Data Center TCP(DCTCP)

Mohammad Alizadeh [*+], Albert Greenberg [*], David A. Maltz [*], Jitendra Padhye [*], Parveen Patel [*], Balaji Prabhakar[+], Sudipta Sengupta [*], Murari Sridharan [*]

[*] Microsoft Research

[+]Stanford University

Contributions

1. **Identify impairments hurting performance in data center environment**
   - -- Incast
   - -- Queue buildup
   - -- Buffer Pressure
2. **Propose Data Center TCP(DCTCP)**
   - -- Explicit Congestion Notification(ECN)
   - -- A control scheme at the **sources** in order to lower switch buffer queue length while achieving high throughput

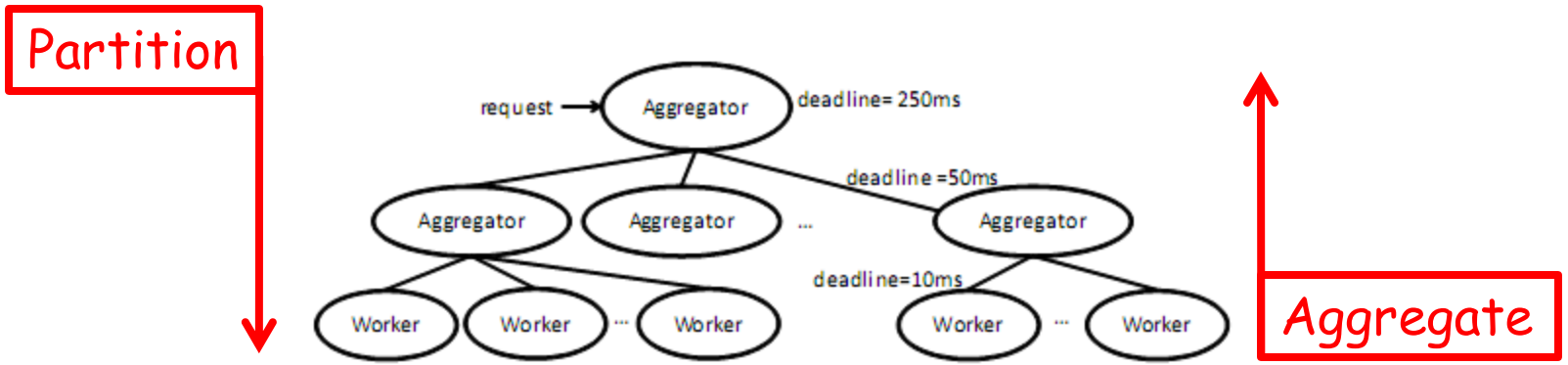
Traffic Characterization

1. Partition/Aggregate design pattern

2. Throughput-sensitive large flows, latency-critical short flows co-exist

Traffic Characterization

1. Partition/Aggregate design pattern



2. Throughput-sensitive large flows, latency-critical short flows co-exist

Traffic Characterization

1. Partition/Aggregate design pattern

2. Throughput-sensitive large flows, latency-critical short flows co-exist

-- query traffic: integrated with urgent short message traffic

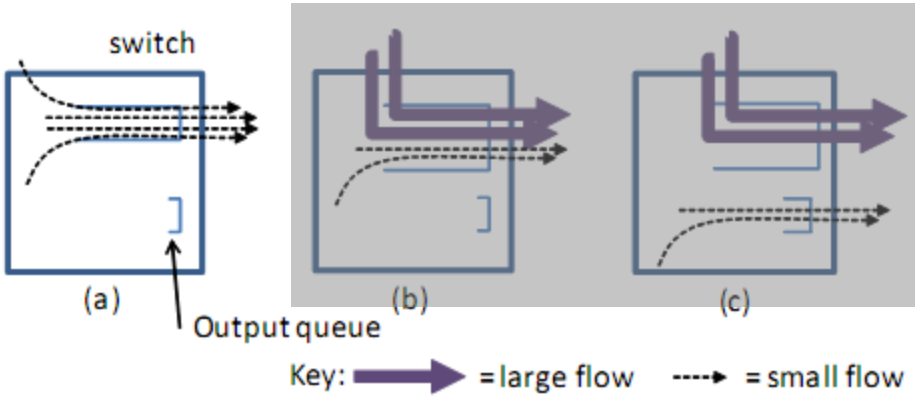
-- background traffic: organize the massive data for query **response**

Performance Impairments

(a). Incast
(b). Queue buildup
(c). Buffer Pressure

Performance Impairments

(a). Incast



-- Exhaust the maximum permitted buffer for the interface → Packet losses

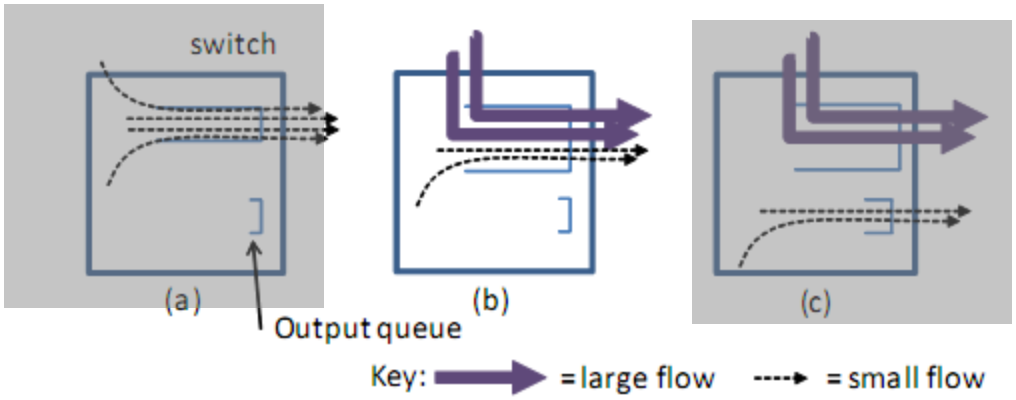
(b). Queue buildup

(c). Buffer Pressure

Performance Impairments

(a). Incast

(b). Queue buildup



Incast or great latency!

--When long and short flows traverse the same queue

If a short flow is behind a large flow?

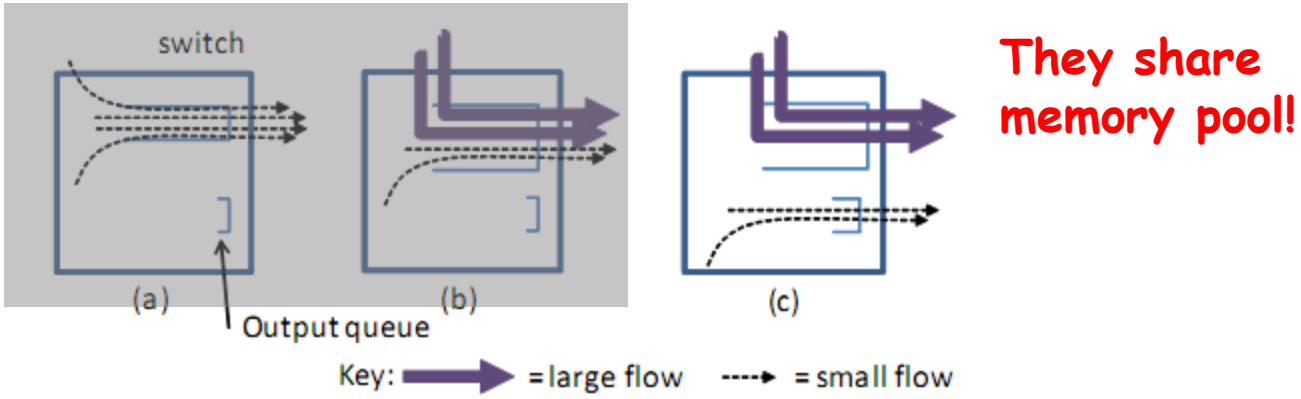
(c). Buffer Pressure

Performance Impairments

(a). Incast

(b). Queue buildup

(c). Buffer Pressure



They share memory pool!

-- The loss rate of short flow depends on the large flows traversing other ports, Why?

DCTCP Algorithm

1. **Simple Marking at the Switch**
2. **ECN-Echo at the Receiver**
3. **Controller at the Sender**

DCTCP Algorithm

1. Simple Marking at the Switch

-- Only one parameter, the marking threshold ***K***, based on **instantaneous** queue length

-- while arriving packet P_n :

if Queue length > K :

P_n's CE codepoint marked

Recommended:
K=20 for 1Gbps
K=65 for 10Gbps

2. ECN-Echo at the Receiver

3. Controller at the Sender

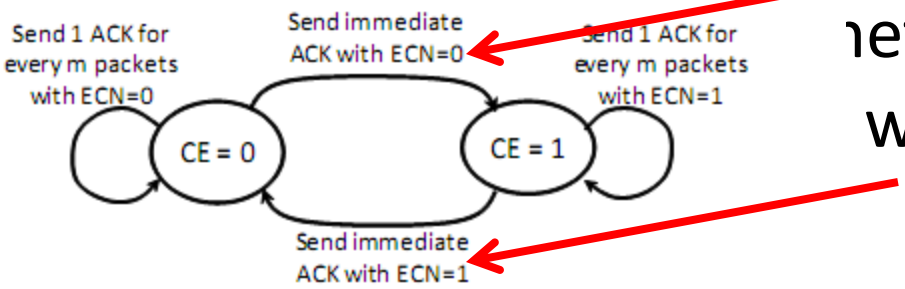
DCTCP Algorithm

1. Simple Marking at the Switch
2. **ECN-Echo at the Receiver**

   -- Convey the CE codepoint back to the sender.

   only difference from TCP receiver

   -- Using delayed ACKs to set ECN-Echo bit



ECN=1

nether the last with the CE

ECN=0

Figure 10: Two state ACK generation state machine.

3. Controller at the Sender

DCTCP Algorithm

1. Simple Marking at the Switch
2. ECN-Echo at the Receiver
3. **Controller at the Sender**

   -- The sender maintain an estimate of the fraction of packets that are marked, called α , updated for **every** window of data

$$\alpha \leftarrow (1\text{-}g) \times \alpha + g \times F$$

-- $0<g<1$ is the weight given to new samples

-- F is the fraction of packets that marked in the last window of data

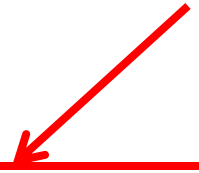
α estimates the probability that the queue size is greater than K

DCTCP Algorithm

1. Simple Marking at the Switch
2. ECN-Echo at the Receiver
3. **Controller at the Sender**

-- react when receiving an ACK with the ECN-Echo flag set

$$cwnd \leftarrow cwnd \times (1 - \alpha/2)$$



Only difference from TCP sender

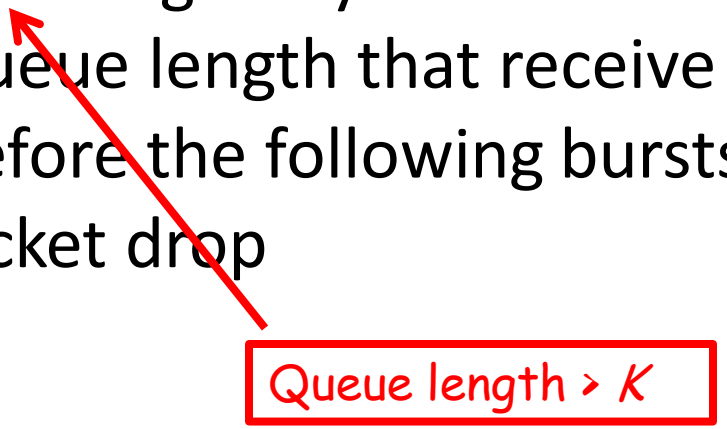
Theoretical Benefits

(a). Incast

-- Scenario: In practice, each packet drop occurs in subsequent RTTs

Numerous flows + Slow start!

-- DCTCP starts marking early + measure **instantaneous** queue length that receive enough marks before the following bursts which lead to packet drop



(b). Queue buildup

(c). Buffer Pressure

Theoretical Benefits

(a). Incast

(b). Queue buildup

-- senders of large flow starts reacting as soon as queue length > *K,* short flow doesn't need to wait too long

-- more buffer space is available to absorb transient micro-bursts

(c). Buffer Pressure

Theoretical Benefits

(a). Incast

(b). Queue buildup

(c). Buffer Pressure

-- A congested port's queue length does not grow exceedingly large, thus will not exhaust the buffer resources

Experiment Results

(1). Examine basic properties of the DCTCP

-- throughput, convergence, fairness…

(2). Microbenchmarks for specific performance

--Incast, queue buildup, buffer pressure

(3). Evaluate DCTCP

All comparisons are between DCTCP and TCP New Reno!

Experiment Results

(1). **Examine basic properties of the DCTCP**

	DCTCP	TCP
Throughput(1 Gbps links)	Maximum 0.95G/1G(*K=20*)	Maximum 0.95G/1G
Queue length(1 Gbps links)	Stable around 20 packets (*K*=20)	10 times larger and varies widely
Fairness and convergence (a single long-lived flow)	Quick, fair	Quick, fair but higher variation

(2). Microbenchmarks for specific performance

(3). Evaluate DCTCP

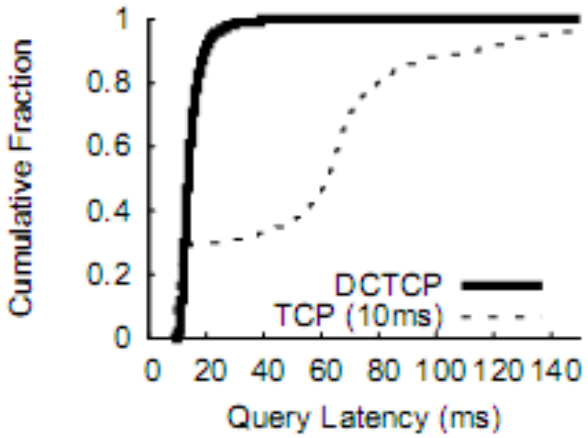
Experiment Results

(1). Examine basic properties of the DCTCP

(2). Microbenchmarks for specific performance

- Incast (RTO = 10ms)

- CDF of the response time



(3). Evaluate DCTCP

Experiment Results

(1). Examine basic properties of the DCTCP

(2). Microbenchmarks for specific performance

-Queue buildup

-CDF of completion times for 1000 20KB transfers

Data transferred is **small**, the completion time is dominated by the **RTT**, which is dominated by the **queue length** at the switch.

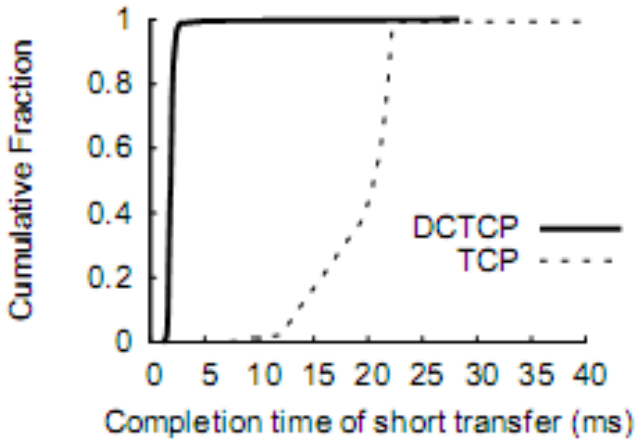


Figure 21: Short transfers see low delay with DCTCP.

(3). Evaluate DCTCP

Experiment Results

(1). Examine basic properties of the DCTCP

(2). Microbenchmarks for specific performance

--buffrer pressure

	Without background traffic	With background traffic
TCP	9.87ms	46.94ms
DCTCP	9.17ms	9.09ms

Table 2: 95^{th} percentile of query completion time. DCTCP prevents background traffic from affecting performance of query traffic. $RTO_{min} = 10ms$, $K = 20$.

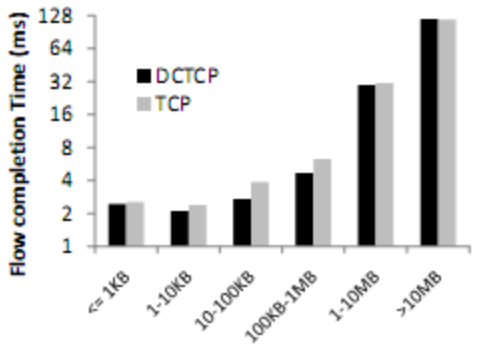
(3). Evaluate DCTCP

Experiment Results

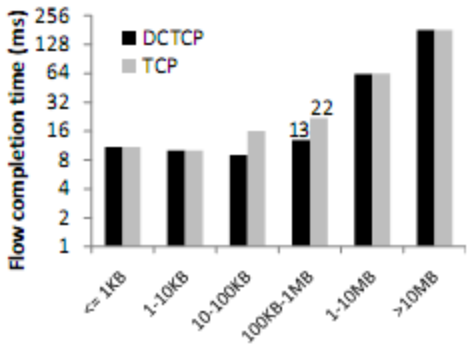
(1). Examine basic properties of the DCTCP

(2). Microbenchmarks for specific performance

(3). Evaluate DCTCP(RTO=10ms)



(a) Mean



(b) 95th Percentile

Figure 22: Completion time of background traffic. Note the log scale on the Y axis.

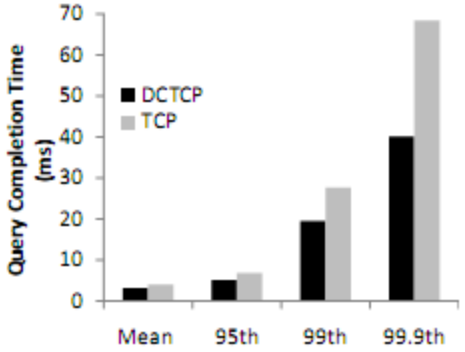


Figure 23: Completion time: query traffic

Summary

Why Data center choose DCTCP as their transport protocols?

-- Implementable with mechanisms in existing hardware (ECN)

-- Achieve low latency for small flows and high throughput for large flows

-- DCTCP requires only 30 lines of code change to TCP

Questions?

Thank you!