Deadline-Aware Datacenter TCP (D^2TCP)

Balajee Vamanan, Jahangir Hasan, and
T. N. Vijaykumar

Google

Purdue University

Datacenters and OLDIs

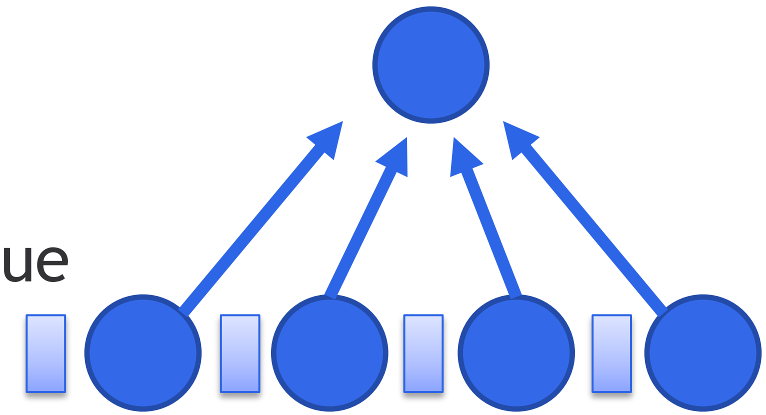
- OLDI = **O**n**L**ine **D**ata **I**ntensive applications
    - e.g., Web search, retail, advertisements
- An important class of datacenter applications
- Vital to many Internet companies

OLDIs are critical datacenter applications

Challenges Posed by OLDIs

<u>Two important properties:</u>

1) **Deadline bound** (e.g., 300 ms)
   - Missed deadlines affect revenue
2) **Fan-in bursts**
   - Large data, 1000s of servers
     - Tree-like structure (high fan-in)
     - Fan-in bursts → long "**tail latency**"



- Network shared with many apps (OLDI and non-OLDI)

Network must meet deadlines & handle fan-in bursts

Current Approaches

TCP: deadline agnostic, long tail latency

- Congestion → timeouts (slow), ECN (coarse)

Datacenter TCP (DCTCP) [SIGCOMM '10]

- first to *comprehensively* address tail latency
- **Finely** vary sending rate based on **extent** of congestion
- shortens tail latency, but is not deadline aware
  - ~25% missed deadlines at high fan-in & tight deadlines

DCTCP handles fan-in bursts, but is not deadline-aware

Current Approaches

Deadline Delivery Protocol (D³) [SIGCOMM '11]:

- first **deadline-aware** flow scheduling
  - Proactive & centralized
  - No per-flow state → FCFS
    - Many deadline priority inversions at fan-in bursts
- Other practical shortcomings
  - Cannot coexist with TCP, requires custom silicon

D³ is deadline-aware, but does not handle fan-in bursts well; suffers from other practical shortcomings

D²TCP's Contributions

1) **Deadline-aware** and handles **fan-in bursts**
   - *Elegant* **gamma-correction** for congestion avoidance
     - far-deadline → back off more
       near-deadline → back off less
   - Reactive, decentralized, state (end hosts)
2) Does **not** hinder long-lived (non-deadline) flows
3) **Coexists** with TCP → incrementally deployable
4) **No** change to switch hardware → deployable today

D²TCP achieves 75% and 50% fewer missed deadlines than DCTCP and D³

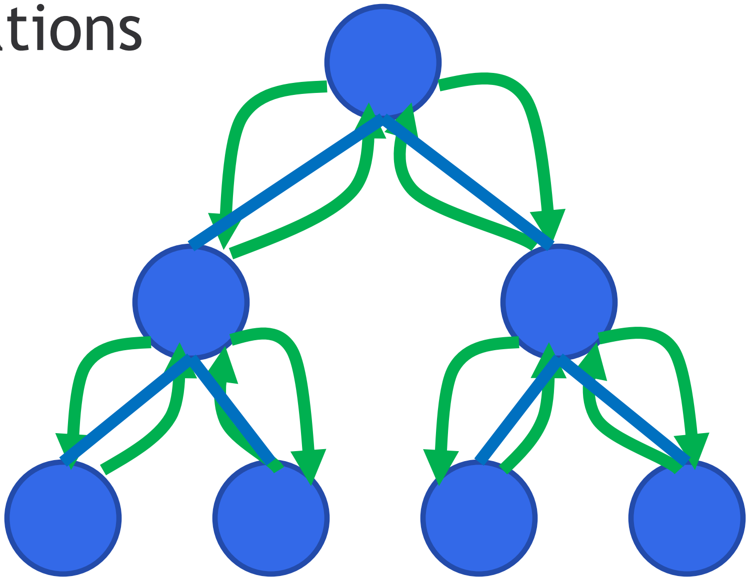
Outline

- Introduction
- OLDIs
- D^2TCP
- Results: Small Scale Real Implementation
- Results: At-Scale Simulation
- Conclusion

OLDIs

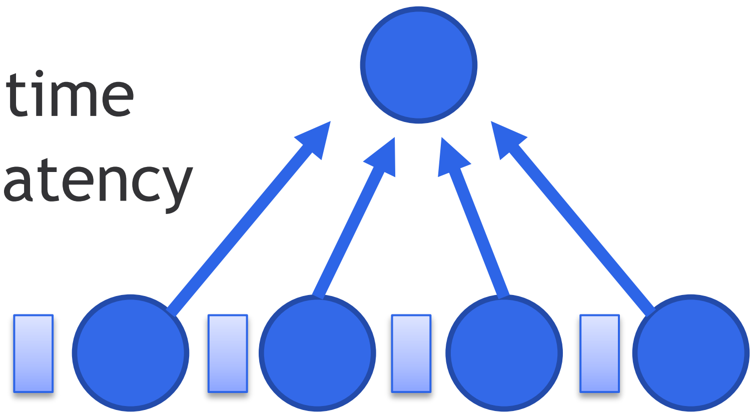
OLDI = **O**n**L**ine **D**ata **I**ntensive applications

- **Deadline bound**, handle large data
- Partition-aggregate
  - **Tree-like structure**
  - Root node sends *query*
  - Leaf nodes respond with *data*
- Deadline budget split among nodes and network
  - E.g., total = 300 ms, parents-leaf RPC = 50 ms
- Missed deadlines → incomplete responses
  → affect user experience & revenue



Long Tail Latency in OLDIs

- Large data → High Fan-in degree
- **Fan-in bursts**
  - Children respond around same time
    - Packet drops: Increase tail latency
  - Hard to absorb in buffers
  - Cause many missed deadlines



- Current solutions either
  - Over-provision the network → high cost
  - Increase network budget → less compute time

Current solutions are insufficient

Outline

- Introduction
- OLDIs
- D^2TCP
- Results: Small Scale Real Implementation
- Results: At-Scale Simulation
- Conclusion

D²TCP

Deadline-aware and handles **fan-in bursts**

Key Idea: Vary sending rate based on both deadline and extent of congestion

- Built on top of DCTCP
- Distributed: uses per-flow state at end hosts
- Reactive: senders react to congestion
  - no knowledge of other flows

D²TCP: Congestion Avoidance

A D²TCP sender varies sending window (W) based on <u>both</u> extent of congestion and deadline

$$\boxed{W := W * (1 - p / 2)}$$

<u>Note:</u> Larger p ⇒ smaller window. p = 1 ⇒ W/2. p = 0 ⇒ W/2

P is our gamma correction function

D²TCP: Gamma Correction Function

Gamma Correction (p) is a function of congestion & deadlines

$$\boxed{p = \alpha^d}$$

- α: extent of congestion, same as DCTCP's α ($0 \le \alpha \le 1$)
- d: *deadline imminence factor*
  - "completion time with window (W)" ÷ "deadline remaining"
  - d < 1 for far-deadline flows, d > 1 for near-deadline flows

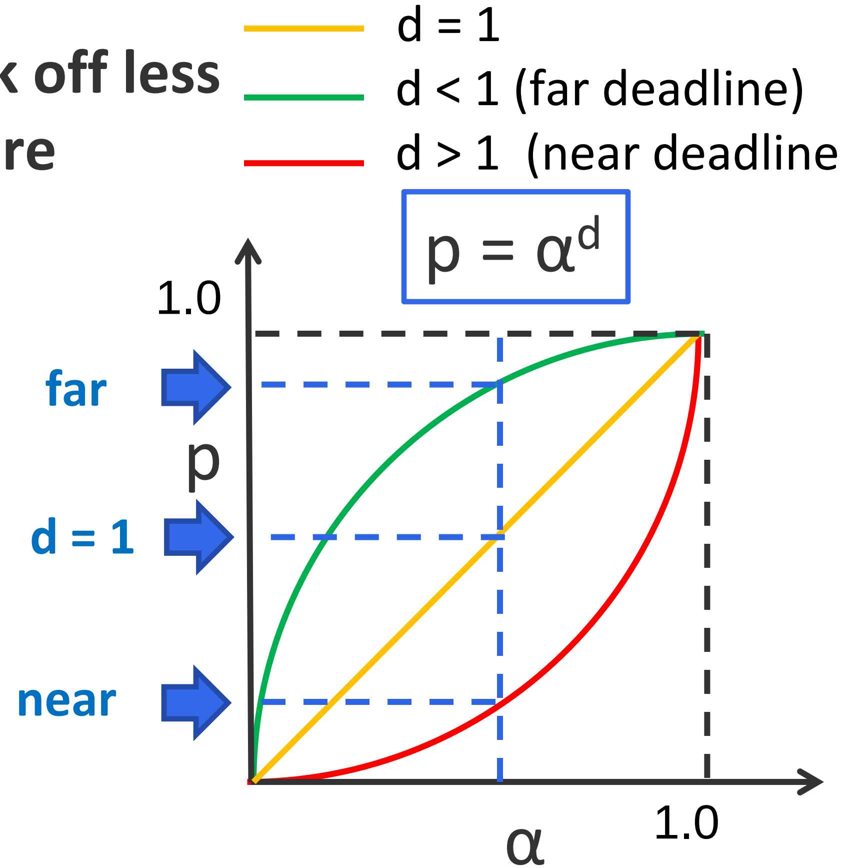
Gamma Correction Function (cont.)

Key insight: **Near-deadline flows back off less while far-deadline flows back off more**

$W := W * (1 – p / 2)$

- d < 1 for far-deadline flows
  → p large → shrink window
- d > 1 for near-deadline flows
  → p small → retain window
- Long lived flows → d = 1
  - → DCTCP behavior

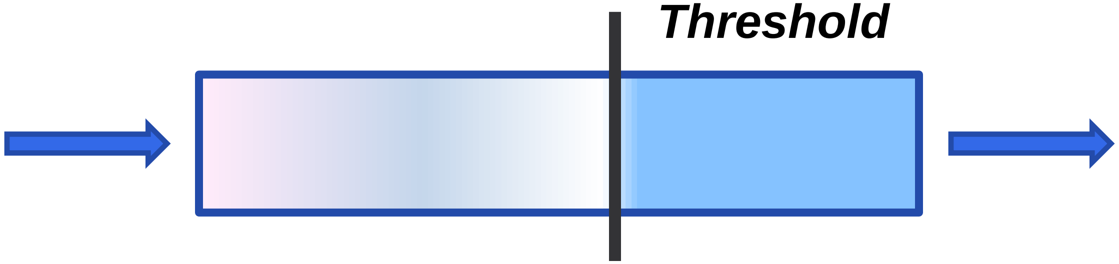
$p = \alpha^d$



Gamma correction elegantly combines congestion and deadlines

Gamma Correction Function (cont.)

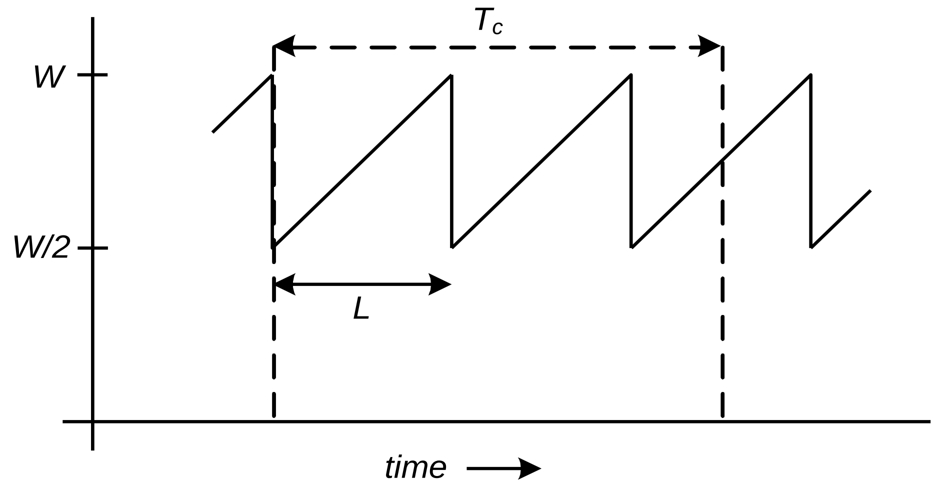
- *α* is calculated by aggregating ECN (like DCTCP)
  - Switches mark packets if **queue_length > threshold**
    - ECN enabled switches *common*



  - Sender computes the fraction of marked packets averaged over time

Gamma Correction Function (cont.)

- The deadline imminence factor (d):
  "completion time with window (W)" ÷ "deadline remaining"
  ($d = T_c / D$)
- B → Data remaining, W → Current Window Size



Avg. window size ~= 3/4 * W ⇒ T_c ~= B / (3/4 * W)

A more precise analysis in the paper!

D^2TCP: Stability and Convergence

$\boxed{W := W * (1 – p / 2)}$ $\boxed{p = \alpha^d}$

- D^2TCP's control loop is stable
  - Poor estimate of d corrected in subsequent RTTs
- When flows have tight deadlines (d >> 1)

1. d is capped at 2.0 → flows not over aggressive
2. As α (and hence p) approach 1, D^2TCP defaults to TCP

→ D^2TCP **avoids** congestive collapse

D^{2}TCP: Practicality

- Does **not** hinder background, long-lived flows
- **Coexists** with TCP
  - Incrementally deployable
- Needs **no** hardware changes
  - ECN support is commonly available

D^{2}TCP is deadline-aware, handles fan-in bursts, and is deployable today

Outline

- Introduction
- OLDIs
- D²TCP
- Results: Real Implementation
- Results: Simulation
- Conclusion

Methodology

1) Real Implementation
   - Small scale runs
2) Simulation
   - Evaluate production-like workloads
   - At-scale runs
   - Validated against real implementation

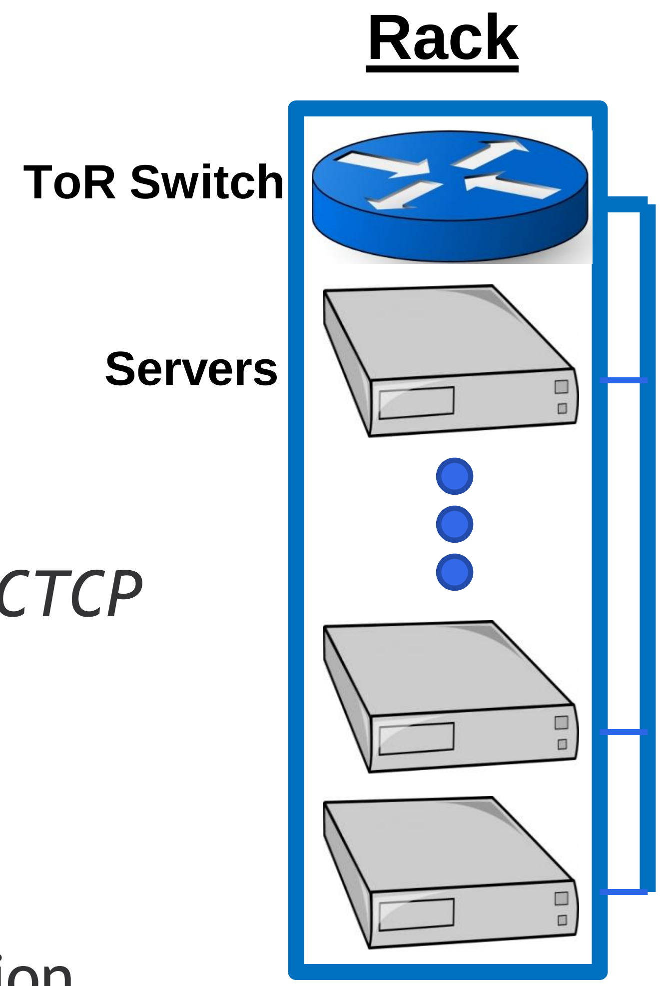
Real Implementation

- 16 machines connected to ToR
  - 24x 10Gbps ports
  - 4 MB shared packet buffer
- Publicly available DCTCP code
- D^2TCP → ~100 lines of code *over DCTCP*

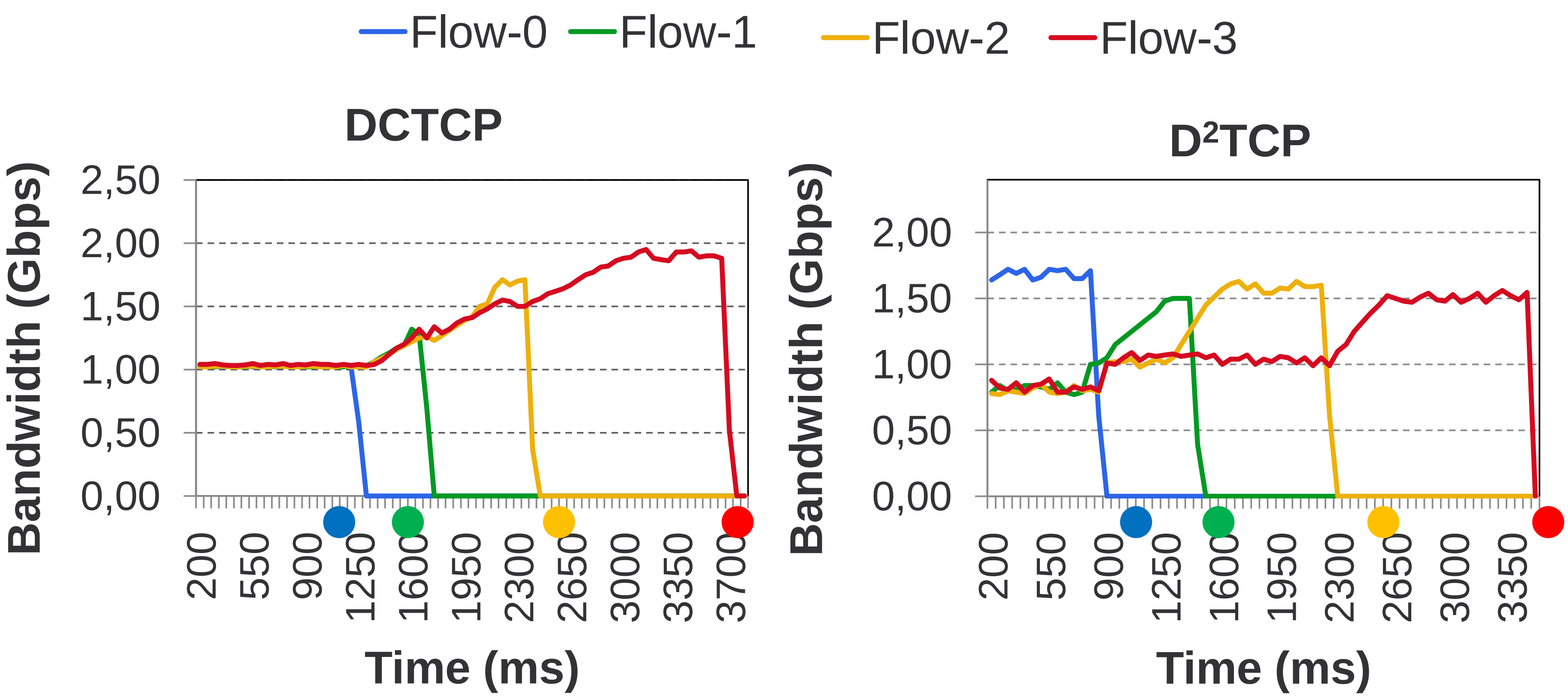
All parameters match DCTCP paper

D^3 requires custom hardware →

comparison with D^3 only in simulation

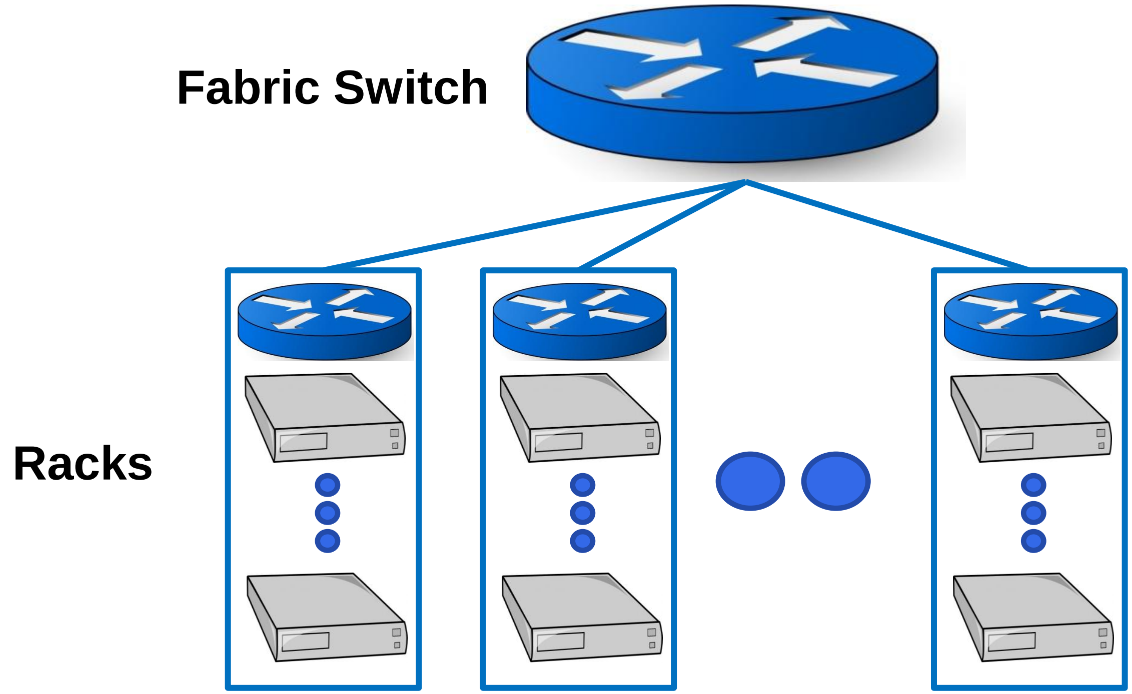


D²TCP: Deadline-aware Scheduling



- DCTCP → All flows get same b/w irrespective of deadline
- D²TCP → Near-deadline flows get more bandwidth

At-Scale Simulation



- 1000 machines
  → 25 Racks x 40 machines-per-rack
- Fabric switch is non-blocking
  → simulates *fat-tree*

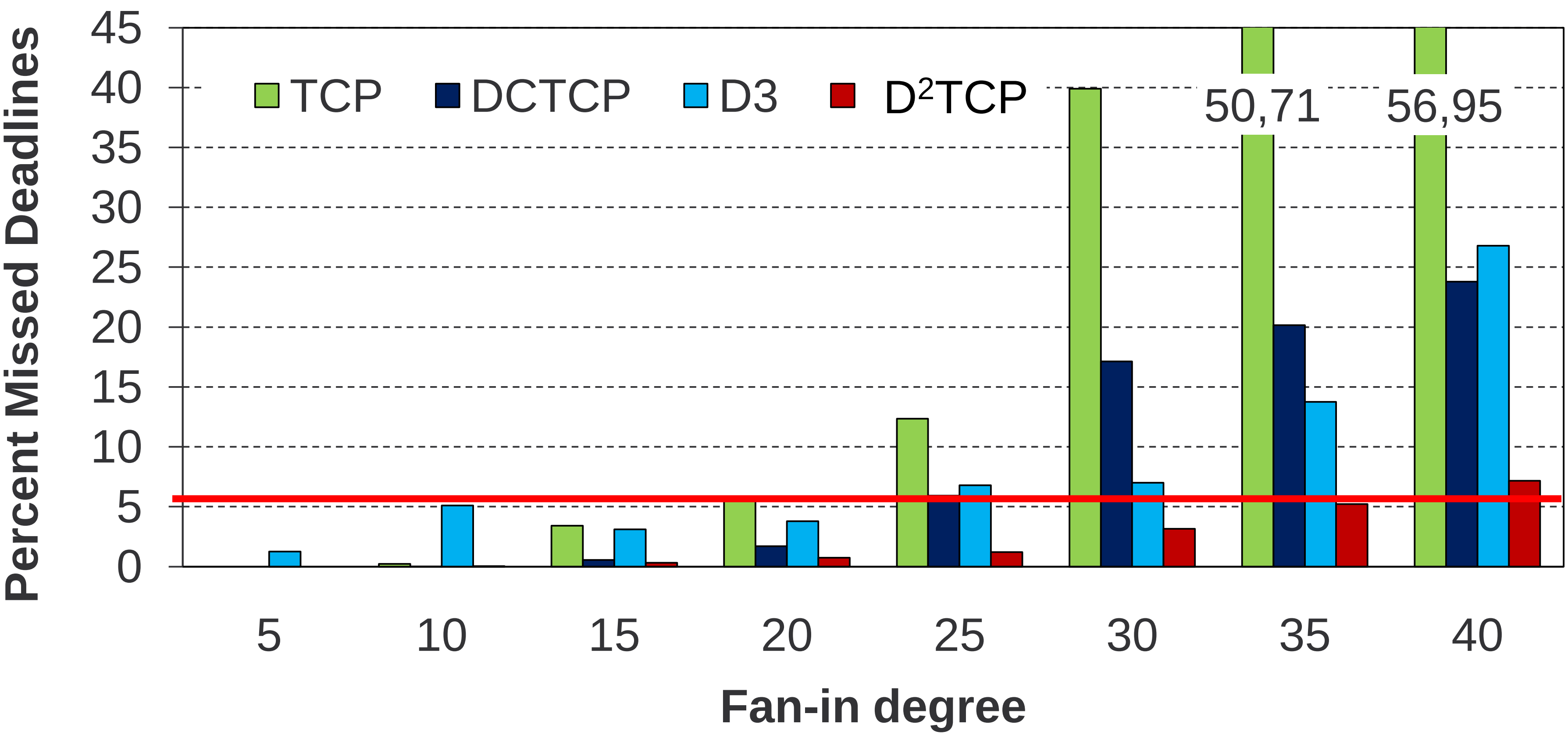
At-Scale Simulation (cont.)

- ns-3
- Calibrated to unloaded RTT of ~200 μs
  - Matches real datacenters
- DCTCP, D^3 implementation matches specs in paper

Workloads

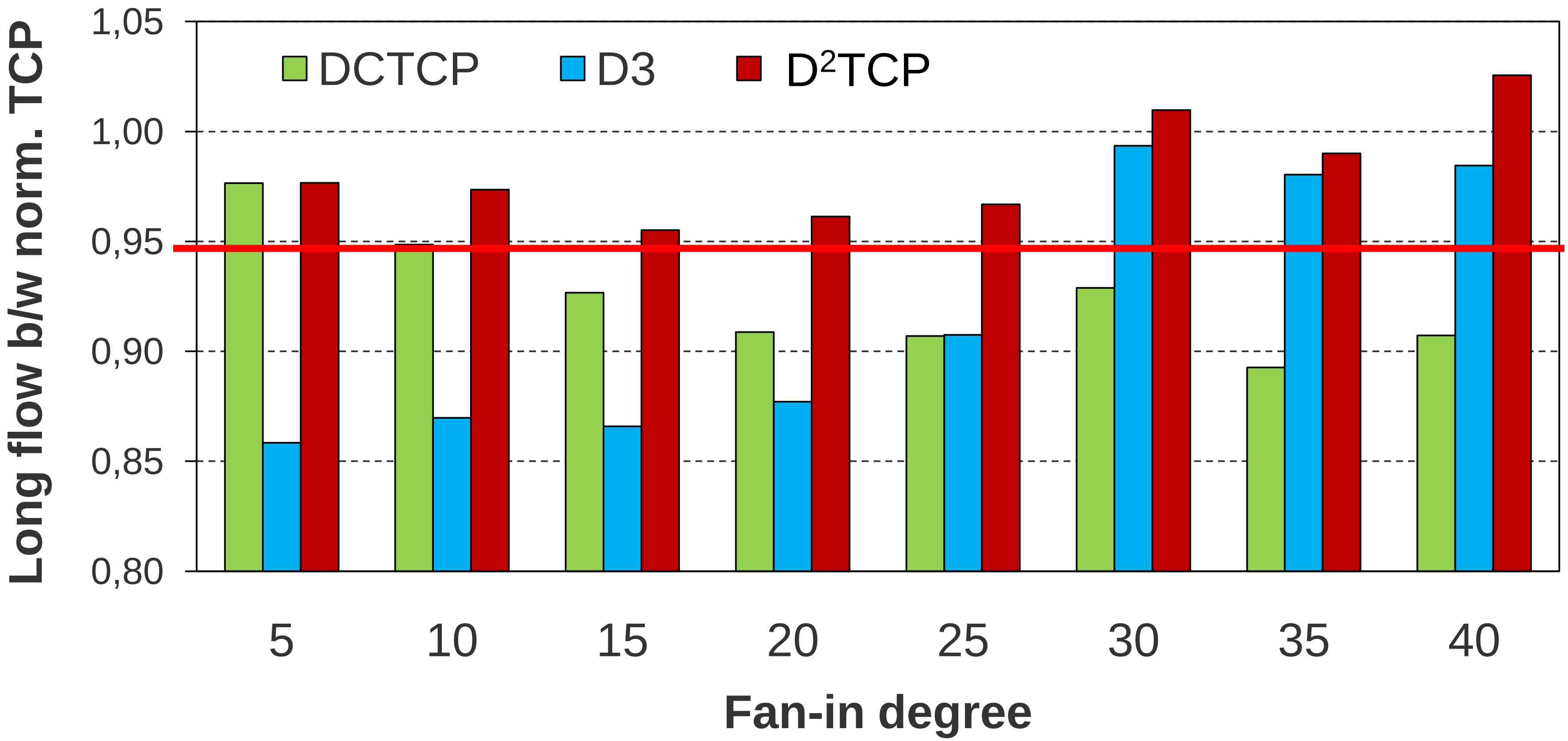
- 5 synthetic OLDI applications
- Message size distribution from DCTCP/D^3 paper
  - Message sizes: {2,6,10,14,18} KB
- Deadlines calibrated to match DCTCP/D^3 paper results
  - Deadlines: {20,30,35,40,45} ms
- Use random assignment of threads to nodes
- Long-lived flows sent to root(s)
- Network utilization at 10-20% → typical of datacenters

Missed Deadlines



- At fan-in of 40, both DCTCP and D3 miss ~25% deadlines
- At fan-in of 40, D2TCP misses ~7% deadlines

Performance of Long-lived Flows



- Long-lived flows achieve similar b/w under D^{2}TCP (within 5% of TCP)

The next two talks …

- Address similar problems
- Allow them to present their work
- Happy to take comparison questions offline

Conclusion

- D^{2}TCP is **deadline-aware** and handles **fan-in bursts**
  - 50% fewer missed deadlines than D^3
- Does **not** hinder background, long-lived flows
- **Coexists** with TCP
  - Incrementally deployable
- Needs no hardware changes

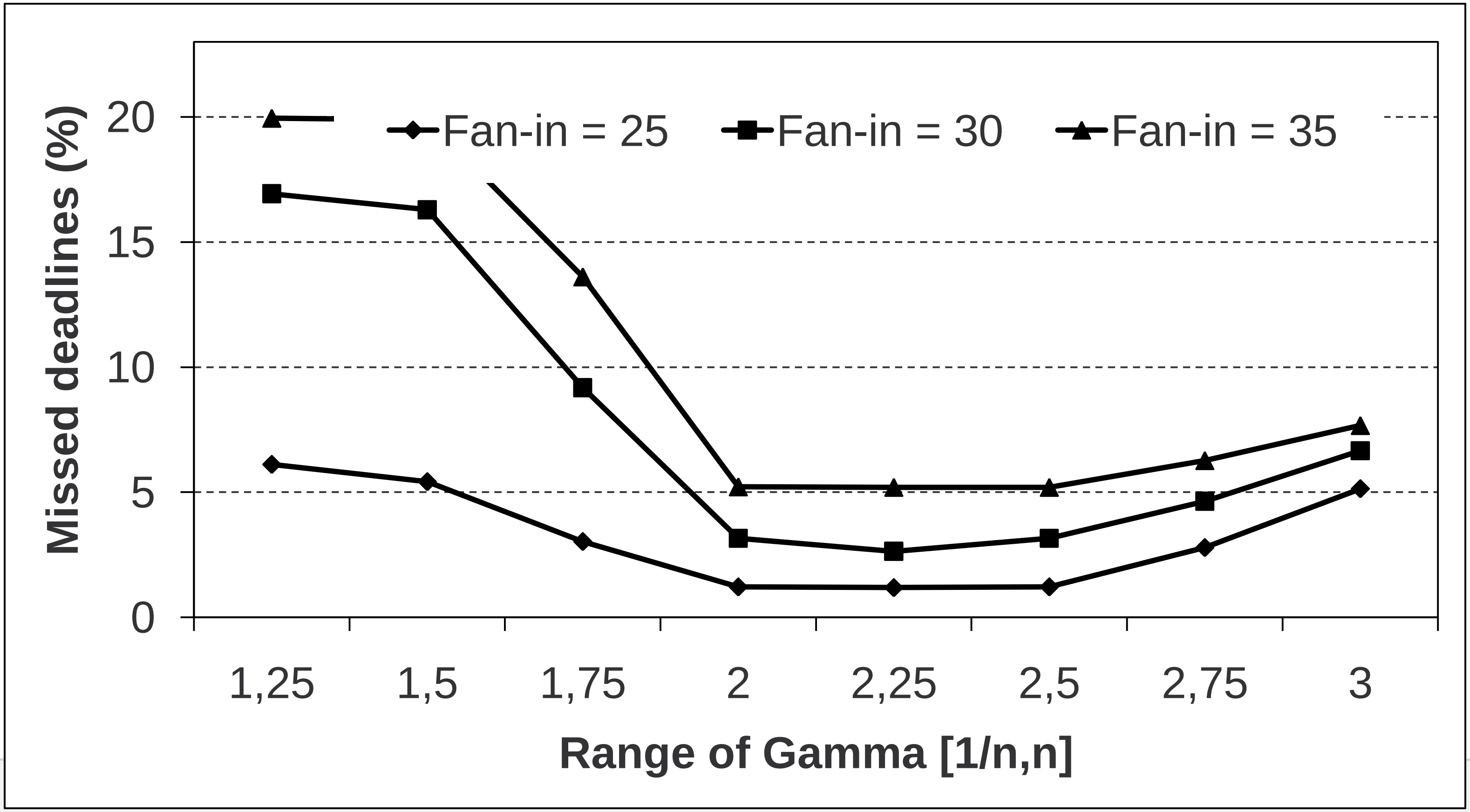
D^{2}TCP is an elegant and practical solution to the challenges posed by OLDIs

Backup Slides

- D^{2}TCP Vs PDQ
- D^{2}TCP Vs DeTail
- D^{2}TCP Vs RCP
- Priority Inversions
- Pri. Inv. in next RTTs
- Gamma cap
- Without gamma cap
- Real Vs. Sim
- "d" computation
- TCP quirks like LSO
- RTO_{Min} = 10 ms
- Coexistence with TCP
- Pri. Inv. possible with Qos?
- Deadline distribution
- Tighter deadlines
- Mean , Variance

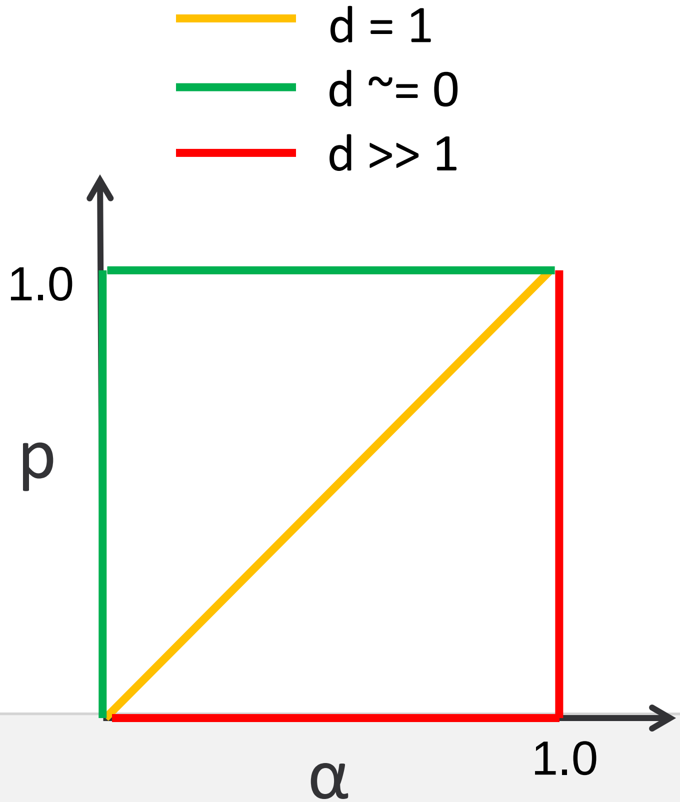
How did you choose a gamma cap of 2.0?

sweet spot across many OLDI apps & fan-in degrees



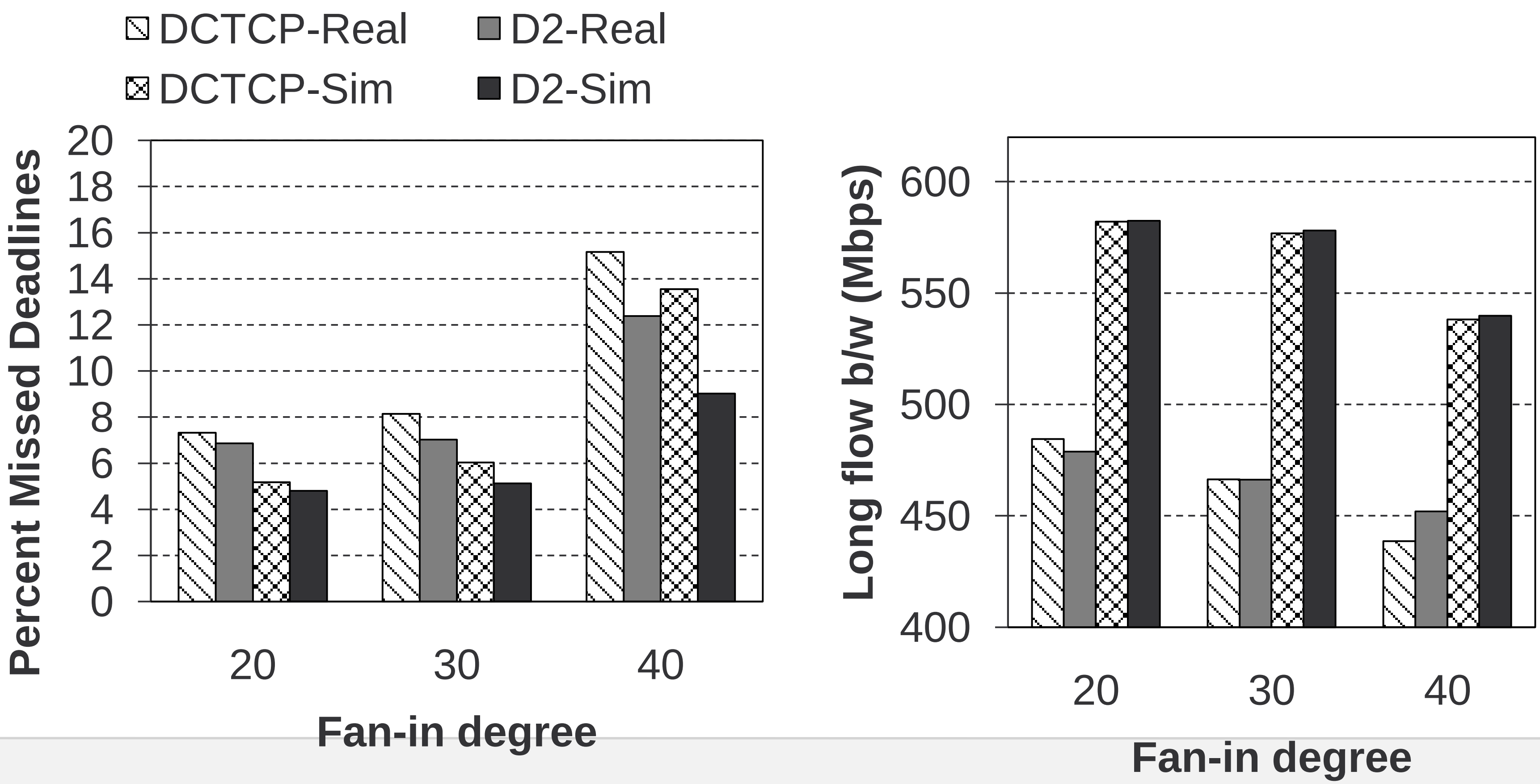
Why do you need a cap on “d”?

When d >> 1 or when d ~= 0, gamma function *no longer* reacts to the extent of congestion. It adversely (coarsely) reacts to *mere presence/absence* of congestion



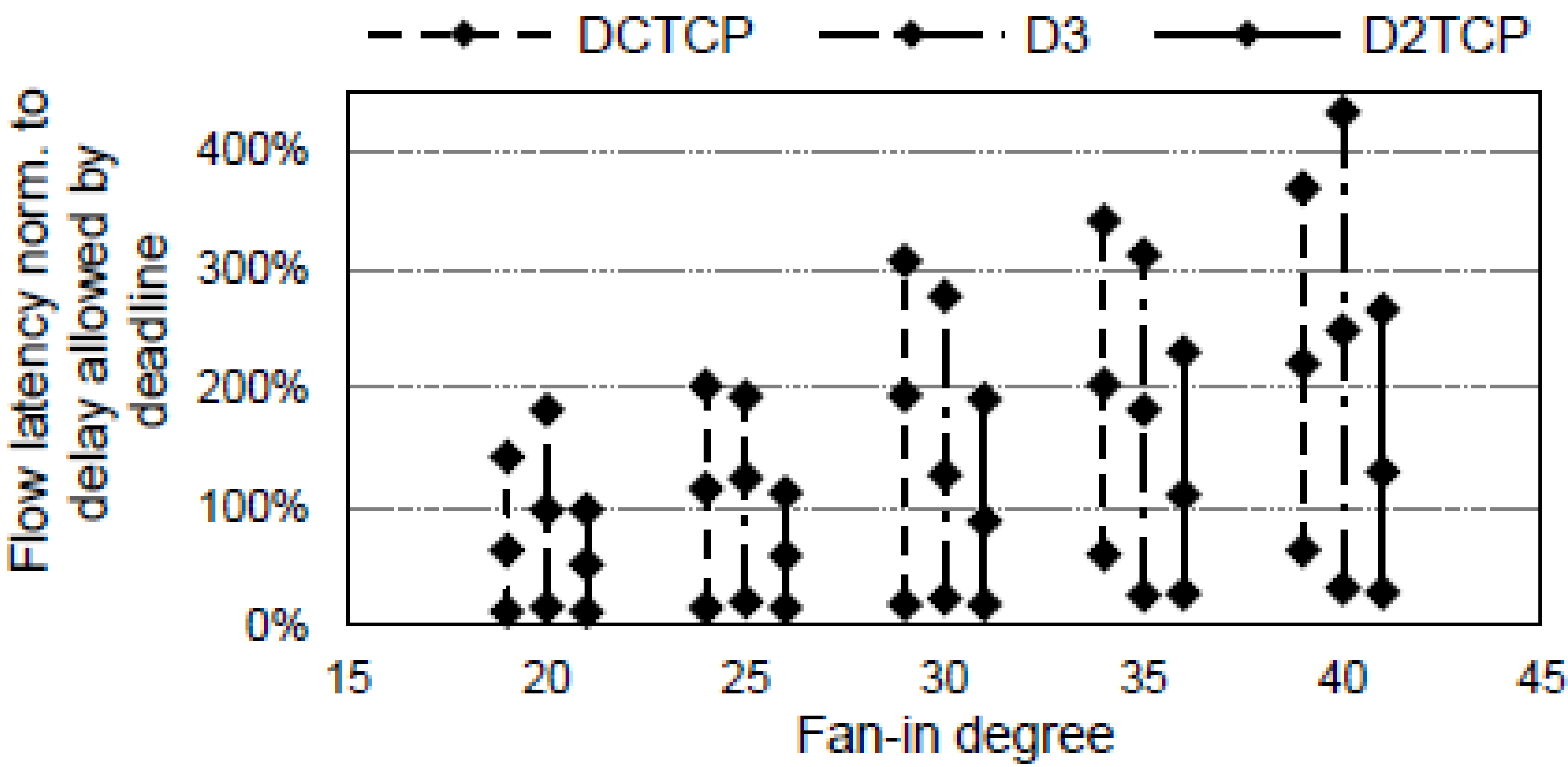
Does your simulation results match with real implementation?

Simulation trends <u>match</u> our real implementation trends



Does D^2TCP target the mean or variance of latency distribution?

D^2TCP reduces *both mean and variance* of latency distribution



How are your deadlines distributed?

We take base deadlines as {20, 30, 35, 40, 45} ms
We evaluate three distributions

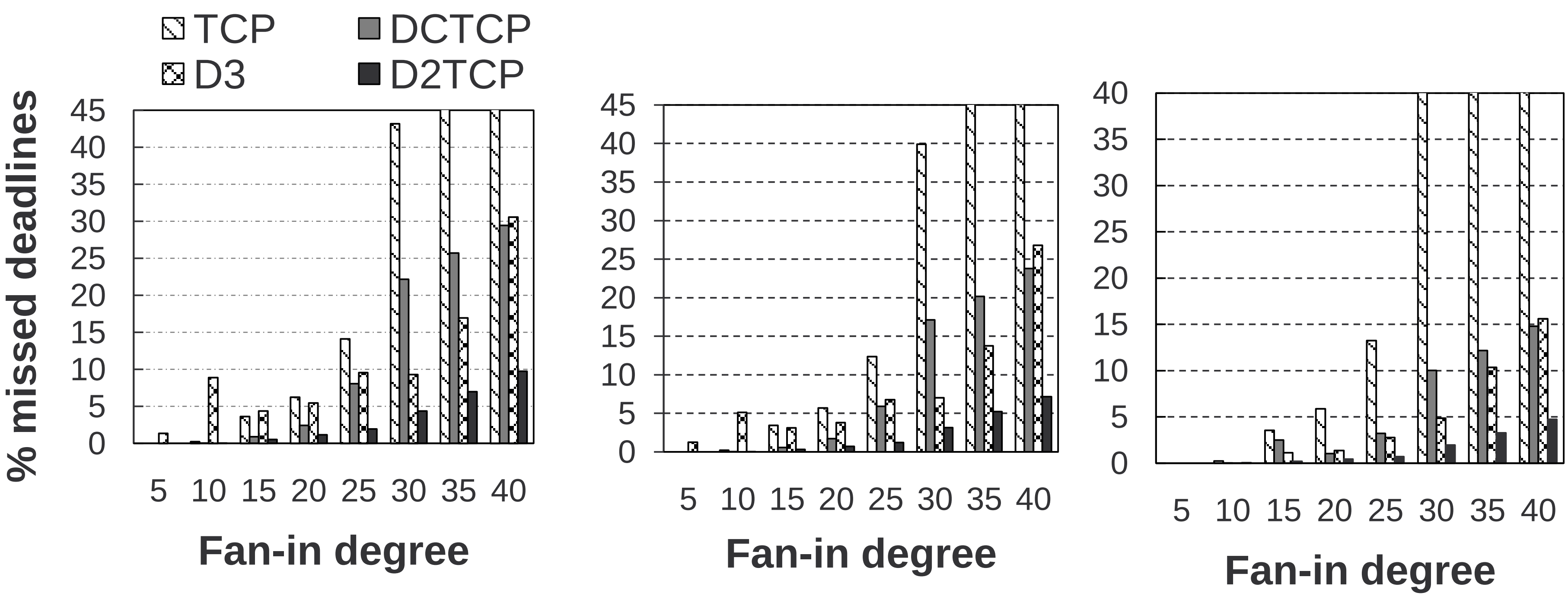
- Low Variance: +10% uniform random variation
- Medium Variance: +50% uniform random variation
- High Variance: One-sided exp. distribution

D^3 paper models only "high variance" deadlines, and our results match results from D3 paper.

D^2TCP performs well across all the three distributions.

Results across Distributions

Trends similar across distributions. *D²TCP performs well across all three distributions.*



How many times does D^3 inverts priority?

Priority Inversion: No of times an earlier deadline request was denied while a later deadline request was accepted.

Fan-in Degree	Low-Variance	Med-Variance	Hi-Variance
20	31.9	26.3	24.1
25	33.2	28.7	24.6
30	35.7	30.8	28.6
35	41.9	33.4	31.5
40	48.6	40.5	33.1

Why does D[3]'s priority inversion not get fixed in subsequent RTTs?

1. The priority inversion will get fixed when demand < capacity.
2. But when demand > capacity (during fan-in bursts with close deadlines), remembering total demand won't prevent race condition (priority inversion) in subsequent RTTs. To fix this, the switch needs *per-flow* state. Any *aggregated state* seems messy and hard.

How well does D2TCP coexist with TCP?

We run 5 OLDIs and long flows

- All TCP – All 5 OLDIs, long flows use TCP
- Mix #1 – 3 OLDIs, long flows use TCP. 2 OLDIs use D^2TCP
- Mix #2 - 3 OLDIs use TCP. 2 OLDIs, long flows use D^2TCP

Table 2: Long-flow b/w when D^2TCP & TCP coexist

Fan-in degree	Long flow bandwidth (Mbps)		
	All TCP	Mix #1	Mix #2
15	90	90	90
20	86	86	86

(1) Moving *some OLDIs to D^2TCP* does not affect long flow b/w
(2) Moving *long flows to D^2TCP* does not affect long flow b/w
(3) We show OLDIs that use TCP do not miss more deadlines when ***some other*** OLDIs move to D^2TCP – in the paper!

Does D^2TCP handle tighter deadlines?

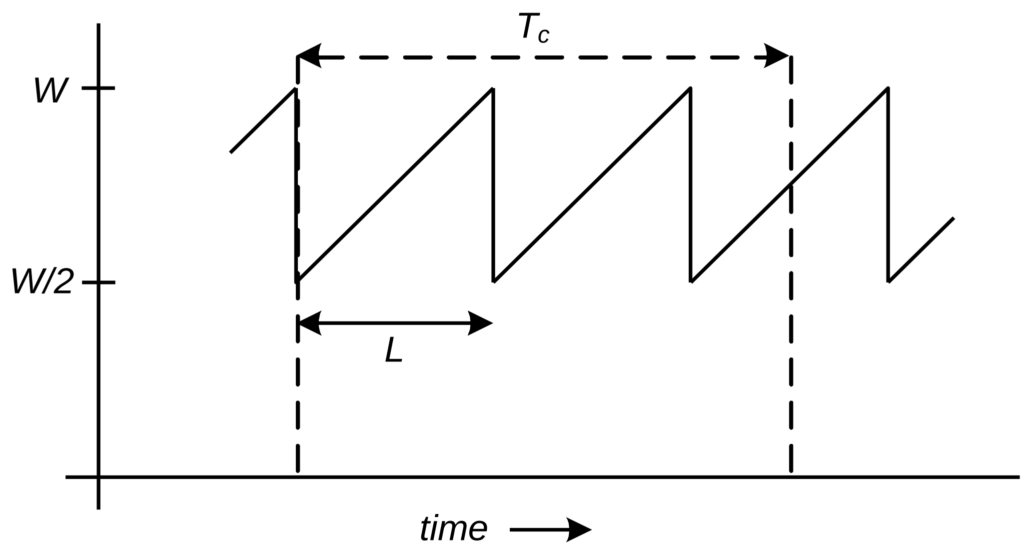
D^2TCP can meet 35-55% tighter deadlines than D^3 while maintaining the similar % missed deadlines

Table 3: Deadlines achieved by D^3 and D^2TCP for similar fraction of missed deadlines

Fan-in degree	D3's missed deadlines (%)	D^2TCP's missed deadlines (%)	D^2TCP's tighter deadline (%)
10	0.71	0.84	55
15	3.61	3.49	45
20	4.7	4.88	35

How is deadline imminence calculated?

- d: deadline imminence factor
  = "completion time with window (W)" ÷ "deadline remaining" : **d = T_c / D**



Avg. window size = 3/4 * W ⇒ T_c ~= B / (3/4 * W)

A more precise analysis in the paper!

How does D^{2}TCP compare with PDQ?

Idea: Fix D^3 priority inversion by preempting lower priority flows (adds per-flow state)

Contrast with D^{2}TCP:

- Quantitative comparison not available
- Inherits D^3's practical issues
    1. Requires custom hardware (silicon)
    2. Requires per-flow state. State may not scale in future when many OLDI flows congest.
    3. Coexistence with TCP possible, but requires static bandwidth partitioning between PDQ and non-PDQ flows → unused (wasted) bandwidth!

Real D^{2}TCP implementation exists today running on TCP cluster

How does D²TCP compare with DeTail?

Idea: *Identify congestion (link layer), find alternate routing paths (network layer), and support reordered packets (transport layer)*

Contrast with D²TCP:

1. **Fan-in Congestion** : *Fan-in Congestion cannot be handled by using path diversity* - the bottleneck is the output port of the ToR switch that connects to the *root node* (no alternate paths).
2. **Priority Levels**: DeTail is limited by the number of priority (8-16) levels that can supported in hardware (PFC). But it is well known [D³ paper] that deadline diversity is high ⇒ needs many more priority levels.

TCP quirks like LSO are absent in sims. How do you capture that?

1. Yes TCP quirks are absent in our simulations but we tuned our workloads to match DCTCP's & D^3's absolute performance (not only traffic) under D^3's real implementation. So, our simulated D^2TCP numbers are likely to be realistic.
2. Our real implementation results corroborate well with our simulation results. (see real vs sim.)

How does your results change with RTO_{Min} of 10 ms?

1. Retransmits are rare except in TCP, so 10ms (faster retransmits) will improve TCP but not DCTCP, D^3, or D^2TCP.
2. Google's production TCP uses something close to 20ms within the clusters, therefore we decided that our original choice of 20 ms was more appropriate.

Can D²TCP and QoS counter interact and cause priority inversion?

Today

- Each class gets its own queue in the packet buffer
- ECN marking separate for each queue (separate α)

D²TCP would schedule flows based on deadline, independent of other queues

Across different queues, the switch hardware provides guarantee for bandwidth and isolation.

D2TCP operates independently within each class, and reduce % missed deadlines within each class.

How does D^2TCP compare with RCP?

- RCP has similarities with D^3
  - Replace TCP slow start with immediate allocation
  - Optimize completion time
- Custom switch silicon needed
  - hardware grants bandwidth equal to fair share
- RCP is deadline-agnostic
  - D^3 outperforms RCP
  - D^2TCP outperforms D^3