Better Never than Late: Meeting Deadlines in Datacenter Networks

Christo Wilson, Hitesh Ballani,
Thomas Karagiannis, Ant Rowstron

Microsoft Research, Cambridge

User-facing online services

Two common underlying themes

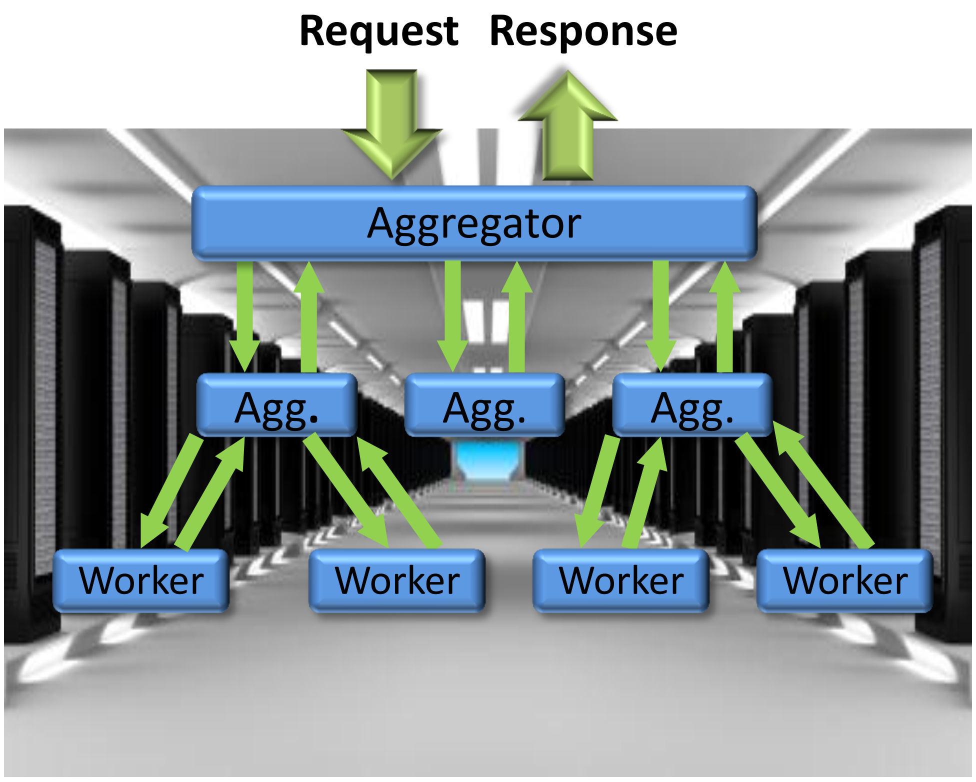
1. Soft real-time nature

- Online services have **SLAs** baked into their operation
- **Example**: 300ms response time for 99.9% requests

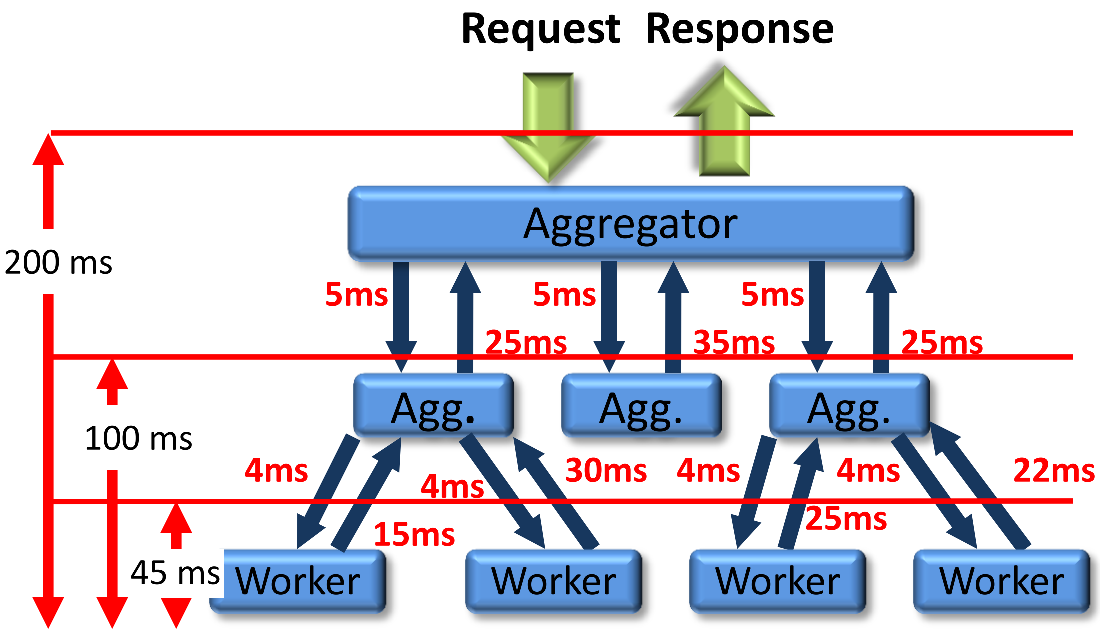
Impact of breached SLAs:

- Amazon: extra 100ms costs 1% in sales

2. Partition-aggregate workflow



User-facing online services



Application SLAs

Cascading SLAs

SLAs for components at each level of the hierarchy

Network SLAs

Deadlines on communications between components

Flow Deadlines

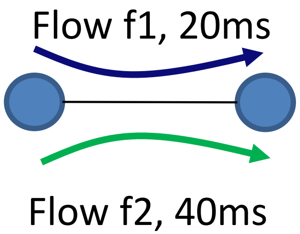
A flow is useful **if and only if** it satisfies its deadline

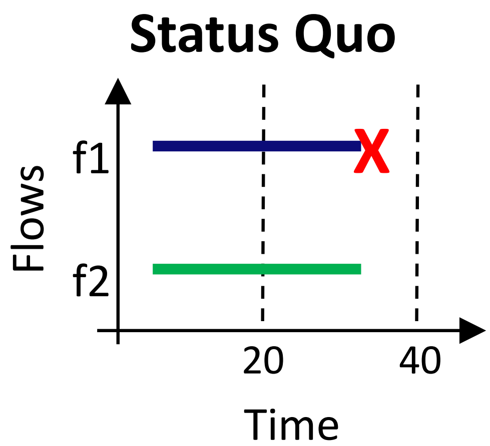
Today's transport protocols:

Deadline agnostic and strive for fairness

Limitations of Fair Sharing

Case for unfair sharing:



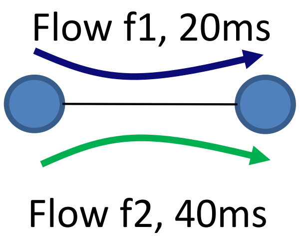


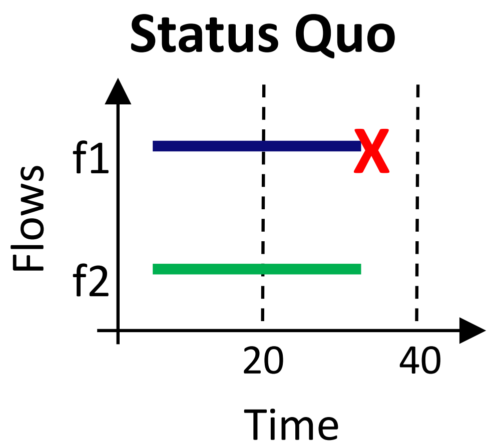
Flows f1 and f2 get a fair share of bandwidth

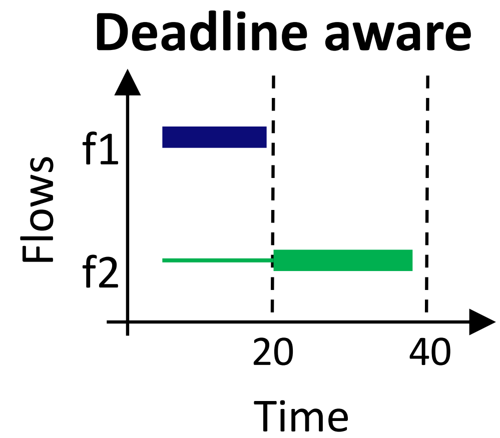
Flow f1 misses its deadline (incomplete response to user)

Limitations of Fair Sharing

Case for unfair sharing:



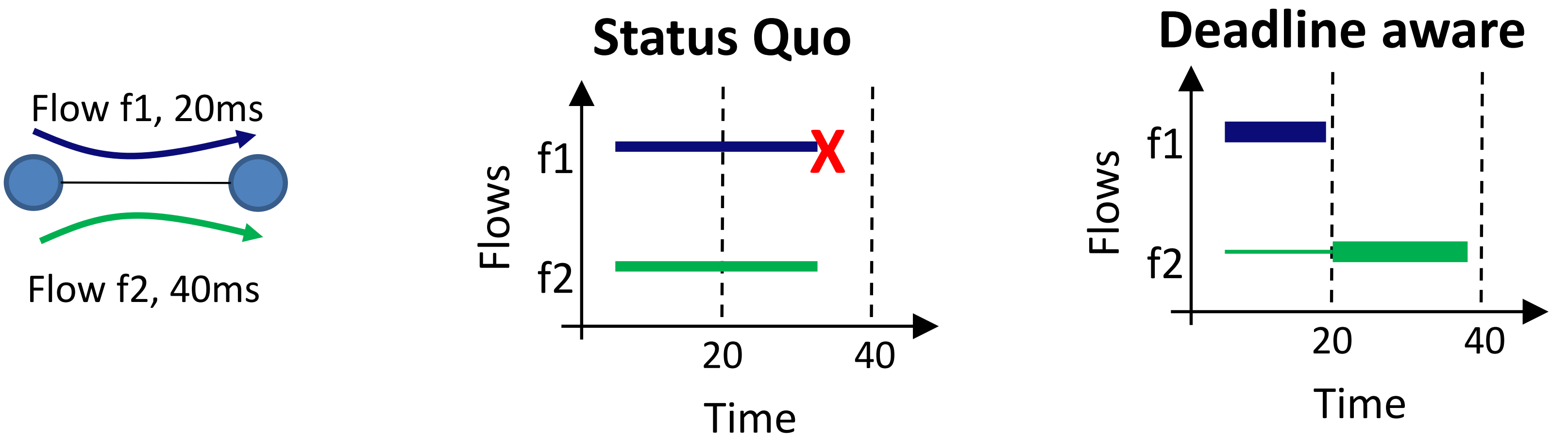




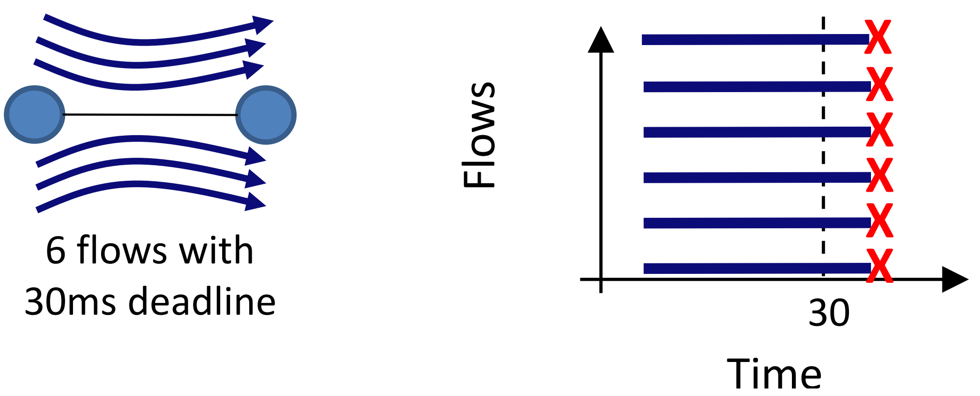
Flows get bandwidth in accordance to their deadlines
Deadline awareness ensures both flows satisfy deadlines

Limitations of Fair Sharing

Case for unfair sharing:



Case for flow quenching:

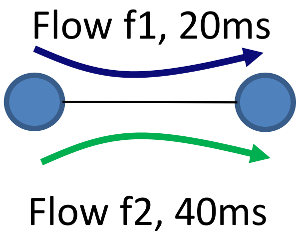


Insufficient bandwidth to satisfy all deadlines

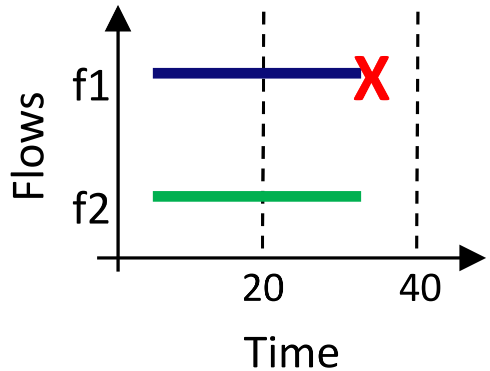
With fair share, all flows miss the deadline (empty response)

Limitations of Fair Sharing

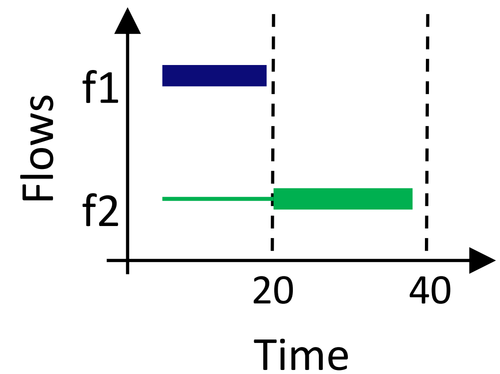
Case for unfair sharing:



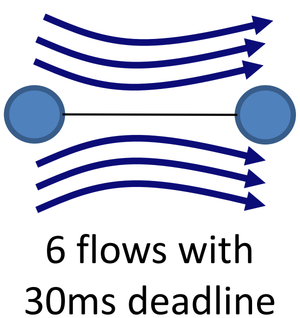
Status Quo

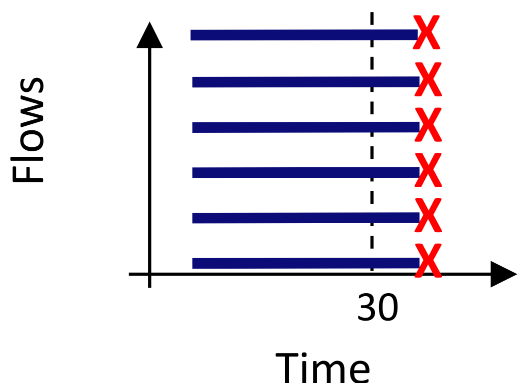


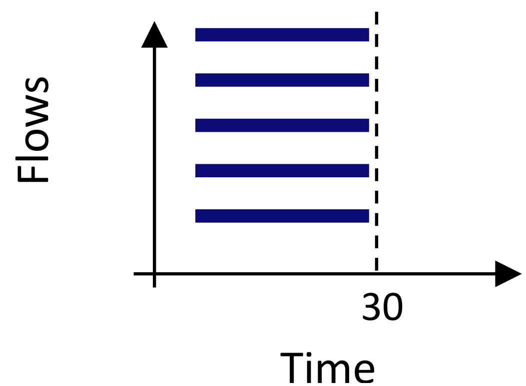
Deadline aware



Case for flow quenching:







With deadline awareness, one flow can be quenched

All other flows make their deadline (partial response)

D^3: Deadline-driven Delivery

Main idea: Make the network aware of flow deadlines

- Prioritize flows based on deadlines

A deadline-aware datacenter transport protocol that:

- schedules network traffic based on SLAs
- can double the peak load a datacenter supports
- performs well as a congestion control protocol

Advantages:

- Improve quality of responses
- Save resources

Challenges

- Deadlines are associated with flows, not packets
  - Packet scheduling mechanisms (e.g., EDF)

- Short flows (<50 KB) and minimal RTTs (~300 µsec)
  - Reservation schemes (e.g., IntServ, DiffServ)

- Deadlines can vary significantly
- Beyond documented TCP problems in datacenters (e.g., incast, buffer pressure)

D^3 Design

Design goals

— Maximize application throughput (i.e., deadlines satisfied)

— Burst tolerance

— High utilization

Non-goals: Incremental deployment, backwards compatibility, being friendly to legacy protocols

Key Insight:

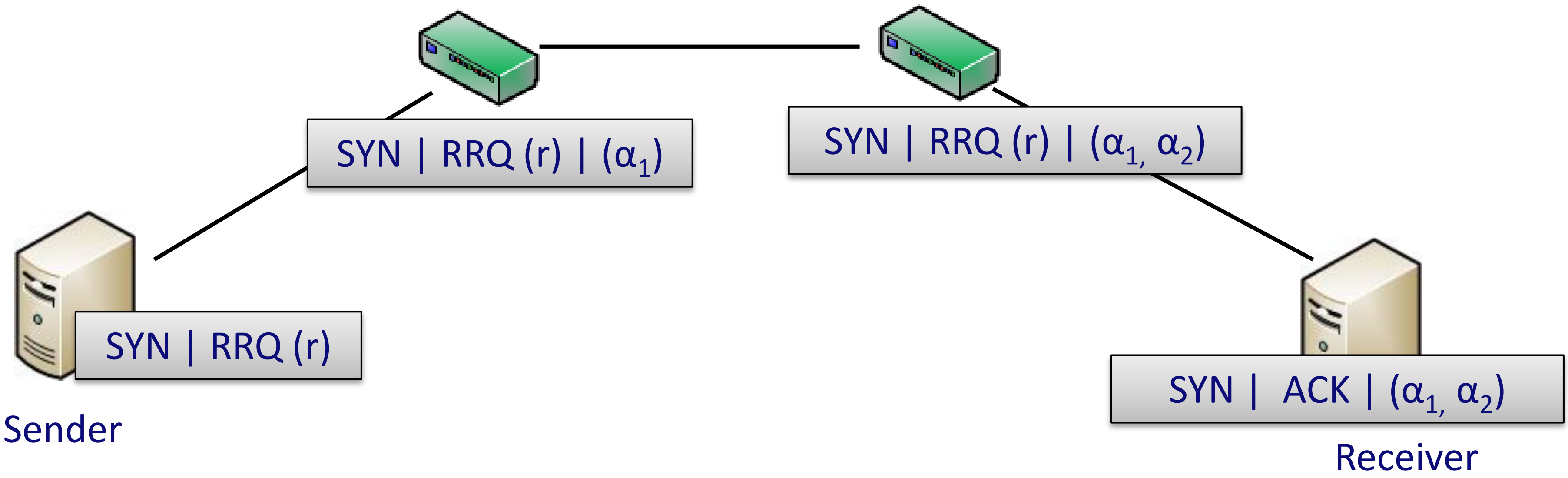
— Rate required to satisfy a flow deadline:

$$r = \frac{s}{d}$$

s : flow size

d: deadline

D³ overview



1. $Application\ exposes\ (s, d)$

2. $Desired\ rate : r = \frac{s}{d}$

3. Routers allocate rates (α) based on traffic load

4. $Sending\ rate\ for\ next\ RTT : \mathrm{sr} = \min(\alpha_1, \alpha_2)$

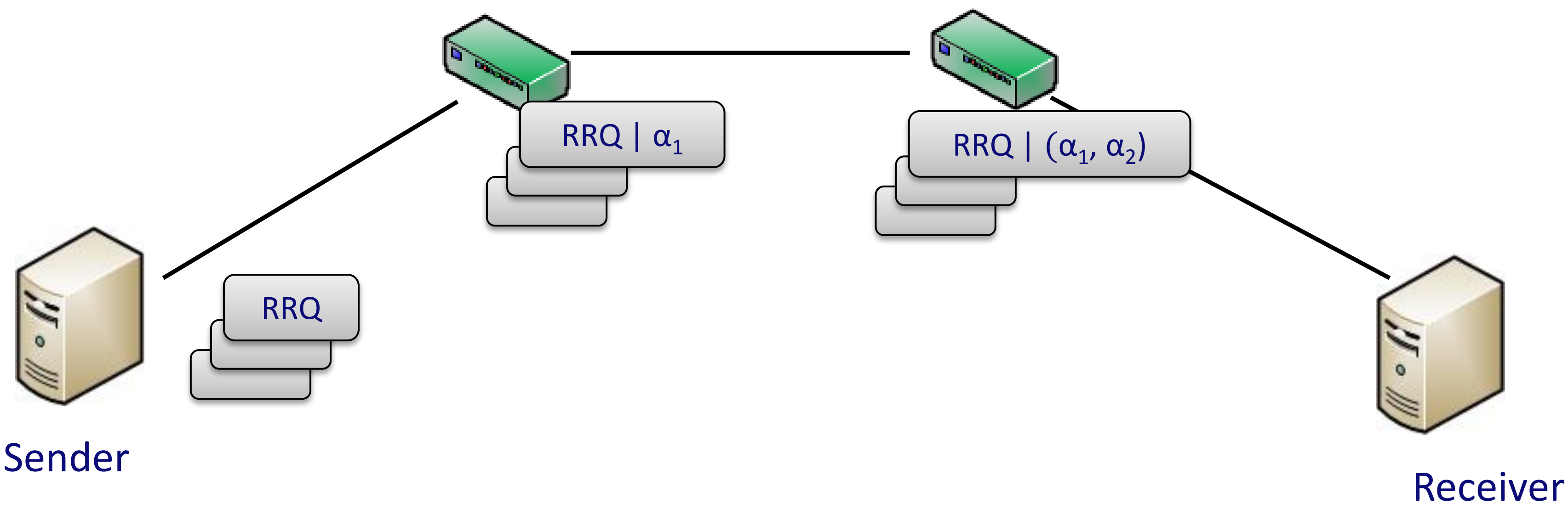
s : flow size

d : deadline

RRQ : Rate Request

α : allocated rate

D^3 overview



1. $Application\ exposes\ (s, d)$

2. $Desired\ rate : r = \frac{s}{d}$

3. Routers allocate rates (α) based on traffic load

4. $Sending\ rate\ for\ next\ RTT : \mathrm{sr} = \min(\alpha_1, \alpha_2)$

5. Send data at rate sr

6. One of the packets contains and updated RRQ based on the remaining flow size and deadline

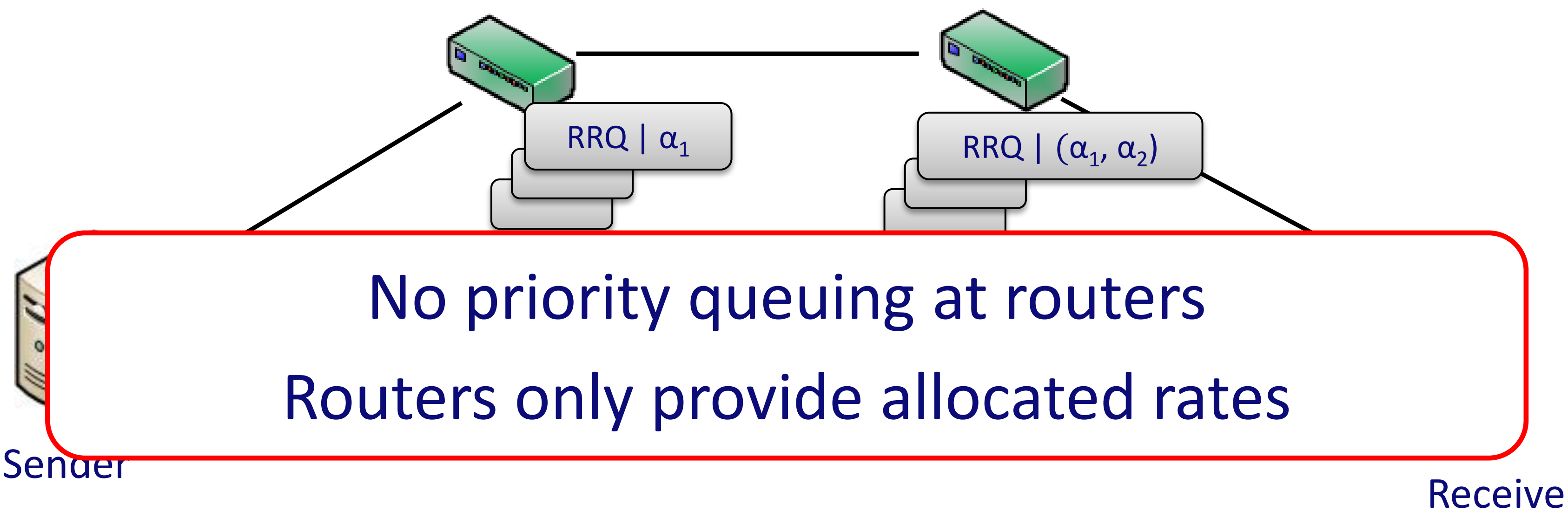
s : flow size

d : deadline

RRQ : Rate Request

α : allocated rate

D^3 overview



No priority queuing at routers

Routers only provide allocated rates

1. $Application\ exposes\ (s, d)$

2.

Rate control is performed at the end host which enforces the minimum of the allocated rates

3.

α : allocated rate

4. $Sending\ rate\ for\ next\ RTT$: $\mathrm{sr} = \min(\alpha_1, \alpha_2)$

5. Send data at rate sr

6. One of the packets contains and updated RRQ based on the remaining flow size and deadline

Rate allocation

Goals:

- Maximize the number of deadlines satisfied
- Fully utilize the network

Router needs to track :

2

1

3

Allocated rate (α):

- Available capacity > ∑ (desired rates)
  - Deadline flow : α = r + fs
  - Non-deadline flow (r=0) : α = fs
- Available capacity not enough to satisfy all requests
  - greedily satisfy requests
  - remaining flows are assigned a base_rate (header only packet)

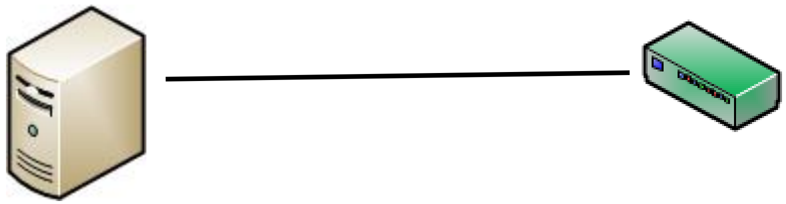
fs : fair-share after satisfying flow requests

r : desired rate

Rate allocation

Router needs to track:

1. Sum of desired rates (demand)
2. Available Capacity
3. Fair-share (fs)



RRQ (r) | α_{t-1} | r_{t-1} | (α_t)

$A = A - \alpha_{t-1}$

$D = D - r_{t-1} + r_t$

If available capacity > D:

$\alpha_t = r + fs$

$A = A + \alpha_t$

Three aggregate values (no per-flow state)

Allocations : $A = \sum a$

Demand : $D = \sum r$

Number of flows : N

=> $fs = \frac{C-D}{N}$

available capacity = C-A

r : desired rate

α : allocated rate

C : link capacity

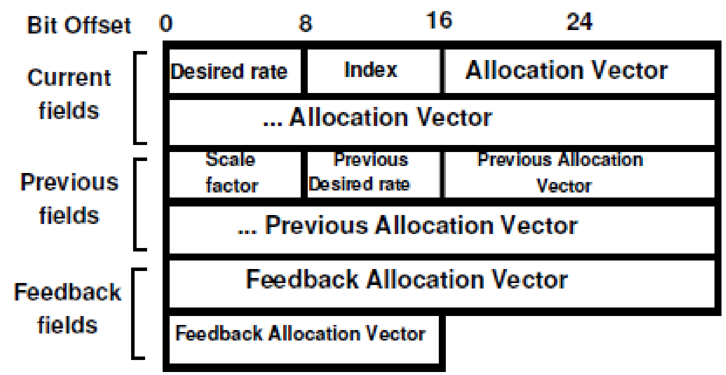
RRQ : Rate Request

Implementation

End host:

— Complete transport protocol

— Sockets-like API where applications expose flow length and deadlines



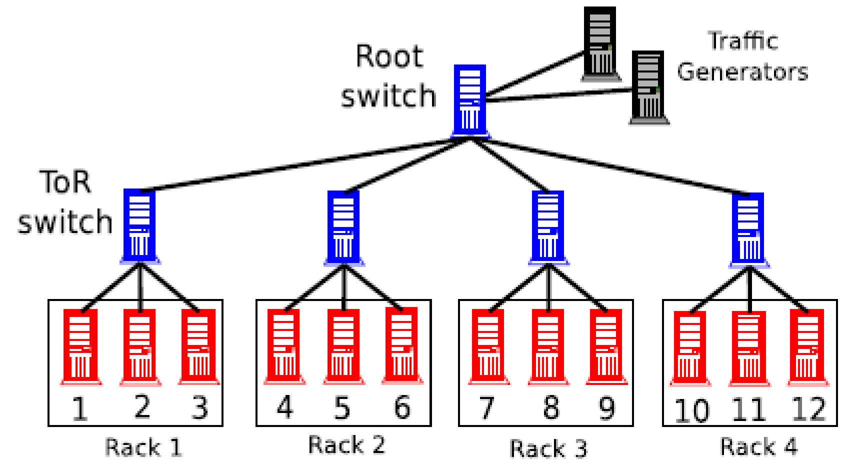
Router:

— User-space PC-based implementation

— Able to sustain four links at full duplex line rate

— Packet processing overhead < 1μsec

Evaluation: Testbed and metrics



Metric:

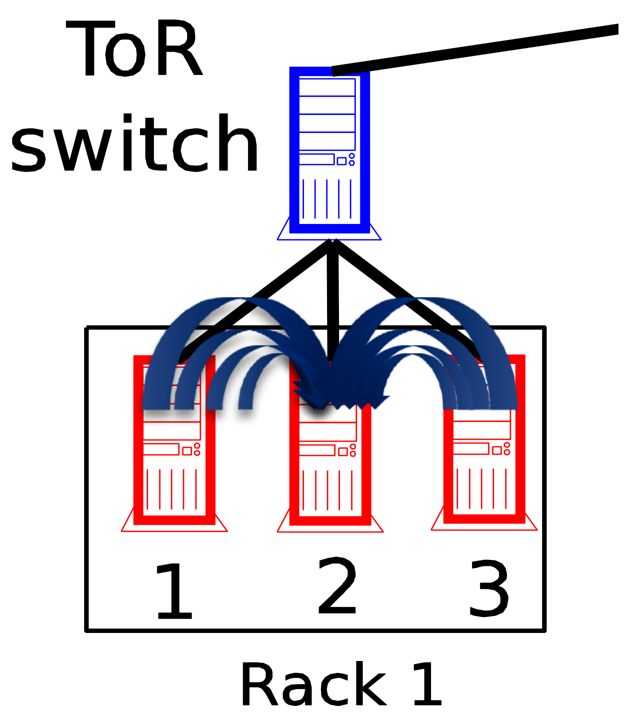
— Application throughput
  — Number of flows satisfying deadlines
— Operational regime
  — Application throughput > 99%

Protocols:

1. TCP
2. RCP_{dc} : D^3 in fair-share mode only (best-case for fair-sharing protocols)
3. TCP_{pr} : TCP wih priority queuing
4. D^3

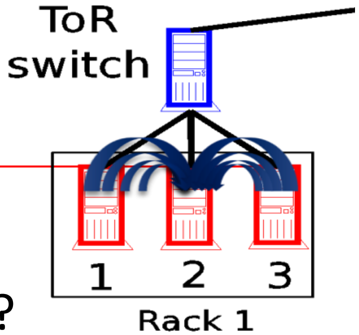
1. Flow microbenchmarks: Synthetic workload

Experiment: Multiple workers sending traffic



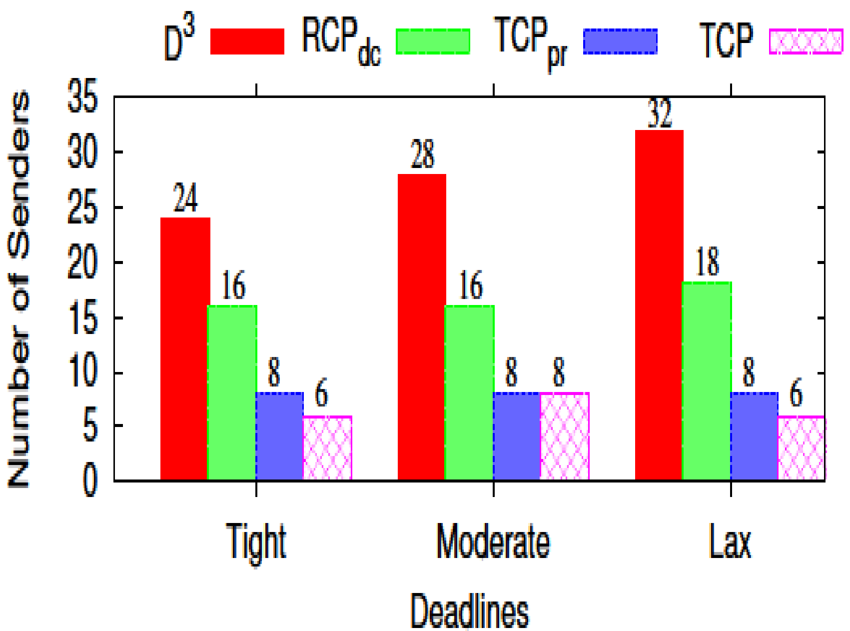
How many workers can be supported while satisfying >99% flow deadlines?

1. Flow microbenchmarks

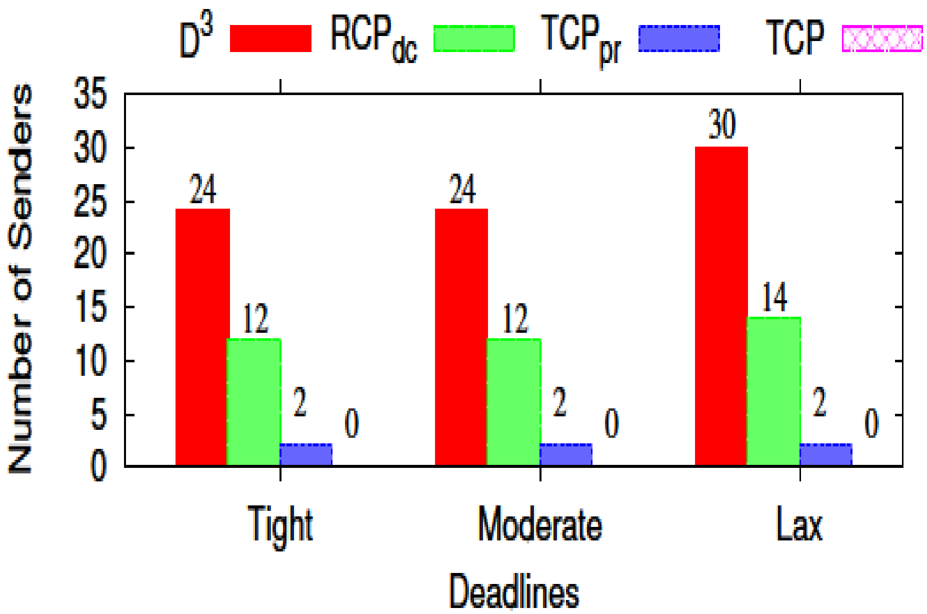


Experiment: Multiple workers sending traffic

How many workers can be supported while satisfying >99% deadlines?

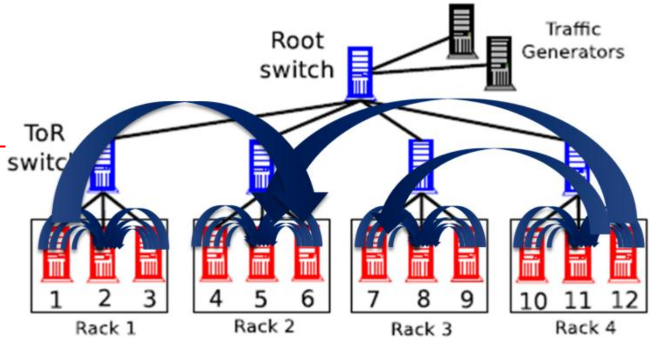


With background flows



D^3 can support roughly ***twice as many workers*** while satisfying application deadlines

2. Datacenter-like traffic dynamics

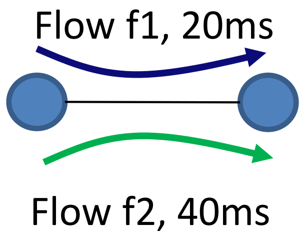


	TCP	TCP (optimized)	Fair Share Protocols	D^3
Peak Load Supported (flows/sec)	100	1100	1300	2000

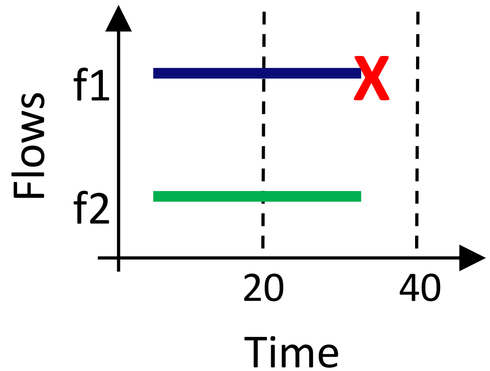
- D^3 maintains low flow completion times
  - completion times similar to RCP
- Long background flows are not penalized
  - throughput similar to TCP

Results so far…

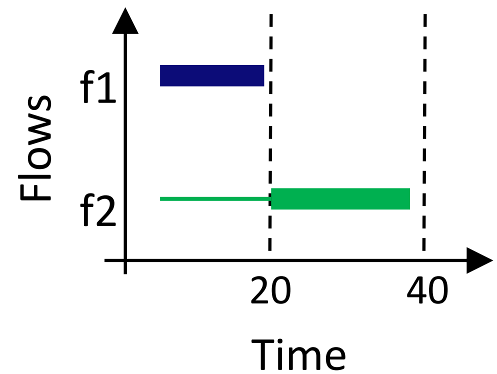
Case for unfair sharing:



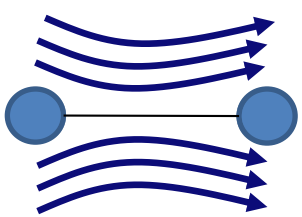
Status Quo



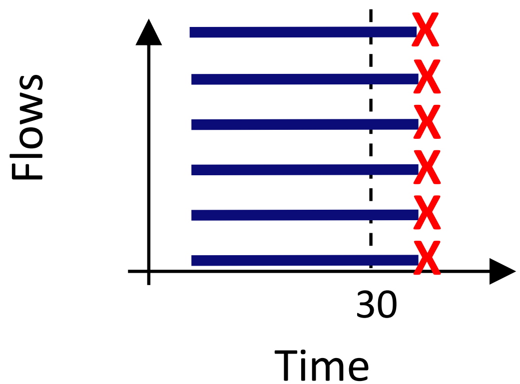
Deadline aware

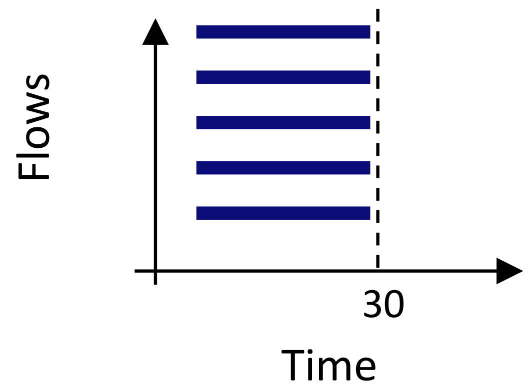


Case for flow quenching:

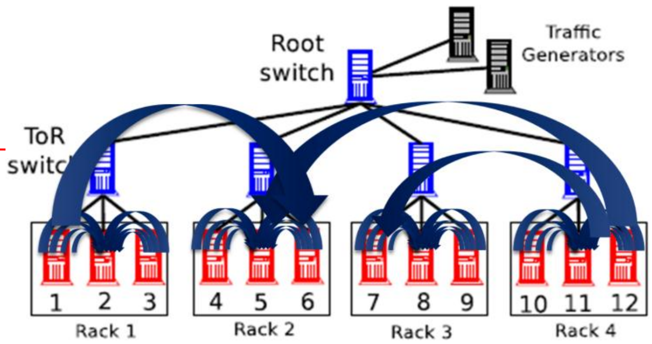


6 flows with
30ms deadline





3. Flow quenching

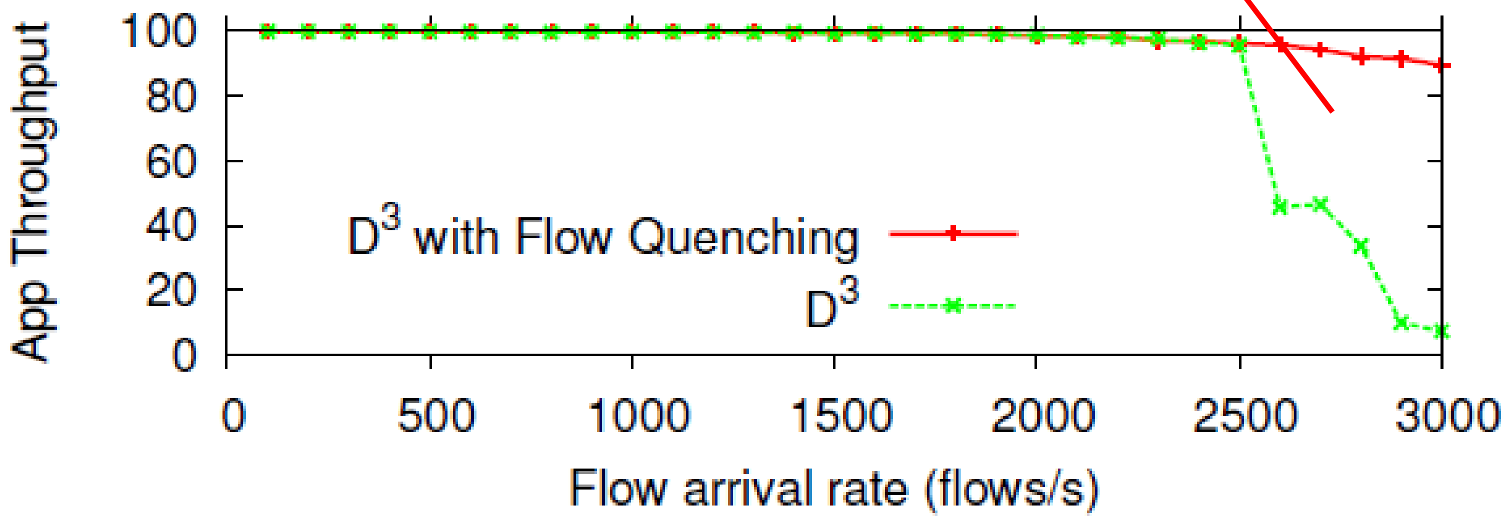


Terminate "useless flows" when:

— Desired rate exceeds link capacity

— Deadline has expired

Allows for graceful degradation of performance



Conclusions

Case for deadline-aware allocation of bandwidth

— tension between today's offered functionality and application requirements

D^3:

— schedules network traffic based on SLAs

— can double the peak load a datacenter supports

— design based on challenges and luxuries of datacenter environment

Thank you!

www.research.microsoft.com