Söze: One Network Telemetry Is All You Need for Per-flow Weighted Bandwidth Allocation at Scale (To appear in OSDI'25)

T. S. Eugene Ng

with

Weitao Wang

Rice University

Optical circuit switched reconfigurable networks are **important in the long run**

- Low power
- Reliable
- Data rate agnostic

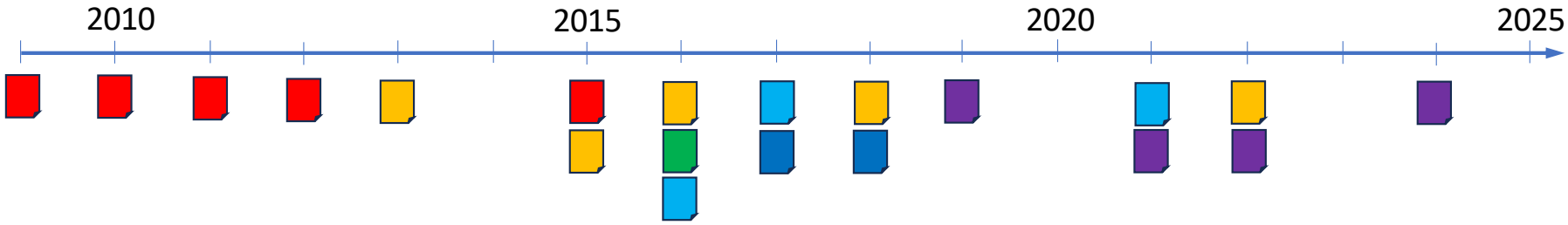
} Cost effective

- Not to mention there are many technical challenges

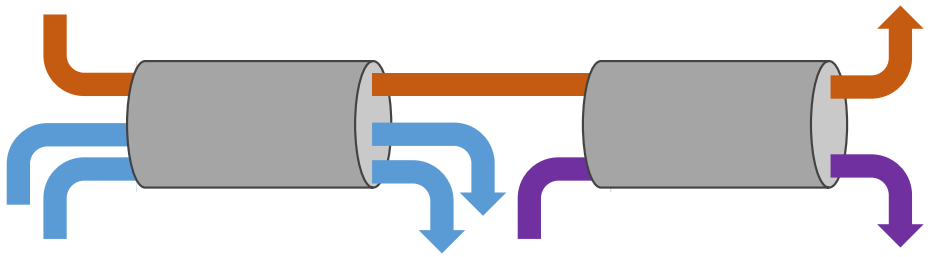
- Moving bandwidth
  - c-Through HotNets'09, SIGCOMM'10, Switching divide SoCC'11, For big data HotSDN'12, OmniSwitch HotCloud'15
- Optical multicast
  - *-cast CCR'13, Blast INFOCOM'15, HyperOptics HotCloud'16, Republic ICNP'18, Shufflecast ToN'22
- Coflow aware circuit scheduling
  - Sunflow CoNEXT'16
- Whole network topology transformation
  - Flat-tree HotNets'16, SIGCOMM'17, Transtate OptSys'21
- Failure resilience
  - ShareBackup HotNets'17, SIGCOMM'18
- Reshaping traffic at the edge
  - RDC SOSR'19, NSDI'22, OSSV OptSys'21, INFOCOM'24

Former PhD students

Guohui Wang
Yiting Xia
Xiaoye Sun
Xin Sunny Huang
Dingming Wu
Afsaneh Rahbar
Sushovan Das
Weitao Wang



Motivation: Differentiated Services Benefit Applications



Weighted fair sharing allocation:

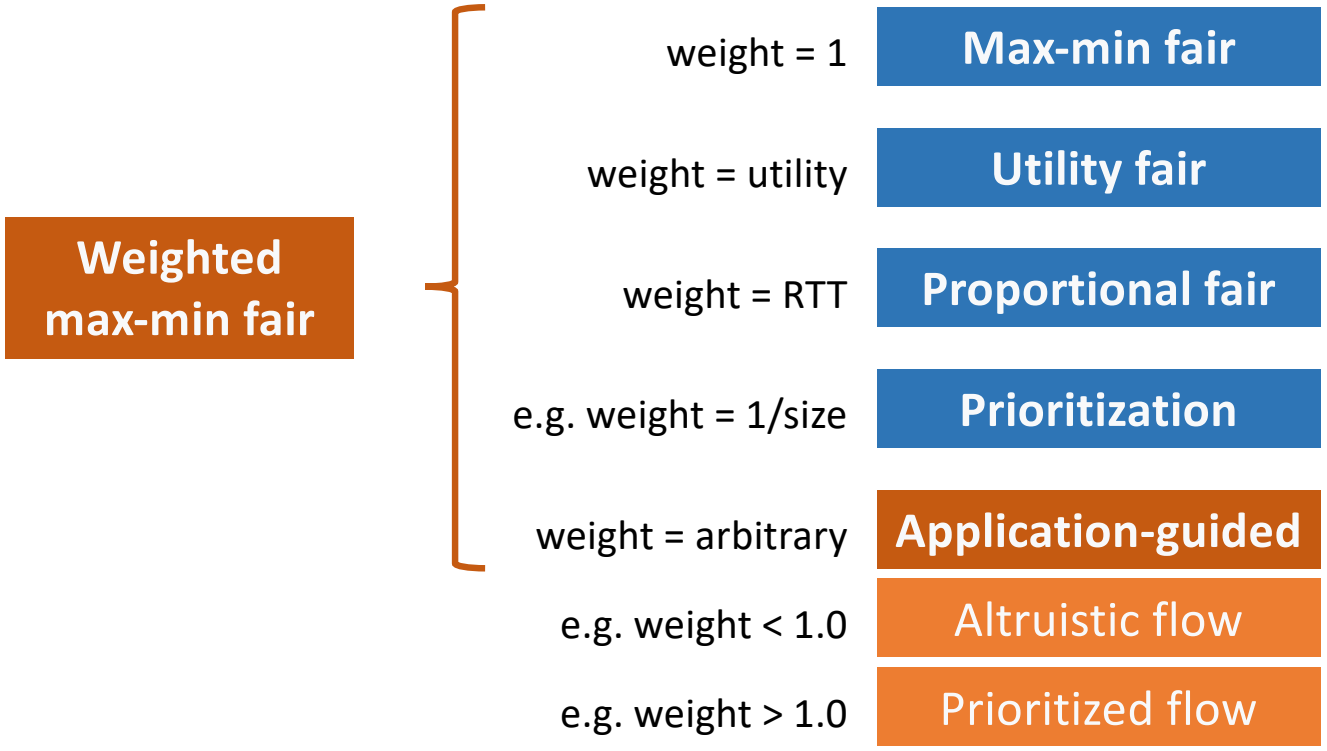
Orange (w = 2) = 50%

Blue (w = 1) = 25%

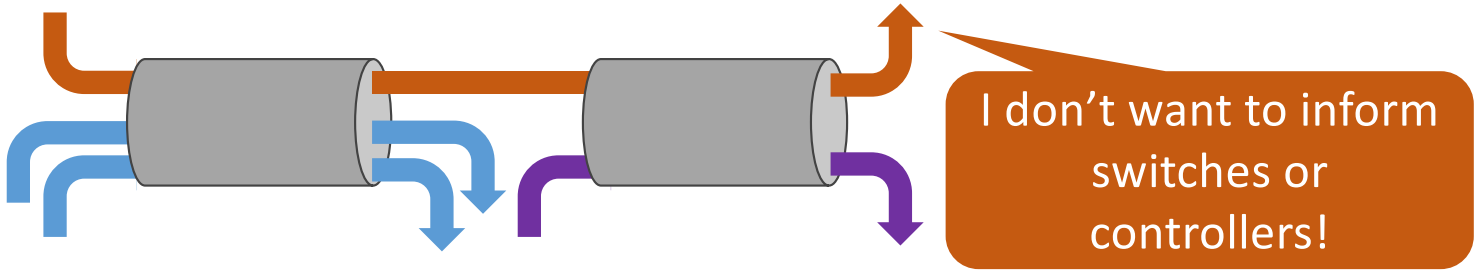
Purple (w = 1) = 50%

Weighted allocation could benefit critical applications, or critical path in applications.

Benefits of weighted bandwidth allocation



Challenge: Change Weight with Minimal Information Exchange



Weighted fair sharing allocation:

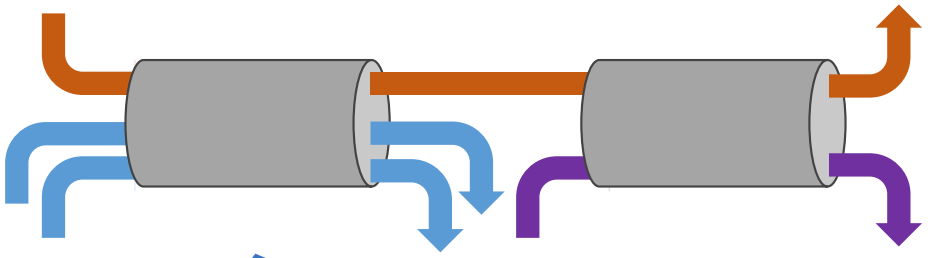
Orange (w = 2) = 50%

Blue (w = 1) = 25%

Purple (w = 1) = 50%

How can orange flow get more bandwidth with minimal overhead?

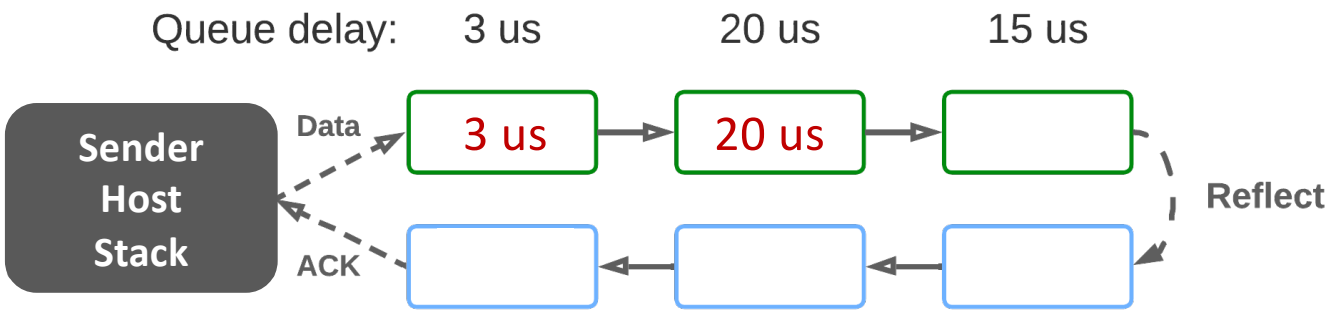
Key Insight: Minimize Information Exchange With In-band Network Telemetry (INT)



Flows share the same bottleneck link.
Can we use the bottleneck link to exchange information?

Opportunity --- In-band Network Telemetry (INT)

- The link could report some basic information to the data packet
  - Queuing length / queuing delay
  - Link utilization / available bandwidth
  - …
- CSIG is Google's INT standard, released in 2024
  - Only the bottleneck hop's signal will be collected: max hop delay, max hop utilization
  - INT will be collected in the forwarding path, reflected through reverse path



Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

$$r_k = \frac{w_k}{w_1 + w_2 + \cdots + w_n} \cdot B$$

Goal

Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

Targets of allocation

Full utilization
Weighted Fair

1. Split into two equations

$$r_1 + r_2 + \cdots + r_n = B \qquad \frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

$$r_k = \frac{w_k}{w_1 + w_2 + \cdots + w_n} \cdot B$$

Goal

Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

Targets of allocation

Full utilization
Weighted Fair

$$r_1 + r_2 + \cdots + r_n = B \qquad \frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

$$r_k = \frac{w_k}{w_1 + w_2 + \cdots + w_n} \cdot B$$

Goal

Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

$$r_1 + r_2 + \cdots + r_n = B$$

$$\frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

1. Split into two equations

2. Find a signal for each equation

Consider a single link:
All flows on link are $\{f_1, f_2, \ldots, f_n\}$;
The weights are $\{w_1, w_2, \ldots, w_n\}$;
The rates are $\{r_1, r_2, \ldots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

2. Find a signal for each equation

$$r_1 + r_2 + \cdots + r_n = B$$

IF $r_1 + r_2 + \cdots + r_n > B$;
THEN queue increases
IF $r_1 + r_2 + \cdots + r_n < B$;
THEN queue decreases

IF queue **stabilizes** around any level
THEN $r_1 + r_2 + \cdots + r_n = B$

$$\frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

2. Find a signal for each equation

$$r_1 + r_2 + \cdots + r_n = B$$

IF $r_1 + r_2 + \cdots + r_n > B$;
THEN queue increases
IF $r_1 + r_2 + \cdots + r_n < B$;
THEN queue decreases

IF queue **stabilizes** around any level
THEN $r_1 + r_2 + \cdots + r_n = B$

$$\frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

How to reach the same $\frac{rate}{weight}$?

Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

2. Find a signal for each equation

$$r_1 + r_2 + \cdots + r_n = B$$

IF $r_1 + r_2 + \cdots + r_n > B$;
THEN queue increases
IF $r_1 + r_2 + \cdots + r_n < B$;
THEN queue decreases

IF queue **stabilizes** around **any level**
THEN $r_1 + r_2 + \cdots + r_n = B$

$$\frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

How to reach the same $\frac{rate}{weight}$?

Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

2. Find a signal for each equation

3. Give signal an extra meaning

$$r_1 + r_2 + \cdots + r_n = B$$

IF $r_1 + r_2 + \cdots + r_n > B$;
THEN queue increases
IF $r_1 + r_2 + \cdots + r_n < B$;
THEN queue decreases

IF queue **stabilizes** around **any level**
THEN $r_1 + r_2 + \cdots + r_n = B$

$$\frac{r_1}{w_1} = \frac{r_2}{w_2} = \cdots = \frac{r_n}{w_n}$$

How to reach the same $\frac{rate}{weight}$?

Create a mapping between:

$$queueing = T\left(\frac{rate}{weight}\right)$$

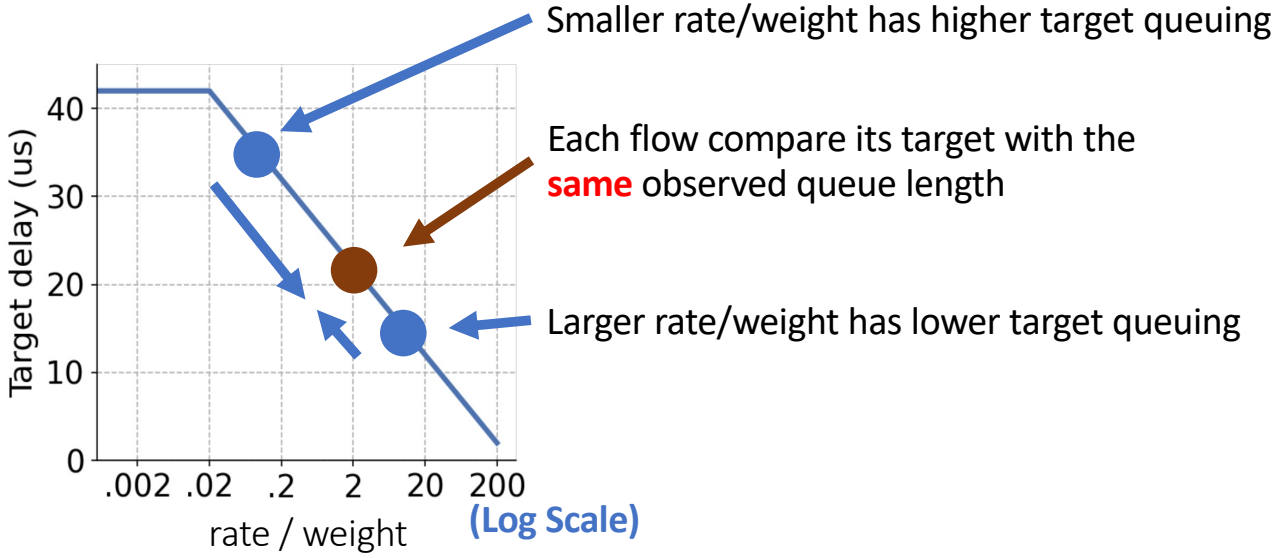
Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

2. Find a signal for each equation

3. Give signal an extra meaning

4. Create convergence algorithms



Design a mapping between rate/weight and queuing:

$$queuing = T\left(\frac{rate}{weight}\right) = p \cdot \frac{ln(200) - ln\left(\frac{r}{w}\right)}{ln(200) - ln(0.02)} + k$$

Any queuing indicates an anchor rate/weight:

$$\frac{rate}{weight} = T^{-1}(queuing)$$

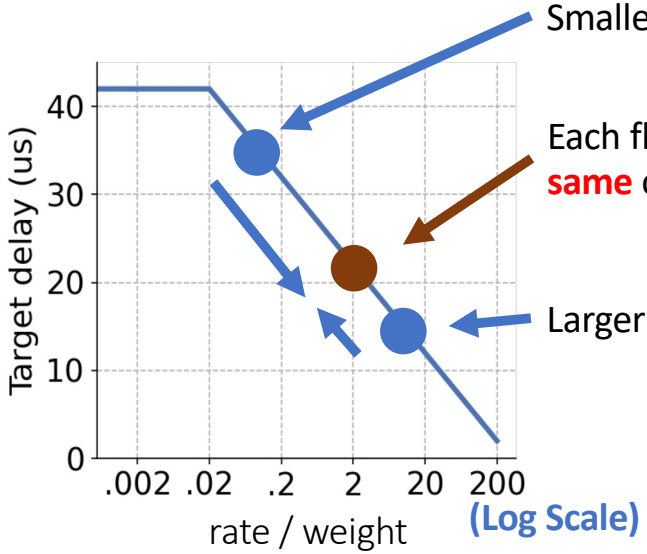
Consider a single link:
All flows on link are $\{f_1, f_2, \dots, f_n\}$;
The weights are $\{w_1, w_2, \dots, w_n\}$;
The rates are $\{r_1, r_2, \dots, r_n\}$;
Link bandwidth is B.

1. Split into two equations

2. Find a signal for each equation

3. Give signal an extra meaning

4. Create convergence algorithms



Smaller rate/weight has higher target queuing

Each flow compare its target with the **same** observed queue length

Larger rate/weight has lower target queuing

Converge all the flows' rate/weight to anchor:

$$new\_rate = rate \cdot \left(\frac{T^{-1}(queuing)}{rate/weight}\right)^{m}$$

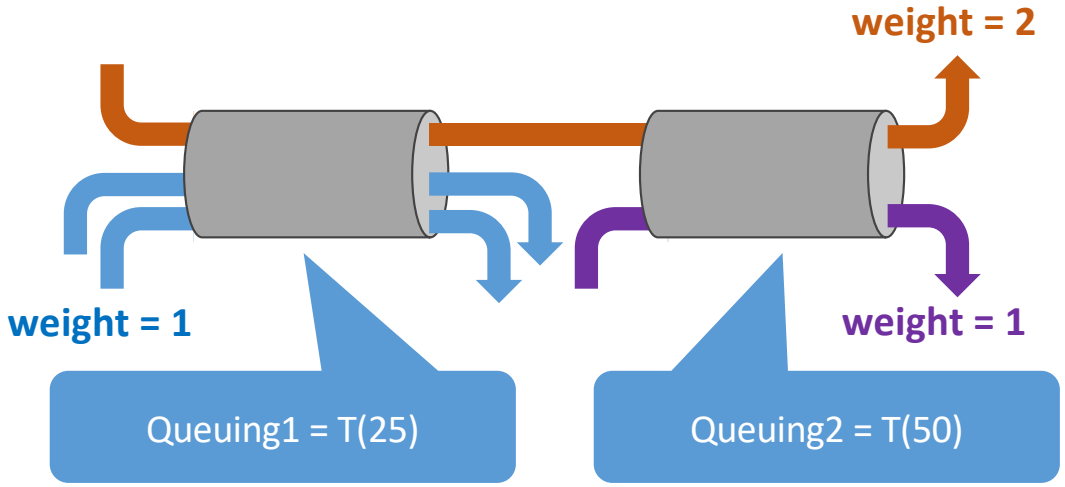
For Arbitrary Networks, Convergence Can Be Proved

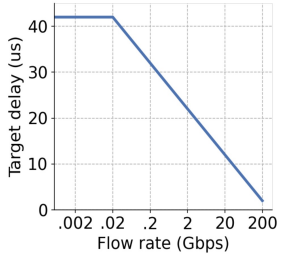
Lemma 3.3: Bottleneck Hop Properties

When achieving weighted max-min fair, each flow will have the largest rate per weight among all flows on its bottleneck hop and not on any other saturated hop.

Theorem 3.1: Weighted Max-min Fairness

For every flow in an arbitrary network, Söze converges to a weighted max-min fair allocation, if and only if $0 < m < 2$ in the update function and $p > \frac{\Delta t}{2} \cdot [\ln(\alpha) - \ln(\beta)]$ in the target function.





T(25) > T(50)

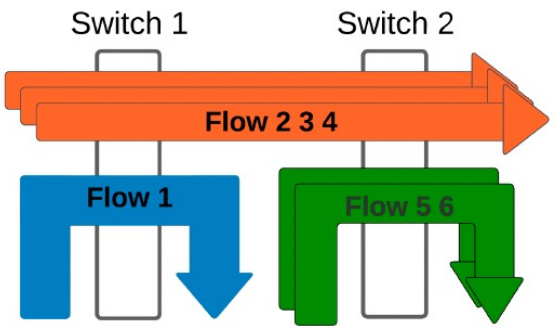
Thus, queuing1 is the bottleneck for orange

The link with the **largest** queuing delay is the bottleneck.

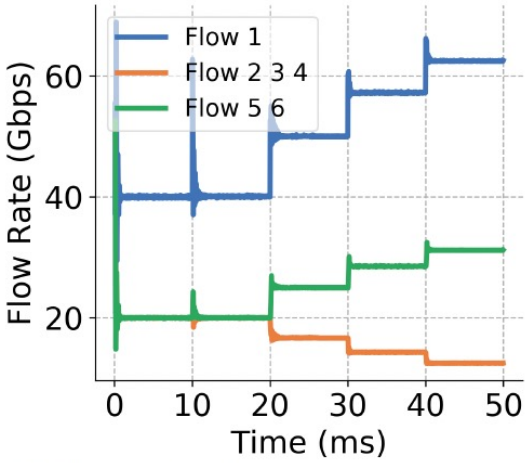
Implementation

- Linux kernel module
  - Implementation
    - 90 lines of kernel module code
    - Use socket parameter as interface between application and network stack
    - Application could change socket config to change priority
  - Topology
    - 10 hosts + 1 Tofino switches[3]
- Kernel-bypass Implementation - eRPC
  - Modified packet format for INT-signal
  - Provide inherent application interface for changing priority
- NS-3 simulator implementation

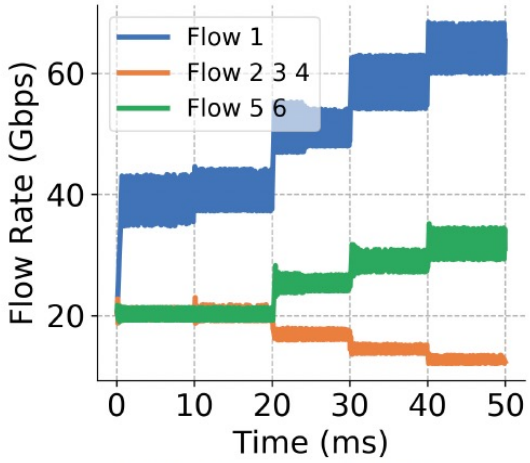
Evaluation: Changing weight



(a) Experiment scenario.



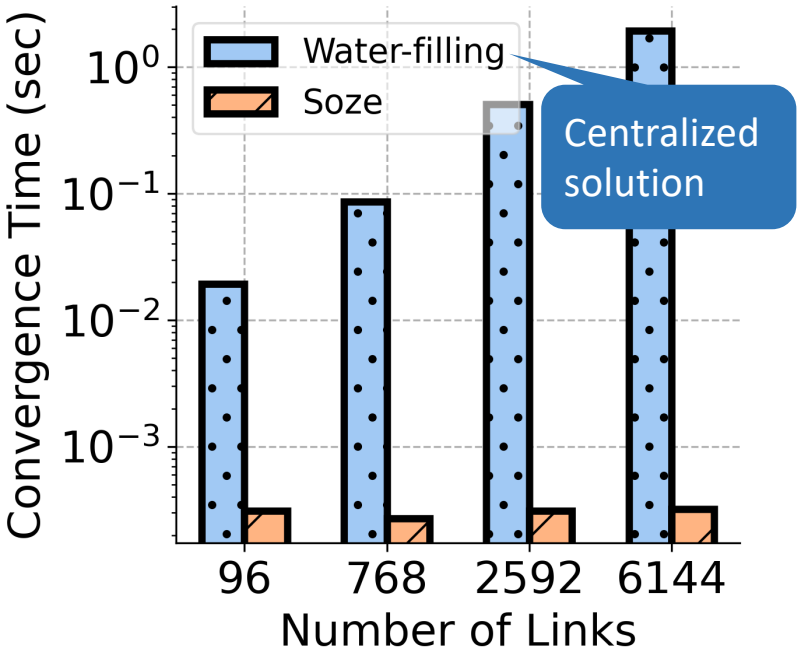
(b) Söze with increasing weight.

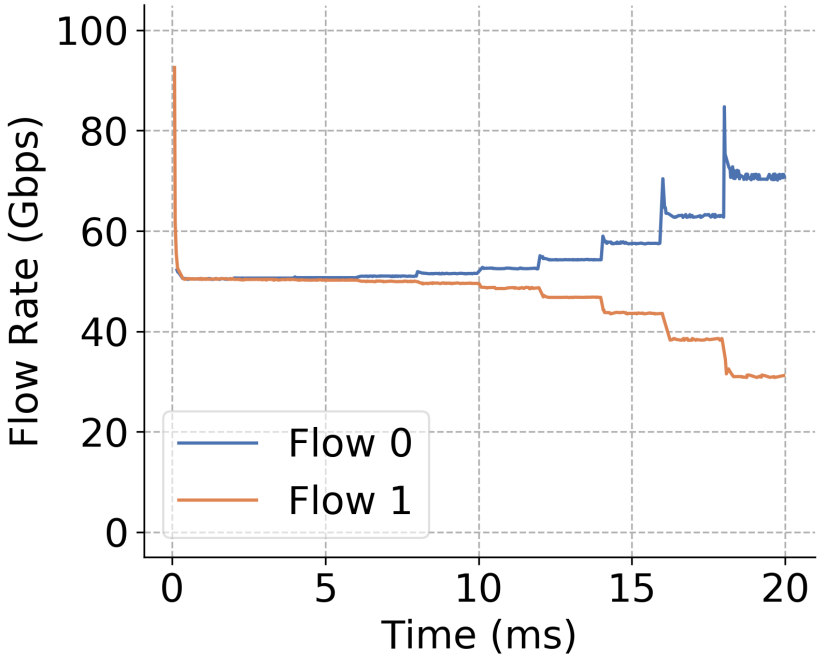


(c) WRR with AIMD.

Soze is accurate and stable

Evaluation: Agility and Granularity

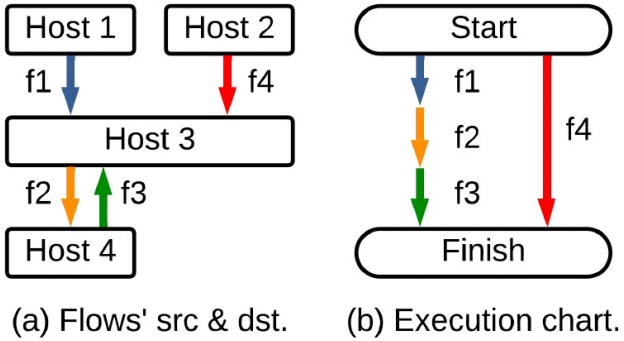




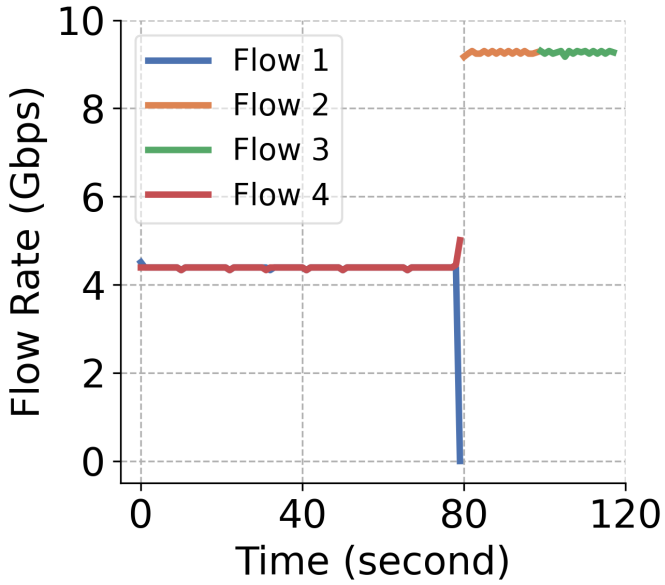
With different fat-tree topology size, Soze achieves the same convergence speed.

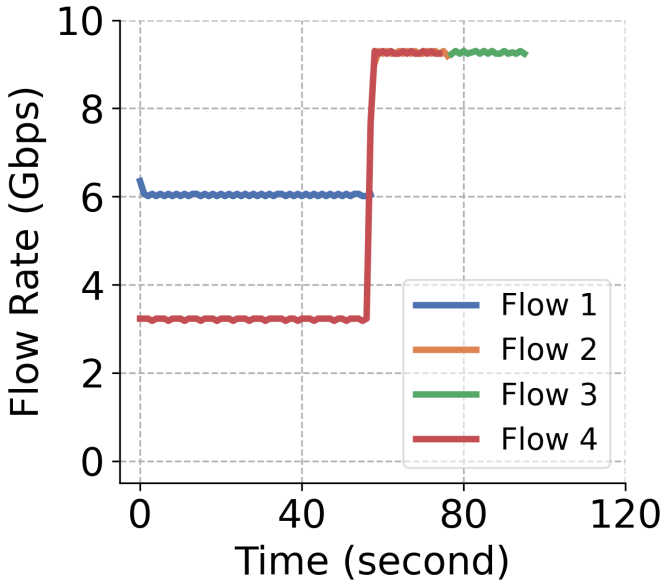
With the changing weights, Soze could enforce weight accurately.

Evaluation: Critical-path Prioritization



(a) Flows' src & dst. (b) Execution chart.





Experiment setup | **Fair allocation** | **Weighted allocation**

Please check out our OSDI paper for many more details and results in a few weeks!

Closing the circuit -- How does Soze relate to reconfigurable networks?

- Reconfigurable networks can have sudden and massive change in end-to-end performance characteristics
- INT/Soze can potentially provide a powerful and general framework for end-to-end adaptation in reconfigurable networks
  - What INT(s) is optimal for reconfigurable networks?
  - How to overcome circuit down-time that disrupts the delivery of INT?
  - …
- Looking forward to discussions