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

Simon Scherrer
ETH Zurich

Markus Legner
ETH Zurich

Adrian Perrig
ETH Zurich

Stefan Schmid
TU Berlin/Fraunhofer SIT

ABSTRACT

Google's BBR is the most prominent result of the recently revived quest for efficient, fair, and flexible congestion-control algorithms (CCAs). While 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.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Network performance modeling**.

ACM Reference Format:
Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid. 2022. Model-Based Insights on the Performance, Fairness, and Stability of BBR. In *Internet Measurement Conference (IMC '22), October 25–27, 2022, Nice, France.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3517745.3561420

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, 51, 54, 58] 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, 41, 52].

Still, the characterization of BBR performance remains incomplete. Experiment-based studies [20, 24, 31, 41, 51, 54], 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 valuable for specific settings (e.g., deep buffers [57] or wireless links [60]); however, a deep theoretical understanding requires a model that allows investigation of general settings and the convergence process.

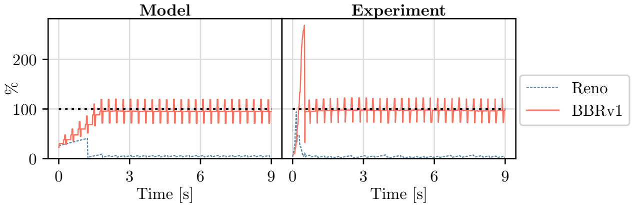


Figure 1: Competition of sending rates (in % of link bandwidth) between a Reno flow and a BBRv1 flow.

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, 37, 39, 48, 53, 56]. 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, 56]. 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 with the network emulator mininet [34]. The validated model confirms BBR performance issues from previous studies and yields new insights. Moreover, we apply dynamical-system analysis (i.e., 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 general fluid model for BBR (versions 1 and 2), using new techniques such as sigmoid pulses and mode variables.

IMC '22, October 25–27, 2022, Nice, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9259-4/22/10…$15.00
https://doi.org/10.1145/3517745.3561420

- 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 analytically confirm the previous insight that BBRv1 can lead to unfair bandwidth allocations—especially when it competes against loss-based CCAs, but also in competition with itself given unequal RTTs.
- We confirm 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 $\dot{f}$, unless the argument 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 $y_\ell(t)$ at link ℓ is

$$y_\ell = \sum_{i \in U_\ell} x_i\left(t - d^{\mathrm{f}}_{i,\ell}\right), \tag{1}$$

where U_ℓ is the set of agents using link ℓ, $x_i(t)$ is the sending rate of agent i at time t, and $d^{\mathrm{f}}_{i,\ell}$ 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:

$$\dot{q}_\ell = (1 - p_\ell) \cdot y_\ell - C_\ell, \quad q_\ell(t) \in [0, B_\ell], \tag{2}$$

where $p_\ell(t)$ 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 $q_\ell(t)$. The latency of a path is the sum of link latencies:

$$\tau_{\pi_i} = \sum_{\ell \in \pi_i} \tau_\ell = \sum_{\ell \in \pi_i} d_\ell + \frac{q_\ell}{C_\ell}. \tag{3}$$

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 [59]. To facilitate analytical treatment, we refine previous models by a smooth approximation:

$$p_\ell(t) = \sigma\big(y_\ell(t) - C_\ell\big) \cdot \left(1 - \frac{C_\ell}{y_\ell}\right) \cdot \left(\frac{q_\ell}{B_\ell}\right)^L \tag{4}$$

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

$$\sigma(v) = \frac{1}{1 + \mathrm{e}^{-K \cdot v}} \tag{5}$$

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. More precisely, RED keeps the drop probability at 0 if the queue size is below a configurable threshold q_0, increases the drop probability linearly to a configurable value p_1 for queue sizes up to q_1, and drops all packets for larger queue sizes. We approximate the RED behavior as follows, representing the general idea of RED:

$$p_\ell = \frac{q_\ell}{B_\ell} \quad \in [0, 1]. \tag{6}$$

which corresponds to a RED configuration with $p_1 = 1$, $q_1 = B_\ell$, and $q_0 = 0$. The extension of the fluid-model simulator to other RED configurations is straightforward.

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

$$p_{\pi_i}(t) = 1 - \prod_{\ell \in \pi_i} \left(1 - p_\ell\left(t + d^{\mathrm{f}}_{i,\ell}\right)\right) \approx \sum_{\ell \in \pi_i} p_\ell\left(t + d^{\mathrm{f}}_{i,\ell}\right). \tag{7}$$

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:

$$x_i = \frac{w_i}{\tau_i}. \tag{8}$$

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 `BtlBw` and the minimal round-trip time (RTT) `RTprop` (i.e., propagation delay). To estimate these properties, BBR constantly switches between two states, namely the `ProbeBW` state and the `ProbeRTT` state. While the `ProbeBW` state consumes most of flow lifetime and is considerably different across the two BBR versions, the `ProbeRTT` state is only infrequently and briefly entered and is mostly identical across both BBR versions.

`ProbeRTT` state. BBR enters the `ProbeRTT` state if no smaller round-trip time than the existing `RTprop` estimate is observed for 10 seconds. To discover the propagation delay, the `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 `ProbeRTT` inflight limit in BBRv2 has been chosen to half the estimated bandwidth-delay product, i.e., half the product of `BtlBw` and `RTprop`.

`ProbeBW` state in BBRv1. The `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 `RTprop`. In one phase randomly chosen from the first 7 phases, BBRv1 sets its pacing rate to $5/4 \cdot$ `BtlBw` to find the capacity limit of the path. In the subsequent phase, BBRv1 decreases its pacing rate to $3/4 \cdot$ `BtlBw` to drain the queues potentially built up during the aggressive previous phase. In the other 6 phases of the period, BBRv1 paces at rate `BtlBw`. 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. 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, 58], letting BBRv1 degenerate into a window-based CCA in these circumstances.

`ProbeBW` state in BBRv2. This unintentional relevance of the inflight limit in some circumstances, plus the unfairness towards loss-based CCAs in shallow buffers, led Google to revise the `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 `BtlBw` for one `RTprop`, with the goal to achieve an inflight corresponding to the bandwidth-delay product. Then, BBRv2 sets its pacing rate to $5/4 \cdot$ `BtlBw` and increases the inflight until it reaches 5/4 of the estimated BDP *or* the loss rate exceeds 2%. At this point, the bottleneck-bandwidth estimate `BtlBw` is updated to the maximum delivery rate from the last two `ProbeBW` periods. Moreover, BBRv2 also records the maximum tenable inflight in state variable `inflight_hi`, which tracks the observed inflight, but is reduced by a multiplicative decrease β if the exponential-increase phase has been terminated by excessive loss. Afterwards, BBRv2 chooses a pacing rate of $3/4 \cdot$ `BtlBw` 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 `inflight_hi` (where the erased 15% are termed `headroom` in BBRv2). Once the inflight has been reduced to that level, BBRv2 enters into *cruising* mode. In cruising mode, BBRv2 aims to keep its inflight on a safe level by introducing the additional inflight bound `inflight_lo`, which is activated if packet loss occurs: `inflight_lo` starts from the congestion-window size at the moment of loss and is reduced by β upon packet loss. In contrast to `inflight_hi`, which serves as a *long-term* inflight bound, `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 long-term bound `inflight_hi` (discounted by `headroom` in cruising mode), and the short-term bound `inflight_lo` (if activated).

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 `RTprop`, which we represent with variable $\tau_i^{\min}(t)$ for the `RTprop` estimate of agent i at time t. The variable $\tau_i^{\min}$ is continuously adjusted downwards upon encountering smaller RTTs:

$$\dot{\tau}_i^{\min} = -\Gamma\left(\tau_i^{\min}(t) - \tau_i(t - d_i^{\mathrm{p}})\right) \tag{9}$$

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):

$$\Gamma(v) = v \cdot \sigma(v). \tag{10}$$

In Eq. (9), this formulation of Γ leads to a proportional decrease in minimum RTT estimate $\tau_i^{\min}$ if the currently observed delay $\tau_i(t - d_i^{\mathrm{p}})$ is below the previously observed minimum $\tau_i^{\min}$, i.e., if the argument of Γ exceeds 0. Otherwise, $\tau_i^{\min}$ is preserved.

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

$$\Delta m_i^{\mathrm{prt}} = \sigma\left(t_i^{\mathrm{prt}} - T_i^{\mathrm{prt}}\right) \cdot \left((1 - m_i^{\mathrm{prt}}) - m_i^{\mathrm{prt}}\right) \tag{11}$$

where T_i^{prt} is the time period between entries and exits of the `ProbeRTT` state for agent i. Note that Eq. (11) represents an update rule for simulations rather than a differential equation, as m_i^{prt} is discrete. Upon time-out of the current `ProbeRTT` timer (i.e., $\sigma \approx 1$), Eq. (11) leads to an inversion of m_i^{prt}:

$$m_i^{\mathrm{prt}} + \Delta m_i^{\mathrm{prt}} = \begin{cases} 1 + 1 - 2 \cdot 1 = 0 & \text{if } m_i^{\mathrm{prt}} = 1 \\ 0 + 1 - 2 \cdot 0 = 1 & \text{if } m_i^{\mathrm{prt}} = 0. \end{cases} \tag{12}$$

The two time-related variables in Eq. (11) behave as follows:

$$T_i^{\mathrm{prt}} = m_i^{\mathrm{prt}} \cdot 0.2 + \left(1 - m_i^{\mathrm{prt}}\right) \cdot 10 \tag{13}$$

$$\dot{t}_i^{\mathrm{prt}} = 1 - \sigma\left(t_i^{\mathrm{prt}} - T_i^{\mathrm{prt}}\right) \cdot t_i^{\mathrm{prt}} - \sigma\left(\tau_i^{\min} - \tau_i(t - d_\ell)\right) \cdot t_i^{\mathrm{prt}} \tag{14}$$

The constants in Eq. (13) cause BBR to remain in `ProbeRTT` state for 0.2 seconds and to wait 10 seconds before re-entering the state after exiting it. Eq. (14) causes a reset of the `ProbeRTT` timer to 0 if the timer has reached the limit T_i^{prt} or a lower RTT has been measured, and to tick up otherwise.

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

$$x_i = m_i^{\mathrm{prt}} \cdot \frac{w_i^{\mathrm{prt}}}{\tau_i} - \left(1 - m_i^{\mathrm{prt}}\right) \cdot x_i^{\mathrm{pbw}} \tag{15}$$

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

$$x_i^{\text{pbw}} = \min\left(\frac{w_i^{\text{pbw}}}{\tau_i}, x_i^{\text{pcg}}\right) \tag{16}$$

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

$$\dot{t}_i^{\text{pbw}} = 1 - \sigma\left(t_i^{\text{pbw}} - T_i^{\text{pbw}}\right) \cdot t_i^{\text{pbw}} \tag{17}$$

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

$$x_i^{\text{dlv}} = \frac{x_i\left(t - d_i^{\text{p}}\right)}{y_\ell\left(t - d_{i,\ell}^{\text{b}}\right)} \cdot \begin{cases} C_\ell & \text{if } q_\ell(t - d_{i,\ell}^{\text{b}}) > 0 \\ y_\ell\left(t - d_{i,\ell}^{\text{b}}\right) & \text{otherwise} \end{cases} \tag{18}$$

where d_i^{p} is the propagation delay of flow i, and $d_{i,\ell}^{\text{b}}$ is the propagation delay from link ℓ to sender i (via the destination host). As a result, the fraction in Eq. (18) denotes the share of flow i's traffic, emitted one RTT before time t, among the aggregate traffic simultaneously arriving at link ℓ.

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

$$\dot{x}_i^{\max} = \Gamma(x_i - x_i^{\max}) - \sigma(0.01 - t_i^{\text{pbw}}) \cdot x_i^{\max} \tag{19}$$

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

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

$$\dot{v}_i = x_i - x_i^{\text{dlv}} \tag{20}$$

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 $\tau_i^{\min}$, i.e., $T_i^{\text{pbw}} = 8 \cdot \tau_i^{\min}$. The bottleneck-bandwidth estimate x_i^{btl} is updated to the maximum delivery rate $x_i^{\max}$ at the end of the period, which we formalize as follows:

$$\dot{x}_i^{\text{btl}} = \sigma\left(t_i^{\text{pbw}} - T_i^{\text{pbw}} + 0.01\right) \cdot \left(x_i^{\max} - x_i^{\text{btl}}\right) \tag{21}$$

In general, BBRv1 prescribes a pacing rate x_i^{pcg} equal to x_i^{btl} in each phase, but increases x_i^{pcg} to $5/4 \cdot x_i^{\text{btl}}$ 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 Φ, which is 1 if BBRv1 is in phase ϕ and 0 otherwise:

$$\Phi_i(t, \phi) = \sigma\left(t^{\text{pbw}}(t) - \phi \cdot \tau_i^{\min}\right) \cdot \sigma\left((\phi + 1) \cdot \tau_i^{\min} - t^{\text{pbw}}\right) \tag{22}$$

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

$$x_i^{\text{pcg}} = x_i^{\text{btl}} \cdot \left(1 + \tfrac{1}{4} \cdot \Phi_i(t, \phi_i) - \tfrac{1}{4} \cdot \Phi_i(t, \phi_i + 1)\right) \tag{23}$$

In the implementation of BBRv1, the phase ϕ_i is randomly chosen from $\{0, ..., 6\}$ every time BBRv1 switches from ProbeRTT state back to ProbeBW state. Since such randomness is incompatible with the determinism of fluid models, we mimic the randomness of ϕ_i by choosing it as $i \bmod 6$, where we assume the agent identifier i to be a natural number. This agent-dependent choice of ϕ_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:

$$w_i^{\text{prt}} = 4 \qquad\qquad w_i^{\text{pbw}} = 2 \cdot \overline{w}_i = 2 \cdot x_i^{\text{btl}} \cdot \tau_i^{\min} \tag{24}$$

where $\overline{w}_i$ 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 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:

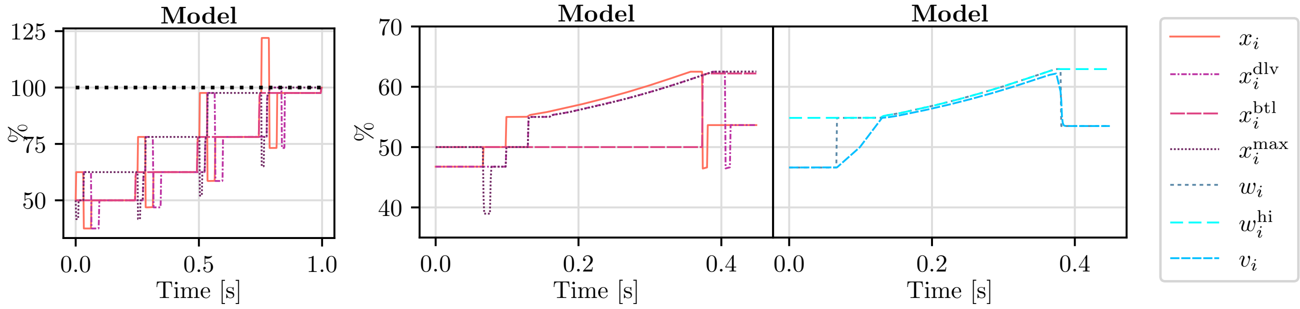
$$T_i^{\text{pbw}} = \min\left(62 \cdot \tau_i^{\min}, 2 + \frac{i}{N}\right) \tag{25}$$

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 $m_i^{\text{dwn}}(t)$, which indicates whether agent i is attempting to reduce its inflight at time t, and $m_i^{\text{crs}}(t)$, which indicates whether agent i is cruising at time t. The mode variable m_i^{dwn} affects the pacing rate x_i^{pcg} as follows:

$$x_i^{\text{pcg}} = x_i^{\text{btl}} \cdot \left(1 + \tfrac{1}{4} \cdot \sigma\left(t_i^{\text{pbw}} - \tau_i^{\min}\right) \cdot \left(1 - m_i^{\text{dwn}}\right) - \tfrac{1}{4} \cdot m_i^{\text{dwn}}\right) \tag{26}$$

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

Third, while we modelled phase transitions in BBRv1 as purely dependent on time t_i^{pbw}, 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 $5/4 \cdot \overline{w}_i$ or loss p_{π_i} exceeds 2%, and is disabled once the reduced pacing rate has reduced the inflight v_i to the draining target $w_i^- = \min(\overline{w}_i, 0.85 \cdot w_i^{\text{hi}})$,



(a) BBRv1.

(b) BBRv2 (left: rate, right: inflight limits).

Figure 2: Visualization of BBR fluid-model variables link capacity normalized to 100%, single flow)

where w^{hi} is the variable accommodating the long-term bound `inflight_hi` in our model:

$$\begin{aligned}\Delta m_i^{\text{dwn}} = \ & \left(1 - m_i^{\text{crs}}\right) \cdot \left(1 - m_i^{\text{dwn}}\right) \cdot \sigma\left(t_i^{\text{pbw}} - \tau_i^{\min}\right) \\ & \cdot \min\left(\sigma\left(v_i - 5/4 \cdot \overline{w}_i\right) + \sigma\left(p_{\pi i} - 0.02\right),\ 1\right) \\ & - m_i^{\text{dwn}} \cdot \sigma\left(w_i^- - v_i\right)\end{aligned} \tag{27}$$

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:

$$\Delta m_i^{\text{crs}} = -\Delta m_i^{\text{dwn}} - \sigma\left(t_i^{\text{pbw}} - T_i^{\text{pbw}}\right) \cdot m_i^{\text{crs}} \tag{28}$$

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

$$\dot{x}_i^{\text{btl}} = m_i^{\text{dwn}} \cdot \left(\max\left(x_i^{\max},\ x_i^{\max}(t - T^{\text{pbw}})\right) - x_i^{\text{btl}}\right) \tag{29}$$

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^{hi} is exponentially adjusted upwards when it represents the relevant constraint on the sending rate ($v_i = w_i^{\text{hi}}$) 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:

$$\begin{aligned}\dot{w}_i^{\text{hi}} = & \left(1 - m_i^{\text{crs}}\right) \cdot \sigma\left(t_i^{\text{pbw}} - \tau_i^{\min}\right) \cdot \sigma\left(v_i - w^{\text{hi}}\right) \cdot 2^{t_i^{\text{pbw}}/\tau_i^{\min}} \\ & - \sigma\left(p_{\pi_i} - 0.02\right) \cdot \frac{0.3}{\tau_i^{\min}} \cdot w_i^{\text{hi}}\end{aligned} \tag{30}$$

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:

$$\dot{w}_i^{\text{lo}} = \left(1 - m_i^{\text{crs}}\right) \cdot \left(w_i^- - w_i^{\text{lo}}\right) - m_i^{\text{crs}} \cdot \sigma(p_{\pi_i}) \cdot \frac{0.3 w_i^{\text{lo}}}{\tau_i^{\min}} \cdot \tag{31}$$

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

$$w_i^{\text{pbw}} = \min\left(2 \cdot \overline{w}_i,\ \left(1 - m_i^{\text{crs}}\right) \cdot w_i^{\text{hi}} + m_i^{\text{crs}} \cdot w_i^{\text{lo}}\right) \tag{32}$$

A final difference between BBRv1 and BBRv2 concerns the congestion-window size in `ProbeRTT` 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:

$$w_i^{\text{prt}} = \frac{\overline{w}_i}{2} \tag{33}$$

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 Set-up

4.1.1 Model-Based Computations. The implementation of the fluid models uses NumPy [42] and is available online [50]. 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 using iPerf [38]. All experiments are run with an Intel Core Intel Xeon E5-2695 v4 CPU.

4.1.3 Topology. As usual in the literature [20, 24, 31, 41, 51, 52, 58], we consider the dumbbell topology in Fig. 3. In this topology, N agents a_i, $i \in \{1, ..., N\}$, communicate with a destination host a_{d} via a switch S. In all paths, the shared link ℓ between switch S and destination host a_{d} constitutes the bottleneck link. The links ℓ_i, $i \in \{1, ..., N\}$, which connect the individual senders to switch S, are never saturated and therefore do not affect the sending rates. The propagation delays of these non-shared links are heterogeneous (randomly selected from a given range) such that the individual senders experience different RTTs. Switch S is equipped with a buffer, the size of which is measured in bandwidth-delay product (BDP) of the bottleneck link ℓ.

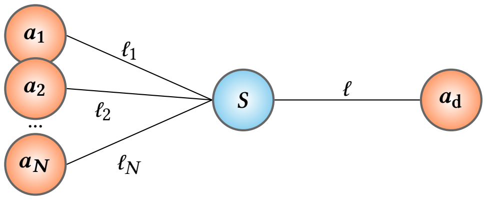


Figure 3: Dumbbell topology.

4.2 Validation of Trace Results

Using the validation set-up described in §4.1, we first verified traces as predicted by the fluid models regarding their similarity with traces from experimental measurements. The concrete network setting in this validation included a single sender, a bottleneck link ℓ with a rate of $C_\ell = 100$ Mbps (as recommended by mininet [33]) and with a propagation delay of $d_\ell = 10$ ms, a non-bottleneck link ℓ_1 with a delay of $d_{\ell_1} = 5.6$ 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 for 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. The model thus also reflects the relatively low buffer usage of BBRv2 under RED, although it slightly overestimates the buffer usage of BBRv2 under RED (cf. Fig. 5b).

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. In effect, a larger queue can build up until the increased drop probability stabilizes the queue, which translates into larger buffer usage in the experiments.

Difference: `ProbeRTT` state in BBRv2. The BBRv2 flow in the model simulation for drop-tail regularly enters the `ProbeRTT` state, unlike in the corresponding experiment. This observation can be explained as follows. In the model, BBRv2 regularly manages to drain the queue, therefore discovers the propagation delay early, and cannot detect a lower RTT afterwards; hence, it enters the `ProbeRTT` state every 10 seconds. In the experiment, however, BBRv2 never fully uncovers the propagation delay, and experiences random fluctuations in the RTT measurements. Hence, BBRv2 occasionally observes RTTs that fall short of the current minimum-RTT estimate, which keeps it from entering the `ProbeRTT` state.

In summary, the fluid models capture the differences among CCAs and queuing disciplines with high accuracy, especially relatively (e.g., which CCAs lead to lower buffer usage) and to a lesser degree also absolutely (e.g., level of buffer usage).

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 $C_\ell = 100$ Mbps, a bottleneck-link propagation delay $d_\ell = 10$ 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).

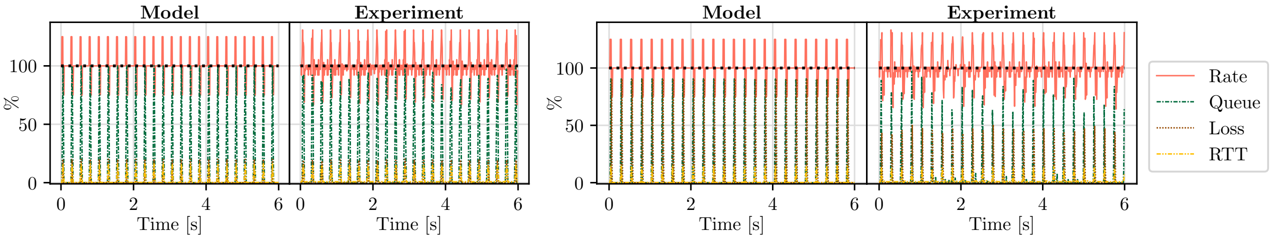
As in the trace validation, the comparisons reveal that the model predictions and the experiment results have both striking similarities and notable differences regarding all metrics.

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) competes with BBRv1 in shallow buffers, which has already been well documented in previous research [51, 58]. 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 of the congestion window 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.

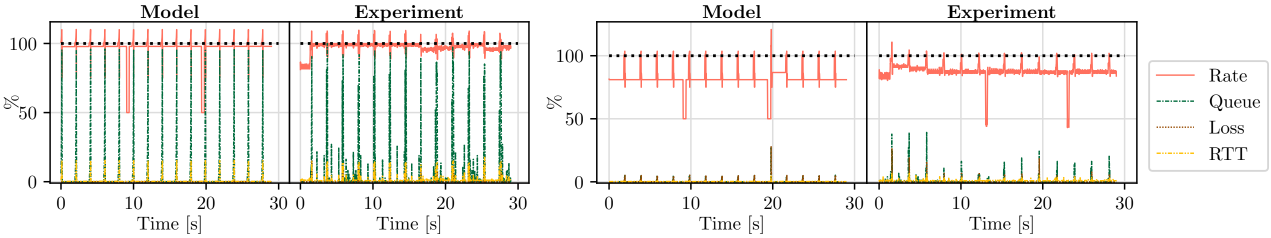
The fairness issues of BBRv1 have been largely resolved in BBRv2, as the fluid model and the experiment results show. However, BBRv2 is still unfair towards loss-based CCAs in RED buffers, where the higher loss sensitivity of loss-based CCAs is revealed.

One substantial difference between the fluid-model predictions and the experiment results is the decreasing fairness of BBRv1 in homogeneous settings 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 [51, 58], although for higher RTT differences than used in our network setting. This RTT unfairness stems from the inflight limit of BBRv1,



(a) Drop-tail (b) RED

Figure 4: BBRv1 trace validation



(a) Drop-tail (b) RED

Figure 5: BBRv2 trace validation

which becomes relevant in deep buffers: Since flows with a lower RTT are estimating a lower BDP and hence maintain a smaller congestion window, lower-RTT flows are more severely restricted by the inflight limit. Our fluid model can reveal this effect. In deep buffers with large queues, each BBRv1 sender i is restricted by its congestion-window size w_i, resulting in the following sending rate:

$$x_i = \frac{w_i}{\tau_i} = \frac{2\tau_i^{\text{min}} x_i^{\text{btl}}}{\tau_i} = \frac{2d_i^{\text{p}}}{d_i^{\text{p}} + q_\ell/C_\ell} x_i^{\text{btl}}, \tag{34}$$

where ℓ is the bottleneck link shared among all flows, and we assume $\tau_i^{\text{min}} = d_i^{\text{p}}$ (i.e., the propagation delay has been successfully uncovered). With these sending rates, the delivery rate for sender i is according to Eq. (18):

$$x_i^{\text{dlv}} = \frac{x_i C_\ell}{x_i + \sum_{j\neq i} x_j}. \tag{35}$$

Moreover, since the sending rates are static (because the varying pacing rate is overruled by the congestion-window constraint), the delivery rates are static across the bandwidth-probing interval as well, resulting in $x_i^{\text{max}} = x_i^{\text{dlv}}$. This maximum delivery rate x_i^{max} is monotonically increasing in propagation delay d_i^{p} of sender i:

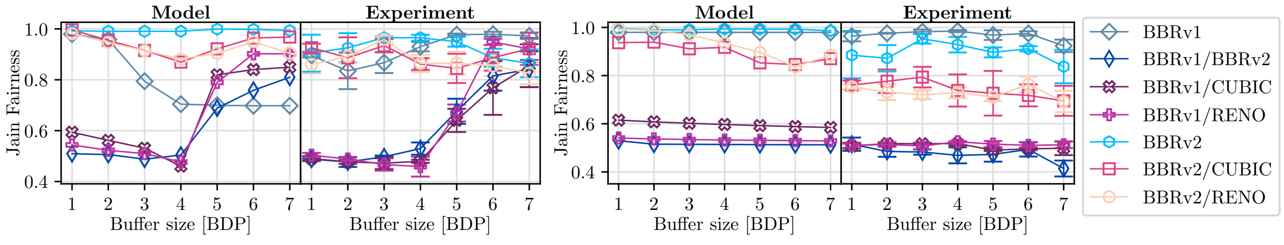
$$\frac{\partial x_i^{\text{max}}}{\partial d_i^{\text{p}}} = \frac{2x_i^{\text{btl}} \sum_{j\neq i} x_j}{\left(2d_i^{\text{p}} x_i^{\text{btl}} + (d_i^{\text{p}} + q_\ell/C_\ell)\sum_{j\neq i} x_j\right)^2} > 0 \tag{36}$$

Hence, flows with a higher RTT have a larger congestion window, can thus send at a higher rate, measure a higher maximum delivery rate, and in turn estimate a higher bottleneck bandwidth. This bottleneck-bandwidth estimate then increases the congestion-window size, leading to a positive feedback loop. However, the representation of the delivery rate in Eq. (35) idealizes the noisy relationship between sending rates and delivery rates in real-world buffers. This noise can eliminate the difference in measured delivery rates for flows with small RTT differences, and thus break the positive feedback loop in that case. Hence, the described effect only appears for relatively large RTT differences in reality.

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, 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 queuing discipline keeps loss rates roughly consistent across buffer sizes.

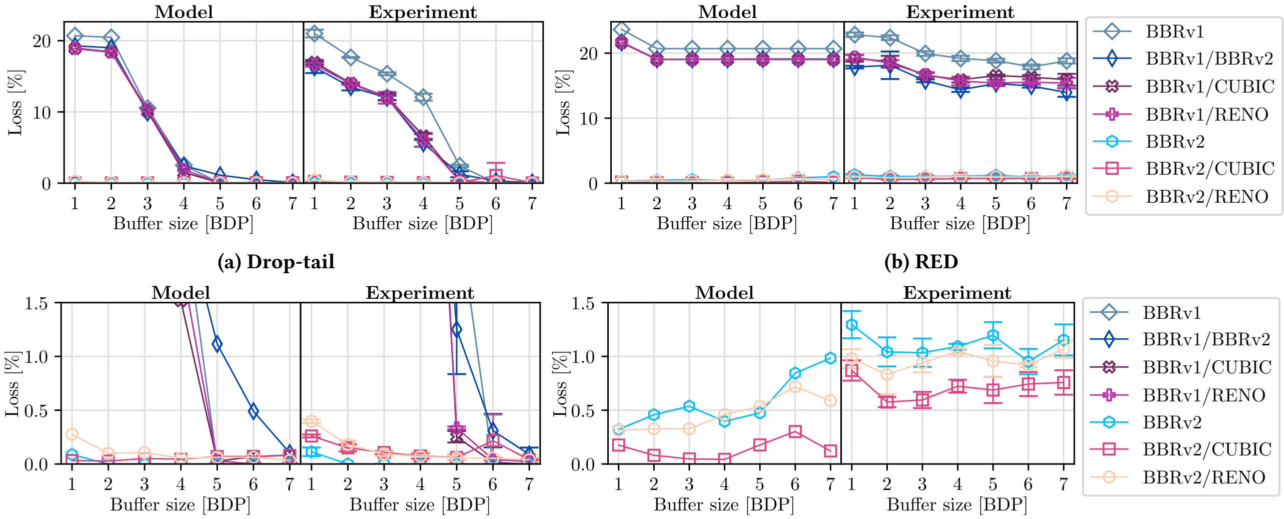
One obvious prediction error of the fluid model is the underestimation of loss rates for loss-sensitive CCAs given RED in Fig. 7d. This underestimation stems again from an idealization of the RED queue in the model, which determines the loss rate based on the current queue length. In contrast, real RED 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.



(a) Drop-tail

(b) RED

Figure 6: Fairness validation



(a) Drop-tail

(b) RED

(c) Drop-tail (zoomed in)

(d) RED (zoomed in)

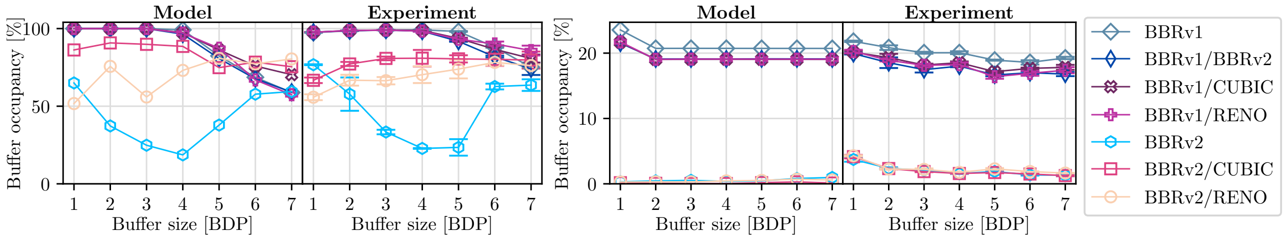
Figure 7: Loss validation

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. This effect is visible in combined settings of BBRv2 with loss-based CCAs, as 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 or in combined settings. 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. (32)). 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^{hi} (cf. Eq. (30)) 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.

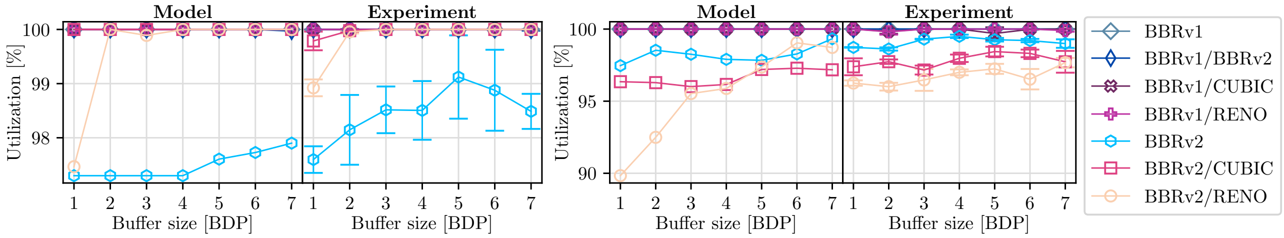
4.3.4 Utilization. The fluid model captures three important aspects of link utilization (cf. Fig. 9). First, the fluid model correctly predicts that BBRv1 (or combinations with BBRv1) lead to full utilization of the bottleneck link, both under drop-tail and RED. This high utilization by BBR is unsurprising 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



(a) Drop-tail

(b) RED

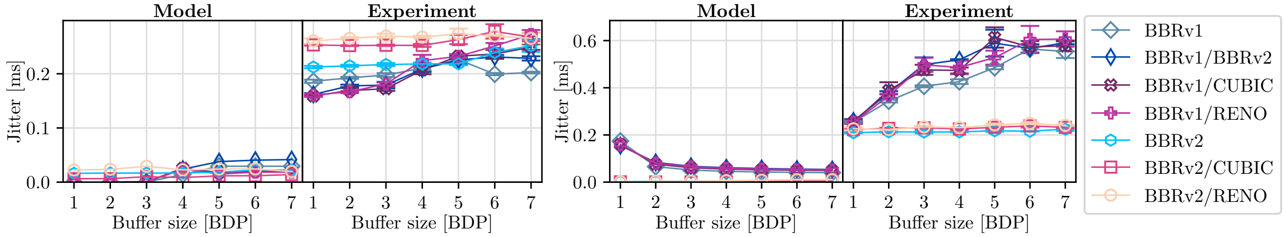
Figure 8: Queuing validation



(a) Drop-tail

(b) RED

Figure 9: Utilization validation



(a) Drop-tail

(b) RED

Figure 10: Jitter validation

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 queueing disciplines.

The only major prediction error of the model is the underestimation of utilization by the BBRv2/Reno combination in shallow RED buffers. This result again points to the idealization of instantly reacting RED queues: The Reno flows in the model back off as soon as the arrival rate exceeds the bottleneck capacity, whereas this back-off is delayed in the experiments. Hence, the arrival-rate evolution moves on a lower level in the model than in the experiment, leading to an underestimation of utilization.

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 $g \cdot N/C_\ell$ seconds, where g is a given packet size.

However, as 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. Unless given directly, 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 a high-level model 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 flow discover their propagation delay and generally achieves this goal. Hence, we assume that $\tau_i^{\min} = \sum_{\ell \in \pi_i} d_\ell =: d_i$. Since this minimum-RTT measurement is present in the congestion-window size $w_i^{\text{pbw}} = 2\tau_i^{\min} x_i^{\text{btl}}$, also w_i^{pbw} can be simplified to $2d_i x_i^{\text{btl}}$. 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^{dlv} over a period, which in turn depends on the sending rates of all flows and the queue length at the bottleneck link (cf. Eq. (18)). The sending rate of a sender i is either $\min(w_i^{\text{pbw}}/\tau_i,\ 5/4 x_i^{\text{btl}})$ (if flow i is probing), $\min(w_i^{\text{pbw}}/\tau_i,\ 3/4 x_i^{\text{btl}})$ (if draining), or $\min(w_i^{\text{pbw}}/\tau_i,\ x_i^{\text{btl}})$ (otherwise). Hence, the maximum delivery rate depends on the concurrent behavior of the other flows on the bottleneck link, which may be probing or draining during the measurements of flow i. For many flows, the probing and draining flows can be expected to offset each other such that the total background-traffic volume is similar as if $x_j^{\text{pbw}} = \min(w_j^{\text{pbw}}/\tau_j,\ x_j^{\text{btl}})$ for all flows $j \in U_{\ell_i}, j \neq i$. Given this background traffic, a sender i measures the maximum delivery rate in the probing phase. Hence, the maximum measurement is determined as follows:

$$x_i^{\max} = \begin{cases} \frac{\min(5/4,\ \Delta_i) \cdot x_i^{\text{btl}} \cdot C_{\ell_i}}{\min(5/4,\ \Delta_i) \cdot x_i^{\text{btl}} + \sum_{j \neq i} \min(1,\ \Delta_j) \cdot x_j^{\text{btl}}} & \text{if } q_{\ell_i} > 0 \\ \min(5/4, \Delta_i) \cdot x_i^{\text{btl}} & \text{otherwise} \end{cases} \tag{37}$$

where $\Delta_i = 2d_i/(d_i + \sum_{\ell \in \pi_i} \frac{q_\ell}{C_\ell})$, and ℓ_i is flow i's bottleneck link.

The final simplification step concerns the adaptation of the bottleneck-bandwidth estimate x_i^{btl}. Over a long duration, this regular update can be approximated by a continuous assimilation:

$$\dot{x}_i^{\text{btl}} = x_i^{\max} - x_i^{\text{btl}} \tag{38}$$

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 flow i differs from the bottleneck-bandwidth estimate x_i^{btl}, or if the queue length q_ℓ of some link ℓ grows; both events may lead to subsequent rate changes. To formalize this condition, we henceforth consider N senders which share a single bottleneck link ℓ^*. Moreover, we first assume that buffer capacities do not constrain the dynamics, and modify this assumption later.

Definition 1. *N BBRv1 senders sharing a bottleneck link ℓ^* are in equilibrium if and only if $\{x_i^{\text{btl}}\}_{i \in U_{\ell^*}}$ and q_{ℓ^*} satisfy:*

$$\sum_{i \in U_{\ell^*}} \min(1,\ \Delta_i) \cdot x_i^{\text{btl}} = C_{\ell^*} \quad \forall i \in U_{\ell^*}.\ x_i^{\text{btl}} = x_i^{\max} \tag{39}$$

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

Theorem 1. *N BBRv1 senders sharing a bottleneck link ℓ^* are in equilibrium if and only if propagation delay equals queuing delay for every sender, i.e.,*

$$\forall i \in U_{\ell^*}.\ d_i = \sum_{\ell \in \pi_i} \frac{q_\ell}{C_\ell}.$$

Interestingly, Theorem 1 suggests that the equilibria of BBRv1 in single-bottleneck scenarios (with non-limiting buffers) can be arbitrarily unfair as long as $\sum_{i \in U_{\ell^*}} x_i^{\text{btl}} = C_{\ell^*}$. 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 ℓ^*. 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 ℓ^* are zero, which is a scenario frequently investigated in the literature [1, 25, 63]. In this case, we can prove asymptotic stability of BBRv1 with the indirect Lyapunov method, meanining 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.*

Proof. In the scenario under consideration, it holds that $q_\ell = 0\ \forall \ell \neq \ell^*$. Given Theorem 1, it thus holds that the equilibrium is valid only for equal RTTs:

$$\forall i \in U_{\ell^*}.\ d_i = \frac{q_{\ell^*}}{C_{\ell^*}} =: d. \tag{40}$$

Hence, we can simplify: $\Delta_i = \Delta(q_{\ell^*}) := 2d/(d + q_{\ell^*}/C_{\ell^*})$. As a result, the equilibrium requires that $\Delta(q_{\ell^*}) = 1 \iff q_{\ell^*} = dC_{\ell^*}$.

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

$$\dot{x}_i^{\text{btl}} = \frac{\Delta(q_{\ell^*}) x_i^{\text{btl}} C_{\ell^*}}{\Delta(q_{\ell^*}) \sum_{k \in U_{\ell^*}} x_k^{\text{btl}}} - x_i^{\text{btl}} = \frac{x_i^{\text{btl}} C_{\ell^*}}{\sum_{k \in U_{\ell^*}} x_k^{\text{btl}}} - x_i^{\text{btl}} \tag{41}$$

Moreover, the dynamics of the bottleneck-link queue q_{ℓ^*} are:

$$\dot{q}_{\ell^*} = y_{\ell^*} - C_{\ell^*} = \Delta(q_{\ell^*}) \sum_{i \in U_{\ell^*}} x_i^{\text{btl}} - C_{\ell^*} \tag{42}$$

where y_{ℓ^*} is the arrival rate at bottleneck link ℓ^*. Based on these dynamics, we derive the dynamics of the arrival rate y_{ℓ^*}:

$$\begin{aligned} \dot{y}_{\ell^*} &= \dot{\Delta}(q_{\ell^*}) \sum_{i \in U_{\ell^*}} x_i^{\text{btl}} + \Delta(q_{\ell^*}) \sum_{i \in U_{\ell^*}} \dot{x}_i^{\text{btl}} \\ &= -\frac{1}{C_{\ell^*}(d + \frac{q_{\ell^*}}{C_{\ell^*}})} y_{\ell^*}^2 + (\frac{1}{d + \frac{q_{\ell^*}}{C_{\ell^*}}} - 1) y_{\ell^*} + \Delta(q_{\ell^*}) C_{\ell^*} \end{aligned} \tag{43}$$

Building on this formalization, we can define a classic non-linear dynamic system with y_{ℓ^*} and q_{ℓ^*} as state variables, and $\dot{y}_{\ell^*}$ and $\dot{q}_{\ell^*}$ 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 $\mathbf{J}_f \in \mathbb{R}^{2 \times 2}$ has the following entries:

$$\frac{\partial \dot{y}_{\ell^*}}{\partial y_{\ell^*}} = -\frac{2}{C_{\ell^*}(d + \frac{q_{\ell^*}}{C_{\ell^*}})} y_{\ell^*} + \frac{1}{d + \frac{q_{\ell^*}}{C_{\ell^*}}} - 1 \tag{44}$$

$$\frac{\partial \dot{y}_{\ell^*}}{\partial q_{\ell^*}} = \frac{y_{\ell^*}^2}{C_{\ell^*}^2 (d + \frac{q_{\ell^*}}{C_{\ell^*}})^2} - \frac{y_{\ell^*}}{C_{\ell^*}(d + \frac{q_{\ell^*}}{C_{\ell^*}})^2} - \frac{2d}{(d + \frac{q_{\ell^*}}{C_{\ell^*}})^2} \tag{45}$$

$$\frac{\partial \dot{q}_{\ell^*}}{\partial y_{\ell^*}} = 1 \qquad \frac{\partial \dot{q}_{\ell^*}}{\partial q_{\ell^*}} = 0 \tag{46}$$

Evaluating this Jacobian matrix at the equilibrium, i.e., $y_{\ell^*} = C_{\ell^*}$ and $q_{\ell^*} = dC_{\ell^*}$, yields a matrix for which the maximum eigenvalue λ^+ can be found via the characteristic equation:

$$\begin{aligned} \mathbf{J}_f(C_{\ell^*},\ dC_{\ell^*}) &= \begin{pmatrix} -\frac{1}{2d} - 1 & -\frac{1}{2d} \\ 1 & 0 \end{pmatrix} \\ \Longrightarrow \quad \lambda^+ &= -(\frac{1}{4d} + \frac{1}{2}) + \frac{1}{2d}(\pm(d - \frac{1}{2})) \end{aligned} \tag{47}$$

Performing a case distinction on d confirms that the maximum eigenvalue λ^+ is always negative:

$$\begin{aligned} d \leq \frac{1}{2}: \quad \lambda^+ &= \frac{-(\frac{1}{2d} + 1) - \frac{1}{d}(d - \frac{1}{2})}{2} = -1 < 0 \\ d > \frac{1}{2}: \quad \lambda^+ &= \frac{-(\frac{1}{2d} + 1) + \frac{1}{d}(d - \frac{1}{2})}{2} = -\frac{1}{2d} < 0 \end{aligned} \tag{48}$$

Hence, we observe that the Jacobian matrix $\mathbf{J}_f$ has consistently negative eigenvalues, which by the indirect Lyapunov method proves the asymptotic stability of the dynamics. □

As the proof of Theorem 2 makes clear, the BBRv1 equilibrium from Theorem 1 is only viable if the bottleneck-link buffer capacity B_{ℓ^*} permits the equilibrium queue length $q_{\ell^*} = dC_{\ell^*}$. Intuitively, the equilibrium is valid for a bottleneck buffer that is large enough for the congestion-window constraint Δ_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_{ℓ^*} is restricted by the buffer size B_{ℓ^*} such that the congestion-window limit has no effect for any flow i, i.e., $\Delta_i \geq 5/4$ for all $i \in U_{\ell^*}$ (cf. Eq. (37)). With this assumption, we find a different equilibrium for BBRv1:

Theorem 3. *N BBRv1 senders sharing a bottleneck link ℓ^* that has a shallow buffer (i.e. $\Delta_i > 5/4\ \forall i \in U_{\ell^*}$) are in equilibrium if and only if each flow i has the following bottleneck-bandwidth estimate x_i^{btl}:*

$$x_i^{\text{btl}} = \frac{5C_{\ell^*}}{4N + 1}.$$

This equilibrium is perfectly fair and asymptotically stable.

Theorem 3 thus implies that without an effective congestion-window limit, the aggregate rate y_{ℓ^*} in equilibrium consistently exceeds the link capacity C_{ℓ^*}, 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 $N \longrightarrow \infty$). 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 must converge to perfect fairness in shallow buffers, whereas such fairness is only possible, but not required in deep buffers (cf. Theorem 1).

5.2 BBRv2 Stability Analysis

This section again 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 $5/4 \cdot x_i^{\text{btl}}$ and the inflight volume to $5/4 \cdot \overline{w}_i$, except the loss exceeds 2%. Since we limit our analysis to networks with buffers large enough to prevent loss, the traffic-pulse rate is:

$$x_i^{\text{pls}} = 5/4 \cdot \min(1,\ \delta_i) \cdot x_i^{\text{btl}},\ \delta_i = \frac{d_i}{d_i + \sum_{\ell \in \pi_i} q_\ell / C_\ell}. \tag{49}$$

In addition to this pulse rate, the background traffic co-determines the maximum delivery rate $x_i^{\max}$. 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 flow i sets its pacing rate to x_i^{btl} and keeps its inflight volume at the minimum of the estimated BDP $\overline{w}_i$ and 85% of the upper inflight bound `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 $5/4 \cdot \overline{w}_i$. Since the estimated BDP $\overline{w}_i$ is consistently smaller than $0.85 \cdot 5/4 \cdot \overline{w}_i$, the sending rate in cruising mode is:

$$x_i = \min(1, \delta_i) \cdot x_i^{\text{btl}} \tag{50}$$

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

$$x_i^{\max} = \begin{cases} \frac{5/4 \cdot \min(1,\ \delta_i) \cdot x_i^{\text{btl}} \cdot C_{\ell_i}}{5/4 \cdot \min(1,\ \delta_i) \cdot x_i^{\text{btl}} + \sum_{j \neq i} \min(1,\ \delta_j) \cdot x_j^{\text{btl}}} & \text{if } q_{\ell_i} > 0 \\ 5/4 \cdot \min(1,\ \delta_i) \cdot x_i^{\text{btl}} & \text{otherwise} \end{cases} \tag{51}$$

5.2.2 Stability Analysis. For BBRv2, the equilibrium conditions match the equilibrium conditions for BBRv1 (cf. Definition 1), with Δ_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 ℓ^* are in a perfectly fair equilibrium if propagation delay and queuing delay for each flow have the following relation:*

$$\forall i \in U_{\ell^*}.\ \frac{N-1}{4N+1} \cdot d_i = \sum_{\ell \in \pi_i} \frac{q_\ell}{C_\ell}$$

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-dependency 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 only the bottleneck link has a non-empty queue. For our stability analysis, we thus again focus on a scenario where the queue lengths on all involved links except the bottleneck link ℓ^* 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.*

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

5.3 Summary of Theoretical Results

To provide a practical interpretation of our theoretical findings, we summarize the key takeaways from the preceding sections below.

BBRv1 in deep buffers. In deep buffers, BBRv1 converges to rate distributions with no fairness guarantees (cf. Theorems 1 and 2). Indeed, previous research has already demonstrated that unfairness can arise among BBRv1 flows with strongly different RTTs, with longer-RTT flows obtaining higher bandwidth shares [24, 58]. However, we find that RTT diversity is not necessary for unfairness: Even flows with equal RTTs can obtain different bandwidth shares as long as the propagation delay of each flow equals the cumulative queuing delay on the used path. Hence, if flows with equal RTT d share a bottleneck link ℓ^* with capacity C_{ℓ^*} and only encounter a queue at that bottleneck link, the steady-state queue length of BBRv1 is the product of path propagation delay and bottleneck capacity, i.e., $d \cdot C_{\ell^*}$.

BBRv1 in shallow buffers. BBRv1 converges to different steady states (i.e., equilibria) in shallow buffers than in deep buffers (cf. Theorem 3). In particular, the steady states that are attained given shallow buffers involve perfectly fair rate distributions. However, the aggregate equilibrium rate for shallow buffers necessarily exceeds the bottleneck capacity for more than one concurrent flow; hence, BBRv1 causes permanently full buffers and enduring loss (of up to 20% for a high number of flows) in shallow buffers.

BBRv2 improvements. Our analysis illustrates the improvements of BBRv2 over BBRv1 for the scenario where all flows have equal propagation delay and only encounter a queue at the bottleneck link. In that case, BBRv2 necessarily converges to a steady state (cf. Theorems 4 and 5), which is preferable to the BBRv1 steady state in two respects. First, the steady state that is attained by BBRv2 necessarily involves a perfectly equitable rate distribution (unlike the steady states of BBRv1, which can be arbitrarily unfair). Second, the steady state involves a bottleneck queue length which is shorter than the BBRv1 equilibrium queue length by at least 75% and is even 0 for a single sender. Our theoretical analysis thus confirms that BBRv2 improves upon BBRv1 in the essential aspects of fairness and buffer utilization.

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

In the following, we will distinguish previously known insights that were confirmed by our fluid model, and novel insights that our fluid model disclosed. For this distinction, we will indicate the type of insight by **(P)** (for previous) and **(N)** (for novel).

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.*** *BBRv1 causes considerable loss of up to 20% of traffic under drop-tail, while the loss-sensitive CCAs Reno, CUBIC, and BBRv2 cause loss rates of around 1%* ***(P)****. The same behavior is observed for a RED queuing discipline* ***(N)****.*

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 [51, 58]:

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)* ***(P)*** *or buffers of any size (given a RED queuing discipline)* ***(N)****. In large drop-tail buffers, the congestion window of BBRv1 becomes effective, leading to improvements in fairness towards loss-sensitive CCAs* ***(P)****.*

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 discipine* ***(P)****.*

Many 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* ***(P)****.*

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* `inflight_hi` *estimate in the start-up phase* ***(N)****.*

Insight 6. ***BBRv2 in RED Buffers.*** *When competing with loss-based CCAs (Reno and CUBIC) 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* ***(N)****.*

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:

Insight 9. ***Sources of Inaccuracy.*** *Inaccurate predictions by fluid models can result from at least three sources:*

- *idealizations, e.g., assuming instantly reacting RED queues (cf. §4.3.2);*
- *difficulty of capturing discrete phenomena, e.g., jitter (cf. §4.3.5);*
- *and negligence of the start-up phase, e.g., BBRv2 has to be simulated with varying initial conditions to find issues arising from the start-up phase (cf. §4.3.3).*

If the developers of CCAs are aware of the above pitfalls of fluid models, they can interpret the fluid-model results in the context of these caveats.

7 RELATED WORK

While our focus on this paper is on congestion-control algorithms (CCAs), we note that there exist several orthogonal approaches to deal with congestion, for example, related to buffer management [4, 12, 13], scheduling [3, 26, 43, 44], and bandwidth reservation [7, 8].

Since the seminal work by Jacobson [28], a wide range of CCAs have been proposed and analyzed [6, 25, 27, 29, 30, 45]. While traditional CCAs are based on loss (timeout) signals, more recent protocols leverage explicit congestion notification (ECN) [2, 55, 62] or delay [1, 23, 32, 35, 40] to react in a more informed and fine-grained manner. With BBR [9], recently another flavor of CC has been introduced, which is often referred to as model-based.

Fluid models (also known as differential-equation models) provide a particularly powerful framework for an analytical understanding of CC protocols and their equilibria [53], and have been widely used in the literature [18, 36, 37, 39, 48, 56]. These models are attractive for their flexibility (e.g., supporting different topologies and queuing disciplines), and for allowing fast initial analyses. In general, the models come in different flavors and can for example be analyzed using dynamical-systems techniques [47]. In one prominent work [39], 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, 51, 54]. In particular, Scholz et al. [51] show that BBRv1 flows are robustly able to claim a disproportionate share of the bandwidth. Ware et al. [58] 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. [60] 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, 41, 52], 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 [61] 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.

8 CONCLUSION

In this paper, we take a deep dive into the recent CCA proposals of BBRv1 and BBRv2 by complementing previous analyses with an approach based on *fluid models*. Fluid models are a classic but lately seldom employed approach to evaluating CCA properties, and are unique in their ability to allow both theoretical stability analysis and efficient simulation for a wide range of network scenarios. We devise such a fluid model for both BBR versions by using new modelling techniques such as sigmoid pulses and mode variables, and perform an experiment-based validation to show that the model is highly predictive regarding performance and fairness properties. We further leverage the model for both an extensive simulation and a theoretical stability analysis. This investigation confirms previously found issues in BBRv1, but also yields new insights, e.g., regarding the structure and asymptotic stability of BBR equilibria, as well as regarding bufferbloat and inter-CCA unfairness in BBRv2.

While our model is accurate and general, we understand our analysis as a first step in exploring the investigation opportunities that our fluid model opens up. Indeed, it will be interesting to evaluate the BBR fluid models in multiple-bottleneck scenarios, both through simulations and further theoretical analysis.

ACKNOWLEDGMENTS

We gratefully acknowledge support from ETH Zurich, from SNSF for project ESCALATE (200021L_182005), and from the German Federal Ministry of Education and Research (BMBF) for project '6G-RIC' (grant 16KISK020K, 2021-2025). Moreover, we thank Alan Zaoxing Liu for shepherding, the anonymous reviewers and Joel Wanner for valuable feedback, and James Dermelj for his contributions to the experiment code.

REFERENCES

[1] Vamsi Addanki, Oliver Michel, and Stefan Schmid. 2022. PowerTCP: Pushing the Performance Limits of Datacenter Networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 51–70.

[2] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 conference*. 63–74.

[3] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pfabric: Minimal near-optimal datacenter transport. *ACM SIGCOMM Computer Communication Review (CCR)* 43, 4 (2013), 435–446.

[4] Maria Apostolaki, Laurent Vanbever, and Manya Ghobadi. 2019. FAB: Toward flow-aware buffer sharing on programmable switches. In *Proceedings of the Workshop on Buffer Sizing*. 1–6.

[5] Venkat Arun, Mina Arashloo, Ahmed Saeed, Mohammad Alizadeh, and Hari Balakrishnan. 2021. Formally Verifying Congestion Control Performance. In *Proceedings of SIGCOMM 2021*.

[6] Wei Bao, Vincent WS Wong, and Victor CM Leung. 2010. A model for steady state throughput of TCP CUBIC. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 1–6.

[7] Cristina Basescu, Raphael M. Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. 2016. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*.

[8] Lloyd Brown, Ganesh Ananthanarayanan, Ethan Katz-Bassett, Arvind Krishnamurthy, Sylvia Ratnasamy, Michael Schapira, and Scott Shenker. 2020. On the Future of Congestion Control for the Public Internet. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 30–37.

[9] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control: Measuring Bottleneck Bandwidth and Round-Trip Propagation Time. *Queue* 14, 5 (Oct. 2016), 20–53. https://doi.org/10.1145/3012426.3022184

[10] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, Ian Swett, Jana Iyengar, Victor Vasiliev, and Van Jacobson. 2017. BBR Congestion Control: IETF 99 Update. In *Presentation in ICCRG at IETF 99*. IETF. https://datatracker.ietf.org/meeting/99/materials/slides-99-iccrg-iccrg-presentation-2-00.pdf

[11] Neal Cardwell, Yuchung Cheng, S Hassas Yeganeh, Ian Swett, Victor Vasiliev, Priyaranjan Jha, Yousuk Seung, Matt Mathis, and Van Jacobson. 2019. BBRv2: A model-based congestion control. In *Presentation in ICCRG at IETF 104th meeting*.

[12] Abhijit K Choudhury and Ellen L Hahne. 1998. Dynamic queue length thresholds for shared-memory packet switches. *IEEE/ACM Transactions on Networking (ToN)* 6, 2 (1998), 130–140.

[13] Cisco. 2021. Nexus 9000 Series Switches. https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-738488.html.

[14] Hamza Dahmouni, André Girard, and Brunilde Sansò. 2012. An analytical model for jitter in IP networks. *annals of telecommunications-annales des télécommunications* 67, 1 (2012), 81–90.

[15] Edward J Daniel, Christopher M White, and Keith A Teague. 2003. An interarrival delay jitter model using multistructure network delay characteristics for packet networks. In *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. IEEE, 1738–1742.

[16] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. {PCC} vivace: Online-learning congestion control. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 343–356.

[17] Thomas Erneux. 2009. *Applied delay differential equations*. Springer Science & Business Media.

[18] Nicky van Foreest, Michel Mandjes, and Werner Scheinhardt. 2003. Analysis of a feedback fluid model for heterogeneous TCP sources. (2003).

[19] Linux Foundation. 2021. OVS: Open vSwitch. https://www.openvswitch.org/.

[20] Jose Gomez, Elie Kfoury, Jorge Crichigno, Elias Bou-Harb, and Gautam Srivastava. 2020. A performance evaluation of TCP BBRv2 alpha. In *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 309–312.

[21] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42, 5 (2008), 64–74. https://doi.org/10.1145/1400097.1400105

[22] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. TCP CUBIC Implementation in Linux Kernel. https://github.com/torvalds/linux/blob/master/net/ipv4/tcp_cubic.c.

[23] David A Hayes and Grenville Armitage. 2011. Revisiting TCP congestion control using delay gradients. In *International Conference on Research in Networking*. Springer, 328–341.

[24] Mario Hock, Roland Bless, and Martina Zitterbart. 2017. Experimental evaluation of BBR congestion control. In *Proceedings of the IEEE Conference on Network Protocols (ICNP)*. IEEE, 1–10.

[25] Christopher V Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. 2001. A control theoretic analysis of RED. In *Proceedings of the Conference of the IEEE Computer and Communications Society (INFOCOM)*, Vol. 3. IEEE, 1510–1519.

[26] Chi-Yao Hong, Matthew Caesar, and P Brighten Godfrey. 2012. Finishing flows quickly with preemptive scheduling. *ACM SIGCOMM Computer Communication Review (CCR)* 42, 4 (2012), 127–138.

[27] Xiaomeng Huang, Chuang Lin, and Fengyuan Ren. 2006. Generalized modeling and stability analysis of highspeed TCP and scalable TCP. *IEICE Transactions on Communications* 89, 2 (2006), 605–608.

[28] V. Jacobson. 1988. Congestion Avoidance and Control. *ACM SIGCOMM Computer Communication Review (CCR)* (1988), 314–329. https://doi.org/10.1145/52324.52356

[29] Cheng Jin, David X Wei, and Steven H Low. 2004. FAST TCP: motivation, architecture, algorithms, performance. In *IEEE INFOCOM 2004*, Vol. 4. IEEE, 2490–2501.

[30] Frank P Kelly, Aman K Maulloo, and David KH Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society* 49, 3 (1998), 237–252.

[31] Elie F Kfoury, Jose Gomez, Jorge Crichigno, and Elias Bou-Harb. 2020. An emulation-based evaluation of TCP BBRv2 alpha for wired broadband. *Computer Communications* 161 (2020), 212–224.

[32] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan MG Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, et al. 2020. Swift: Delay is simple and effective for congestion control in the datacenter. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 514–528.

[33] Bob Lantz, Nikhil Handigol, Brandon Heller, and Vimal Jeyakumar. 2021. Introduction to Mininet. https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[34] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. ACM, Article 19, 6 pages. https://doi.org/10.1145/1868447.1868466

[35] Changhyun Lee, Chunjong Park, Keon Jang, Sue Moon, and Dongsu Han. 2015. Accurate latency-based congestion feedback for datacenters. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 403–415.

[36] Yong Liu, Francesco Lo Presti, Vishal Misra, Don Towsley, and Yu Gu. 2003. Fluid models and solutions for large-scale IP networks. In *Proceedings of the ACM SIGMETRICS Conference*. 91–101.

[37] Steven H Low, Fernando Paganini, and John C Doyle. 2002. Internet congestion control. *IEEE Control Systems Magazine* 22, 1 (2002), 28–43. https://doi.org/10.1109/37.980245

[38] Robert McMahon. 2021. iPerf. https://sourceforge.net/projects/iperf2/.

[39] Vishal Misra, Wei-Bo Gong, and Don Towsley. 2000. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. 151–160.

[40] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based congestion control for the datacenter. *ACM SIGCOMM Computer Communication Review (CCR)* 45, 4 (2015), 537–550.

[41] Aarti Nandagiri, Mohit P Tahiliani, Vishal Misra, and KK Ramakrishnan. 2020. BBRvl vs BBRv2: Examining performance differences through experimental evaluation. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*. IEEE, 1–6.

[42] NumPy Project. 2021. NumPy. https://numpy.org/.

[43] Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. 2017. Flowtune: Flowlet control for datacenter networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 421–435.

[44] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: A centralized" zero-queue" datacenter network. In *Proceedings of the ACM SIGCOMM Conference*. 307–318.

[45] Sudheer Poojary and Vinod Sharma. 2019. An asymptotic approximation for TCP CUBIC. *Queueing Systems* 91, 1 (2019), 171–203.

[46] Chutiphon Pukdeboon. 2011. A review of fundamentals of Lyapunov theory. *The Journal of Applied Science* 10, 2 (2011), 55–61.

[47] Gaurav Raina. 2006. *Congestion control for the Internet: stability and bifurcation analysis*. Ph.D. Dissertation. University of Cambridge.

[48] Gaurav Raina and Damon Wischik. 2005. Buffer sizes for large multiplexers: TCP queueing theory and instability analysis. In *Next Generation Internet Networks, 2005*. IEEE, 173–180.

[49] Injong Rhee, Lisong Xu, Sangtae Ha, Alexander Zimmermann, Lars Eggert, and Richard Scheffenegger. 2018. *CUBIC for Fast Long-Distance Networks*. RFC 8312. IETF. https://doi.org/10.17487/RFC8312

[50] Simon Scherrer and James Dermelj. 2022. BBR Fluid-Model Simulator and Mininet Experiment Suite. https://github.com/simonschdev/imc22-bbr-fluid-model. Uploaded on 2022-09-01.

[51] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle. 2018. Towards a deeper understanding of TCP BBR congestion control. In *Proceedings of the IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 1–9.

[52] Yeong-Jun Song, Geon-Hwan Kim, Imtiaz Mahmud, Won-Kyeong Seo, and You-Ze Cho. 2021. Understanding of bbrv2: Evaluation and comparison with bbrv1 congestion control algorithm. *IEEE Access* 9 (2021), 37131–37145.

[53] Rayadurgam Srikant. 2004. *The mathematics of Internet congestion control*. Springer Science & Business Media. https://doi.org/10.1007/978-0-8176-8216-3

[54] Belma Turkovic, Fernando A Kuipers, and Steve Uhlig. 2019. Fifty shades of congestion control: A performance and interactions evaluation. *arXiv preprint arXiv:1903.03852* (2019).

[55] Balajee Vamanan, Jahangir Hasan, and TN Vijaykumar. 2012. Deadline-aware datacenter TCP (d2tcp). *ACM SIGCOMM Computer Communication Review (CCR)* 42, 4 (2012), 115–126.

[56] Gayane Vardoyan, C. V. Hollot, and Don Towsley. 2021. Towards Stability Analysis of Data Transport Mechanisms: A Fluid Model and Its Applications. *IEEE/ACM Transactions on Networking (ToN)* (2021), 1–15. https://doi.org/10.1109/TNET.2021.3075837

[57] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. ACM, 17–24. https://doi.org/10.1145/3365609.3365855

[58] Ranysha Ware, Matthew K Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Modeling BBR's interactions with loss-based congestion control. In *Proceedings of the ACM Internet Measurement Conference (IMC)*. ACM, 137–143. https://doi.org/10.1145/3355369.3355604

[59] D. Wischik. 2006. Queueing theory for TCP. In *Stochastic Networks Conference. University of Illinois at Urbana-Champaign*.

[60] Ming Yang, Peng Yang, Chaozhun Wen, Qiong Liu, Jingjing Luo, and Li Yu. 2019. Adaptive-BBR: Fine-grained congestion control with improved fairness and low latency. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–6.

[61] Doron Zarchy, Radhika Mittal, Michael Schapira, and Scott Shenker. 2019. Axiomatizing congestion control. In *Proceedings of the ACM SIGMETRICS Conference*, Vol. 3. 1–33.

[62] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. *ACM SIGCOMM Computer Communication Review (CCR)* 45, 4 (2015), 523–536.

[63] Yibo Zhu, Monia Ghobadi, Vishal Misra, and Jitendra Padhye. 2016. ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY. In *Proceedings of the International on Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. 313–327.

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 π_i [37]:

$$\begin{aligned}\dot{w}_i = {} & x_i\big(t - d_i^{\mathrm{p}}\big) \cdot \big(1 - p_i\big(t - d_i^{\mathrm{p}}\big)\big) \cdot \frac{1}{w_i} \\ & - x_i\big(t - d_i^{\mathrm{p}}\big) \cdot p_{\pi_i}\big(t - d_i^{\mathrm{p}}\big) \cdot \frac{w_i}{2}\end{aligned} \tag{52}$$

B.2 CUBIC

In contrast, TCP CUBIC cannot directly be described with a differential equation for the congestion-window size. Instead, Vardoyan et al. [56] 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, $w_i^{\max}$:

$$\dot{s}_i = 1 - s_i \cdot x_i\big(t - d_i^{\mathrm{p}}\big) \cdot p_i\big(t - d_i^{\mathrm{p}}\big), \tag{53a}$$

$$\dot{w}_i^{\max} = \big(w_i - w_i^{\max}\big) \cdot x_i\big(t - d_i^{\mathrm{p}}\big) \cdot p_i\big(t - d_i^{\mathrm{p}}\big). \tag{53b}$$

The intuition behind Eq. (53a) is that s_i is increased by 1 in absence of loss ($p_{\pi_i} = 0$) and reduced to 0 when a loss occurs. Equation (53b) describes that $w_i^{\max}(t)$ should be updated to $w_i(t)$ in presence of loss. Knowing s_i and $w_i^{\max}$, the congestion-window size w can be determined by the CUBIC window-growth function [21],

$$w_i = c \cdot \left(s_i - \sqrt[3]{\frac{w_i^{\max} \cdot b}{c}}\right)^3 + w_i^{\max}, \tag{54}$$

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 BBRv1, 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 $q_{\ell^*} = 0$. In that case, the only solution to the conditions in Eq. (39) in terms of $\{\Delta_i\}_{i \in U_{\ell^*}}$ is $\Delta_i = 1 \ \forall i \in U_{\ell^*}$, which shows that the network is in equilibrium for $q_{\ell^*} = 0$ and $d_i = \sum_{\ell \in \pi_i} q_\ell / C_\ell \ \forall i \in U_{\ell^*}$.

For $q_{\ell^*} > 0$, we note that the conditions in Eq. (39) can be transformed into the following conditions for each x_i^{btl}:

$$x_i^{\mathrm{btl}} = \max(1,\ {}^1\!/\!{}_{\Delta_i}) \cdot \big(C_{\ell^*} - \sum_{j \neq i} \min(1,\ \Delta_j) \cdot x_j^{\mathrm{btl}}\big) \tag{55}$$

$$x_i^{\mathrm{btl}} = C_{\ell^*} - \max({}^4\!/\!{}_5,\ {}^1\!/\!{}_{\Delta_i}) \cdot \sum_{j \neq i} \min(1,\ \Delta_j) \cdot x_j^{\mathrm{btl}} \tag{56}$$

Clearly, the previously found solution $\Delta_i = 1 \ \forall i \in U_{\ell^*}$ is also a solution to the conditions in Eqs. (55) and (56). Hence, we have proven that the network is in equilibrium if $d_i = \sum_{\ell \in \pi_i} q_\ell / C_\ell \ \forall i \in U_{\ell^*}$.

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 $\exists i \in U_{\ell^*}.\ \Delta_i > 1$. For that agent i, the maximum term in Eq. (55) is exactly 1, and the maximum term in Eq. (56) is smaller than 1, which makes the conditions contradictory and rules out an equilibrium. Conversely, if assuming an equilibrium where $\exists i \in U_{\ell^*}.\ \Delta_i < 1$, the maximum term in Eq. (55) is $1/\Delta_i > 1$, and the maximum term in Eq. (56) is also $1/\Delta_i$, which again leads to contradictory equations. Hence, no equilibria other than the equilibrium with $\forall i \in U_{\ell^*}.\ \Delta_i = 1$ are possible, which concludes the proof.

D.2 Proof of Theorem 3

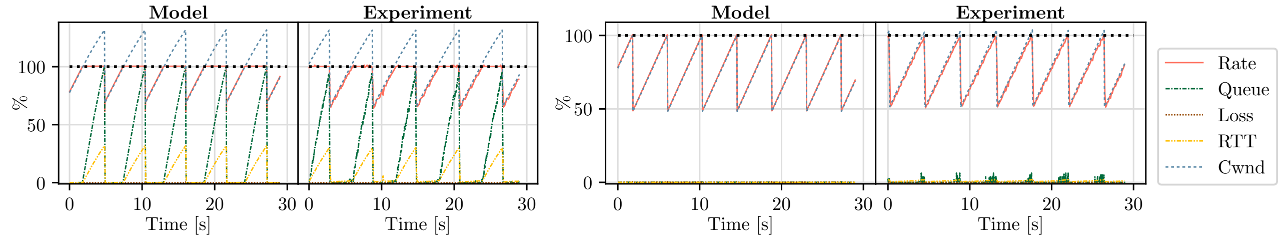
Given $\Delta_i \geq {}^5\!/\!{}_4$ for all $i \in U_{\ell^*}$, the equilibrium condition on $\{x_i^{\mathrm{btl}}\}_{i \in U_{\ell^*}}$ is:

$$\forall i \in U_{\ell^*}.\ x_i^{\mathrm{btl}} = \frac{{}^5\!/\!{}_4 x_i^{\mathrm{btl}} C_{\ell^*}}{{}^5\!/\!{}_4 x_i^{\mathrm{btl}} + \sum_{j \neq i} x_j^{\mathrm{btl}}} = C - {}^4\!/\!{}_5 \sum_{j \neq i} x_j^{\mathrm{btl}} \tag{57}$$

This equation system requires all $\{x_i^{\mathrm{btl}}\}_{i \in U_{\ell^*}}$ to be equal, which allows a straightforward solution:

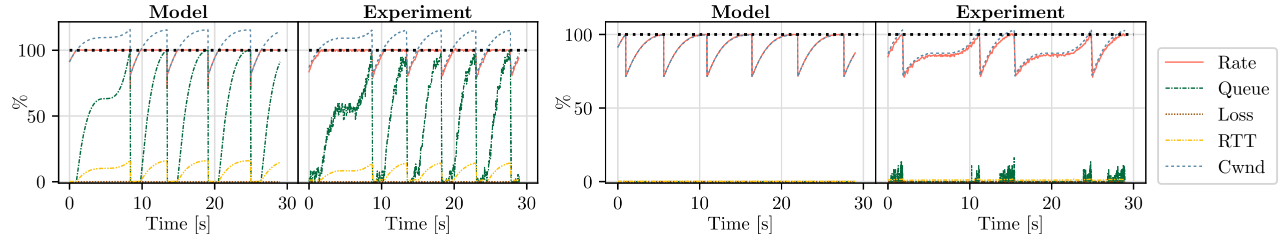
$$\forall i \in U_{\ell^*}.\ x_i^{\mathrm{btl}} = \frac{{}^5\!/\!{}_4 x_i^{\mathrm{btl}} C_{\ell^*}}{(N + {}^1\!/\!{}_4) x_i^{\mathrm{btl}}} = \frac{5 C_{\ell^*}}{4N + 1} \tag{58}$$

It remains to show that this equilibrium is asymptotically stable, for which we employ the indirect Lyapunov method. We apply this



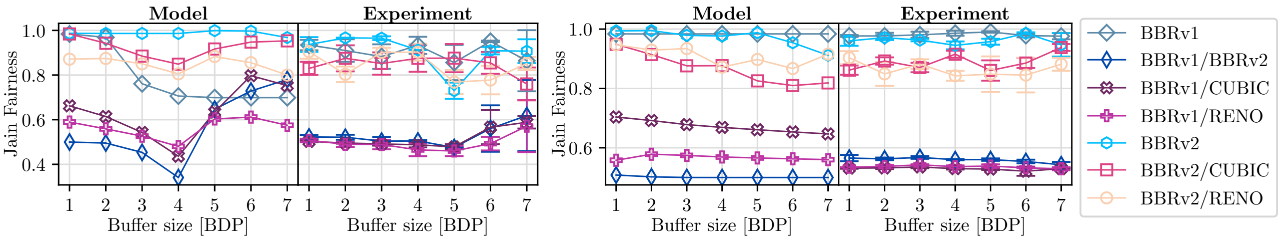
(a) Drop-tail (b) RED

Figure 11: Reno trace validation



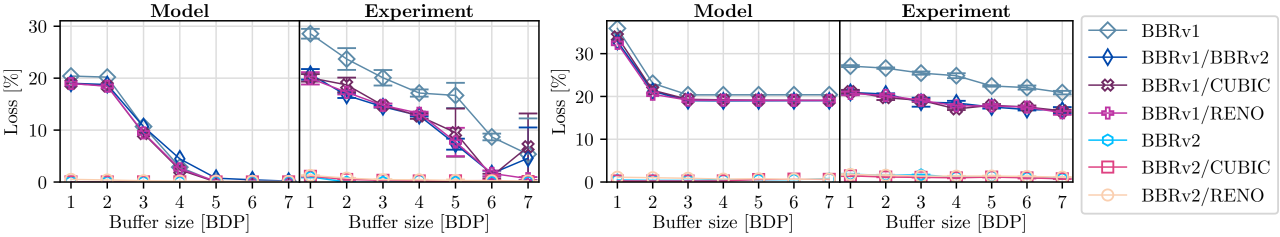
(a) Drop-tail (b) RED

Figure 12: CUBIC trace validation



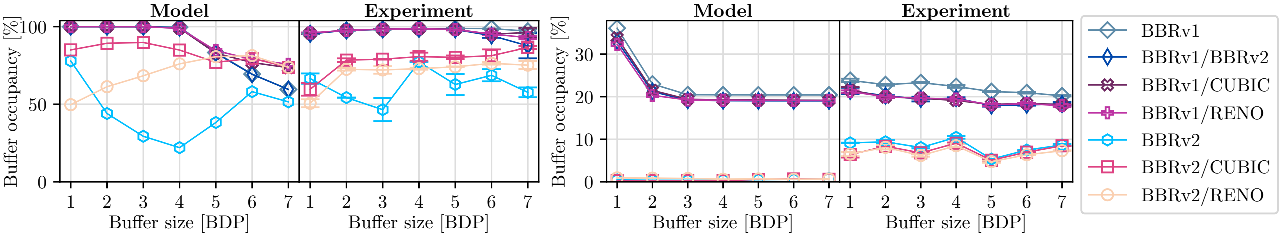
(a) Drop-tail (b) RED

Figure 13: Fairness validation



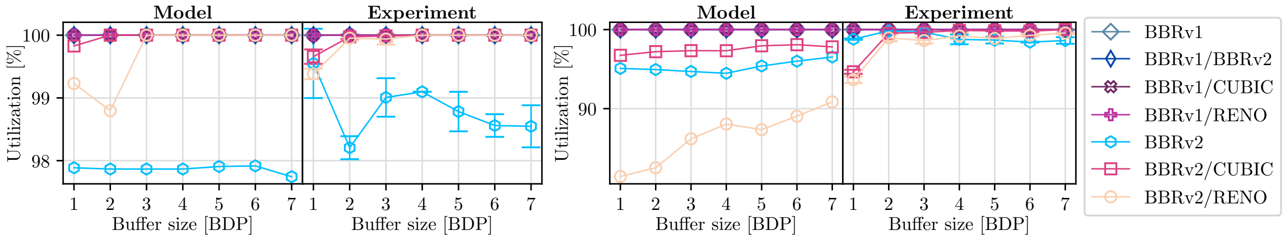
(a) Drop-tail (b) RED

Figure 14: Loss validation



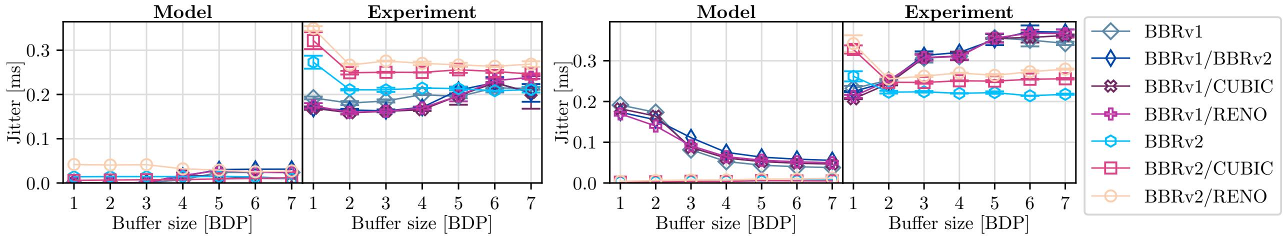
(a) Drop-tail (b) RED

Figure 15: Queuing validation



(a) Drop-tail **(b) RED**

Figure 16: Utilization validation



(a) Drop-tail **(b) RED**

Figure 17: Jitter validation

method to a non-linear dynamic process with $\{x_i^{\text{btl}}\}_{i\in U_{\ell^*}}$ as state variables and $\{\dot{x}_i^{\text{btl}}\}_{i\in U_{\ell^*}}$ as vector-valued evolution function f. The Jacobian matrix $\mathbf{J}_f$ has the following entries, which we evaluate at the equilibrium:

$$\frac{\partial \dot{x}_i}{\partial x_i} = \frac{5/4C\sum_{j\neq i} x_j^{\text{btl}}}{(5/4x_i + \sum_{j\neq i} x_j)^2} - 1 \overset{Eq.}{=} -\frac{5}{4N+1} =: J_{ii} \tag{59}$$

$$\frac{\partial \dot{x}_i}{\partial x_j} = -\frac{5/4Cx_i}{(5/4x_i + \sum_{j\neq i} x_j)^2} \overset{Eq.}{=} -\frac{4}{4N+1} =: J_{ij} \tag{60}$$

$$\tag{61}$$

The eigenpairs $(\lambda, \mathbf{v})$ of $\mathbf{J}_f$ at the equilibrium satisfy the following conditions:

$$\forall i \in U_{\ell^*}.\ (J_{ii} - \lambda)v_i + J_{ij}\sum_{j\neq i} v_j = 0 \tag{62}$$

The first type of solution for this equation system is given by $\lambda = J_{ii} - J_{ij} < 0$ and every $\mathbf{v}$ with $\|\mathbf{v}\|_1 = 0$. The second type of solution is found by assuming $\lambda \neq J_{ii} - J_{ij}$, which implies equal v_i $\forall i \in U_{\ell^*}$ 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.3 Proof of Theorem 4

The equilibrium conditions can be translated into the following constraints given $q_{\ell^*} > 0$:

$$x_i^{\text{btl}} = \max(1,\ 1/\delta_i)\cdot(C_{\ell^*} - \sum_{j\neq i}\min(1,\ \delta_j)\cdot x_j^{\text{btl}}) \tag{63}$$

$$x_i^{\text{btl}} = C_{\ell^*} - 4/5\cdot\max(1,\ 1/\delta_i)\cdot\sum_{j\neq i}\min(1,\ \delta_j)\cdot x_j^{\text{btl}} \tag{64}$$

While these constraints potentially admit multiple equilibria, the equilibrium from Theorem 4 is a special equilibrium for which δ_i is equal across all $i \in U_{\ell^*}$, i.e., $\delta_i = \delta$. Substituting δ for all δ_i, and equating Eq. (63) with Eq. (64), which first yields:

$$\begin{aligned} &\forall i \in U_{\ell^*}.\ \sum_{j\neq i} x_j^{\text{btl}} = 5\cdot(\max(1,\ 1/\delta) - 1)\cdot C \\ \Longrightarrow\quad &\forall i \in U_{\ell^*}.\ x_i^{\text{btl}} = \max(1,\ 1/\delta)\cdot\frac{C}{N}, \end{aligned} \tag{65}$$

where the equation systems requires that all x_i^{btl} $\forall i \in U_{\ell^*}$ equal the same value (hence perfect fairness), where this value can be found using Eq. (63). By inserting x_i^{btl} from Eq. (65) into Eq. (64), 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 $\delta = \frac{4N+1}{5N}$, which is equivalent to the condition in Theorem 4. This insight concludes the proof.

D.4 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 $q_\ell = 0\ \forall \ell \neq \ell^*$. Hence, we can simplify: $\delta_i = \delta(q_{\ell^*}) := d/(d + q_{\ell^*}/C_{\ell^*})$, where d is the propagation delay experienced by all agents. As a result, the equilibrium requires that $\delta(q_{\ell^*}) = \frac{4N+1}{5N} \iff q_{\ell^*} = \frac{N-1}{4N+1}dC_{\ell^*}$.

We translate the reduced model from §5.2.1 into a nonlinear dynamic process with the sending rates $\{x_i\}_{i\in U_{\ell^*}}$ and the queue length q_{ℓ^*} as state variables. The evolution of these state variables is given by vector-valued function f with the following entries:

$$\begin{aligned} \dot{x}_i &= \dot{\delta}(q_{\ell^*})x_i^{\text{btl}} + \delta(q_{\ell^*})\dot{x}_i^{\text{btl}} \\ &= \left(\frac{C_{\ell^*} - \sum_{k\in U_{\ell^*}} x_k}{C_{\ell^*}(d + q_{\ell^*}/C_{\ell^*})} + \frac{5/4\delta C}{5/4x_i + \sum_{j\neq i} x_j} - 1\right)\cdot x_i \end{aligned} \tag{66}$$

$$\dot{q}_{\ell^*} = \sum_{i\in U_{\ell^*}} x_i - C_{\ell^*} \tag{67}$$

The corresponding Jacobian matrix $\mathbf{J}_f$ is composed of the following entries:

$$\frac{\partial \dot{x}_i}{\partial x_i} = \frac{C_{\ell^*} - 2x_i - \sum_{j \neq i} x_j}{C_{\ell^*}(d + q_{\ell^*}/C_{\ell^*})} + \frac{5/4 \delta C \sum_{j \neq i} x_j^{\text{btl}}}{(5/4 x_i + \sum_{j \neq i} x_j)^2} - 1 \tag{68}$$

$$\frac{\partial \dot{x}_i}{\partial x_j} = -\frac{x_i}{C_{\ell^*}(d + q_{\ell^*}/C_{\ell^*})} - \frac{5/4 \delta C x_i}{(5/4 x_i + \sum_{j \neq i} x_j)^2} \tag{69}$$

$$\frac{\partial \dot{x}_i}{\partial q} = \frac{1}{d + \frac{q_{\ell^*}}{C_{\ell^*}}} \left(\frac{C_{\ell^*} - \sum_{k \in U_{\ell^*}} x_k}{C_{\ell^*}^2 \left(d + \frac{q^{\ell^*}}{C_{\ell^*}}\right)} - \frac{5/4 \delta C}{5/4 x_i + \sum_{j \neq i} x_j} \right) x_i \tag{70}$$

$$\frac{\partial \dot{q}}{\partial x_i} = 1 \quad \frac{\partial \dot{q}}{\partial q} = 0 \tag{71}$$

Evaluating the Jacobian matrix at the equilibrium point from Theorem 4 yields the following matrix $\mathbf{J}$:

$$\frac{\partial \dot{x}_i}{\partial x_i} = -\frac{4N+1}{5N^2 d} - \frac{5}{4N+1} =: J_{ii} \qquad \frac{\partial \dot{q}}{\partial x_i} = 1 \tag{72}$$

$$\frac{\partial \dot{x}_i}{\partial x_i} = -\frac{4N+1}{5N^2 d} - \frac{4}{4N+1} =: J_{ij} \qquad \frac{\partial \dot{q}}{\partial q} = 0 \tag{73}$$

$$\frac{\partial \dot{x}_i}{\partial q} = -\frac{4N+1}{5N^2 d} =: J_{iq} \tag{74}$$

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 $\mathbf{J}\mathbf{v} = \lambda \mathbf{v}$, the eigenvalue λ must be lower than 0. To verify this property $\mathbf{J}$, we concretize the eigenvalue condition:

$$\forall i \in U_{\ell^*}. \quad J_{ii} v_i + J_{ij} \sum_{j \neq i} v_j + J_{iq} v_q = \lambda v_i \tag{75}$$

$$\sum_{i \in U_{\ell^*}} v_i = \lambda v_q \tag{76}$$

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

$$\forall i \in U_{\ell^*}. \sum_{k \in U_{\ell^*}} v_k = \frac{\lambda}{J_{iq}} \left((\lambda - J_{ii}) v_i - J_{ij} \sum_{j \neq i} v_j \right) \tag{77}$$

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 $\sum_{k \in U_{\ell^*}} v_k$; more importantly for the proof, λ is negative. Second, for $\lambda \neq J_{ii} - J_{ij}$, the values v_i $\forall i \in U_{\ell^*}$ must be equal such that the equation system from Eq. (77) can be collapsed into a single quadratic equation, which yields the maximum eigenvalue λ^+:

$$N \cdot v_i = \frac{\lambda}{J_{iq}} \left((\lambda - J_{ii}) v_i - J_{ij}(N-1) v_i \right) \implies \lambda^+ = -1 \tag{78}$$

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