Model-Based Insights on the Performance, Fairness, and Stability of BBR

Simon Scherrer  
simon.scherrer@inf.ethz.ch  
ETH Zurich  
Switzerland

Markus Legner  
markus.legner@inf.ethz.ch  
ETH Zurich  
Switzerland

Adrian Perrig  
adrian.perrig@inf.ethz.ch  
ETH Zurich  
Switzerland

Stefan Schmid  
stefan_schmid@univie.ac.at  
TU Berlin/Uni. Vienna  
Germany/Austria

ABSTRACT

Google’s BBR is the most prominent result of the recently revived quest for efficient, fair, and flexible congestion-control algorithms (CCAs). While the performance of BBR has been investigated by numerous studies, previous work still leaves gaps in the understanding of BBR performance: Experiment-based studies generally only consider network settings that researchers can set up with manageable effort, and model-based studies neglect important issues like convergence.

To complement previous BBR analyses, this paper presents a fluid model of BBRv1 and BBRv2, allowing both efficient simulation under a wide variety of network settings and analytical treatment such as stability analysis. By experimental validation, we show that our fluid model provides highly accurate predictions of BBR behavior. Through extensive simulations and theoretical analysis, we arrive at several insights into both BBR versions, including a previously unknown bufferbloat issue in BBRv2.

1 INTRODUCTION

To this day, ever changing applications, traffic patterns, network capacities, and path types prompt research into new and better congestion-control algorithms (CCAs). Most prominent in recent years was Google’s introduction of BBR [9], which was promptly enabled in 2017 for some Google services and thus widely deployed in the public Internet [10]. Since then, several theoretical and experimental studies of the behavior of BBR [16, 24, 50, 53, 57] have identified issues with this first version of BBR, relating to both fairness (especially towards loss-based CCAs) and efficiency (e.g., excessive queue buildup). As a result, BBRv2 [11] has been proposed, triggering another series of evaluation studies [20, 31, 40, 51].

Still, the characterization of BBR performance remains incomplete. Experiment-based studies [20, 24, 31, 40, 50, 53], by their nature, allow statements relating to the concrete network settings in the experiments. Given the variety of scenarios in which a CCA might be deployed, such experimental investigations could only be made exhaustive with great effort and large-scale testbeds which only a minority of researchers has access to. Previous model-based studies contain BBR steady-state models that are highly valuable for specific settings (e.g., deep buffers [56] or wireless links [59]); however, a deep theoretical understanding of BBR also requires a model that is valid for general settings and allows investigation of the convergence process.

In this paper, we complement previous approaches to BBR analysis with a classic approach in CCA research: fluid models consisting of differential equations [18, 36–38, 48, 52, 55]. Such fluid models are unique in their suitability for both efficient simulation and theoretical stability analysis. Enabling efficient simulation is critical because the model must be simulated under a plethora of configurations, including settings that are expensive to build. Enabling theoretical stability analysis is crucial because the equilibria (i.e., steady states) of the CCA dynamics are only relevant for performance characterization if stable in a control-theoretic sense, i.e., if the dynamics actually converge to the equilibria.

While a fluid model for BBR is thus well-suited to complement previous work, constructing such a model is challenging because BBR does not naturally fit into the existing fluid-model framework for loss-based CCAs [37, 55]. In fact, BBR does not exclusively rely on a congestion window affected by loss, but includes traffic pulses for capacity probing and measurement-driven state transitions. By using new techniques, e.g., by mimicking the probing pulses with sigmoid functions, this work establishes the first highly accurate and highly general model of BBR, both for versions 1 and 2.

Our fluid model predicts BBR behavior with high accuracy, which we validate with experiments by means of the network emulator mininet [34]. The validated model allows us to perform extensive systematic simulations of BBR by itself or coexisting with traditional CCAs, confirming BBR performance issues from previous studies and yielding new insights. Moreover, we apply dynamical-system analysis (i.e.,
the Lyapunov method) to our fluid model to identify asymptotically stable equilibria of the BBR dynamics.

Our main contributions are the following:

- We introduce the first accurate and general fluid model for BBR (versions 1 and 2), using new techniques such as sigmoid pulses and mode variables.
- We present extensive, systematic model-based calculations, experimentally validate the results of these calculations, and provide profound insights into fundamental metrics of BBR.
- We analytically identify asymptotically stable equilibria of BBRv1 and BBRv2.
- We confirm both experimentally and analytically that BBRv2 eliminates most of the undesirable behavior of BBRv1, but we also identify settings in which BBRv2 leads to bufferbloat and unfairness.

2 NETWORK FLUID MODEL

In this section, we present our network fluid model, which closely follows the work by Low et al. [37]. However, we have made several improvements to the network model, which we will highlight in the following. We denote function \( f(t) \) by \( f \) and its derivative by \( f' \), unless the argument of the function differs from the default time variable \( t \).

In our model, the network consists of links \( ℓ \) with capacity \( C_ℓ \), buffer size \( B_ℓ \), and propagation delay \( d_ℓ \).

**Link-Arrival Rate.** The arrival rate \( y_ℓ(t) \) at link \( ℓ \) is

\[
y_ℓ(t) = \sum_{i \in U_ℓ} x_ℓ(t - d_ℓ^{i,ℓ}),
\]

where \( U_ℓ \) is the set of agents using link \( ℓ \), \( x_ℓ(t) \) is the sending rate of agent \( i \) at time \( t \), and \( d_ℓ^{i,ℓ} \) is the propagation delay between agent \( i \) and link \( ℓ \). Together with Low et al. [37], we neglect queuing delay and packet losses previous to link \( ℓ \).

**Queue Length.** In general, the queue length grows or shrinks according to the discrepancy between combined arrival rate \( y_ℓ \) and the transmission capacity \( C_ℓ \) at the respective link [37], but never exceeds its buffer size \( B_ℓ \):

\[
q_ℓ(t) = (1 - p_ℓ) \cdot y_ℓ - C_ℓ, \quad q_ℓ(t) \in [0, B_ℓ],
\]

where \( p_ℓ(t) \) is the loss probability of link \( ℓ \) at time \( t \) (cf. §2). We have refined the model by Low et al. to additionally capture the effect of packet drops on the queue length.

**Latency.** The link latency is the fixed link propagation delay plus the queuing delay, which depends on queue size \( q_ℓ(t) \).

The latency of a path is the sum of link latencies:

\[
τ_n = \sum_{ℓ ∈ π_ℓ} τ_ℓ = \sum_{ℓ ∈ π_ℓ} d_ℓ + \frac{q_ℓ}{C_ℓ}.
\]

**Loss Probability and Queuing Disciplines.** Without active queuing discipline, loss occurs if the buffer of a link is full. Given such a simple drop-tail policy, the loss probability is given by the relative excess rate whenever the queue is full, and is 0 otherwise [58]. To facilitate analytical treatment, we refine previous models by a smooth approximation:

\[
p_ℓ(t) = \sigma(y_ℓ(t) - C_ℓ) \cdot \left(1 - \frac{C_ℓ}{y_ℓ} \right) \cdot \left(\frac{q_ℓ}{B_ℓ}\right)^L
\]

where \( L \gg 1 \) and \( \sigma(v) \) is a relatively sharp sigmoid function:

\[
\sigma(v) = \frac{1}{1 + e^{-K \cdot v}}
\]

with \( K \gg 1 \) controlling the sharpness of the increase at \( v = 0 \).

In contrast to drop-tail, the loss probability under the RED queuing discipline moves synchronously with the queue size. We approximate RED behavior as follows:

\[
p_ℓ = \frac{q_ℓ}{B_ℓ} \in [0, 1].
\]

Regarding the loss probability of paths, link-specific loss probabilities are assumed to be small enough such that the following approximations regarding loss hold:

\[
p_π_ℓ(t) = 1 - \prod_{ℓ ∈ π_ℓ} (1 - p_ℓ(t + d_ℓ^{i,ℓ})) \approx \sum_{ℓ ∈ π_ℓ} p_ℓ(t + d_ℓ^{i,ℓ}).
\]

**Congestion Window and Sending Rate.** For the window-based congestion-control algorithms Reno and CUBIC (cf. Appendix B), the sending rate of agent \( i \) is determined by the congestion-window size \( w_i \) and round-trip latency:

\[
x_ℓ = \frac{w_i}{τ_i}
\]

3 BBR FLUID MODEL

In this section, we introduce the first fluid model for BBR, both for BBRv1 [9] and BBRv2 [11]. Interestingly, the fluid-model techniques used for the loss-based CCAs cannot reflect essential BBR features, in particular its phases with different behavior. Hence, we construct our BBR model using new techniques, i.e., periodic probing pulses and mode variables (for simulating the BBR state machine). In the following, we first describe the behavior of BBR for both versions 1 and 2. Then, we present our fluid model for BBR by means of a basic fluid model, which can be concretized for each version.

3.1 Description of BBR

Fundamentally, BBR continuously performs measurements to estimate two core properties of the network path, namely the bottleneck bandwidth \( Bt18w \) and the minimal round-trip
time (RTT) \( \text{RTprop} \) (as given by the propagation delay). To estimate these properties, BBR constantly switches between two states, namely the \( \text{ProbeBW} \) state and the \( \text{ProbeRTT} \) state. While the \( \text{ProbeBW} \) state consumes most of flow lifetime and is considerably different across the two BBR versions, the \( \text{ProbeRTT} \) state is only infrequently and briefly entered and is mostly identical across both BBR versions.

**ProbeRTT state.** BBR enters the \( \text{ProbeRTT} \) state if no smaller round-trip time than the existing \( \text{RTprop} \) estimate is observed for 10 seconds. To discover the propagation delay, the \( \text{ProbeRTT} \) state tries to eliminate queuing delay by restricting the data in flight (inflight in BBR terminology) to a small volume during 200 ms. In BBRv1, this small volume has a fixed size of 4 segments; since this volume has been found to be too conservative, the \( \text{ProbeRTT} \) inflight limit in BBRv2 has been chosen to half the estimated bandwidth-delay product, i.e., half the product of \( Bt1Bw \) and \( \text{RTprop} \).

**ProbeBW state in BBRv1.** The \( \text{ProbeBW} \) state aims at measuring the bottleneck bandwidth of the network path, and includes a periodic probing strategy with the pacing rate as the primary control of the sending rate. In this probing strategy, each period consists of 8 phases with the duration of \( \text{RTprop} \). In one phase randomly chosen from the first 7 phases, BBRv1 sets its pacing rate to \( \frac{3}{4} \cdot Bt1Bw \) to find the capacity limit of the path. In the subsequent phase, BBRv1 decreases its pacing rate to \( \frac{3}{4} \cdot Bt1Bw \) to drain the queues potentially built up during the aggressive previous phase. In the other 6 phases of the period, BBRv1 paces at rate \( Bt1Bw \). At the period end, the maximum delivery rate from the period is then considered the new bottleneck bandwidth estimate and thus serves as a base pacing rate for the next period.

**BBRv1 congestion window.** While the focus on the pacing rate distinguishes BBRv1 from window-based CCAs like Reno or CUBIC, BBRv1 accidentally degenerates into a window-based CCA in certain circumstances. In particular, BBRv1 also maintains a congestion window, which amounts to twice the estimated BDP and was intended as a safeguard against ‘common network pathologies’ such as delayed ACKs [9]. Contrary to design intention, this inflight limit by the congestion window is the essential constraint on the sending rate of BBRv1 when competing with loss-based CCAs given large buffers [24, 57].

**ProbeBW state in BBRv2.** This unintentional relevance of the inflight limit in some circumstances, plus the unfairness towards loss-based CCAs in other situations, led Google to revise the \( \text{ProbeBW} \) mechanism for BBRv2. This revision mainly aimed at making BBR less aggressive, through increasing its sensitivity to loss and ECN signals (where we henceforth only consider loss for simplicity), less frequent probing, and a persistent coupling between inflight limits and the sending rate. To be precise, BBRv2 tries to obtain additional bandwidth only every few seconds, where the time between such probings is given by the minimum of 62 estimated RTTs (chosen for fairness reasons) and a random value between 2 and 3 seconds. In this probing, BBRv2 first paces at the rate given by \( Bt1Bw \) for one \( \text{RTprop} \), with the goal to achieve an inflight corresponding to the bandwidth-delay product. Then, BBRv2 sets its pacing rate to \( \frac{3}{4} \cdot Bt1Bw \) and increases the inflight until it reaches \( \frac{3}{4} \) of the estimated BDP or the loss rate exceeds 2%. At this point, the bottleneck-bandwidth estimate \( Bt1Bw \) is updated to the maximum delivery rate from the last two \( \text{ProbeBW} \) periods. Moreover, BBRv2 also records the maximum tenable inflight in state variable \( \text{inflight\_hi} \), which tracks the observed inflight, but is reduced by a multiplicative decrease \( \beta \) if the exponential-increase phase has been terminated by excessive loss. Afterwards, BBRv2 chooses a pacing rate of \( \frac{3}{4} \cdot Bt1Bw \) until the inflight is reduced to an arguably safe level, which corresponds to the minimum of the estimated BDP and 85% of the previously measured \( \text{inflight\_hi} \) (where the erased 15% are termed \( \text{headroom} \) in BBRv2). Once the inflight has been reduced to that level, BBRv2 enters into the cruising mode. In cruising mode, BBRv2 aims to keep its inflight on a safe level by introducing the additional inflight bound \( \text{inflight\_lo} \), which is activated if packet loss occurs: \( \text{inflight\_lo} \) starts from the congestion-window size at the moment of loss and is reduced by \( \beta \) upon packet loss. In contrast to \( \text{inflight\_hi} \), which serves as a long-term inflight bound, \( \text{inflight\_lo} \) serves as a short-term inflight bound and is therefore reset at the end of the bandwidth-probing period. In summary, at any point in time, the congestion-window size of a BBRv2 flow is the minimum of the general BBR congestion window of two BDP, the short-term bound \( \text{inflight\_lo} \) (if set) and the long-term bound \( \text{inflight\_hi} \) (discounted by \( \text{headroom} \) in cruising mode).

### 3.2 Basic fluid model for BBR

We rely on a skeleton fluid model that captures the common properties of BBRv1 and BBRv2. As mentioned in the previous section, the two versions of BBR are mostly similar regarding the estimation of the minimum RTT given by \( \text{RTprop} \), which we represent with variable \( t_i^\text{min}(t) \) for the \( \text{RTprop} \) estimate of agent \( i \) at time \( t \). The variable \( t_i^\text{min} \) is continuously adjusted downwards upon encountering smaller RTTs:

\[
t_i^\text{min} = -\Gamma(t_i^\text{min}(t) - t_i(t - d_i^\text{q}))
\]

where \( \Gamma(v) \) is a differentiable function approximating the ReLU function \( \max(0, v) \). Such a function can be constructed using the sigmoid function from Eq. (5):

\[
\Gamma(v) = \delta \cdot \sigma(v).
\]

To describe that BBR is in \( \text{ProbeRTT} \) state, we use a *discrete mode variable* \( m_i^\text{pr} \), which is 1 if BBR is in \( \text{ProbeRTT} \) state and 0 otherwise. In both BBR versions, the \( \text{ProbeRTT} \) mode is
switched on or off upon time-out of the ProbeRT timer \( t_i^{\text{prt}} \):

\[
\Delta m_i^{\text{prt}} = \sigma (t_i^{\text{prt}} - T_i^{\text{prt}}) \cdot \left( (1 - m_i^{\text{prt}}) - m_i^{\text{prt}} \right) 
\]

where \( T_i^{\text{prt}} \) is the time period between entries and exits of the ProbeRT state for agent \( i \). Note that Eq. (11) represents an update rule for simulations rather than a differential equation, as \( m_i^{\text{prt}} \) is discrete. The two time-related variables in Eq. (11) behave as follows:

\[
T_i^{\text{prt}} = m_i^{\text{prt}} \cdot 0.2 + (1 - m_i^{\text{prt}}) \cdot 10 
\]

\[
i_i^{\text{prt}} = 1 - \sigma (t_i^{\text{prt}} - T_i^{\text{prt}}) \cdot i_i^{\text{prt}} - \sigma (\tau_i^{\text{min}} - \tau_i (t - d_i)) \cdot i_i^{\text{prt}} 
\]

In ProbeRT mode, the sending rate is limited by a version-dependent inflight limit \( w_i^{\text{prt}} (t) \):

\[
x_i = m_i \cdot \frac{w_i^{\text{prt}}}{\tau_i} - (1 - m_i^{\text{prt}}) \cdot x_i^{\text{pbw}} 
\]

where the ProbeBW sending rate \( x_i^{\text{pbw}} \) follows the relevant constraint (congestion window \( w_i^{\text{pbw}} \) or pacing rate \( x_i^{\text{pcg}} \)):

\[
x_i^{\text{pbw}} = \min \left( \frac{w_i^{\text{pbw}}}{\tau_i}, x_i^{\text{pcg}} \right) 
\]

Similar to the ProbeRT state, we also introduce the two time-related variables \( T_i^{\text{pbw}} \) and \( t_i^{\text{pbw}} \) for the ProbeBW state, where \( T_i^{\text{pbw}} \) is the duration of a ProbeBW period and \( t_i^{\text{pbw}} \) is the time within the current period. While \( T_i^{\text{pbw}} \) is fully version-dependent, \( t_i^{\text{pbw}} \) grows with time and is reset to 0 when exceeding the period duration for both BBV versions:

\[
t_i^{\text{pbw}} = 1 - \sigma (t_i^{\text{pbw}} - T_i^{\text{pbw}}) \cdot t_i^{\text{pbw}} 
\]

In the ProbeBW state, the bottleneck-bandwidth estimation is based on measurements of the delivery rate \( x_i^{\text{dv}} \) (with link \( \ell \) being the bottleneck link of agent \( i \)):

\[
x_i^{\text{dv}} = \frac{x_i(t - d_i^\ell)}{y_i(t - d_i^\ell)} \cdot \left\{ \begin{array}{ll} C_i & \text{if } q_i (t - d_i^\ell) > 0 \\ y_i(t - d_i^\ell) & \text{otherwise} \end{array} \right. 
\]

We accommodate the recorded maximum delivery rate \( x_i^{\text{max}} (t) \) per ProbeBW period as follows:

\[
x_i^{\text{max}} = \Gamma (x_i - x_i^{\text{max}}) - \sigma (0.01 - t_i^{\text{pbw}}) \cdot x_i^{\text{max}} 
\]

where the second term provokes a reset of \( x_i^{\text{max}} \) in the first ten milliseconds of the period. The mechanism for adjusting the bottleneck-bandwidth estimate \( x_i^{\text{btl}} \) (corresponding to Bt1bw) to \( x_i^{\text{max}} \) is specific to each BBV version.

Finally, we choose the following natural formulation to model the inflight volume \( o_i (t) \):

\[
\dot{o}_i = x_i - x_i^{\text{dv}} 
\]

\[\text{Figure 2: Visualization of BBR fluid-model variables link capacity normalized to 100%, single flow}\]

### 3.3 BBRv1 Fluid Model

Given the basic BBR fluid-model framework, the biggest challenge in modelling BBRv1 is to model the randomized probing behavior with varying pacing rates. As described in §3.1, BBRv1 proceeds in bandwidth-probing periods that are 8 phases long, where each phase has a duration of \( \tau_i^{\text{min}} \), i.e., \( T_i^{\text{pbw}} = 8 \cdot \tau_i^{\text{min}} \). The bottleneck-bandwidth estimate \( x_i^{\text{btl}} \) is updated to the maximum delivery rate \( x_i^{\text{max}} \) at the end of the period, which we formalize by the following equation:

\[
x_i^{\text{btl}} = \sigma (t_i^{\text{pbw}} - T_i^{\text{pbw}} + 0.01) \cdot (x_i^{\text{max}} - x_i^{\text{btl}}) 
\]

In general, BBRv1 prescribes a pacing rate \( x_i^{\text{pcg}} \) equal to \( x_i^{\text{btl}} \) in each phase, but increases \( x_i^{\text{pcg}} \) to \( 3/4 \cdot x_i^{\text{btl}} \) in one randomly chosen phase and decreases it to \( 3/4 \cdot x_i^{\text{btl}} \) in the subsequent phase. To restrict a given behavior to a certain phase \( \phi \in \{0, ..., 7\} \), we introduce the following pulse function \( \Phi \), which is 1 if BBRv1 is in phase \( \phi \) and 0 otherwise:

\[
\Phi_i (t, \phi) = \sigma (t_i^{\text{pbw}} (t) - \phi \cdot \tau_i^{\text{min}}) \cdot \sigma (\phi + 1) \cdot \tau_i^{\text{min}} - t_i^{\text{pbw}} 
\]

This pulse function allows to model the pacing behavior of an agent \( i \) that employs the augmented pacing rate in phase \( \phi_i \):

\[
x_i^{\text{pcg}} = x_i^{\text{btl}} \cdot \left( 1 + \frac{1}{8} \cdot \Phi_i (t, \phi_i) - \frac{1}{8} \cdot \Phi_i (t, \phi_i + 1) \right) 
\]

In the implementation of BBRv1, the phase \( \phi_i \) is randomly chosen from \( \{0, ..., 6\} \) every time BBRv1 switches from ProbeRT state back to ProbeBW state. Since such randomness is incompatible with the determinism of fluid models, we mimic the randomness of \( \phi_i \) by choosing it as \( i \mod 6 \), where we assume the agent identifier \( i \) to be a natural number. This agent-dependent choice of \( \phi_i \) desynchronizes the pacing-rate variation of agents \( i \) on paths with equal RTT, which is the central goal of the randomization, without sacrificing the determinism of the fluid model. The interplay of BBRv1 variables in pacing-based mode is visualized in Fig. 2a.

The basic BBR fluid-model allows a straightforward integration of the state-dependent inflight limits of BBRv1:

\[
w_i^{\text{btl}} = 4 \quad w_i^{\text{pbw}} = 2 \cdot w_i = 2 \cdot x_i^{\text{btl}} \cdot \tau_i^{\text{min}} 
\]

where \( w_i \) denotes the BDP estimated by agent \( i \).
3.4 BBRv2 Fluid Model

BBRv2 mostly differs from BBRv1 with regard to the structure of the bandwidth-probing phase in several ways.

First, a bandwidth-probing period is considerably longer than in BBRv1. The duration of bandwidth-probing periods in BBRv2 is given by the minimum of 62 estimated RTTs and a random value between 2 and 3 seconds. This randomness in BBRv2 poses a similar challenge in modeling as the randomness in BBRv1, such that we again use an approach based on the agent identifier to achieve the central goal of agent desynchronization without sacrificing determinism:

\[ T_i^{bw} = \min (65 \cdot \tau_i, 2 + \frac{i}{N}) \]  

Second, the behavior in the bandwidth-probing phases of BBRv2 differs from BBRv1. To model the BBRv2 phases, we introduce two additional mode variables, namely \(m_i^{dwn}(t)\), which indicates whether agent \(i\) is attempting to reduce its inflight at time \(t\), and \(m_i^{crs}(t)\), which indicates whether agent \(i\) is cruising at time \(t\). The mode variable \(m_i^{dwn}\) affects the pacing rate \(x_i^{bw}\) as follows:

\[ x_i^{bw} = x_i^{dwn} \left(1 + \frac{1}{2} \sigma (t_i^{bw} - \tau_i^{min}) - (1 - m_i^{dwn}) - \frac{1}{2} \cdot m_i^{dwn} \right) \]  

where \(m_i^{dwn}\) increases the pacing rate to \(\frac{5}{4} \cdot x_i^{dwn}\) if \(m_i^{dwn} = 0\) (and one RTT has passed in the bandwidth-probing period), and decreases the pacing rate to \(\frac{3}{4} \cdot x_i^{dwn}\) if \(m_i^{dwn} = 1\).

Third, while we modeled phase transitions in BBRv1 as purely dependent on time \(t_i^{bw}\), the phase transitions in BBRv1 are triggered by probing observations. In particular, the inflight-reducing mode \(m_i^{dwn}\) is activated if the inflight \(v_i\) exceeds \(\frac{5}{4} \cdot \overline{w}_i\) or loss \(p_{\pi i}\) exceeds 2%, and is disabled once the reduced pacing rate has managed to reduce the inflight \(v_i\) to the draining target \(w_i^{dwn} = \min(\overline{w}_i, 0.85 \cdot w_i^{hi})\):

\[ \Delta m_i^{dwn} = (1 - m_i^{crs}) \cdot (1 - m_i^{dwn}) \cdot \sigma (t_i^{bw} - \tau_i^{min}) \cdot \min \left(\sigma(v_i - \frac{5}{4} \cdot \overline{w}_i) + \sigma(p_{\pi i} - 0.02), 1 \right) \]

\[ -m_i^{dwn} \cdot \sigma (w_i^{dwn} - v_i) \]  

where \(w_i^{hi}\) is the variable accommodating the long-term bound \(inflight_{hi}\) in our model. Moreover, the disabling of \(m_i^{dwn}\) automatically leads to the activation of \(m_i^{crs}\), which is then disabled again when a new bandwidth-probing period starts:

\[ \Delta m_i^{crs} = -\Delta m_i^{dwn} - \sigma (t_i^{bw} - t_i^{bw}) \cdot m_i^{crs} \]  

The fourth difference concerns the adjustment of the bottleneck-bandwidth estimate \(x_i^{dwn}\). In BBRv2, \(x_i^{dwn}\) is adjusted to the maximum delivery rate from the last two probing periods when the inflight-growing phase has stopped:

\[ x_i^{dwn} = m_i^{dwn} \cdot \left( \max \left( x_i^{max}, x_i^{max} - \overline{T}^{bw} \right) - x_i^{dwn} \right) \]  

Fifth, BBRv2 operates with another two additional state variables, namely \(inflight_{hi}\) and \(inflight_{lo}\), which we accommodate in our fluid model with \(w_i^{hi}\) and \(w_i^{lo}\), respectively. The upper inflight bound \(w_i^{hi}\) is exponentially adjusted upwards when it represents the relevant constraint on the sending rate \((v_i = w_i^{hi})\) during the aggressive probing phase and no excessive loss occurs. In contrast, \(inflight_{hi}\) is reduced by a multiplicative decrease of 30% if encountering loss exceeding 2%. To be precise, the BBRv2 implementation applies this multiplicative decrease at most once per bandwidth-probing period. We approximate this behavior with a reduced multiplicative decrease in presence of excessive loss:

\[ w_i^{hi} = (1 - m_i^{crs}) \cdot \sigma (t_i^{bw} - \tau_i^{min}) \cdot \sigma (v_i - w_i^{hi}) \cdot 2^{t_i^{bw} - \tau_i^{min}} \cdot \sigma (p_{\pi i} - 0.02) \cdot \frac{0.3}{\tau_i^{min}} \cdot w_i^{hi} \]  

Outside of cruising mode, the lower inflight bound \(w_i^{lo}\) is unset (which we represent with an assimilation to \(w_i\)). In cruising mode, \(w_i^{lo}\) is also decreased by 30% per RTT upon encountering loss:

\[ w_i^{lo} = (1 - m_i^{crs}) \cdot (w_i - w_i^{lo}) - m_i^{crs} \cdot \sigma (p_{\pi i} - 0.02) \cdot \frac{0.3}{\tau_i^{min}} \cdot w_i^{lo} \]  

In summary, the congestion-window size in ProbesBW state is given as follows in BBRv2:

\[ w_i^{pbt} = \min \left(2 \cdot \overline{w}_i, (1 - m_i^{crs}) \cdot w_i^{hi} + m_i^{crs} \cdot w_i^{lo} \right) \]  

A final difference between BBRv1 and BBRv2 concerns the congestion-window size in ProbesRT mode. Instead of using a fixed congestion-window size of 4 segments, BBRv2 cuts the congestion window to half the estimated BDP in this mode:

\[ w_i^{pbt} = \frac{\overline{w}_i}{2} \]  

The interplay of the variables in the BBRv2 fluid model is visualized by means of an example in Fig. 2b.

4 EXPERIMENTAL VALIDATION

In this section, we experimentally validate our BBR fluid model, building on the network emulator mininet [34].

4.1 Validation Setup

4.1.1 Model-Based Computations. The fluid models have been implemented using NumPy [42]. We will make this implementation available. Differential equations are solved with the method of steps [17, §1.1.2] with a step size of 10 µs.

4.1.2 Experiments. To compare the model output with implementation behavior, we perform experiments using the network emulator mininet [34]. In mininet, we use OVS to emulate switches [19]. The emulated hosts send traffic by
The sending rate is given in percent of the bottleneck-link (as recommended by mininet [33]) and with a propagation dictates that the sending rate of the loss-sensitive BBRv2 barely is insensitive to loss. Finally, the fluid model correctly pre-
considerably smaller loss (barely visible) than BBRv1, which 
elicits correctly capture that the loss-sensitive BBRv2 lead to 
garding the rate patterns over time. In addition, the fluid mod-
tween the model and the experiments.

4.2 Validation of Trace Results

Using the validation set-up described in §4.1, we first veri-
ied traces as predicted by the fluid models regarding their similarity with traces from experimental measurements. The concrete network setting in this trace validation included a single sender, a bottleneck link \( \ell \) with a rate of \( C_\ell = 100 \text{ Mbps} \) (as recommended by mininet [33]) and with a propagation delay of \( d_\ell = 10 \text{ ms} \), a non-bottleneck link \( \ell_i \) with a delay of \( d_i = 5.6 \text{ ms} \), and a switch buffer of 1 BDP.

Figs. 4 and 5 visualize the comparison of the thus obtained traces for the BBRv1 and BBRv2, respectively, where each CCA was tested under both a drop-tail and a RED queuing discipline (A validation of Reno and CUBIC can be found in Appendix B). All measurements have been normalized: The sending rate is given in percent of the bottleneck-link rate, the queue in percent of buffer volume, loss in percent of traffic volume, the RTT as relative excess delay, and the congestion window in percent of the path BDP. The comparisons highlight both important commonalities and differences between the model and the experiments.

**Similarities.** The fluid models are highly predictive regarding the rate patterns over time. In addition, the fluid models correctly capture that the loss-sensitive BBRv2 lead to considerably smaller loss (barely visible) than BBRv1, which is insensitive to loss. Finally, the fluid model correctly predicts that the sending rate of the loss-sensitive BBRv2 barely exceeds the bottleneck rate under RED, while RED has no impact on the loss-insensitive BBRv1. As a result, the smaller buffer usage of BBRv1 under RED is also reflected in the model, although the difference between RED and drop-tail is more pronounced in the model.

**Difference: RED idealization.** The difference above is due to the idealization of RED: In the model, the queue size affects the loss probability instantly; in reality, RED relies on outdated and averaged measurements of the queue size, causing some lag between queue build-ups and loss surges.

**Difference: ProbeRTT state in BBRv2.** In the model, the BBRv2 flow regularly manages to drain the queue, therefore discovers the propagation delay early, and cannot detect a lower RTT afterwards; hence, it enters the \( \text{ProbeRTT} \) state every 10 seconds. Thanks to the low queue build-up, the BBRv2 flow in the RED experiment also discovers the propagation delay. However, the BBRv2 flow in the drop-tail experiment can regularly decrease its RTprop estimate because of the small, but persistent and fluctuating queue, and therefore never enters the \( \text{ProbeRTT} \) state.

In summary, the fluid models capture the differences among CCAs and queuing disciplines with high accuracy, especially qualitatively and to a lesser degree also quantitatively.

4.3 Validation of Aggregate Results

The trace validations in the previous subsection indicate that the presented CCA fluid models yield reasonable predictions for single senders. The more important question, however, is whether these fluid models can acceptably predict network-performance metrics given interacting senders. To test the fluid models in this metric-oriented aspect, we compare aggregate results from model computations and experiments for a wide variety of network parameters, in particular with respect to Jain fairness (Fig. 6), packet loss (Fig. 7), buffer occupancy (Fig. 8), bottleneck-link utilization (Fig. 9) and jitter, i.e., packet-delay variation (Fig. 10). All metrics were obtained from the aggregation of 5-second traces, where the experiment results are averaged over 3 runs. In contrast, fluid models are deterministic and do not require averaging. The network setting was based on the topology in Fig. 3, \( N = 10 \) senders, a bottleneck-link rate of \( C_\ell = 100 \text{ Mbps} \), a bottleneck-link propagation delay \( d_\ell = 10 \text{ ms} \) and total RTTs randomly selected between 30 and 40ms. For heterogeneous CCAs, each CCA was employed by \( N/2 = 5 \) senders. To strengthen our validation, we conduct the same analysis for shorter delays, which confirms our results (cf. Appendix C).

As in the trace validation, the comparisons reveal that the model predictions and the experiment results have striking similarities as well as notable differences. In the following, we will discuss each metric separately.
4.3.1 Fairness. Regarding fairness (cf. Fig. 6), we first observe that the least fairness arises when a loss-sensitive CCA (Reno, CUBIC or BBRv2) is in competition with BBRv1 in shallow buffers, which has already been well documented in previous research [50, 57]. This unfairness is the result of the loss insensitivity of BBRv1, which maintains its rate despite loss while loss-sensitive CCAs practically stop sending in reaction to the loss caused by BBRv1. Starting at buffer sizes from 4 BDP, however, the fairness in these settings increases for two reasons. First, these large buffers reduce the occurrence of loss, which prevents the back-off of loss-sensitive CCAs. Second, in large buffers, the inflight limit given by the congestion window of BBRv1 restricts the sending rate of BBRv1 and allows competing flows to obtain a higher share of bandwidth than in shallow buffers. Given a RED queue, however, the fairness of BBRv1 towards loss-sensitive CCAs is consistently low because RED (1) increases loss and (2) restricts the buffer build-up such that the inflight of the BBRv1 flows is substantially below their inflight limit.

The fairness issues of BBRv1 appear to have been largely resolved with BBRv2, as both the fluid model and the experiment results show. However, BBRv2 is still unfair towards loss-based CCAs in RED buffers, where the higher loss sensitivity of loss-based CCAs is revealed.

One substantial difference between the fluid-model predictions and the experiment results is the decreasing fairness of BBRv1 in deep drop-tail buffers, which only appears in the fluid model. The fluid model reveals the RTT unfairness of BBRv1, which has indeed been experimentally confirmed [50, 57], although for higher RTT differences than used in our network setting. This RTT unfairness stems from the inflight limit of BBRv1, which becomes relevant in deep buffers: Since flows with a lower RTT estimate a lower BDP and hence maintain a smaller congestion window, lower-RTT flows are more severely restricted by the inflight limit. Moreover, these differences are increased over time because flows with a higher rate also measure a higher maximum delivery rate, which affects the bottleneck-bandwidth estimate in the estimated BDP. This effect can be analytically revealed by means of Eq. (17) for the delivery rate $\mathbf{d}$: Given an existing queue $q_\ell > 0$, any reduction in total arrival rate $y_\ell$ (e.g., because of one sender $j$ decreasing its pacing rate $x_{\text{pcg}}$ to $3/4 \cdot x_{\text{pcg}}$) results in a larger absolute $x_{\text{d}}$ for senders with a high rate $x$. This representation of the delivery rate idealizes the noisy relationship between sending rates and delivery rates in real-world buffers. Hence, the described effect only appears for larger RTT differences in experiments, where the noise in delivery rates does not eliminate the difference in sending rates between flows with different RTTs.

In conclusion, the fluid models correctly predict fairness effects from a qualitative perspective, i.e., they rank CCA settings correctly according to their fairness, and approximately also from a quantitative perspective. Interestingly, the fluid model also predicts RTT unfairness among BBRv1 flows in deep drop-tail buffers, which does not appear in the corresponding experiments. However, this RTT unfairness is a real issue in more extreme settings than tested in this validation, i.e., for higher RTT differences between the senders. Since the role of a fluid model is to reveal problematic CCA features for further investigation, we argue that the exaggeration of an existing problem barely weakens the methodological value of the BBR fluid model.
4.3.2 Loss. Fig. 7 suggests that fluid models are highly suitable to predict loss rates for different CCAs, both in homogeneous settings and heterogeneous settings and both qualitatively and quantitatively. Our model correctly predicts (1) that the loss rate of loss-sensitive CCAs (Reno, CUBIC, BBRv2 and combinations thereof) in drop-tail buffers is below 1% and goes to 0% for increasing buffer size, (2) that BBRv1 leads to considerable loss of at most 20%, where the loss rate is indirectly proportional to the buffer size for drop-tail queuing, and (3) that a RED queueing discipline keeps loss rates roughly consistent across buffer sizes.

One obvious prediction error of the fluid model is the underestimation of loss rates for loss-sensitive CCAs given RED in Fig. 7d. This underestimation stems again from an idealization of the RED queue in the model, which determines the loss rate based on the current queue length. In contrast, real RED tracks the queue length with a moving average and hence reacts to queue build-up with delay. Since the queue has more time to accumulate until stabilization by RED, the queue length is higher than given an instantaneously reacting RED algorithm (cf. also Fig. 8b). Moreover, since the RED dropping probability is proportional to the queue size, the loss rate given delayed RED is slightly higher than for instantaneous RED. However, since this underestimation only amounts to 0.5 percentage points, we still consider our fluid model highly predictive with respect to loss.

4.3.3 Queuing. Fig. 8 shows the average queue size as a share of buffer capacity. Notably, the fluid model captures the effect that the traditional loss-based CCAs Reno and CUBIC cause bufferbloat, i.e., lead to consistently high buffer utilization. Also combined settings of BBRv2 with loss-based CCAs include high buffer utilization, where the generally lower buffer usage of BBRv2 in a homogeneous setting demonstrates that BBRv2 is not responsible for the bufferbloat.

Interestingly, BBRv1 leads to even more intense buffer usage than loss-sensitive CCAs, whether in homogeneous settings or in heterogeneous settings with loss-sensitive CCAs. Under drop-tail (Fig. 8a), BBRv1 uses most of the buffer independent of buffer size, where the relative buffer usage is only moderately reduced in large buffers. This effect is surprising, as a major design goal of BBR is exactly to avoid the bufferbloat caused by traditional loss-based CCAs [9].
The validation analysis reveals another unexpected phenomenon, which concerns the buffer utilization of BBRv2 in homogeneous settings given a drop-tail queuing discipline. In particular, BBRv2 leads to constant absolute buffer usage for buffer sizes up to 4 BDP, which is visible as decreasing relative buffer usage in Fig. 8a. In these scenarios, the adjustments to BBR appear to have resolved the issue of bufferbloat in BBRv1. In large buffers, however, the buffer utilization increases again with buffer size. Through trace inspection, we found that this phenomenon is caused by initial measurements of inflight_hi during the start-up phase of BBRv2. Given large buffers, this initial inflight_hi bound may be set too high or not set at all. Moreover, inflight_lo may never be set either, because large buffers prevent loss, which would activate inflight_lo. In absence of stringent bounds given by inflight_hi and inflight_lo, BBRv2 falls back on the standard BBR congestion-window size of 2 estimated BDP (cf. Eq. (31)). In comparison with the empirically found inflight_hi and inflight_lo, this congestion-window size is a loose bound that allows higher sending rates of BBRv2 and thus causes more intense buffering. To the best of our knowledge, this behavior of BBRv2 has not been publicly documented so far. While our BBRv2 fluid model does not model the start-up phase which causes this issue, the same effect can be observed in the model when choosing the initial condition of the differential equation for \( w_i \) (cf. Eq. (29)) dependent on the buffer size. Therefore, we note that fluid models have to be evaluated under a variety of initial conditions to reveal design issues.

4.3.4 Utilization. The fluid model captures three important aspects of link utilization (cf. Fig. 9). First, the fluid model correctly predicts that BBRv1 (or combinations with BBRv1) lead to full utilization of the bottleneck link, both under drop-tail and RED. This high utilization by BBR is unsurprising given the aggressiveness of the CCA, which also manifests in high loss (Fig. 7) and intense queuing (Fig. 8). Second, the fluid model mirrors the increasing link utilization by loss-sensitive CCAs for increasing buffer size under drop-tail: Loss-based CCAs grow their rate while the buffer is filling up and cut it by a constant factor when the buffer is full, so larger buffers imply higher rates and thus higher utilization. Third, the fluid model reflects that BBRv2 yields the lowest utilization given drop-tail among all CCAs, although the wasted capacity only amounts to 3% at most. This incomplete utilization stems from the ProbeRTT state of BBRv2, in which the inflight is reduced to half the estimated BDP. Under RED, BBRv2 does not enter the ProbeRTT phase for 10 senders; neither does BBRv1 for both queuing disciplines.

The only major prediction error of the model is the under-estimation of utilization by the BBRv2/Reno combination in shallow RED buffers. This result indicates an exaggeration of the loss reaction in the Reno fluid model by Low et al. [37].

4.3.5 Jitter. Jitter corresponds to the mean delay difference between consecutive packets. The experimental jitter results in Fig. 10 are calculated in this manner. As the fluid model misses a notion of packets, we compute the jitter for fluid-model traces by sampling the RTT at a virtual packet rate, i.e., every \( g \cdot N/C_l \) seconds, where \( g \) is a given packet size.

Howevet, Fig. 10 makes clear, also this makeshift calculation unsurprisingly fails to predict jitter: Fluid models intentionally abstract from small-scale fluctuations and describe only the macroscopic tendency of network indicators with smooth curves. Nonetheless, fluid models could be combined with packet-level models aimed at modeling jitter [14, 15]; we leave this challenge for future research.

5 THEORETICAL ANALYSIS
In this section, we analyze the BBR fluid models to characterize the stability of these CCAs. The proofs of all theorems are presented in Appendix D.

5.1 BBRv1 Stability Analysis
While the fluid model in §3.3 is suitable for simulation, we have to simplify it to obtain a high-level model suitable for analysis (§5.1.1). In §5.1.2, we investigate the existence, form and stability of BBRv1 equilibria.

5.1.1 Model Reduction. The first simplification step is given by disregarding the ProbeRTT state, which lets agent discover their propagation delay and generally achieves this goal.
Hence, we assume that \( t_i^{\min} = \frac{\sum_{t \in \pi} d_t}{d_i} \). Since this minimum-RTT measurement is present in the congestion-window size \( w_i^{\text{pbw}} = 2t_i^{\min}A_i^{\text{blt}} \), also \( w_i^{\text{pbw}} \) can be simplified to \( 2d_iA_i^{\text{blt}} \). Apart from affecting the congestion-window size, the probeRTT state has no lasting effects and can hence be omitted for the purpose of stability analysis.

The second step involves understanding the evolution of the maximum measurement \( x_i^{\max} \). This maximum measurement is the maximum delivery rate \( x_i^{\text{dlv}} \) over a period, which in turn depends on the sending rates of all agents and the queue length at the bottleneck link (cf. Eq. (17)). The sending rate of an agent \( i \) is either \( \min(w_i^{\text{pbw}}/\tau_i, \Delta_i^{\text{blt}}) \) (if agent \( i \) is probing), \( \min(w_i^{\text{pbw}}/\tau_i, \frac{5}{4}\Delta_i^{\text{blt}}) \) (if draining), or \( \min(w_i^{\text{pbw}}/\tau_i, \Delta_i^{\text{blt}}) \) (otherwise). Hence, the maximum delivery rate depends on the concurrent behavior of the other agents on the bottleneck link, which may be probing or draining during the measurements of agent \( i \). For many agents, the probing and draining agents can be expected to offset each other such that the total background-traffic volume is similar as \( x_i^{\text{blt}} = \min(w_i^{\text{pbw}}/\tau_i, \Delta_i^{\text{blt}}) \) for all agents \( j \in \mathcal{U}_t, j \neq i \). Given this background traffic, an agent \( i \) measures the maximum delivery rate in the probing phase. Hence, the maximum measurement is determined as follows:

\[
x_i^{\max} = \begin{cases} 
\min(\frac{<4}{4}, \Delta_i) \cdot x_i^{\text{blt}} - C_i & \text{if } q_{t_i} > 0 \\
\min(\frac{<4}{4}, \Delta_i) \cdot x_i^{\text{blt}} & \text{otherwise}
\end{cases}
\]

(33)

where \( \Delta_i = 2d_i/(d_i + \sum_{t \in \pi} q_{t}/C_t) \), and \( t_i \) is the bottleneck link for agent \( i \).

The third and final simplification step concerns the adaptation of the bottleneck-bandwidth estimate \( x_i^{\text{blt}} \). Considered over a long duration, this regular update can be approximated by a continuous assimilation:

\[
x_i^{\text{blt}} = x_i^{\max} - x_i^{\text{blt}}
\]

(34)

5.1.2 Stability Analysis. To characterize stability of a CCA, we first need to identify its equilibria, i.e., configurations from which the fluid-model dynamics cannot depart. In the case of BBRv1, the network state may change if the maximum measurement \( x_i^{\max} \) by some agent \( i \) differs from the bottleneck-bandwidth estimate \( x_i^{\text{blt}} \), or if the queue length \( q_t \) of some link \( t \) grows; both events may lead to subsequent rate changes. To formalize this condition, we henceforth consider \( N \) senders which share a single bottleneck link \( t^* \). Moreover, we first assume that buffer capacities do not constrain the dynamics, and modify this assumption later.

**Definition 1.** \( N \) BBR\(v1 \) senders sharing a bottleneck link \( t^* \) are in equilibrium if and only if \( \{x_i^{\text{blt}}\}_{i \in \mathcal{U}_{t^*}} \) and \( q_{t^*} \) satisfy:

\[
\sum_{i \in \mathcal{U}_{t^*}} \min(1, \Delta_i) \cdot x_i^{\text{blt}} = C_{t^*} \quad \forall i \in \mathcal{U}_{t^*}, x_i^{\text{blt}} = x_i^{\max}
\]

(35)

The first condition keeps the aggregate rate \( y_{t^*} \) at line rate \( C_{t^*} \) and hence ensures a static queue length. The remaining constraints rule out rate adaptations.

**Theorem 1.** \( N \) BBR\(v1 \) senders sharing a bottleneck link \( t^* \) are in equilibrium if and only if propagation delay equals queuing delay for every sender, i.e.,

\[
\forall i \in \mathcal{U}_{t^*}, d_i = \sum_{t \in \pi} \frac{q_t}{C_t}
\]
Interestingly, Theorem 1 suggests that the equilibria of BBRv1 in single-bottleneck scenarios (with non-limiting buffers) can be arbitrarily unfair as long as \( \sum_{i \in U_r} x_i^{btl} = C_r \). Furthermore, we note that the BBRv1 equilibrium requires equal path propagation delay \( d \) for all senders if all senders only encounter a non-empty queue at the bottleneck link \( \ell^* \). For our stability analysis, we focus on that case, i.e., a scenario where the queue lengths on all involved links except the bottleneck link \( \ell^* \) are zero, which is a scenario frequently investigated in the literature [1, 25, 62]. In this case, we can prove asymptotic stability of BBRv1 with the indirect Lyapunov method, meaning that initial configurations exist for which the BBRv1 dynamics converge to the equilibrium.

**Theorem 2.** In a single-bottleneck network with a queue exclusively at the bottleneck, the BBRv1 equilibrium from Theorem 1 is asymptotically stable.

As the proof of Theorem 2 in Appendix D.2 makes clear, the BBRv1 equilibrium from Theorem 1 is only viable if the bottleneck-link buffer capacity \( B_r \) permits the equilibrium queue length \( q_{r} = dC_r \). Intuitively, the equilibrium is valid for a bottleneck buffer that is large enough for the congestion-window constraint \( \Delta_i \) to have an impact. To analytically investigate the shallow-buffer case where the congestion-window limit is not effective, we assume that the bottleneck queue length \( q_r \) is restricted by the buffer size \( B_r \) such that the congestion-window limit has no effect for any agent \( i \), i.e., \( \Delta_i \geq \frac{5}{4} \) for all \( i \in U_r \) (cf. Eq. (33)). With this assumption, we find a different equilibrium for BBRv1:

**Theorem 3.** \( N \) BBRv1 senders sharing a bottleneck link \( \ell^* \) that has a shallow buffer (i.e. \( \Delta_i > \frac{5}{4} \) for all \( i \in U_r \)) are in equilibrium if and only if each agent \( i \) has the following bottleneck-bandwidth estimate \( x_i^{btl} \):

\[
x_i^{btl} = \frac{5C_r}{4N + 1}.
\]

This equilibrium is perfectly fair and asymptotically stable.

Theorem 3 thus implies that without an effective congestion-window limit, the aggregate rate \( y_r \) in equilibrium consistently exceeds the link capacity \( C_r \), except for \( N = 1 \). As a result, multiple BBRv1 senders fill the shallow bottleneck-link buffer, eventually incurring a loss rate equal to the excess sending rate (20% for \( N \to \infty \)). While this consistent packet loss does not reduce the rate of loss-insensitive BBRv1 senders, the loss is fatal for loss-based CCAs on the same bottleneck link, which produces high inter-CCA unfairness. Among each other, BBRv1 flows forcibly converge to perfect fairness in shallow buffers, where such fairness is only possible, but not required by the BBRv1 equilibria for deep buffers (cf. Theorem 1).

### 5.2 BBRv2 Stability Analysis

Similar to the preceding section, this section presents a condensed version of the BBRv2 fluid model from §3.4, which is then used for stability analysis.

#### 5.2.1 Model Reduction

Thanks to the shared foundation of BBRv1 and BBRv2, our reduced fluid model for BBRv2 largely matches the reduced model for BBRv1 (cf. §5.1.1) such that we only discuss different simplification steps.

In particular, the specific probing process of BBRv2 affects the maximum measurement \( x_i^{max} \). This probing process is centered around a traffic pulse, which raises the pacing rate to \( \frac{5}{4} \cdot x_i^{btl} \) and the inflight volume to \( \frac{5}{4} \cdot \bar{w}_i \), expect the loss exceed 2%. Since we limit our analysis to networks with buffers large enough to prevent loss, the traffic-pulse rate is:

\[
x_i^{pls} = \frac{5}{4} \cdot \min(1, \delta_i) \cdot x_i^{btl}, \quad \delta_i = \frac{d_i}{d_i + \sum_{\ell \in \pi_i} q_{\ell}/C_{\ell}}
\]

In addition to this pulse rate, the background traffic co-determines the maximum delivery rate \( x_i^{\text{max}} \). This background traffic can be assumed to consist of flows in cruising mode, in which any BBRv2 flow spends the vast majority of its lifetime. In cruising mode, a BBRv2 agent \( i \) sets its pacing rate to \( x_i^{btl} \) and keeps its inflight volume at the minimum of the estimated BDP \( \bar{w}_i \) and 85% of the upper inflight bound \( \text{inflight}_{hi} (w_i^{hi}) \). Since we can exclude packet loss, \( w_i^{hi} \) corresponds to the maximum measured inflight from the probing pulse, which is \( \frac{5}{4} \cdot \bar{w}_i \). Since the estimated BDP \( \bar{w}_i \) is consistently smaller than 0.85 * \( \frac{5}{4} \cdot \bar{w}_i \), the sending rate in cruising mode is:

\[
x_i = \min(1, \delta_i) \cdot x_i^{btl}
\]

Based on the sending rates of pulses and the cruising mode, the evolution of the maximum delivery rate \( x_i^{\text{max}} \) is approximated as follows (with \( \ell_i \) as the bottleneck link):

\[
x_i^{\text{max}} = \begin{cases} 
\frac{\frac{5}{4} \cdot \min(1, \delta_i) \cdot x_i^{btl} \cdot C_{\ell_i}}{d_i + \sum_{\ell \in \pi_i} \min(1, \delta_i) \cdot x_i^{btl}} & \text{if } q_{\ell_i} > 0 \\
\frac{\frac{5}{4} \cdot \min(1, \delta_i) \cdot x_i^{btl}}{4N + 1} & \text{otherwise}
\end{cases}
\]

#### 5.2.2 Stability Analysis

For BBRv2, the equilibrium conditions match the equilibrium conditions for BBRv1 (cf. Definition 1), with \( \Delta_i \) substituted by \( \delta_i = \Delta_i/2 \). However, the modified adaptation rule for \( x_i^{btl} \) induces a different equilibrium for BBRv2:

**Theorem 4.** \( N \) BBRv2 senders sharing a bottleneck link \( \ell^* \) are in a perfectly fair equilibrium if propagation delay and queuing delay for each agent have the following relation:

\[
\forall i \in U_r, \quad \frac{N - 1}{4N + 1} \cdot d_i = \sum_{\ell \in \pi_i} \frac{q_{\ell}}{C_{\ell}}
\]
Importantly, the above equilibrium is not necessarily the only equilibrium for BBRv2, which may thus induce unfair equilibria like BBRv1. Nevertheless, the above BBRv2 equilibrium has an inter-depency with the rate distribution; no BBRv1 equilibrium involves such an inherent dependency.

Similar to the BBRv1 equilibrium, however, the above equilibrium implies equal path propagation delay \( d \) for all senders if all senders only encounter a non-empty queue at the bottleneck link \( t^* \). For our stability analysis, we thus again focus on a scenario where the queue lengths on all involved links except the bottleneck link \( t^* \) are zero:

**Theorem 5.** In a single-bottleneck network with a queue exclusively at the bottleneck, the BBRv2 equilibrium from Theorem 4 is asymptotically stable.

Importantly, if a network has a queue exclusively at the bottleneck link, Theorem 4 implies equilibrium queue length \( q_{t^*} = \frac{N-1}{4N} dC_t \). In comparison with BBRv1, BBRv2 thus reduces buffer utilization by at least 75% (for \( N \to \infty \)), assuming the buffer is large enough to accommodate the BBRv1 equilibrium queue.

### 6 INSIGHTS AND DISCUSSION

In this section, we summarize the most interesting insights from our experimental validation (§4), and our theoretical analysis (§5). These insights reflect properties of CCAs (§6.1) and properties of the fluid-model methodology (§6.2).

#### 6.1 Insights into CCA Performance

One of the most consistent findings in the previous sections relates to the packet loss caused by different types of CCAs:

**Insight 1.** *Loss Rates of CCAs.* Both under drop-tail and RED queuing disciplines, BBRv1 causes considerable loss of up to 20% of traffic, while the loss-sensitive CCAs Reno, CUBIC, and BBRv2 cause loss rates of around 1%.

While such a difference between BBRv1 and loss-based CCAs is not surprising given different loss sensitivity, the large extent of the loss caused by BBRv1 is unexpected. Importantly, while the loss insensitivity of BBRv1 does not lead to throughput reductions of BBRv1, the high packet loss will still lead to unsatisfactory application performance. Moreover, in competition with loss-sensitive CCAs, the loss insensitivity of BBRv1 poses a fairness concern, which is also reflected in previous work [50, 57]:

**Insight 2.** *BBRv1’s Unfairness Towards Loss-Based CCAs.* BBRv1 is highly unfair towards loss-sensitive CCAs, leading to near starvation of loss-based flows in shallow buffers (given a drop-tail queuing discipline) or buffers of any size (given a RED queuing discipline). In large drop-tail buffers, the congestion window of BBRv1 becomes effective, leading to improvements in fairness towards loss-sensitive CCAs.

The aggressiveness of BBRv1, causing high loss and unfairness, has two other effects:

**Insight 3.** *Utilization and Buffer Usage of BBRv1.* In all investigated settings, BBRv1 (also in combination with other CCAs) achieves full link utilization, but also significant bufferbloat independent of the queuing discipline.

Most of these insights gained from our fluid model have already been identified by previous, experiment-based analyses. In response to documentations of these issues, Google has begun to develop BBRv2, which can be characterized as follows:

**Insight 4.** *Performance of BBRv2.* BBRv2 mostly achieves the redesign goals of reduced buffer usage, avoiding excessive loss, and preserving fairness to loss-based CCAs.

However, we have identified two settings in which BBRv2 does not achieve its design goal:

**Insight 5.** *BBRv2 in Large Drop-Tail Buffers.* In drop-tail buffers with a size exceeding five BDP, BBRv2 causes higher buffer utilization than for smaller buffers, caused by distortions in an initial in-flight size estimate in the start-up phase.

**Insight 6.** *BBRv2 in RED Buffers of High-Capacity Links.* When competing with loss-based CCAs (Reno and CUBIC) on a high-capacity link under a RED queuing discipline, BBRv2 is unfair towards the loss-based CCAs. The reason for this unfairness is that on high-capacity links, the loss sensitivity of loss-based CCAs is markedly higher than the loss sensitivity of BBRv2.

#### 6.2 Insights into Fluid Models

The preceding sections not only yield valuable insights into the performance characteristics of BBR, but also illustrates the strengths and limitations of fluid models as an analysis tool. We assess the predictive power of fluid models as follows:

**Insight 7.** *Qualitative Accuracy of Fluid Models.* Fluid models are highly predictive from a qualitative perspective, i.e., they accurately capture the direction of correlations between CCA performance and network parameters as well as the ranking of different CCAs according to performance metrics.

**Insight 8.** *Quantitative Accuracy of Fluid Models.* The accuracy of the quantitative predictions by fluid models depends on the metric: While the quantitative predictions of fluid models are highly accurate regarding loss and fairly accurate regarding buffer usage, the quantitative predictions regarding fairness and utilization are only partially accurate.

Despite their overall high predictive power, fluid models yield misleading results in some cases. We identified the following sources of potentially inaccurate predictions:
While our focus on this paper is on congestion-control algorithms [16, 50, 53], in particular, Scholz et al. [50] show that we are not aware of any work devising fluid models for BBR.

A dynamic model of TCP behavior is proposed using dynamical-systems techniques [47]. In one prominent work [38], a dynamic model of TCP behavior is proposed using a fluid-flow and stochastic differential-equation analysis. Using the Runge-Kutta algorithm, the fluid model also allows efficient time-stepped network simulations [36]. However, we are not aware of any work devising fluid models for BBR.

That said, BBR has been studied in a number of papers. In particular, Hock et al. [24] present a first independent study of BBRv1 and found fairness issues, and that multiple BBR flows operate at their in-flight cap in buffer-bloated networks. This work led to several interesting follow-up works [16, 50, 53]. In particular, Scholz et al. [50] show that BBRv1 flows are robustly able to claim a disproportionate share of the bandwidth. Ware et al. [57] recently complement these empirical studies by presenting a first analytical model (although not based on differential equations) capturing BBR’s behavior in competition with loss-based CCAs in deep buffers. Yang et al. [59] devise a simple fluid model for Adaptive-BBR, i.e., their adaptation of BBR specialized for wireless links. Neither of these model-based works possesses the generality of our fluid model, nor do they include a rigorous convergence analysis. BBRv2 has been investigated by a number of experiment-based studies [20, 31, 40, 51], finding mostly that BBRv2 resolves the most serious issues of BBRv1, but also identifying problematic facets of BBRv2 behavior, although not the ones found by this paper.

Recently, CCA research methodology has experienced innovation with promising proposals for an axiomatic approach [60] and a formal-verification approach [5]. These approaches are complementary to the fluid-model approach: While the axiomatic approach allows to identify fundamental design constraints and the formal-verification approach allows to identify network configurations in which CCA performance does not conform to specifications, neither of them is equally well-suited as the fluid-model approach to reveal the qualitative and quantitative effects of network settings and competing CCAs on CCA performance.
Injong Rhee, Lisong Xu, Sangtae Ha, Alexander Zimmermann, Lars Changhyun Lee, Chunjong Park, Keon Jang, Sue Moon, and Dongsu Belma Turkovic, Fernando A Kuipers, and Steve Uhlig. 2019. Fifty
Yeong-Jun Song, Geon-Hwan Kim, Imtiaz Mahmud, Won-Kyeong Seo,
Gaurav Raina. 2006.
Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah,
A ETHICS
This work raises no ethical issues.

B LOSS-BASED CCA MODELS
In this section, we discuss the existing CCA models for Reno and CUBIC, which have been used in our simulations. We also validate these CCA fluid models in isolation

B.1 Reno
In its congestion-avoidance phase, TCP Reno increases the congestion-window size by 1/w upon successful transmission (signaled by an ACK) and cuts it in half upon loss. This adaptation logic is approximated by the following differential equation for the congestion-window size $w_i(t)$ of agent $i$ using path $\pi_i$ [37]:

$$
\dot{w}_i = x_i(t - d_i^p) \cdot (1 - p_i(t - d_i^p)) \cdot \frac{1}{w_i} - x_i(t - d_i^p) \cdot p_i(t - d_i^p) \cdot \frac{w_j}{2}
$$

(39)

B.2 CUBIC
In contrast, TCP CUBIC cannot directly be described with a differential equation for the congestion-window size. Instead, Vardoyan et al. [55] suggest to track two instrumental variables in CUBIC, namely the time since last loss of agent $i$, $s_i$, and the congestion-window size at that moment of loss, $w_i^{\text{max}}$:

$$
\dot{s}_i = 1 - s_i \cdot x_i(t - d_i^p) \cdot p_i(t - d_i^p),
$$

(40a)

$$
\dot{w}_i^{\text{max}} = (w_i - w_i^{\text{max}}) \cdot x_i(t - d_i^p) \cdot p_i(t - d_i^p).
$$

(40b)

The intuition behind Eq. (40a) is that $s_i$ is increased by 1 in absence of loss ($p_i = 0$) and reduced to 0 when a loss occurs. Equation (40b) describes that $w_i^{\text{max}}(t)$ should be updated to $w_i(t)$ in presence of loss. Knowing $s_i$ and $w_i^{\text{max}}$, the congestion-window size $w$ can be determined by the CUBIC window-growth function [21],

$$
w_i = c \cdot \left( s_i - \sqrt{\frac{w_i^{\text{max}} \cdot b}{c}} \right)^3 + w_i^{\text{max}},
$$

(41)

where $c$ and $b$ are configurable parameters with standardized values of 0.4 and 0.7, respectively [49]. Moreover, the CUBIC implementation in the Linux kernel uses a time unit of around 1 second for $s_i(t)$ [22].

B.3 Trace Validation of Models
Figs. 11 and 12 present a comparison of single-sender traces obtained from running both model simulation and mininet experiments. The fluid model correctly predicts that the rate growth of Reno and CUBIC decouples from the congestion-window growth as soon as the buffer fills up. In addition, the fluid models correctly capture that Reno and CUBIC lead to considerably smaller loss (barely visible) than BBVr1, which is insensitive to loss (cf. §4.2). Finally, the fluid model correctly predicts that the sending rate of loss-based CCAs never exceeds the bottleneck rate under RED, while the congestion windows can temporarily exceed the network BDP under a drop-tail queuing discipline. As a result, the smaller buffer usage under RED is also reflected in the model, although the difference between RED and drop-tail is more pronounced in the model. This last difference is due to the idealization of the RED algorithm in the model.

C AGGREGATE VALIDATION FOR SHORT RTT
Figures 13, 14, 15, 16, and 17 extend the validation, performed in §4.3, of fluid models regarding the aggregate metrics Jain fairness, loss rates, buffer occupancy, utilization and jitter, respectively. In contrast to the validation in the body of the paper, the fluid-model predictions are experimentally validated for a bottleneck-link delay of 5 milliseconds and total RTTs between 10 and 20 milliseconds.

D ANALYSIS PROOFS
This appendix section contains the proofs of the theorems in §5.

D.1 Proof of Theorem 1
First, we consider the case for $q_r = 0$. In that case, the only solution to the conditions in Eq. (35) in terms of $\{\Delta_i\}_{i \in U_r}$ is $\Delta_i = 1 \forall i \in U_r$, which shows that the network is in equilibrium for $q_r = 0$ and $d_i = \sum_{e \in \pi_i} q_e/C_i \forall i \in U_r$.

For $q_r > 0$, we note that the conditions in Eq. (35) can be transformed into the following conditions for each $x_i^{\text{bl}}$:

$$
x_i^{\text{bl}} = \max(1, \frac{1}{\Delta_i}) \cdot (C_r - \sum_{j \neq i} \min(1, \Delta_j) \cdot x_j^{\text{bl}})
$$

(42)

$$
x_i^{\text{bl}} = C_r - \max(\frac{q}{5}, \frac{1}{\Delta_i}) \cdot \sum_{j \neq i} \min(1, \Delta_j) \cdot x_j^{\text{bl}}
$$

(43)

Clearly, the previously found solution $\Delta_i = 1 \forall i \in U_r$ is also a solution to the conditions in Eqs. (42) and (43). Hence, we have proven that the network is in equilibrium if $d_i = \sum_{e \in \pi_i} q_e/C_i \forall i \in U_r$.

It remains to prove that the previously mentioned equilibria are the only possible equilibria. To confirm the uniqueness of these equilibria, we first assume an equilibrium where $U_r \in U_r$, $\Delta_i > 1$. For that agent $i$, the maximum term in Eq. (42) is exactly 1, and the maximum term in Eq. (42) is smaller than 1, which makes the conditions contradictory and rules out an equilibrium. Conversely, if assuming an equilibrium where $U_r \in U_r$, $\Delta_i < 1$, the maximum term in Eq. (42) is $1/\Delta_i > 1$, and the maximum term in Eq. (43) is also $1/\Delta_i$. 

IMC ’22, October 25–27, 2022, Nice, France
Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid
Figure 11: Reno trace validation

Figure 12: CUBIC trace validation

Figure 13: Fairness validation

Figure 14: Loss validation

Figure 15: Queuing validation
which again leads to contradictory equations. Hence, no equilibria other than the equilibrium with \( \forall i \in U_r, \Delta_i = 1 \) are possible, which concludes the proof.

### D.2 Proof of Theorem 2

In the scenario under consideration, the equilibrium requires that the propagation delay \( d_i \) is equal for all senders. Moreover, it holds that \( q_t = 0 \ \forall t \neq t^\prime \). Hence, we can simplify: \( \Delta_i = \Delta(q_T) = 2d/(d + q_T/C_r) \), where \( d \) is the propagation delay experienced by all agents. As a result, the equilibrium requires that \( \Delta(q_T) = 1 \iff q_T = dC_r \).

We now consider a configuration where the senders are out of equilibrium and constrained by the congestion-window limit, i.e., \( \Delta(q_T) < 1 \iff q_T > dC_r \). The dynamics of the bottleneck-bandwidth estimates by sender \( i \) are then given by:

\[
\dot{x}_i^{bl} = \frac{\Delta(q_T) x_i^{bl} C_r}{\Delta(q_T) \sum_{k \in U_r} x_k^{bl}} - x_i^{bl} = \frac{x_i^{hl} C_r}{\sum_{k \in U_r} x_k^{hl}} - x_i^{bl} \tag{44}
\]

Moreover, the dynamics of the bottleneck-link queue \( q_T \) are:

\[
\dot{q}_T = y_T - C_T = \Delta(q_T) \sum_{i \in U_r} x_i^{hl} - C_T \tag{45}
\]

where \( y_T \) is the arrival rate at bottleneck link \( t^\prime \). Based on these dynamics, we derive the dynamics of the arrival rate \( y_T \):

\[
\dot{y}_T = \frac{\Delta(q_T)}{C_r} \sum_{i \in U_r} x_i^{hl} + \Delta(q_T) \sum_{i \in U_r} x_i^{hl} = -\frac{1}{C_r (d + \frac{q_T}{C_r})} y_T^2 + \left( 1 - \frac{1}{d + \frac{q_T}{C_r}} \right) y_T + \Delta(q_T) C_r \tag{46}
\]

Building on this formalization, we can define a classic non-linear dynamic system with \( y_T \) and \( q_T \) as state variables, and \( \dot{y}_T \) and \( \dot{q}_T \) as entries of the vector-valued function \( f \) describing the dynamics. To characterize the stability of that system, we can then employ the indirect Lyapunov method [46]. This method states that a system is locally asymptotically stable if the Jacobian matrix of the system dynamics \( f \) has eigenvalues with exclusively negative real parts when evaluated at the equilibrium. The Jacobian matrix \( J_f \in \mathbb{R}^{2 \times 2} \) has the following entries:

\[
\frac{\partial \dot{y}_T}{\partial y_T} = \frac{2}{C_r (d + \frac{q_T}{C_r})} y_T + \frac{1}{d + \frac{q_T}{C_r}} - 1 \]

\[
\frac{\partial \dot{y}_T}{\partial q_T} = \frac{y_T^2}{C_r (d + \frac{q_T}{C_r})^2} - \frac{y_T}{C_r (d + \frac{q_T}{C_r})^2} - \frac{2d}{(d + \frac{q_T}{C_r})^2} \tag{47}
\]

Evaluating this Jacobian matrix at the equilibrium, i.e., \( y_T = C_T \) and \( q_T = dC_T \), yields a matrix for which the maximum eigenvalue \( \lambda^* \) can be found via the characteristic
equation:
\[
J_f(C_{r^*}, dC_{r^*}) = \begin{pmatrix}
\frac{-1}{2d} & -\frac{1}{2d} \\
\frac{1}{2} & 0
\end{pmatrix}
\]
\[\implies \lambda^* = -\left(\frac{1}{4d} + \frac{1}{2} \pm \frac{1}{2d}(d - \frac{1}{2})\right)\]

Performing a case distinction on \( d \) confirms that the maximum eigenvalue \( \lambda^* \) is always negative:
\[
d \leq \frac{1}{2} : \quad \lambda^* = -\left(\frac{1}{4d} + \frac{1}{2} - \frac{1}{2}(d - \frac{1}{2})\right) = -1 < 0
\]
\[
d > \frac{1}{2} : \quad \lambda^* = -\left(\frac{1}{4d} + \frac{1}{2} + \frac{1}{2}(d - \frac{1}{2})\right) = -\frac{1}{2d} < 0
\]

Hence, we observe that the Jacobian matrix of the system dynamics has consistently negative eigenvalues, which by the indirect Lyapunov method proves the asymptotic stability of the dynamics.

### D.3 Proof of Theorem 3

Given \( \Delta_i \geq \frac{5}{4} \) for all \( i \in U_r \), the equilibrium condition on \( \{x_i^{\text{tgt}}\}_{i \in U_r} \) is:
\[
\forall i \in U_r, \quad x_i^{\text{tgt}} - \frac{5/4 x_i^{\text{tgt}} C_r^*}{(N + \frac{1}{4} x_i^{\text{tgt}})} = \frac{5C_r^*}{4N + 1}
\]

This equation system requires all \( \{x_i^{\text{tgt}}\}_{i \in U_r} \) to be equal, which allows a straightforward solution:
\[
\forall i \in U_r, \quad x_i^{\text{tgt}} = \frac{5/4 x_i^{\text{tgt}} C_r^*}{(N + \frac{1}{4} x_i^{\text{tgt}})} = \frac{5C_r^*}{4N + 1}
\]

It remains to show that this equilibrium is asymptotically stable, for which we employ the indirect Lyapunov method. We apply this method to a non-linear dynamic process with \( \{x_i^{\text{tgt}}\}_{i \in U_r} \) as state variables and \( \{\dot{x}_i^{\text{tgt}}\}_{i \in U_r} \) as vector-valued evolution function \( f \). The Jacobian matrix \( J_f \) has the following entries, which we evaluate at the equilibrium:
\[
\frac{\partial \dot{x}_i}{\partial x_i} = \frac{5/4 C_{r^*} \sum_{j \neq i} x_j^{\text{tgt}}}{(5/4 x_i + \sum_{j \neq i} x_j^{\text{tgt}})^2} - \frac{5}{4N + 1} = J_{ii}
\]
\[
\frac{\partial \dot{x}_j}{\partial x_i} = -\frac{5/4 C_{r^*} x_j^{\text{tgt}}}{(5/4 x_i + \sum_{j \neq i} x_j^{\text{tgt}})^2} = -\frac{5}{4N + 1} = J_{ij}
\]

The eigenpairs \( (\lambda, v) \) of \( J_f \) at the equilibrium satisfy the following conditions:
\[
\forall i \in U_r, \quad (J_{ii} - \lambda)v_i + J_{ij} \sum_{j \neq i} v_j = 0
\]

The first type of solution for this equation system is given by \( \lambda = J_{ii} - J_{ij} < 0 \) and every \( v \) with \( ||v||_1 = 0 \). The second type of solution is found by assuming \( \lambda \neq J_{ii} - J_{ij} \), which implies equal \( v_i \) \( \forall i \in U_r \) and hence (together with \( v_i \neq 0 \)) \( \lambda = J_{ii} + (N - 1)J_{ij} < 0 \). Since the eigenvalues of the Jacobian are thus consistently negative, the indirect Lyapunov method suggests that the dynamics are asymptotically stable.

### D.4 Proof of Theorem 4

The equilibrium conditions can be translated into the following constraints given \( q_r > 0 \):
\[
\lambda = J_{ii} + (N - 1)J_{ij} < 0
\]
\[
\lambda = J_{ii} + (N - 1)J_{ij} < 0
\]
\[
\lambda = J_{ii} + (N - 1)J_{ij} < 0
\]

While these constraints potentially admit multiple equilibria, the equilibrium from Theorem 4 is a special equilibrium for which \( \delta_i \) is equal across all \( i \in U_r \), i.e., \( \delta_i = \delta \). Substituting \( \delta \) for all \( \delta_i \) and equating Eq. (56) with Eq. (57), which first yields:
\[
\forall i \in U_r, \quad x_i^{\text{tgt}} = \frac{5}{4} \sum_{j \neq i} x_j^{\text{tgt}} = \frac{5}{4} \sum_{j \neq i} x_j^{\text{tgt}}
\]

where the equation systems require that all \( x_i^{\text{tgt}} \in U_r \) equal the same value (hence perfect fairness), where this value can be found using Eq. (56). By inserting \( x_i^{\text{tgt}} \) from Eq. (58) into Eq. (57), it can be shown that \( \delta \leq 1 \) by producing a contradiction for \( \delta > 1 \). In contrast, solving that equation given \( \delta \leq 1 \) yields \( \delta = \frac{5N^2}{8N^2} \), which is equivalent to the condition in Theorem 4. This insight concludes the proof.

### D.5 Proof of Theorem 5

In the scenario under consideration, the equilibrium requires that the propagation delay \( d_i \) is equal for all senders. Moreover, it holds that \( q_t = 0 \forall t \neq t' \). Hence, we can simplify:
\[
\delta_i = \delta(q_r) = d/(d + q_r/C_r^*)
\]

We translate the reduced model from §5.2.1 into a nonlinear dynamic process with the sending rates \( \{x_i\}_{i \in U_r} \) and the queue length \( q_r \) as state variables. The evolution of these state variables is given by vector-valued function \( f \) with the following entries:
\[
\dot{x}_i = \left(\frac{C_{r^*} - \sum_{k \in U_r} x_k}{C_{r^*}(d + q_r/C_{r^*})} + \frac{5/4 C_{r^*}}{5/4 x_i + \sum_{j \neq i} x_j} - 1\right) x_i
\]

\[
\dot{q}_r = \sum_{i \in U_r} x_i - C_{r^*}
\]
The corresponding Jacobian matrix $J_f$ is composed of the following entries:

$$\frac{\partial \mathbf{x}}{\partial x_i} = \frac{C_r}{C_r (d + q_r / C_r)} - \frac{5/4 \delta C \sum_{j \neq i} x_j}{(5/4 x_i + \sum_{j \neq i} x_j)^2} = 1 \quad (61)$$

$$\frac{\partial \mathbf{x}}{\partial x_j} = - \frac{C_r (d + q_r / C_r)}{C_r} - \frac{5/4 \delta C x_i}{(5/4 x_i + \sum_{j \neq i} x_j)^2} = 0 \quad (62)$$

$$\frac{\partial \mathbf{x}}{\partial q_i} = \frac{1}{d + \frac{q_r}{C_r}} \left( \frac{C_r - \sum_{k \in U_r} x_k}{C_r} - \frac{5/4 \delta C}{(d + \frac{q_r}{C_r})^2} \right) x_i = 1 \quad (63)$$

$$\frac{\partial \mathbf{q}}{\partial q_i} = 1 \quad (64)$$

Evaluating the Jacobian matrix at the equilibrium point from Theorem 4 yields the following matrix $J$:

$$\frac{\partial x_i}{\partial x_i} = - \frac{4N + 1}{5N^2 d} \frac{5}{4N + 1} =: J_{ii} \quad \frac{\partial q}{\partial x_i} = 1 \quad (65)$$

$$\frac{\partial x_i}{\partial x_j} = - \frac{4N + 1}{5N^2 d} \frac{4}{4N + 1} =: J_{ij} \quad \frac{\partial q}{\partial x_j} = 0 \quad (66)$$

$$\frac{\partial x_i}{\partial q} = - \frac{4N + 1}{5N^2 d} =: J_{iq} \quad (67)$$

By Lyapunov’s indirect method, the above Jacobian matrix must have exclusively negative eigenvalues in order for the equilibrium to be asymptotically stable, i.e., for every pair $(\lambda, \mathbf{v})$ with $J\mathbf{v} = \lambda \mathbf{v}$, the eigenvalue $\lambda$ must be lower than 0. To verify this property $J$, we concretize the eigenvalue condition:

$$\forall i \in U_r: \quad J_{ii} v_i + J_{ij} \sum_{j \neq i} v_j + J_{iq} v_q = \lambda v_i \quad (68)$$

$$\sum_{i \in U_r} v_i = \lambda v_q \quad (69)$$

By solving these equations for $v_q$ and equating the resulting terms, we obtain the following conditions:

$$\forall i \in U_r: \sum_{k \in U_r} v_k = \frac{\lambda}{J_{iq}} \left( (\lambda - J_{ii}) v_i - J_{ij} \sum_{j \neq i} v_j \right) \quad (70)$$

This equation allows two types of solutions. First, for $\lambda = J_{ii} - J_{ij}$, the set of valid eigenvectors $\mathbf{v}$ is only constrained by a condition on $\sum_{k \in U_r} v_k$; more importantly for the proof, $\lambda$ is negative. Second, for $\lambda \neq J_{ii} - J_{ij}$, the values $v_i \forall i \in U_r$ must be equal such that the equation system from Eq. (70) can be collapsed into a single quadratic equation, which yields the maximum eigenvalue $\lambda^*$:

$$N \cdot v_i = \frac{\lambda}{J_{iq}} \left( (\lambda - J_{ii}) v_i - J_{ij} (N - 1) v_i \right) \Rightarrow \lambda^* = -1 \quad (71)$$

Since the maximum eigenvalue $\lambda^*$ is negative, all eigenvalues of $J$ are negative, which by the indirect Lyapunov method proves that the dynamic process defined by $f$ is asymptotically stable.