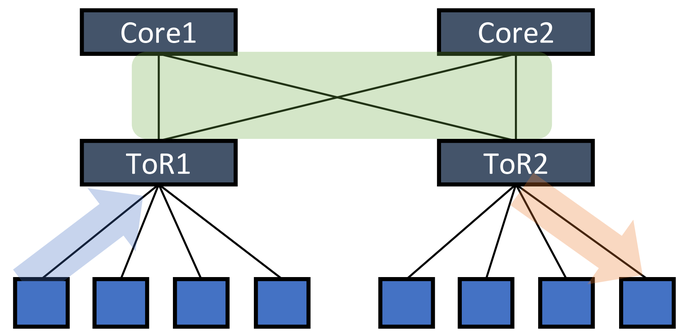
SIRD: A fully receiver-driven datacenter transport with shallow network queues

Konstantinos Prasopoulos*, Edouard Bugnion*, Marios Kogias+

EPFL * Imperial College London +

Datacenter Congestion Control

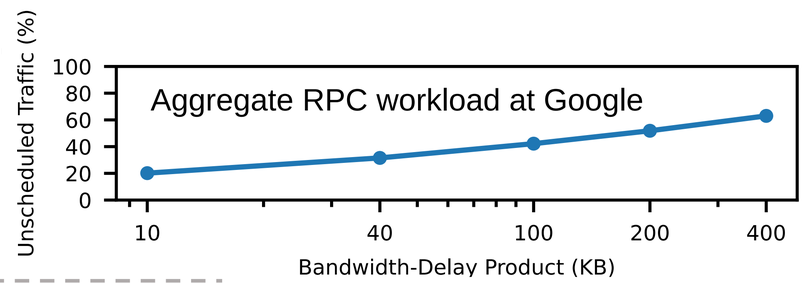
- **Sender-driven**
  Traditional approach (TCP). Senders adjust their rate based on feedback from the network. Senders independently converge to a fair bandwidth allocation.
- **Receiver-driven**
  Receivers explicitly schedule packets to arrive on their exclusive link. Zero convergence time if this link is the bottleneck.



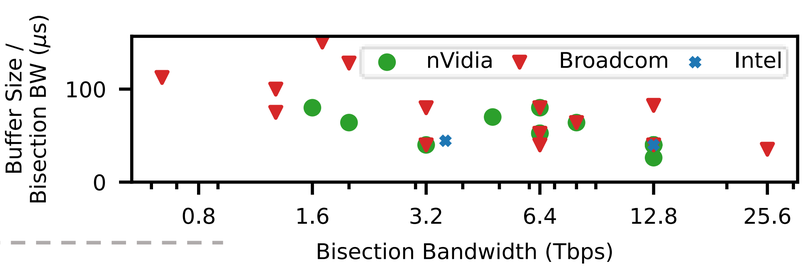
Receivers are the most common bottleneck in datacenter networks.
⇒ Receivers should make decisions.

Problem: Current receiver-driven schemes are compelling but they..

1. Unconditionally transmit the first BDP[1] bytes of every message.
   → As bandwidths increase, more traffic is transmitted this way.



2. Make extensive use of switch buffers to address congestion at senders in a brute-force manner.
   → Switch buffer sizes vs bandwidth are trending down.



3. Cannot handle scenarios in which the core of the fabric is the bottleneck.

1. Bandwidth-Delay Product

Sender-Informed Receiver-Driven

Each receiver has a limited amount of credit..

..to distribute to senders upon explicit request ❶ and according to policy (cost: extra RTT[1]). Senders consume credit to send data packets.

⇒ Queue length at switch-receiver link is capped.

X>Thr Congested senders that hoard credit from multiple receivers inform them using ECN[2].

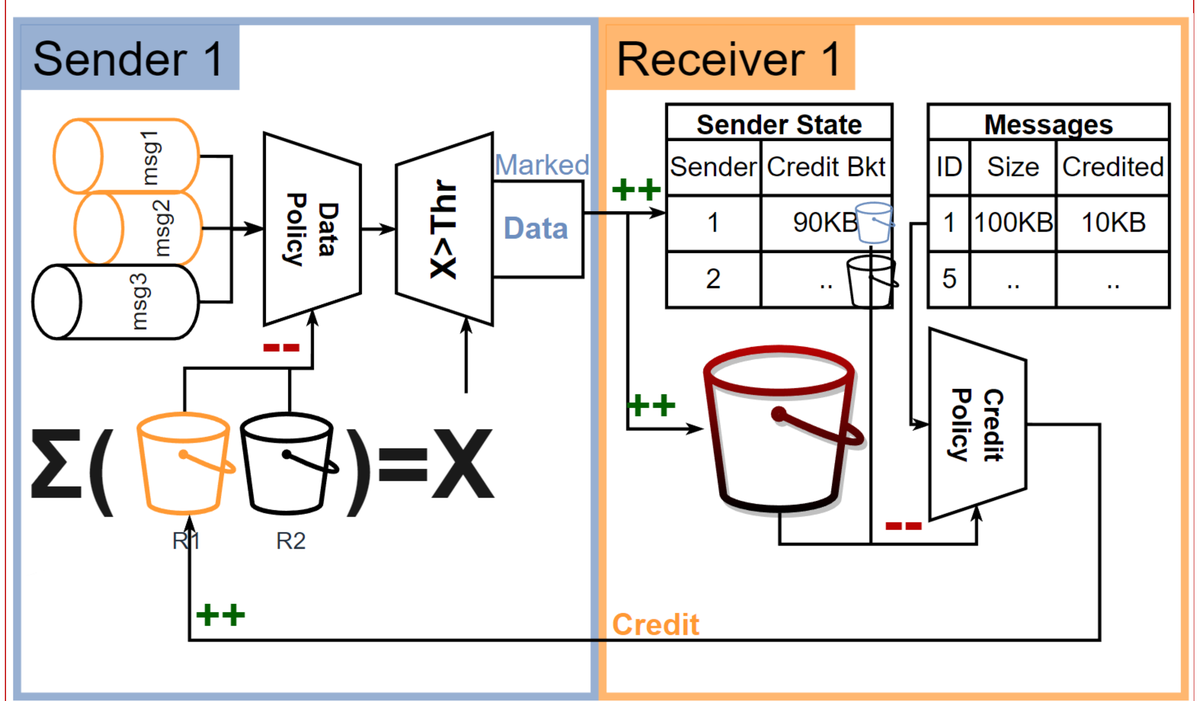
Receivers then reduce the amount of credit that can be allocated to the congested sender.

⇒ Receivers reclaim and reallocate unusable credit.

⇒ Receivers maximize throughput despite small. ❷

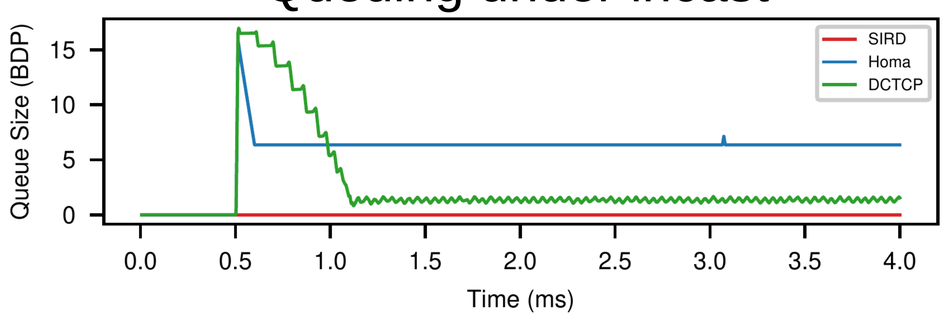
1. Round Trip Time | 2. Explicit Congestion Notification

Receivers use the same mechanism to reduce the rate of a sender if the network core is congested (signal: ECN from switches) ❸

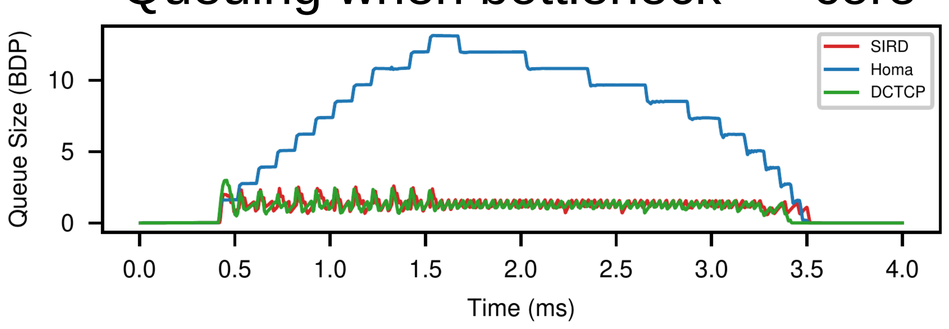


Microbenchmarks

Queuing under incast



Queuing when bottleneck == core



Large-scale simulations

144 hosts across 9 ToRs
Uniform server selection
Poisson interarrivals

Production workloads:
WKa: All RPCs @Google
WKb: Hadoop @Facebook
WKc: Search @Google