HPCC++: Enhanced High Precision Congestion Control

draft-miao-tsv-hpcc-01

Rui Miao, Surendra Anubolu, Rong Pan, JK Lee, Barak Gafni, Y Shpigelman
Jeff Tantsura, Guy Caspray

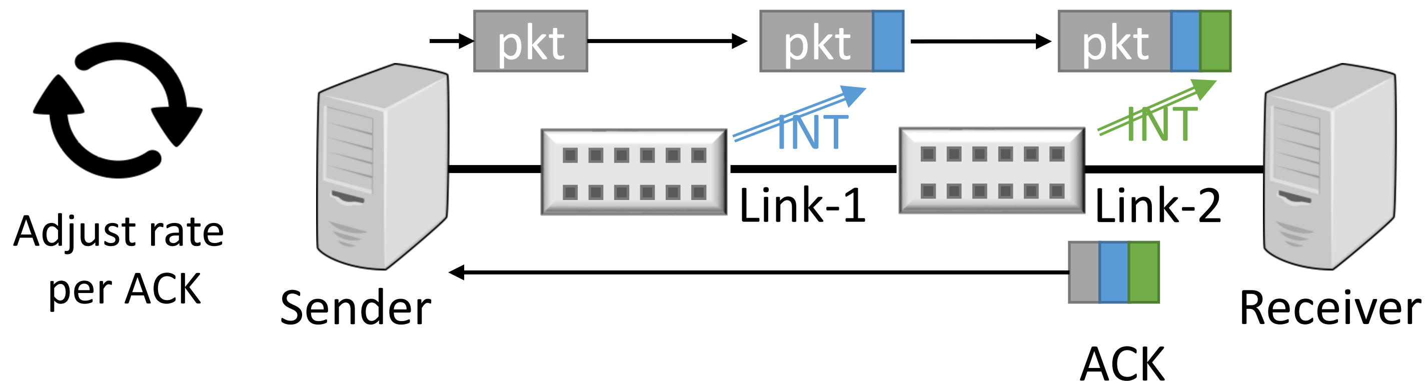
HPCC++: Gaining wide industry adoption

- Broadcom and Cisco joined HPCC++ IETF effort
- HPCC++ is supported by: Broadcom, Cisco, Intel, Nvidia silicon
- Good progress on alignment wrt encodings and meta-data across different technologies
- More deployment data showing benefits of HPCC++
  - Production deployment in Alibaba
- HPCC++ significantly enhances congestion control and drop prevention for:
  - storage and AI workloads requiring low latency/ high bandwidth
- HPCC++ is progressing with support for multi-q
- Algorithmic enhancements that show benefits of Receiver based HPCC++

- Metadata format captured in draft-miao-tsv-hpcc-info-01
- HPCC++ algorithm updated in draft-miao-tsv-hpcc-01.txt

Recap: High Precision Congestion Control (HPCC)

- Basic behavior
  - Each switch inserts INT information to packet headers
  - The receiver echoes INT information back to the sender via ACK packets
  - The sender adjusts its rate based on the INT information in ACKs



HPCC++ Deployment

Alibaba

- Alibaba Cloud has deployed HPCC++ for EBS, AI training, and database applications.
- The RDMA network running inside the storage cluster has reduced latency to about 30% on average and 80% in the tail.
- In EBS, HPCC++ is running in about 100 compute clusters 1000 storage clusters with in total of 100K servers.
    - The EBS bandwidth capacity is improved by about 30%
    - The average end-to-end latency is reduced by 10%.
- By enabling HPCC++ in our EBS cluster, Alibaba Cloud had managed to mostly eliminate congestion-triggered packet loss during the double 11 festival in 2021.

Interest from multiple operators for test deployment

Info draft for HPCC metadata format

IFA meta data format for HPCC++

- I-D.ietf-kumar-ippm-ifa

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  lns  |  deviceID                                  |   rsvd    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Speed |     rsvd      |           rxTimestampSec               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        egressPort           |          ingressPort             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        rxTimeStampNs                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        residenceTime                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           txBytes                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        rsvd                 |       Queue Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             rsvd                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

INT

- P4-INT

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Node ID (Nth hop)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Ingress Interface ID      |      Egress Interface ID      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Queue ID   |               Queue occupnacy                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Egress timestamp                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Egress timestamp (cont'd)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Egress interface Tx utilization               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Node ID (N-1th hop)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               ...                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Node ID (1st hop)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               ...                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IOAM metadata format

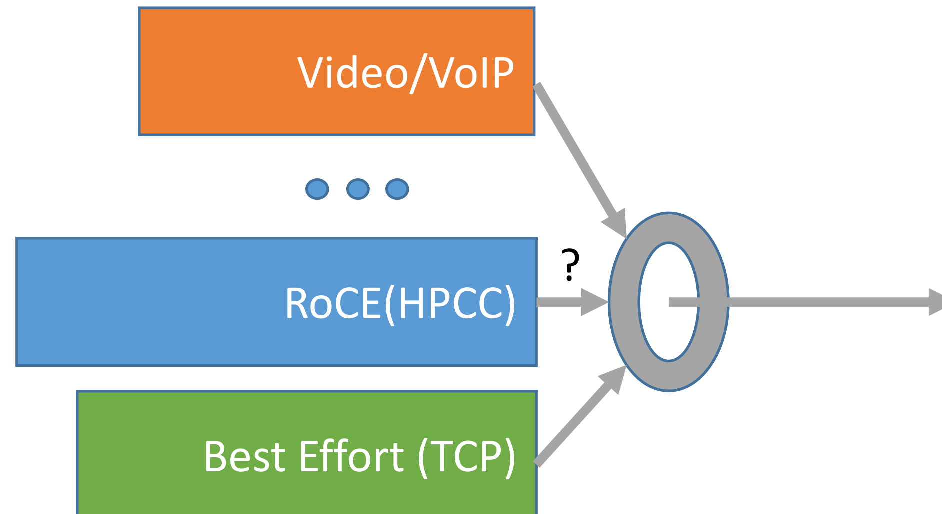
- I-D.draft-gafni-ippm-ioam-additional-data-fields

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Hop_Lim     |              node_id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ingress_if_id             |         egress_if_id          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       queue depth                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   timestamp fraction                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         tx_bytes                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MultiQ for HPCC++

How does HPCC++ co-exist with other traffic classes?

- HPCC assumes the knowledge of the link speed that is stamped into the telemetry packet headers, yet
  - In modern switches, all traffic classes share a specific link, and no class can take the whole link bandwidth
  - Each class can be guaranteed a minimum rate, but it can be served at any rate between the minimum rate and the link rate, depending on the traffic mix from other classes
  - How can we extend HPCC++ so that it can handle multi-queue sharing without the knowledge of other classes' traffic intensity?



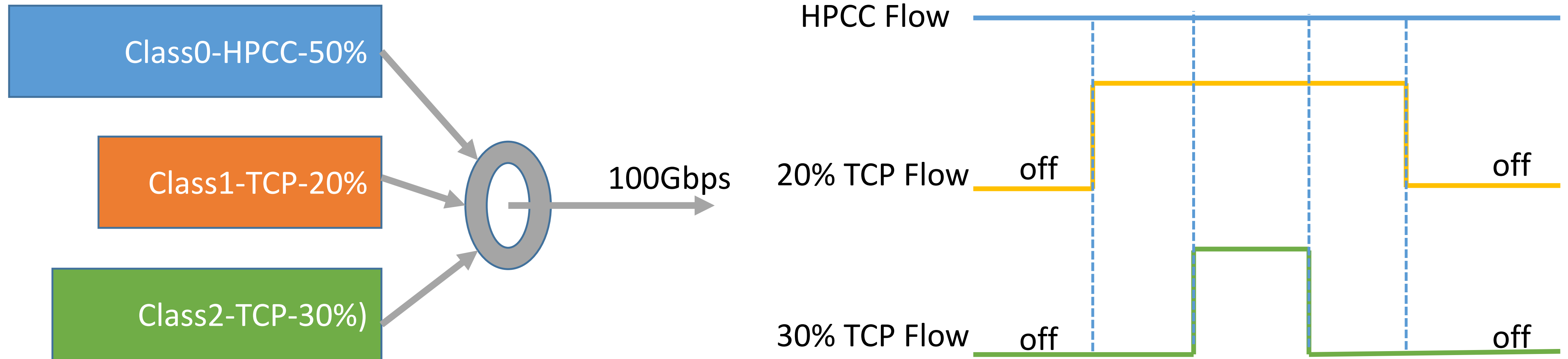
HPCC++ MultiQ Algorithm

- Parameter Definitions:
  - B: link speed
  - T: RTT
  - eta: the configured target link utilization
  - wB_j: the configured rate for Class j
  - txRate: the measured draining rate of a link
  - $txRate_j$: the measured draining rate of Class j at a specific link
  - qlen: the queue length at a specific link (assume one class in the original HPCC)
  - $qlen_j$: the queue length for Class j at a specific link
  - AI: HPCC's additive parameter

HPCC++ MultiQ Algorithm

- We assume switch calculate txRate and calculate U for HPCC as
  - $U = qlen/(B*T)+txRate/B$
- For multiQ extension:
  - If $txRate_j <= wB_j$: $U = qlen_j/(wB_j*T)+ txRate_j /wB_j$; //Paper's formula adapted for Class j that is sending at a guaranteed rate
  - Else if $qlen_j$!= 0 (or > small amount, e.g. 1.5KB): $U = qlen_j/(txRate_j*T) + 1$; //The draining rate is higher than the guaranteed rate and the queue is not empty, i.e. the allocated rate is fully utilized and there is a backlog, Paper's formula is adapted
  - Else: $U = txRate_j /((txRate_j + B)/2)$ //when the allocated rate is bigger than the guaranteed rate and there is no backlog, need to figure out what is the limit between the current rate and the maximum rate B, jump halfway between current_rate and link rate's target rate i.e. to binary-search the limit

HPCC++ MultiQ Simulation Setup*

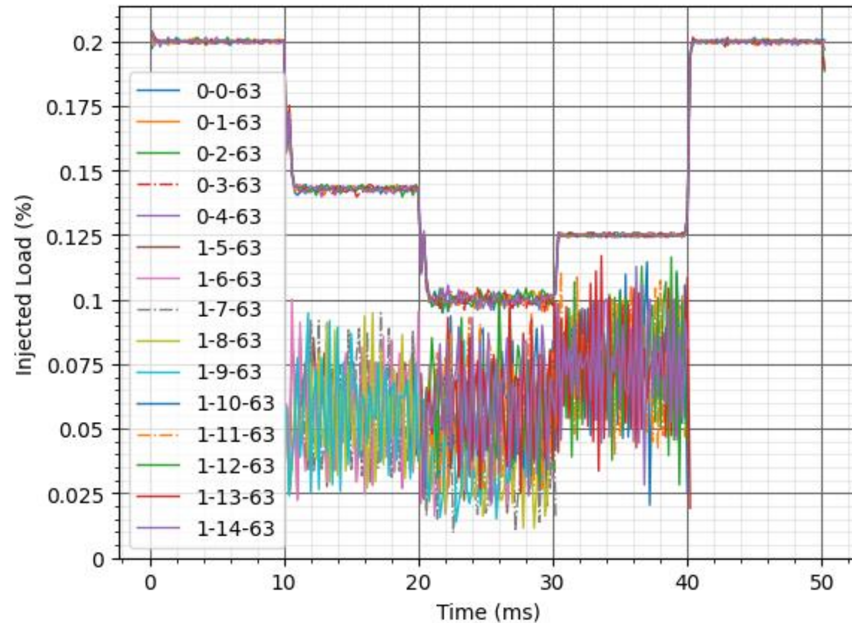


- 15-to-1 incast: 5 HPCC++ flows at TC=0, 5 TCP flows at TC=1 (TCP1) and 5 TCP flows at TC=2 (TCP2);
- AI = 0.005 (5.0Gbps); Queue length unit 1 pkt (1536 MTU); Line rate = 100Gbps

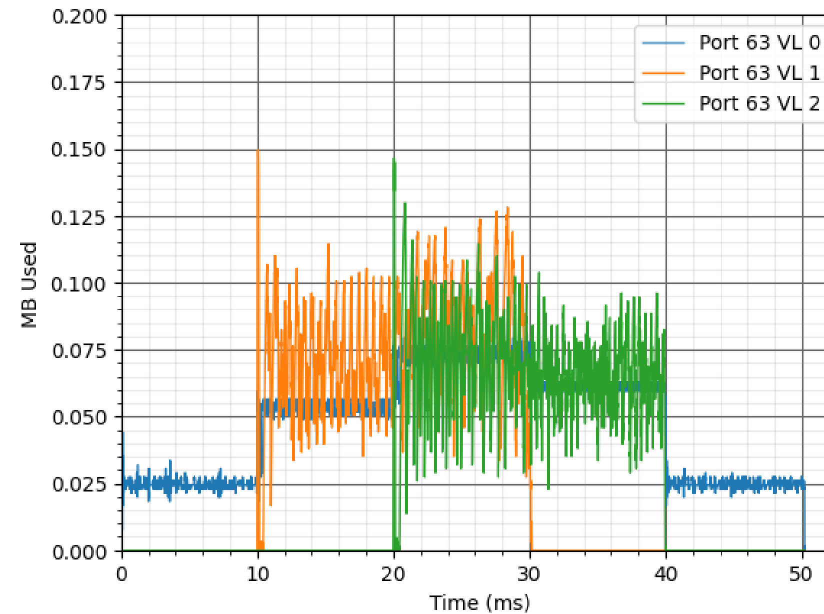
*We would like to thank Md Ashiwur Rahman and Roberto Penaranda from Intel for providing the simulation results

HPCC applied for MultiQ w/o any changes

Flow Rate



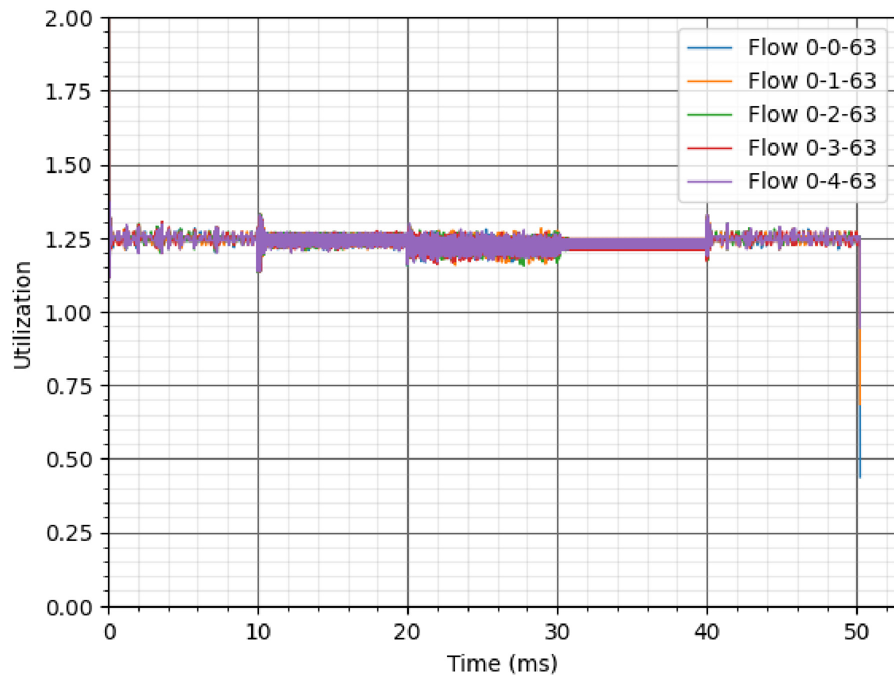
Memory used (per TC)



- HPCC calculated
  - U = qlen/(B*T)+txRate/B
  - If txRate is only a fraction of B, then qlen needs to be increased to make U approx. 1 in order to make HPCC work
  - The lower txRate is, the higher the queue length is

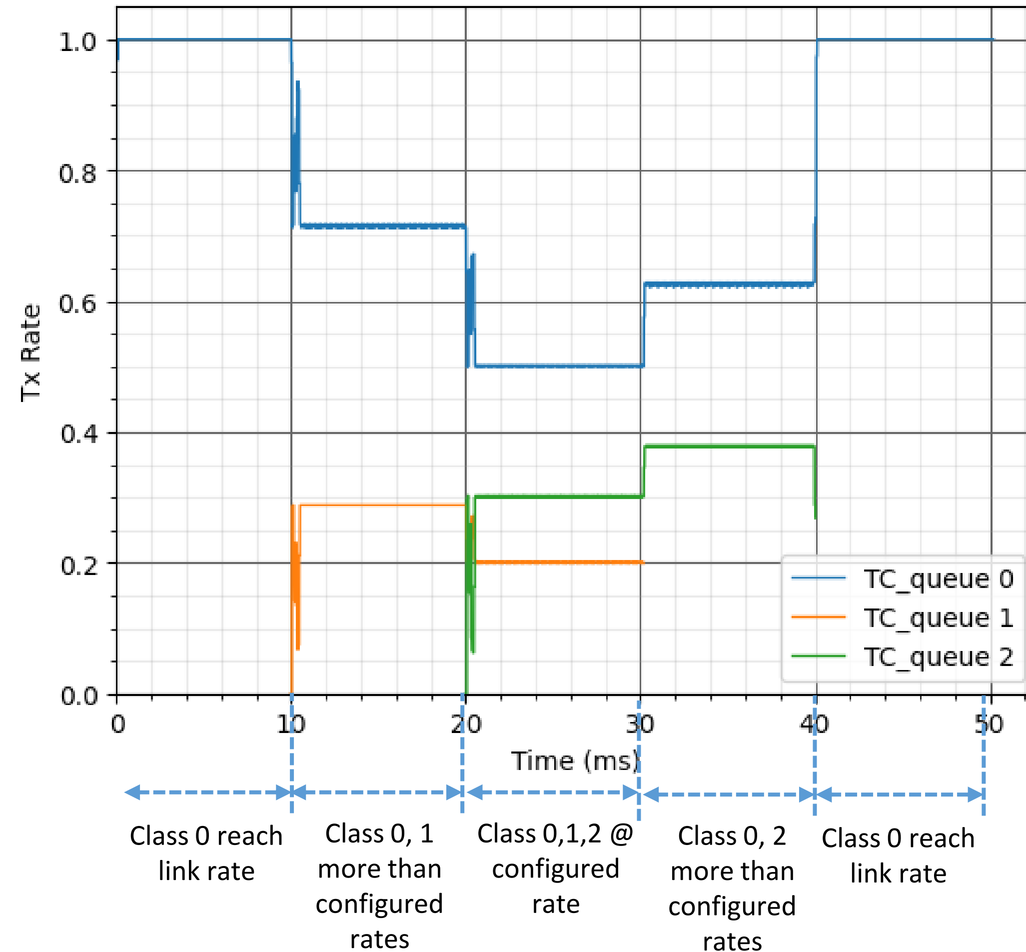
HPCC applied for MultiQ w/o any changes (2)

Utilization

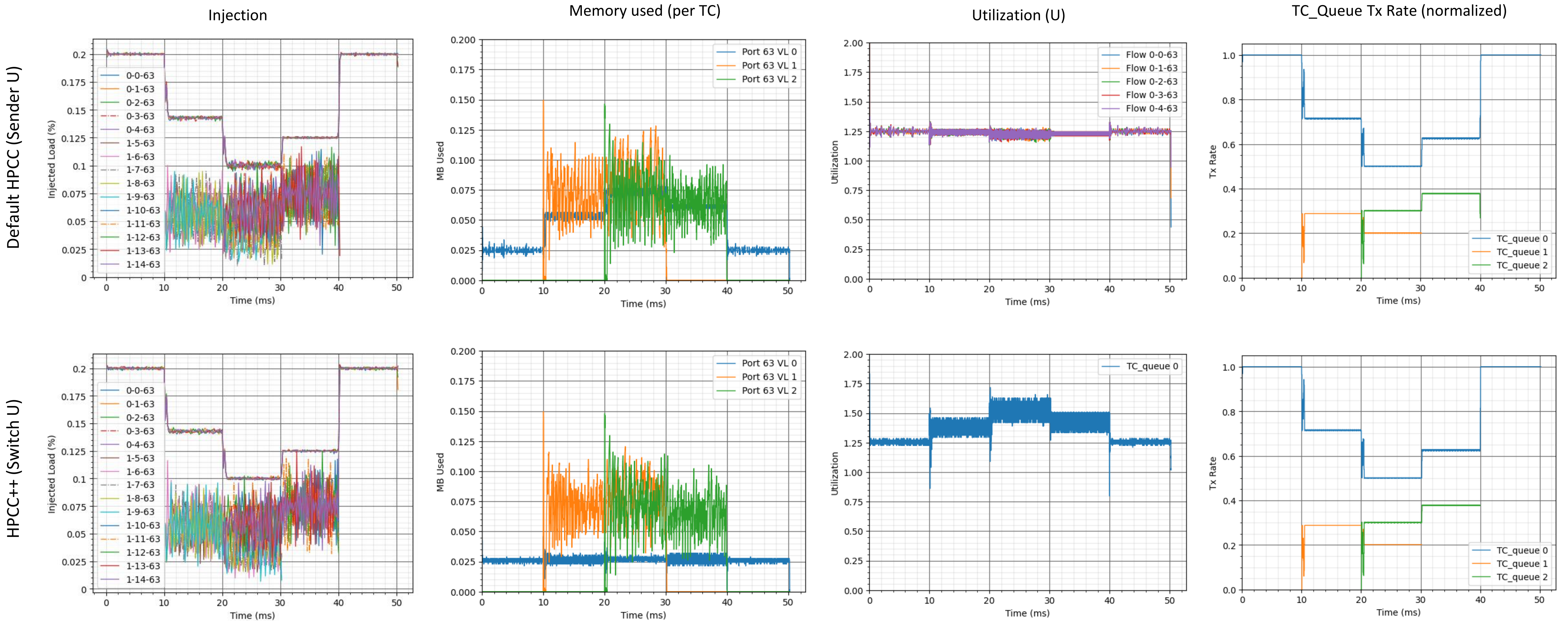


Utilization is maintained above 1 due to additive increase

TC_Queue Tx Rate (normalized)



HPCC vs MultiQ HPCC++ (Before and After)



By adjusting U properly, the queue length under HPCC++ class is maintained constant while each class's rate can achieve its fair rate

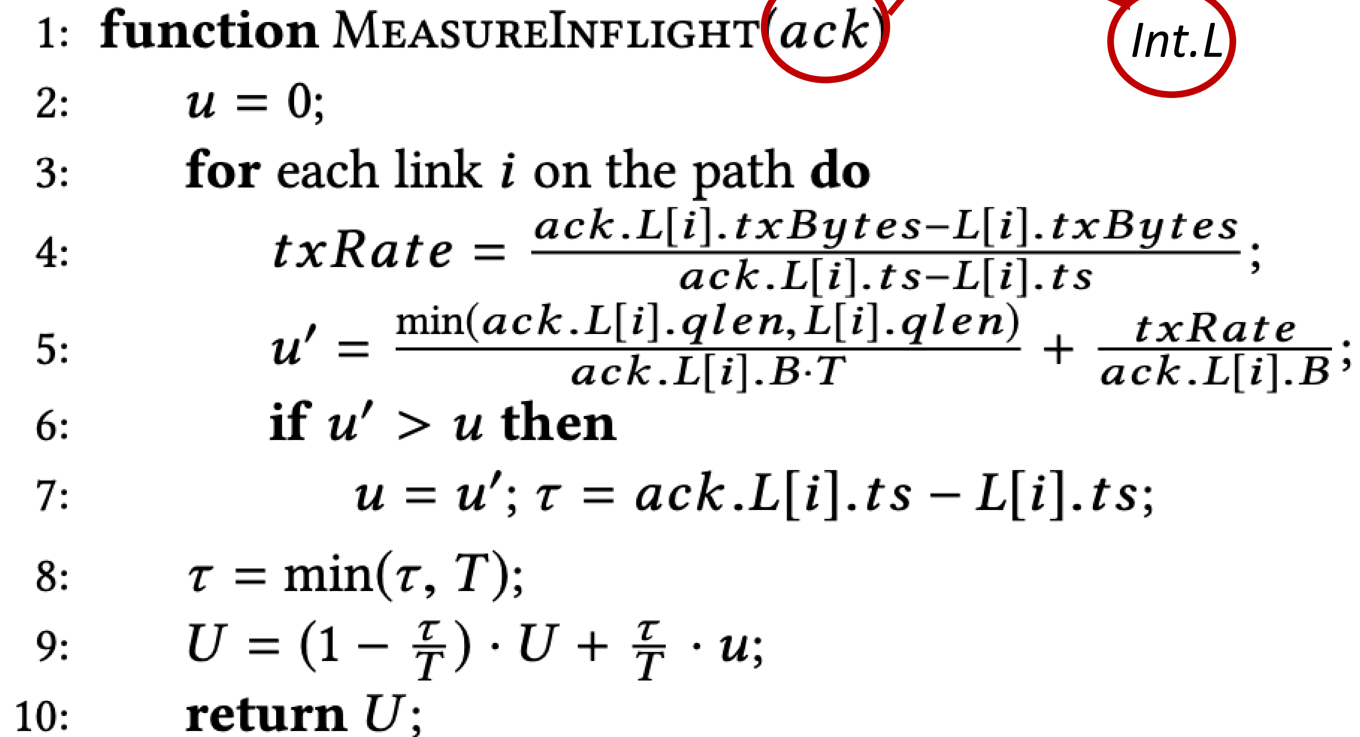
Receiver based HPCC

Receiver-based HPCC

- Sender’s rate is calculated in the receiver side
  - Receiver is aware of all incoming traffic
  - ack.L is one-to-one mapping of int.L

- Every RTT or when a sudden congestion level change, a notification packet (np) is sent for each flow
  - Updating the injection rate
  - Acknowledging the received packets, thus eliminating the need for ACKs

HPCC Design (1) → Rx-HPCC Design (1)



Experiments

Scenarios

- 2 to 1 in-cast
- 8 to 1 in-cast
- Flow come and go

- Goal
  - Performance comparation:
    - Injection bandwidth.
    - Output queue depth.
    - Fairness among flows.

Experiments

Configuration based on original HPCC paper

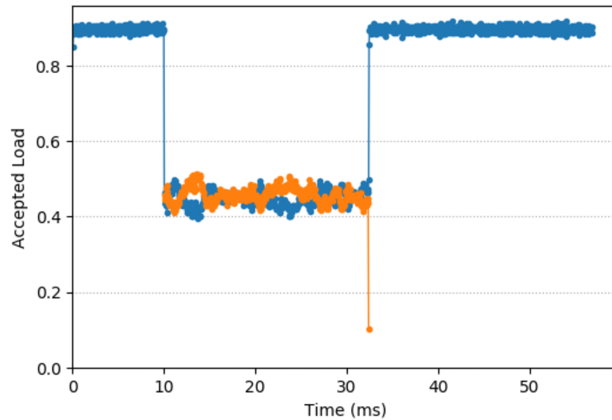
- Network: 2-stage fat tree
- NICs: 25 Gbps (0.025 flits per ns)
- SW: 100 Gbps
- RTT: 8.5 us

- HPCC knobs
  - Max stage = 5
  - Base RTT = 9 us
  - Target utilization (T_uti) = 0.95
  - Additive increase (W_AI)
    - 1.25e-4 flits per ns (128 Mbps) (N = 10)
    - 1.25e-5 flits per ns (12.8 Mbps) (N = 100)
    - Dynamic
  - Change rate threshold = 25%
  - Notification period:
    - 4.5us

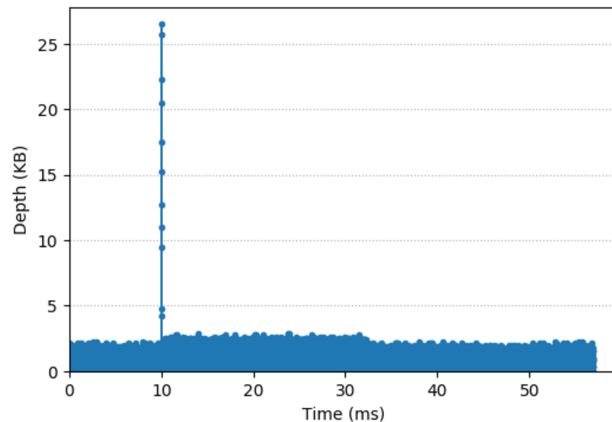
Experiments: In-cast 2, Target_utilization (η) 95%

- Original HPCC

W_AI 1.25 e-4

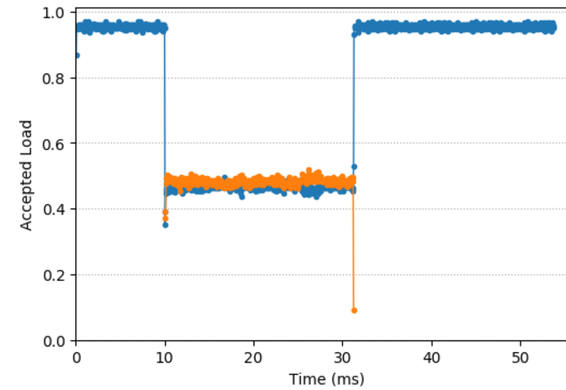


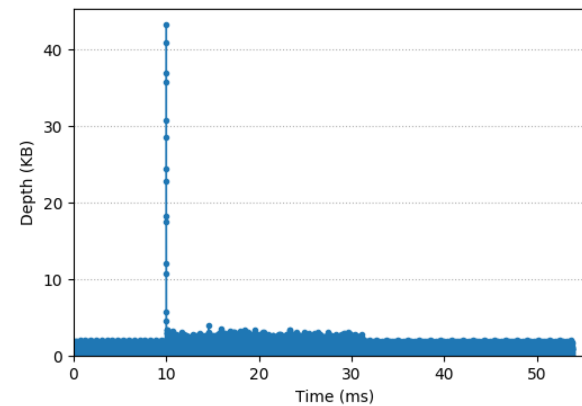
Number of ACKs:
114924



Rx-based HPCC (T = 4.5us)

Dynamic W_AI





Number of NPs:
21913

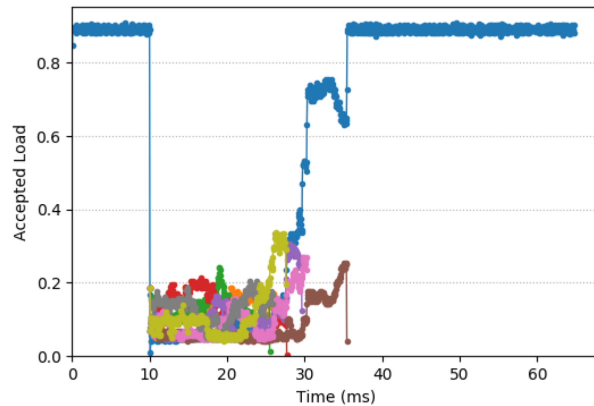
- Much less feedbacks
- Similar transient behavior
- Improved fairness
- Slightly bigger buffer usage

Experiments: In-cast 8, Target_utilization (η) 95%

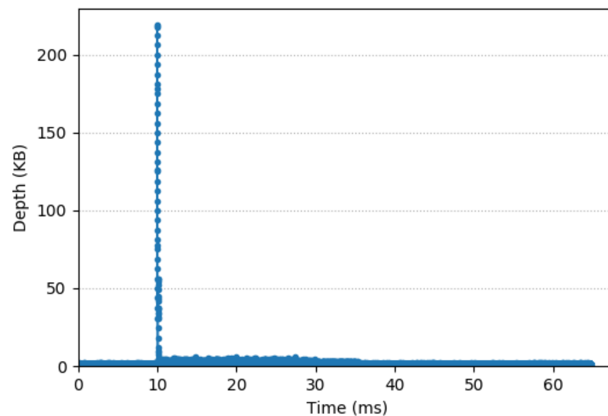
- Original HPCC

Rx-based HPCC (T = 4.5us)

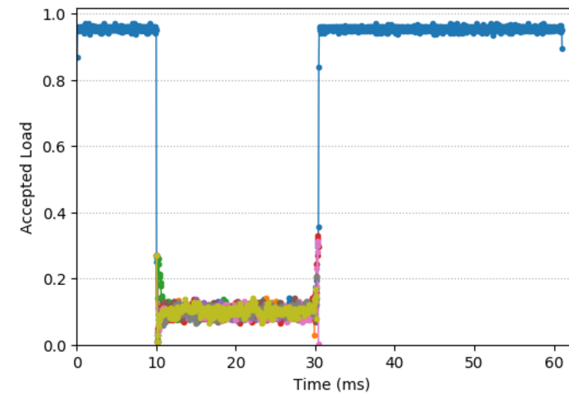
W_AI 1.25 e-5

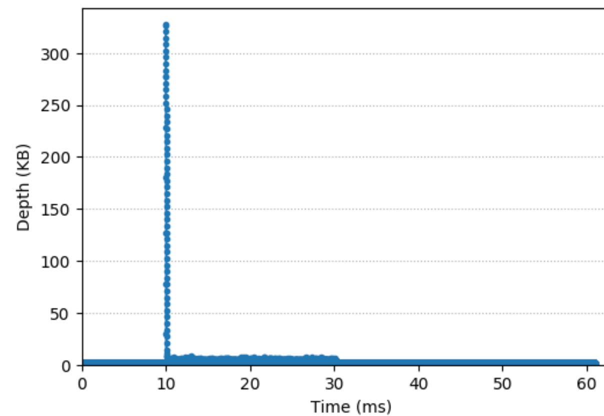


Number of ACKs: 128724



Dynamic W_AI





Number of NPs: 45249

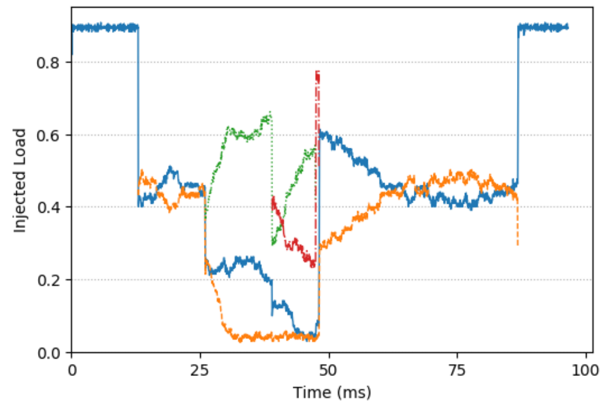
Similarly:

- Much less feedbacks
- Similar transient behavior
- Improved fairness
- Slightly bigger buffer usage

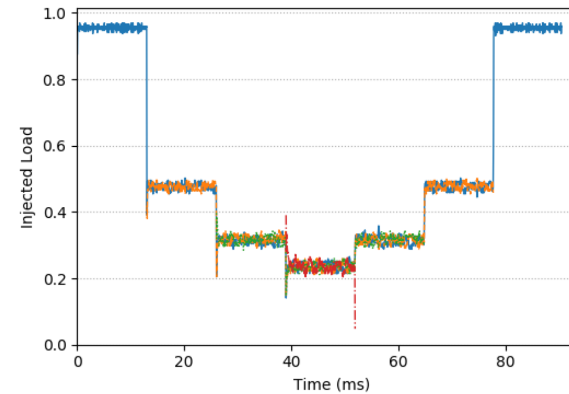
Experiments: Flow Come and Go, Target_utilization (η) 95%

- Original HPCC Rx-based HPCC (T = 4.5us)

W_AI 1.25 e-5

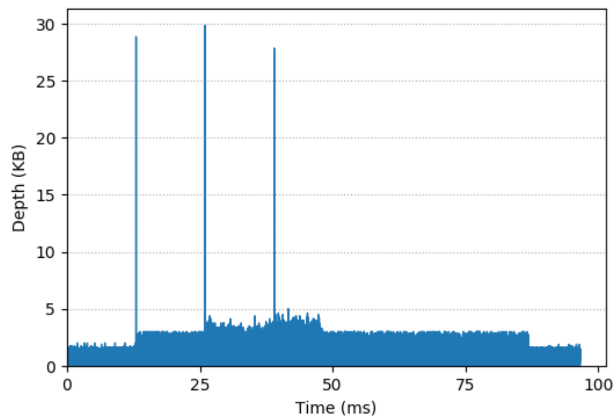


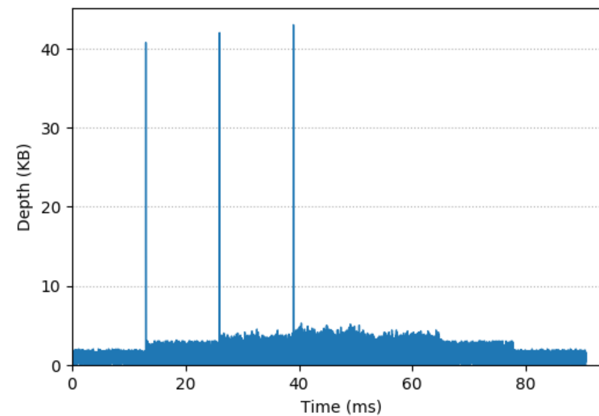
Dynamic W_AI



Number of NPs:
110940

Number of ACKs:
193116





Similarly:

- Much less feedbacks
- Similar transient behavior
- Bigger buffer usage
- No need to configure W_AI

Thank you