JANOG41

TCP輻輳制御技術動向

フリービット株式会社
石崎 豊

freebit
毎日、発明する会社

自己紹介

freebit
毎日、発明する会社

■石崎 豊
フリービット株式会社
R&D所属

■JANOGとわたし

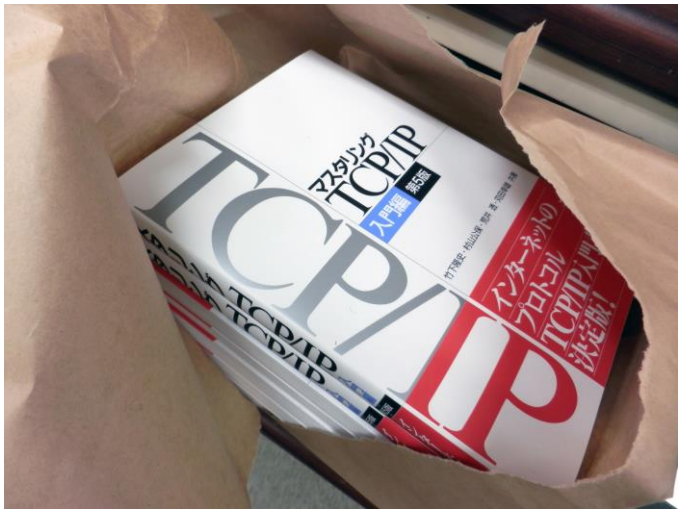
#	場所	内容
JANOG27.5	愛宕	【発表】 Hyper Giants CDNキャッシュサーバーからの配信とISPネットワークへの効果と課題
JANOG28	日本橋	【会場ネットワーク提供】 フレッツIPv6 PPPoE, IPv6/IPv4フォールバック環境提供
JANOG27-30,41	金沢ほか	【スタッフ】 PC, プログラム委員長, ORG

TCPとは (説明不要ですね)

freebit
毎日、発明する会社

■データ送受信の基礎的なプロトコル
　■コネクション型
　■データを確実に送受信をおこなえるように設計
■人生も学べる
■TCPの基本的な情報は「マスタリングTCP/IP」をご参考に

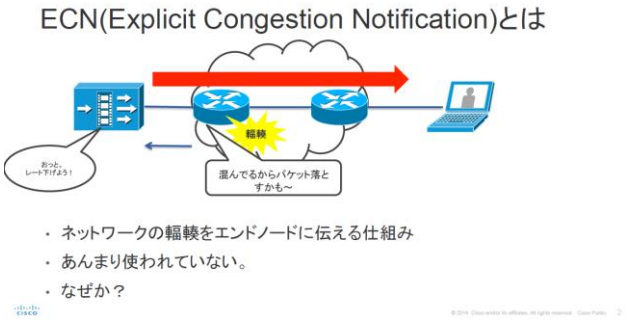
■マスタリングTCP/IP入門編



※ Geekなページ: http://www.geekpage.jp/blog/?id=2008/1/29

TCP輻輳制御

■データ転送量を制御して輻輳を回避する技術
　■TCP輻輳制御技術の方式
　　■送信側で制御する方式
　　= Host Only (Senderベース)
　　■NW側でECNを利用して輻輳通知をおこなう方式
　　= Host + Router(ECN)連携



※ECNのあるネットワーク
　土屋 師子生 (シスコシステムズ合同会社)
　JANOG LT Night #1 2015年8月

種類	以前からある輻輳制御	利用方法
Host Only (Senderベース)	Reno, Vegas, NewReno, Westwood, CUBIC, …	送信側(サーバー)の変更のみで対応可能
Host + Router連携	ECN, CoDel(with ECN), XCP, DCTCP	NW(ルーター), ホスト両方の対応が必要

※ Wikipedia: TCP congestion control
https://en.wikipedia.org/wiki/TCP_congestion_control

TCP輻輳制御アルゴリズムの種類

■古くからある輻輳制御アルゴリズム

方式	リリース時期	動作種別	OSデフォルト採用
(New)Reno	1990年～ (1996年～)	Loss	FreeBSD
Vegas	1994年～	Delay	
Westwood	2001年～	Hybrid (Loss/Delay)	
CUBIC	2005年～	Loss	Linux (2.6.19 以降) Windows 10 Mac OS X

■主な動作種別

■Lossベース

パケットロスを観測し、ロスが増えれば輻輳が発生したと見なして送信量を抑える方式

■Delayベース

パケットのRTTを観測し、送信量を制御する方式

あたらしいTCP輻輳制御アルゴリズム

freebit
毎日、発明する会社

■あたらしい輻輳制御アルゴリズム

方式	リリース時期	動作種別	開発元
YeAH	2007年	Hybrid	University of Roma
Remy	2013年	Computer-generated?	MIT
TIMELY	2015年	RTT(Delay)	Google
BBR	2016年～	Congestion-based	Google

■主な動作種別

■Lossベース

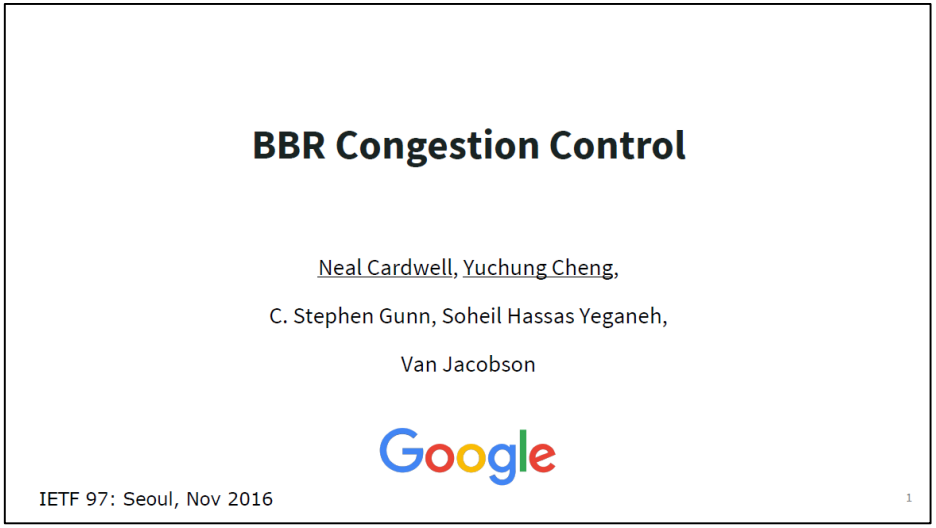
パケットロスを観測し、ロスが増えれば輻輳が発生したと見なして送信量を抑える方式

■Delayベース

パケットのRTTを観測し、RTTが増えたら輻輳状態になったと判断して送信量を抑える方式

TCP BBRとは

■BBR Congestion Control
- ■BBR = Bottleneck Bandwidth & Round-trip propagation time (BtlBw & RTprop)
- ■Googleが開発、IETF97からICCRGへ提案されている輻輳制御アルゴリズム
- ■Linux kernel v4.9からカーネルへマージされた (2016年12月リリース)
- ■Googleはサービスで使い始めている模様
  - ■Internal WANs(B4), Google.com, YouTube
  - ■NETFLIXがFreeBSDへインプリ中.. (Googleプレゼンより)
- ■QUICの輻輳制御アルゴリズムにもBBRが採用される予定



BBR Congestion Control

Neal Cardwell, Yuchung Cheng,

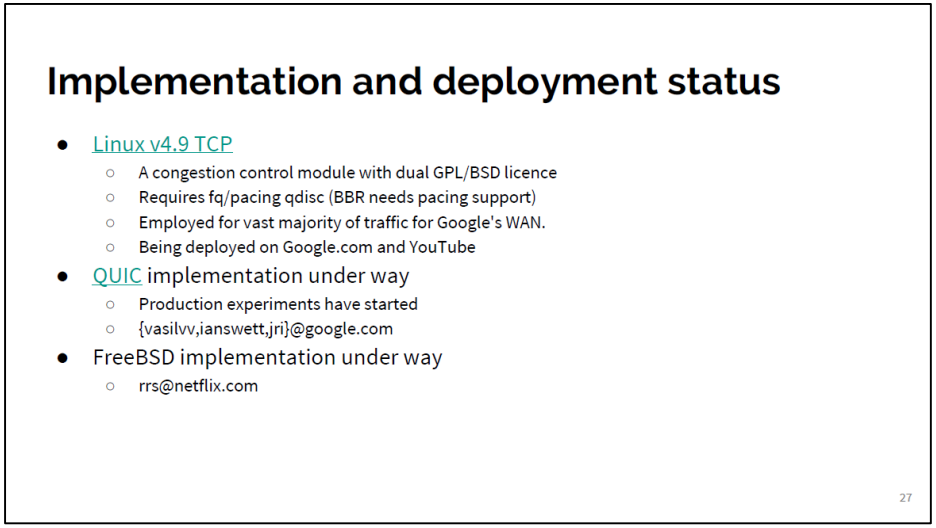
C. Stephen Gunn, Soheil Hassas Yeganeh,

Van Jacobson

Google

IETF 97: Seoul, Nov 2016

1



Implementation and deployment status

- Linux v4.9 TCP
  - A congestion control module with dual GPL/BSD licence
  - Requires fq/pacing qdisc (BBR needs pacing support)
  - Employed for vast majority of traffic for Google's WAN.
  - Being deployed on Google.com and YouTube
- QUIC implementation under way
  - Production experiments have started
  - {vasilvv,ianswett,jri}@google.com
- FreeBSD implementation under way
  - rrs@netflix.com

27

※ IETF97 ICCRG Presentation Slides

Open Networking Summit 2017 (Apr, 2017 @Santa Clara)

freebit
毎日、発明する会社

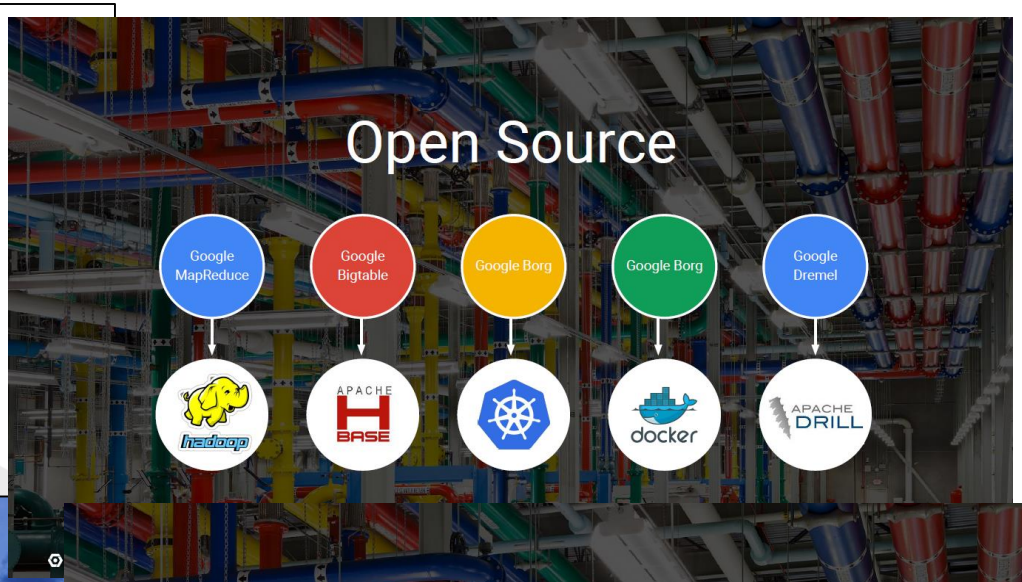
■Keynote: Networking Challenges for the Next Decade - Google Cloud

Google Cloud

Networking Challenges for the Next Decade

Amin Vahdat
On behalf of Google Technical Infrastructure and Google Cloud Platform

APRIL 4, 2017

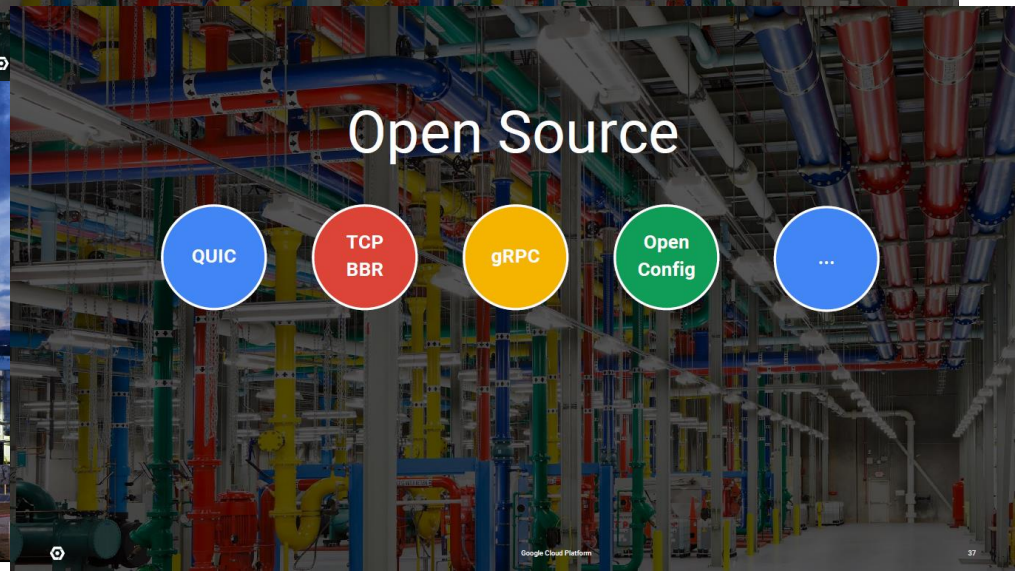


Next Decade Challenges in Networking

The next wave of computing
- Serverless compute in Cloud 3.0
- IoT
- Tightly coupled, general purpose distributed computing

It's time to put it all together
- Agile Scale
- Jitter
- Isolation
- Performance is great, but only meaningful with availability, manageability, and velocity

Google Cloud

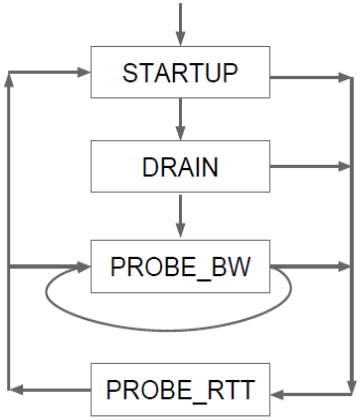


※ ONS2017
https://ons2017.sched.com/

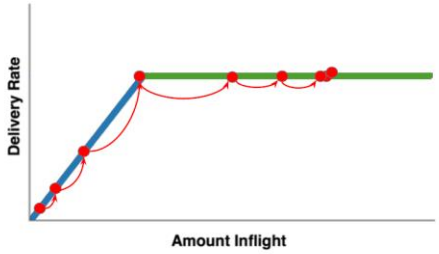
TCP BBRの動作・挙動

freebit
毎日、発明する会社

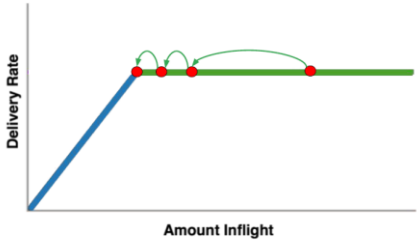
BBR state transition diagram



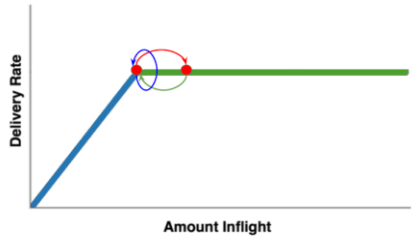
STARTUP: exponential BW search



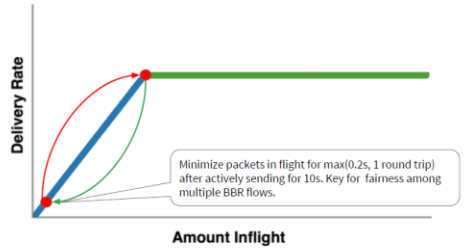
DRAIN: drain the queue created during startup



PROBE_BW: explore max BW, drain queue, cruise

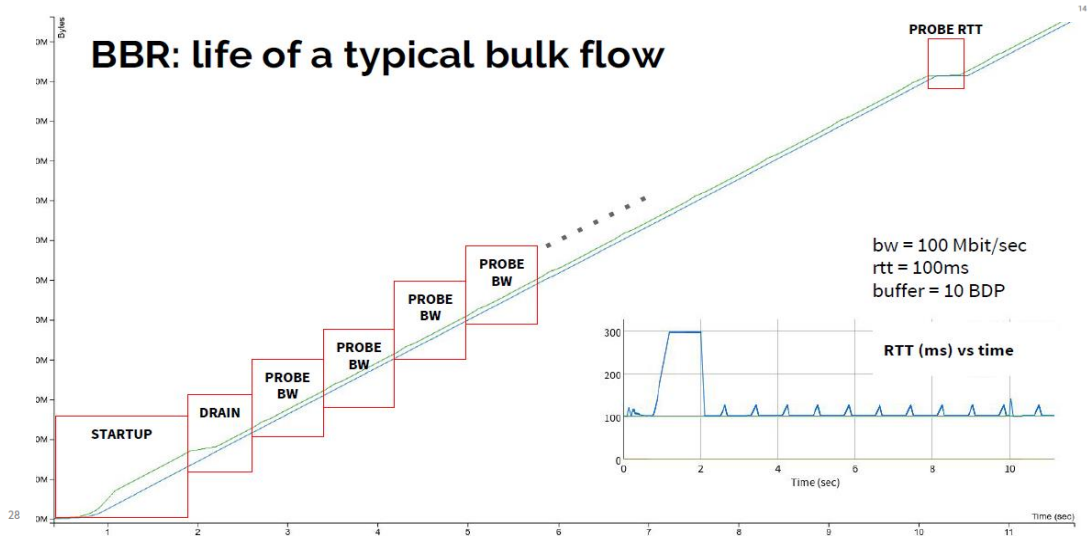


PROBE_RTT drains queue to refresh min_RTT



BBR FAQ

Is BBR fair to Cubic/Reno?	Buffer >= 1.5*BDP: Yes; Else: WIP
Is BBR 1/sqrt(p)?	No
Is BBR {delay|loss|ECN|AIMD}-based?	No. It is congestion-based
Is BBR ack-clocked?	No
Does BBR require pacing?	Yes
Does BBR require an FQ scheduler?	No, but it helps
Does BBR require receiver or network changes	No
Does BBR improve latency on short flows?	Yes



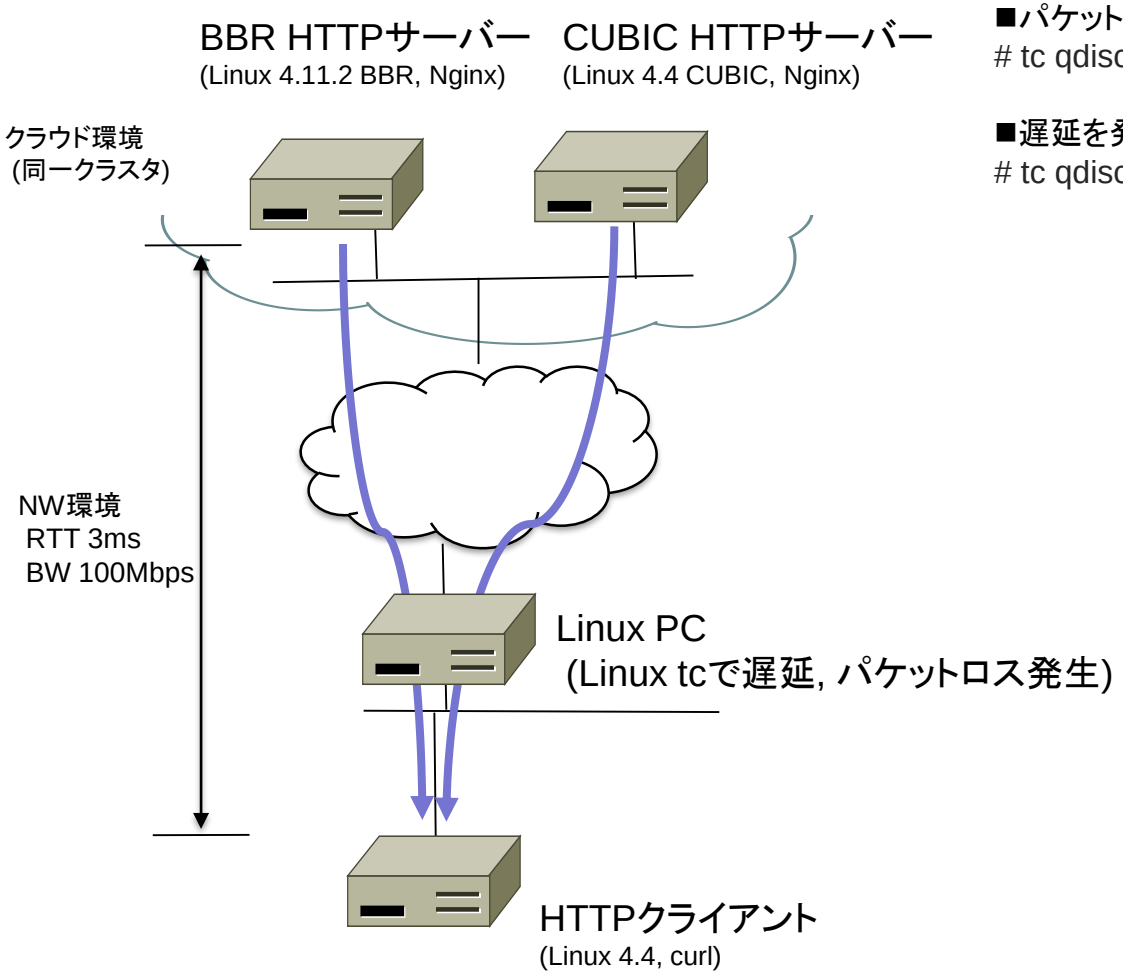
※ IETF97 ICCRG Presentation :
https://www.ietf.org/proceedings/97/slides/slides-97-iccrg-bbr-congestion-control-02.pdf

テスト環境を作って試してみた

テスト環境 (疑似的に低品質NW環境を構築)

freebit
毎日、発明する会社

■構成図



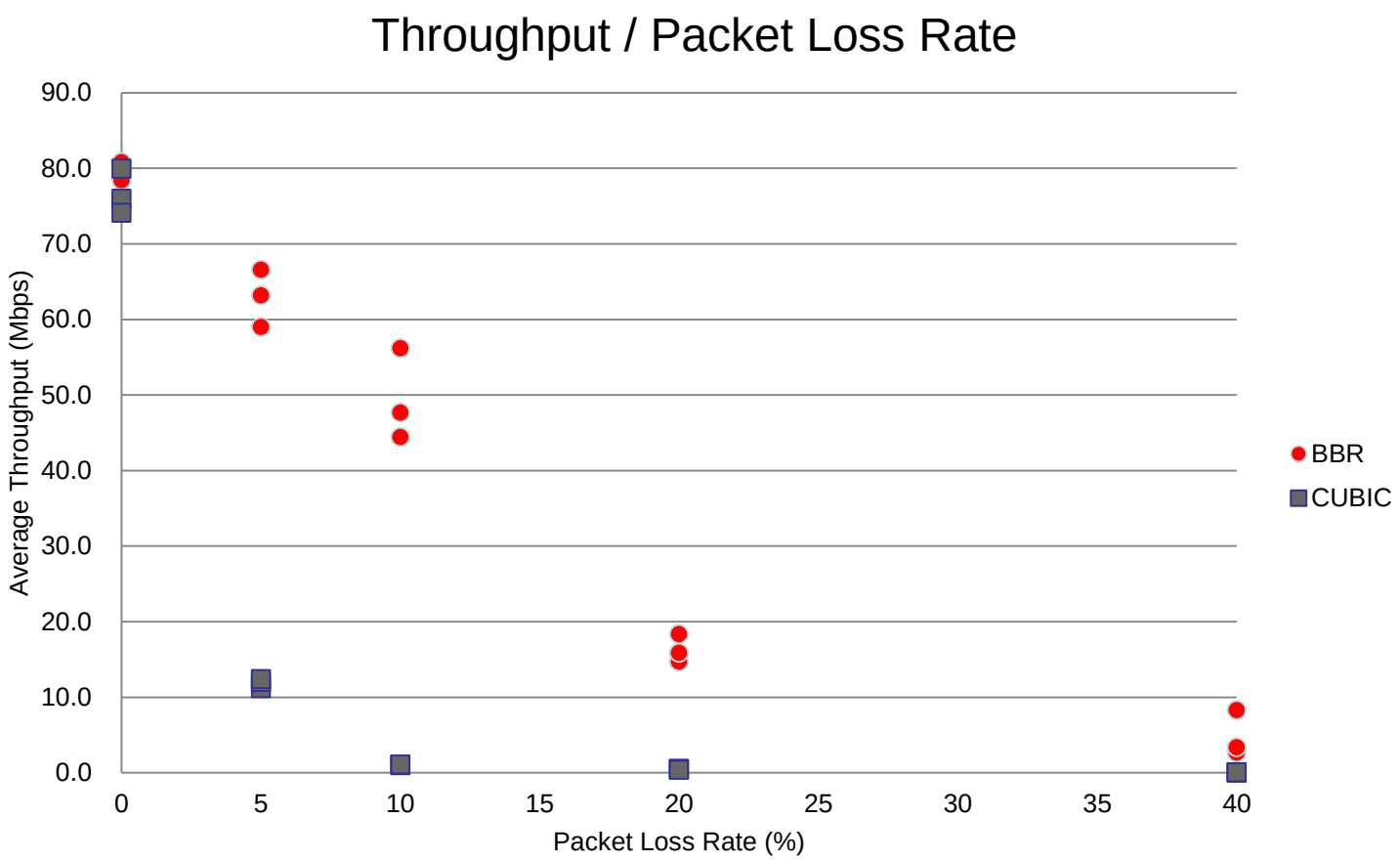
■Linux tc設定

■パケットロス発生させる設定
tc qdisc add dev <ifname> root netem delay 200ms

■遅延を発生させる設定
tc qdisc add dev <ifname> root netem loss 20%

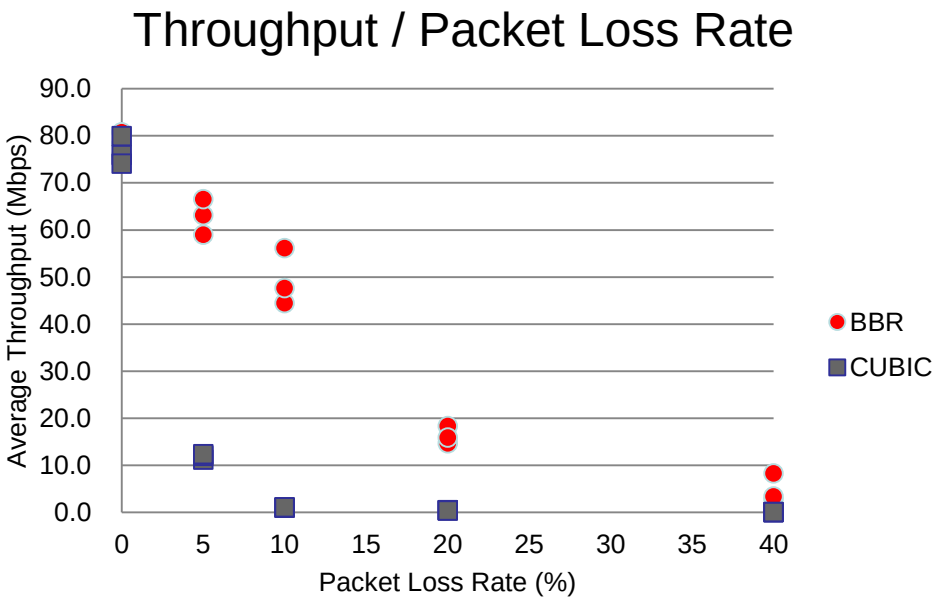
スループット測定

■パケットロス環境下での測定
- ■tc設定で0%, 5%,10%, 20%, 40%ごとのPacket Loss Rateで
- ■BBR, CUBIC 3回計測

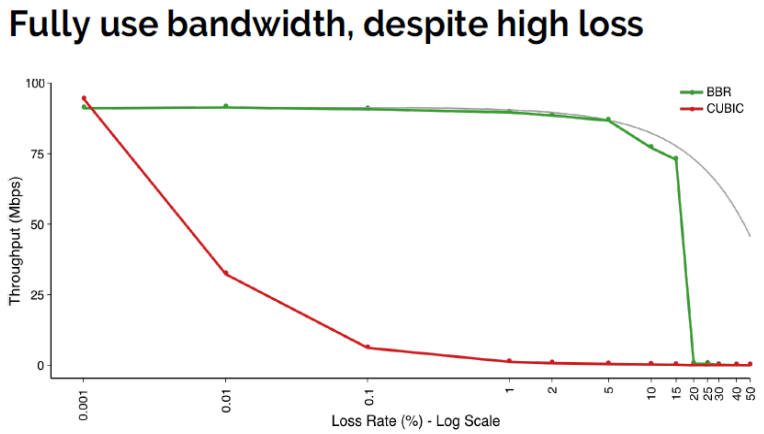


スループット測定

■パケットロス環境下での測定
- ■tc設定で0%, 5%,10%, 20%, 40%ごとのPacket Loss Rateで
- ■BBR, CUBIC 3回計測

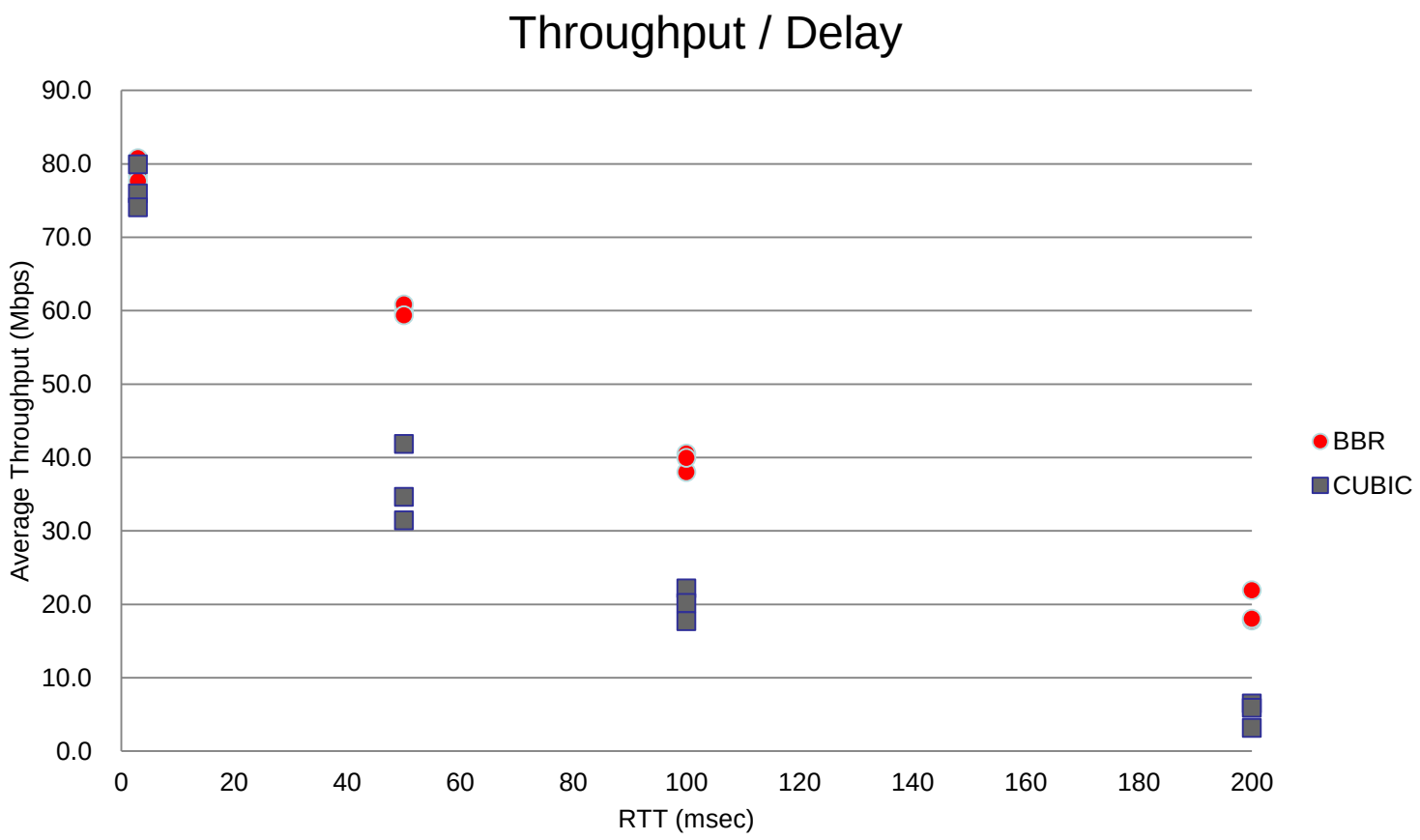


■ IETF97 Google BBRレポートでの計測結果



スループット測定

■高レイテンシ環境下での測定
- ■tc設定で none(3ms), 50, 100, 200msごとのRTTで
- ■BBR, CUBIC 3回計測



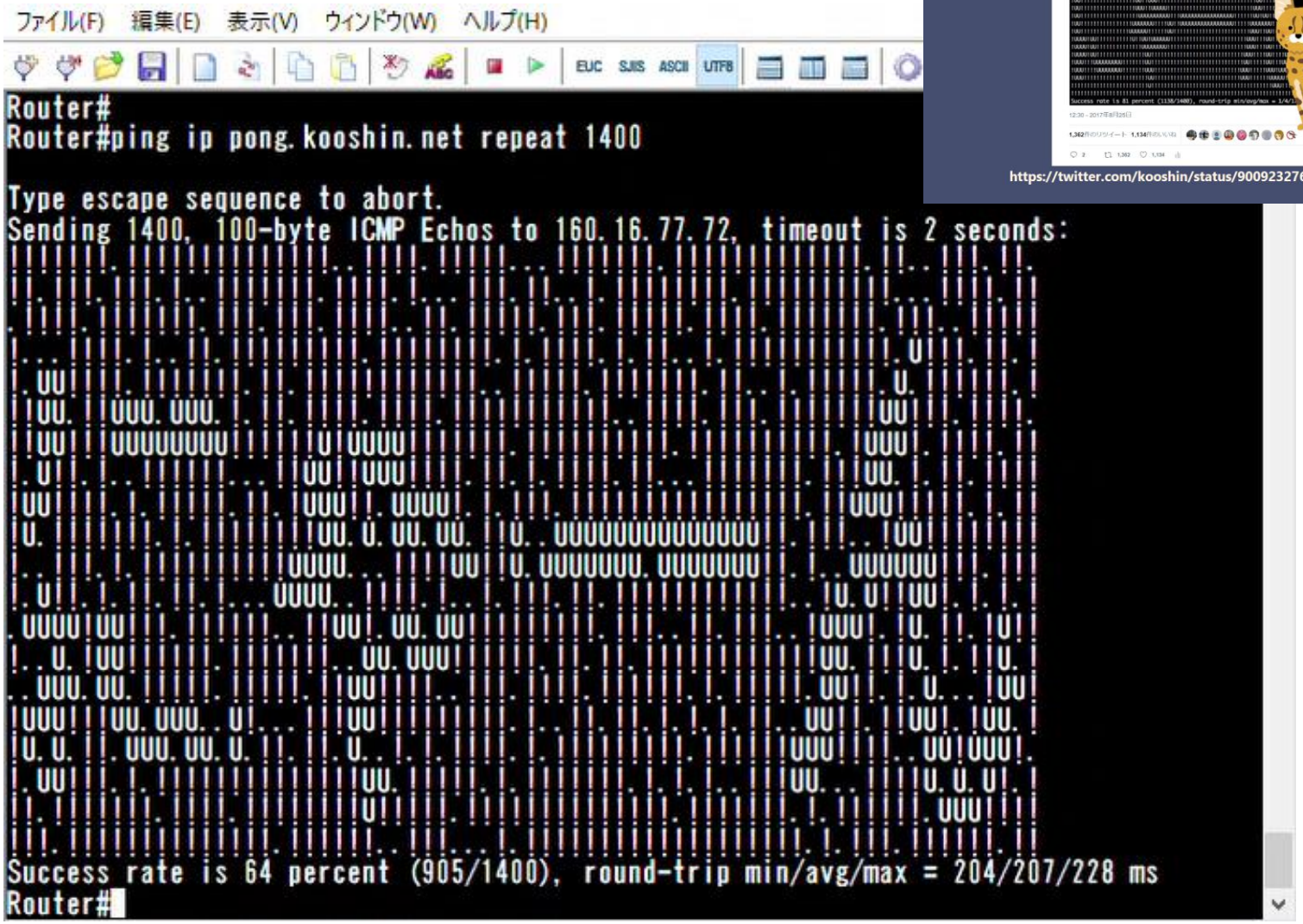
ちなみに、
パケットロス 20%, RTT 200msec 環境とは
どんな感じか？

pingすると「にゃーん」を返すサービス @kooshinさん

freebit
毎日、発明する会社

低品質NW環境で「にゃーん」

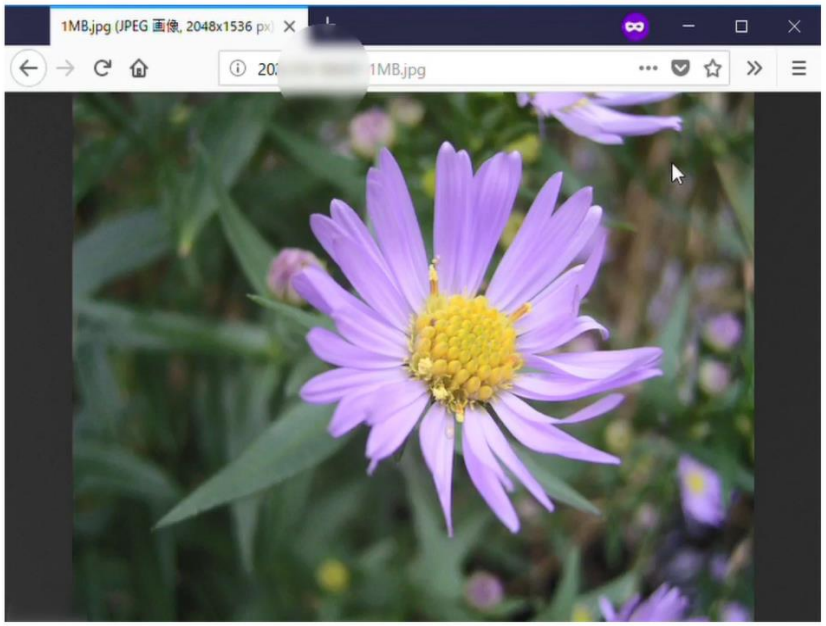
■パケットロス 20%, RTT 200msec 環境



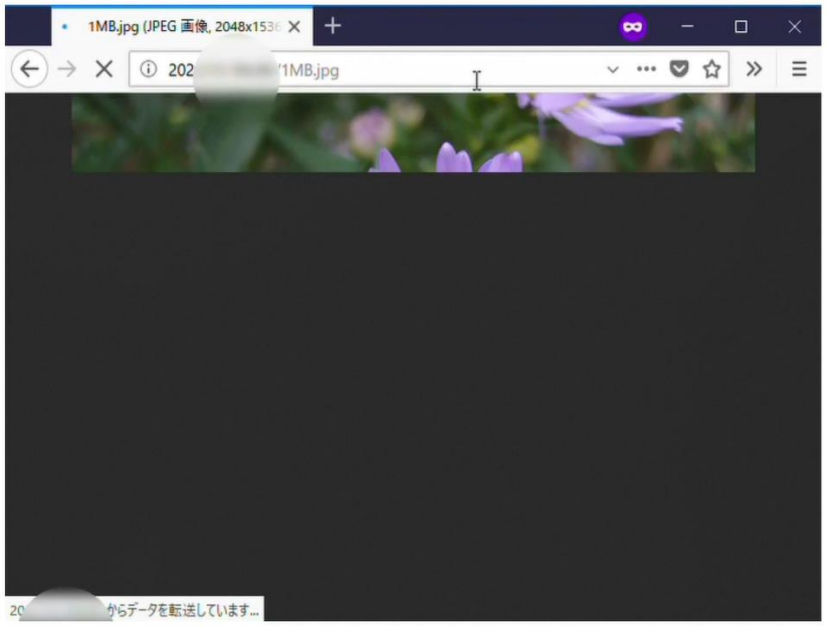
低品質NW環境下でのスループット比較

■パケットロス 20%, RTT 200msec 環境
■1MBの画像ファイルをHTTPダウンロード

TCP BBR



TCP CUBIC



■ダウンロード完了まで
約11秒　　約400秒 ~ timeout

TCP BBR 設定・利用方法

freebit
毎日、発明する会社

■Linux kernelの更新 (BBRサポートは Linux kernel v4.9以上)

```
※Ubuntuの場合
http://kernel.ubuntu.com/~kernel-ppa/mainline/ から最新カーネルをダウンロード

#  ls
linux-headers-4.13.7-041307-generic_4.13.7-041307.201710141430_amd64.deb
linux-image-4.13.7-041307-generic_4.13.7-041307.201710141430_amd64.deb
linux-headers-4.13.7-041307_4.13.7-041307.201710141430_all.deb

# dpkg -i *.deb
# reboot
```

■sysctlで利用可能な輻輳制御アルゴリズムにBBRが追加されていることを確認

```
$ sysctl net.ipv4.tcp_available_congestion_control
net.ipv4.tcp_available_congestion_control = bbr cubic reno
```

■TCP BBRの有効化、キューイングアルゴリズムを fqに変更

```
# vi /etc/sysctl.conf
net.ipv4.tcp_congestion_control = bbr
net.core.default_qdisc = fq
```

TCP BBR 設定・利用方法

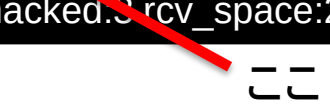
freebit
毎日、発明する会社

■確認コマンド

```
# sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
```

```
# tc qdisc
qdisc noqueue 0: dev lo root refcnt 2
qdisc mq 0: dev eth0 root
qdisc fq 0: dev eth0 parent :1 limit 10000p flow_limit 100p buckets 1024 orphan_mask 1023 quantum 3028
initial_quantum 15140 refill_delay 40.0ms
```

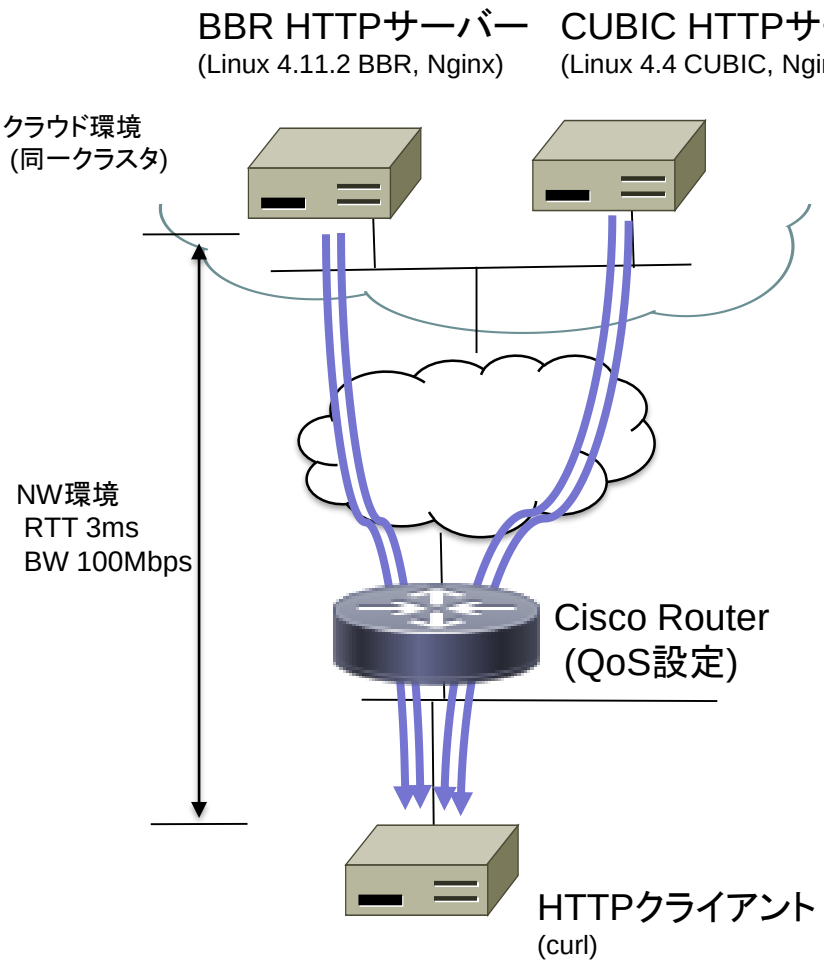
```
# ss --tcp -i
State       Recv-Q Send-Q              Local Address:Port                Peer Address:Port
ESTAB       0      132                 10.246.68.4:ssh                  10.1.0.1:63134
bbr wscale:8,7 rto:240 rtt:38.647/13.194 ato:40 mss:1460 cwnd:147 bytes_acked:17717
bytes_received:7395 segs_out:325 segs_in:391 send 44.4Mbps lastrcv:4 lastack:4 pacing_rate 107.0Mbps
unacked:3 rcv_space:29200
```



そもそも、TCP輻輳制御技術の検証なので、
ネットワーク輻輳環境をつくって測定してみないと…

テスト環境 (QoS環境)

■構成図



■Cisco QoS設定

■ポリシング設定
(CIR 2Mbps, exceed-action drop)

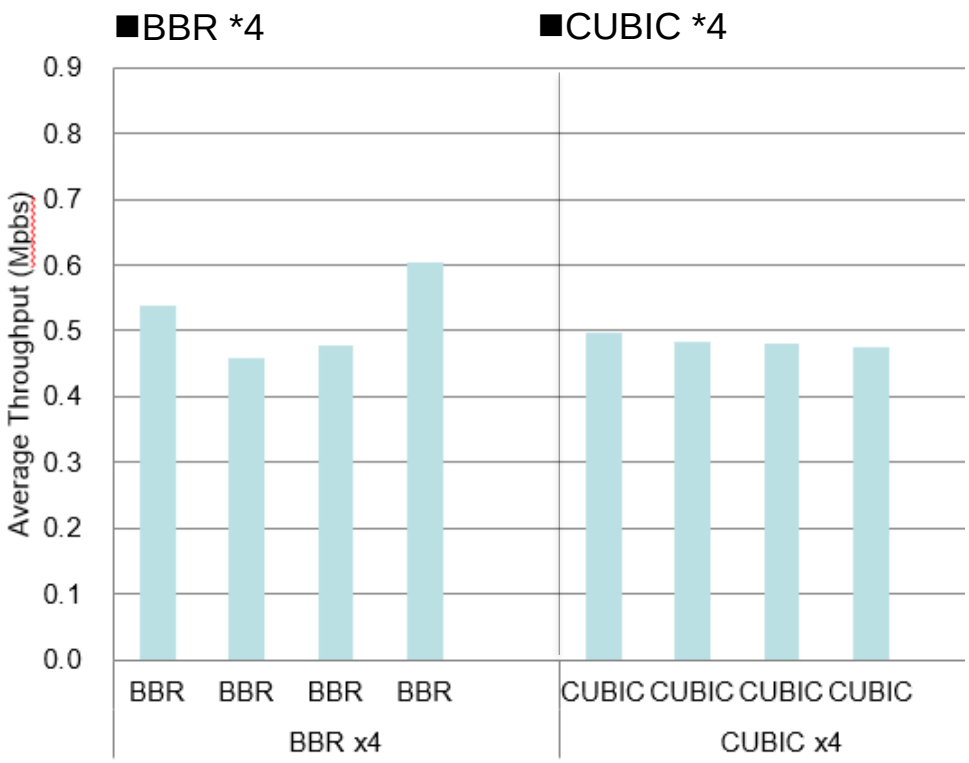
```
!
policy-map class-based-policing
 class class-default
  police 200000 conform-action transmit  exceed-action
drop  violate-action drop
!
interface GigabitEthernet0/0/0
 ip address 10.0.0.1 255.255.255.0
 service-policy output class-based-policing
!
```

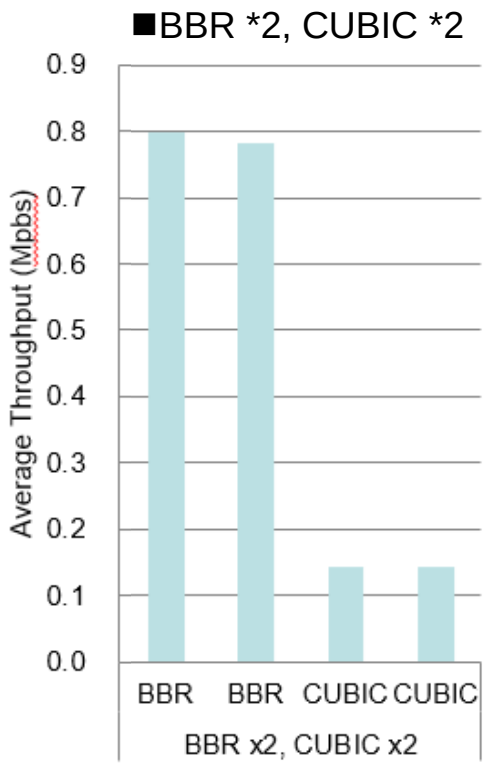
■シェーピング設定
(CIR 2Mbps, target shape rate 4Mbps)

```
!
policy-map class-based-shaping
 class class-default
  shape peak 2000000
!
interface GigabitEthernet0/0/0
 ip address 10.0.0.1 255.255.255.0
 service-policy output class-based-shaping
!
```

帯域制御環境下でのトラフィック影響

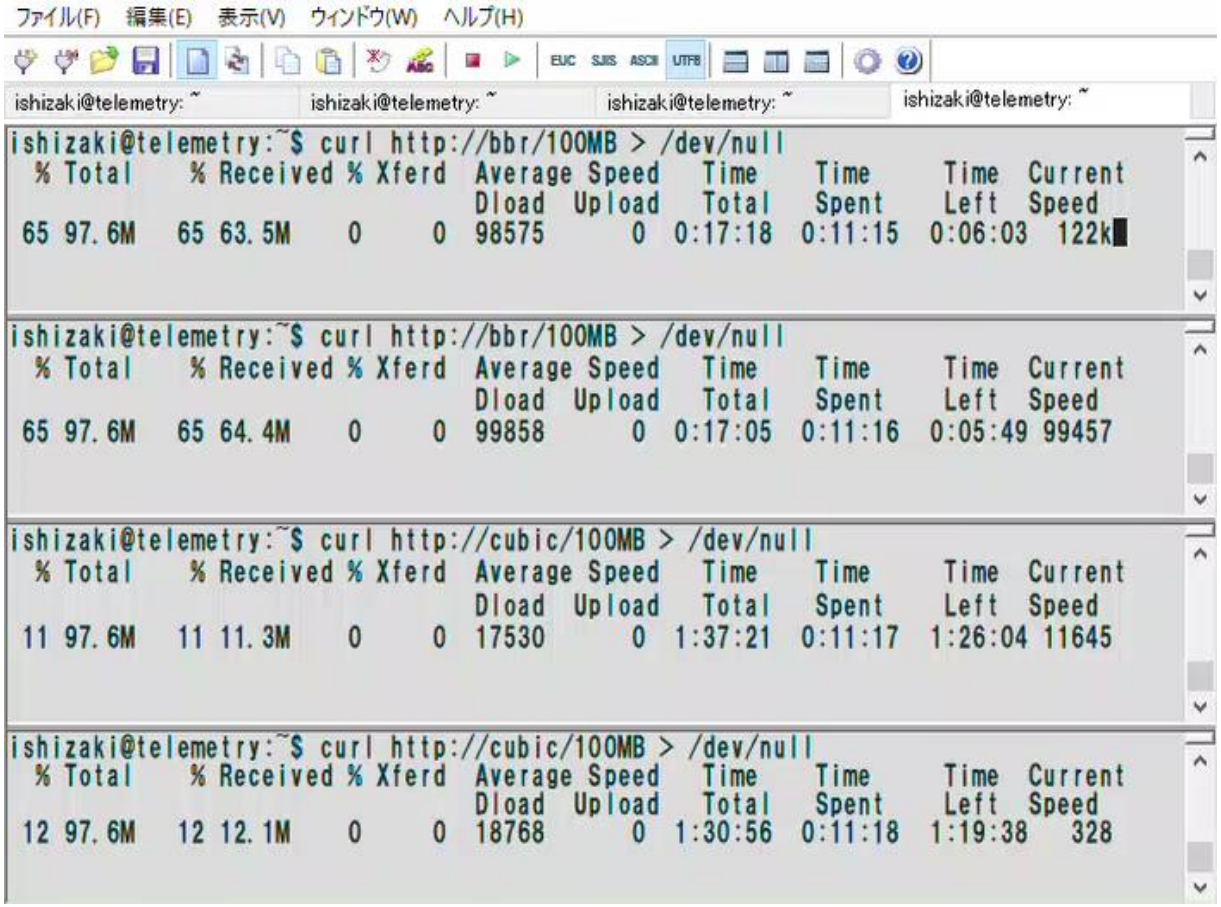
■帯域制御 (ポリシング CIR 2Mbps)
- ■policer CIR 2Mbps, exceed-action drop (Cisco ASR QoS機能を使用)
  - ■計測内容
    - ■BBR *4
    - ■CUBIC *4
    - ■BBR *2 + CUBIC *2

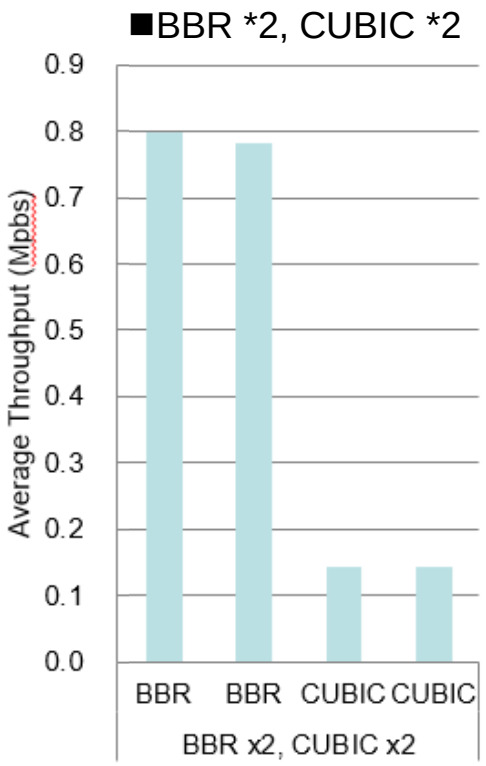




帯域制御環境下でのトラフィック影響

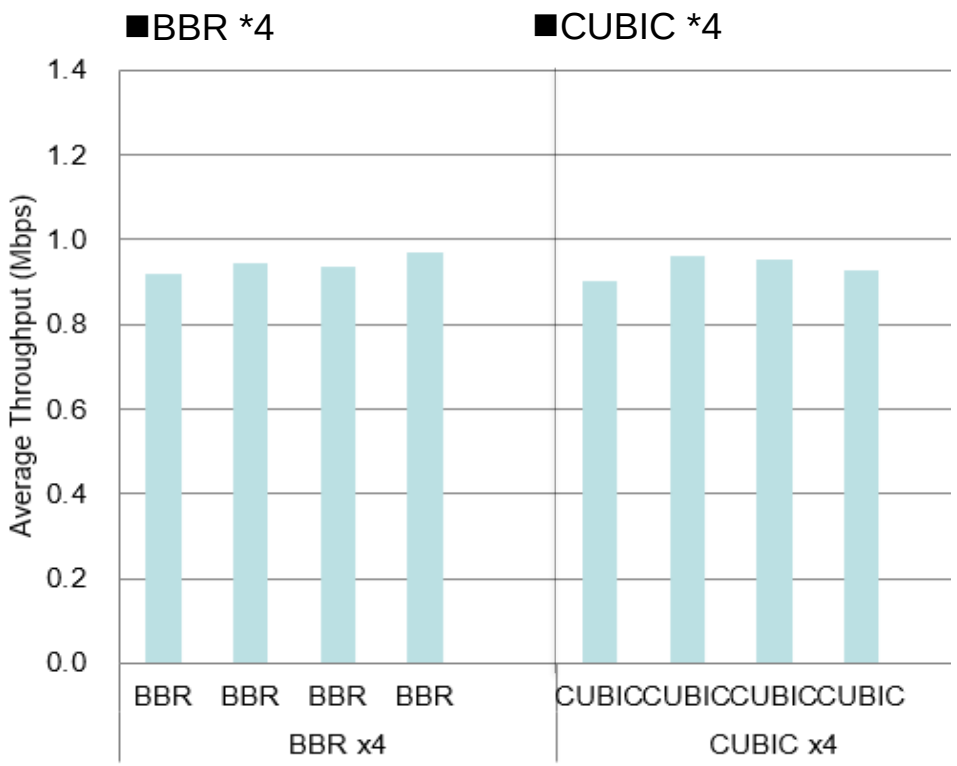
■帯域制御 (ポリシング CIR 2Mbps)
■policer CIR 2Mbps, exceed-action drop (Cisco ASR QoS機能を使用)
■計測内容
■BBR *2 + CUBIC *2

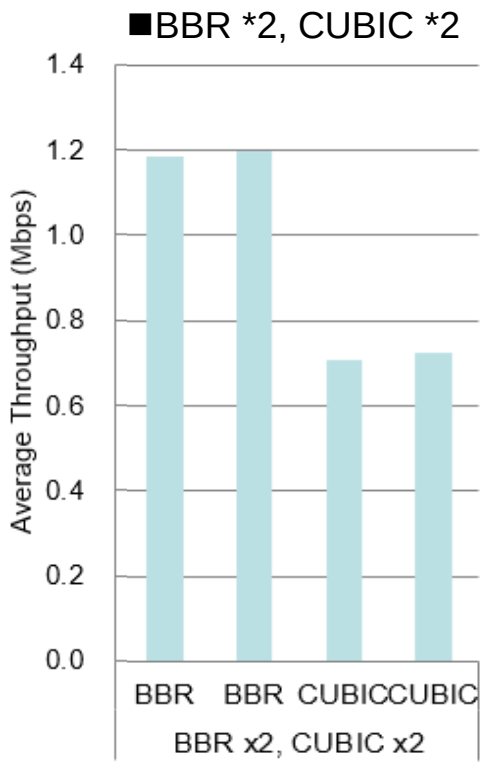




帯域制御環境下でのトラフィック影響

■帯域制御 (シェーピング CIR 2Mbps)
- ■shape CIR 2Mbps, target shape rate 4Mbps (Cisco ASR QoS機能を使用)
  - ■計測内容
    - ■BBR *4
    - ■CUBIC *4
    - ■BBR *2 + CUBIC *2





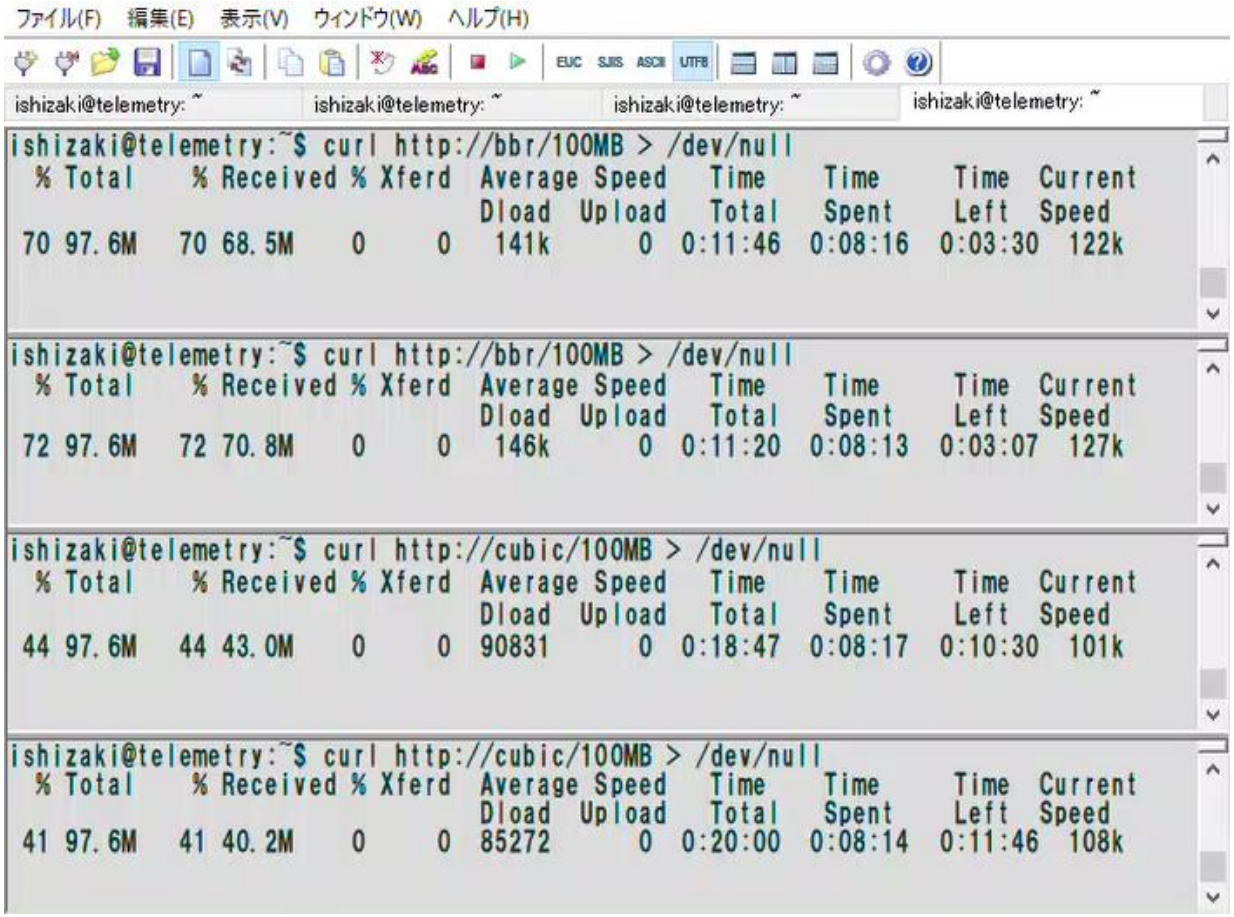
帯域制御環境下でのトラフィック影響

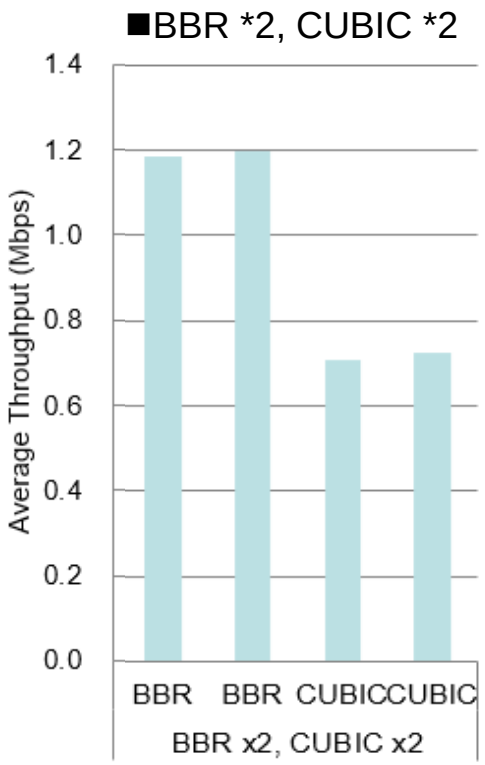
■帯域制御 (シェーピング CIR 2Mbps)

■shape CIR 2Mbps, target shape rate 4Mbps (Cisco ASR QoS機能を使用)

■計測内容

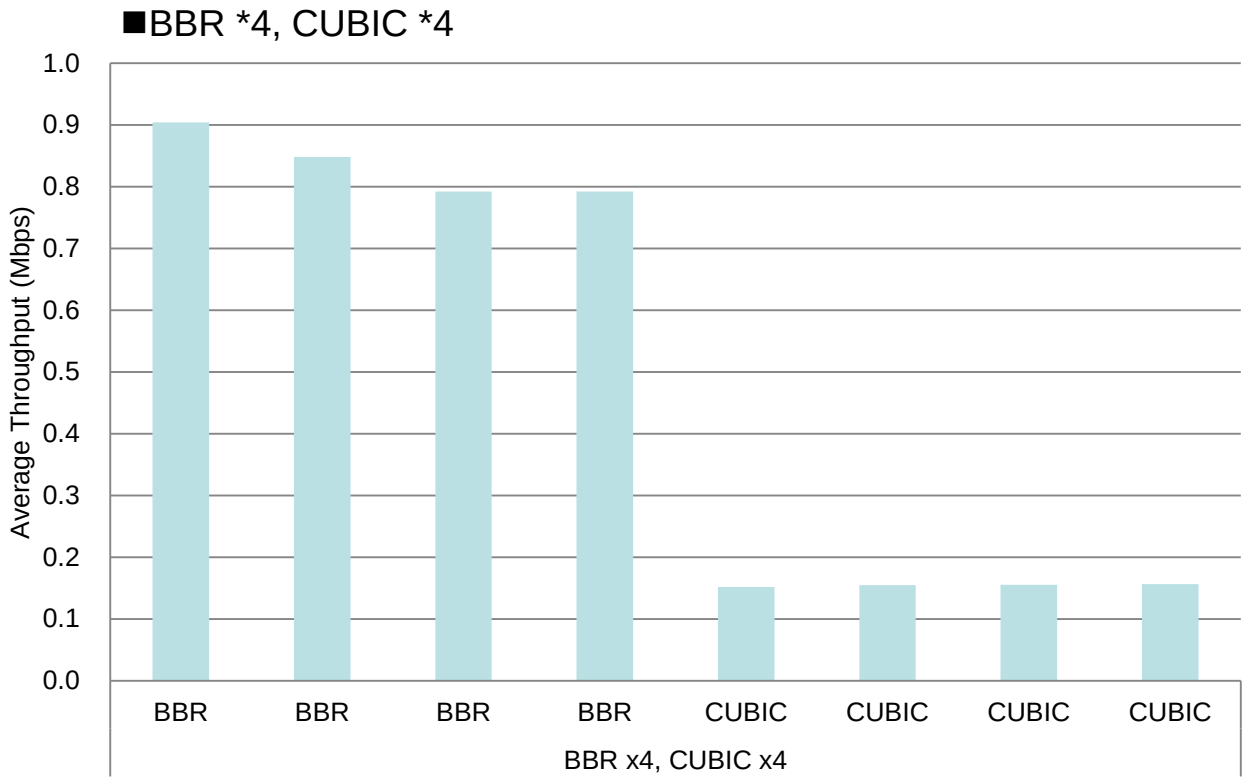
■BBR *2 + CUBIC *2





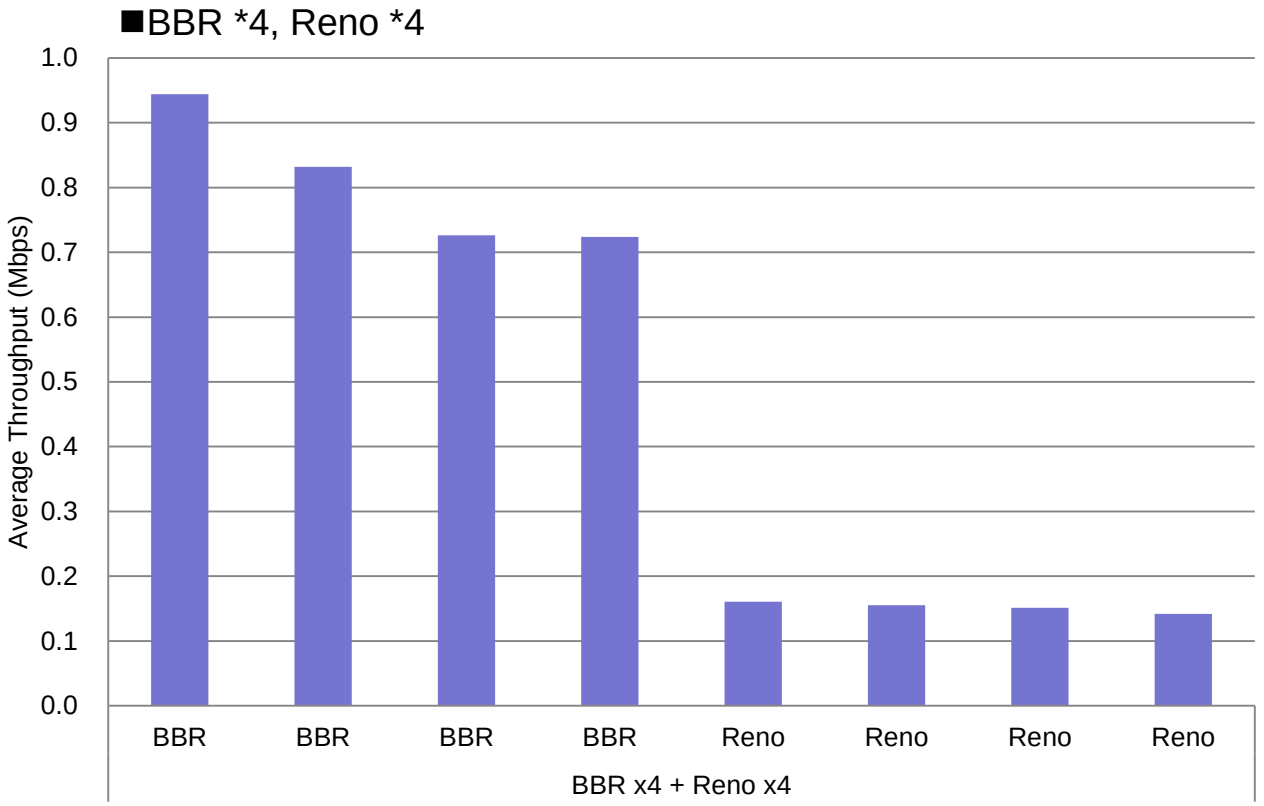
帯域制御環境下でのトラフィック影響

■帯域制御 (シェーピング CIR 2Mbps)
■shape CIR 2Mbps, target shape rate 4Mbps (Cisco ASR QoS機能を使用)
■計測内容
■BBR *4 + CUBIC *4



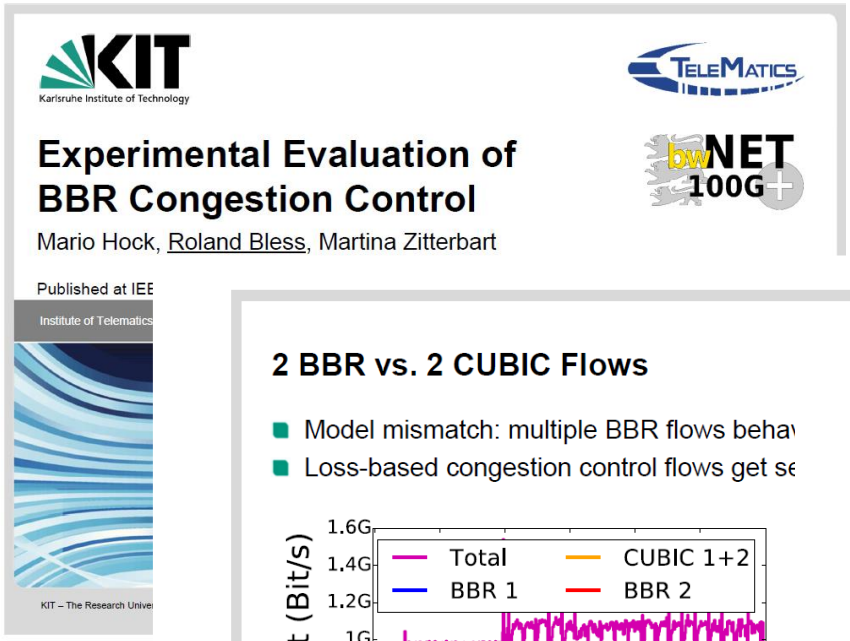
帯域制御環境下でのトラフィック影響

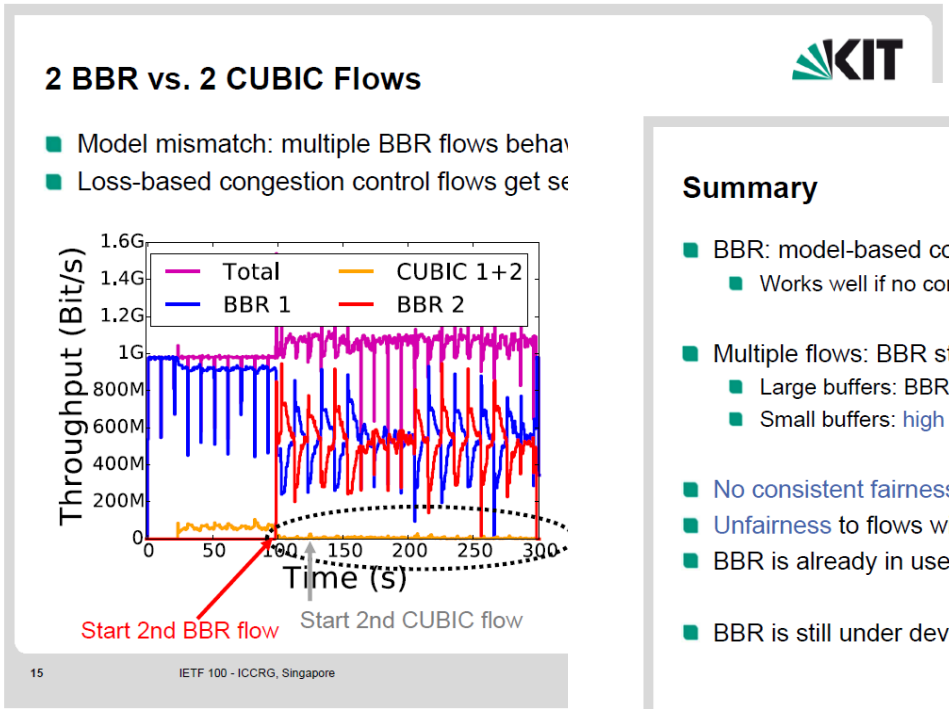
■帯域制御 (シェーピング CIR 2Mbps)
■shape CIR 2Mbps, target shape rate 4Mbps (Cisco ASR QoS機能を使用)
■計測内容
■BBR *4 + Reno *4

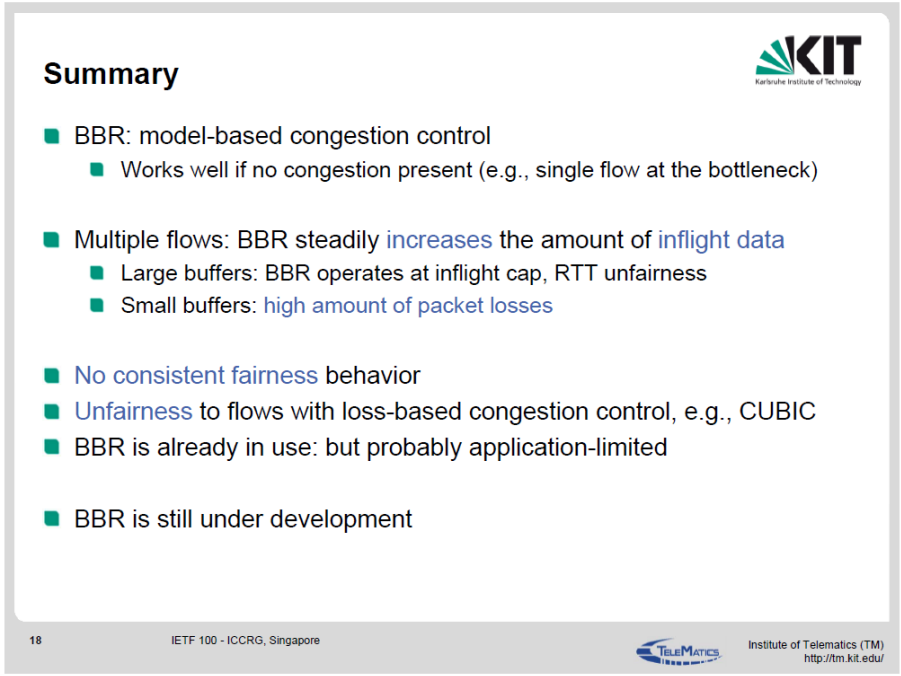


TCP BBR 第三者評価

■IEEE 25th International Conference on Network Protocols (ICNP) 2017 @Toronto
■Experimental evaluation of BBR Congestion Control - KIT





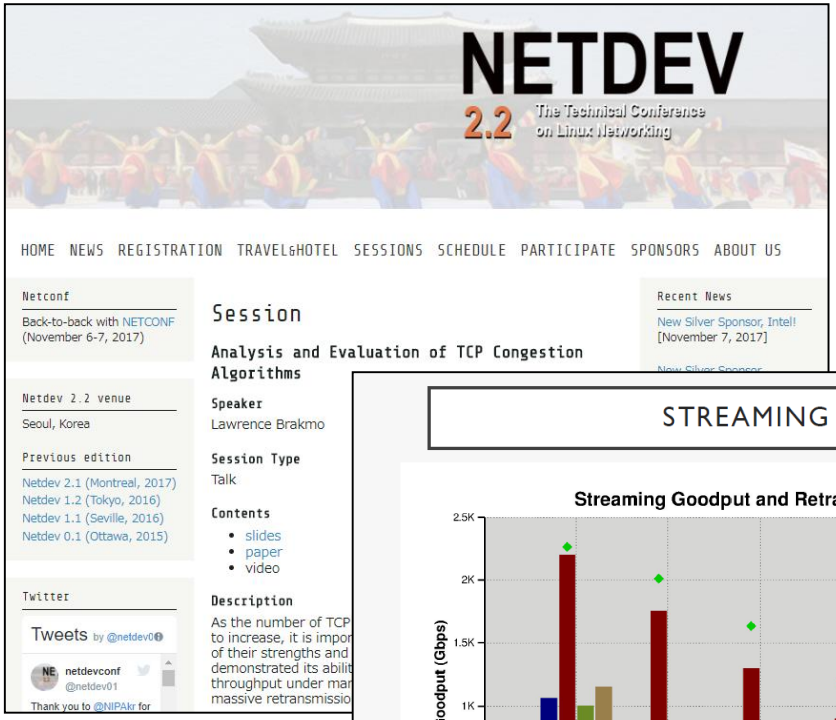


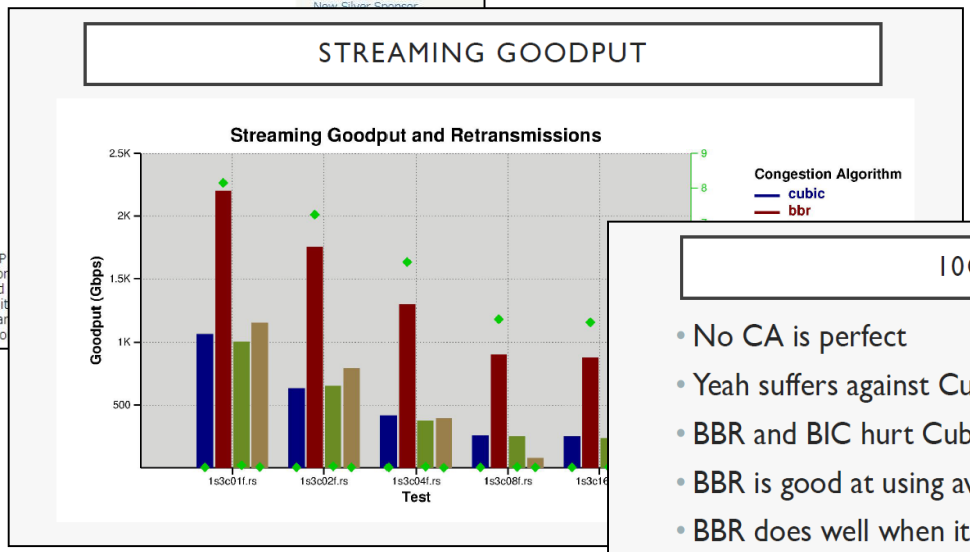
※ http://iqua.ece.toronto.edu/icnp17/

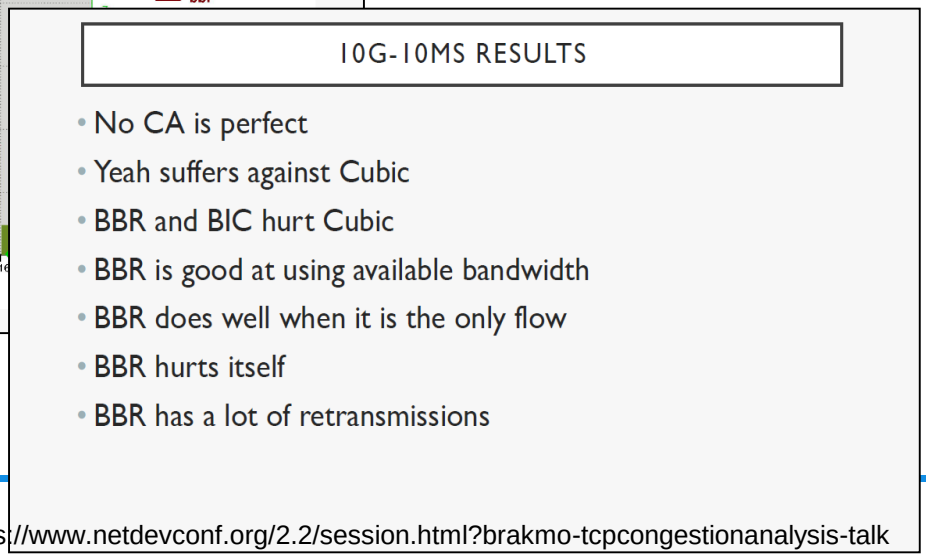
TCP BBR 第三者評価

■NETDEV 2.2 @Seoul, Korea – Nov, 2017

■Analysis and Evaluation of TCP Congestion Algorithms - Lawrence Brakmo



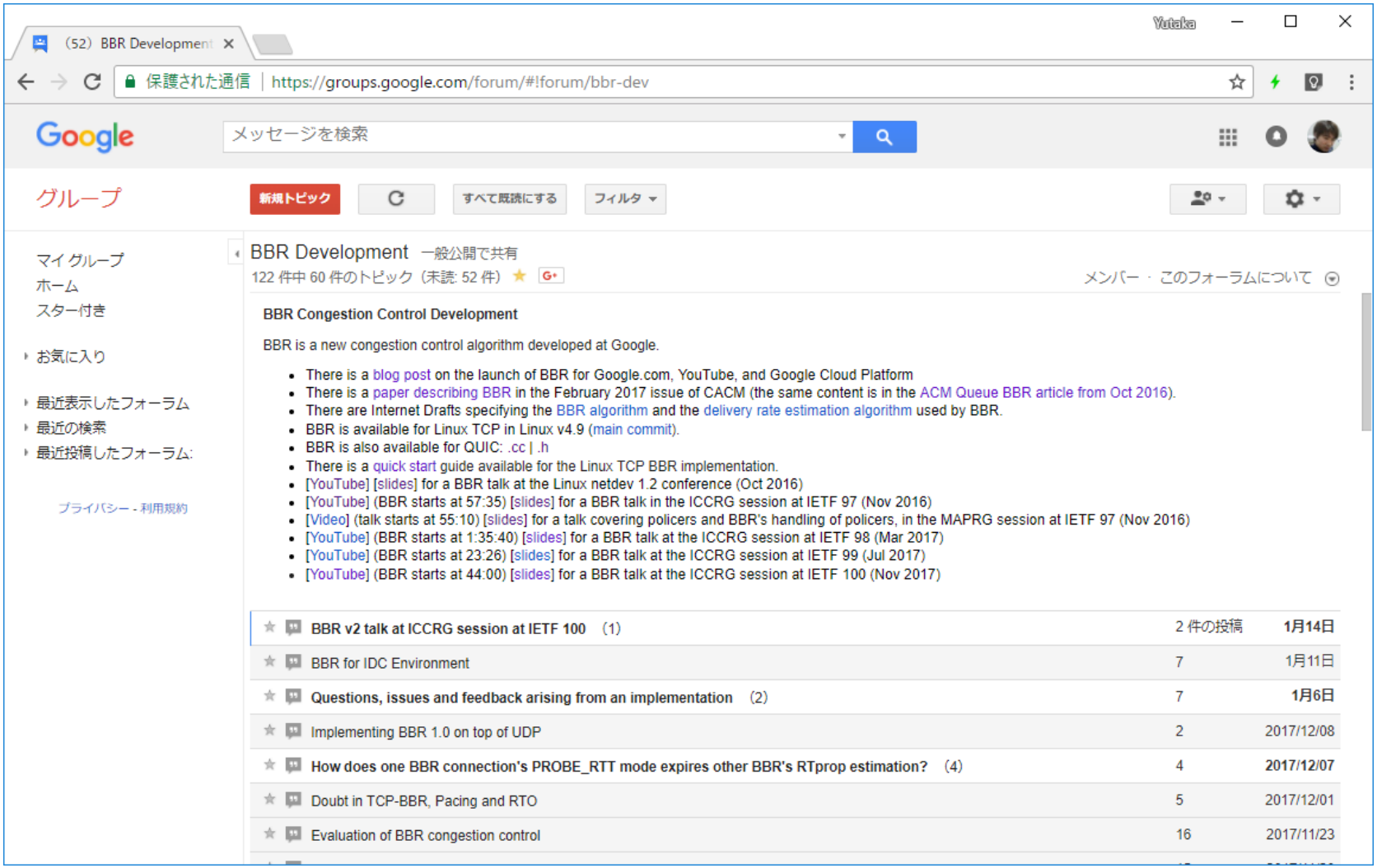




※ https://www.netdevconf.org/2.2/session.html?brakmo-tcpcongestionanalysis-talk

TCP BBR 第三者評価

■Google Groups - BBR Development



BBR v2.0

■IETF 100 ICCRG

■A quick BBR update: BBR in shallow buffers - Google

BBR Congestion Control:

IETF 100 Update: BBR in shallow buffers

Neal Cardwell, Yuchung C

C. Stephen Gunn, Soheil Hassa

Ian Swett, Jana Iyengar, Victo

Van Jacobson

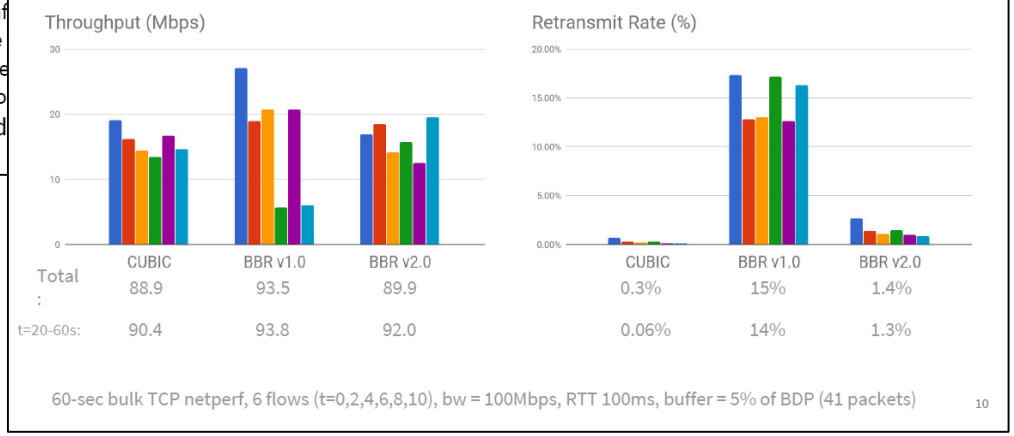
https://groups.google.com/d/fo

IETF 100: Singapore, Nov 13, 2017 Google

BBR v2.0: changes related to shallow buffers

- Goal: reduce queuing delay & packet loss, allow loss-based CC to maintain higher rates
- Generalized and simplified the long-term bandwidth estimator
  - Previously only targeted at policers
  - Now applied from start of any fast recovery until next bandwidth probe phase
  - Estimates long_term_bw = windowed average bandwidth over last 2-3 round trips
- New algorithm parameters to adapt to shallow buffers:
  - Max safe volume o
  - Volume of data wit
- New "full pipe+buffer"
  - Triggers if loss rate
  - Upon "full pipe+buf
    - Set estimate
    - Multiplicative
    - Before re-pro
- WIP: further work und

BBR in shallow buffers: before (v1.0) and after (v2.0)



※ IETF100 Singapore Nov,2017
ICCRGhttps://datatracker.ietf.org/meeting/100/session/iccrg

まとめ

■あたらしいTCP輻輳制御技術が多数出現
■途中のNWでは流れている通信のTCP輻輳制御アルゴリズムを把握できない

■ネットワーク運用者
　■管理しているネットワーク・QoSとの相性の確認をしてみては？

■サーバー運用者
　■品質向上のためにTCP輻輳制御の選定、開発を検討してみては？