On-Ramp: Using Precise Packet Delays to Pause Flows at the Network’s Edge

Shiyu Liu, Ahmad Ghalayini,
Mohammad Alizadeh*, Balaji Prabhakar, Mendel Rosenblum, Anirudh Sivaraman+
Stanford University  *MIT  +NYU
February 13, 2020
Congestion Management: Background

• Over the past decade, many congestion-control (CC) algorithms and transport protocols have been developed:
  1. Signals from switches: ECN (DCTCP, DCQCN), INT (HPCC)
  2. Network support like packet scheduling (pFabric, PIAS, QJump, TIMELY, pHost, Homa) or packet trimming (NDP)

  ➢ They show impressive performance in on-prem data centers

• But they cannot be deployed by *cloud users* because they require a variety of network mechanisms to detect and signal congestion.

• Existing solutions available to cloud users (like CUBIC) incur significant performance penalties in datacenters.
  • Especially under incast-type loads
Our goal

• Develop a simple mechanism that *cloud users* can deploy on their own to improve performance, with *no in-network support*.

• 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
Handling Transience in the Cloud

• Transience, like incast, is difficult to handle since senders must react very \textit{quickly} and \textit{forcefully} to prevent packet drops
  • For drop-based CC, the signal is too late.
  • For ECN/delay-based CC, the best way is sudden, forceful throttling
    \begin{itemize}
      \item[] but it’s in conflict with the stable convergence of CC
    \end{itemize}

• What people do now:
  • buffer headroom (DCTCP, DCQCN)
  • bandwidth headroom (HULL, HPCC)
  • Priority-based Flow Control (PFC) (borrow buffers from upstream switches)

• But these in-network resources are unavailable in the public clouds
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.
  • A gate-keeper of packets at the edge of the network.
  • Decoupling the handling of transience & equilibrium.

• Can be coupled with any CC. Require only end-host modifications.

• In addition to public cloud, On-Ramp can also improve network-assisted CC.
**Why OWD?**

OWD of the next pkt as a function of: the OWD(Huygens), RTT, and OWD(NTP) of the last-acked pkt
Outline

• Design
  • Strawman proposal
  • Final version

• Implementation

• Evaluation
  • Google Cloud
  • CloudLab
  • ns-3

• Deep Dive
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 $\tau$: pause much longer than needed
  • Queue undershoots $T$
  • May cause under-utilization

![Diagram showing the queueing delay and actual queue with threshold $T$. The queue seen by the sender and the OR threshold are also indicated.](image-url)
Final version of On-Ramp

• Need to pause less. Two factors to consider:
  • **Feedback delay**: it is possible the sender paused this flow when the green pkt was in flight and before the sender received its ack
  • **Concurrency**: to account for the contributions to OWD from other senders

  • The rule of pausing needs to account for these.
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

• Deep Dive
Implementation

- Linux kernel modules
  - End-host modifications only.
  - Easy to deploy. Hot-pluggable.
  - Incremental deployment is possible.

- 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

• Deep Dive
Evaluation Setup

• Environments:
  • **VMs in Google Cloud**: 50 VMs, each has 4 vCPUs and 10G net.
  • **Bare-metal cloud in CloudLab**: 100 machines across 6 racks, 10G net.
  • **ns-3**: 320 servers in 20 racks, 100G net.

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

• Clock sync:
  • Huygens for Google Cloud and CloudLab

![Distribution of flow sizes in the background traffic](image)
On-Ramp in Google Cloud

- CUBIC
- WebSearch @ 40% load + incast @ 2% load (fanout=40, each flow 2KB)
**On-Ramp in Google Cloud**

- **BBR**
- WebSearch @ 40% load + incast @ 2% load (fanout=40, each flow 2KB)

**Incast RCT**

**FCT of WebSearch traffic**
On-Ramp in CloudLab

- CUBIC
- WebSearch @ 60% load + incast @ 2% load (fanout=40, each flow 2KB)
On-Ramp with Network-assisted CC (ns-3)

- WebSearch @ 60% load + incast @ 2% load (fanout=40, each flow 2KB)
- Bars: mean. Whiskers: 95th percentile
On-Ramp with Network-assisted CC (ns-3)

- GoogleSearchRPC @ 60% load + incast @ 2% load (fanout=40, each flow 2KB)
- Bars: mean. Whiskers: 95th percentile

GoogleSearchRPC:
~99.85% of flows, ~80% of bytes come from flows < 10KB
Outline

• Design
  • Strawman proposal
  • Final version
• Implementation
• Evaluation
  • Google Cloud
  • CloudLab
  • ns-3
• Deep Dive
  • Interaction of On-Ramp with congestion control
  • The granularity of control
  • Co-existence
Deep-dive 1: Interaction of On-Ramp with CC

10 servers send DCQCN long flows to 1 server. All flows start at t=0.

- **Transience:** On-Ramp helps to suppress queue overshoots much faster.
- **Equilibrium:** DCQCN is still stable. Queue oscillations are even smaller when $T = 40$ or $30\mu s$. 
Deep-dive 2: The Granularity of Control

- On the sender side, General Segmentation Offloading (GSO) affects the granularity of control by On-Ramp.
- Reducing max GSO size further improves performance but with higher CPU overhead.

Google Cloud, CUBIC, WebSearch @ 40% load + incast @ 2% load (fanout=40, each flow 2KB)
Deep-dive 3: Co-existence

• The Google Cloud experiment shows: cloud users can achieve better performance by enabling On-Ramp in their own VM cluster even though there may be non-On-Ramp traffic on their paths.

• Re-visit this question in CloudLab.

• Experiment setup:
  • 100 servers randomly divided into 2 groups.
  • Inside each group, run: WebSearch @ 60% load + incast @ 2% load.
  • Don’t run cross-group traffic.
  • It models 2 users renting servers in a cloud environment but don’t know each other.
Deep-dive 3: Co-existence

1. Group 1 uses On-Ramp, Group 2 does not:
   • Both groups do better than when neither uses On-Ramp.
     • On-Ramp enables Group 1 to transmit their traffic at moments when Group 2 traffic is at low instantaneous load.
     • Group 2’s performance is also improved because of a reduction in overall congestion.

2. Group 1 and Group 2 both use On-Ramp
   • Both improve, Group 1’s performance is only slightly increased compared to the above case.
     • Group 1 obtains almost the same benefit from using On-Ramp whether or not Group 2 uses it.
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 any existing congestion-control algorithm

• On-Ramp contains two ideas:
  • Using synced clocks to improve network performance
  • Decoupling the handling of transience & equilibrium