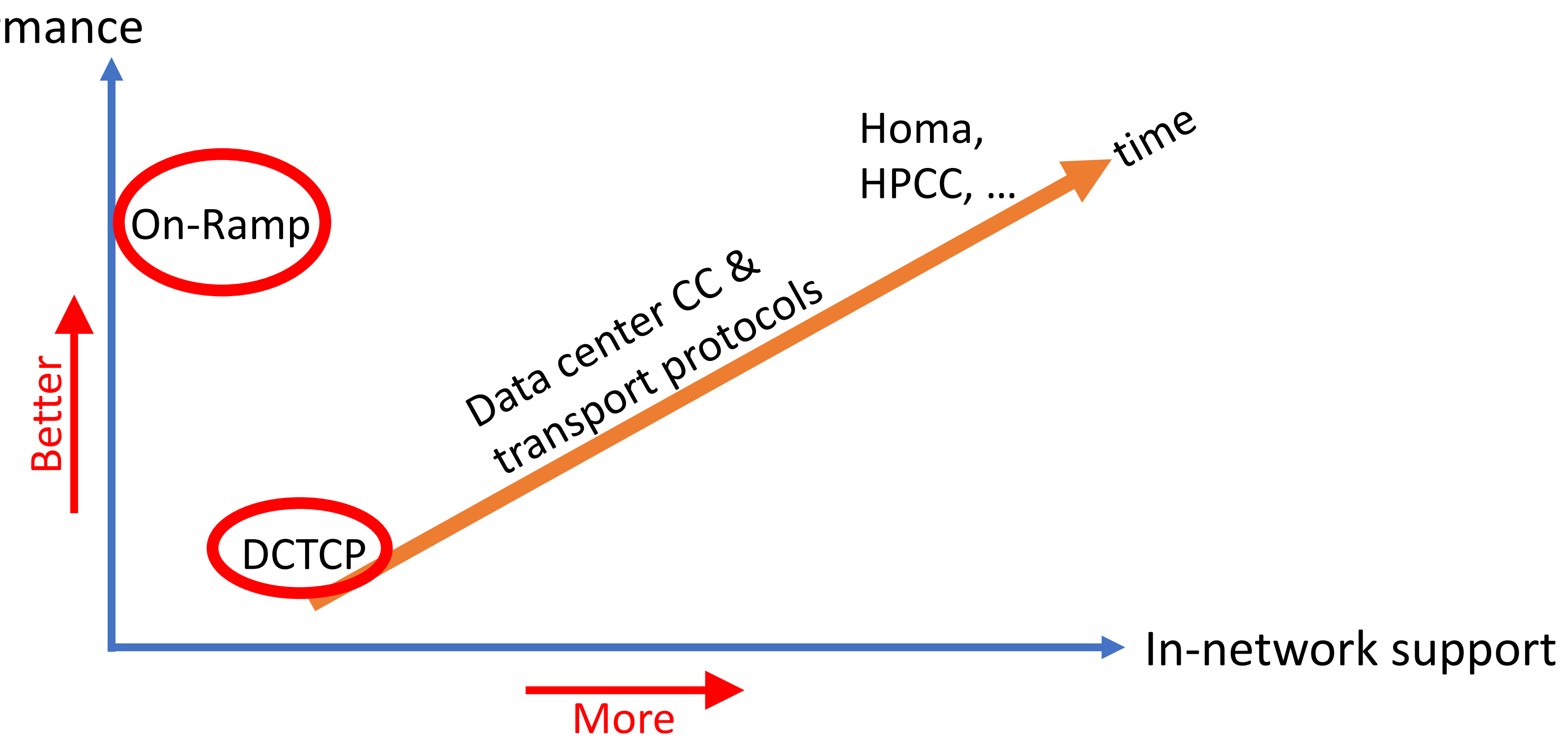
Breaking the Transience-Equilibrium Nexus: A New Approach to Datacenter Packet Transport

Shiyu Liu, Ahmad Ghalayini,
Mohammad Alizadeh*, Balaji Prabhakar, Mendel Rosenblum, Anirudh Sivaraman+

Stanford University *MIT +NYU

April 12, 2021

Evolution of Data Center Congestion Control (CC) Algorithms & Transport Protocols



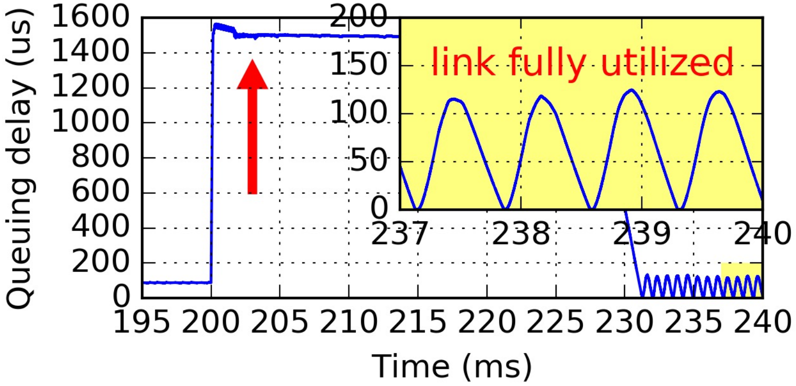
- On-Ramp: a simple mechanism that ***cloud users*** can deploy on their own to improve performance, with ***no in-network support***

Our method

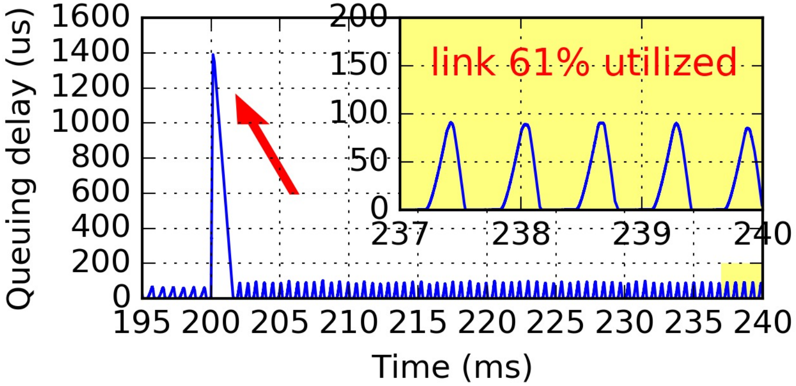
- Focus primarily on detecting and handling transient congestion
  - Most CCs perform well in the long term: high throughput, fairness, etc.
  - Transient, like incast, is difficult to handle since senders must react very ***quickly*** and ***forcefully*** to prevent packet drops
    - which is in conflict with the stable convergence of CC

Why decoupling the handling of transience and equilibrium?

12 servers send TIMELY long flows to 1 server.
2 flows start at t=0. The other 10 flows start at t=200ms.



$\beta = 0.2$



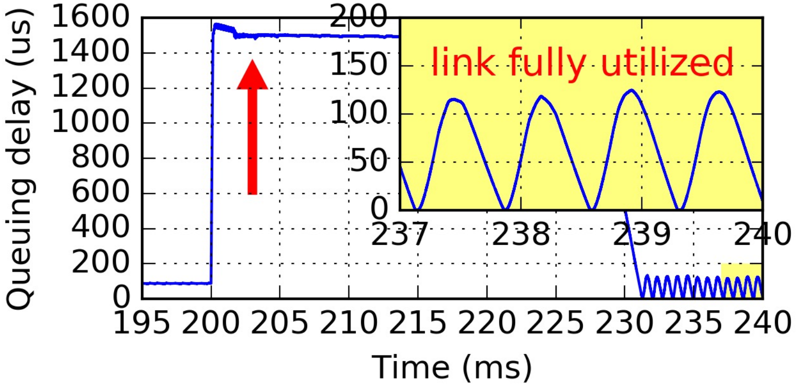
$\beta = 0.8$

Transience-equilibrium tension

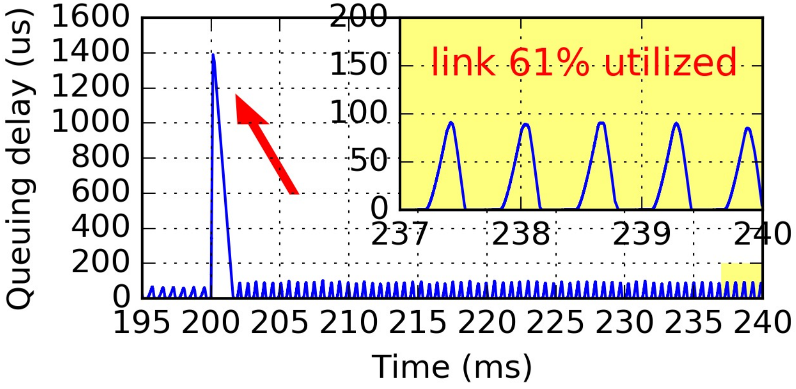
Without rich congestion signals, there is a strong trade-off between
a packet transport's equilibrium & transience performance.

Why decoupling the handling of transience and equilibrium?

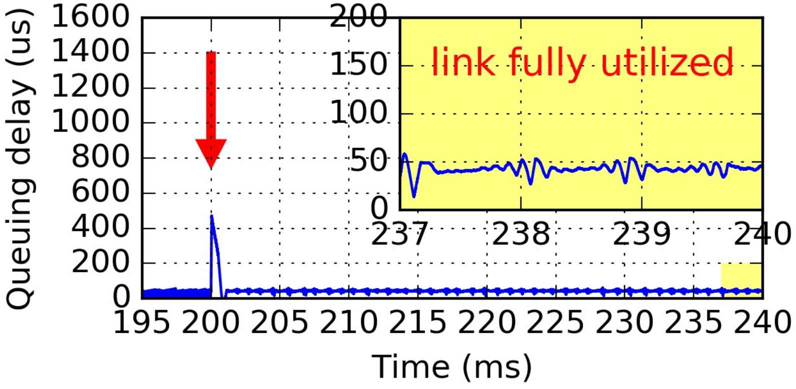
12 servers send TIMELY long flows to 1 server.
2 flows start at t=0. The other 10 flows start at t=200ms.



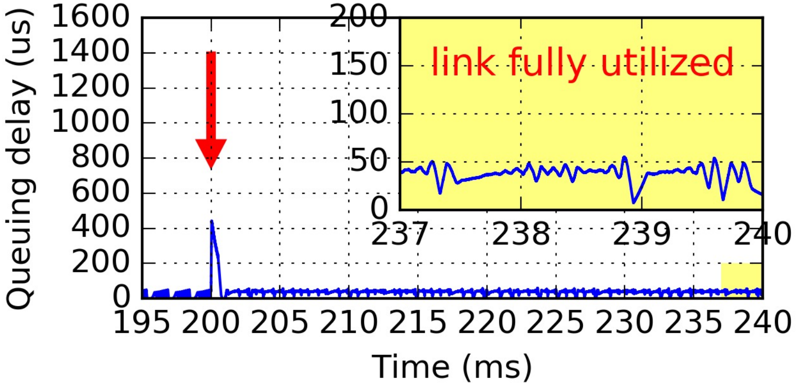
$\beta = 0.2$



$\beta = 0.8$



$\beta = 0.2$, OR threshold $T = 30\mu s$

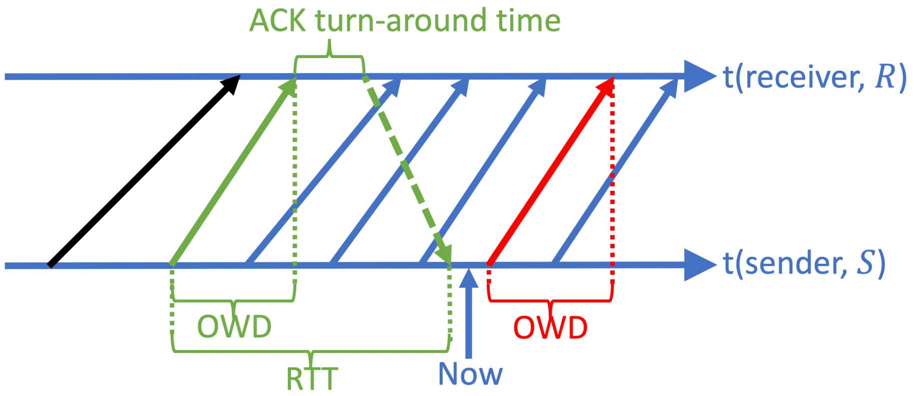


$\beta = 0.8$, OR threshold $T = 30\mu s$

- On-Ramp reacts quickly and forcefully to transient congestion, while still keeps the stable convergence during equilibrium.
- On-Ramp also enhances the performance of existing CC in equilibrium
  - by making them more robust to the choice of CC algorithm parameters (gain)
  - How? On-Ramp compacts the state space in the network
    - More in the paper…

Our proposal of On-Ramp

- On-Ramp: if the one-way delay (OWD) of the most-recently acked packet > threshold T, the sender temporarily holds back the packets from this flow.
  - Pause: reduce the path queuing delays as quickly as possible
  - A gate-keeper of packets at the edge of the network
  - Decoupling transience from equilibrium congestion control
  - Clock sync (e.g. Huygens) makes OWD measurement possible



- Can be coupled with existing CCs, requires only end-host modifications.

- In addition to public cloud, On-Ramp can also improve network-assisted CC.

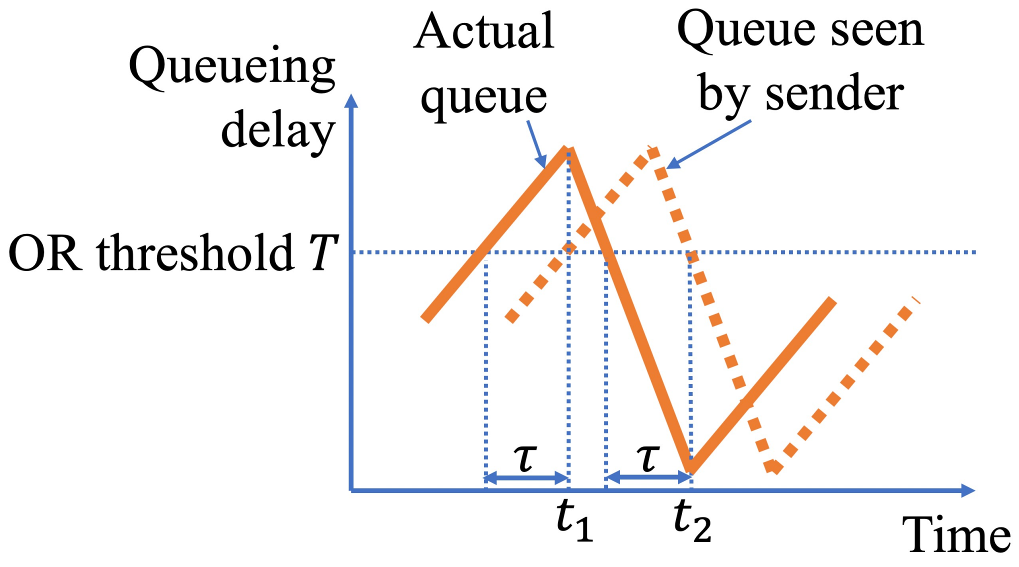
Image source: https://streets.mn/2014/06/02/metering-motoring/

Outline

- **Design**
  - **Strawman proposal**
  - **Final version**
- Implementation
- Evaluation
  - Google Cloud
  - CloudLab
  - ns-3
  - Facebook cluster

Strawman proposal for On-Ramp

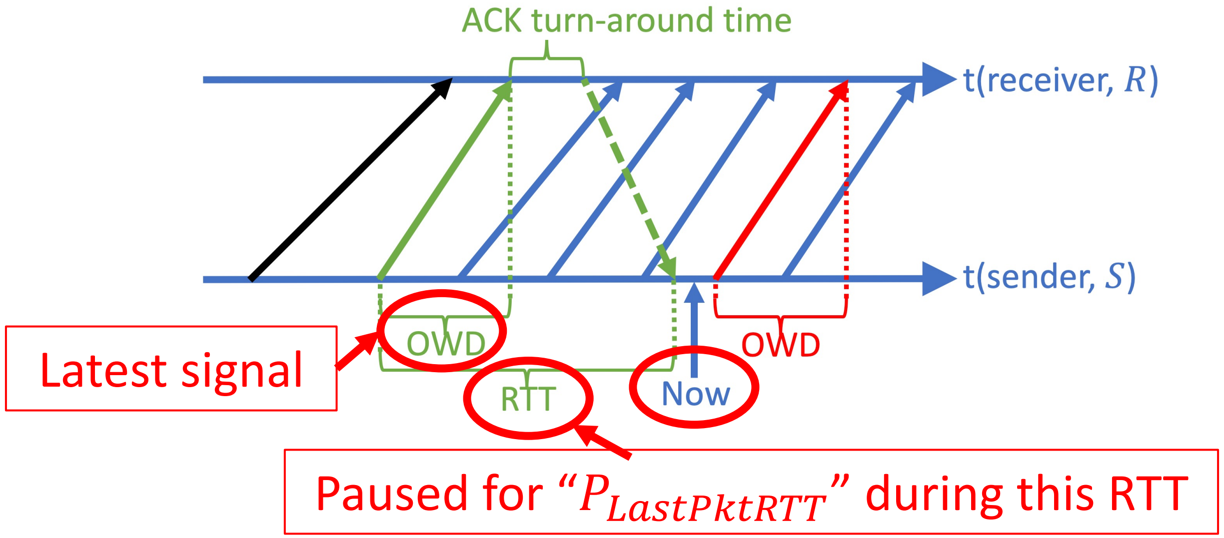
- For a flow, if the measured $OWD > T$, the sender pauses this flow until $t_{Now} + OWD - T$
- Hope: drain the queue down to T
- With feedback delay τ: pause much longer than needed
  - Queue undershoots T
  - May cause under-utilization



Final version of On-Ramp

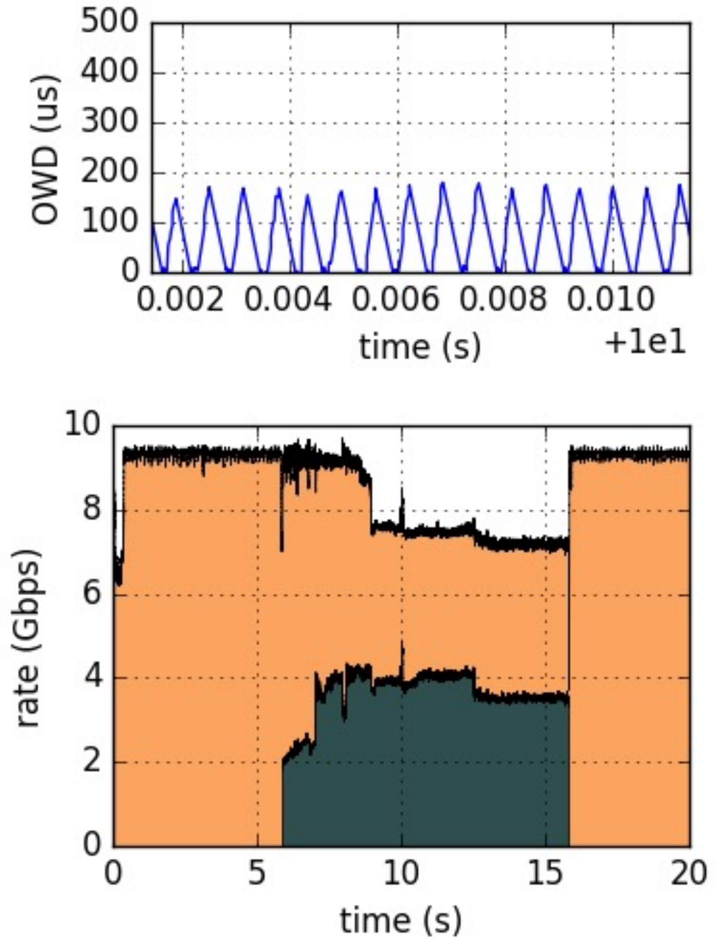
- Need to pause less. Two factors to consider:
  - **Feedback delay**: it is possible the sender also paused this flow when the green pkt was in flight, but the latest signal "OWD of the green pkt" hasn't seen the effects of these pauses
  - **Concurrency**: to account for the contributions to OWD from ***other*** senders

- The rule of pausing needs to account for these
  - Details in the paper…

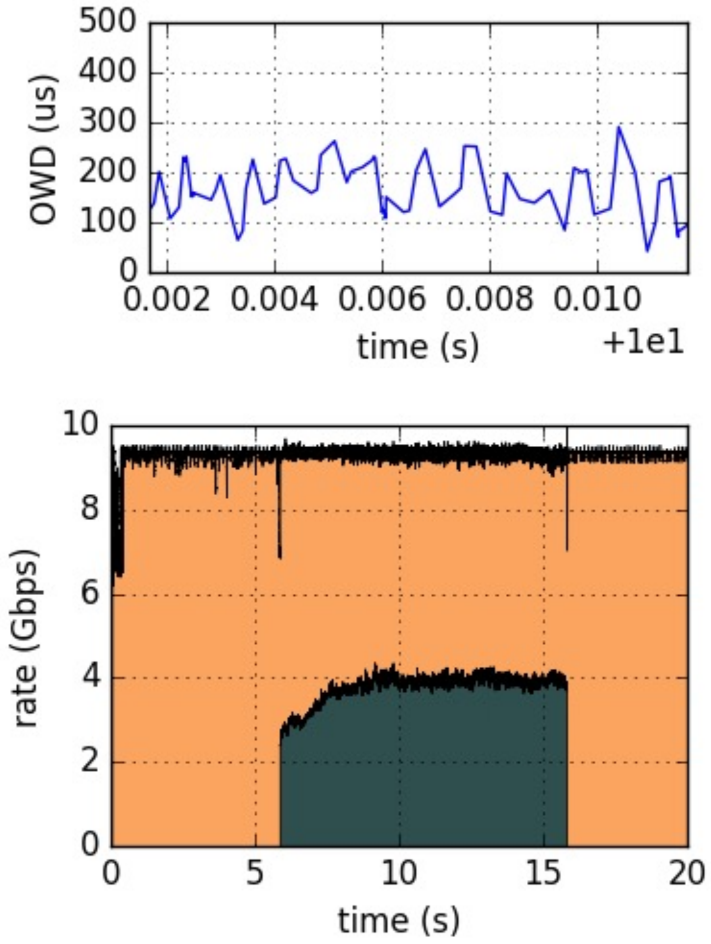


Two long-lived CUBIC flows sharing a link

Strawman On-Ramp



Final version of On-Ramp

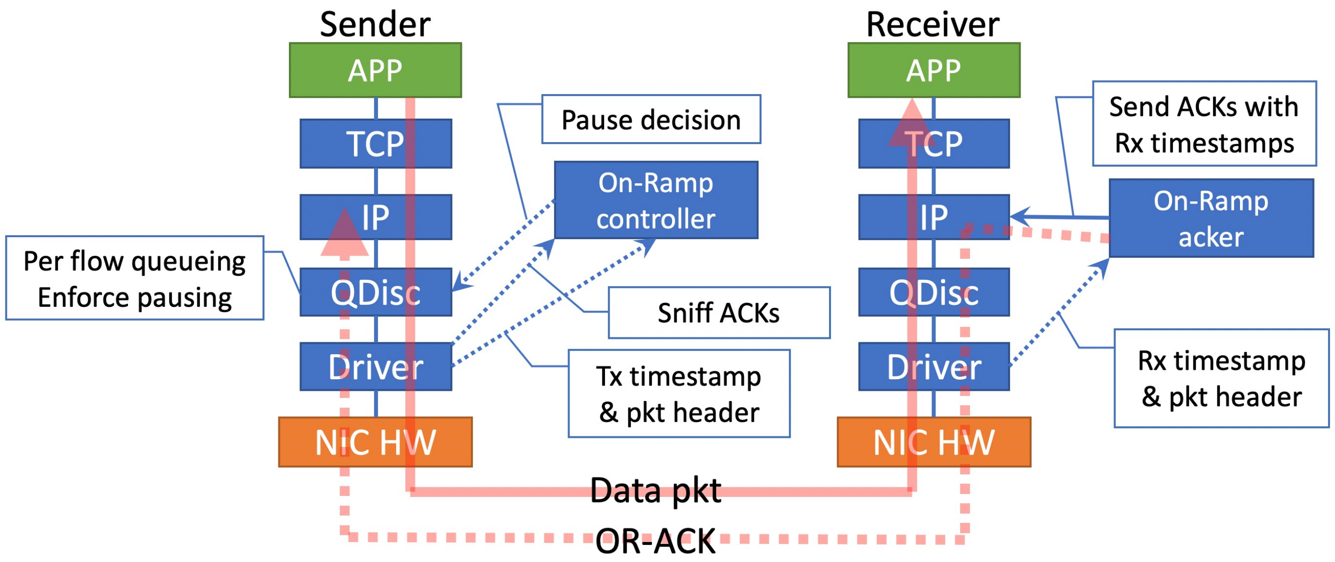


Outline

- Design
  - Strawman proposal
  - Final version
- **Implementation**
- Evaluation
  - Google Cloud
  - CloudLab
  - ns-3
  - Facebook cluster

Implementation

- Linux kernel modules
  - End-host modifications only
  - Easy to deploy
- ns-3
  - Emulate the NIC implementation
  - Built on top of the open-source HPCC simulator



Outline

- Design
  - Strawman proposal
  - Final version
- Implementation
- **Evaluation**
  - **Google Cloud**
  - **CloudLab**
  - **ns-3**
  - **Facebook cluster**

Evaluation Highlights (more in the paper)

- Traffic loads:
  - **Background**: WebSearch, FB_Hadoop, GoogleSearchRPC, load = 40% ~ 80%
  - **Incast**: Fanout=40, each flow=2KB or 500KB, load = 2% or 20%

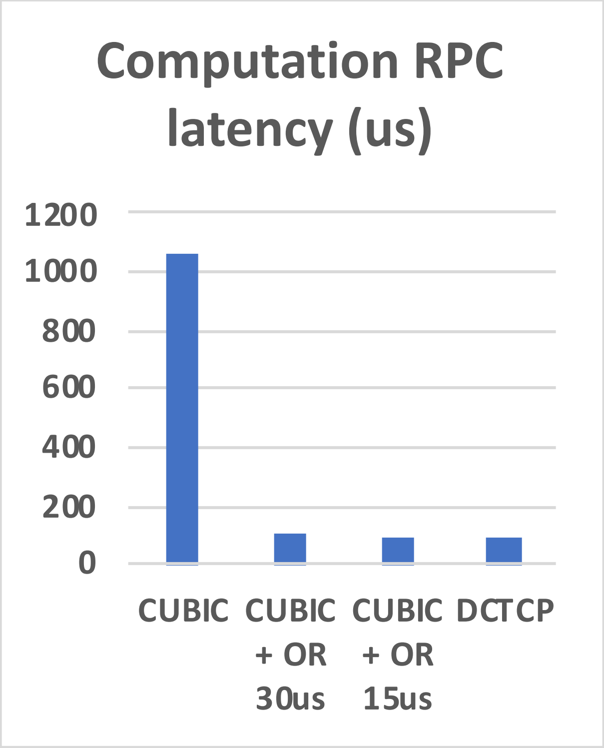
- **VMs in Google Cloud** (50 VMs)
  - On-Ramp improves the 99% request completion time (RCT) of incast traffic of CUBIC by 2.8× and BBR by 5.6×
- **Bare-metal cloud in CloudLab** (100 servers)
  - On-Ramp improves the 99% RCT of CUBIC by 4.1×
- **ns-3 simulation** (320 servers)
  - On-Ramp improves RCTs to varying degrees depending on the workload under DCQCN, TIMELY, DCTCP and HPCC

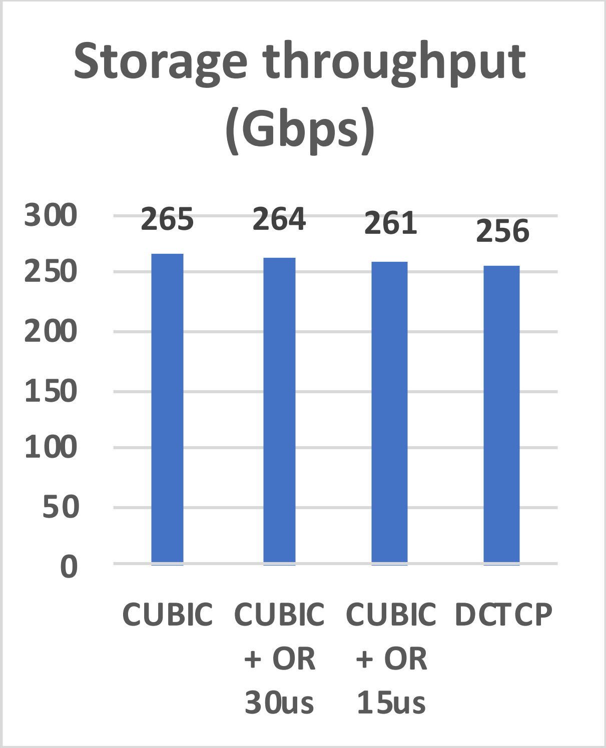
- In all three environments
  - On-Ramp also improves the flow completion time (FCT) of non-incast background traffic

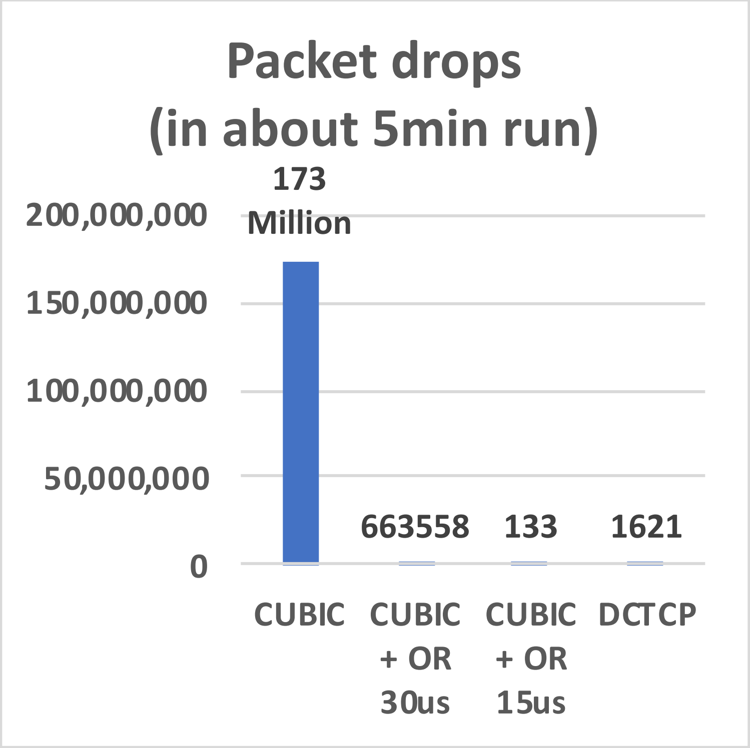
Evaluation in Facebook: Highlights

(more in the paper)

- Two racks in a Facebook production cluster
- Traffic loads: Computation traffic (RPC-type) + Storage traffic (NVMe-over-TCP)







More in the paper

- The importance of using one-way delay vs. round-trip time
- Network and CPU overhead of On-Ramp
- Co-existence of On-Ramp and non-On-Ramp traffic
- The granularity of control by On-Ramp
- ……

Conclusion

- On-Ramp allows ***public cloud users*** to take cloud network performance into their own hands
  - No need to change either the VM hypervisor or the network infrastructure
  - Can couple with existing congestion-control algorithms
- On-Ramp's improvements hold even in more customizable environments like an on-prem cloud or a cloud with SmartNICs
- On-Ramp contains two ideas:
  - Using synced clocks to improve network performance
  - Decoupling the handling of transience & equilibrium