AC⚡DC TCP: Virtual Congestion Control Enforcement for Datacenter Networks

Keqiang He, Eric Rozner, Kanak Agarwal, Yu Gu,

Wes Felter, John Carter, Aditya Akella

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

IBM Research

IBM

Datacenter Network Congestion Control

- Congestion is not rare in datacenter networks [Singh, SIGCOMM'15]

- Tail latenc
  - 99.9th-tile tOS'15]
  - Queueing
- New datac en proposed
  - E.g., DCTC

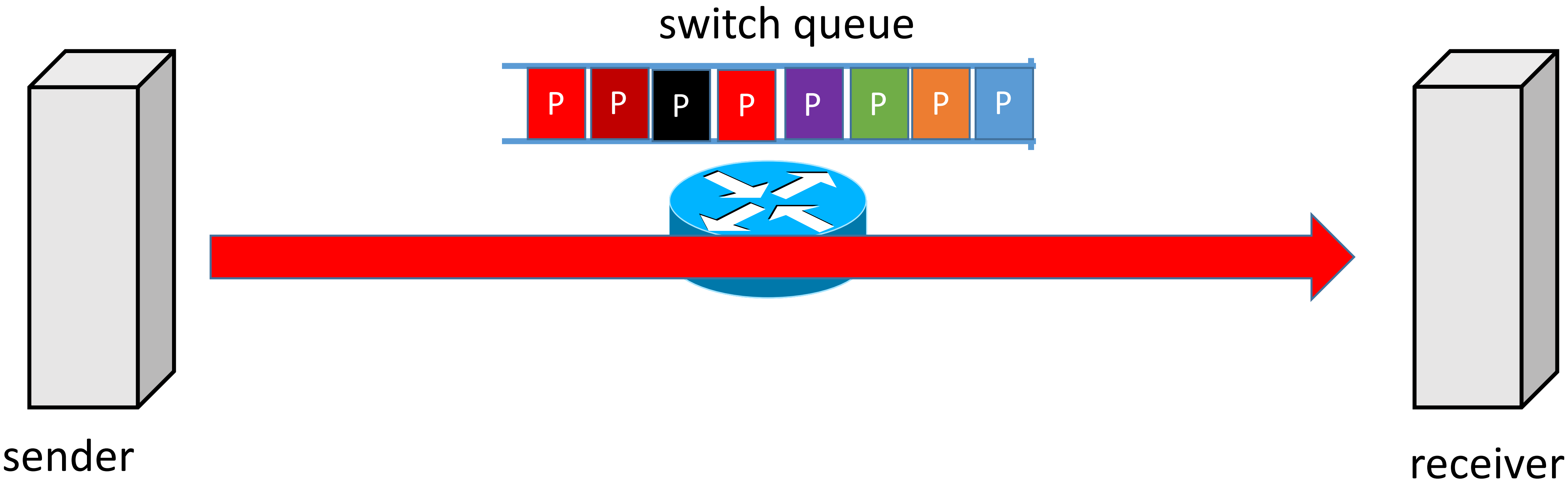
But, We Can Not Control VM TCP Stacks

- In multi-tenant datacenters, admins can not control VM TCP stacks
  - Because VMs are setup and managed by different entities

Therefore, outdated, inefficient, or misconfigured TCP stacks can be implemented in the VMs.

This leads to 2 main problems.

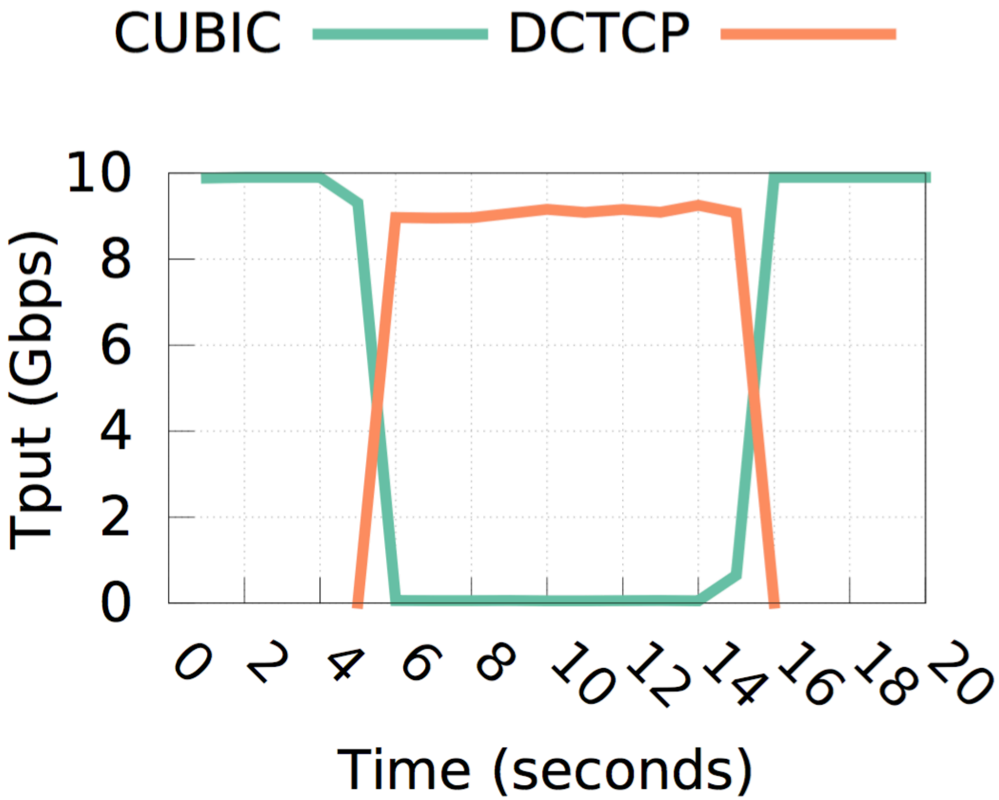
Problem #1: Large Queueing Latency



TCP RTT can reach tens of milliseconds because of packet queueing.

Problem #2: TCP Unfairness

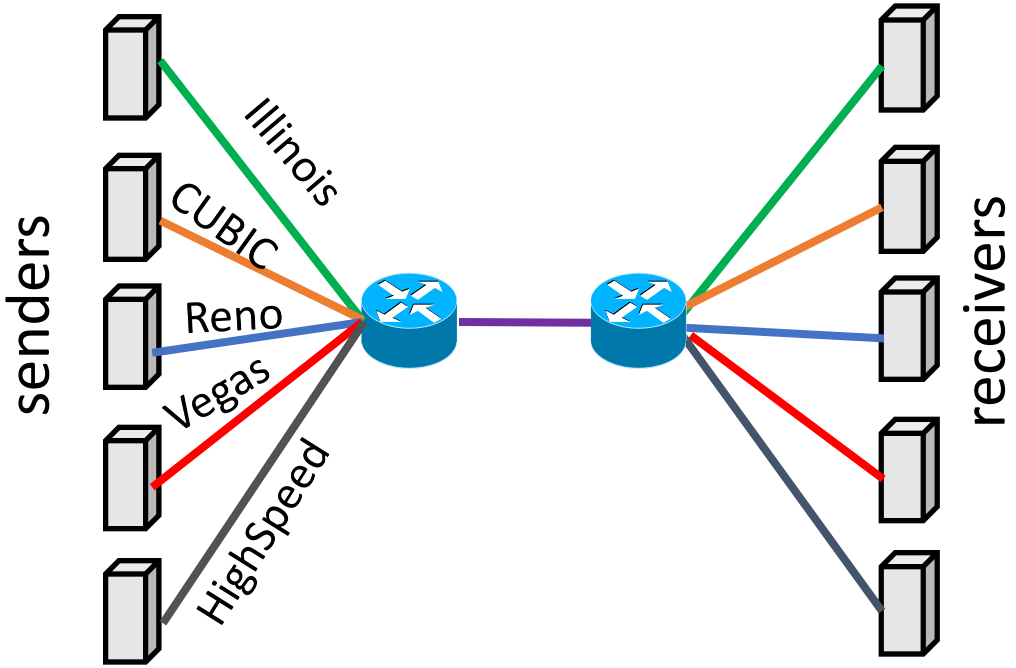
- ECN and non-ECN coexistence problem [Judd, NSDI'15]
  - Non-ECN: e.g., CUBIC
  - ECN: e.g., DCTCP



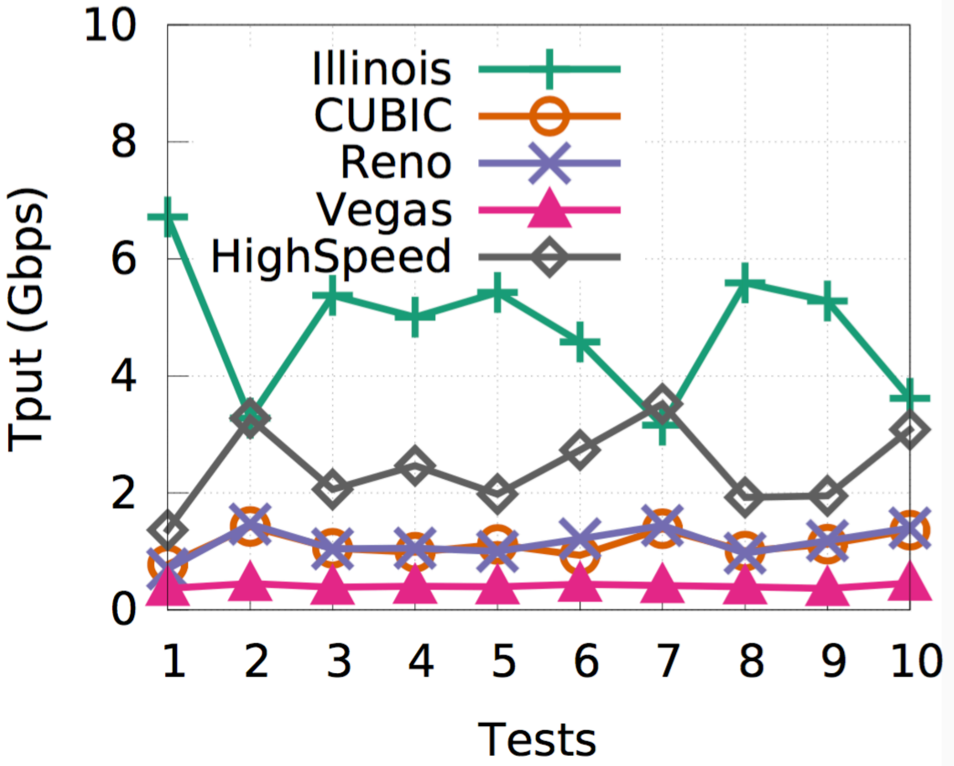
Problem #2: TCP Unfairness (cont.)

CC: Congestion Control

- Different congestion control algorithms lead to unfairness



Dumbbell topology



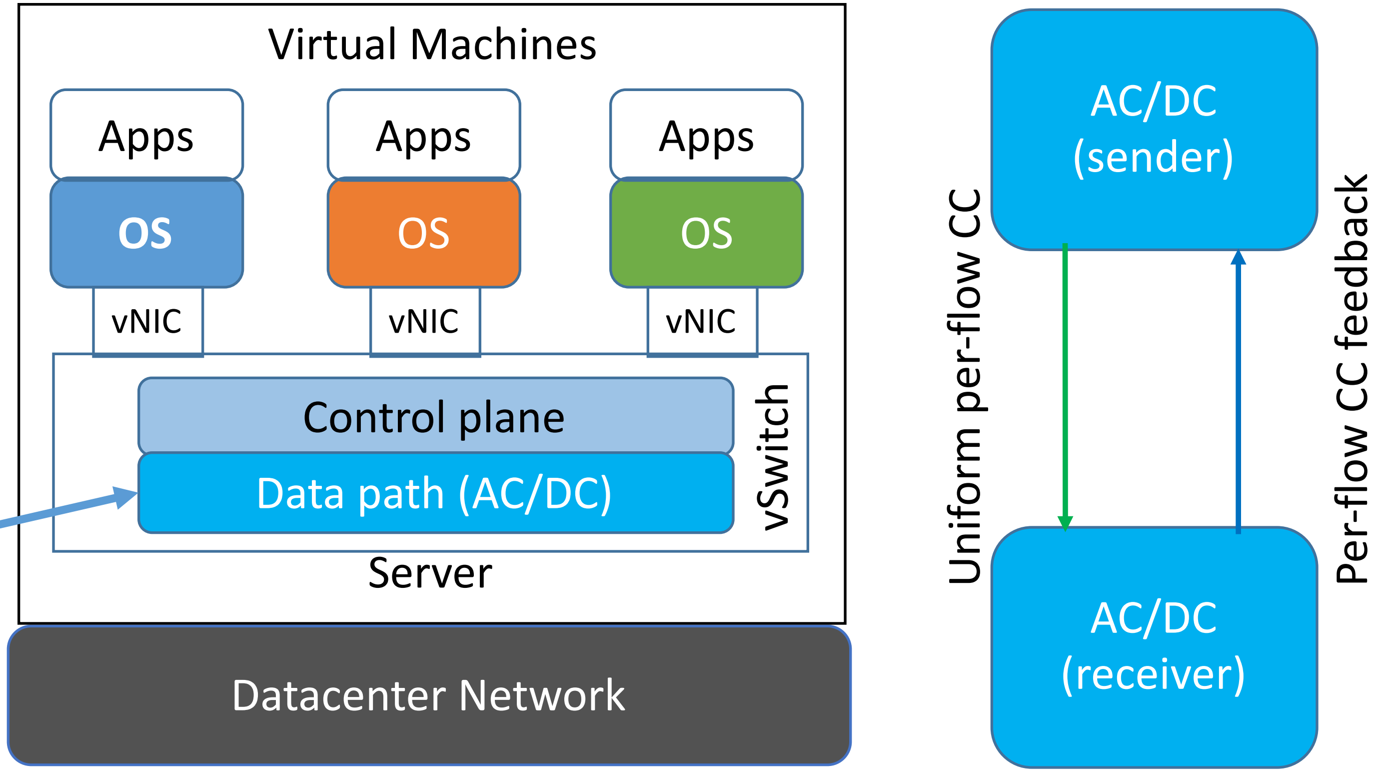
5 flows with different CC algorithms congest a 10G link

AC/DC TCP: Administrator Control over Data Center TCP

Implements TCP congestion control in the Virtual Switch

Ensures VM TCP stacks can not impact the network

AC⚡DC: High Level View



Case study: DCTCP CC in the vSwitch

AC⚡DC Benefits

- No modifications to VMs or hardware
- Low latency provided by state-of-the-art CC algorithms
- Improved TCP fairness and support both ECN and non-ECN flows
- Enforce per-flow differentiation via congestion control, e.g.,
  - East-west and north-south flows can use different CCs (web server)
  - Give higher priority to “mission-critical” traffic (backend VM)

AC⚡DC Design

- Obtaining Congestion Control State
- DCTCP Congestion Control in the vSwitch
- Enforcing Congestion Control
- Per-flow Differentiation via Congestion Control

Obtaining Congestion Control State

- Per-flow connection tracking
  - All traffic goes through the virtual switch
  - We can reconstruct CC via monitoring all the packets of a connection

- Maintain per-flow congestion control variables
  - E.g., CC-related sequence numbers, dupack counter etc

DCTCP Congestion Control in the vSwitch

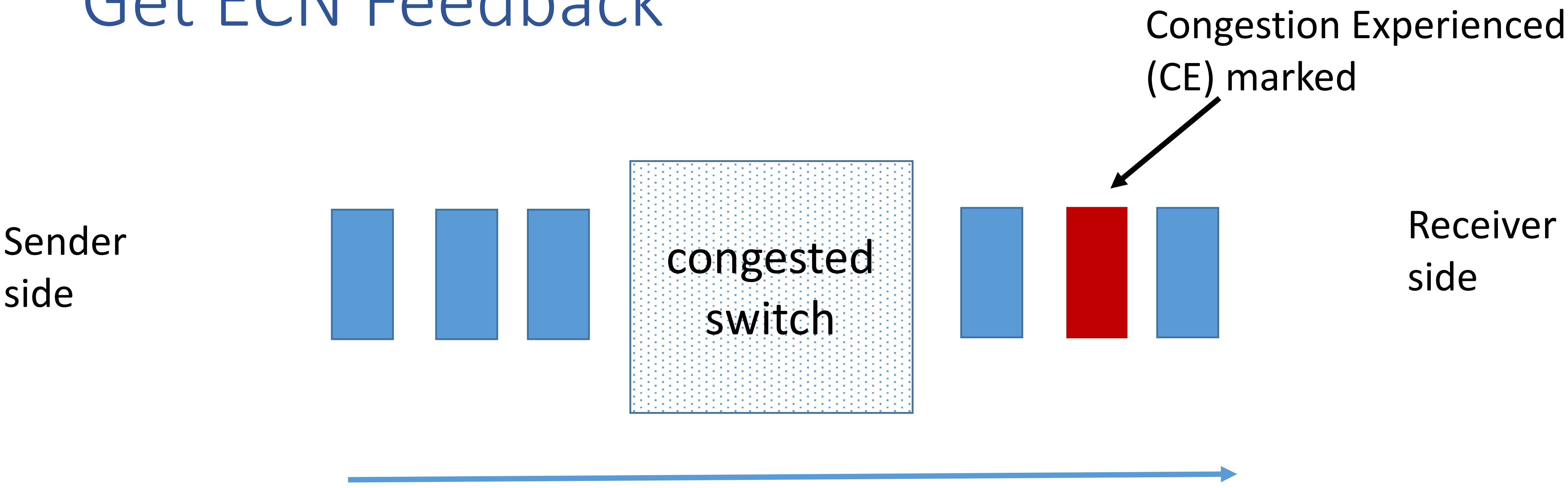
- Universal ECN marking

- Get ECN feedback

Universal ECN Marking

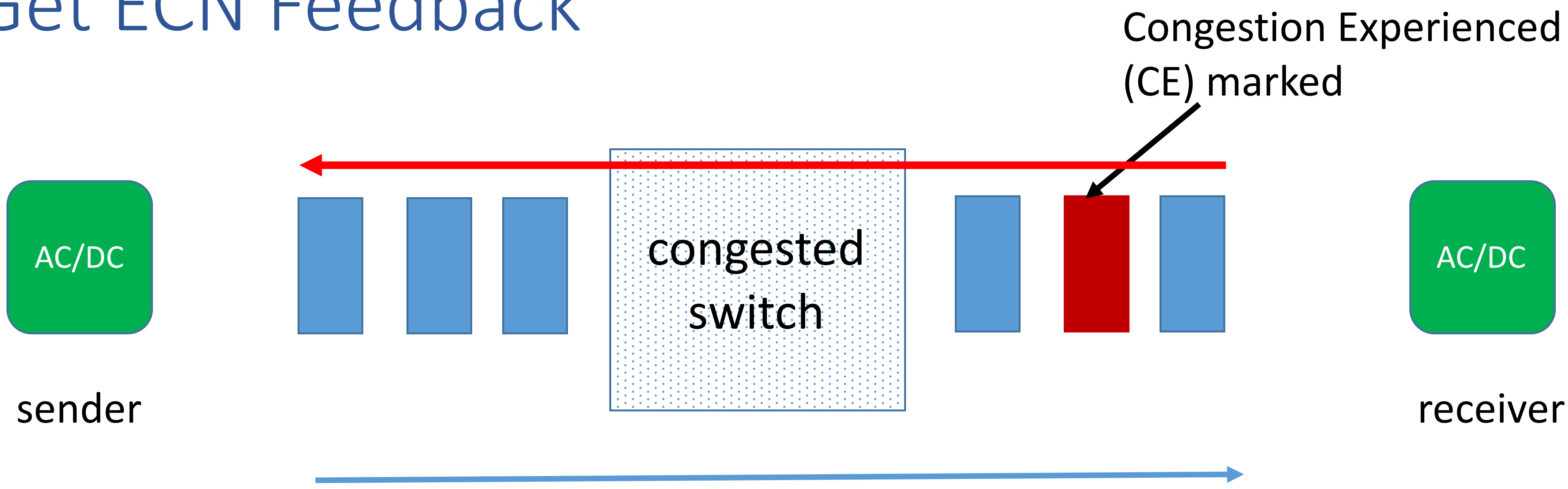
- Why?
  - Not all VMs run ECN-Capable Transports (ECT) like DCTCP
- Universal ECN Marking
  - All packets entering the fabric should be ECN-marked by the virtual switch
  - Solves the ECN and non-ECN coexistence problem

Get ECN Feedback



Need a way to carry the congestion information back.

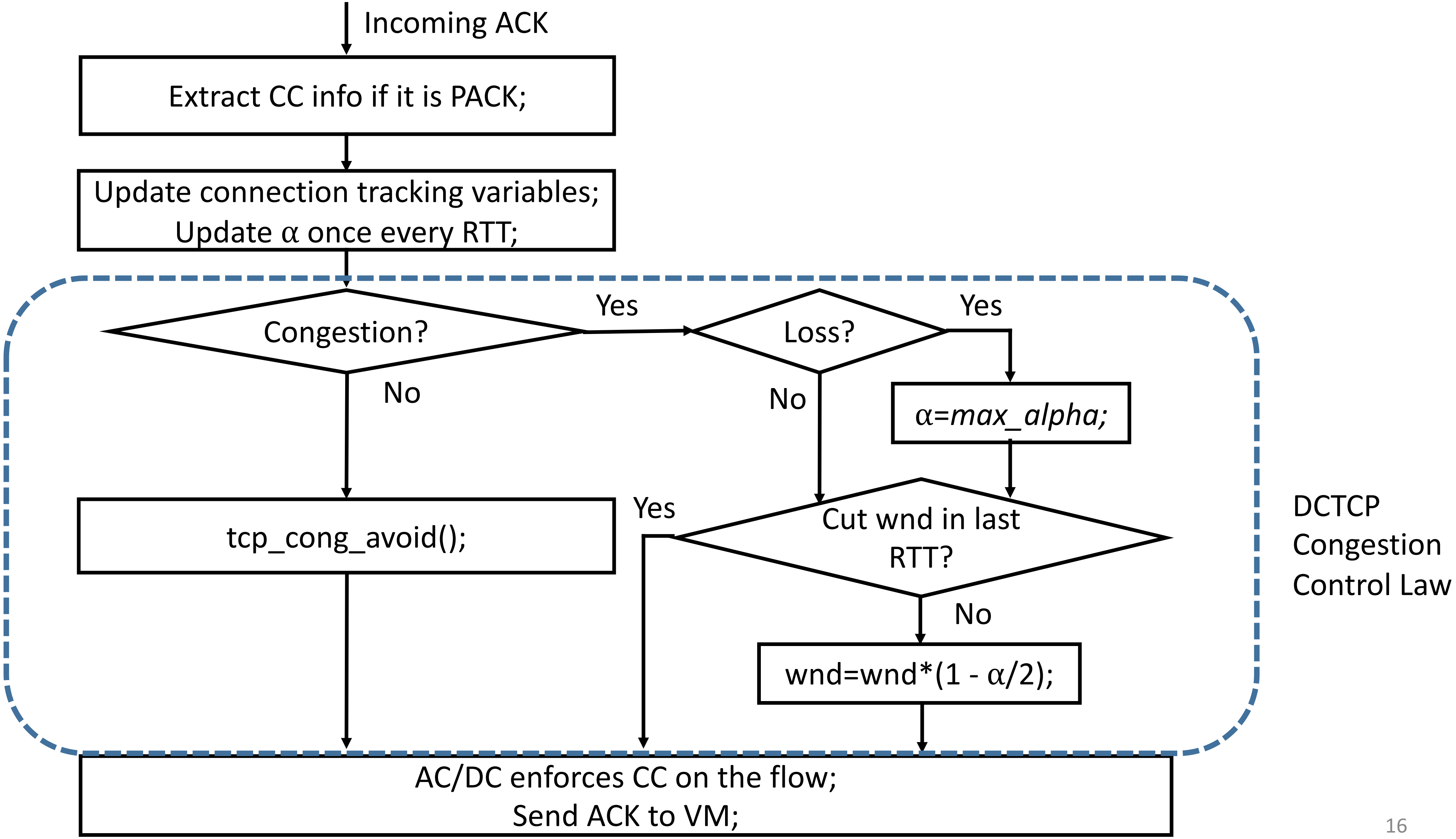
Get ECN Feedback



Congestion feedback is encoded as 8 bytes: {ECN_bytes, Total_bytes}.

Piggybacked on an existing TCP ACK (PACK).

DCTCP Congestion Control in the vSwitch



Enforcing Congestion Control

- TCP sends min(CWND, RWND)
  - CWND is congestion control window (**congestion control**)
  - RWND is receiver's advertised window (**flow control**)

- AC⚡DC reuses RWND for congestion control purpose
  - VMs with unaltered TCP stacks will naturally follow our enforcement

- Non-conforming flows can be policed by dropping any excess packets not allowed by the calculated congestion window
  - Loss has to be recovered e2e, this incentivizes tenants to respect standards

Control Law for Per-flow Differentiation

DCTCP:

$$RWND = RWND * (1 - \frac{\alpha}{2})$$

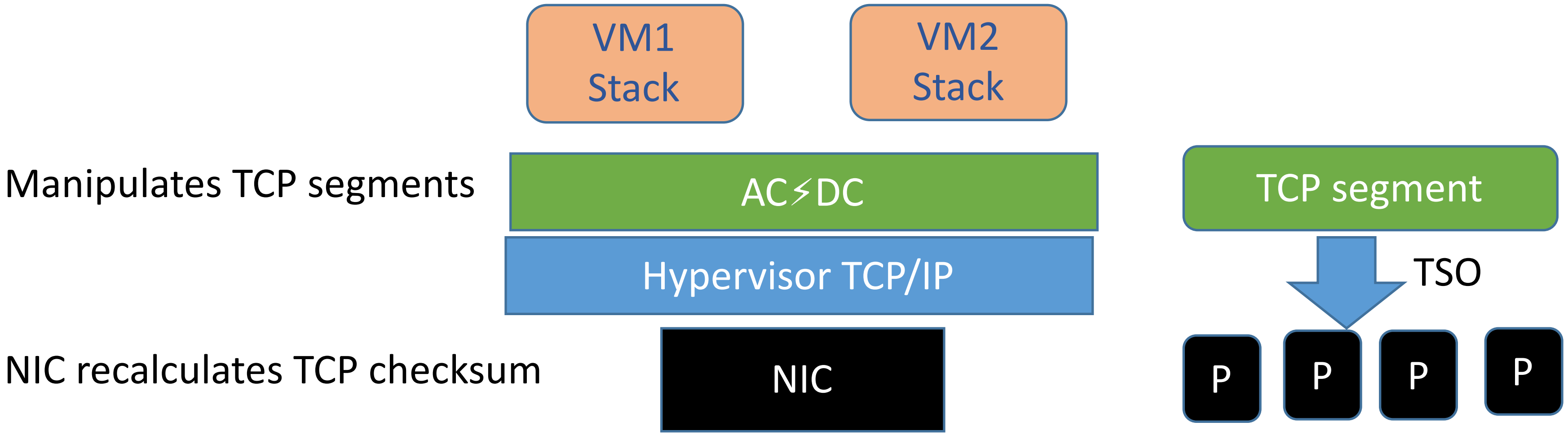
AC⚡DC TCP:

$$RWND = RWND * (1 - (\alpha - \frac{\alpha\beta}{2}))$$

When β is close to 1, it becomes DCTCP.
When β is close to 0, it backs-off aggressively.
Larger β for higher priority traffic.

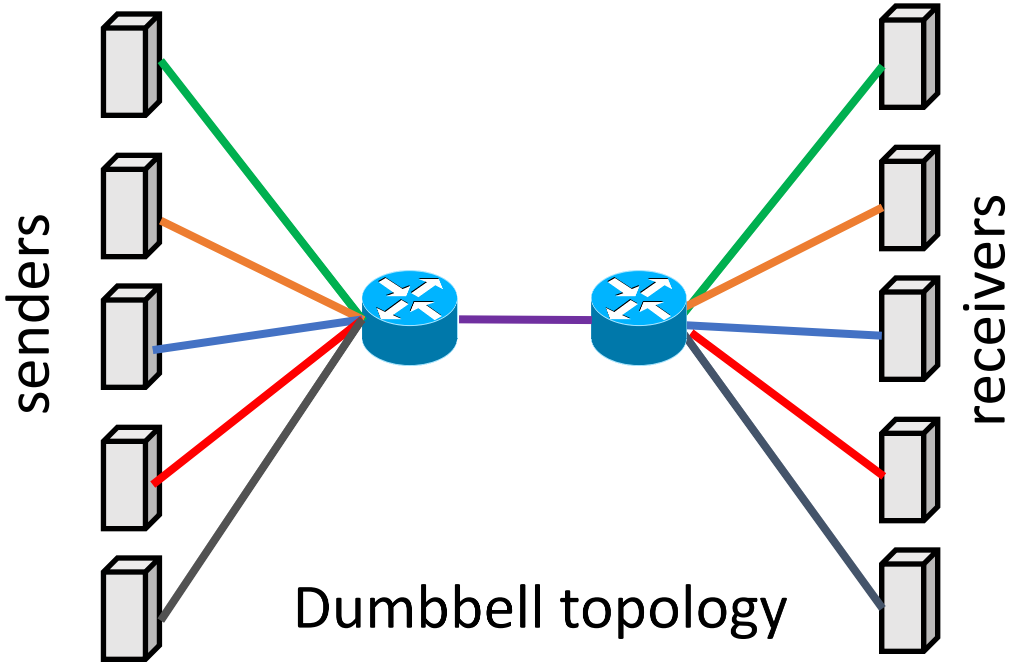
Implementation

- Prototype implementation in Open vSwitch kernel datapath
  - ~1200 LoC added
- Our design leverages available techniques to improve performance
  - RCU-enabled hash tables to perform connection tracking
  - AC⚡DC manipulates TCP segments, instead of MTU-sized packets
  - AC⚡DC leverages NIC checksumming so the TCP checksum does not have to be recomputed after header fields are modified

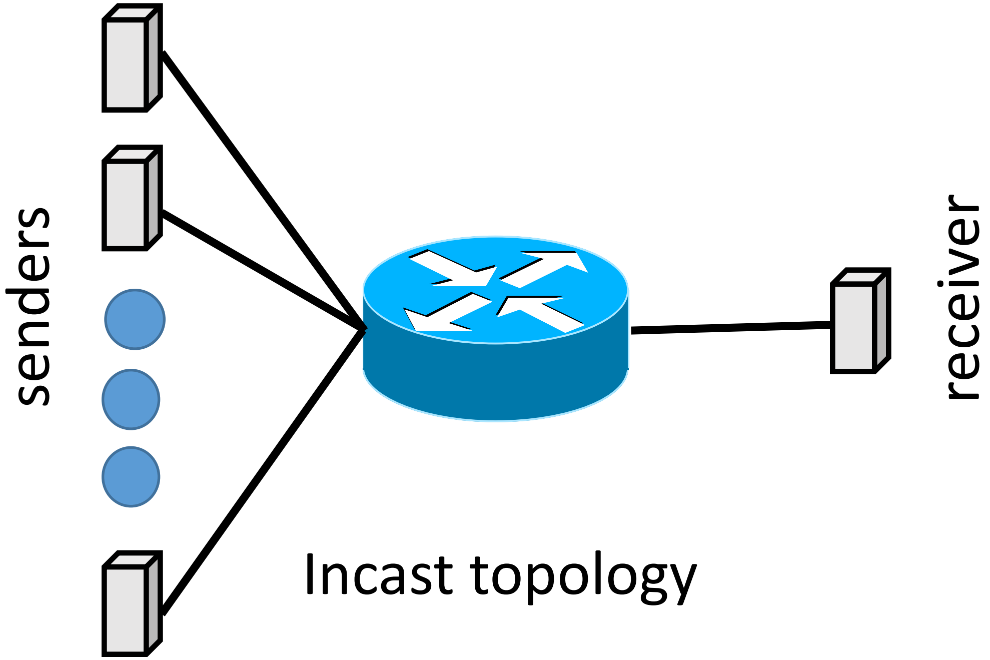


Evaluation

- Testbed: 17 servers (6-core, 60GB memory), 6 10Gbps switches
- Microbenchmark topologies



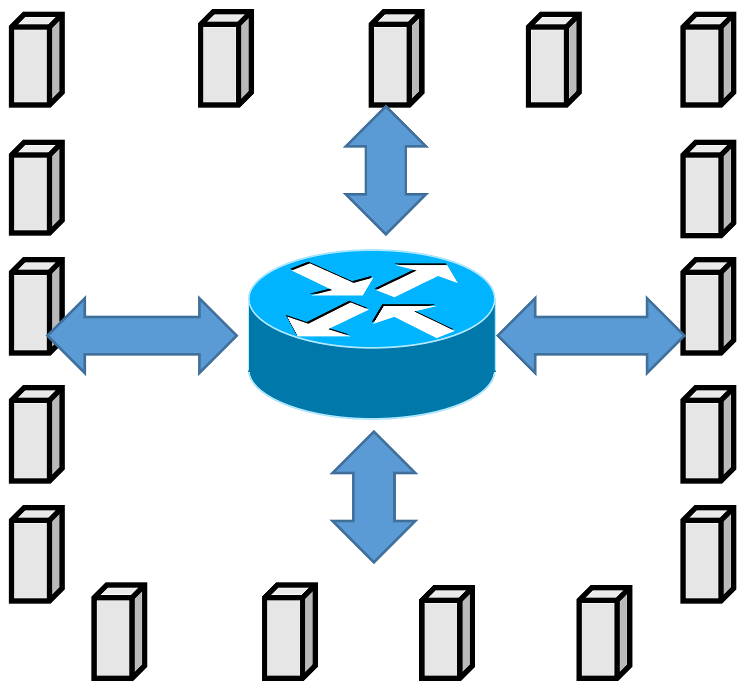
Dumbbell topology



Incast topology

Evaluation

- Macrobechmark topology

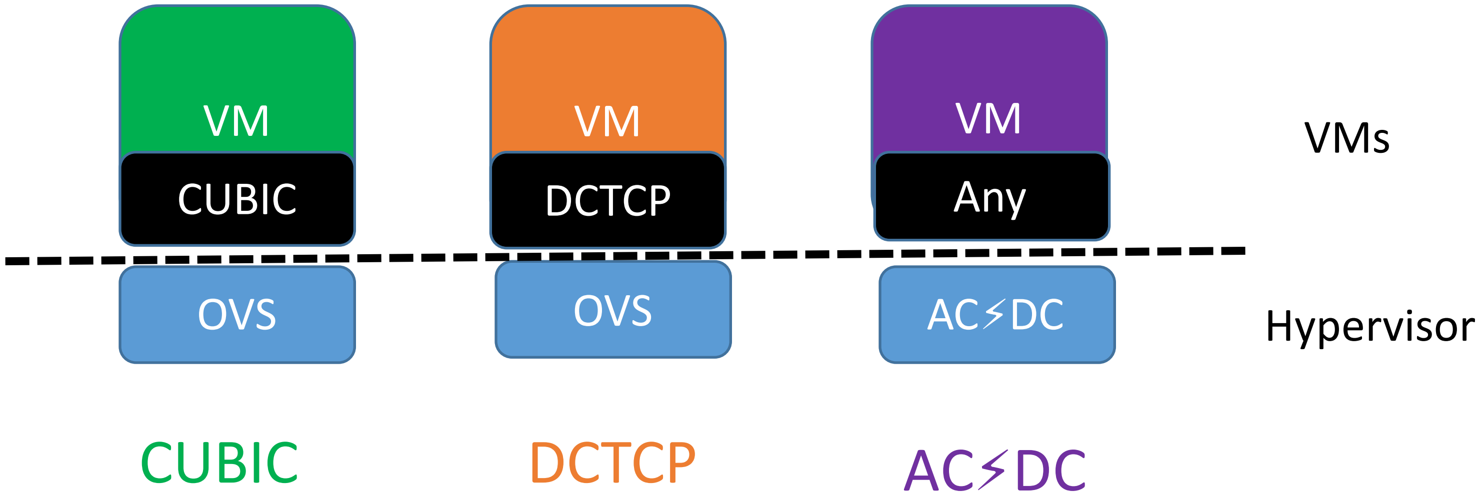


17 servers attached
to a 10G switch.

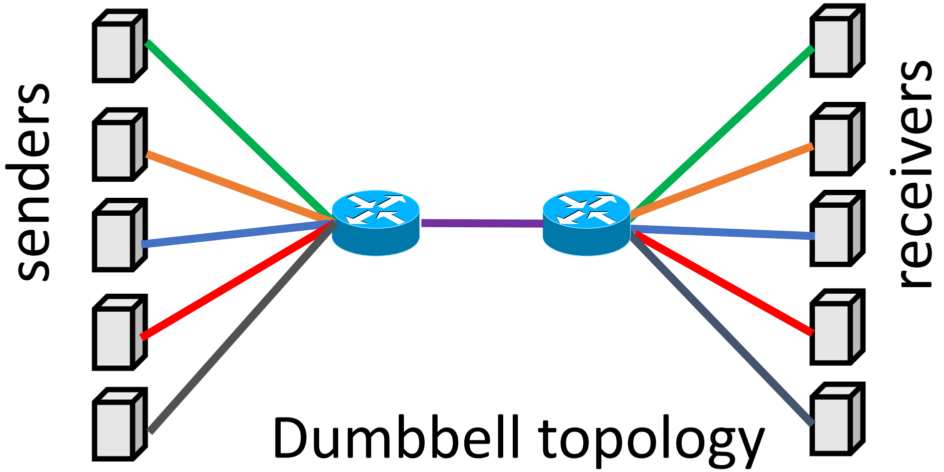
- Metrics: TCP RTT, loss rate, Flow Completion Time (FCT)

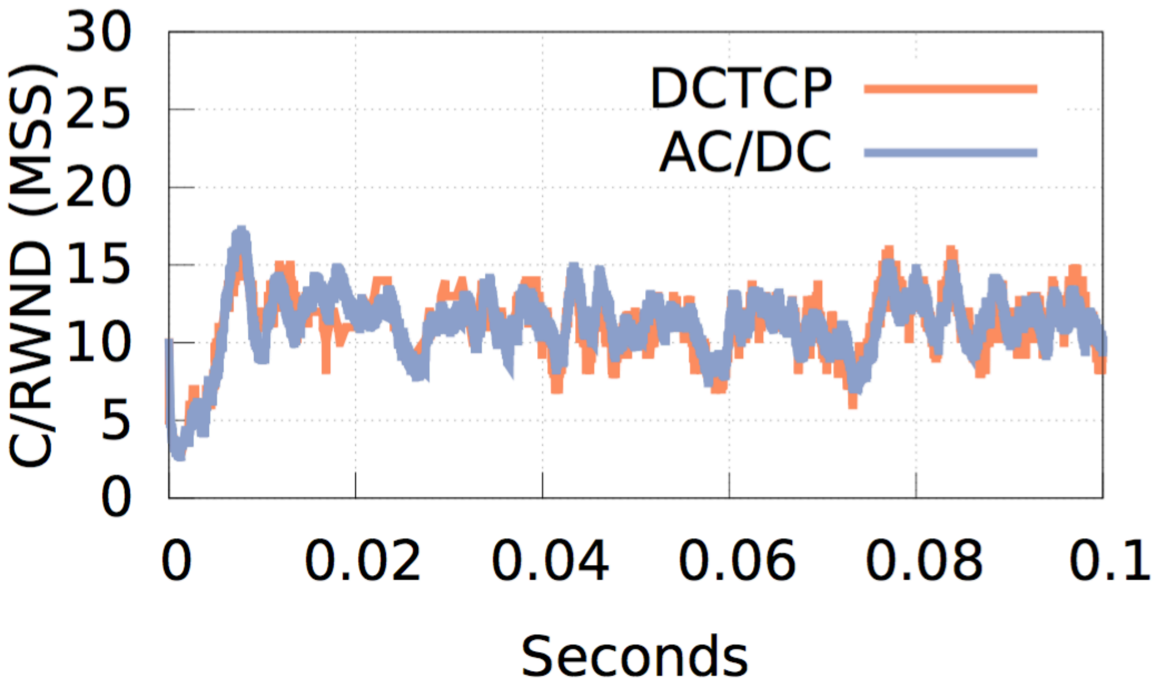
Experiment Setting (compared 3 schemes)

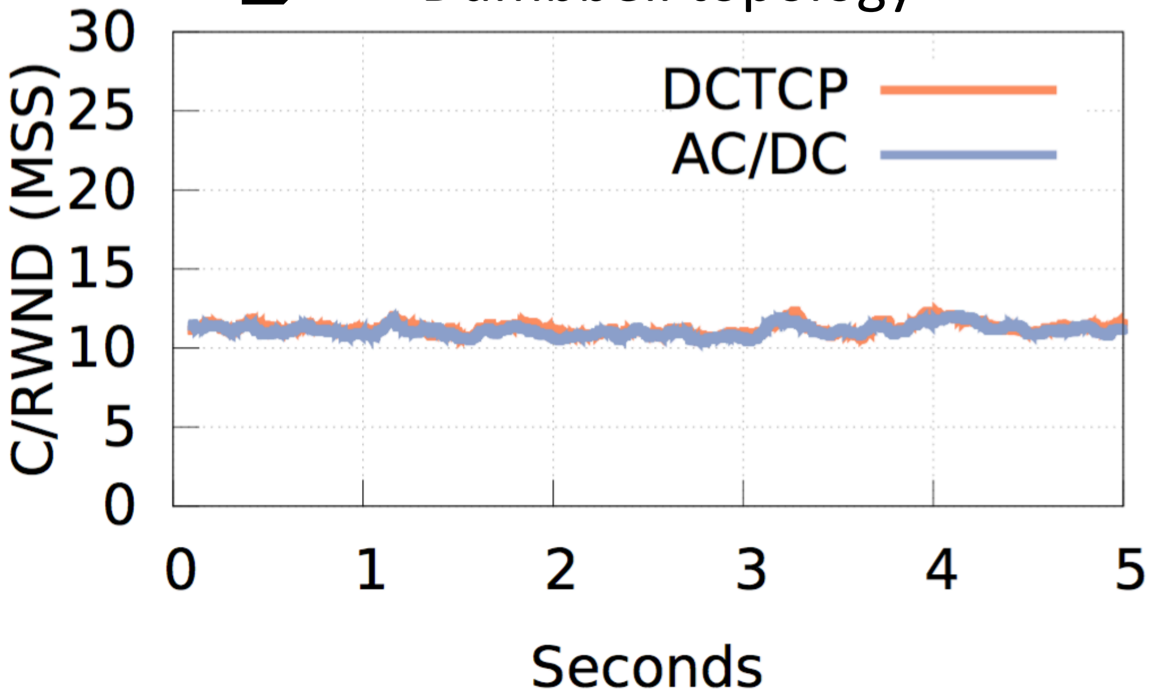
- CUBIC
  - CUBIC stack on top of standard OVS
- DCTCP
  - DCTCP stack on top of standard OVS
- AC⚡DC
  - CUBIC/Reno/Vegas/HighSpeed/Illinois stacks on top of AC⚡DC



Tracking Window Size

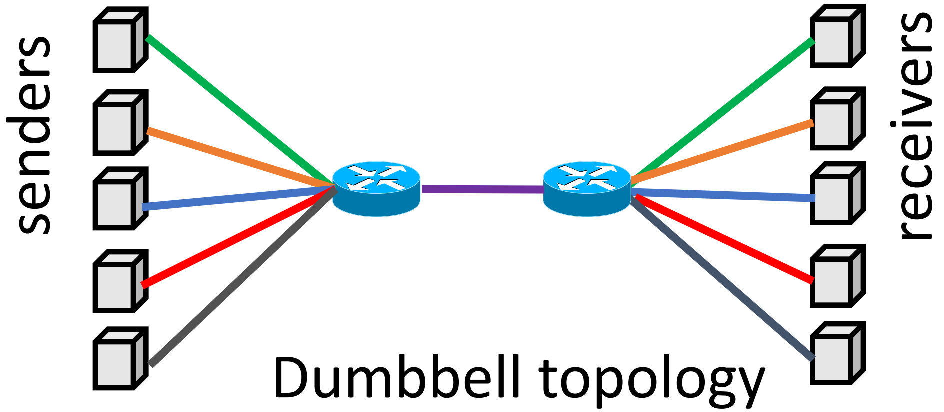


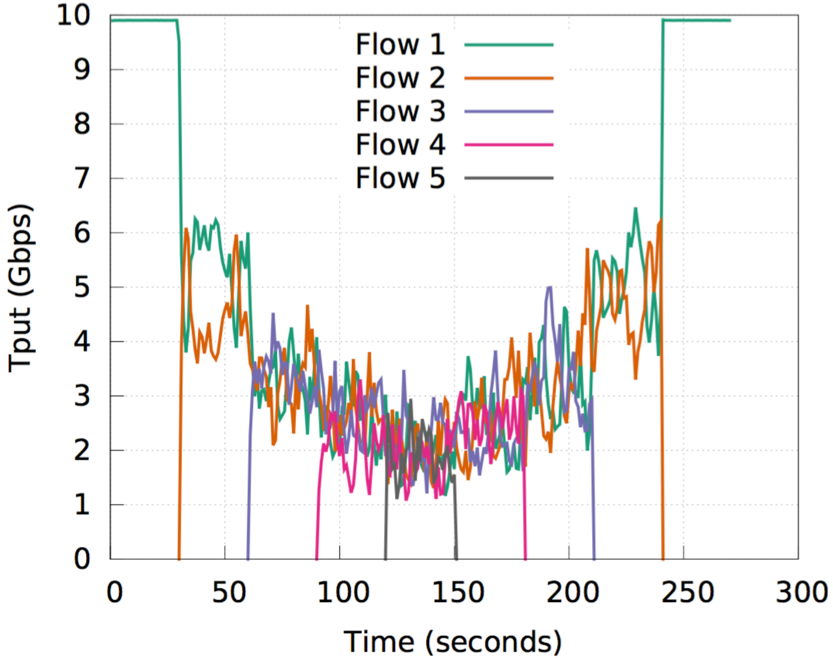




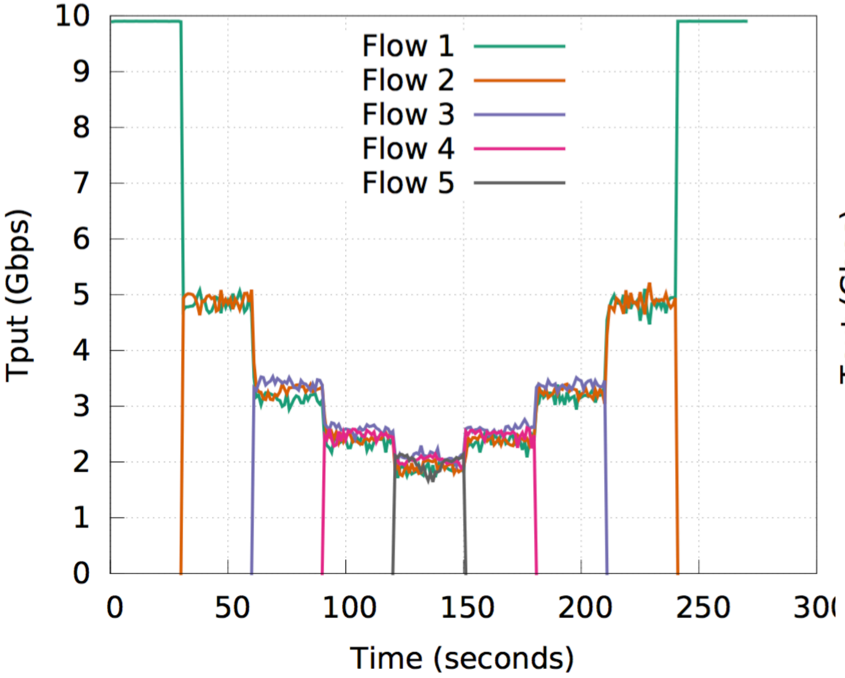
Running DCTCP stack on top of AC⚡DC, only outputs calculated RWND without enforcement.
AC⚡DC closely tracks the window size of DCTCP.

Convergence

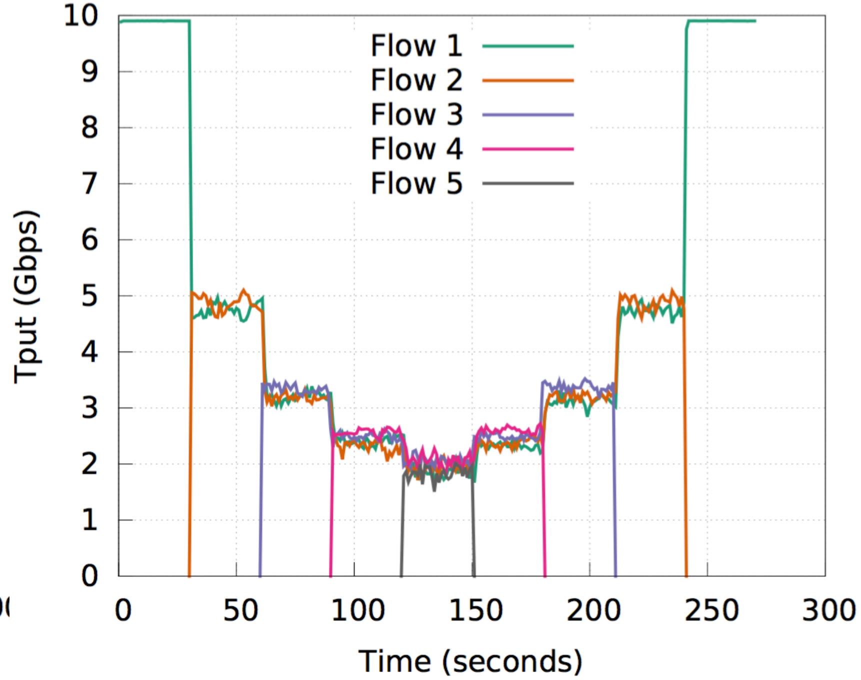




CUBIC



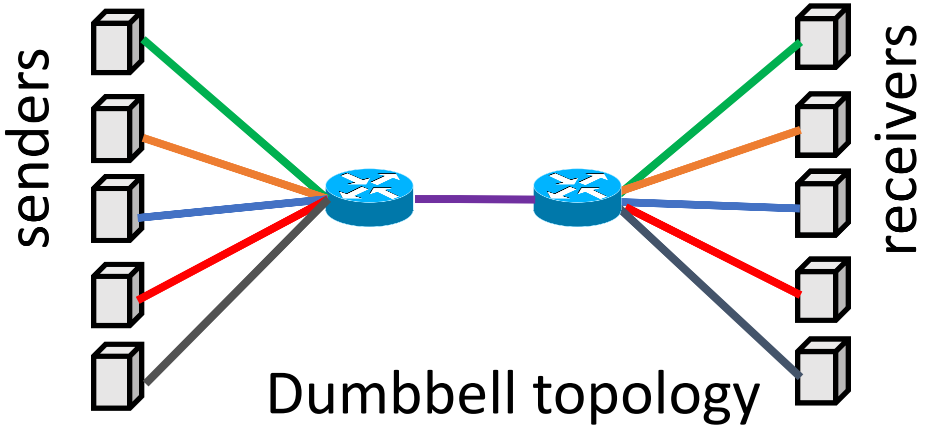
DCTCP

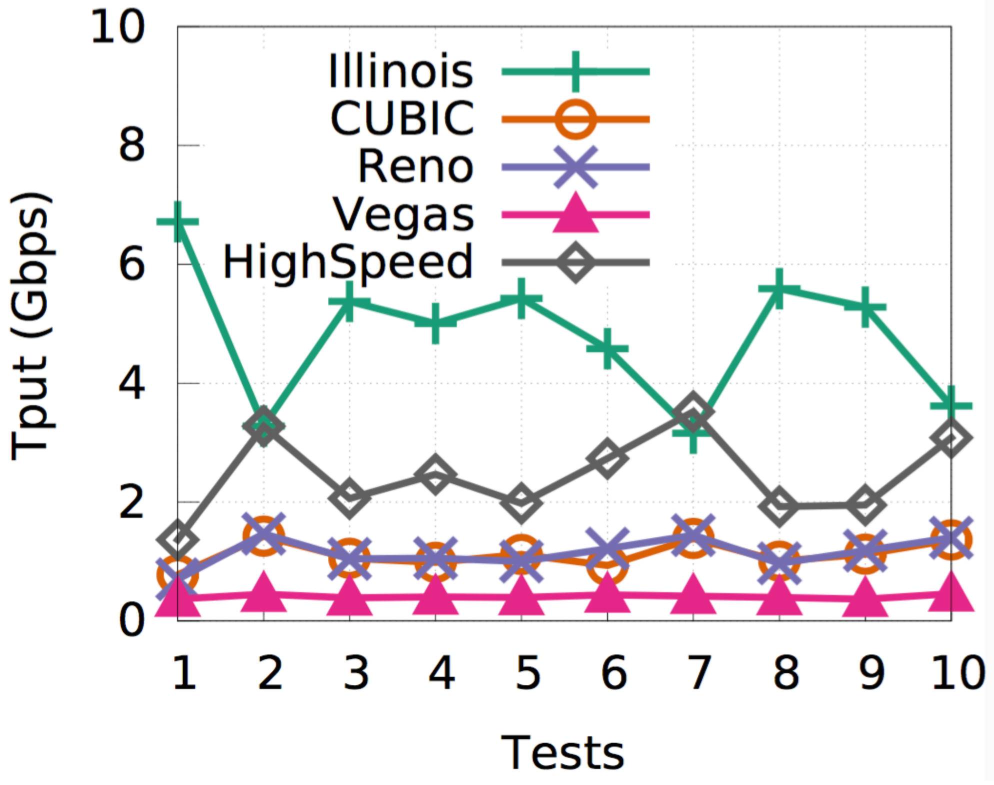


AC/DC

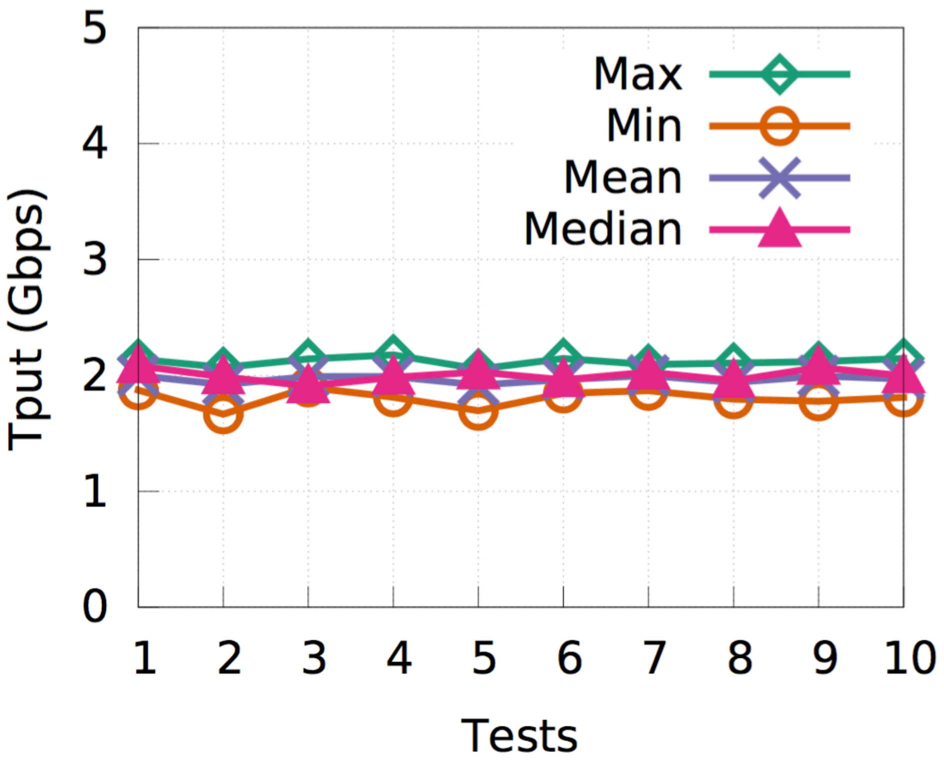
AC/DC has comparable convergence properties as DCTCP and is better than CUBIC.

AC⚡DC improves fairness when VMs use different CCs



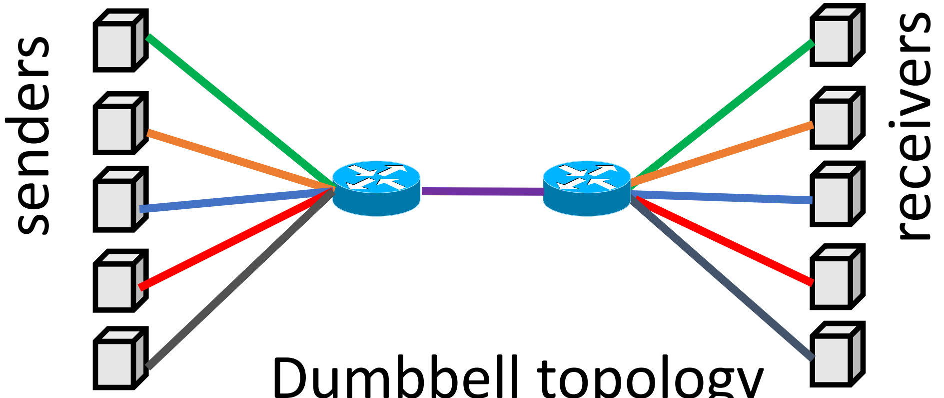


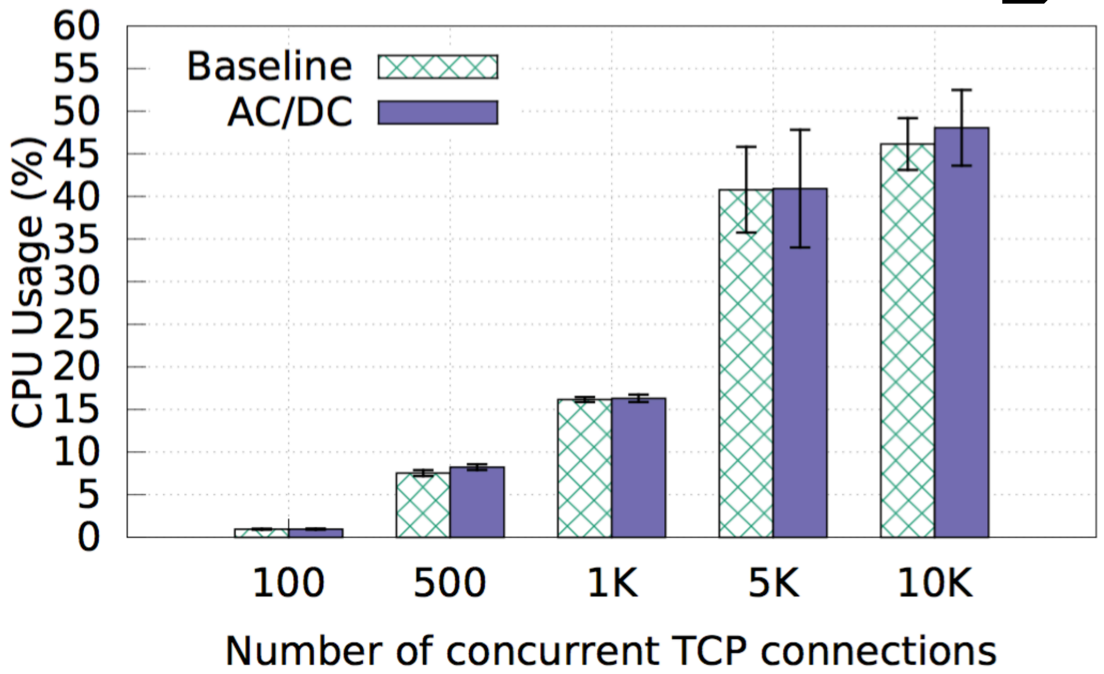
Standard OVS



AC⚡DC

Overhead (CPU and Memory)

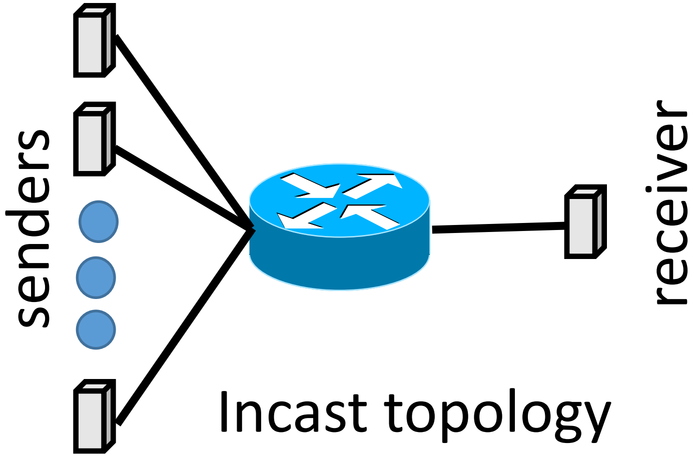


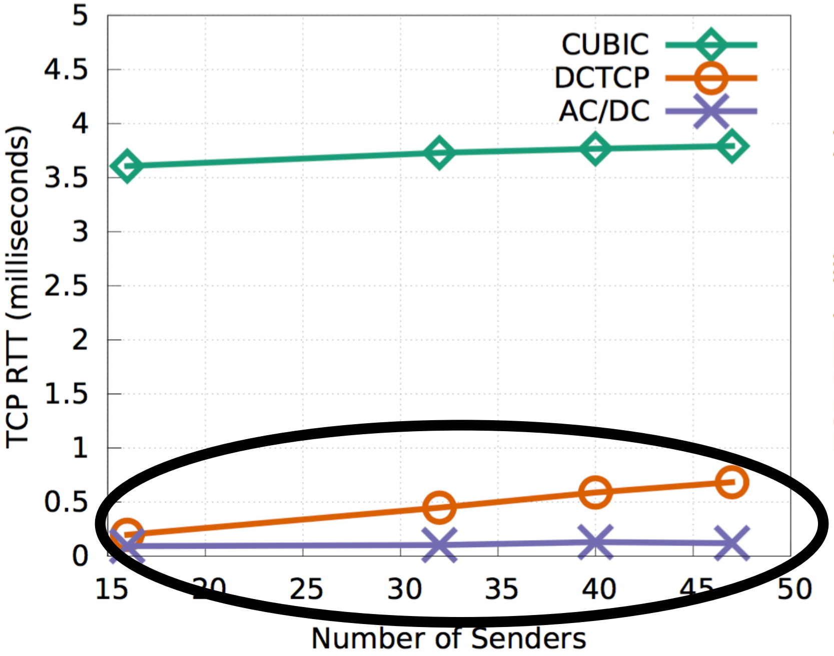


Sender side

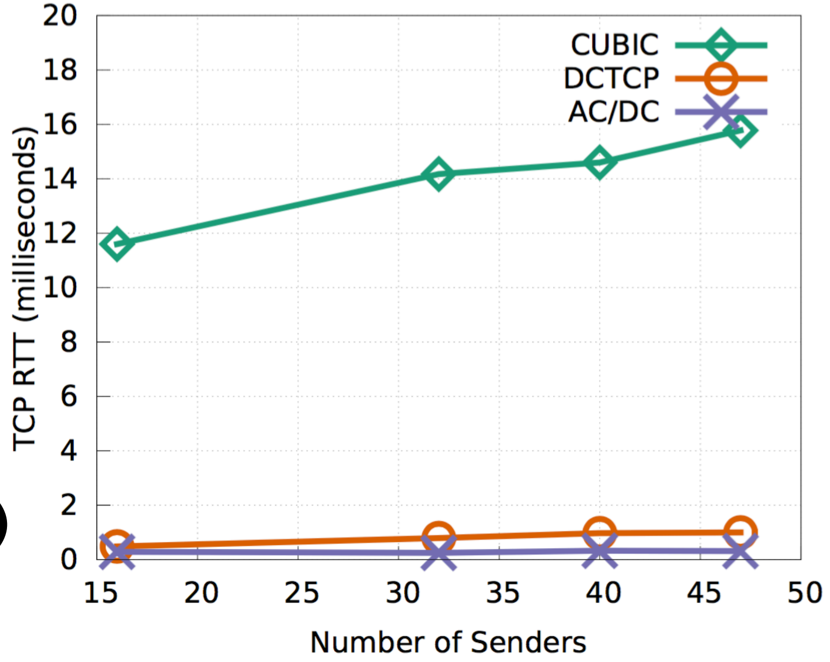
Less than 1% additional CPU overhead compared with the baseline.
Each connection uses 320 bytes to maintain CC variables (10k connections use 3.2MB).

TCP Incast RTT & drop rate

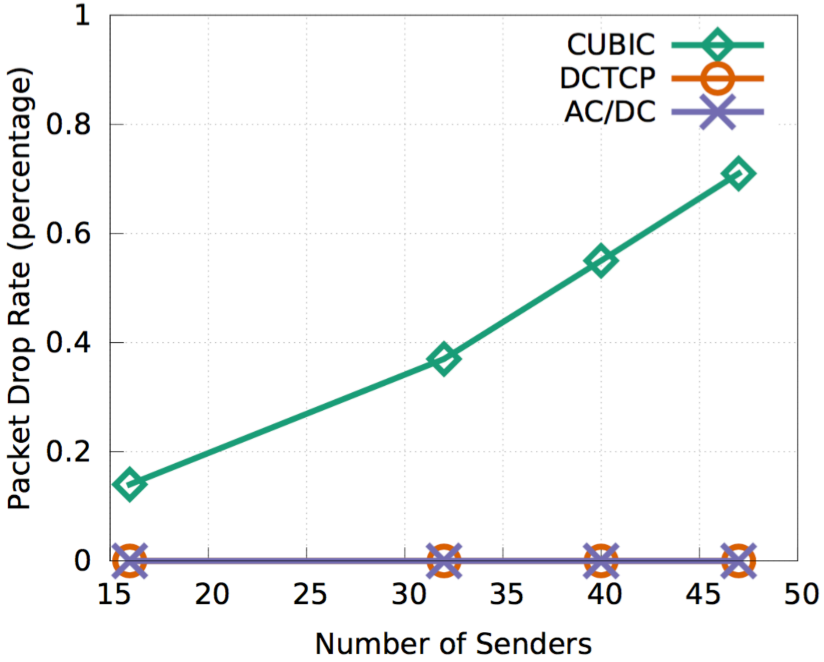




50th percentile RTT



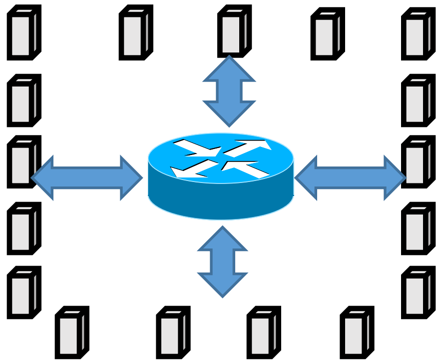
99.9th percentile RTT



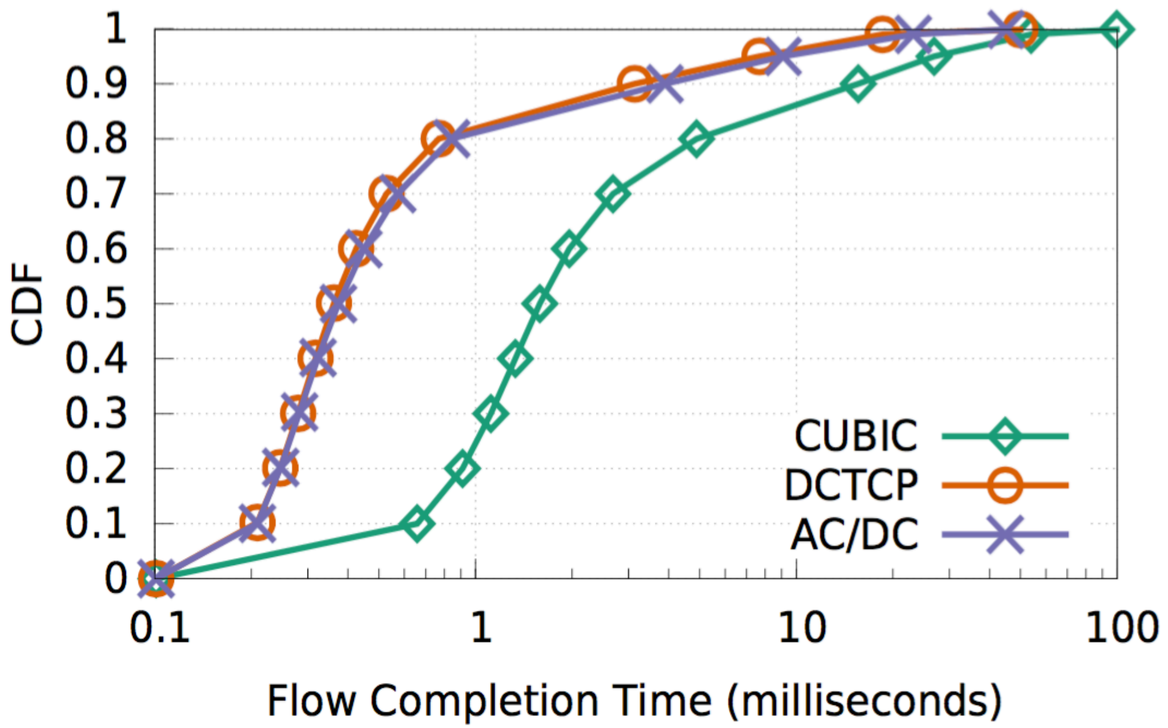
Packet drop rate

AC⚡DC tracks the performance of DCTCP closely.

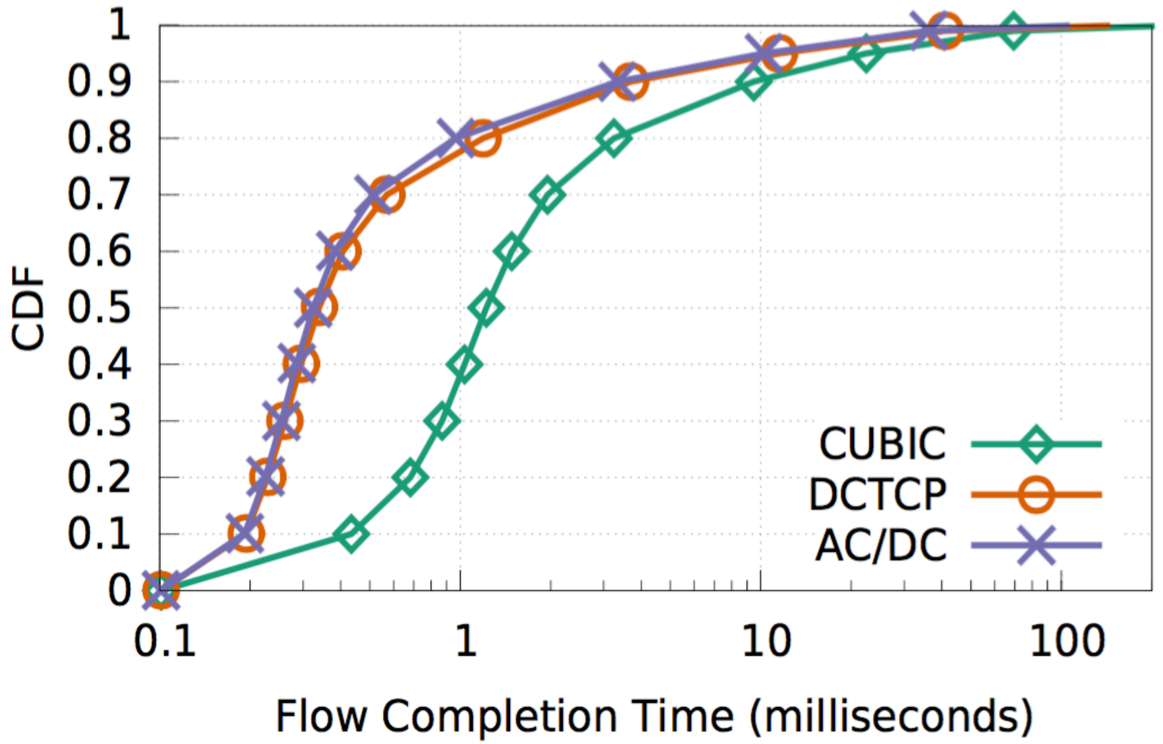
Flow completion time with trace-driven workloads



17 servers attached to a 10G switch.



Web-searching workload (DCTCP)



Data-mining workload (CONGA)

AC⚡DC obtains same performance as DCTCP.
AC⚡DC can reduce FCT by 36% - 76% compared with default CUBIC.

Summary

- AC⚡DC allows administrators to regain control over arbitrary tenant TCP stacks by enforcing congestion control in the virtual switch
- AC⚡DC requires no changes to VMs or network hardware
- AC⚡DC is scalable, light-weight (< 1% CPU overhead) and flexible

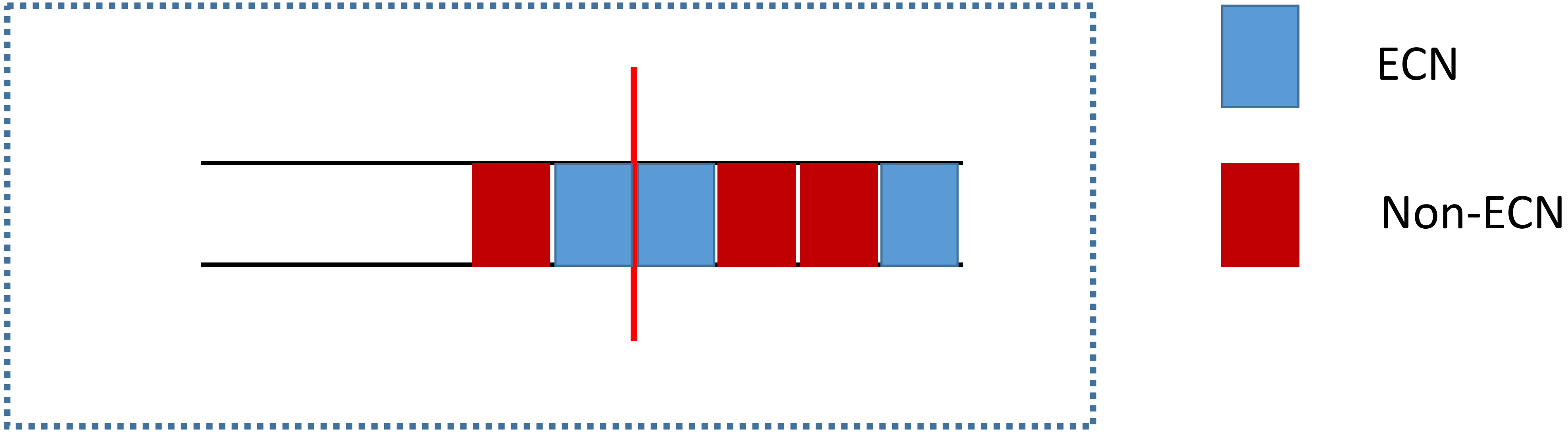
Thanks!

Backup Slides

Related Work

- DCTCP
  - ECN-based congestion control for DCNs
- TIMELY
  - Latency-based congestion control for DCNs
  - Accurate latency measurement provided by accurate NIC timestamps
- vCC
  - vCC and AC⚡DC are closely related works by two independent teams ☺

ECN and non-ECN Coexistence



Switch configured with WRED/ECN

When queue occupancy is larger than marking threshold, non-ECN packets are dropped

IPSec

- AC⚡DC is not able to inspect the TCP headers for IPSec traffic
- May perform approximating rate limiting based on congestion feedback information.