Friends, not Foes – Synthesizing Existing Transport Strategies for Data Center Networks

Ali Munir
Michigan State University

Ghufran Baig, Syed M. Irteza, Ihsan A. Qazi,
Alex X. Liu, Fahad R. Dogar

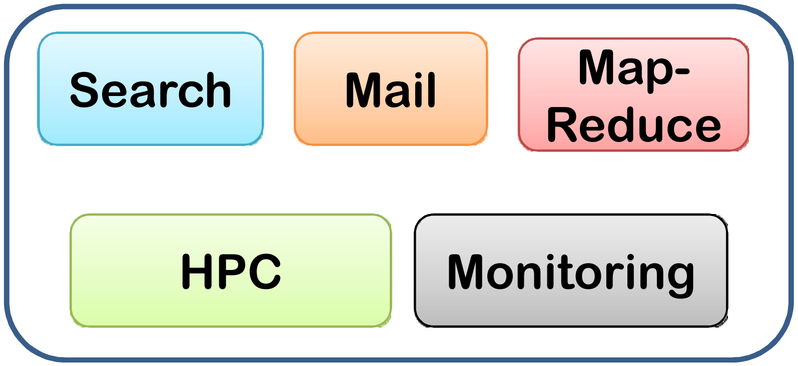
MICHIGAN STATE UNIVERSITY

LUMS

Microsoft Research

Data Center (DC) Applications

- Distributed applications
  Components interact via the network
  e.g., a bing search query touches > 100 machines



- Network impacts performance
  "10% of search responses observe 1 to 14 ms of network queuing delay"
  [DCTCP, SIGCOMM 10]

Image source: http://cdn.slashgear.com/wp-content/uploads/2012/10/google-datacenter-tech-13.jpg

DC Network Resource Allocation

- Fair Sharing

  Equal bandwidth sharing among jobs [TCP, DCTCP]

  - Increases completion time for everyone
  - Traditional "fairness" metrics less relevant

- QoS Aware

  Prioritize some jobs over other jobs (Priority Scheduling)

  - Minimize flow completion times [pFabric, L^2DCT]
  - Meet flow deadlines [D^3, D^2TCP]

DC Transports

DCTCP
SIGCOMM

L^2DCT
INFOCOM’13

D^2TCP
SIGCOMM’12

D^3
SIGCOMM’11

pFabric
SIGCOMM’13

PDQ
SIGCOMM’12

DC Transports

DCTCP
SIGCOMM

L^{2}DCT
INFOCOM'13

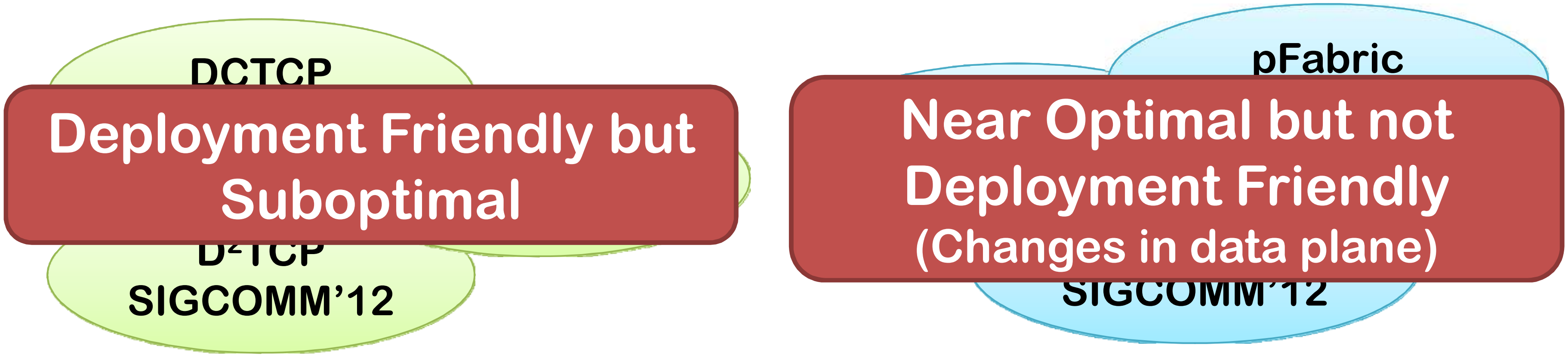
D^{2}TCP
SIGCOMM'12

pFabric

Near Optimal but not Deployment Friendly
(Changes in data plane)

SIGCOMM'12

DC Transports

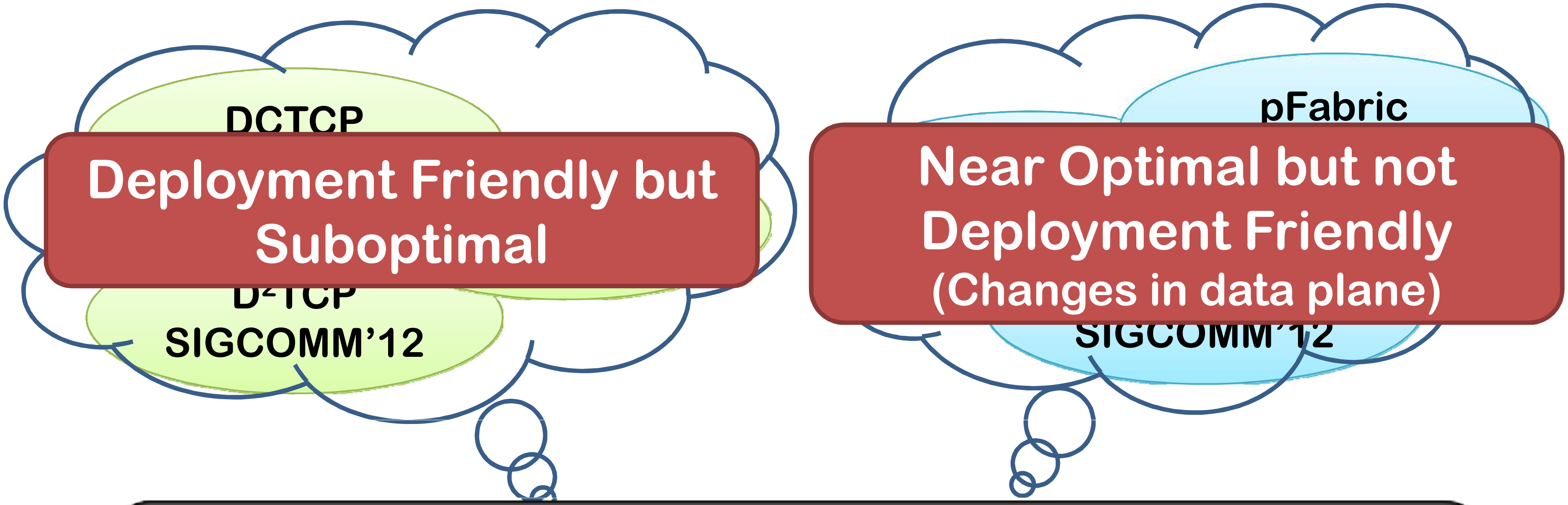


Deployment Friendly but Suboptimal

Near Optimal but not Deployment Friendly

(Changes in data plane)

DC Transports



DCTCP

D²TCP
SIGCOMM’12

Deployment Friendly but Suboptimal

pFabric

SIGCOMM’12

Near Optimal but not Deployment Friendly
(Changes in data plane)

Step back and ask

How can we design a deployment friendly and near optimal data center transport while leveraging the insights offered by existing proposals?

DC Transports

DCTCP

Deployment Friendly but Suboptimal

D^{2}TCP
SIGCOMM'12

pFabric

Near Optimal but not Deployment Friendly
(Changes in data plane)

SIGCOMM'12

Step back and ask

How can we iendly and near optimal dat leveraging the insigh roposals?

PASE

Rest of the Talk …

- DC Transport Strategies
- PASE Design
- Evaluation

Rest of the Talk …

- DC Transport Strategies
- PASE Design
- Evaluation

DC Transport Strategies

- Self-adjusting endpoints e.g., TCP, DCTCP, L^2DCT
  - senders make independent decisions and adjust rate by themselves
- Arbitration e.g., D^3, PDQ
  - a common network entity (e.g., a switch) allocates rates to each flow
- In-network prioritization e.g., pFabric
  - switches schedule and drop packets based on the packet priority

DC Transport Strategies

- Self-adjusting endpoints e.g., TCP, DCTCP, L²DCT
  - senders make independent decisions and adjust rate by themselves
- Ar
  - es

Existing DC transport proposals use only one of these strategies

- In-network prioritization e.g., pFabric
  - switches schedule and drop packets based on the packet priority

Transport Strategies in Isolation

Transport Strategy	Example	Pros	Cons
Self-Adjusting Endpoints	DCTCP, D^2TCP, L^2DCT		
Arbitration	PDQ, D^3		
In-network Prioritization	pFabric		

Transport Strategies in Isolation

Transport Strategy	Example	Pros	Cons
Self-Adjusting Endpoints	DCTCP, D^2TCP, L^2DCT	Ease of deployment	No strict priority scheduling
Arbitration	PDQ, D^3		
In-network Prioritization	pFabric		

Transport Strategies in Isolation

Transport Strategy	Example	Pros	Cons
Self-Adjusting Endpoints	DCTCP, D^{2}TCP, L^{2}DCT	Ease of deployment	No strict priority scheduling
Arbitration	PDQ, D^3	Strict priority scheduling	○ High flow switching overhead ○ Hard to compute precise rates
In-network Prioritization	pFabric		

Transport Strategies in Isolation

Transport Strategy	Example	Pros	Cons
Self-Adjusting Endpoints	DCTCP, D^2TCP, L^2DCT	Ease of deployment	No strict priority scheduling
Arbitration	PDQ, D^3	Strict priority scheduling	○ High flow switching overhead ○ Hard to compute precise rates
In-network Prioritization	pFabric	Low flow switching overhead	○ Switch-local decisions ○ Limited # of priority queues

Transport Strategies in Unison

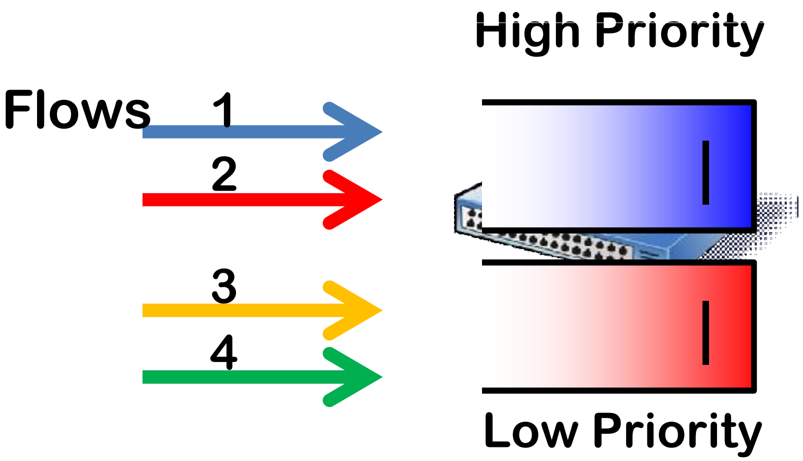
Transport Strategy	Example	Pros	Cons
Self-Adjusting Endpoints	DCTCP, D^2TCP, L^2DCT	Ease of deployment	No strict priority scheduling
Arbitration	PDQ, D^3	Strict priority scheduling	○ High flow switching overhead ○ Hard to compute precise rates
In-network Prioritization	pFabric	Low flow switching overhead	○ Switch-local decisions ○ Limited # of priority queues

Transport Strategies in Unison

In-network Prioritization Alone

Limited # of queues

More # of flows (priorities)



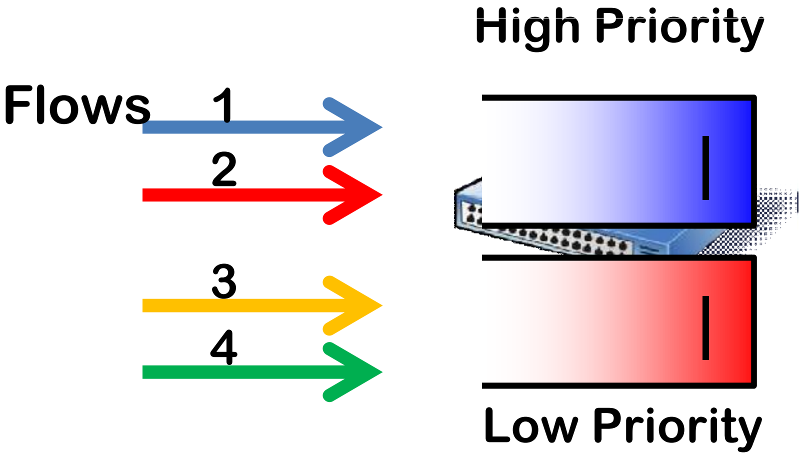
Transport Strategies in Unison

In-network Prioritization Alone

Limited # of queues
More # of flows (priorities)

→

Flow Multiplexing
Limited performance gains!



Any static mapping mechanism degrades performance!

Transport Strategies in Unison

In-network Prioritization + Arbitration

Arbitrator

Dynamic mapping of flows to queues

→

Idea

As a flow’s turn comes, map it to the highest priority queue!

Transport Strategies in Unison

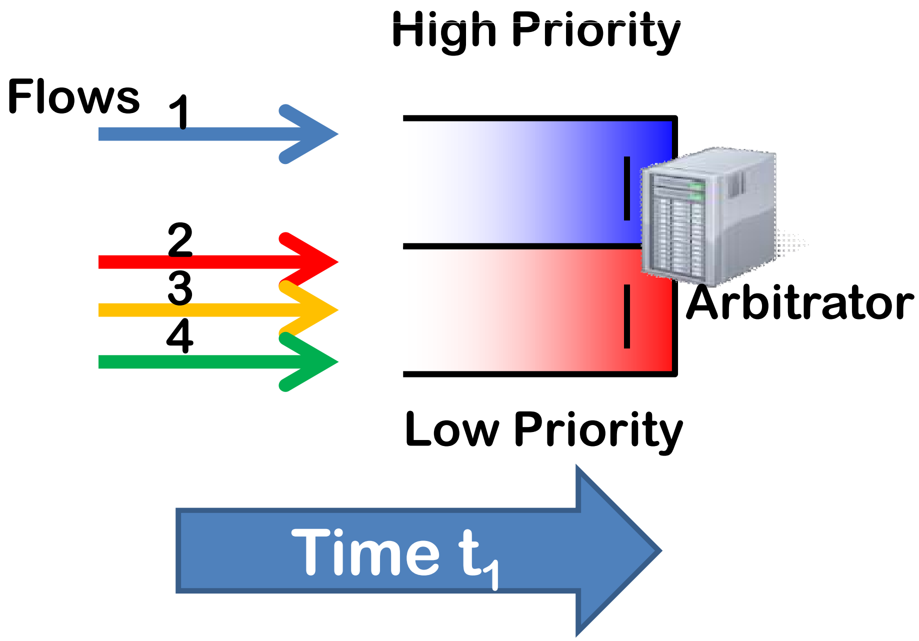
In-network Prioritization + Arbitration

Arbitrator

Dynamic mapping of flows to queues

Idea

As a flow's turn comes, map it to the highest priority queue!



Transport Strategies in Unison

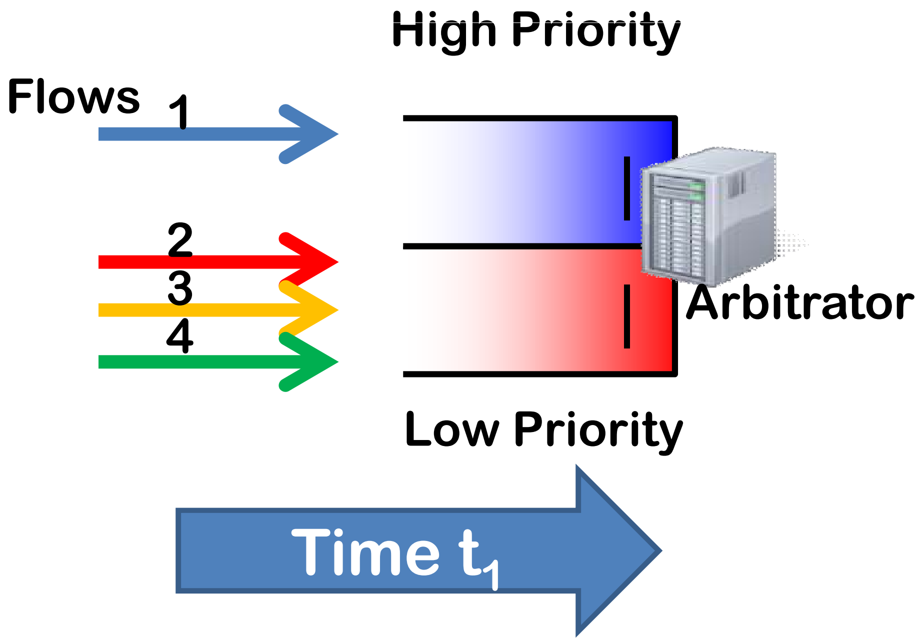
In-network Prioritization + Arbitration

Arbitrator

Dynamic mapping of flows to queues

Idea

As a flow's turn comes, map it to the highest priority queue!



High Priority

Flows

1

2

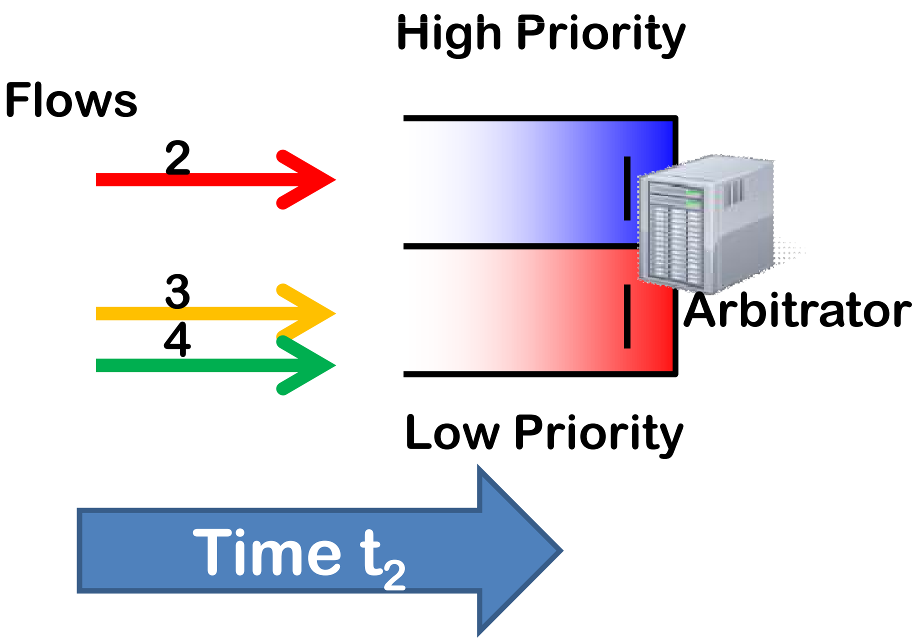
3

4

Arbitrator

Low Priority

Time t_1



High Priority

Flows

2

3

4

Arbitrator

Low Priority

Time t_2

Transport Strategies in Unison

In-network Prioritization + Arbitration

Arbitrator

Dynamic mapping of
flows t

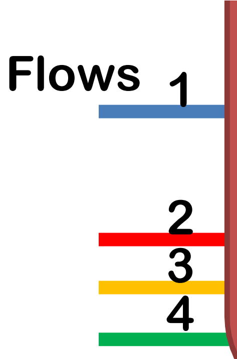
Idea

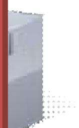
As a flow's turn comes, map it
ue!

Similarly,

- Arbitration + Self-Adjusting Endpoints
- Arbitration + In-network Prioritization

PASE leverages these insights in its design!





Low Priority

Low Priority

Time t_1

Time t_2

Rest of the Talk …

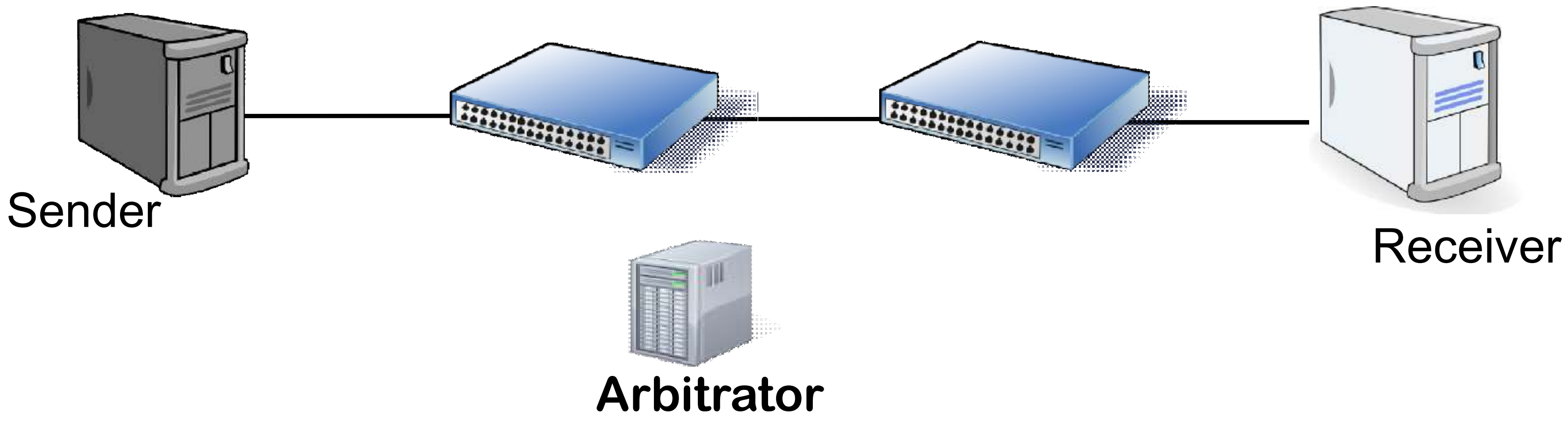
- DC Transport Strategies
- **PASE Design**
- Evaluation

PASE Design Principle

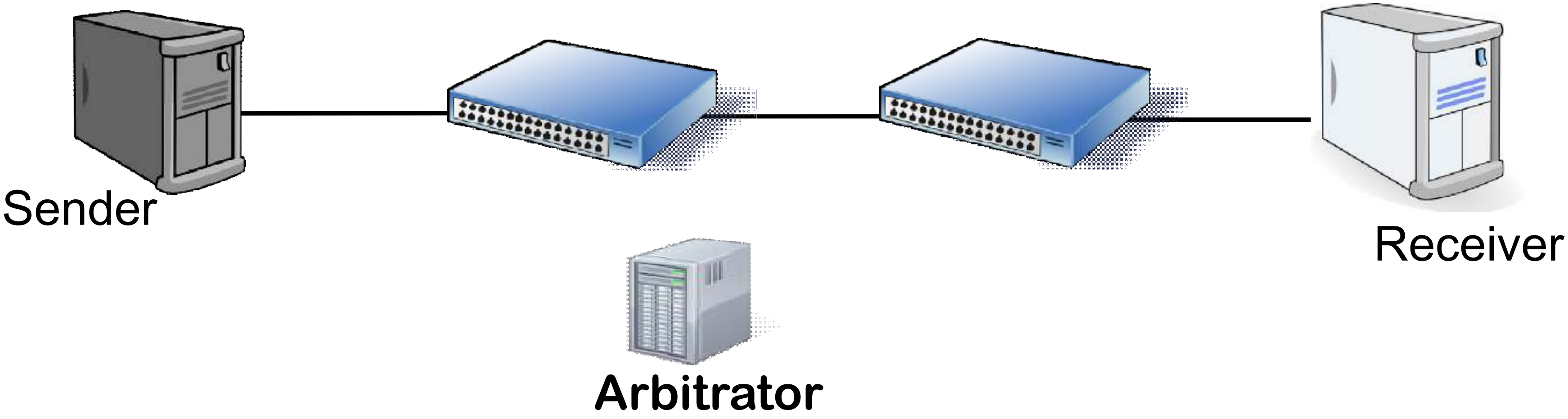
Each transport strategy should focus on what it is best at doing!

- Arbitrators
  - Do inter-flow prioritization at coarse time-scales
- Endpoints
  - Probe for any spare link capacity
- In-network prioritization
  - Do per-packet prioritization at sub-RTT timescales

PASE Overview



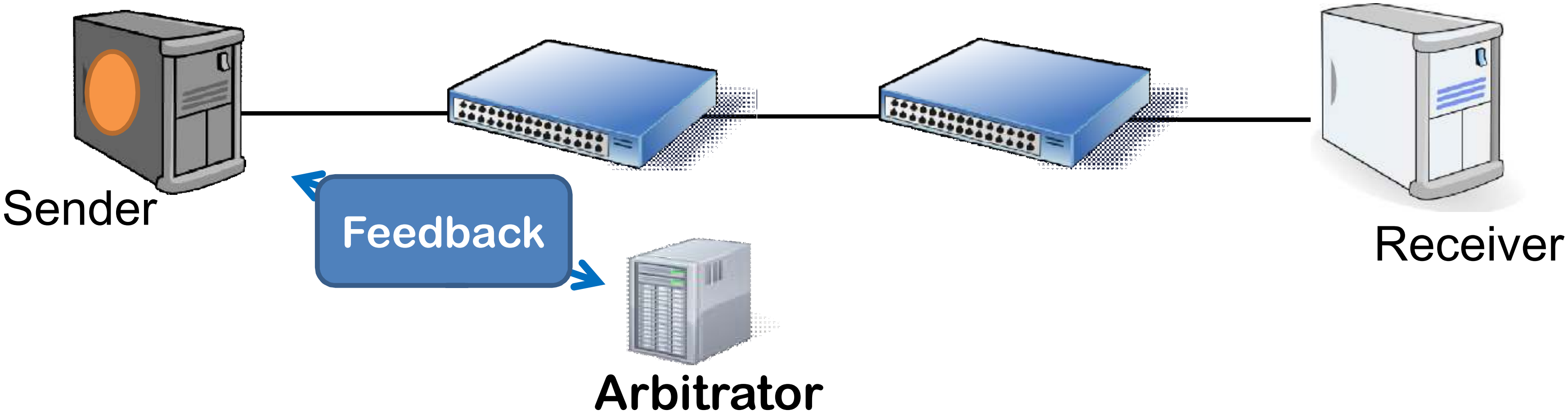
PASE Overview



- Arbitration: Control plane

Calculate “reference rate” and “priority queue”

PASE Overview



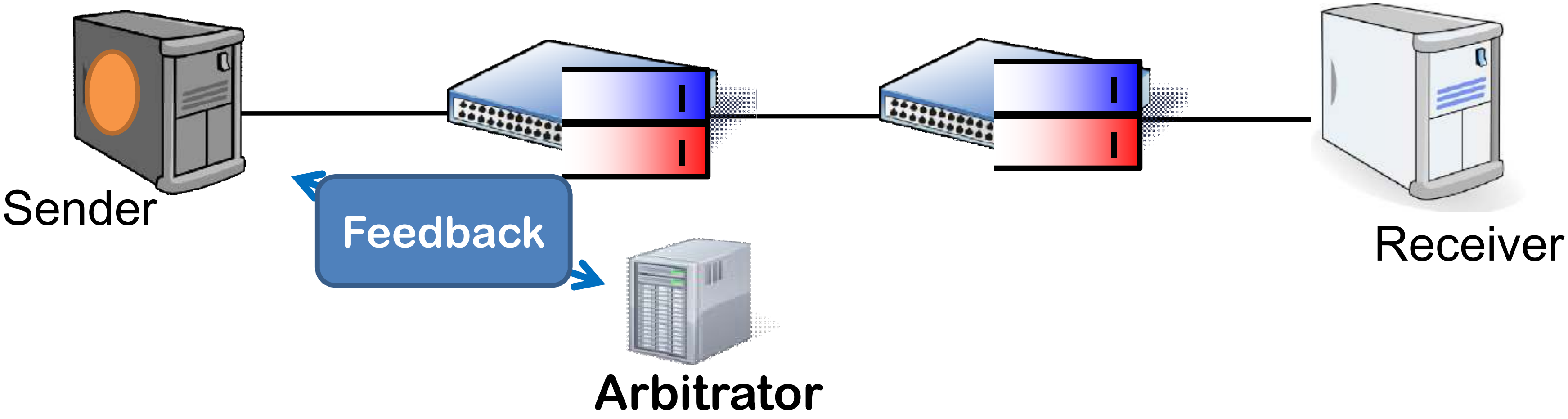
- Arbitration: Control plane

Calculate “reference rate” and “priority queue”

- Self-Adjusting Endpoints: Guided rate control

Use arbitrator feedback as a pivot

PASE Overview



- Arbitration: Control plane

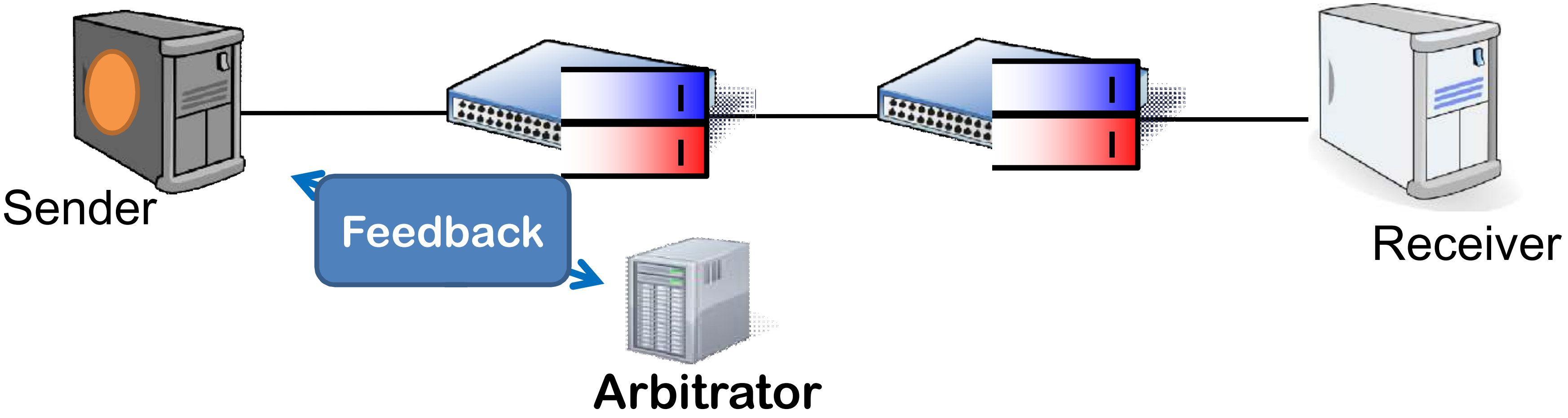
Calculate “reference rate” and “priority queue”

- Self-Adjusting Endpoints: Guided rate control

Use arbitrator feedback as a pivot

- In-network Prioritization: Existing priority queues

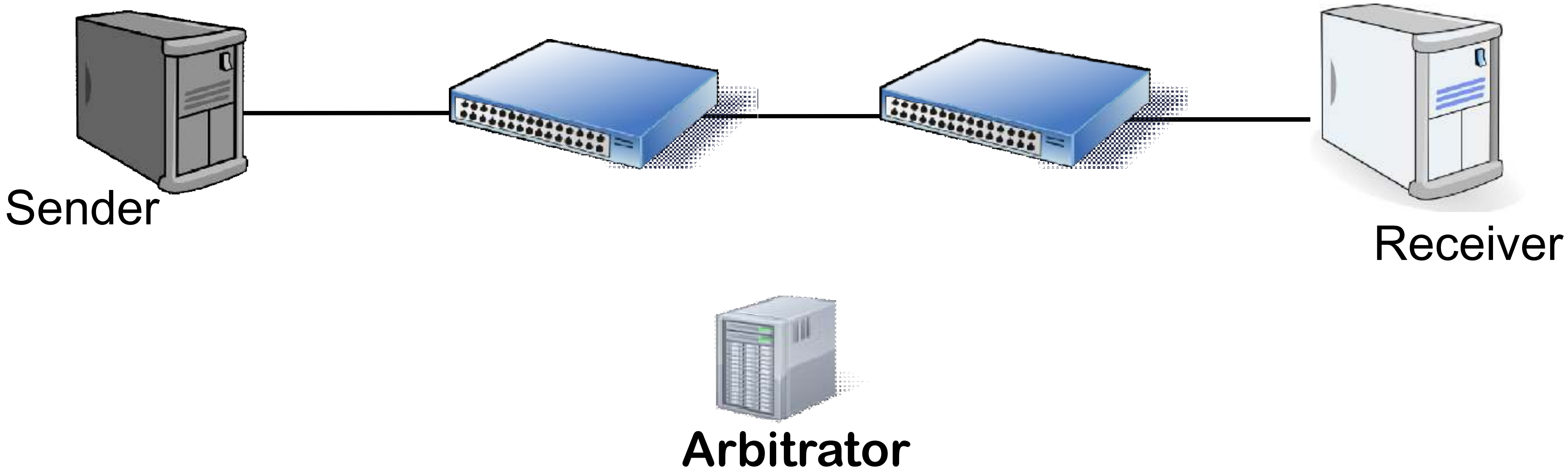
PASE Overview



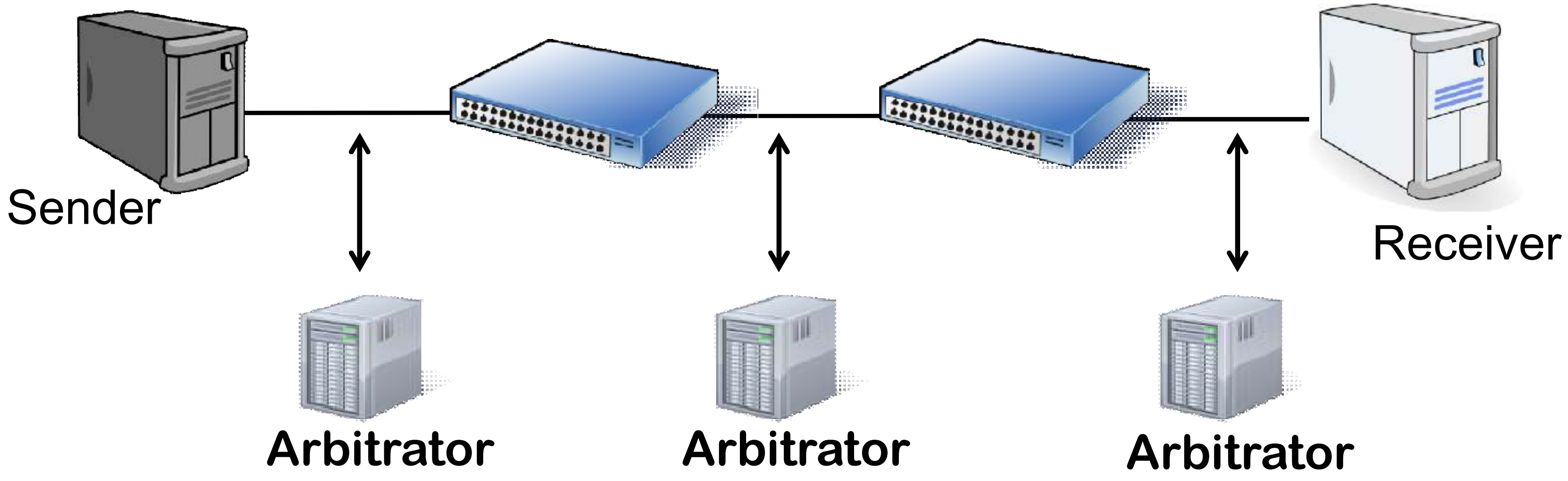
- **Arbitration: Control plane**
  Calculate “reference rate” and “priority queue”
- **Self-Adjusting Endpoints: Guided rate control**
  Use arbitrator feedback as a pivot
- **In-network Prioritization: Existing priority queues**

Key Components

PASE Arbitration



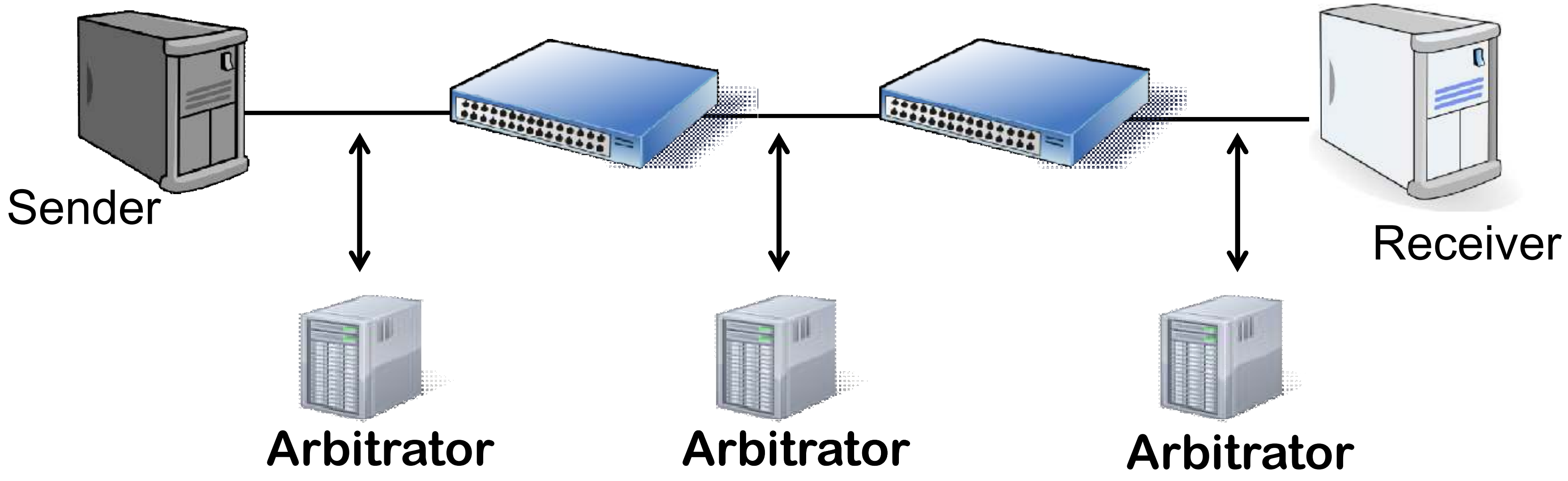
PASE Arbitration



Distributed Arbitration

- per link arbitration done in control plane
- existing protocols implement in data plane

PASE Arbitration



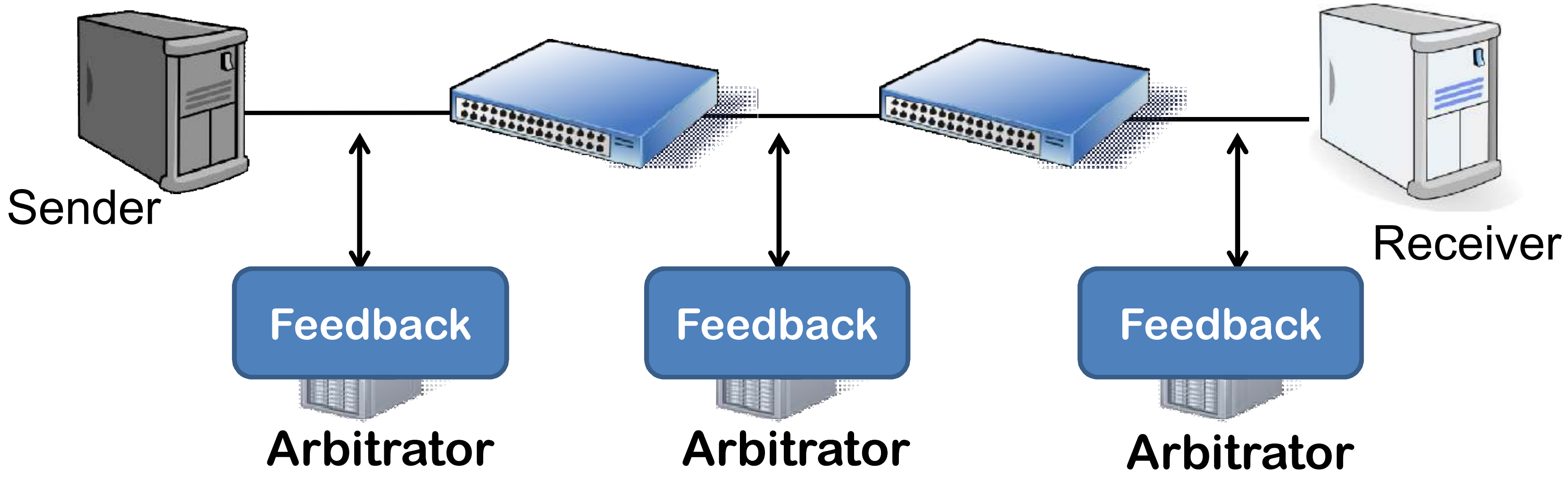
Distributed Arbitration

- per link arbitration done in control plane
- existing protocols implement in data plane

Arbitrator Location

- at the end hosts (e.g., for their own links to the switch) OR
- on dedicated hosts inside the DC

PASE Arbitration



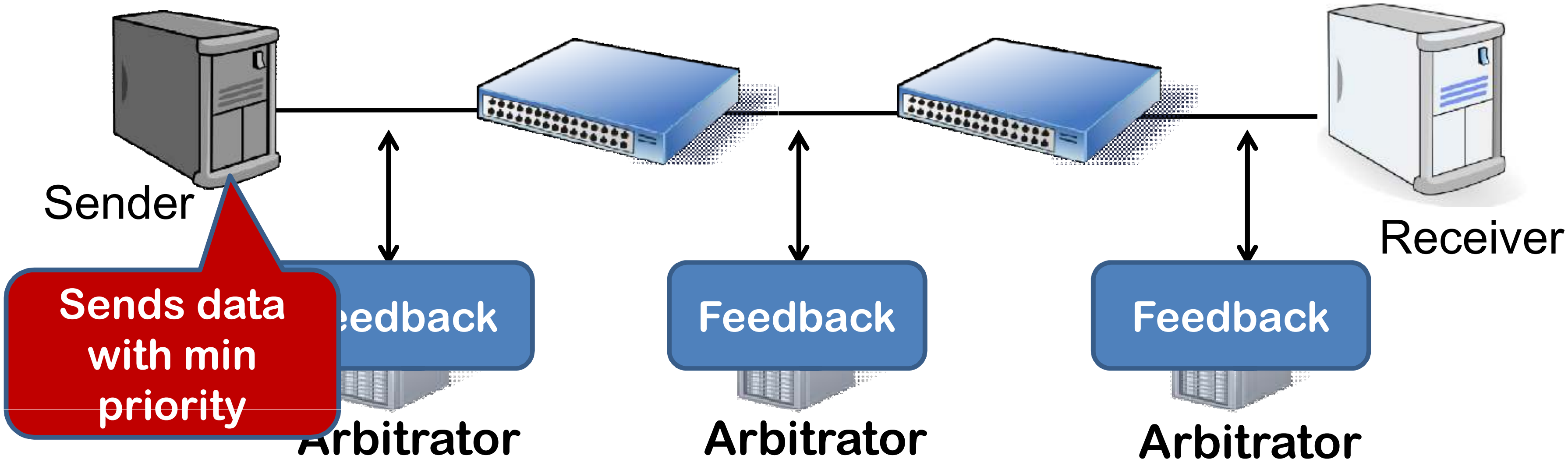
Distributed Arbitration

- per link arbitration done in control plane
- existing protocols implement in data plane

Arbitrator Location

- at the end hosts (e.g., for their own links to the switch) OR
- on dedicated hosts inside the DC

PASE Arbitration



Distributed Arbitration

- per link arbitration done in control plane
- existing protocols implement in data plane

Arbitrator Location

- at the end hosts (e.g., for their own links to the switch) OR
- on dedicated hosts inside the DC

PASE Arbitration – Challenges

- Challenges
  - Arbitration latency
  - Processing overhead
  - Network overhead

PASE Arbitration – Challenges

- Challenges
  - Arbitration latency
  - Processing overhead
  - Network overhead

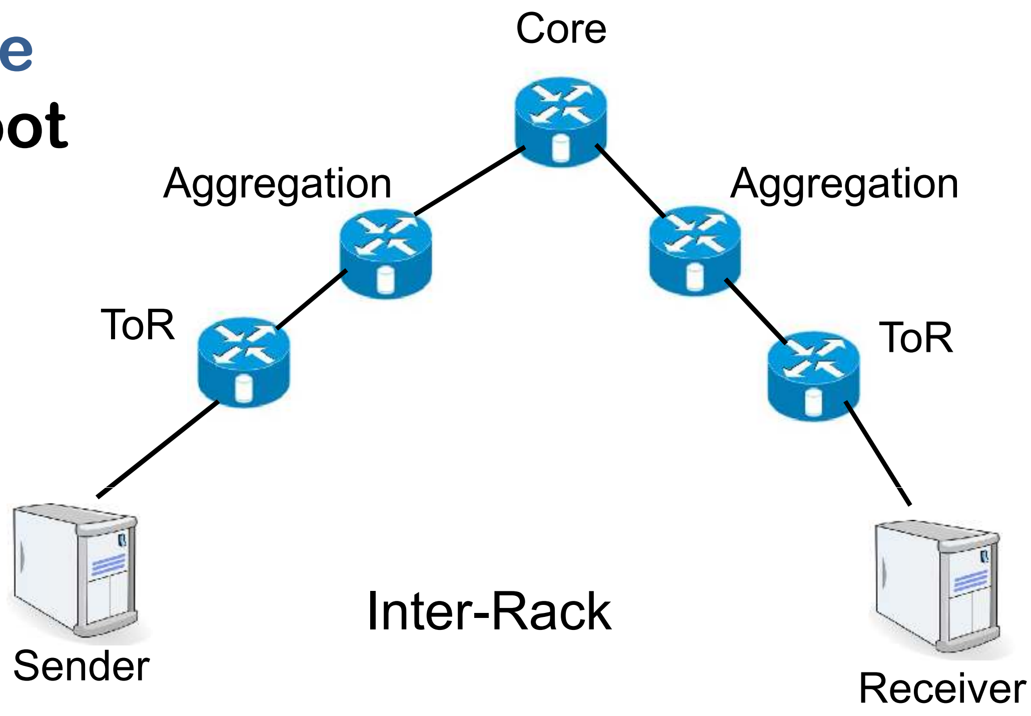
Solution: Leverage the tree-like structure of typical DC topologies

Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root

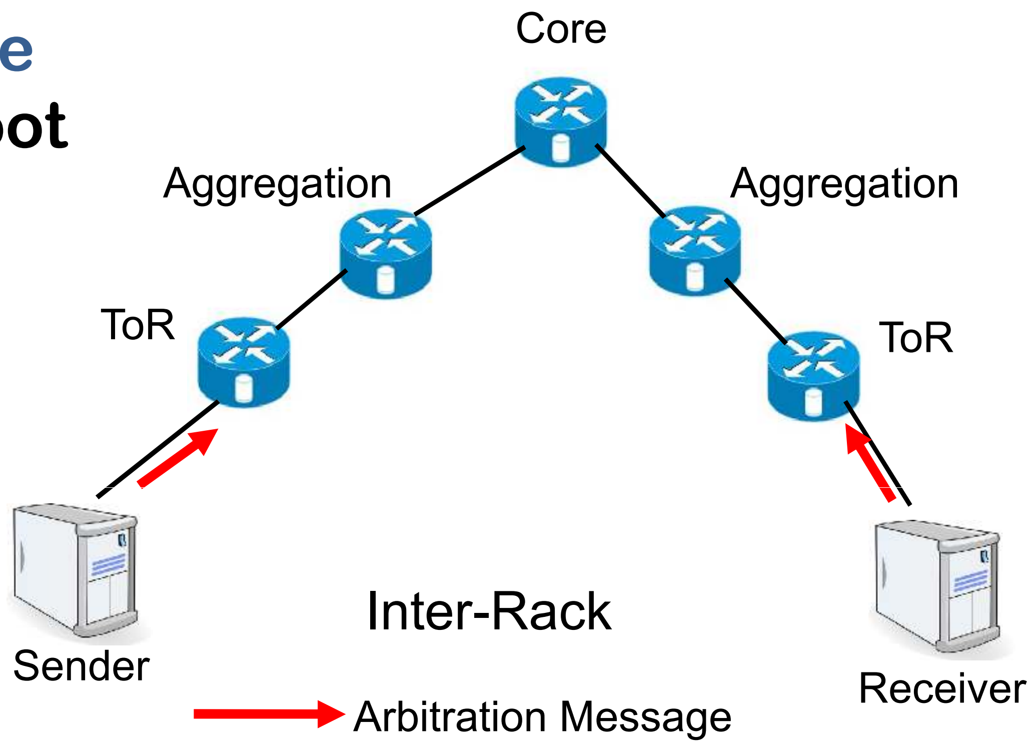
Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root



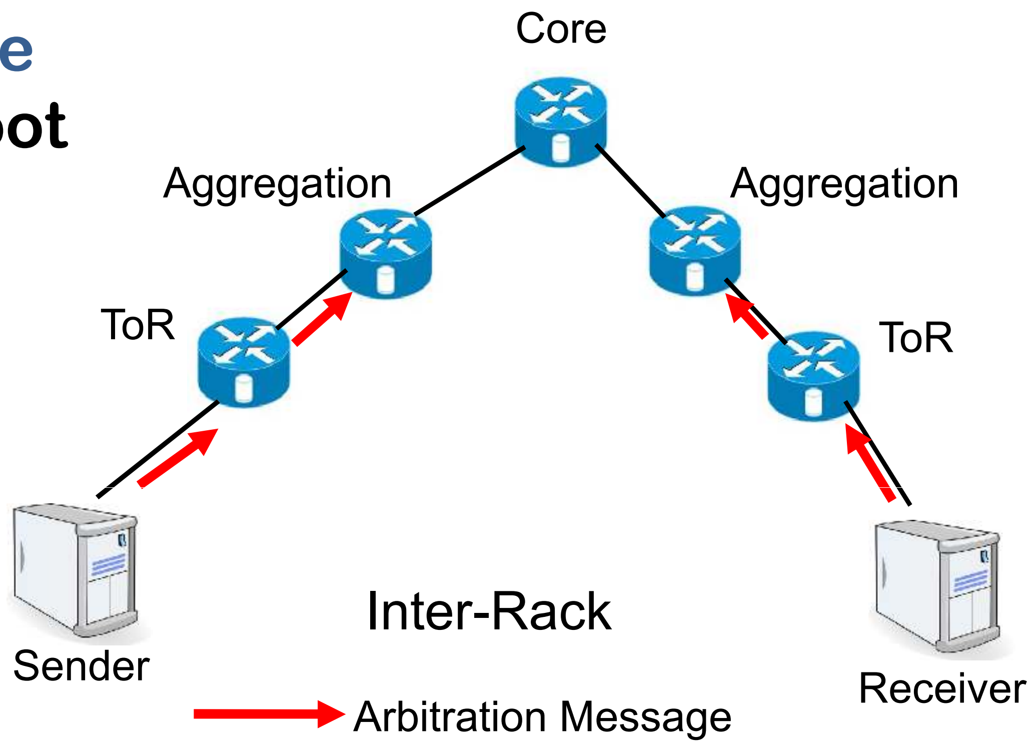
Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root



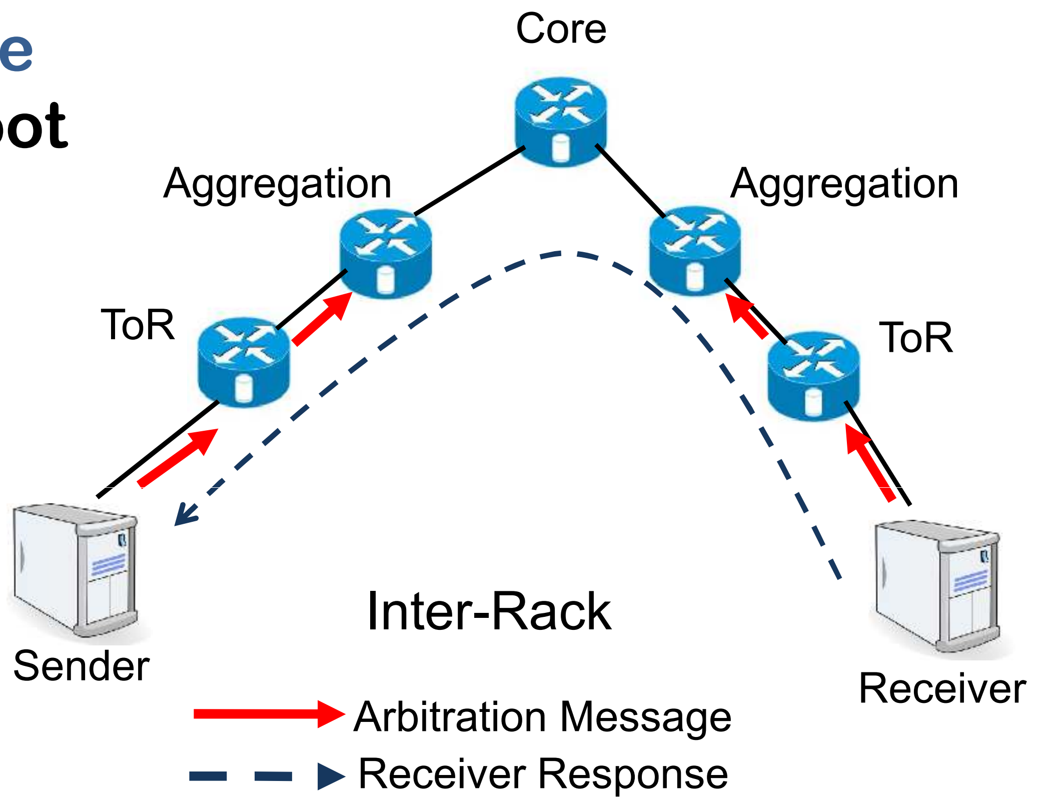
Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root



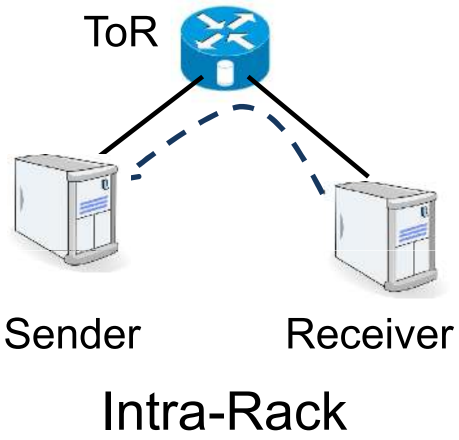
Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root

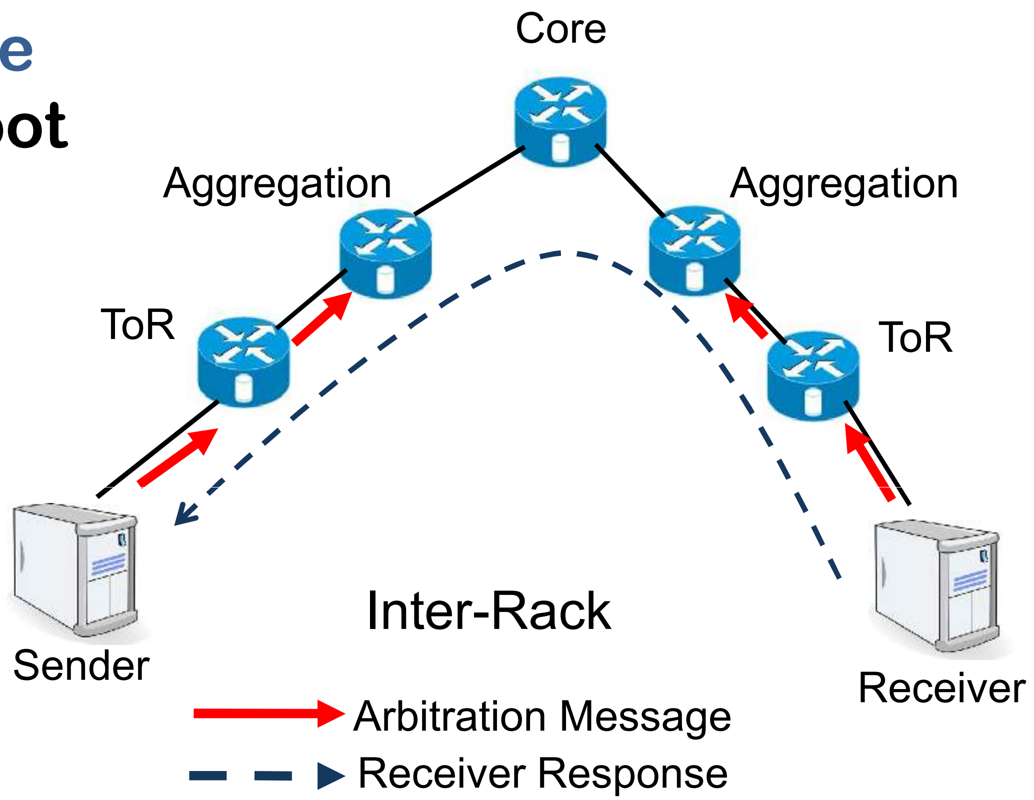


Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root

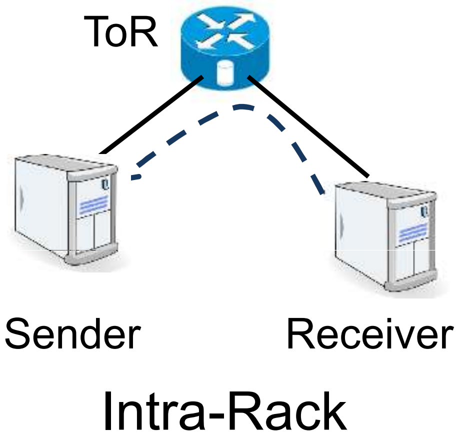


No external arbitrators required!

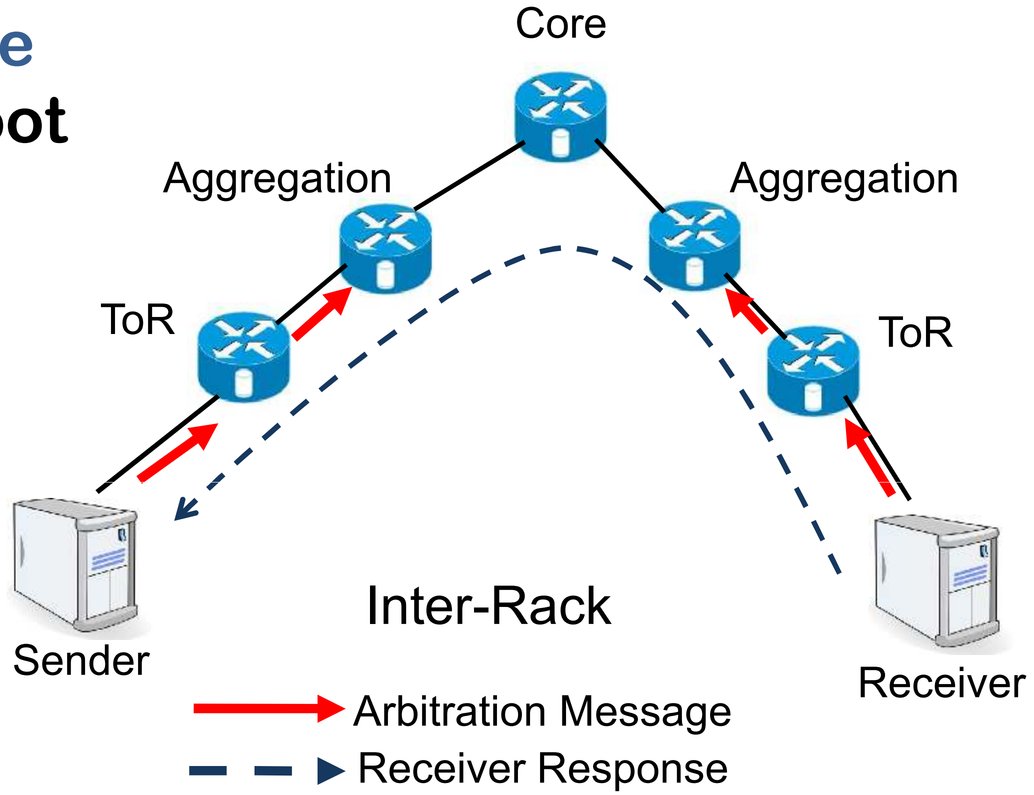


Bottom Up Arbitration

- **Leverage Tree Structure**
  from leaves up to the root

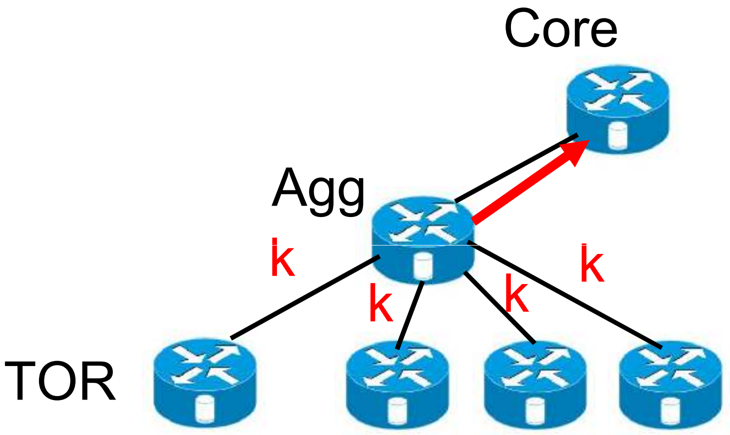


No external arbitrators required!



Facilitates inter-rack optimizations (early pruning & delegation) to reduce arbitration overhead.

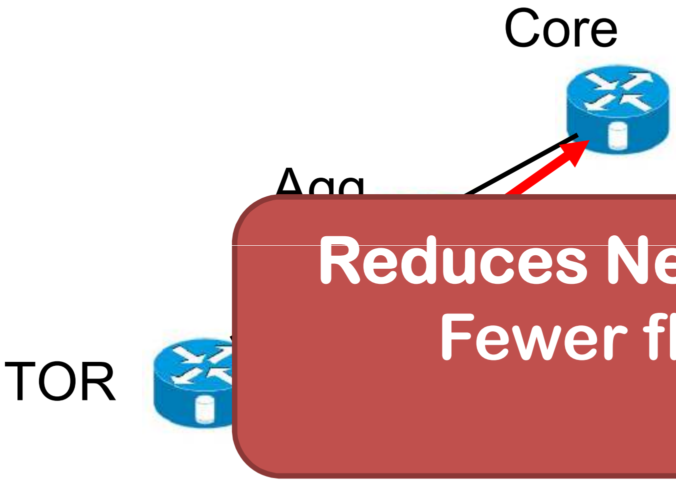
Early Pruning



Arbitration involves sorting flows and picking **top k** for immediate scheduling

Flows that won’t make it to top k queues should be pruned at lower levels

Early Pruning



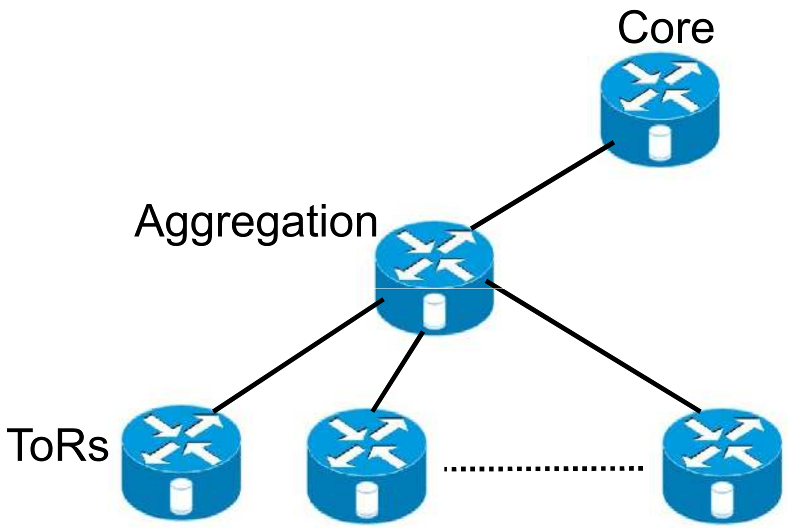
Arbitration involves sorting flows and picking **top k** for immediate scheduling

Flows that won't make it to top k queues should be pruned at lower levels

Reduces Network and Processing overhead
Fewer flows contact the higher level arbitrators!

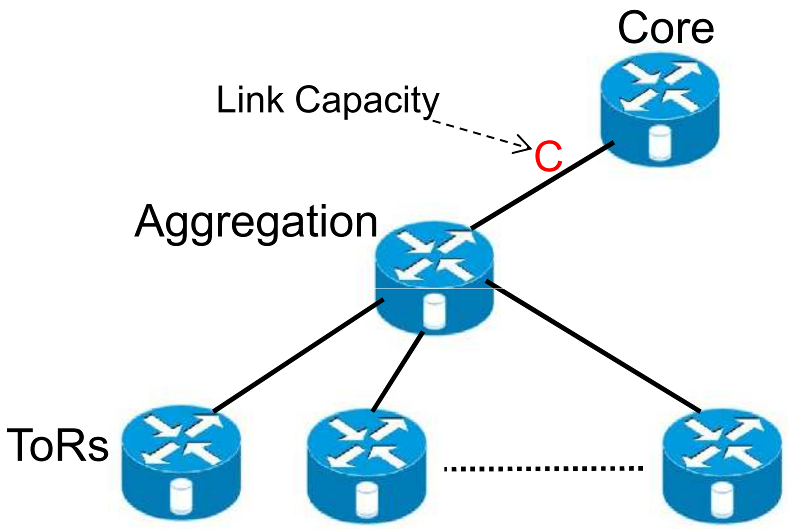
Delegation

Key Idea: Divide a link into virtual links and delegate responsibility to child arbitrators



Delegation

Key Idea: Divide a link into virtual links and delegate responsibility to child arbitrators

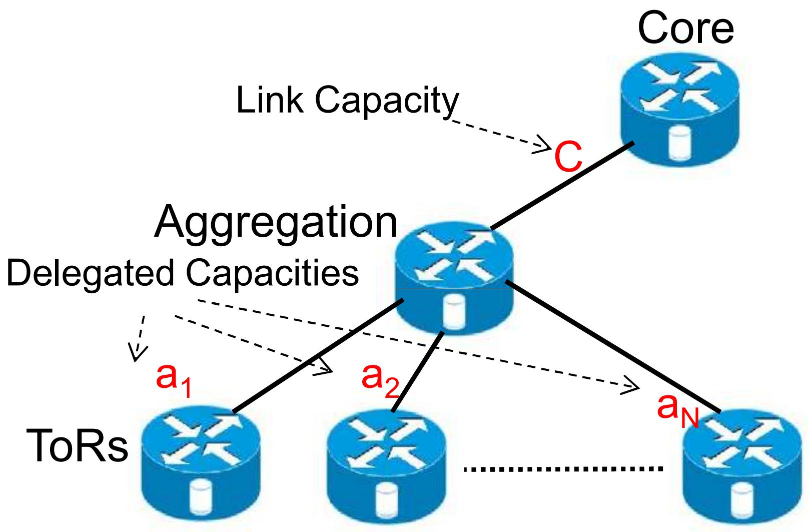


- **Algorithm**

  Link capacity C is split in N virtual links

Delegation

Key Idea: Divide a link into virtual links and delegate responsibility to child arbitrators



- **Algorithm**

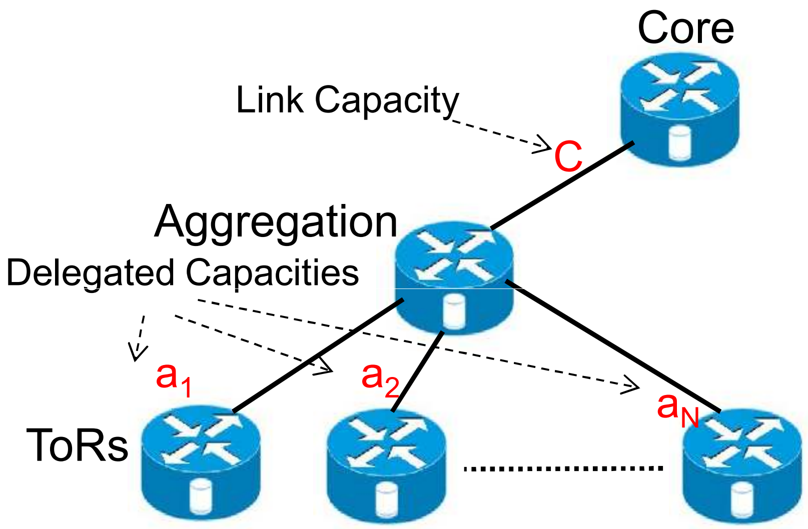
  Link capacity C is split in N virtual links

  ⇓

  Parent arbitrator delegates virtual link to child arbitrator

Delegation

Key Idea: Divide a link into virtual links and delegate responsibility to child arbitrators



- **Algorithm**

Link capacity C is split in N virtual links

↓

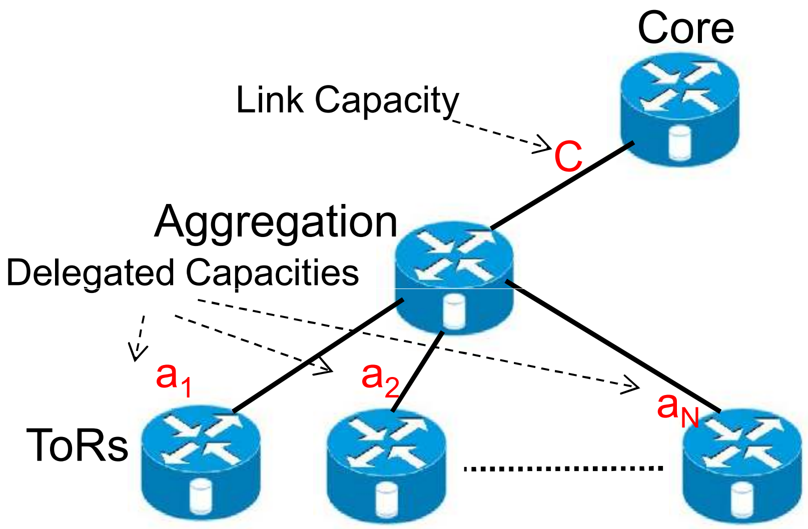
Parent arbitrator delegates virtual link to child arbitrator

↓

Child arbitrator does arbitration for virtual link

Delegation

Key Idea: Divide a link into virtual links and delegate responsibility to child arbitrators



- **Algorithm**

Link capacity C is split in N virtual links

↓

Parent arbitrator delegates virtual link to child arbitrator

↓

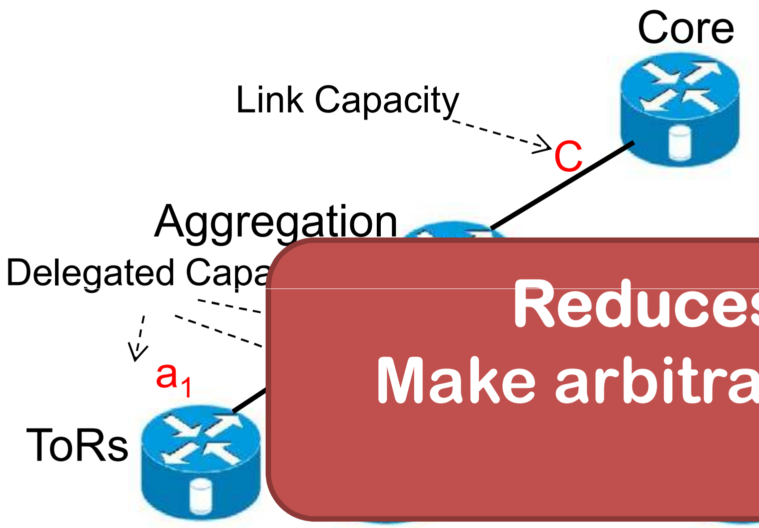
Child arbitrator does arbitration for virtual link

↓

Virtual link capacity is periodically updated based on the top k flows of all child arbitrators

Delegation

Key Idea: Divide a link into virtual links and delegate responsibility to child arbitrators



- **Algorithm**

  Link capacity C is split in N virtual links

  ates

  rator

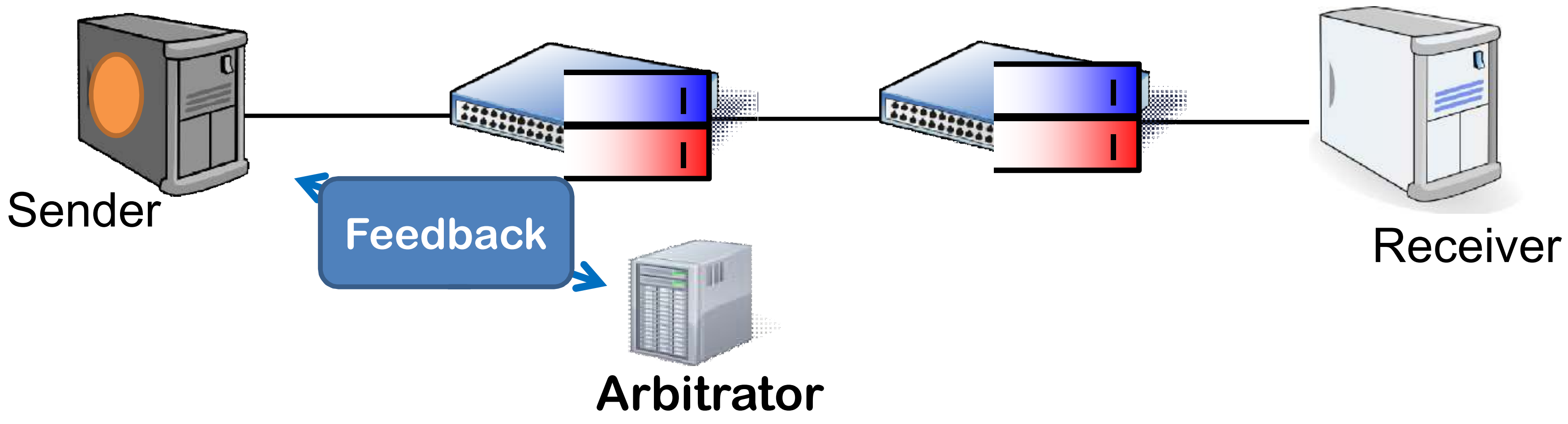
Child arbitrator does arbitration for virtual link

Virtual link capacity is periodically updated based on the top k flows of all child arbitrators

Reduces Arbitration Latency

Make arbitration decision close to the sources

PASE Overview



- Arbitration: Control plane
  Calculate “reference rate” and “priority queue”
- Self-Adjusting Endpoints: Guided rate control
  Use arbitrator feedback as a pivot
- In-network Prioritization: Existing priority queues

PASE Endhost Transport

- Rate Control

- Loss Recovery Mechanism

PASE Endhost Transport

- Rate Control

  Use reference rate and priority feedback from arbitrators

  – Use reference-rate as pivot, and

  – Follow DCTCP control laws

- Loss Recovery Mechanism

PASE Endhost Transport

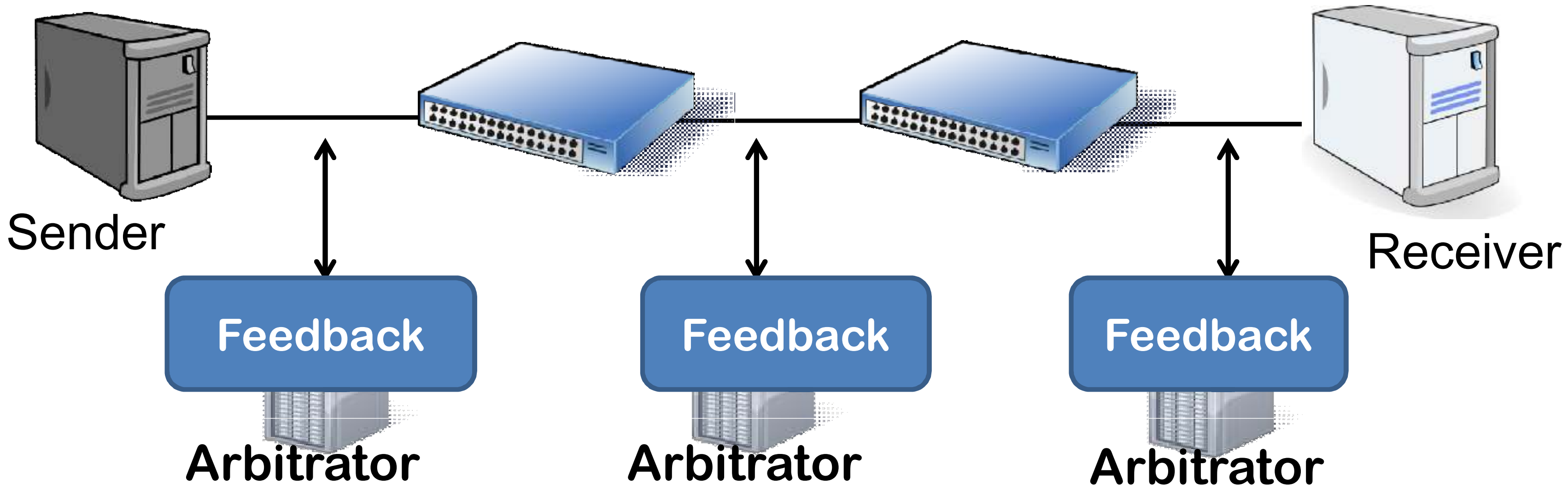
- Rate Control

  Use reference rate and priority feedback from arbitrators

  - Use reference-rate as pivot, and
  - Follow DCTCP control laws

- Loss Recovery Mechanism
  - Packets in lower priority queues can be delayed for several RTTs
    - large RTO OR small probe to avoid spurious retransmissions

PASE -- Putting it Together



- Efficient arbitration control plane
- Simple TCP-like transport
- Existing priority queues inside switches

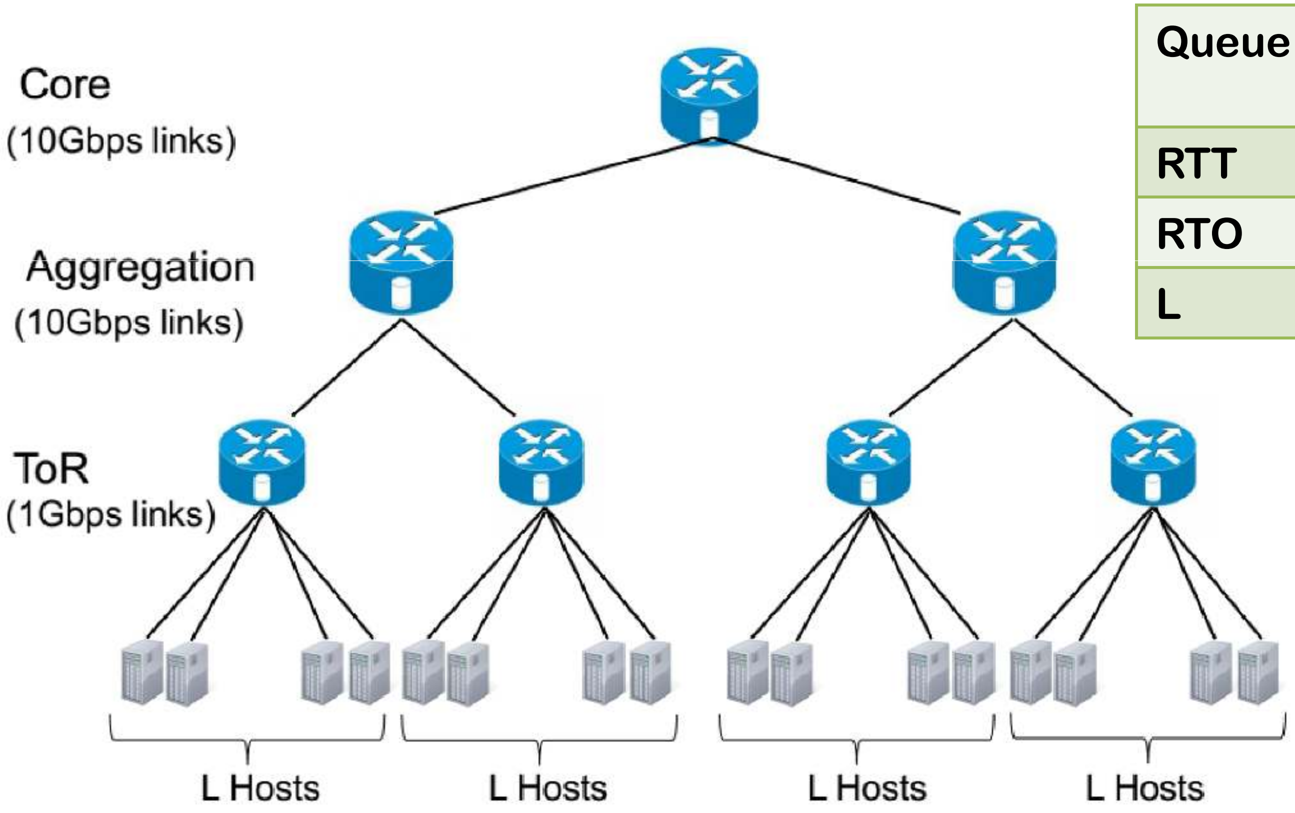
Rest of the Talk …

- DC Transport Strategies
- PASE Design
- Evaluation

Evaluation

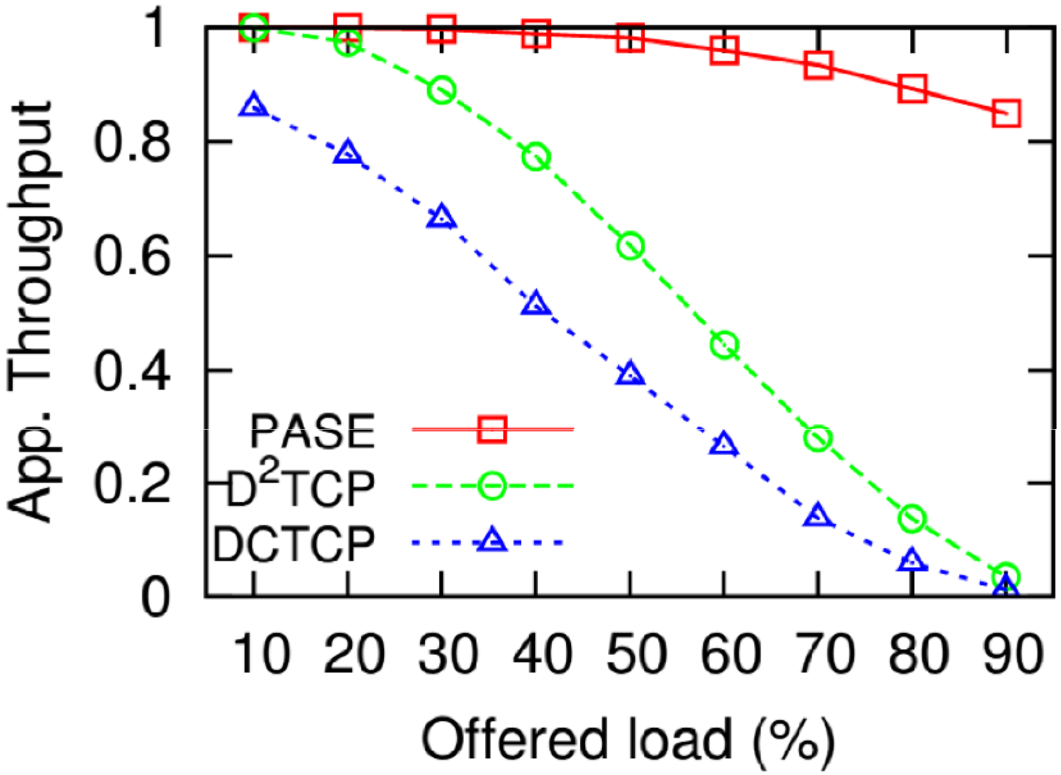
- Platforms
  - Small scale testbed
  - NS2
- Workloads
  - Web search (DCTCP), Data mining (VL2)
- Comparison with deployment friendly
  - DCTCP, D^2TCP, L^2DCT
- Comparison with state of the art
  - pFabric

Simulation Setup



Queue Size	250KB (per queue)
RTT	300usec
RTO	1 msec
L	40

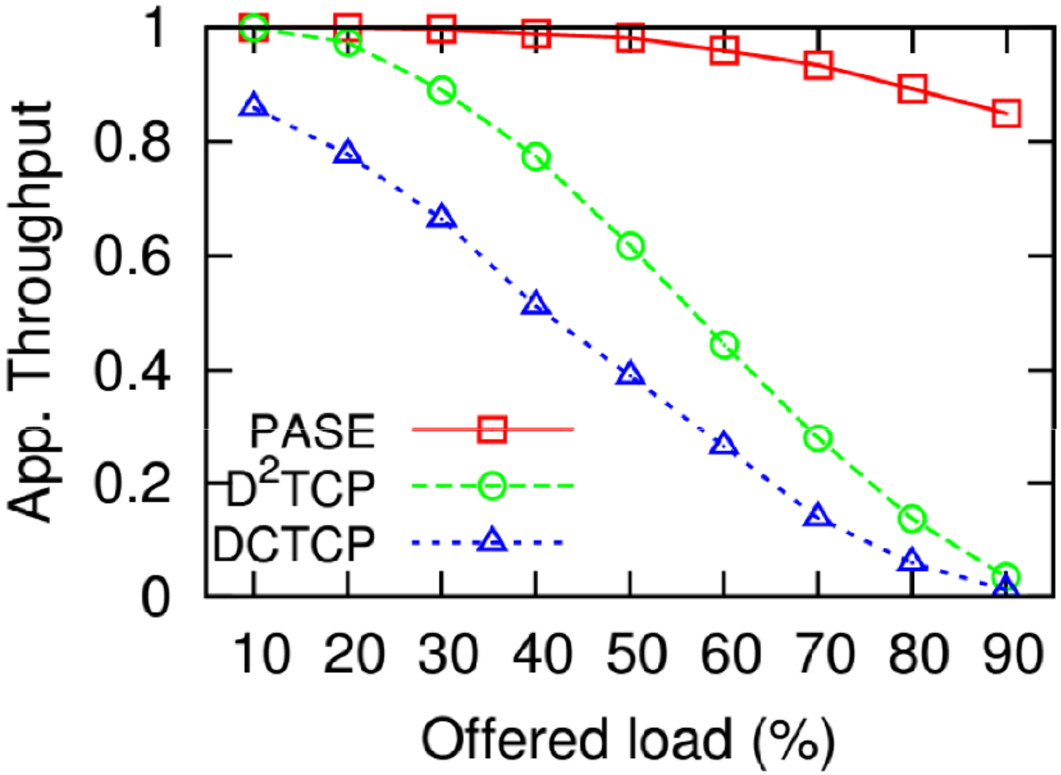
Comparison with Deployment Friendly



Settings similar to D^{2}TCP

- **Flow Sizes: 100-500KB**
- **Deadlines: 5-25msec**

Comparison with Deployment Friendly

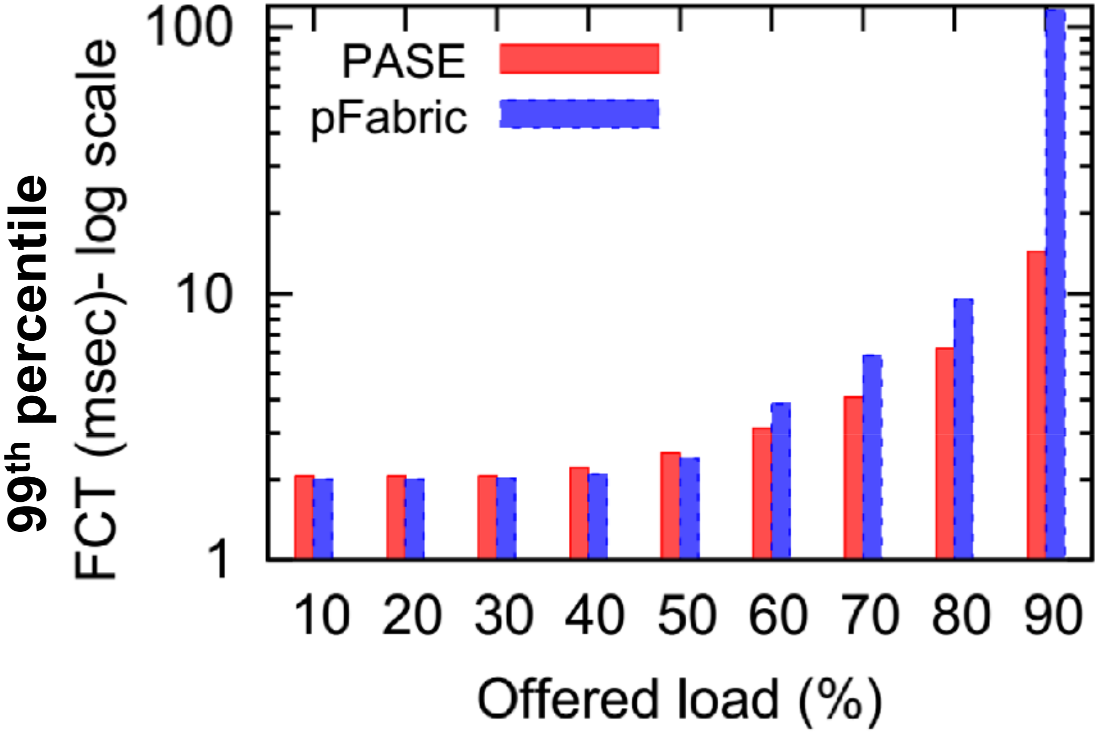


Settings similar to D^{2}TCP

- Flow Sizes: 100-500KB
- Deadlines: 5-25msec

PASE is deployment friendly yet performs BETTER than existing protocols!

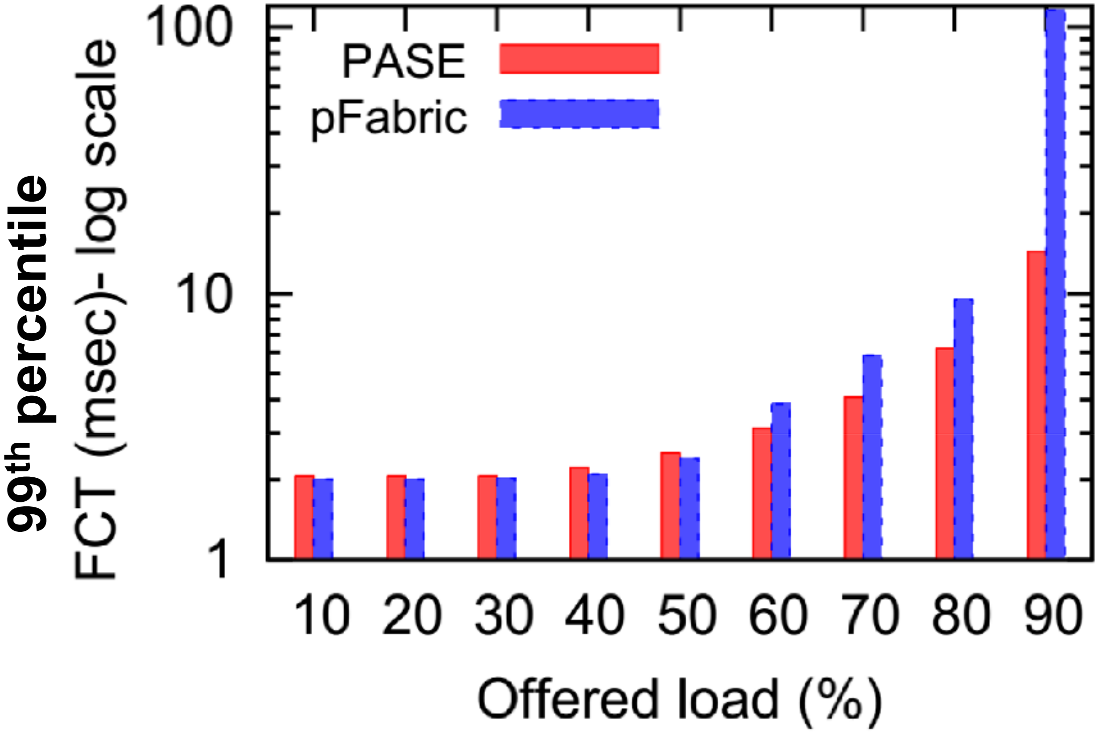
Comparison with State of the Art



Settings

- **Flow Sizes: 2-98KB**
- **Left-to-right traffic**

Comparison with State of the Art



Settings

- **Flow Sizes: 2-98KB**
- **Left-to-right traffic**

PASE performs comparable and does not require changes to data plane

Summary

- Key Strategies for Existing DC Transport
  - Arbitration, in-network Prioritization, Self-Adjusting End-points
  - Complimentary rather than substitutes
- PASE
  - Combines the three strategies
  - Efficient arbitration control plane; simple TCP-like transport; leverages existing priority queues inside switches
- Performance
  - Comparable to or better than earlier proposals that even require changes to the network fabric

Thank you!