{\text{Tpt}}$.

Next, we look at the delay $D_{\alpha,\text{Dst}}(t)$ and queue size of the middle node $Q_1$. A lower bound of the delay can be derived by applying the results $\underline{D_{\alpha,\text{Dst}}}(t) = \text{RTT}_{\text{true}}/2$ and $\underline{Q_1} = 0$ when $C < 1$. Now consider the case $C > 1$ to obtain an upper bound. Since the node 1 is where the bottleneck begins, we can look at the difference between the service curves when $C = 1$ and when $C > 1$. When comparing the service curves $\alpha_{\text{BW},C=1} * \mathcal{P}_S$ and $\alpha_{\text{BW},C>1} * \mathcal{P}_S$, we can see that the only difference is a vertical shift by $(C-1)B\text{RTT}_{\text{true}}$. This vertical distance is the additional queue size, providing the upper bound $\overline{Q_1}$. To determine the horizontal shift, divide by $B$ and we obtain $(C-1)\text{RTT}_{\text{true}}$. This is the added delay on top of $\text{RTT}_{\text{true}}$, so we can upperbound the delay as $\text{RTT}_{\text{true}} \cdot (C - 1/2)$. This completes the proof.