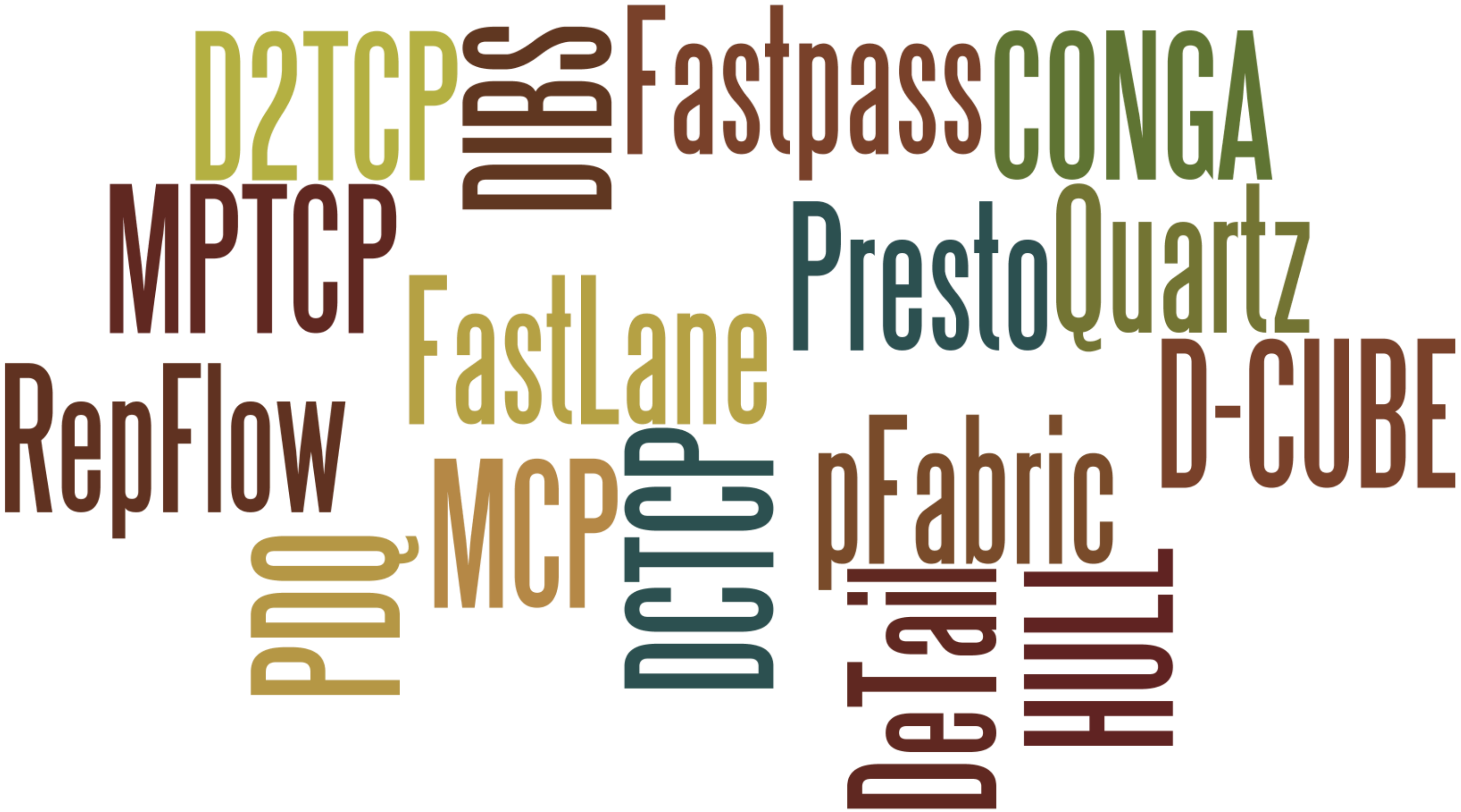
NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters

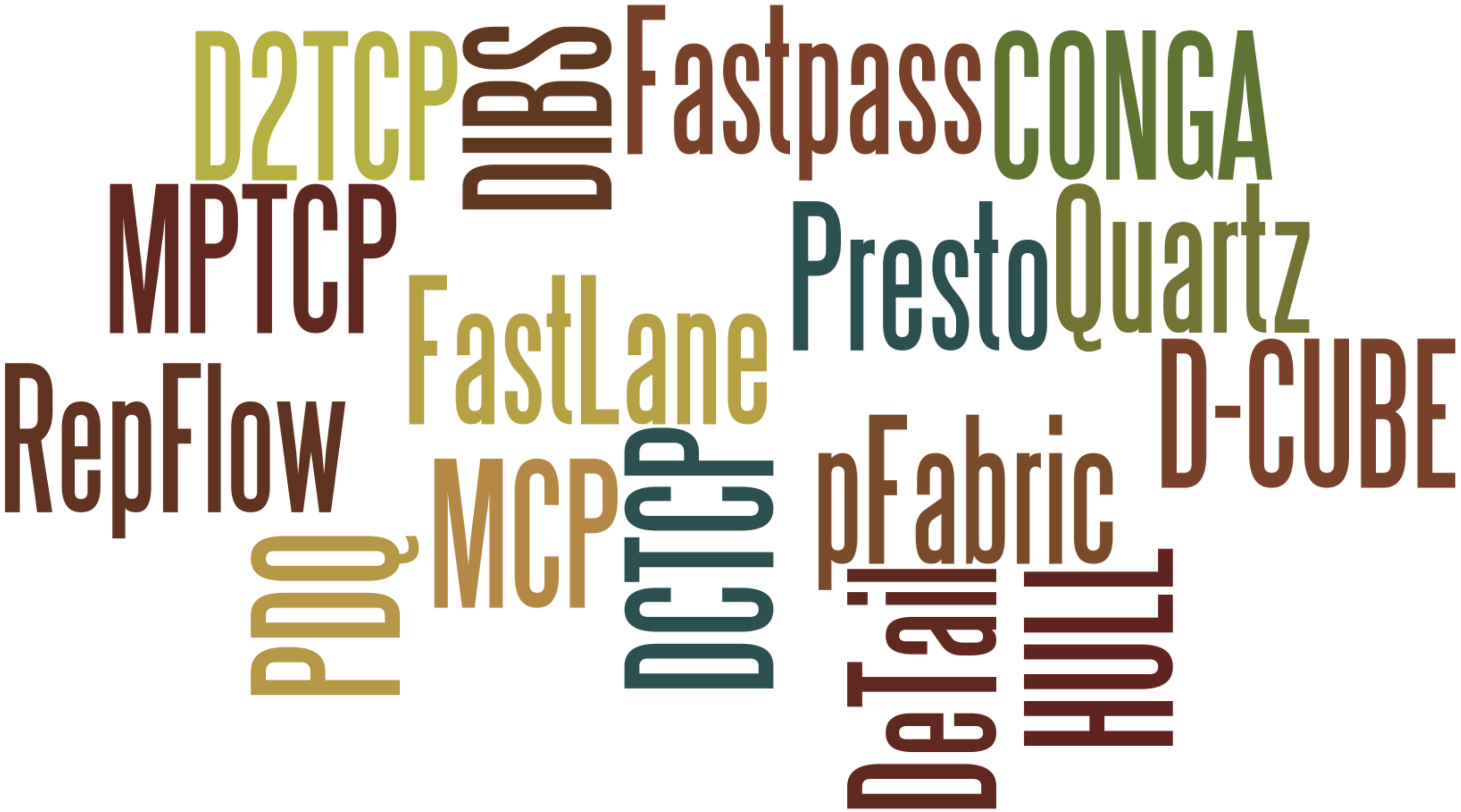
Kanthi Nagaraj (Stanford), Dinesh Bharadia(M.I.T.), Mohammad Alizadeh (M.I.T.), Hongzi Mao (M.I.T.), Sandeep Chinchali (Stanford) and Sachin Katti(Stanford)

Sigcomm 2016

Datacenter fabric proposals



Which one does the operator pick?



Is there a single fabric that provides flexible and fast bandwidth allocation control?

Yes ! NUMFabric provides a flexible fabric that is also fast.

Flexible and Fast

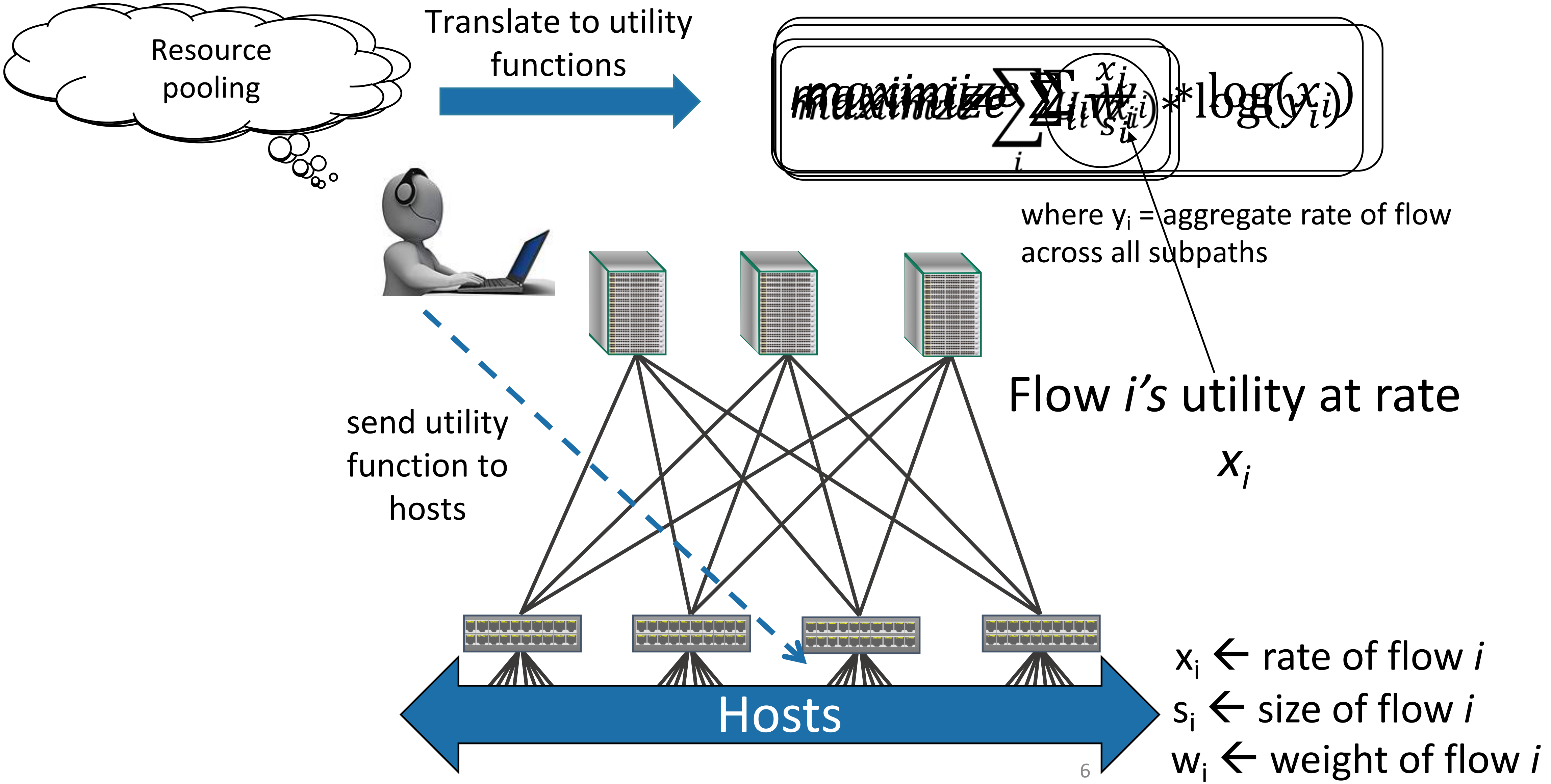
Flexible

- Supports wide variety of bandwidth allocation objectives

Fast

- Flows converge to correct rates before the datacenter workload changes

NUMFabric: Flexibility



Network Utility Maximization in general

$$maximize\, U(X) = \sum_{i \epsilon S} U_i(x_i)$$

$$subject\ to$$

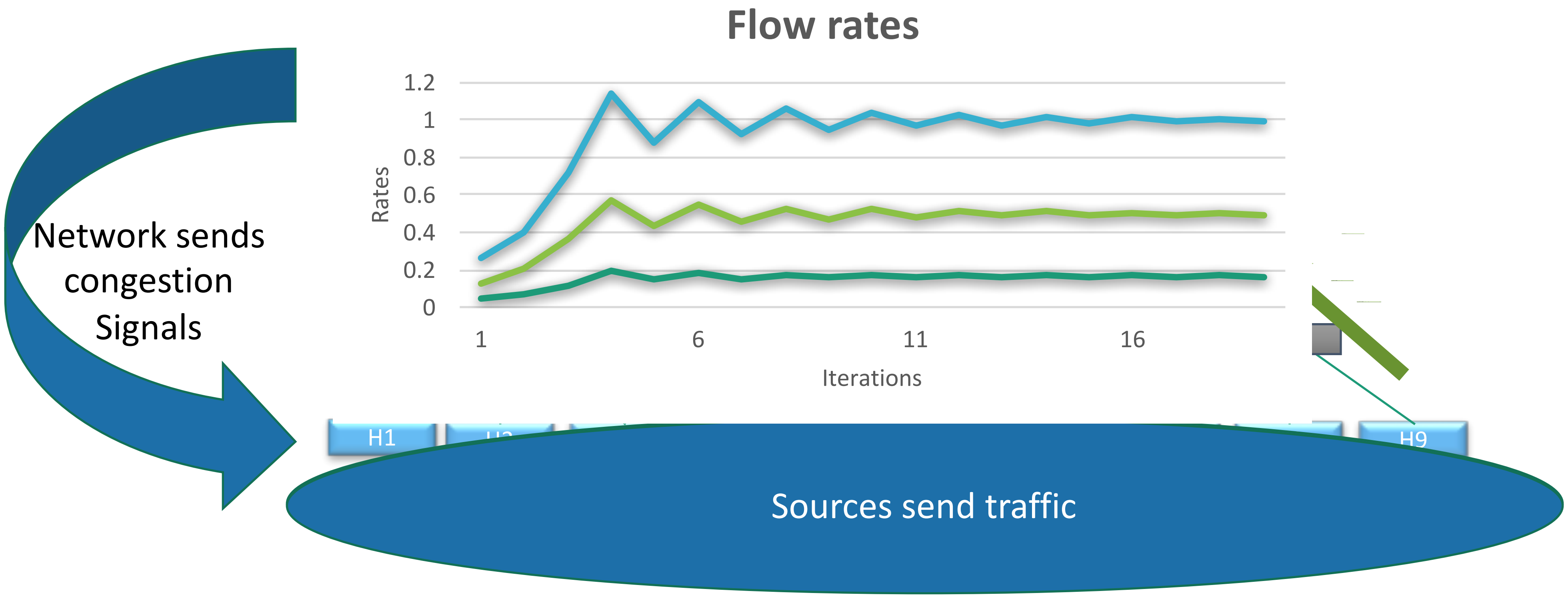
$$AX \leq C$$

$$X \geq 0$$

Problem ?

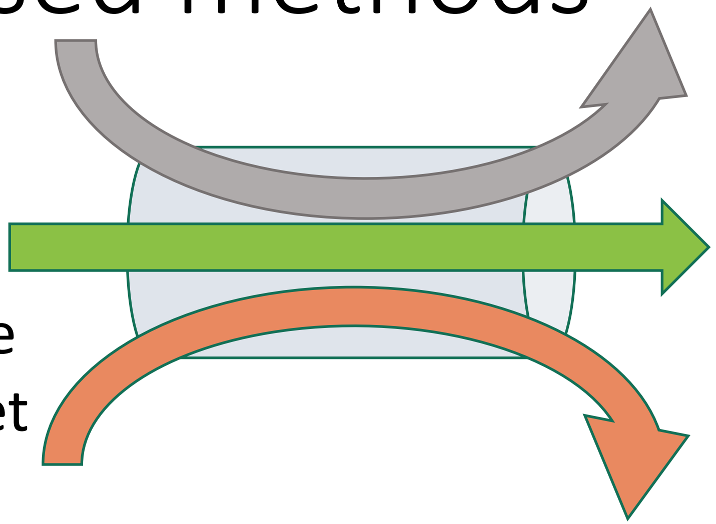
Existing NUM solutions are slow and unsuitable for data center workloads

Existing distributed NUM solutions

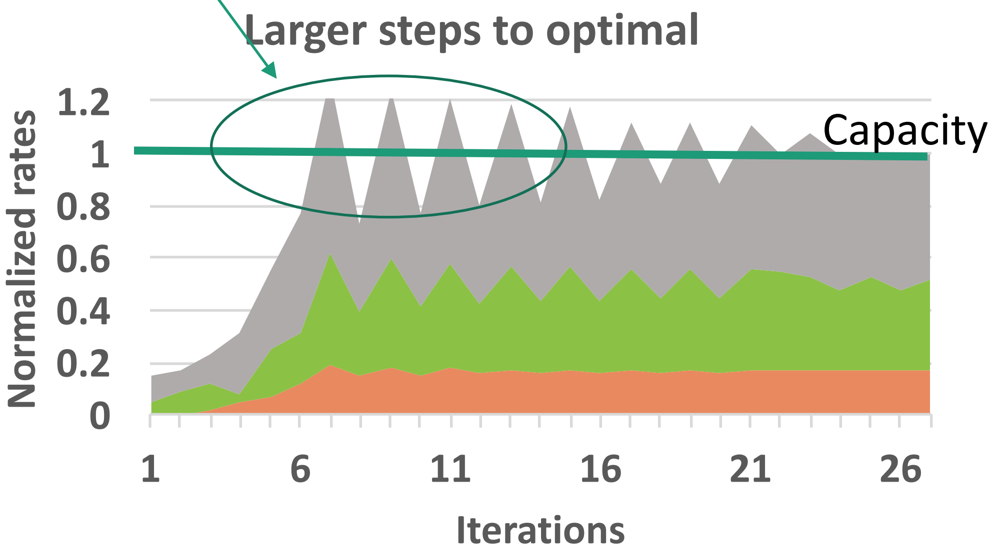


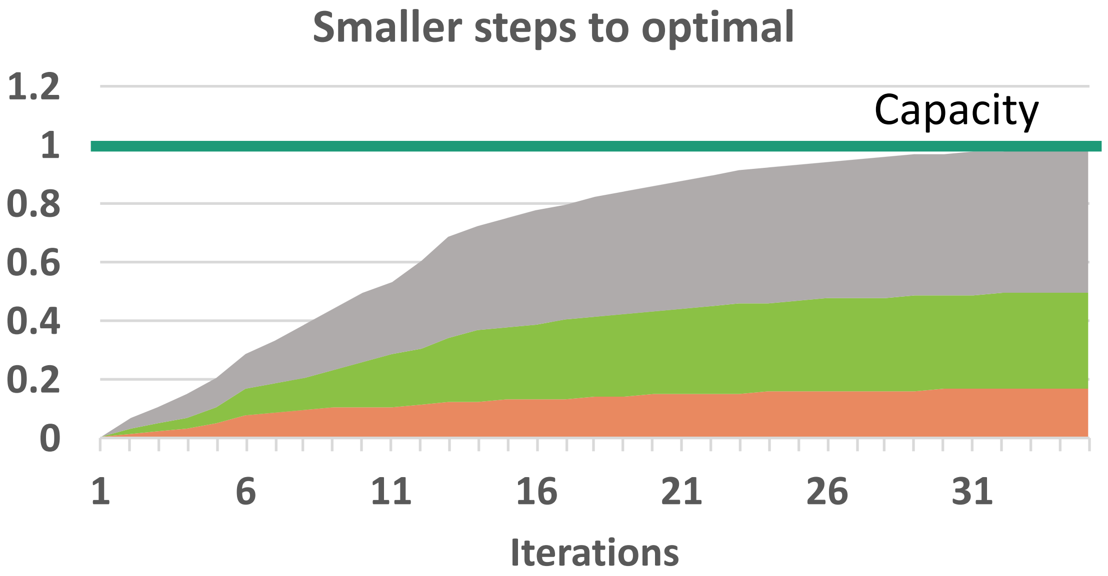
- Each source iteratively adjusts rates following its own gradient towards optimal
- The sum of the rates moves towards the global optimal

Gradient based methods



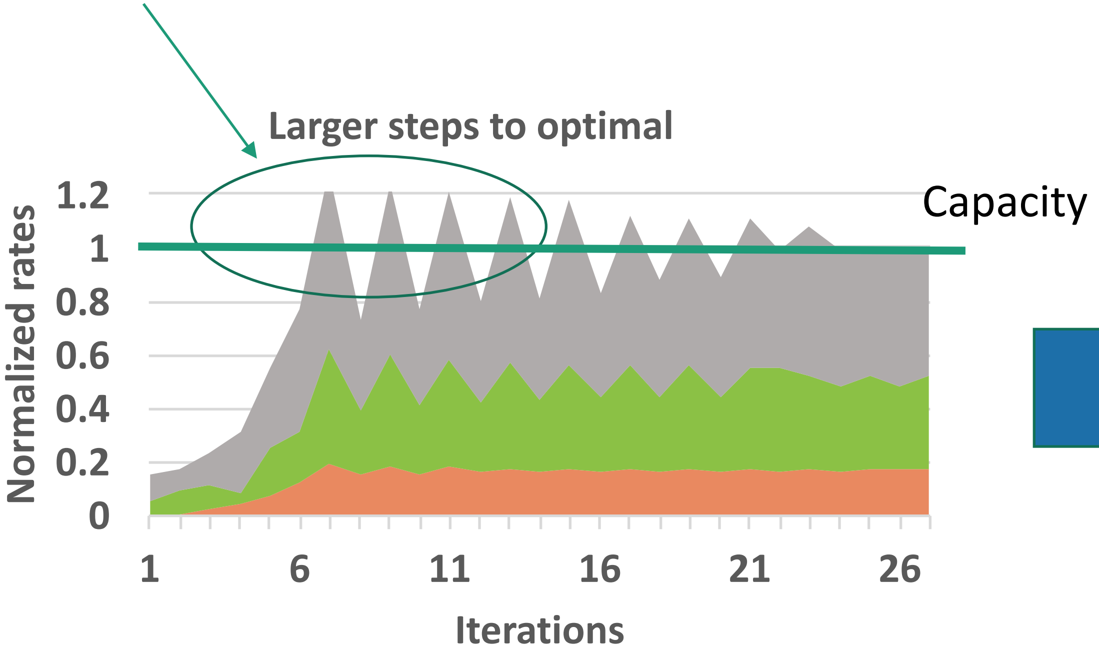
Overshooting might cause bloated queues and packet drops

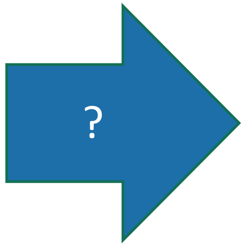




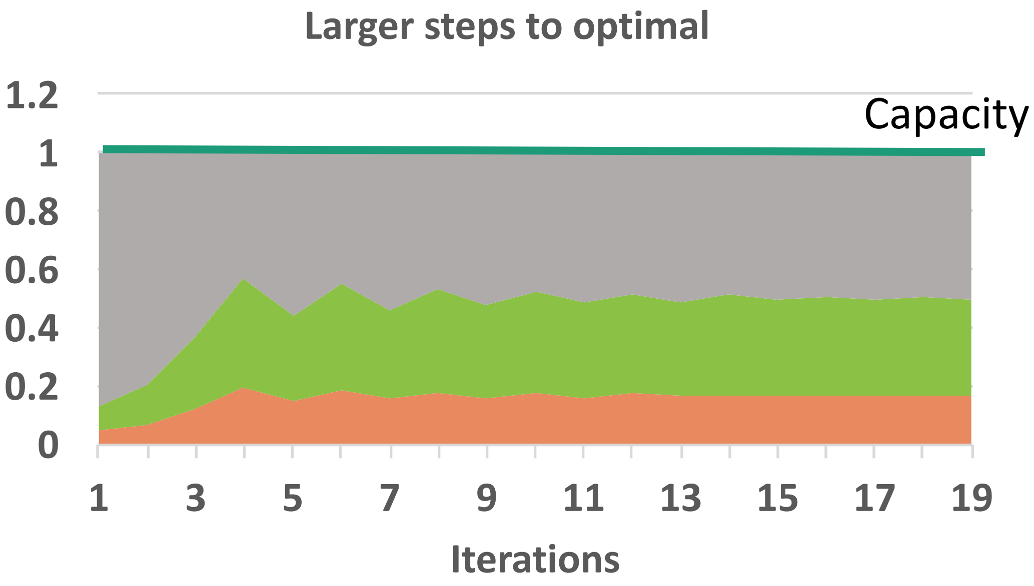
How can we fix this?

Overshooting might cause drops, queues bloating





Can we enable larger steps to optimal but without over-shooting and under-utilization?

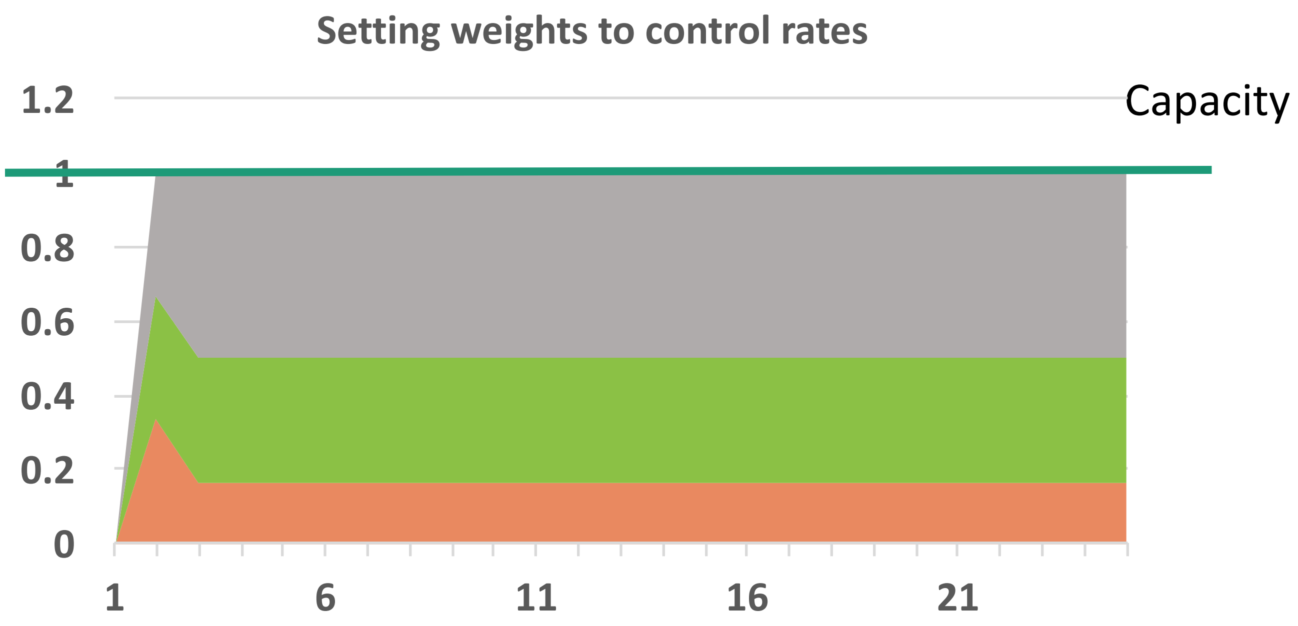


Use Weights instead of rates !

Setting weights of the flow and allowing a fabric to allocate rates proportional to the weights enables exactly this.

NUMFabric key idea

- In NUMFabric, sources give up direct control over rates
- The sources specify “weights” and the Weighted Max-Min fabric allocates relative rates proportional to the weights of all flows



Application level objective

Translate to utility functions

$$maximize \sum_i U_i(x_i)$$

Flexible

Weights

Layer that sets weights of flows based on network feedback

Weighted Max-Min rate allocation according to the weights

Network feedback

Fast

Weight inference

Distributed NUM mechanism

NUM Objective

$$Maximize \sum_i U_i(x_i)$$

KKT Conditions: Equations that must necessarily be true at optimal solution

Price of a link : variable that indicates the congestion level at the switch

$$p_l\left(\sum_{i \in S(l)} x_i - c_l\right) = 0$$

At optimal, either the link is fully utilized or the price of the link is zero

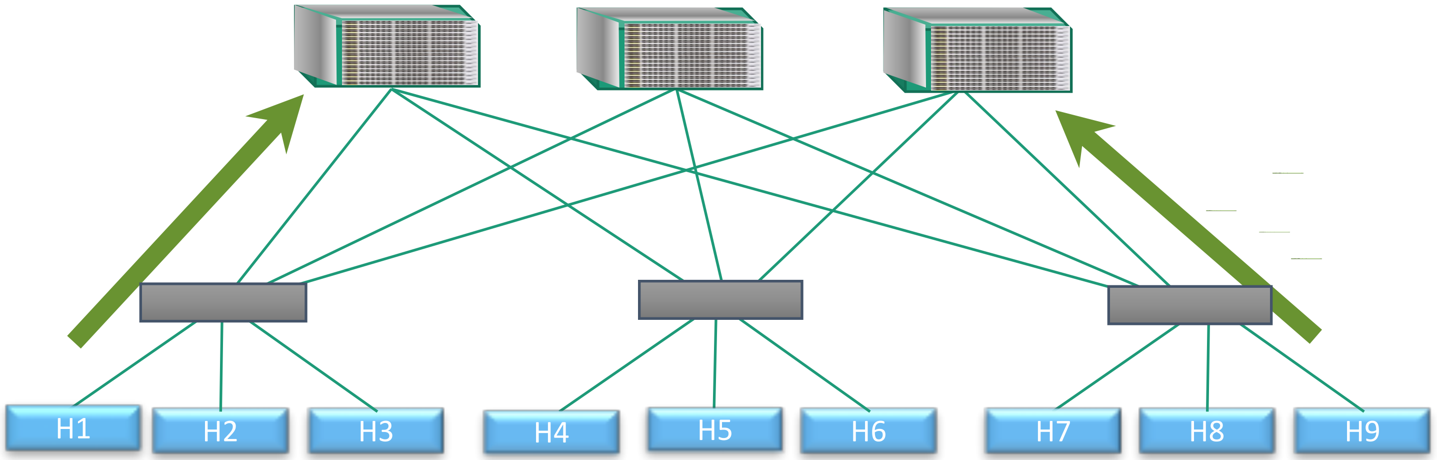
$$U_i'(x_i) = \sum_{l \in L(i)} p_l$$

At optimal, the marginal utility of the source is equal to the sum of the prices along the path of the flow

Distributed NUM mechanism

Switches set their prices measuring congestion

$p_l(\sum_{i \in S(l)} x_i - c_l) = 0$ ← solve ← $p_l = p_l + \alpha * (\sum_{i \in S(l)} x_i - c_l)$



$\sum_{l \in L(i)} p_l$ = Network congestion signals

Sources adapt rates of flows

Sources set the rates of the flows using price feedback

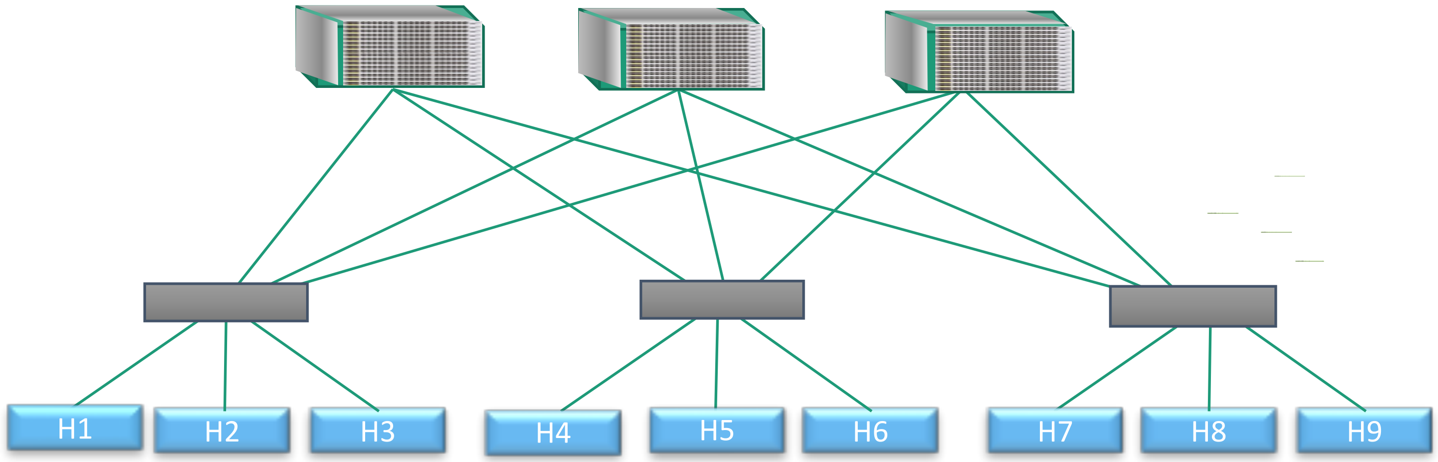
$U_i'(x_i) = \sum_{l \in L(i)} p_l$ ← solve ← $x_i = Inverse\ of\ U_i'\ (\sum_{l \in L(i)} p_l)$

NUMFabric iterations

Controlling rates directly causes the brittleness in the existing solutions.

$$p_l\left(\sum_{i \in S(l)} x_i - c_l\right) = 0$$ ✔

Switches adapt prices at every iteration so that the flow rates move closer to optimal



WMM layer always achieves 100% link utilization

$$U_i'(x_i) = \sum_{l \in L(i)} p_l$$ ✘

$$w_i = Inverse\ of\ U_i'\left(\sum_{l \in L(i)} p_l\right)$$

WMM layer converts these weights to rates

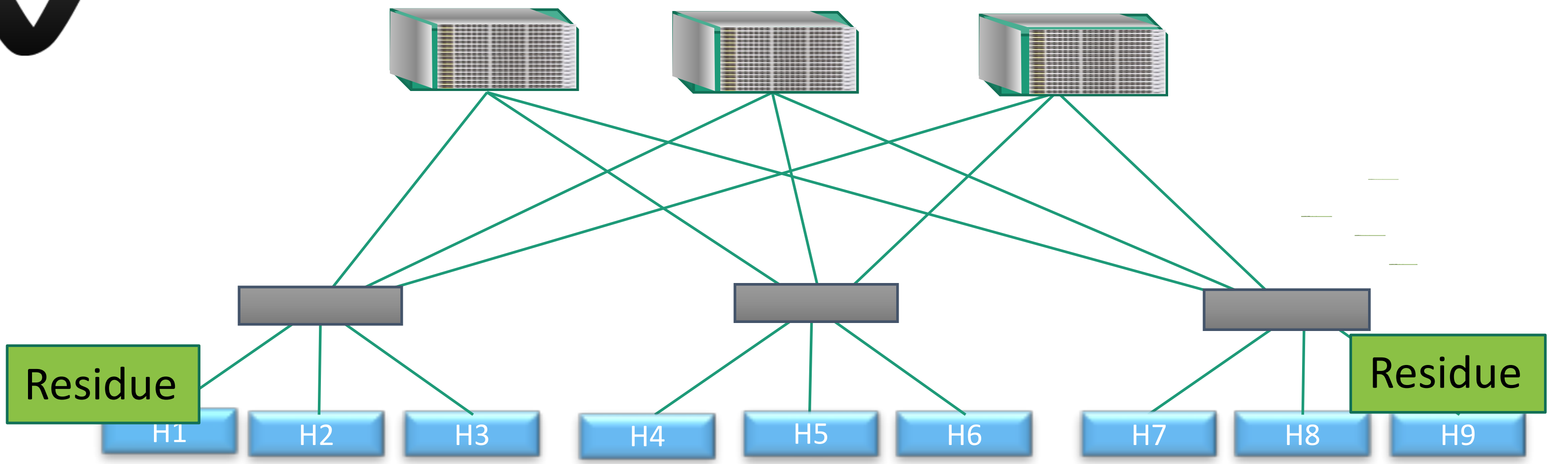
NUMFabric iterations

As we know, controlling rates directly causes the brittleness in the existing solutions.

Swit hat the al

$$p_l = p_l + \min\left(\frac{residue_i}{hops\ traversed\ by\ flow_i}\right)$$

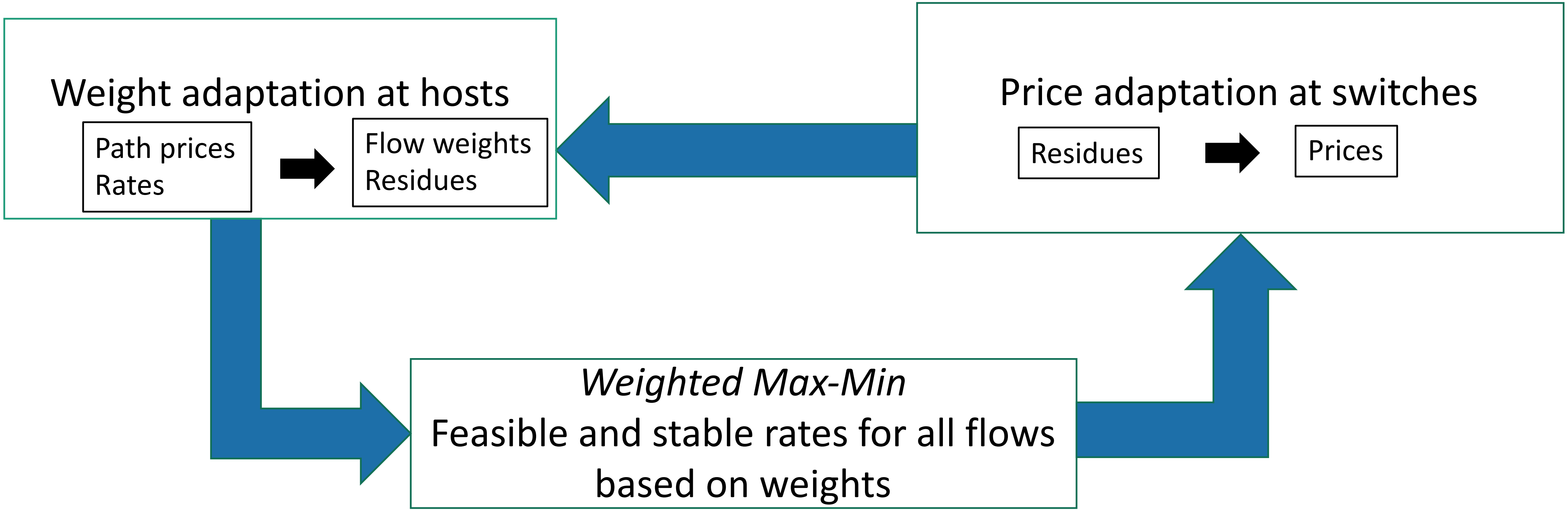
$$p_l\left(\textstyle\sum_{i \in S(l)} x_i - c_l\right) = 0$$



$$U_i'(x_i) = \sum_{l \in L(i)} p_l$$

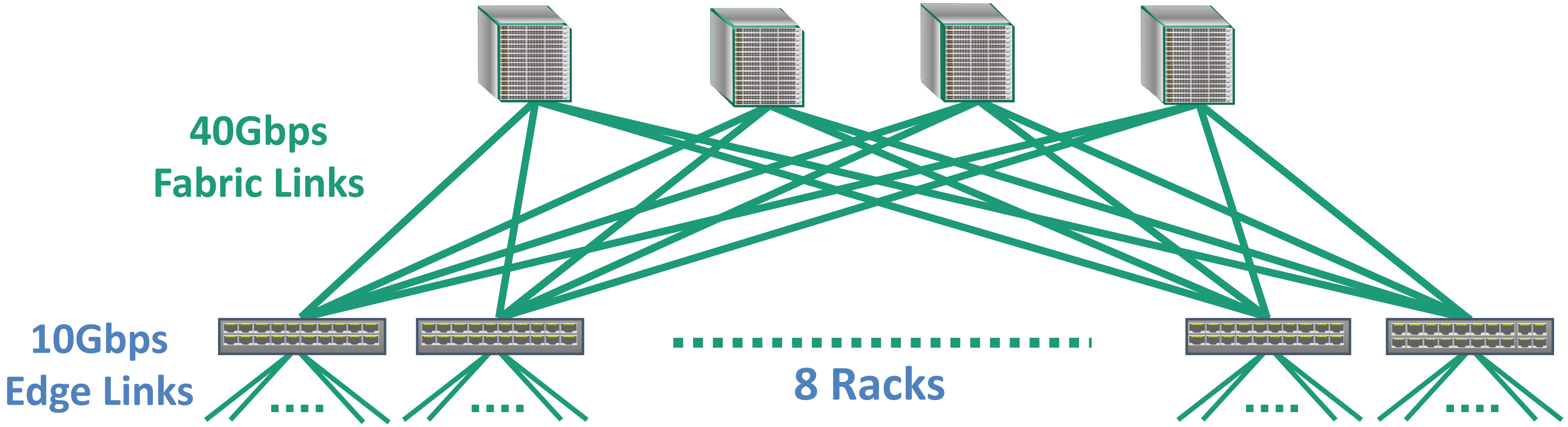
$$Residue = U_i'(x_i) - \sum_{l \in L(i)} p_l$$

Operation summary



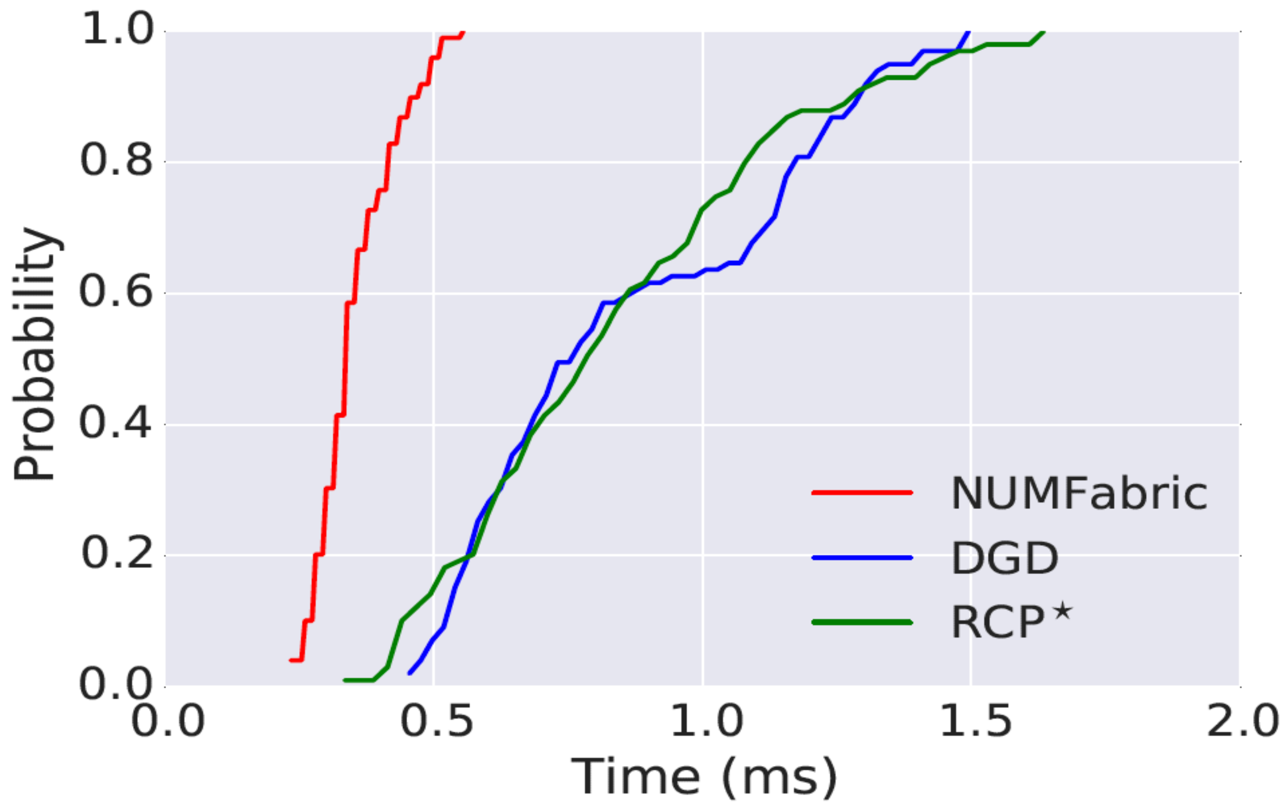
Evaluation

Evaluation setup



- ns3 simulations: 128-port leaf-spine fabric
  - RTT = ~16μs
- Evaluate speed of convergence
- Evaluate flexibility
  - Compare the bandwidth allocations on NUMFabric with different utility functions against point solutions for different objectives– pFabric, MPTCP, etc.

Fast convergence

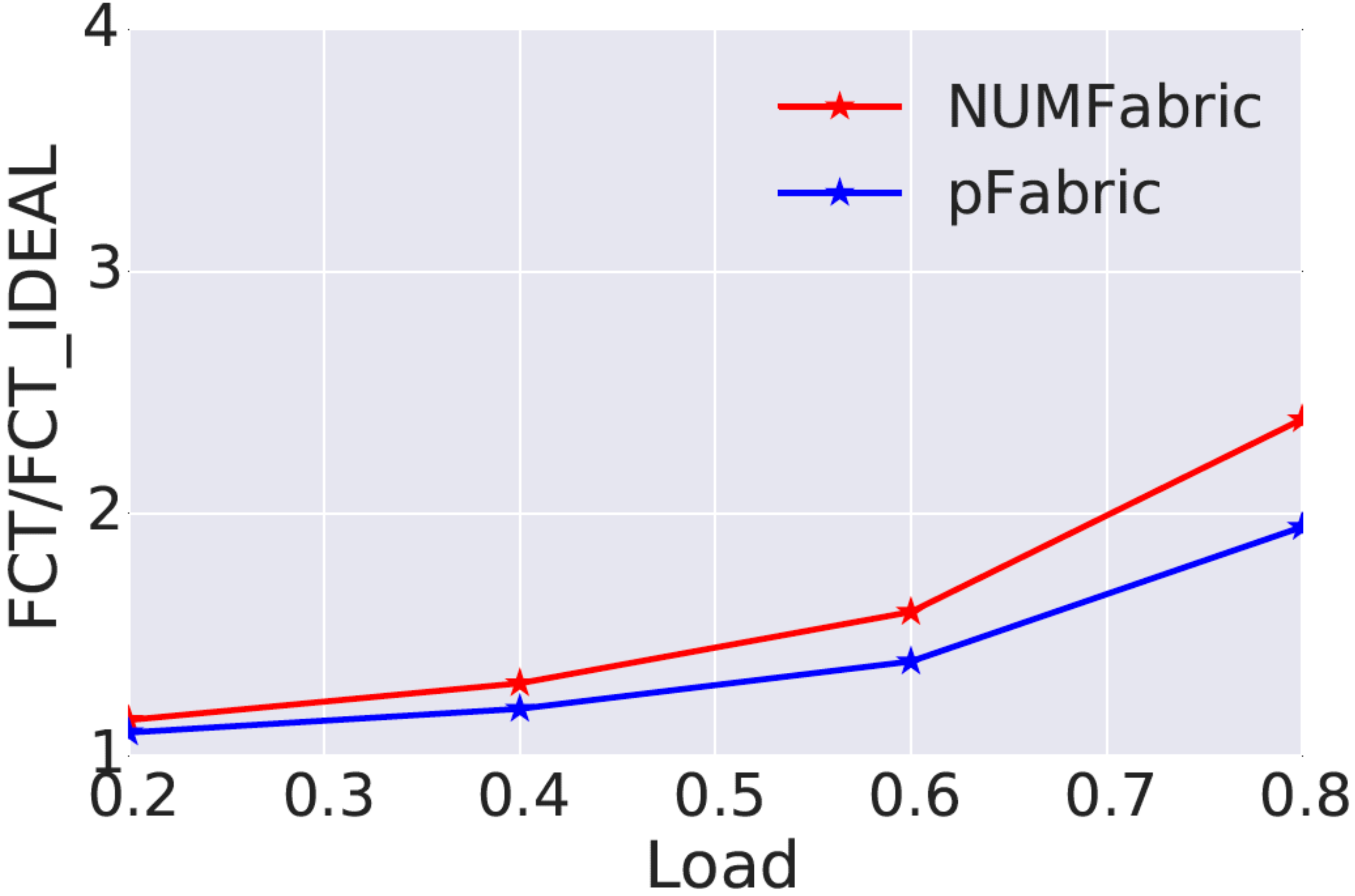


DGD : Dual Gradient Descent algorithm

RCP* : Alpha-Fair RCP

- 100 flows start/stop at every “event”.
- We let the system converge before triggering another event
- Median convergence time (335 us) of NUMFabric is 2.3X better that the other algorithms

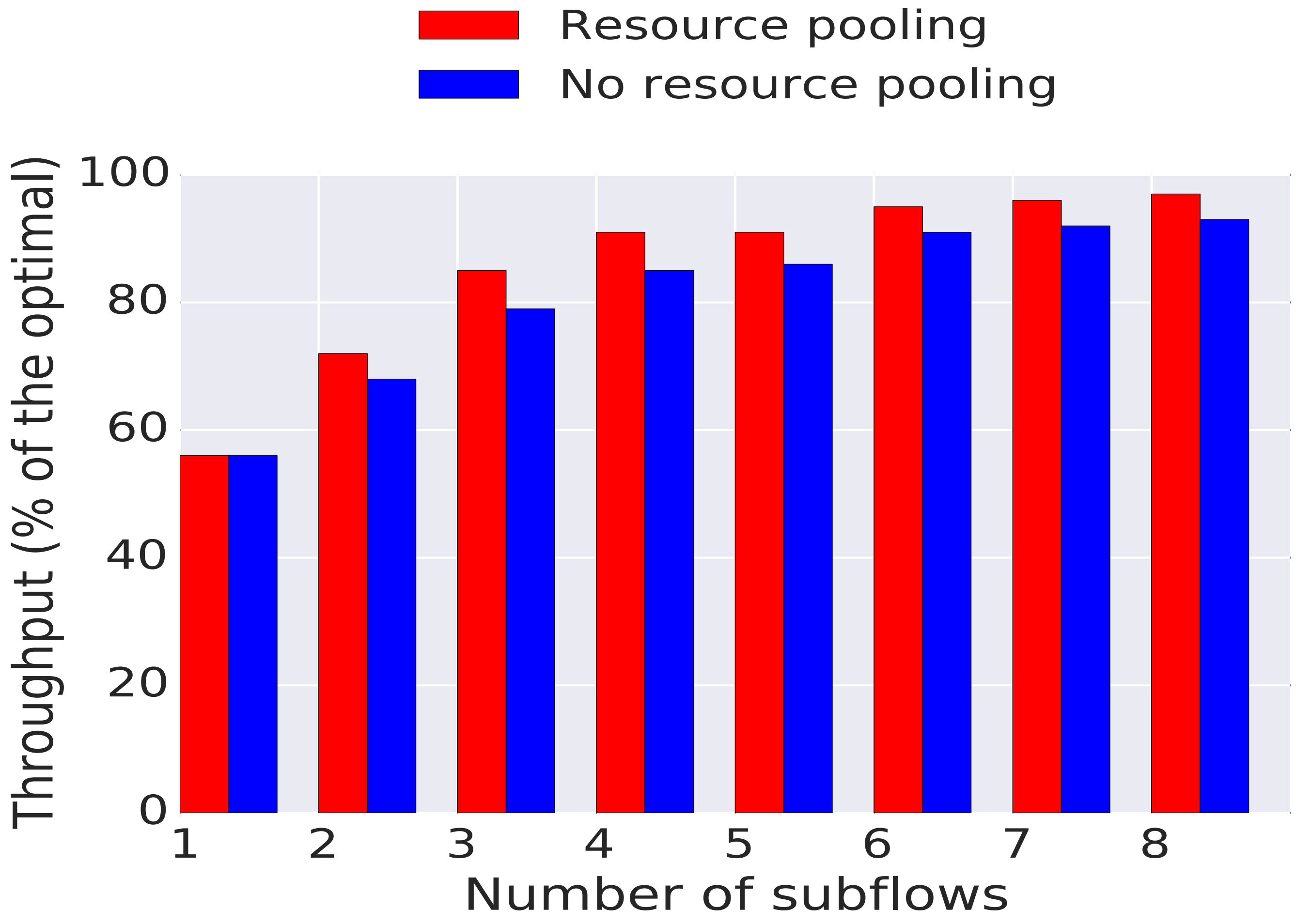
Flexibility : minimize flow completion times



$$maximize \sum_i \frac{x_i}{s_i}$$

x_i → rate of the flow
s_i → size of the flow

Flexibility : minimize flow completion times



$$maximize \ \sum_i \log(y_i)$$

where y_i = aggregate rate of flow across all sub-paths

Conclusions

- NUMFabric enables operators to flexibly optimize network's bandwidth allocation for different bandwidth allocation objectives
- NUMFabric uses weights as knobs to influence rates and thus, decouples the objectives of finding optimal rates and stable rates.This makes it 2-3X faster existing mechanisms.
- Using NUMFabric with objective functions on co-flows, VM-level and tenant-level aggregates is focus of our current and future work.

Thank you