

On Controller Performance in Software-Defined Networks

Amin Tootoonchian

Sergey Gurbanov

Yashar Ganjali



Martin Casado

Rob Sherwood

Latency

Throughput

Control

Software-Defined Networks

Decouple control plane from forwarding elements.

Control apps run on top of controller(s).

- e.g., routing on NOX.

Apps perform & scale differently.

Controller Performance

How fast controller delivers:

- requests to app.
- responses to datapath.

How many requests it handles:

- assuming negligible per-request computation.

Controller overhead regardless of control logic.

Background

Why Controller Performance Matters?

Reactive flow-based proposals have tiny per-event computation.

- e.g., Ethane, NOX forwarding.
- Controller is in the way...

Poor performance motivated:

- DIFANE: proactively push state.
- DevoFlow: reduce ctrl load.

Ideally
 $\text{overhead}(\text{req. handling}) \ll \text{compute}(\text{app})$

Software-Defined Networks

Decouple control plane from forwarding elements.

Control apps run on top of controller(s).

- e.g., routing on NOX.

Apps perform & scale differently.

Controller Performance

How fast controller delivers:

- requests to app.
- responses to datapath.

How many requests it handles.

- assuming negligible per-request computation.

**Controller overhead
regardless of control logic.**

Why Controller Performance Matters?

Reactive flow-based proposals have tiny per-event computation.

- e.g., Ethane, NOX forwarding.
- Controller is in the way...

Poor performance motivated:

- DIFANE: proactively push state.
- DevoFlow: reduce ctrl load.

Ideally

$\text{overhead}(\text{req. handling}) \ll \text{compute}(\text{app})$

Software-Defined Networks

Decouple control plane from forwarding elements.

Control apps run on top of controller(s).

- e.g., routing on NOX.

Apps perform & scale differently.

Controller Performance

How fast controller delivers:

- requests to app.
- responses to datapath.

How many requests it handles:

- assuming negligible per-request computation.

Controller overhead regardless of control logic.

Background

Why Controller Performance Matters?

Reactive flow-based proposals have tiny per-event computation.

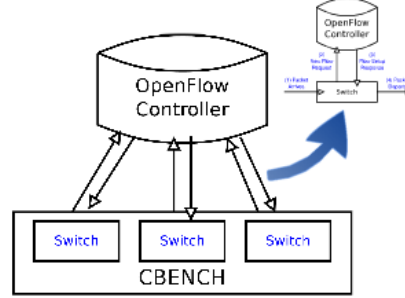
- e.g., Ethane, NOX forwarding.
- Controller is in the way...

Poor performance motivated:

- DIFANE: proactively push state.
- DevoFlow: reduce ctrl load.

Ideally
 $\text{overhead}(\text{req. handling}) \ll \text{compute}(\text{app})$

Cbench: Flow Setup Benchmark



Cbench

Modes of Operation

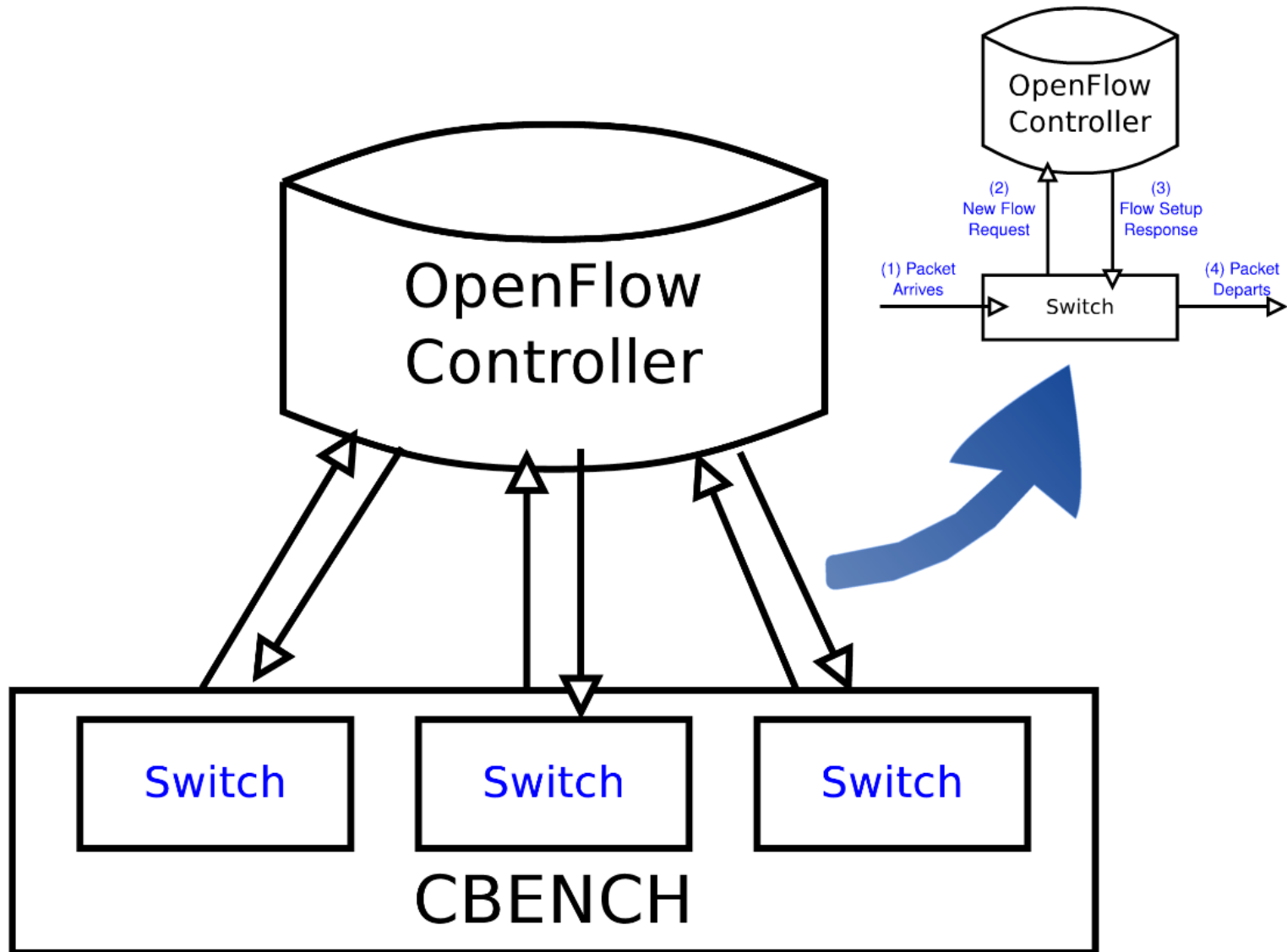
- Delay:**
 - Minimum response time
 - One packet on the fly
- Throughput:**
 - Maximum throughput
 - No limit on packets on the fly
- Hybrid:**
 - Tunable # of packets on the fly

Throughput and delay are related
(Little's law)
 $throughput = \lambda \times delay$
We use it to verify our results.

Benchmark Setup

- Machines:**
 - 8x2GHz CPU
 - 4GB DDR2 RAM
- Tests:**
 - Packet size: 82 bytes
 - Control bandwidth: 2Gbps
- Controllers:**
 - NOX, NOX-MT, Beacon, Maestro

Cbench: Flow Setup Benchmark



Cbench

Modes of Operation

Delay:

- Minimum response time
- One packet on the fly

Throughput:

- Maximum throughput
- No limit on packets on the fly

Hybrid:

- Tunable # of packets on the fly

Throughput and delay are related
(Little's law)

$$\text{avg}(\text{delay}) = \text{avg}(\text{onfly_req}) / \text{avg}(\text{xput})$$

We use it to verify our results.



Benchmark Setup

Machines:

- 8x2GHz CPU
- 4GB DDR2 RAM

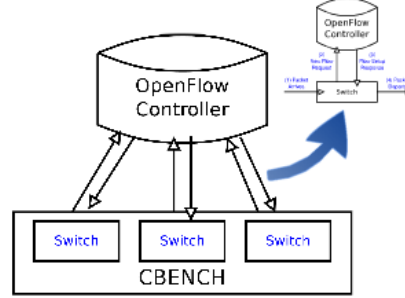
Tests:

- Packet size: 82 bytes
- Control bandwidth: 2Gbps
- 32 switches, 4 threads

Controllers:

- NOX, NOX-MT, Beacon, Maestro

Cbench: Flow Setup Benchmark



Cbench

Modes of Operation

- Delay:**
 - Minimum response time
 - One packet on the fly
- Throughput:**
 - Maximum throughput
 - No limit on packets on the fly
- Hybrid:**
 - Tunable # of packets on the fly

Benchmark Setup

- Machines:**
 - 8x2GHz CPU
 - 4GB DDR2 RAM
- Tests:**
 - Packet size: 82 bytes
 - Control bandwidth: 2Gbps
- Controllers:**
 - NOX, NOX-MT, Beacon, Maestro

Throughput and delay are related
(Little's law)
We use it to verify our results.

NOX-MT

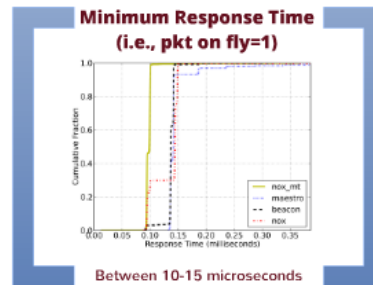
NOX performs poorly.

- 30k rps with 10ms delay.

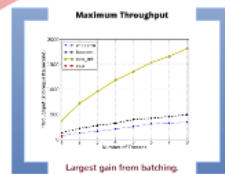
NOX-MT: attempt to fix NOX.

- Batch I/O handling.
- Multi-threaded.
- ... leaves many issues.

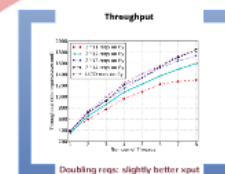
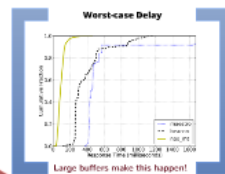
NOX-MT is far from perfect!



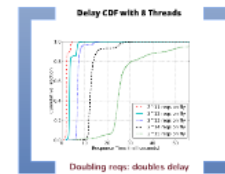
Controller Throughput & Response Time



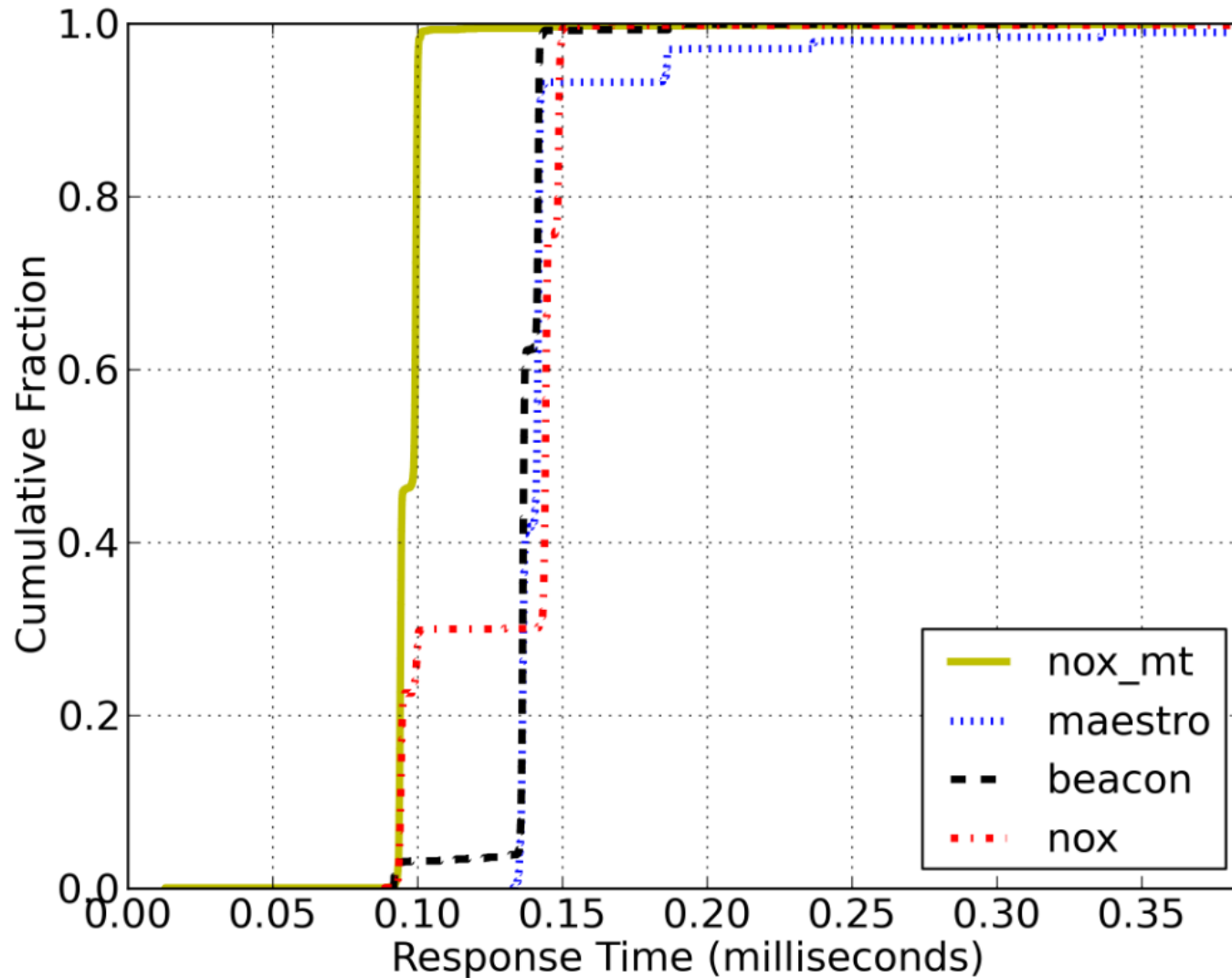
Best Xput & Worst Delay For Various Ctrlrs (i.e., unlimited outstanding requests)



NOX-MT's Performance Under Different Load Levels

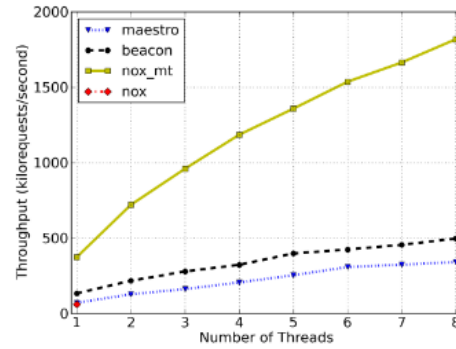


Minimum Response Time (i.e., pkt on fly=1)



Between 10-15 microseconds

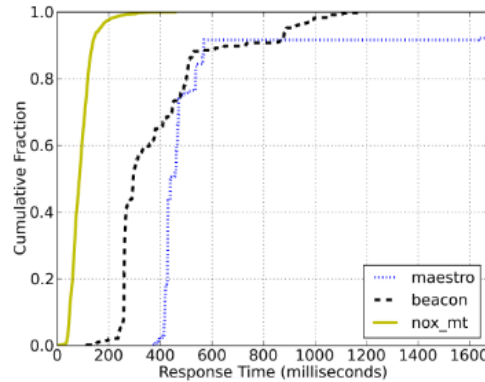
Maximum Throughput



Largest gain from batching.

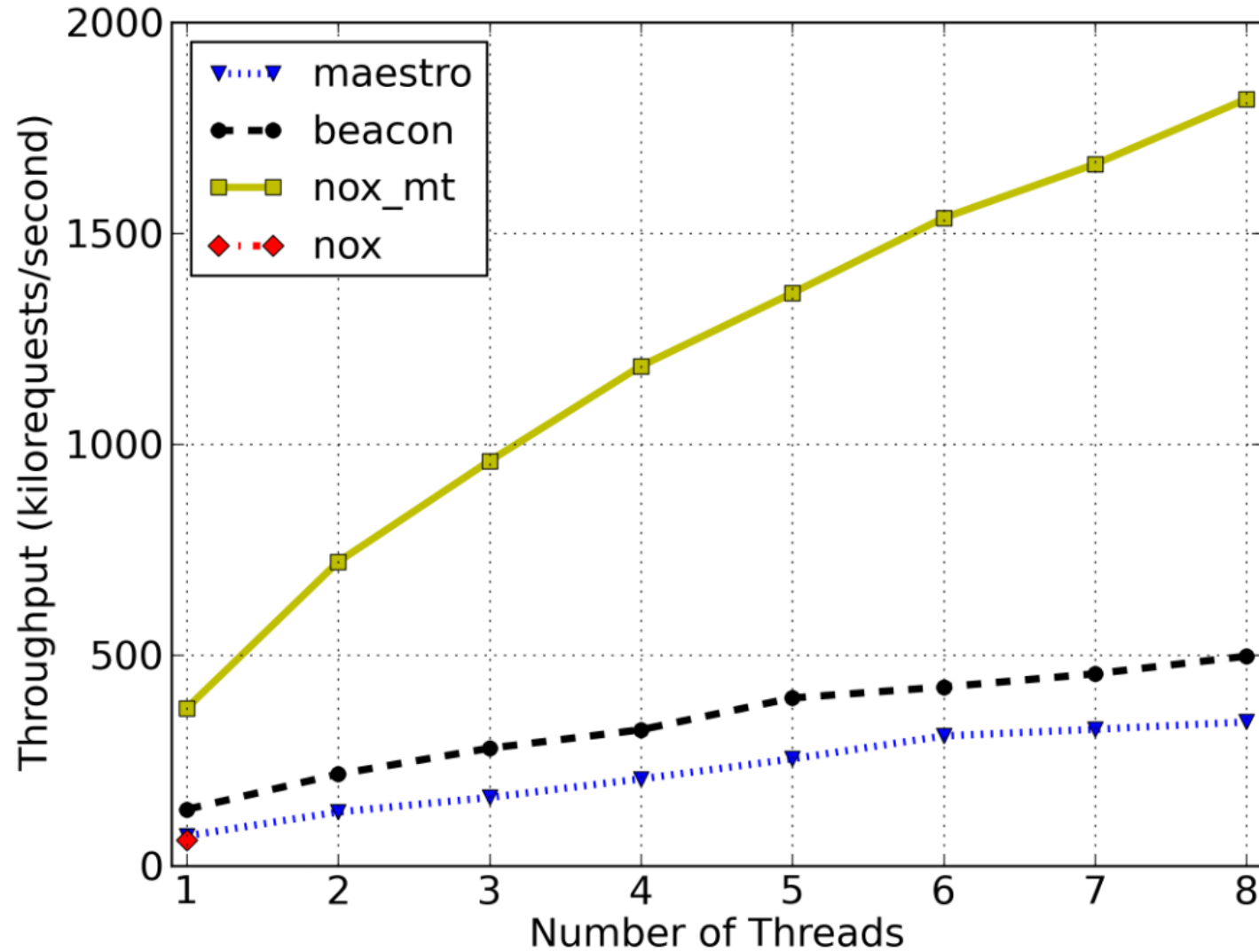
Best Xput & Worst Delay For Various Ctrlrs (i.e., unlimited outstanding requests)

Worst-case Delay



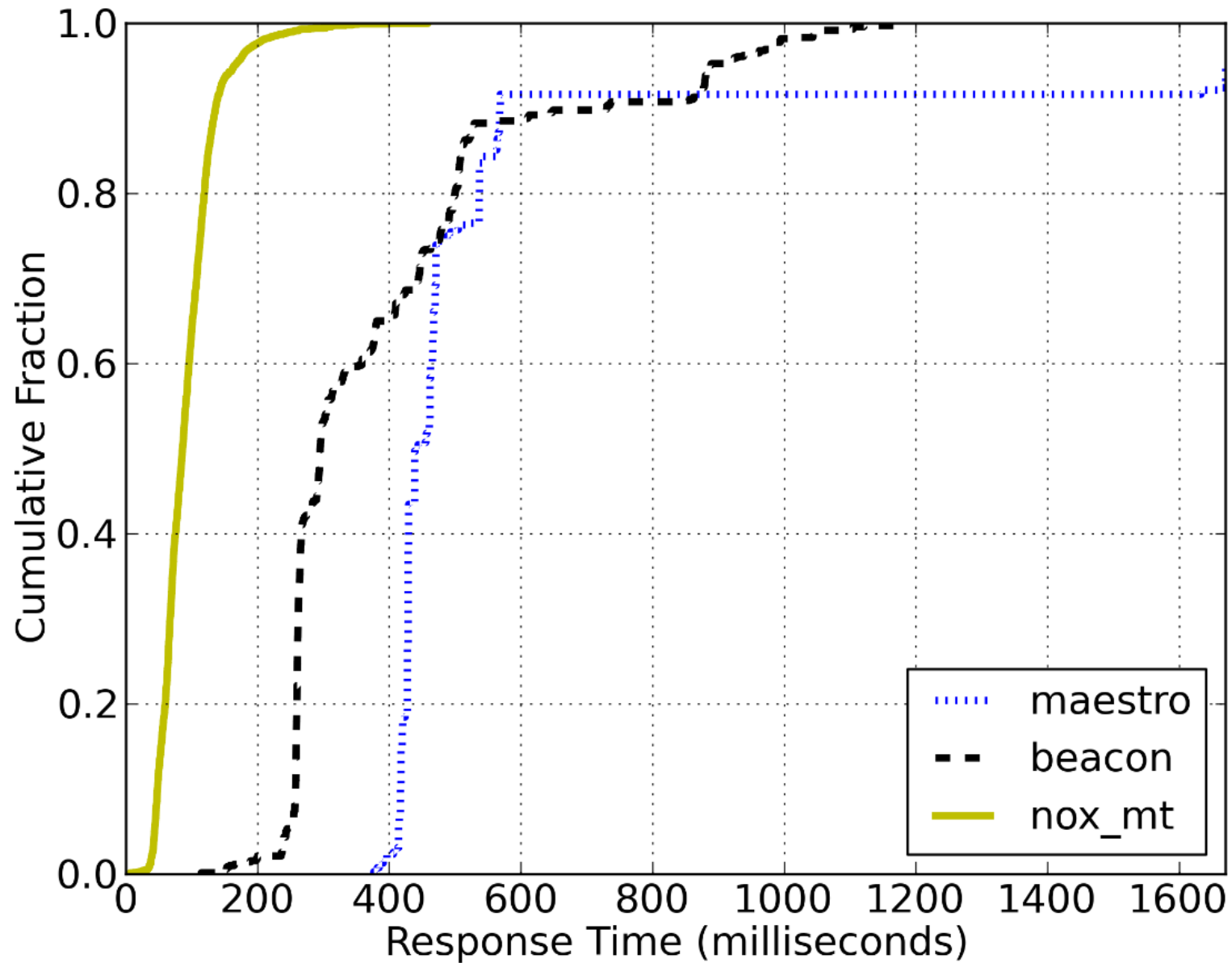
Large buffers make this happen!

Maximum Throughput



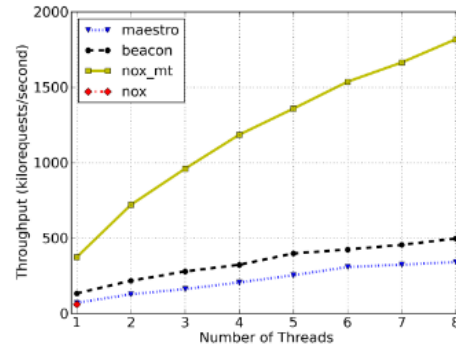
Largest gain from batching.

Worst-case Delay



Large buffers make this happen!

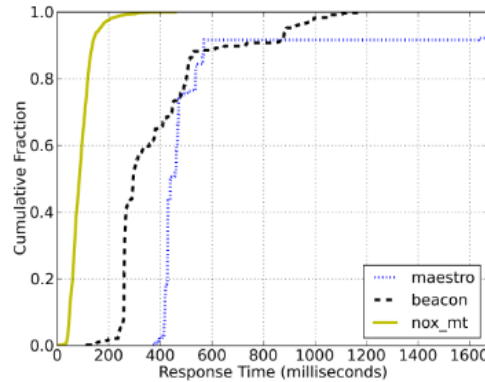
Maximum Throughput



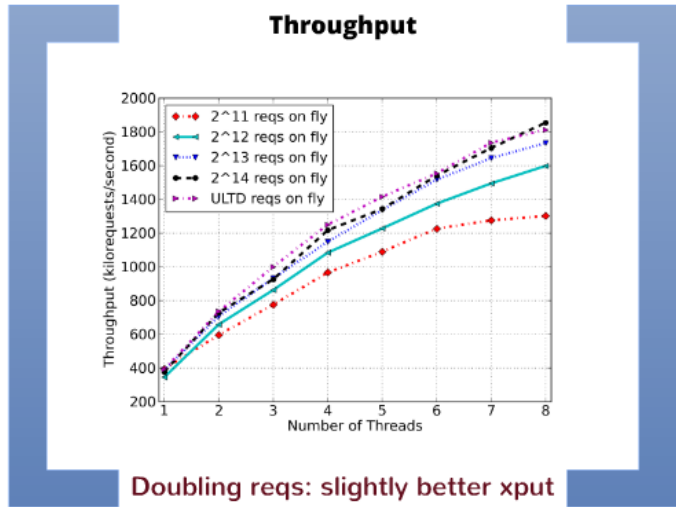
Largest gain from batching.

Best Xput & Worst Delay For Various Ctrlrs (i.e., unlimited outstanding requests)

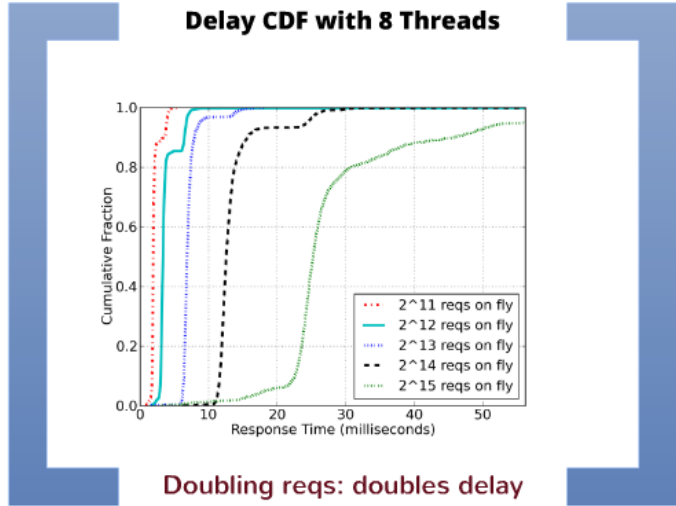
Worst-case Delay



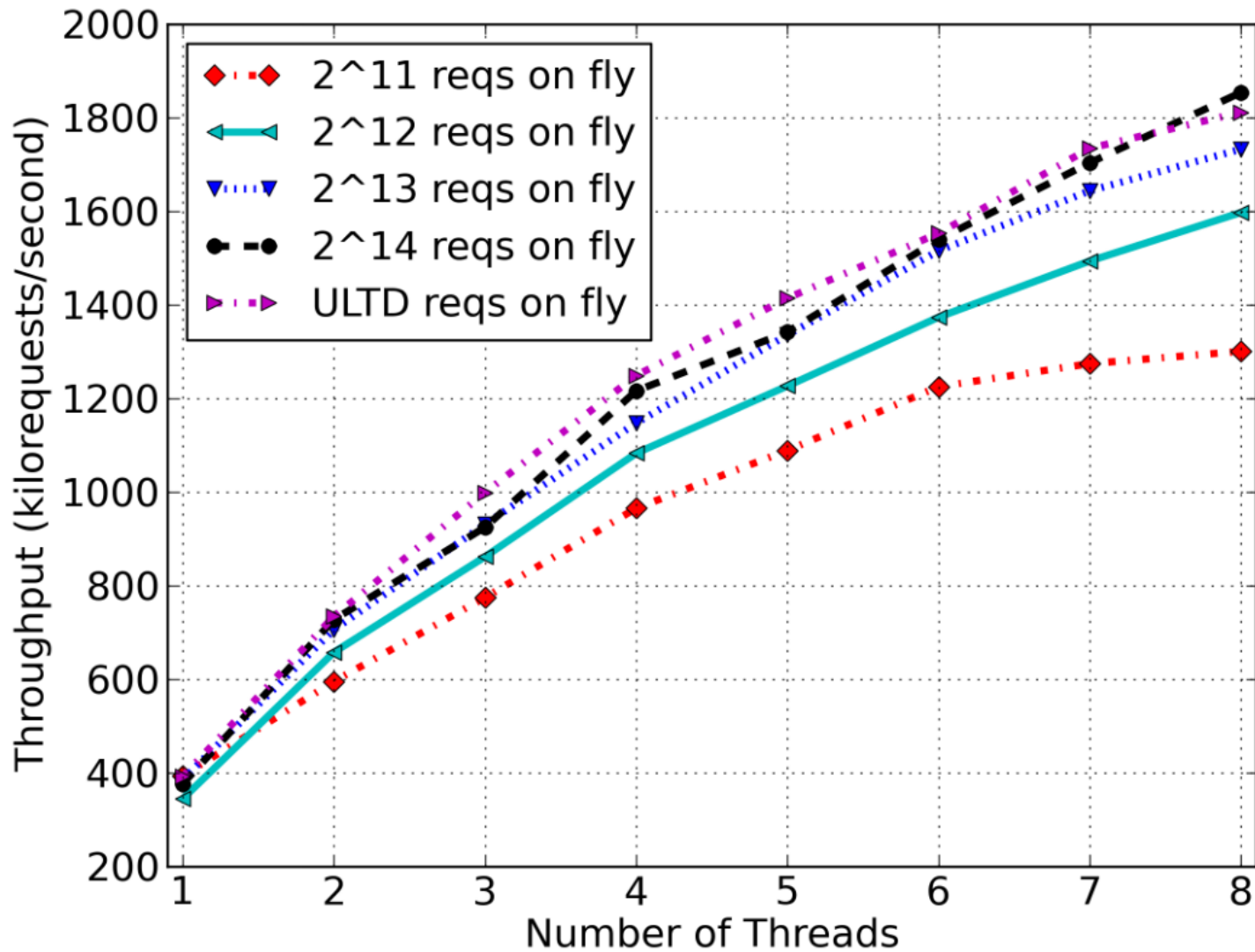
Large buffers make this happen!



NOX-MT's Performance Under Different Load Levels

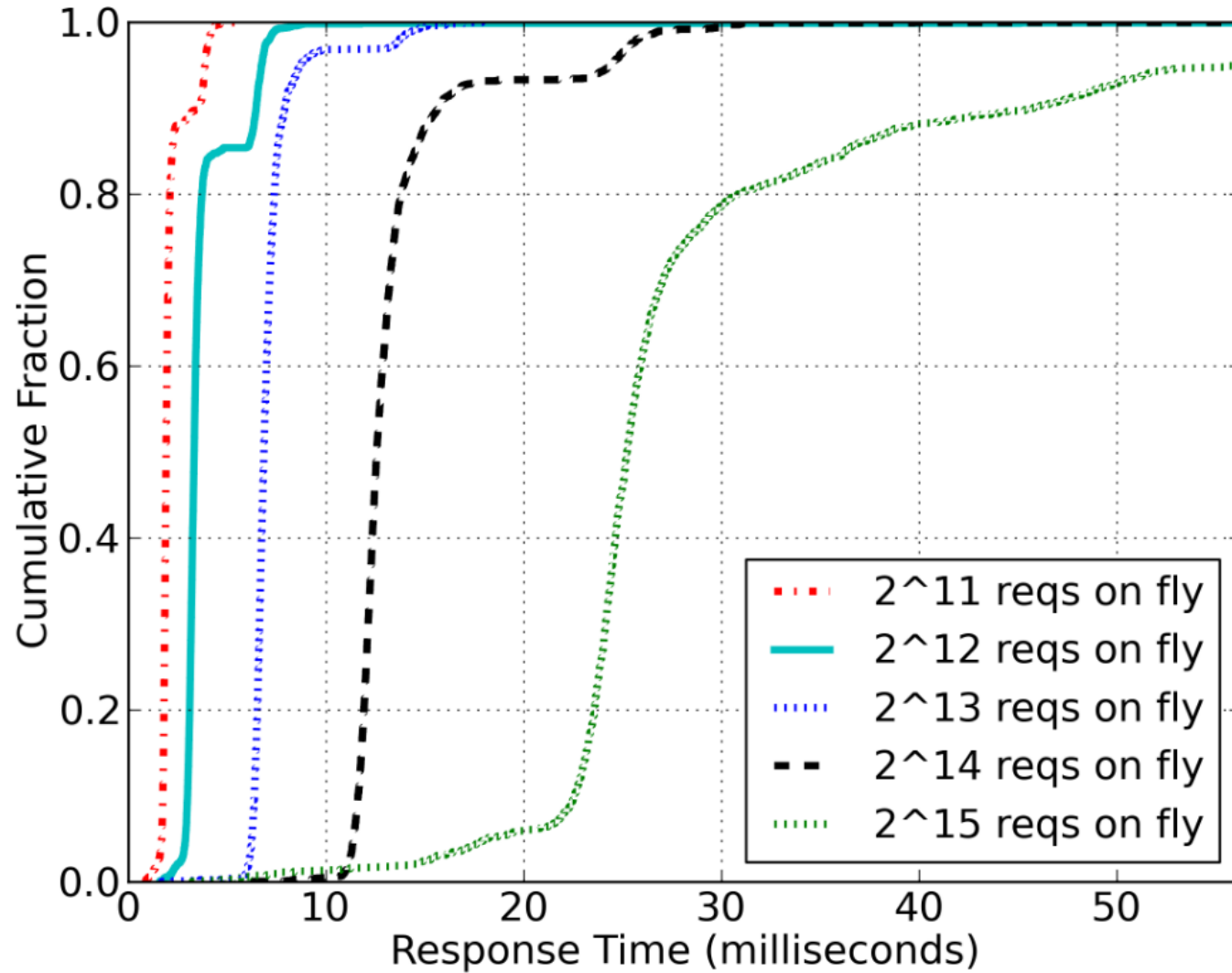


Throughput

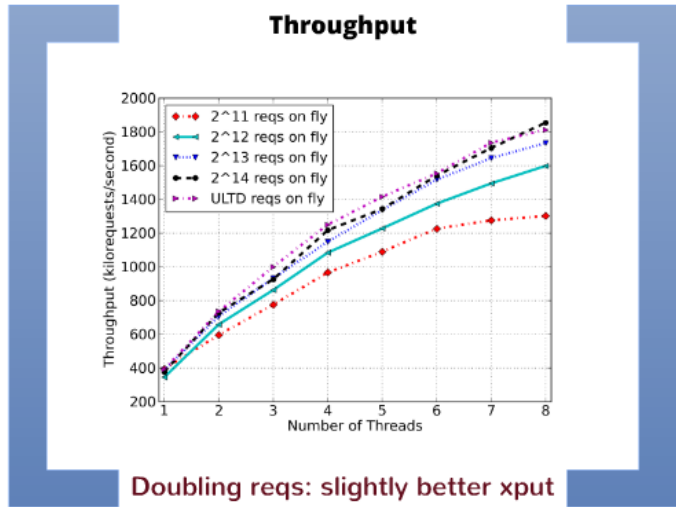


Doubling reqs: slightly better xput

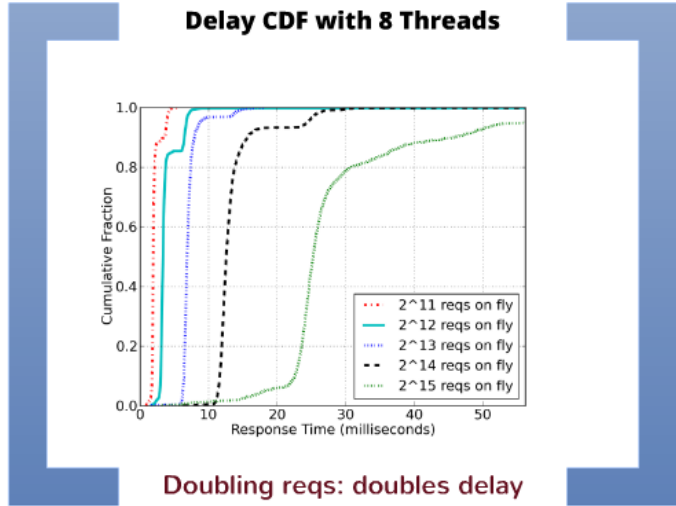
Delay CDF with 8 Threads

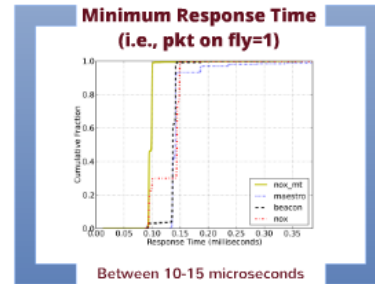


Doubling reqs: doubles delay

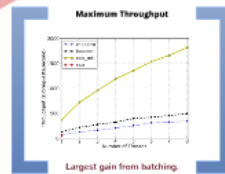


NOX-MT's Performance Under Different Load Levels

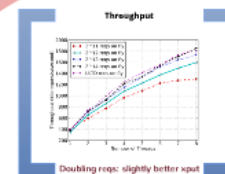




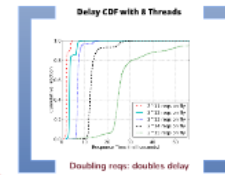
Controller Throughput & Response Time



**Best Xput & Worst Delay For Various Ctrlrs
(i.e., unlimited outstanding requests)**



NOX-MT's Performance Under Different Load Levels



On Controller Performance in Software-Defined Networks

Amin Tootoonchian

Sergey Gurbanov

Yashar Ganjali



Martin Casado

Rob Sherwood

Latency

Throughput

Control



SDN controllers are no longer in the way of control apps.

Throughput and response time should be reported together.

Latency is the important metric.

The new NOX release is based on NOX-MT.

Thanks!

