TCP BBR
A Deep Dive into the Latest IETF Draft

Seyed Pouria Mousavizadeh Tehrani

September 2024

CAPIF3

Whoami

Software Engineer

Technical Manager at Hoopad Cloud

IRNOG PC Member

- Blog: spmzt.net
- Email: info@spmzt.net
- LinkedIn: linkedin.com/in/spmzt
- PGP: 0xC7E57F23C24F542D

10 Gbps with 100ms RTT needs

a packet loss rate below 0.000003%

TCP BBR
Design Overview

High-Level Design Goals

1. The full throughput (or approximate **fair share** thereof) available to a flow
    - Achieved in a fast and scalable manner (using bandwidth in O(log(**BDP**)) time).
    - Achieved with average packet loss rates of up to 1%.
2. Low queue pressure (low queuing delay and low packet loss)
    - Prevent **bufferbloat**

What do we mean by fair share?

In internet infrastructure, each connection is part of a **non-cooperative game**, where every participant seeks to maximize its own bandwidth.

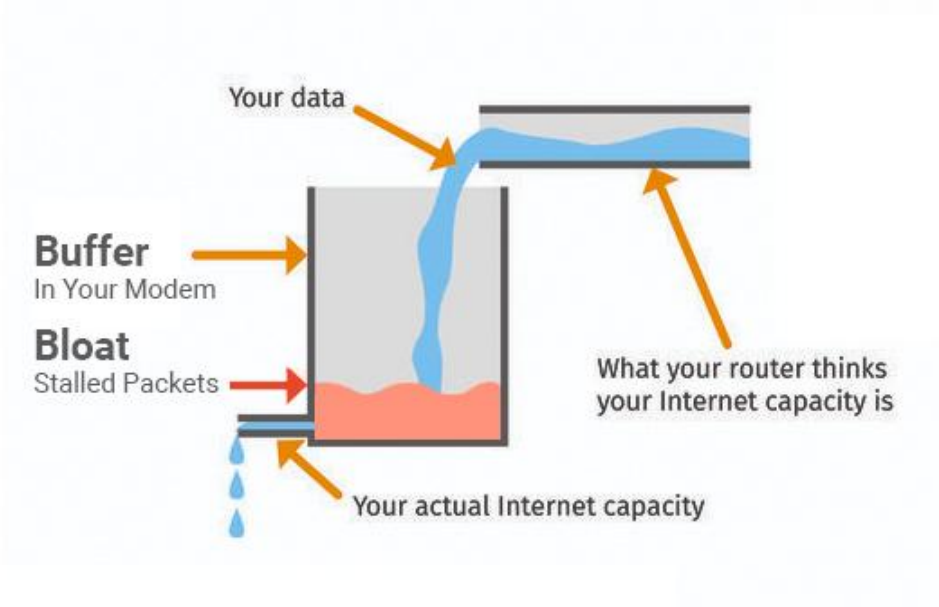
A/B	Wants all the bandwidth	Wants fair share
Wants all the bandwidth	Both suffer from high packet loss and latency (network congestion)	A consumes all the bandwidth, B gets little to none.
Wants fair share	B consumes all the bandwidth, A gets little to none.	Both get an equal share of the bandwidth (efficient and fair use)

What is bufferbloat?

When buffers remain non-empty (“static buffers”), they add delay to every packet passing through the buffer.

Bufferbloat is a cause of:

- high latency
- jitter



What caused the bufferbloat?

- Poor queue management
- Failure of TCP congestion control

TCP
Congestion Control

Design

Those design goals are in tension

sending faster improves the odds of achieving (1) but reduces the odds of achieving (2),

while sending slower improves the odds of achieving (2) but reduces the odds of achieving (1).

Thus the algorithm cannot maximize throughput or minimize queue pressure independently, and must jointly optimize both.

What are the actual physical constraints?

Two physical constraints:

- RTprop (round-trip propagation time)
- BtlBw (bottleneck bandwidth)

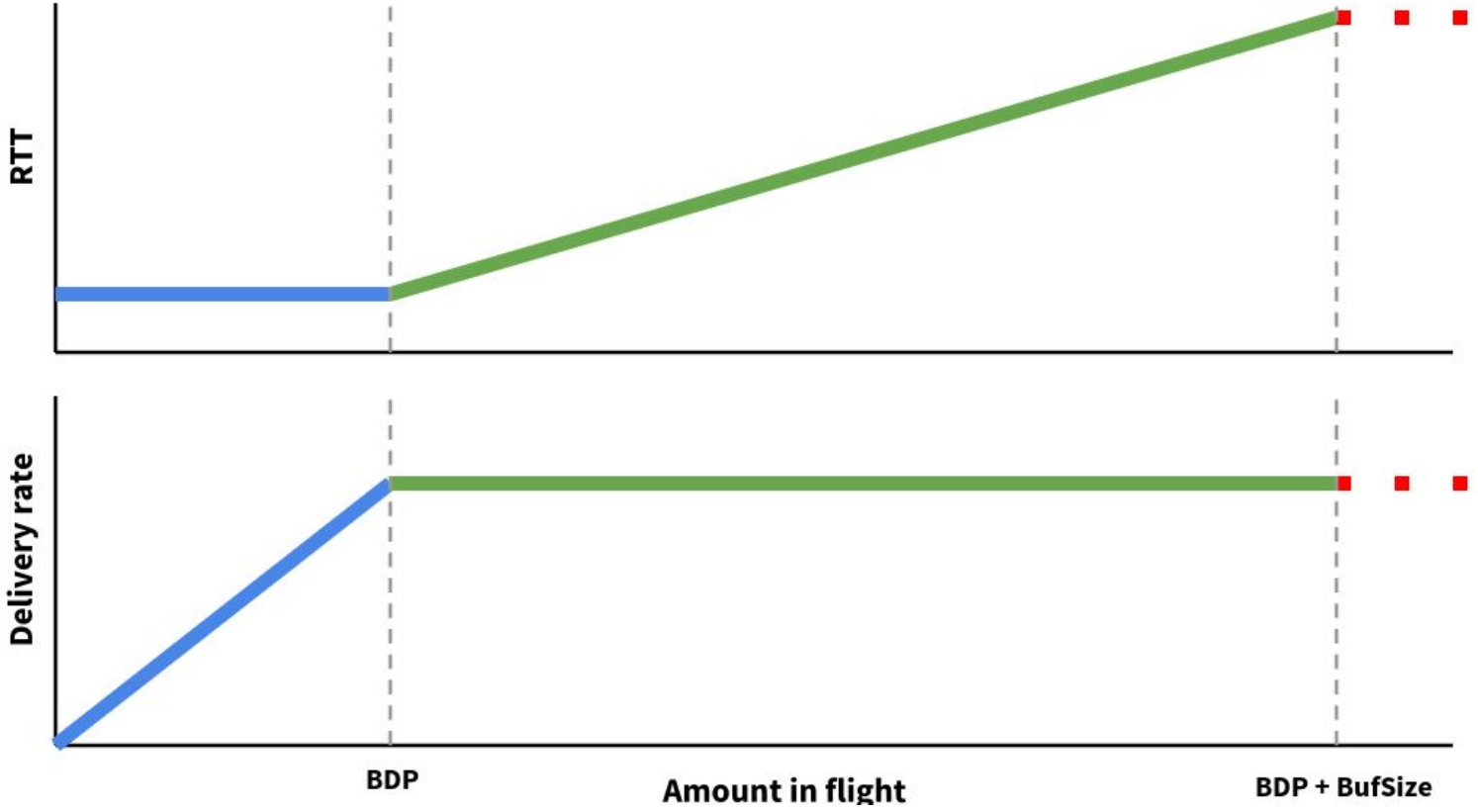
from TCP’s viewpoint an arbitrarily complex path behaves as a **single link** with the same RTT (round-trip time) and bottleneck rate.

When do those constraints apply?

When there isn’t enough data in flight to fill the pipe

- **RTprop** (round-trip propagation time) determines behavior;
- otherwise, **BtlBw** (bottleneck bandwidth) dominates.

How do RTprop and BtlBw looks?



Bandwidth-delay product

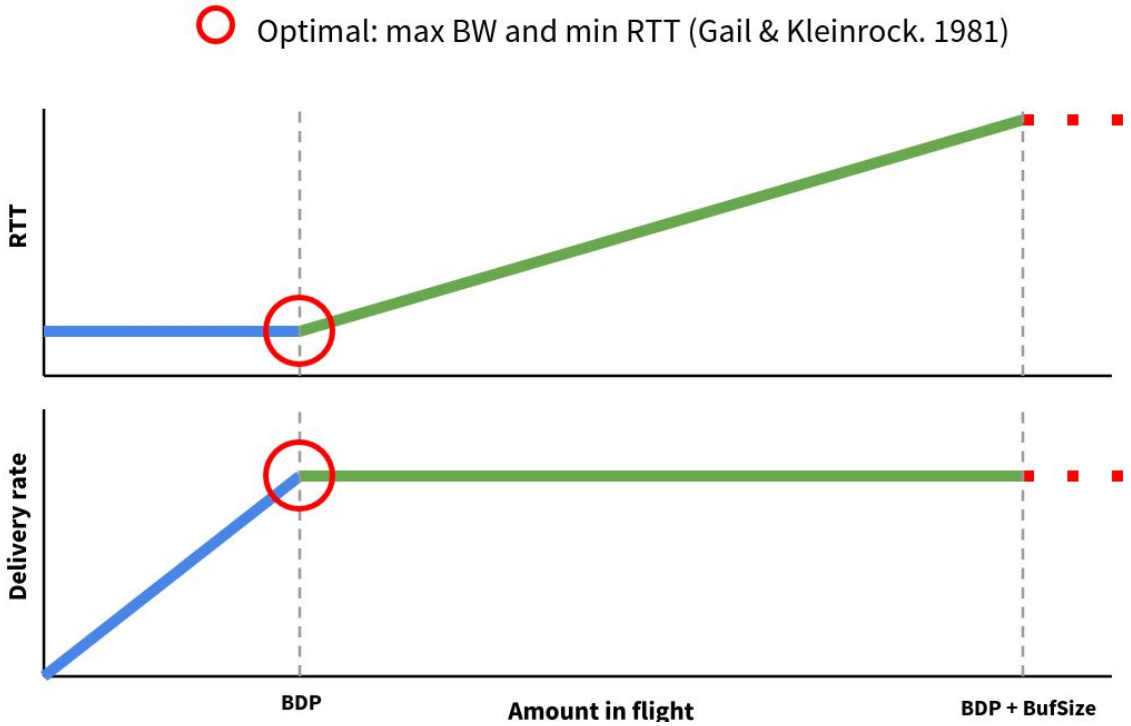
Constraint lines intersect at inflight = BtlBw × RTprop, a.k.a.

the pipe's **BDP** (bandwidth-delay product).

Kleinrock’s optimal operating point

In summary, you want to:

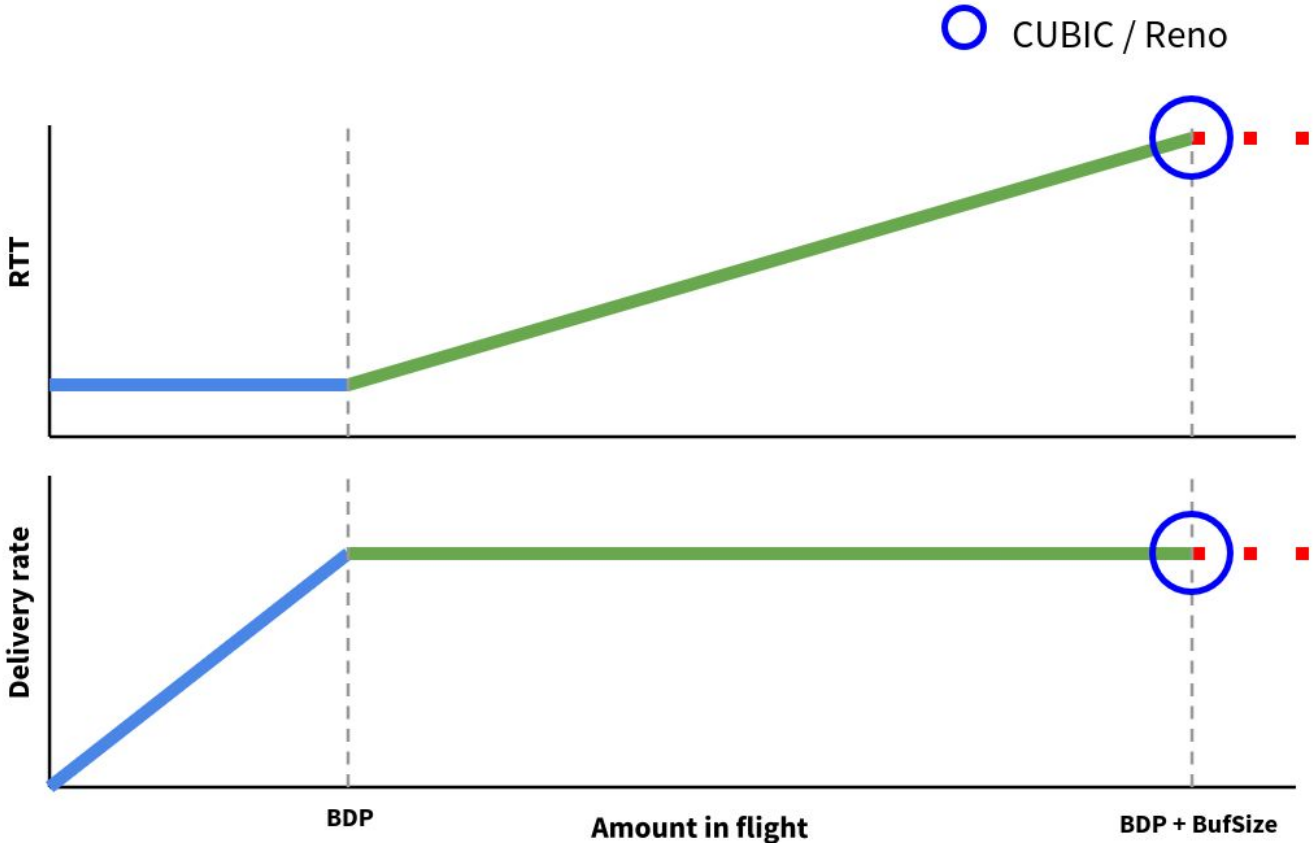
- Maximizing throughput
- Minimizing delay



TCP
Congestion Control

Algorithms

What is wrong with previous TCP CC algorithms?



Problems with loss-based congestion control

Here is the list:

- Packet loss alone is not a good proxy for detecting congestion
- Loss-based CC is overly sensitive to losses that come before congestion
- 10Gbps over 100ms RTT needs <0.000003% packet loss (infeasible)
- 1% loss (feasible) over 100ms RTT gets only 3Mbps
- single-connection HTTP/2 was much slower than multi-conn HTTP/1 on lossy links

Why did loss-based CC work that way?

- When **memory was expensive** buffer sizes were only slightly larger than the BDP, which minimized loss-based congestion control's excess delay.
- Subsequent memory price decreases resulted in **buffers orders of magnitude larger than ISP link**.
- BDPs, and the resulting bufferbloat yielded **RTTs of seconds instead of milliseconds**.

What is BBR?

It’s a congestion-based congestion control algorithm.

The main objective of BBR is to ensure:

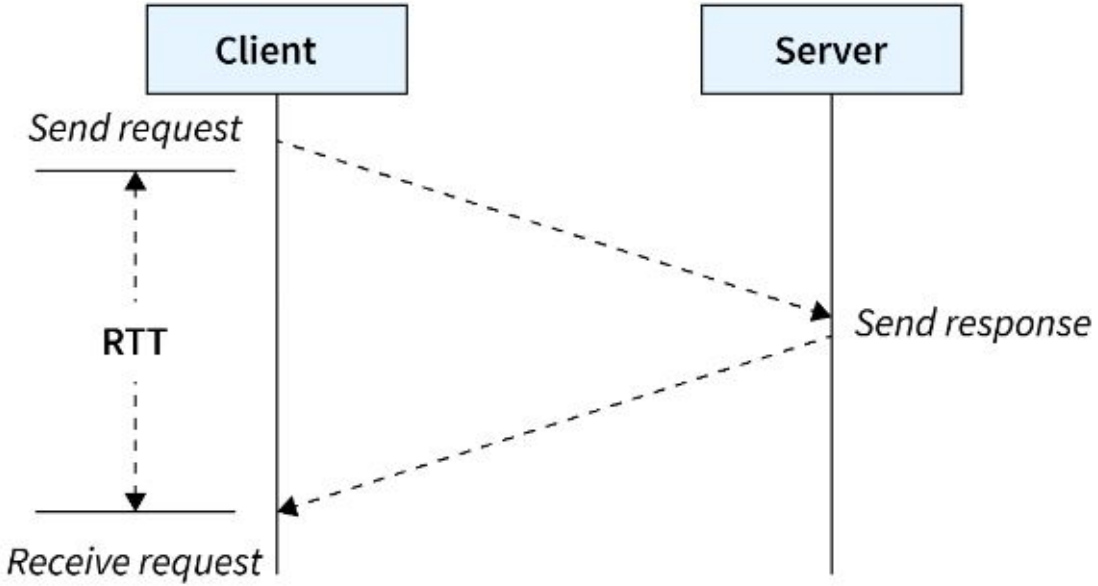
1. the **bottleneck remains saturated**
2. **but not congested**

resulting in maximum throughput with minimal delay.

We need to calculate BtlBw and RTprop

We can get those physical constraints numbers by **RTT** and **delivery rate**.

RTT and RTprop



$$RTT_t = RTprop_t + \eta_t$$

where $\eta \geq 0$ represents the "noise" introduced by queues along the path.

$$\widehat{RTprop} = RTprop + \min(\eta_t) = \min(RTT_t) \quad \forall t \in [T - W_R, T]$$

Delivery Rate and BtlBw

Average delivery rate between send and ack is the ratio of data delivered to time elapsed:

deliveryRate = Δdelivered/Δt

a **windowed-max** of delivery rate is an efficient, unbiased estimator of BtlBw

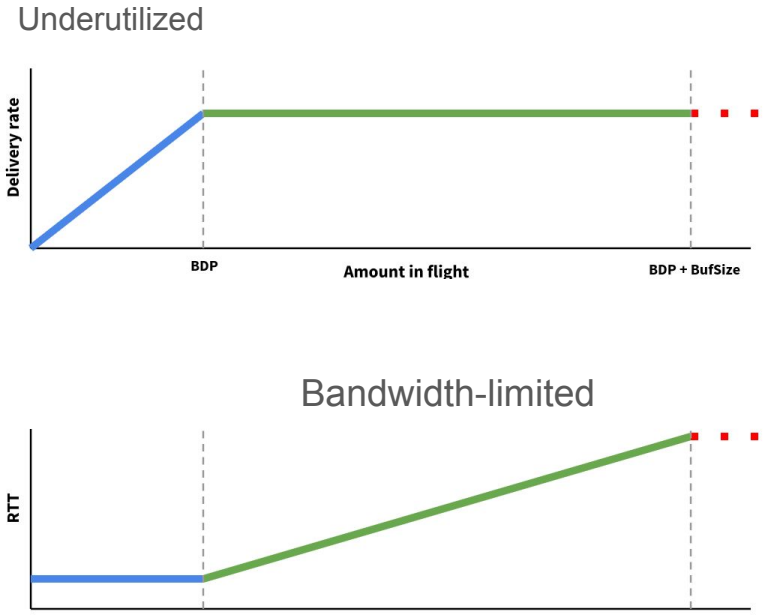
$$\widehat{BtlBw} = \max(deliveryRate_t) \quad \forall t \in [T - W_B, T]$$

RTprop and BtlBW values are completely independent

- RTprop can change (for example, on a **route change**) but still indicate the same bottleneck.
- BtlBw can change (for example, when a **wireless link changes rate**) without the path changing.

Uncertainty principle

- an application running a request/response protocol **might never send enough data to fill the pipe** and observe BtlBw.
- A multi-hour bulk data transfer might **spend its entire lifetime in the bandwidth-limited region** and have only a single sample of RTprop from the first packet's RTT.



BBR state machine goals

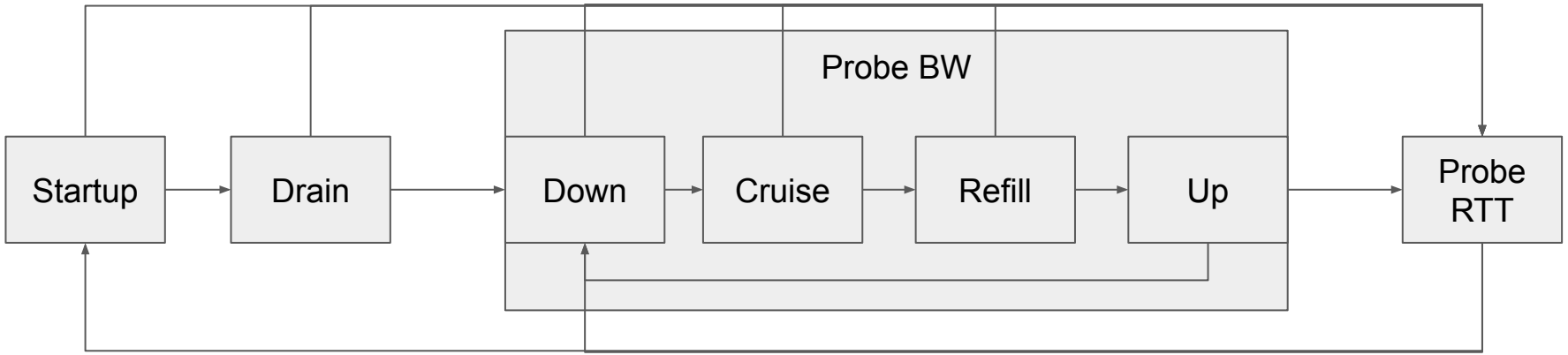
- Achieve high throughput by efficiently utilizing available bandwidth.
- Achieve low latency and packet loss rates by keeping queues bounded and small.
- Share bandwidth with other flows in an approximately fair manner.
- Feed samples to the model estimators to refresh and update the model.

How BBR Implementations Work? (Tactics)

In the BBR framework, at any given time the sender can choose one of the following tactics:

- Acceleration
  - probe the maximum bandwidth
- Cruising
  - try to achieve high utilization of the available bandwidth without increasing queue pressure
- Deceleration
  - Reducing queuing delay
  - Reducing packet loss
  - Probing BBR.min_rtt
  - Bandwidth convergence
  - Burst tolerance

How BBR Implementations Work? (States)



Summary of Control Behavior in the State Machine

- **State**: The state in the BBR state machine
- **Tactic**: The tactic chosen from the "State Machine Tactics"
- **Pacing Gain**: The gain of the TCP pacing calculation use to limit the burst rate of a TCP flow.
- **Cwnd Gain**: The gain of the TCP congestion window calculation.
- **Bandwidth**: The maximum sending bandwidth that the algorithm estimates is appropriate for matching the current network path delivery rate
- **Maximum Bandwidth**: The windowed maximum recent bandwidth sample, obtained using the BBR delivery rate sampling algorithm
- **Rate Cap**: The rate values applied as bounds on the BBR maximum bandwidth value applied to compute BBR bandwidth value.

Summary of Control Behavior in the State Machine

- **inflight:** refers to the amount of data that has been sent but not yet acknowledged by the receiver.
- **inflight_hi**: The long-term maximum volume of in-flight data that the algorithm estimates will produce acceptable queue pressure, based on signals in the current or previous bandwidth probing cycle, as measured by loss.
- **Volume Cap**: The volume values applied as bounds on the BBR maximum inflight value to compute congestion window.
- **inflight_lo**: The short-term maximum volume of in-flight data that the algorithm estimates is safe for matching the current network path delivery process, based on any loss signals in the current bandwidth probing cycle.

Summary of Control Behavior in the State Machine

The control behavior can be summarized as follows. Upon processing each ACK:

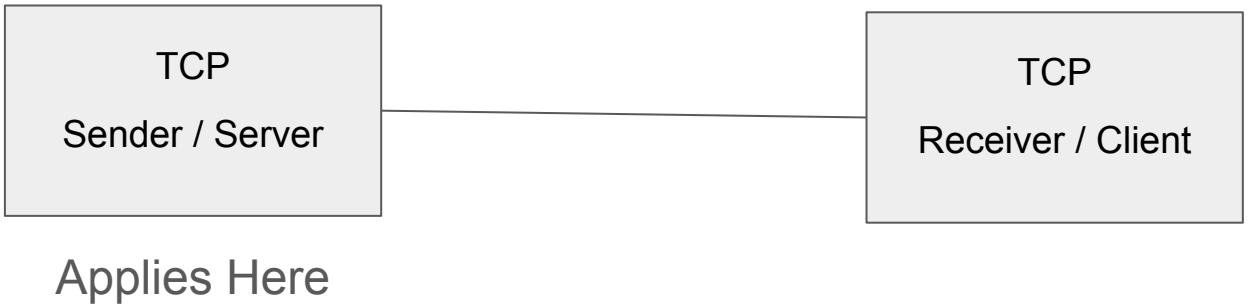
State	Tactic	Pacing Gain	Cwnd Gain	Rate Cap	Volume Cap
Startup	accel	2.77	2	N/A	N/A
Drain	decel	0.5	2	bw_lo	inflight_hi,inflight_lo
ProbeBW_DOWN	decel	0.9	2	bw_lo	inflight_hi,inflight_lo
ProbeBW_CRUISE	cruise	1.0	2	bw_lo	0.85*inflight_hi, inflight_lo
ProbeBW_REFILL	accel	1.0	2		inflight_hi
ProbeBW_UP	accel	1.25	2.25		inflight_hi
ProbeRTT	decel	1.0	0.5	bw_lo	0.85*inflight_hi,inflight_lo

TCP
Congestion Control

Usage

Where it applies

You only need to configure the sender side of your TCP connections for the entire network path to operate using BBR for your data flow.



Where is BBR congestion control used in production?

It was the Google idea so:

- BBRv3 is TCP congestion control for all internal WAN traffic.
- Youtube servers are using BBRv3 for their TCP CC.
- Google Cloud Platform (GCP) and Chrome web browser.

Amazon:

- Amazon CloudFront, their content delivery network, utilizes BBR for improved traffic flow.

Other Companies: BBR is being adopted by other companies as well. Since it's an open-source standard.

How to enable BBR on your servers or clients? (FreeBSD)

To enable the TCP stack you must place the following line in the sysctl.conf(5):

```
net.inet.tcp.functions_default=bbr
```

Or simply execute this command:

```
sysctl net.inet.tcp.functions_default=bbr
```

TCP
Congestion Control

Challenges

Inter congestion control algorithm fairness

Comparing multiple flows of Cubic to each versions of the BBR by measuring Jain's fairness index (JFI)

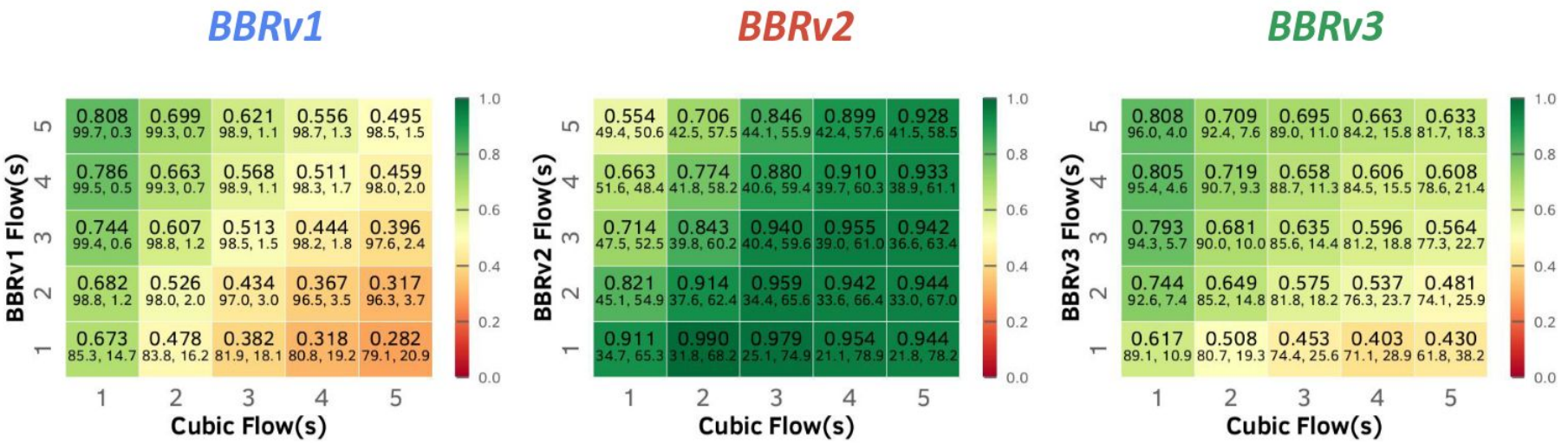


Image Source: Danesh Zeynali, Emilia N. Weyulu, Seifeddine Fathalli Balakrishnan Chandrasekaran, Anja Feldmann - BBRv3 in the public Internet: a boon or a bane? [IETF 120]

Source code for programmers

The pseudo-code already exists in the following draft:

- https://github.com/ietf-wg-ccwg/draft-cardwell-ccwg-bbr

I have a somewhat overly simplified implementation of BBRv3 in C for those curious about how it works:

- https://github.com/spmzt/tcp_bbr

Any Questions?