Cloud e Datacenter Networking

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI
Laurea Magistrale in Ingegneria Informatica

Prof. Roberto Canonico

TCP performance in datacenter networks

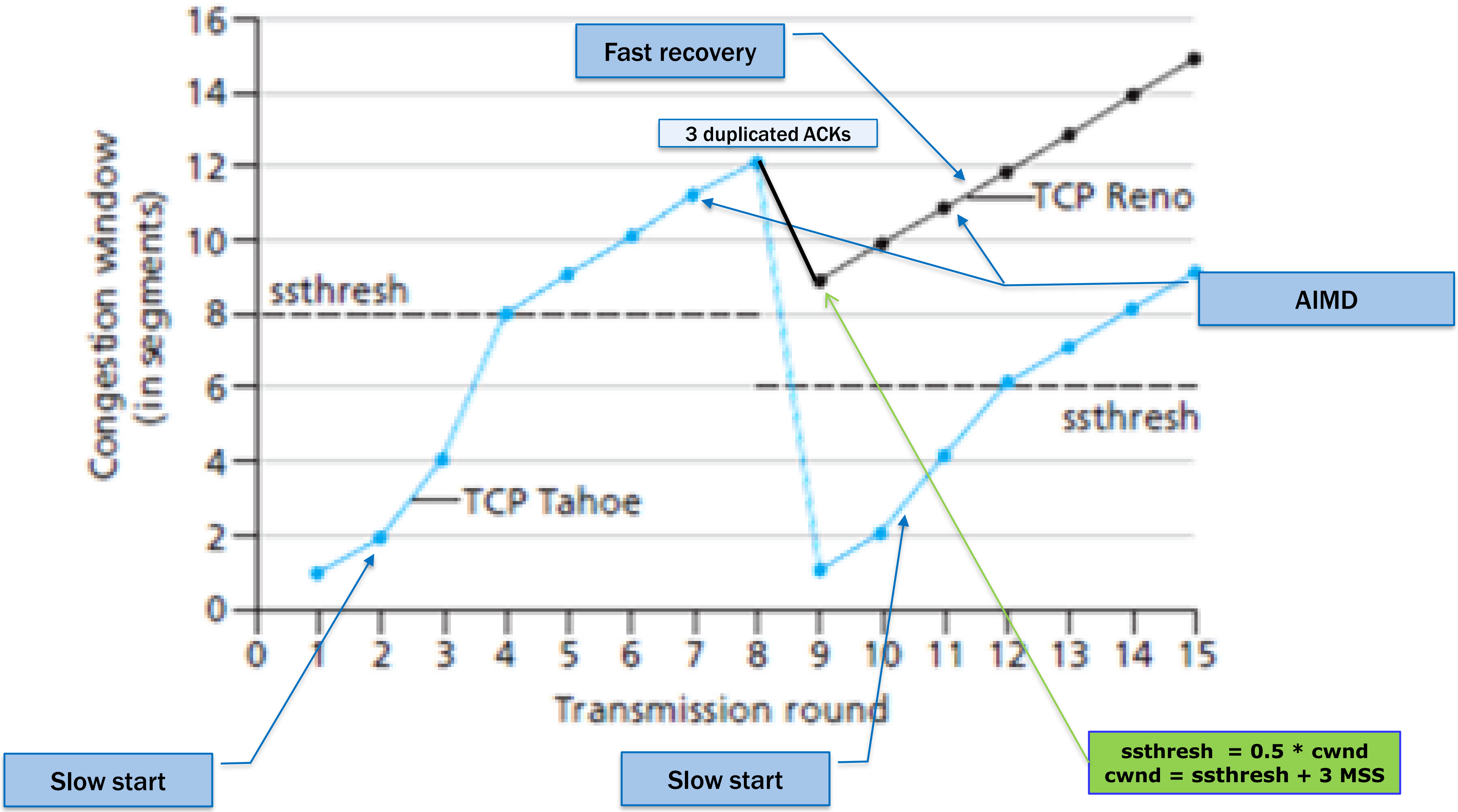
Lesson outline

- Quick recap of TCP congestion control mechanism
  - We assume it is well known from previous courses
- TCP Incast
- Datacenter TCP (DCTCP)

TCP congestion control

- In IP networks, TCP end-points adjust their sending rate according to the TCP congestion control mechanism
- One of the goals of TCP is *fairness*, i.e. guarantee that *n* competing flows of packets traversing a shared link receive a fair amount of the link's bandwidth
- The problem is that end points do not know what is the available bandwidth on a shared link
- TCP end points continuously make an estimate about the possibility to keep transmitting packets by observing packet acknowledgements they receive from their counterpart
  - Each TCP endpoint maintains a *Congestion Window Cwnd* variable, which limits the amount of data that can be transmitted before receiving an ACK
  - If ACKs arrive regularly, more packets may be transmitted (Cwnd is increased)
  - If ACKs do not arrive regularly, less packets may be transmitted (Cwnd is decreased)
- Two different phases: *slow start* and *AIMD*
- Two different congestion events: timeout and arrival of three duplicate ACKs at sender
  - After timeout: slow start phase is repeated
  - After 3 repeated ACKs: TCP Reno performs *Fast Recovery*, i.e. AIMD continues from half the Cwnd size at the moment the third repeated ACK arrived

TCP congestion control mechanism in action



What if TCP is not respected ?

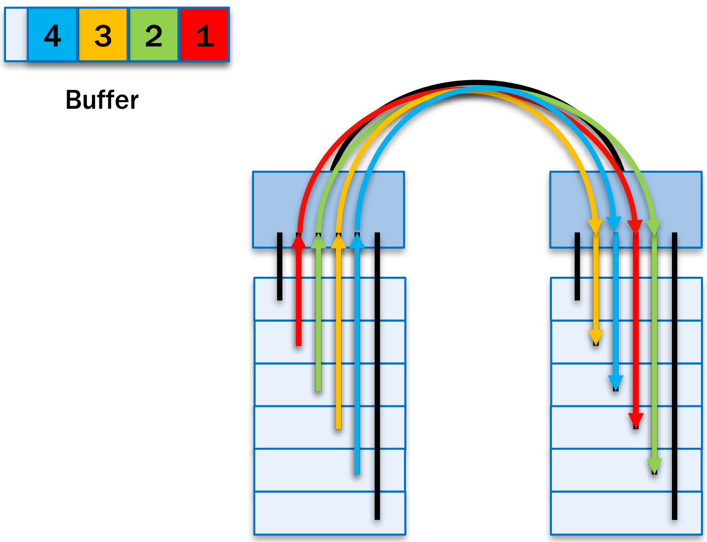
- This may happen either because of large UDP unresponsive flows or not-compliant TCP endpoints
  - TCP friendliness is important even for UDP flows
- In both cases, TCP flows do no get a fair amount of bandwidth
  - Throughput may decrease to zero if links are congested by unresponsive flows
- In general
  - if sources send at a higher rate than appropriate, they experience greater packet loss hence more retransmissions and more severe congestion → reduced *goodput* for all
  - if sources send at a lower rate than appropriate, their flows do not get the throughput they could achieve given current network conditions

Some inefficiencies of TCP congestion control

- In slow start, Cwnd starts from 1 MSS and is doubled at each RTT
  - i.e. every time an ACK arrives back to sender, Cwnd is increased of 1 MSS
  - It takes several RTTs to get a decent throughput
  - If flows are *short lived*, they may die before AIMD phase starts
    - This is the main reason for HTTP/1.1 persistent connections
- In AIMD, in case of three repeated ACKs (weak congestion evidence) Cwnd is decreased to half the Cwnd size (+3*MSS) at the moment the third repeated ACK arrived
  - The flow will take several RTTs to regain a sufficiently high Cwnd size
  - This behavior may be considered too conservative
    - Packet losses in a datacenter network are almost exclusively due to a switch queue overrun, hence they are transient problems that last only for a short time

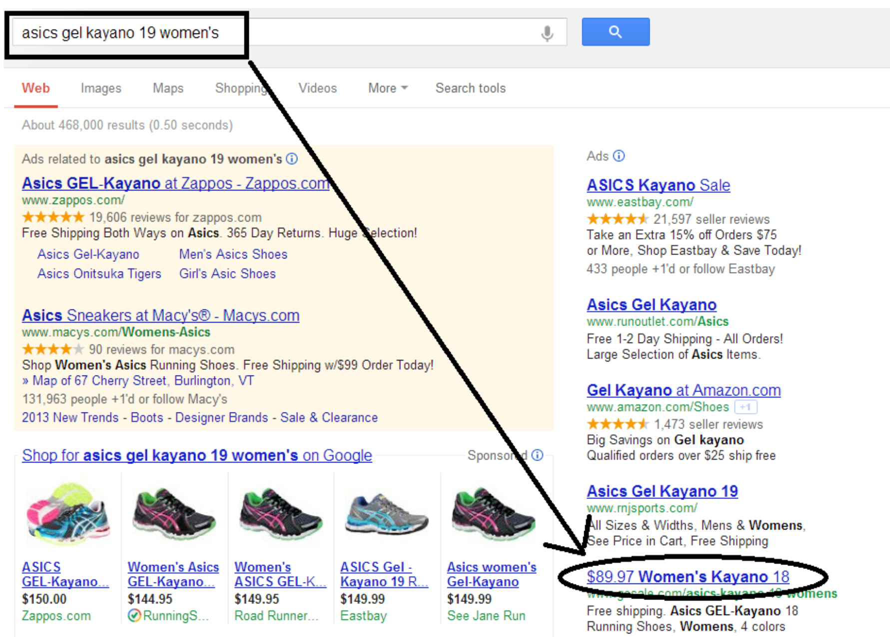
TCP performance in a datacenter network

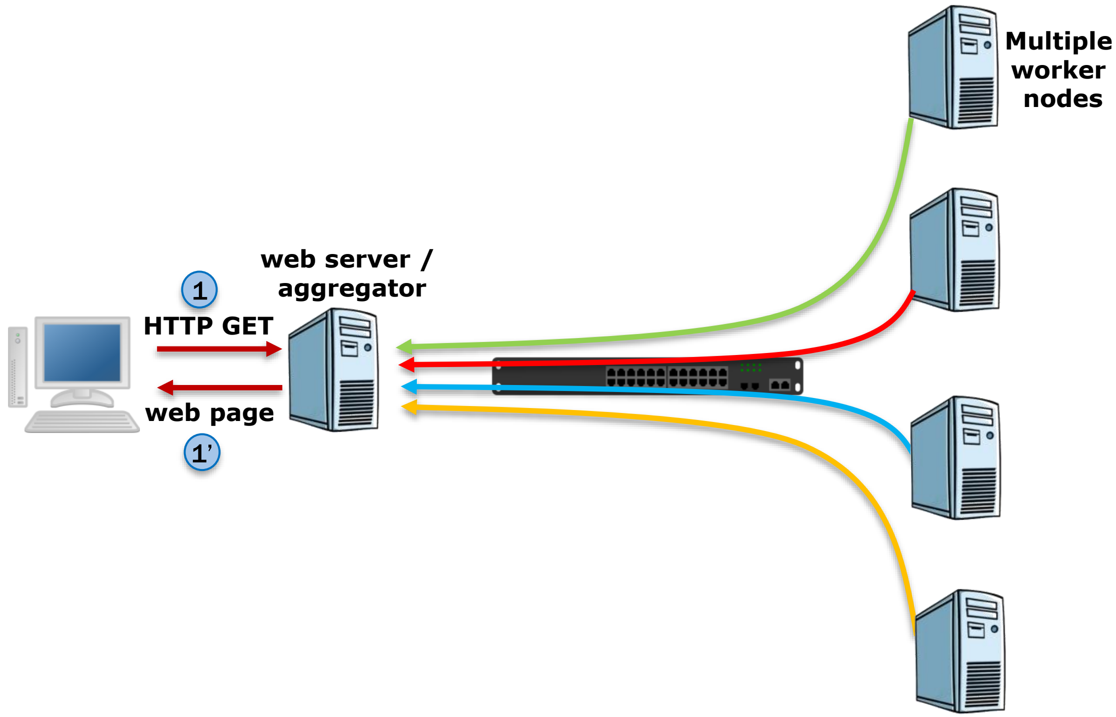
- In datacenter networks packet losses are almost exclusively due to a switch queue overrun
- Propagation delay: 100 meters of network cabling between two nodes adds only 0.5 μs of propagation delay
- Transmission time for a 9000 byte packet at 10 Gbps: ≈ 7.2 μs
- If a packet finds several other packets in a switch queue, **queueing delay dominates transmission time**
  - The problem may exacerbate if the end-to-end paths includes 3–4 filled up queues
- TCP end-to-end performance is limited by buffer occupancy
- The situation is worsened if several concurrent flows want to reach the same end-point: TCP incast



Partition/aggregation pattern workload and latency

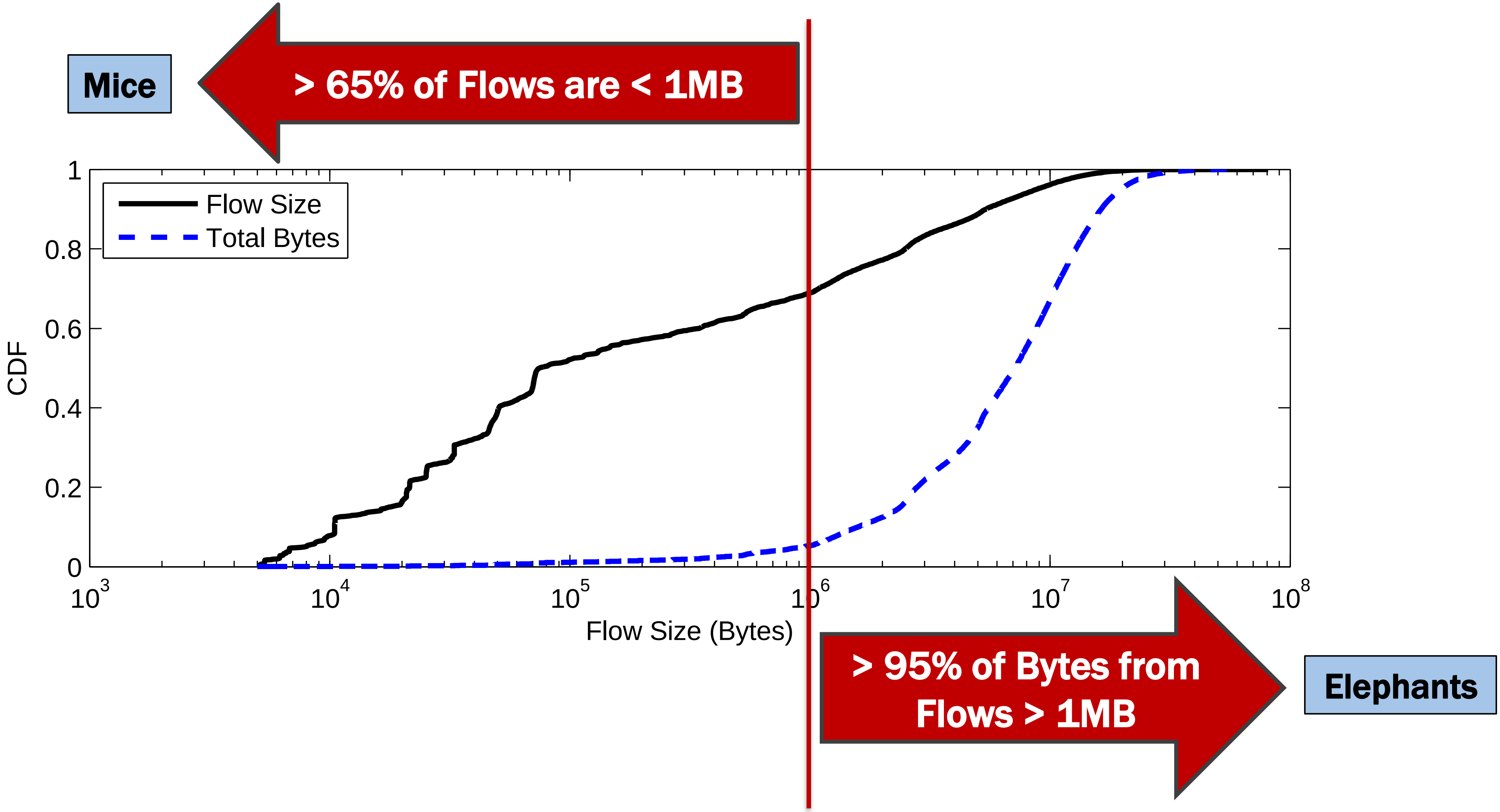
- Web navigation produce web pages that are dynamically built by collecting information from several databases
  - Think of a Google query or a web page with customized ads
- Low latency is crucial for Quality of Experience and service success
  - Every 100 ms increase in load time of Amazon.com decreased sales by one percent
  - Tests at Microsoft on live search showed that when search results pages were slowed by 1 s queries per user declined by 1.0% and ad clicks per user declined by 1.5%
  - Google found an extra 0.5 s in search page generation time dropped traffic by 20%





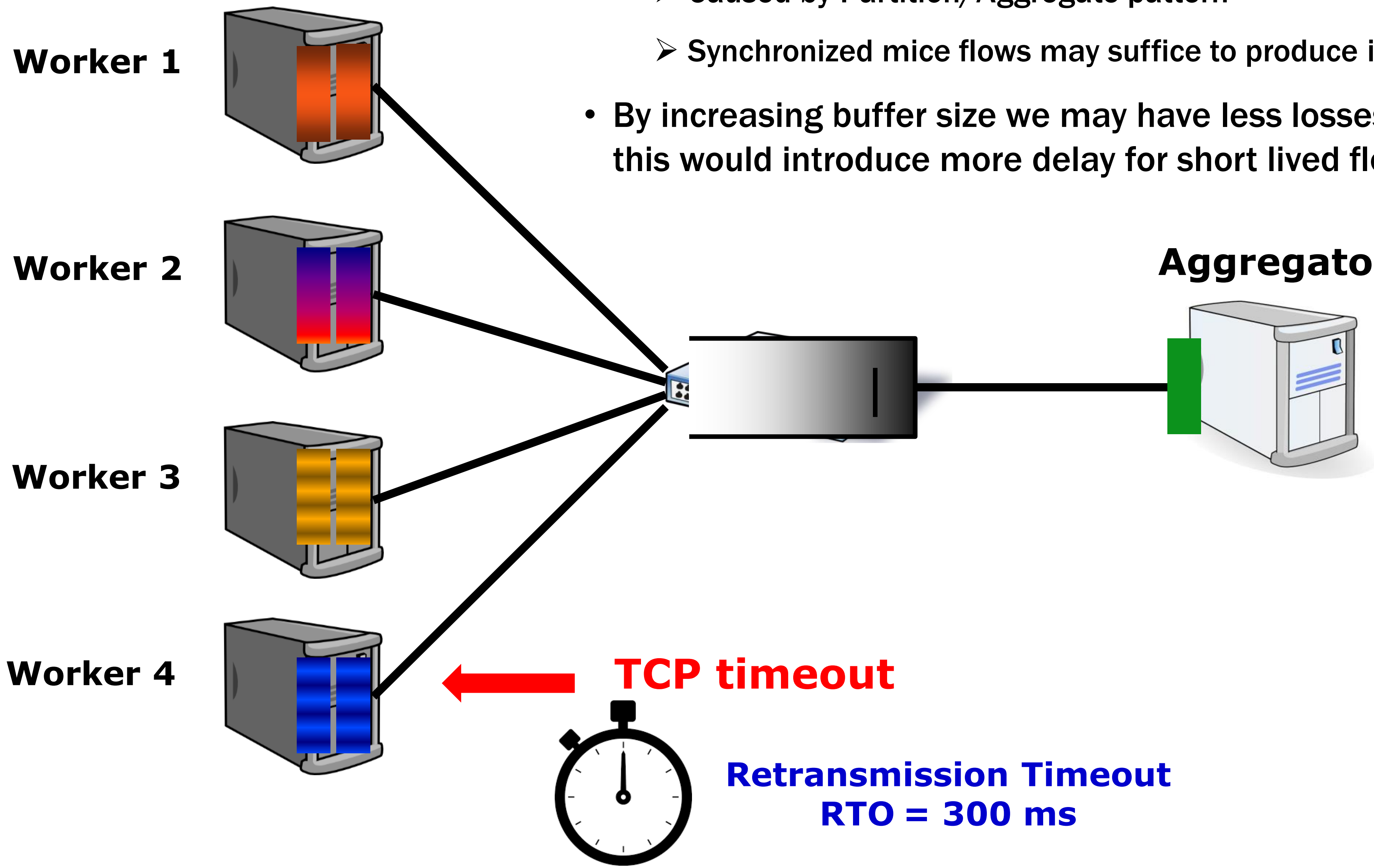
Flow size distribution

- A large amount of web traffic is due to a small number of big flows (*elephants*)
- The remaining traffic volume is due to many short-lived small flows (*mice*)



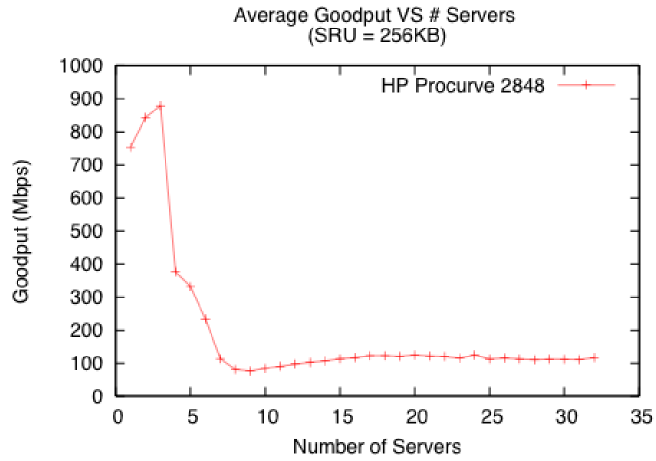
The incast problem

- Synchronized fan-in congestion:
  - Caused by Partition/Aggregate pattern
  - Synchronized mice flows may suffice to produce incast
- By increasing buffer size we may have less losses but this would introduce more delay for short lived flows

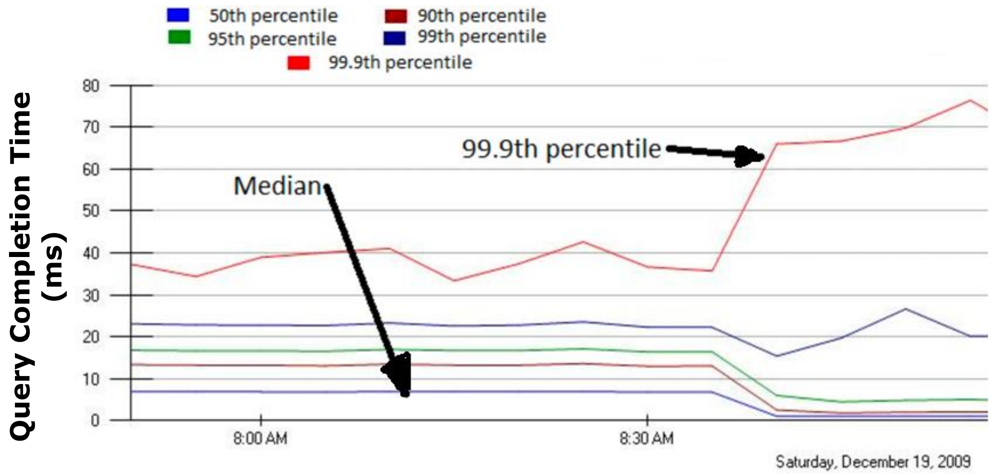


TCP Incast evidence in real datacenters

- Problem known since 2008
  - Amar Phanishayee, Elie Krevat, Vijay Vasudevan, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, Srinivasan Seshan. *Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems*. USENIX FAST, 2008
  - System considered: one client performing parallel read operations on N concurrent servers
  - When the number of servers exceeds 5 servers, goodput collapses
  - Maximum goodput for 3 servers
  - For N > 8, goodput is almost independent from N



- Incast in Bing @ Microsoft
  - At some point, application introduces jitter to avoid synchronization of flows
  - This action produces a positive effect by mitigating the incast problem

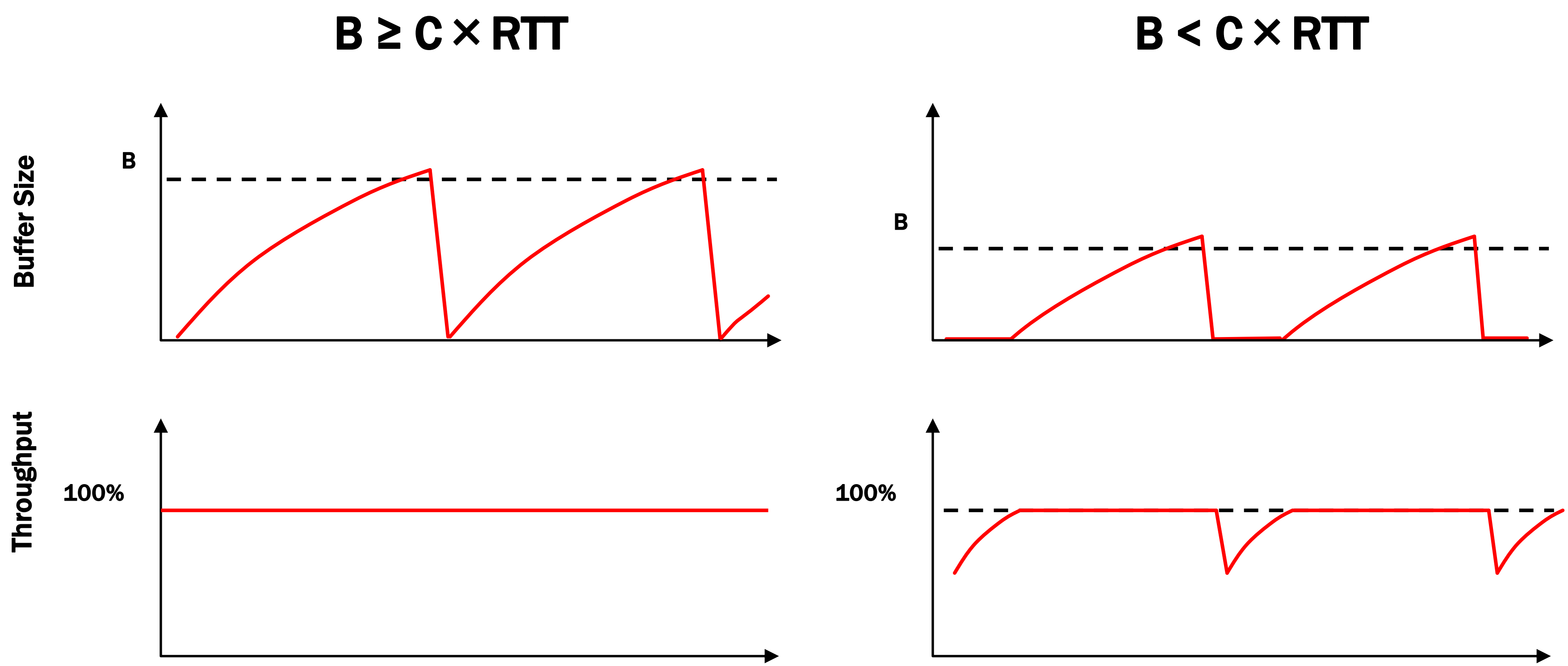


Buffer sizing problem: two conflicting goals

- To achieve high throughput, no packet losses should occur hence switches should have buffers of large size to absorb traffic bursts
- To achieve low latency, packets should not stay in a queue for a long time, hence buffers size should not be too large
- How large should the buffers be in the switches ?
- Small buffers:
  - many packets are dropped due to bursts
  - but lead to small delays
- Large buffers:
  - reduced number of packet drops (due to bursts)
  - but increase delays

Buffer sizing problem

- Bandwidth-delay product rule of thumb:
  - A single flow needs **C × RTT** buffers for 100% Throughput
- The challenge is to make buffer size not too big
- Appenzeller (SIGCOMM '04): for large # of flows: $C \times RTT/\sqrt{N}$ is enough
- In a datacenter, the hypotheses of a large number of flows is not applicable
  - Measurements show typically 1-2 big flows at each server, at most 4



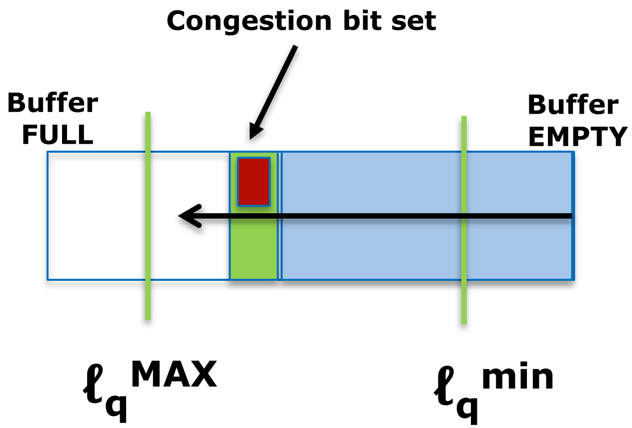
Data Center TCP (DCTCP)

- DCTCP is a TCP variant specifically targeted for datacenter networks, where queuing delay dominates transmission delay and packet loss is almost exclusively due to buffer overrun
- DCTCP leverages ECN (*Explicit Congestion Notification*): network devices may mark packets to signal that congestion is approaching (i.e. buffers are about to be filled up)
- In this way, traffic sources may decrease the transmission rate <u>before</u> packet loss occurs
- An ECN switch measures the average queue length over a recent time window and decides whether or not packets should be marked with a congestion notification bit set to 1
- DCTCP does the same, but takes a decision based on instantaneous queue length rather than its average
  - This simplifies the role of switches

Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 conference* (SIGCOMM '10).

Data Center TCP (DCTCP)

- Queue length ℓ_q is compared against 2 queue length threshold values:
  - a high threshold ℓ_q^{MAX} an a low threshold ℓ_q^{min}
- If $\ell_q > \ell_q^{MAX}$ all packets are marked with the congestion bit set
- If $\ell_q < \ell_q^{min}$ none of the packets is marked with the congestion bit set
- If $\ell_q^{min} \leq \ell_q \leq \ell_q^{MAX}$ packets are marked *probabilistically*
- If the congestion bit is set in a packet on its way from the sender to the receiver the same congestion bit is copied into the ACK packet that travels back to the sender

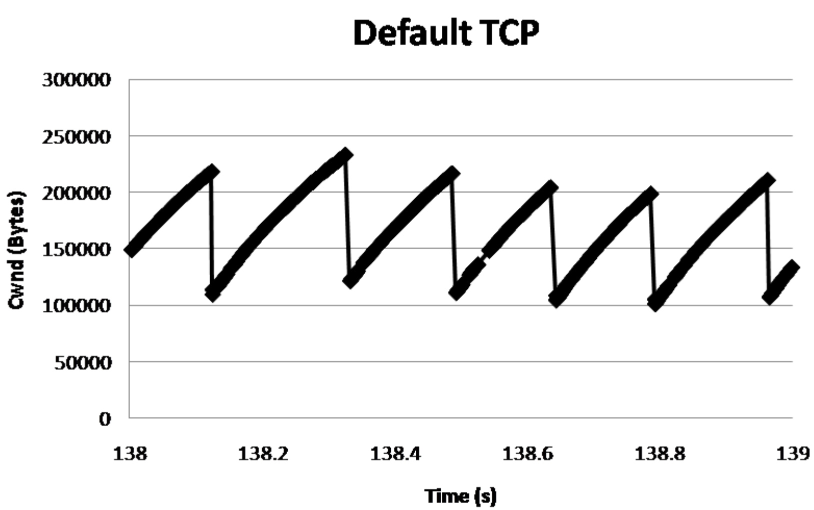


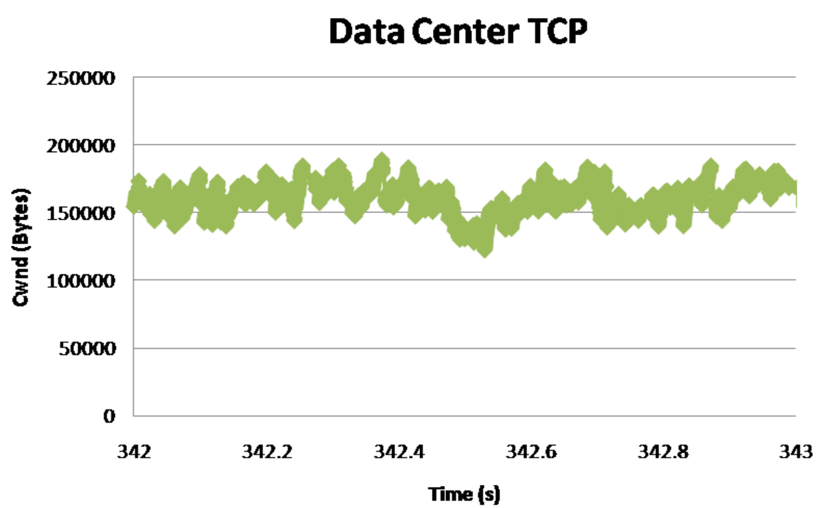
- The sender has a chance to react before packet loss by halving the congestion window size
- Reaction is the same as for a packet loss without paying the throughput cost of a packet loss

DCTCP reaction to congestion

- Congestion not considered as a binary information rather as a stream of bits
- DCTCP reacts in proportion to the extent of congestion
  - Reduce window size based on fraction of marked packets
- This reduce the problem of AIMD being too aggressive in reducing Cwnd

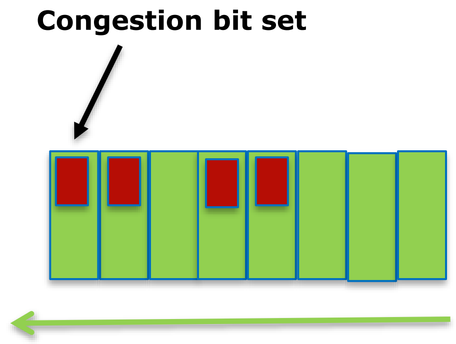
ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%





DCTCP sender

- A DCTCP sender computes a running average of the fraction of packets that have been marked with the congestion bit set and reduces the congestion window accordingly
- 1/3 of marked packets $\rightarrow$ Cwnd reduced of 33%



DCTCP drawbacks

- DCTCP is not TCP friendly
  - When DCTCP competes for bandwidth against regular TCP, DCTCP flows get higher throughput than TCP flows (*unfair*)
- DCTCP does not take into account application requirements