Hierarchical Fair Scheduling: A Reality Check

Natchanon Luangsomboon
Jörg Liebeherr

University of Toronto

(also joint work with Almut Burchard and Sahana Radhaharan)

WoNeCa 2020

Motivation

- Interest in hierarchical bandwidth allocation for data centers

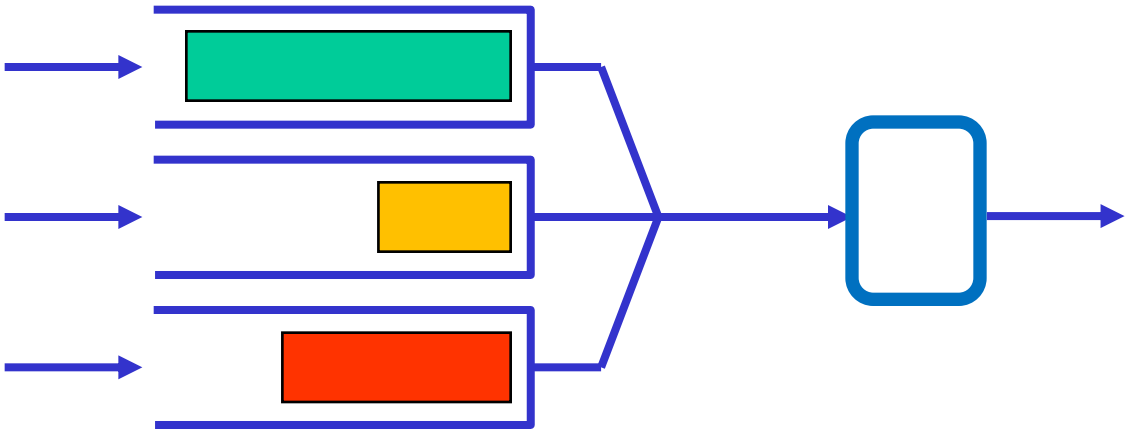
- Available hierarchical packet scheduling algorithms in Linux kernel found to be inadequate:
  - Class Based Queuing (CBQ) [Floyd/ Jacobson `93]
  - Hierarchical Token Bucket (HTB) [devik `03]
  - Hierarchical Fair Service Curve (HFSC) [Stoica/Zhang/Ng `97]

- A scheduler with desirable (provable) properties is available but not implemented:
  - Hierarchical Packet Fair Queuing (HPFQ) [Bennett/Zhang `97]

Fair Queuing

Goal: Realize a scheduler that simultaneously serves all backlogged flows at the same rate

Processor Sharing (PS): Bit-by-bit (fluid flow) round robin



- Realizes **max-min fairness**

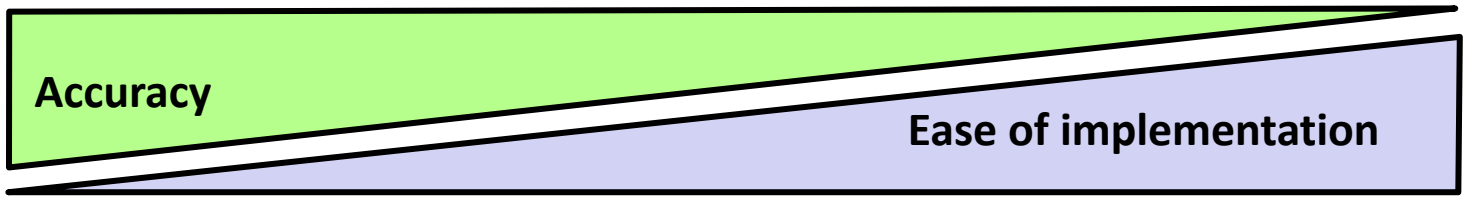
Weighted Fair Queuing

Goal: Serve each backlogged class at a rate proportional to its weight

Generalized Processor Sharing (GPS):
Weighted version of bit-by-bit (fluid flow) round robin

Many packet level implementations:

WFQ/PGPS	**SCFQ**	**DRR**	available in Linux
WF^{2}Q	**SFQ**	**WRR**	



Weighted Max-Min Fair Allocation

$\mathcal{N}$	set of classes
C	Link capacity
x_j	requested rate by class j
y_j	allocated rate by class j $(y_j \leq x_j)$
ϕ_j	weight of class j

Weighted Max-Min Fair Allocation

1. If $y_i < x_i$ then $\frac{y_i}{\phi_i} \geq \frac{y_j}{\phi_j}$ for all $j \in \mathcal{N}$,
2. $\sum_{j \in \mathcal{N}} y_j = \min(\sum_{j \in \mathcal{N}} x_j, C)$.

Weighted Max-Min Fair Allocation

The allocation to class i is

$$y_i = \min\left(x_i, \phi_i \, \mathbf{f_i}\right)$$

with

$$\mathbf{f_i} = \max_{M \subseteq \mathcal{N} \setminus \{i\}} \frac{1}{\sum_{j \notin M} \phi_j} \left(C - \sum_{j \in M} x_j\right)$$

$\mathbf{f_i}$ are uniquely determined

GPS Service Curve

(Burchard/Liebeherr '18)

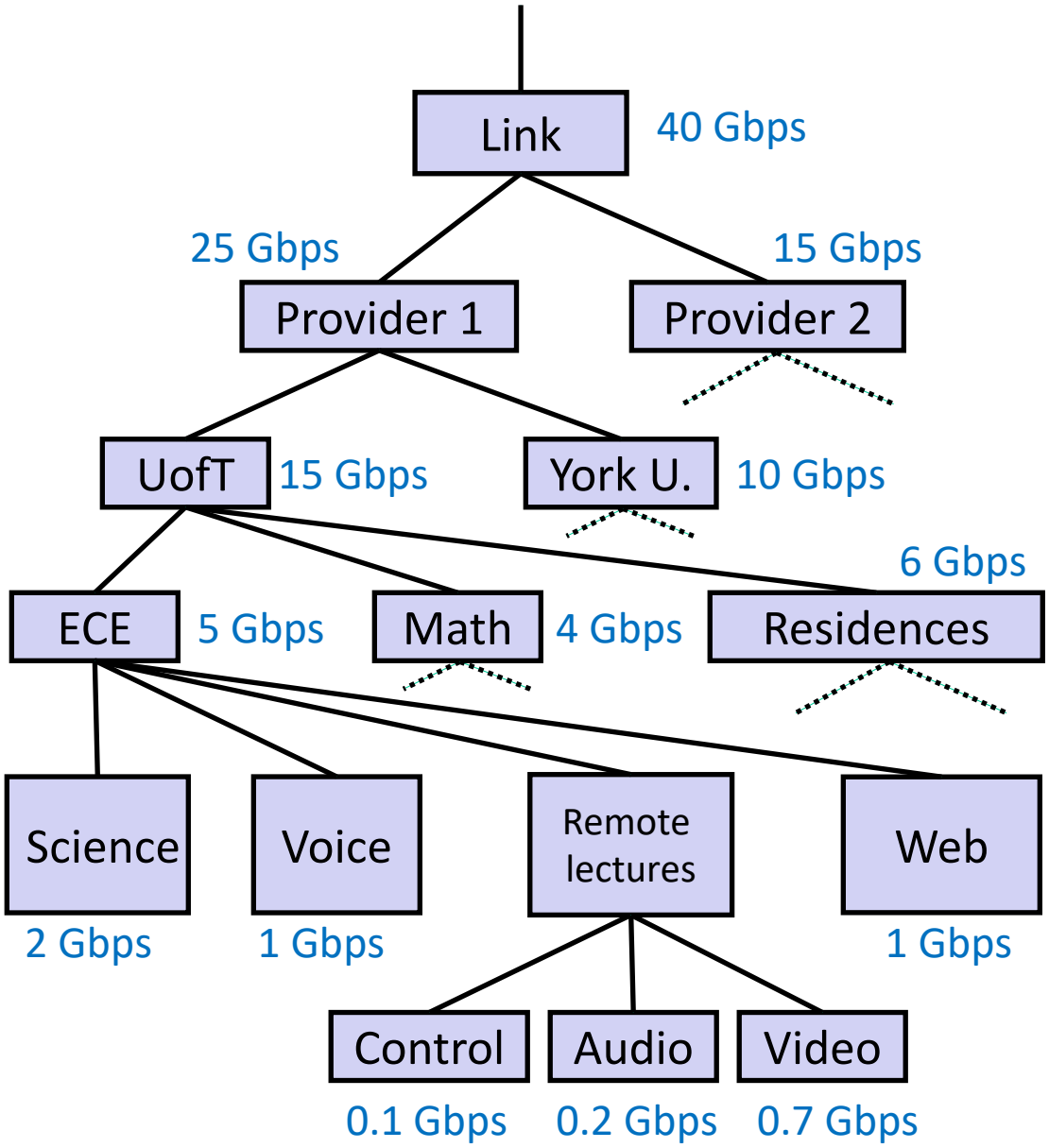
GPS extends max-min fair allocation to time-variable traffic

$A_j(s,t)$	arrivals in $[s,t)$ from class j
$\mathcal{E}_j(t)$	envelope of class j (concave), i.e., $\mathcal{E}_j(t-s) \geq A_j(s,t)$ for all s,t
$\mathcal{C}(t)$	strict service curve of link (convex)

Best possible strict service curve for class i

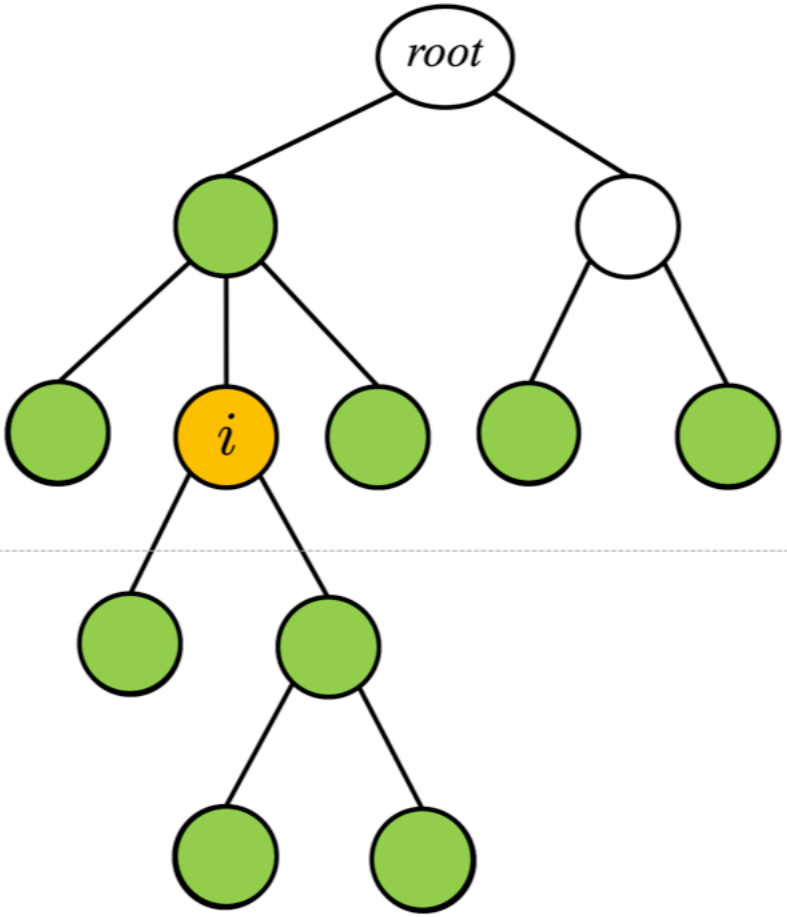
$$\mathcal{S}_i(t) := \max_{M \subseteq \mathcal{N} \setminus \{i\}} \frac{\phi_i}{\sum_{j \notin M} \phi_j} \left(\mathcal{C}(t) - \sum_{j \in M} \mathcal{E}_j(t) \right)$$

Hierarchical Link Sharing



- Resource sharing at multiple levels of aggregation
- Link-Sharing
  - Service to a class should be similar to a dedicated link (with varying capacity)
  - Excess capacity should be shared between sibling classes
- Aggregation levels in Google's BwE B4 (Sigcomm`15):
  - Task flow group
  - Job flow group
  - User flow group
  - Cluster flow group
  - Site flow group

Definitions



Leaf classes

Child classes	$\text{child}(i)$
Parent class	$p(i)$
Descendants	$\text{desc}(i)$
Leaf descendants	$\text{ldesc}(i)$
Sibling	$\text{sib}(i)$
Ancestors	$\text{anc}(i)$

- All traffic arrives at leaf classes

Hierarchical Weighted Max-Min Fair Allocation

$\mathcal{N}$	set of classes
C	Link capacity
x_j	requested rate by class j
y_j	allocated rate by class j $(y_j \leq x_j)$
ϕ_j	weight of class j

Non-leaf class: $x_i = \sum_{j \in \text{child}(i)} x_j, \quad y_i = \sum_{j \in \text{child}(i)} y_j$

Hierarchical Weighted Max-Min Fair Allocation

1. If $y_i < x_i$ then $\frac{y_i}{\phi_i} \geq \frac{y_j}{\phi_j}$ for all $j \in \text{sib}(i)$
2. $y_{root} = \min(x_{root}, C)$

Hierarchical Weighted Max-Min Fair Allocation

The allocation to class i is

$$y_i = \begin{cases} \min\left(x_i, C\right), & i = root \\ \min\left(x_i, \phi_i \, \mathbf{f_i}\right), & \text{otherwise} \end{cases}$$

with

$$\mathbf{f_i} = \max_{M \subseteq \text{sib}(i)} \frac{1}{\sum_{j \notin M} \phi_j} \left(y_{p(i)} - \sum_{j \in M} \sum_{k \in \text{ldesc}(j)} x_k \right)$$

- $\mathbf{f_i}$ are uniquely determined
- $\mathbf{f_i}$ can be computed by a waterfilling algorithm

Hierarchical GPS (H-GPS)

Hierarchical GPS (H-GPS) extends hierarchical max-min fair allocation to time-variable traffic

Best possible strict service curve for class i

$$\mathcal{S}_i(t) = \max_{M \subseteq \mathrm{sib}(i)} \frac{\phi_i}{\sum_{j \notin M} \phi_j} \left(\mathcal{S}_{p(i)}(t) - \sum_{j \in M} \sum_{k \in \mathrm{ldesc}(j)} \mathcal{E}_k(t) \right)$$

with $\mathcal{S}_{root}(t) = \mathcal{C}(t)$

Schedulers for Hierarchical Link Sharing

There is a scheduling algorithm

- **Hierarchical Packet Fair Queuing (HPFQ)** [Bennett/Zhang `97]
  - Multiple stages of WFQ scheduling
  - Bounded deviation from H-GPS

… but HPFQ is nowhere implemented. Available in Linux are:

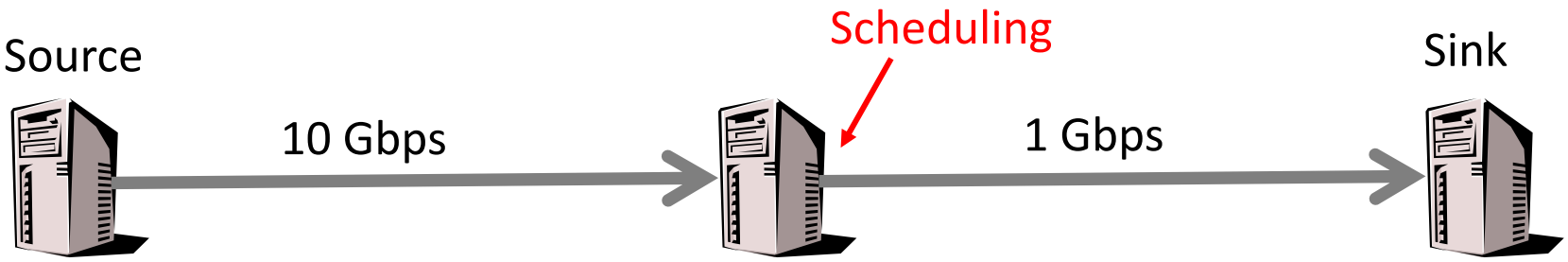
- **Class Based Queueing (CBQ)** [Jacobson/Floyd `93]
  - Multiple sharing policies
- **Hierarchical Token Bucket (HTB)** [devik `03]
  - Seeks to improve CBQ
  - Emerged from Linux developer community

CBQ and HTB

- Note: Neither scheduler specifically targets max-min fairness
- Classes are **rate controlled** at the guaranteed rate:
  - Rate estimation in CBQ
  - Token bucket in HTB
- A rate-limited class can **“borrow” bandwidth** from other classes
- Scheduling is done with **DRR**

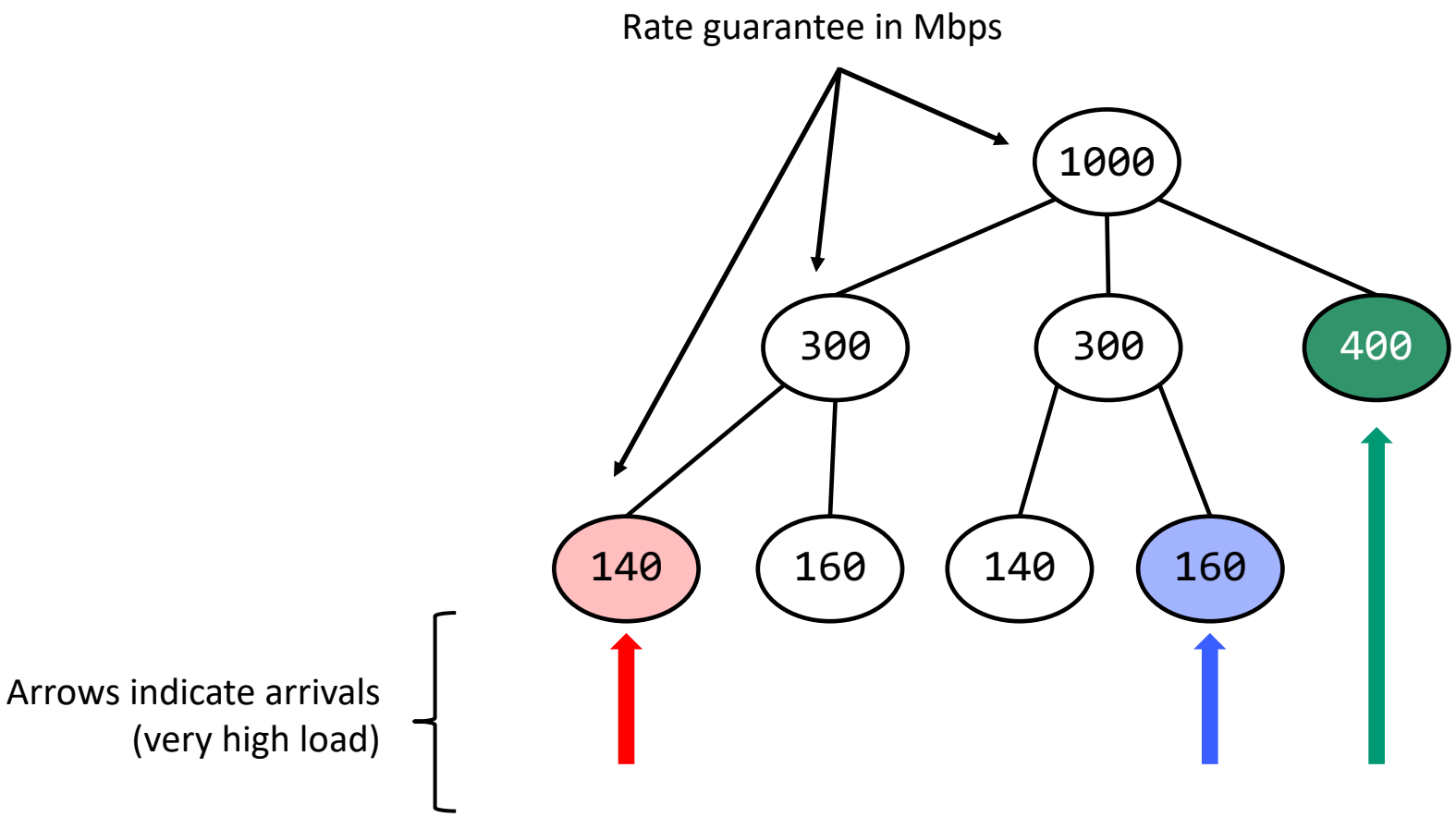
Measurement Experiments with CBQ and HTB

- On Emulab:



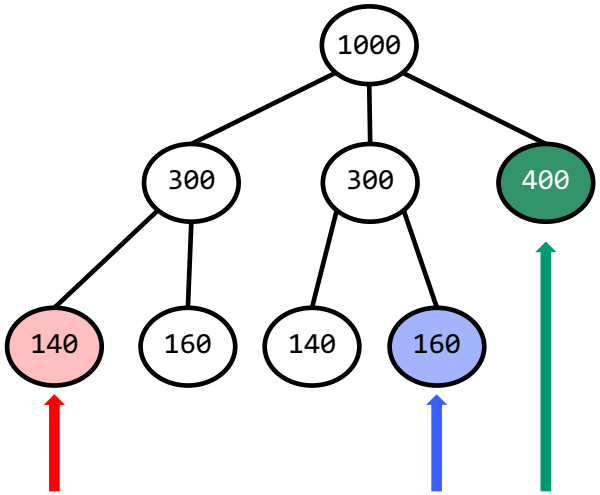
- **Findings:**
  - Confirmed results from published measurements
  - HTB and CBQ are good at supporting minimum rate guarantees, but poor at sharing excess capacity
  - Link sharing is not strategy-proof

Link sharing results



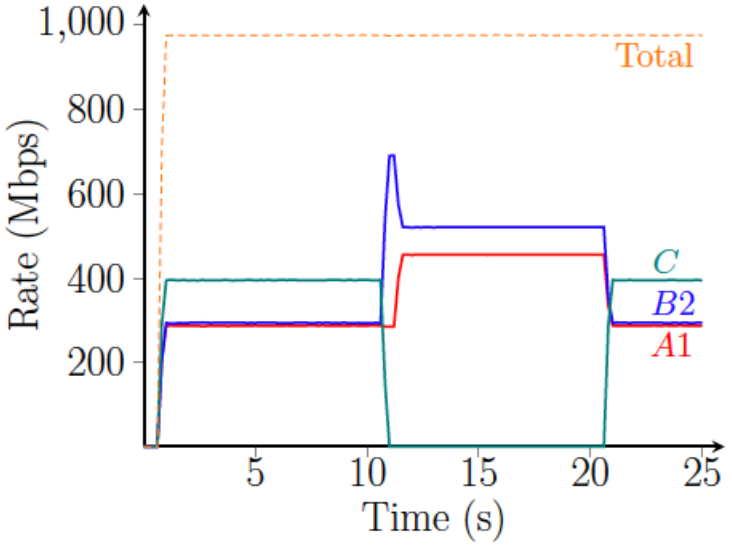
- Blue and red classes should always get the same rate

Link sharing results

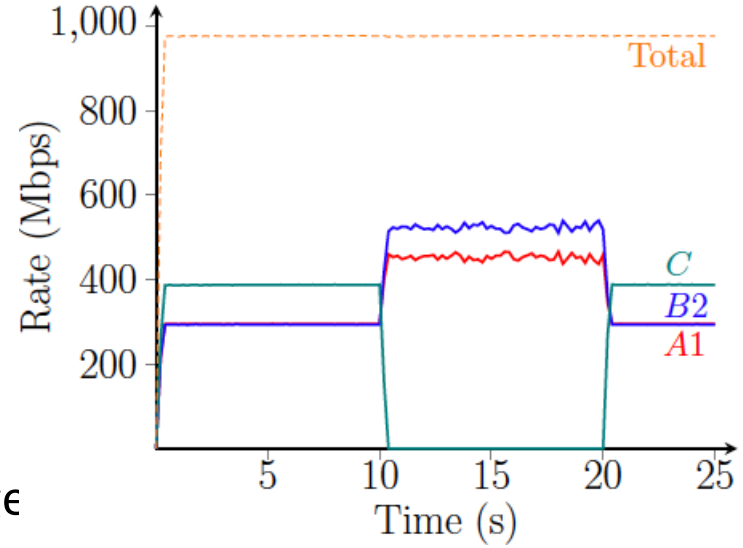


- Experiment lasts 25s
- Green flow stops for 10s ≤ t ≤ 20s

Rate gu



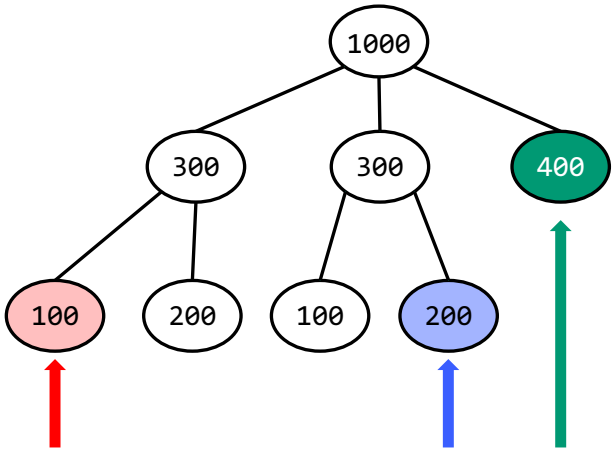
CBQ



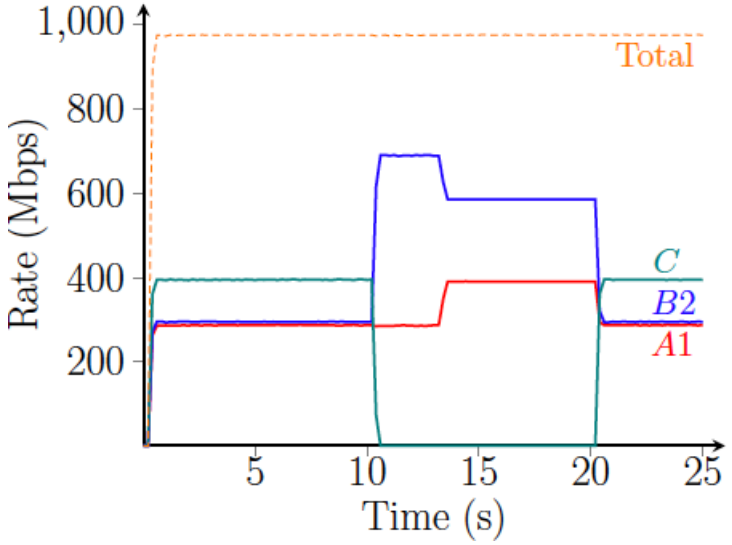
HTB

- Blue and red classes should always ge

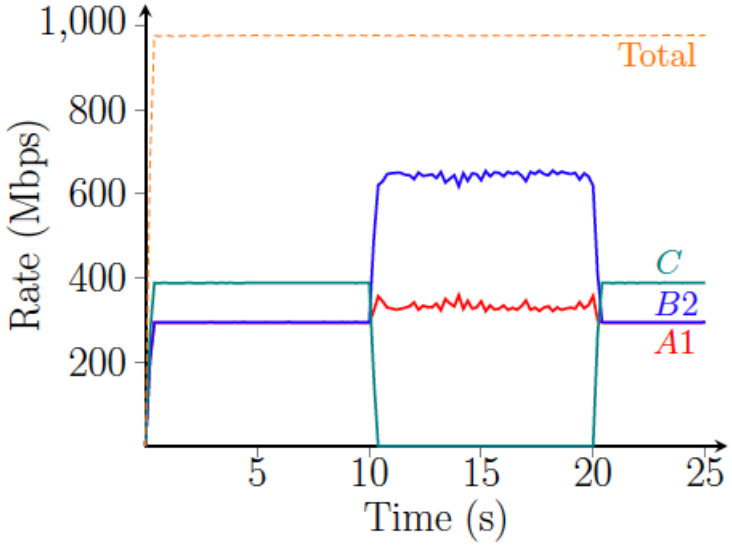
Link sharing results



Green flow stops for 10s ≤ t ≤ 20s

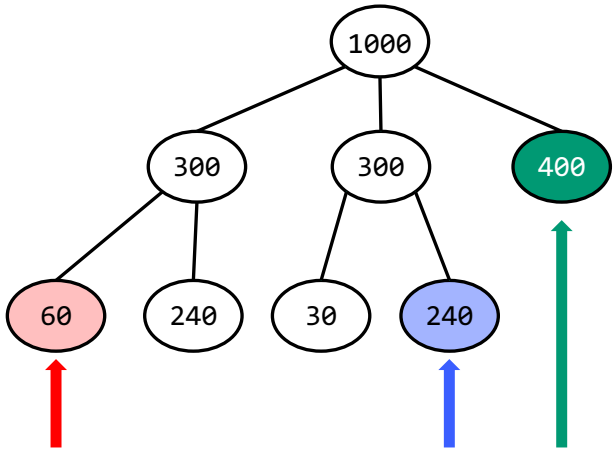


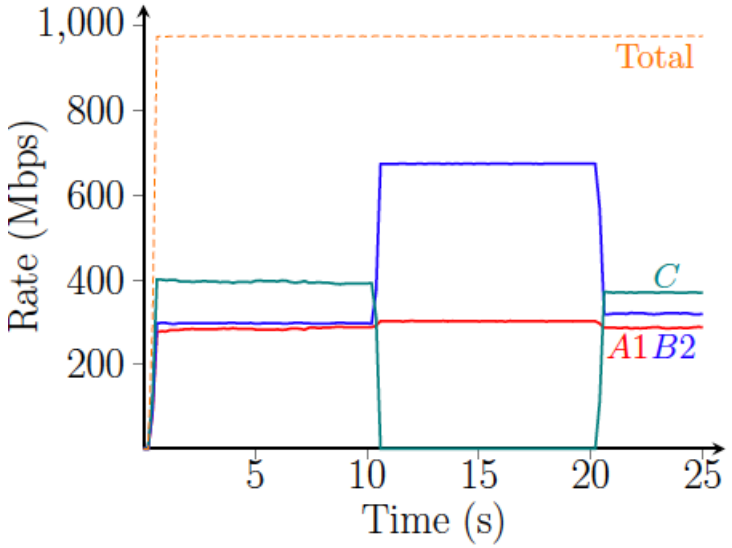
CBQ



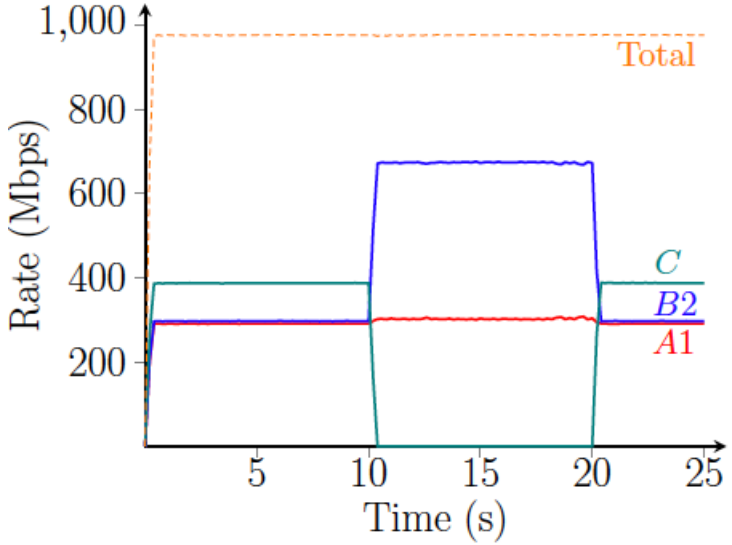
HTB

Link sharing results





CBQ

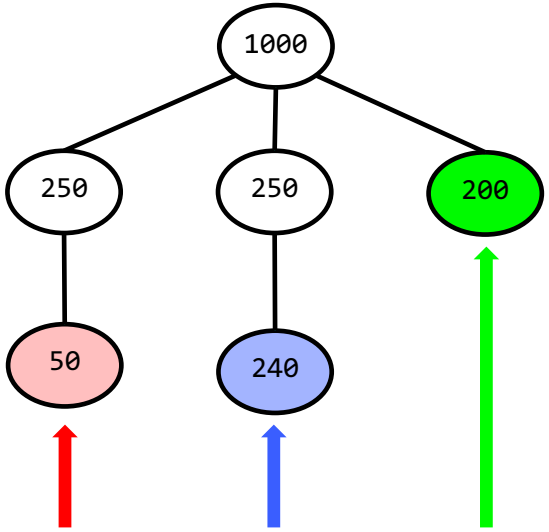


HTB

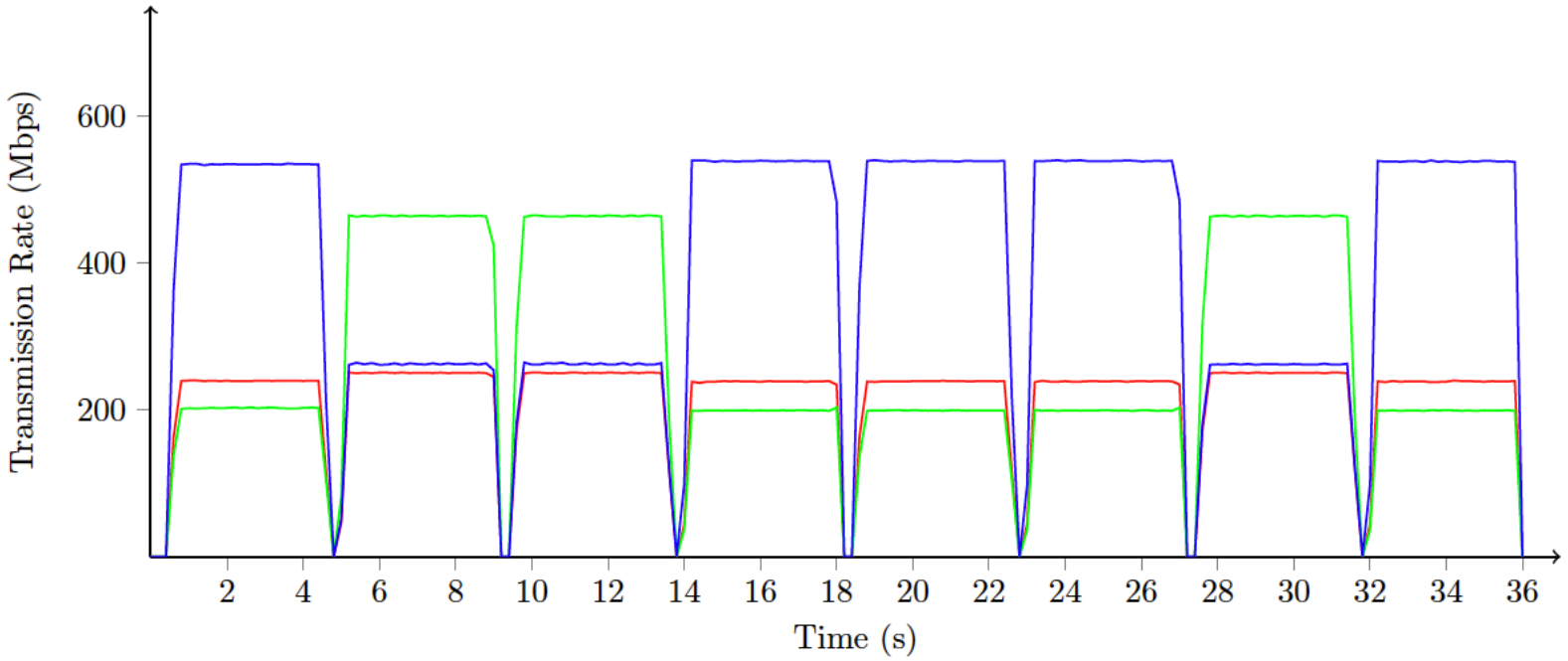
CBQ and HTB: poor link sharing

Root cause: Both only account for weights/rates of leaf classes when sharing bandwidth

Unstable allocation of CBQ



- Every 4 seconds, all senders briefly stop for ½ second and then resume
- There are different outcomes !
- **Root cause:** Sharing policy is underdetermined (hierarchical max-min fairness satisfies sharing rules, but not vice versa)



HLS – Hierarchical Link Sharing Scheduler

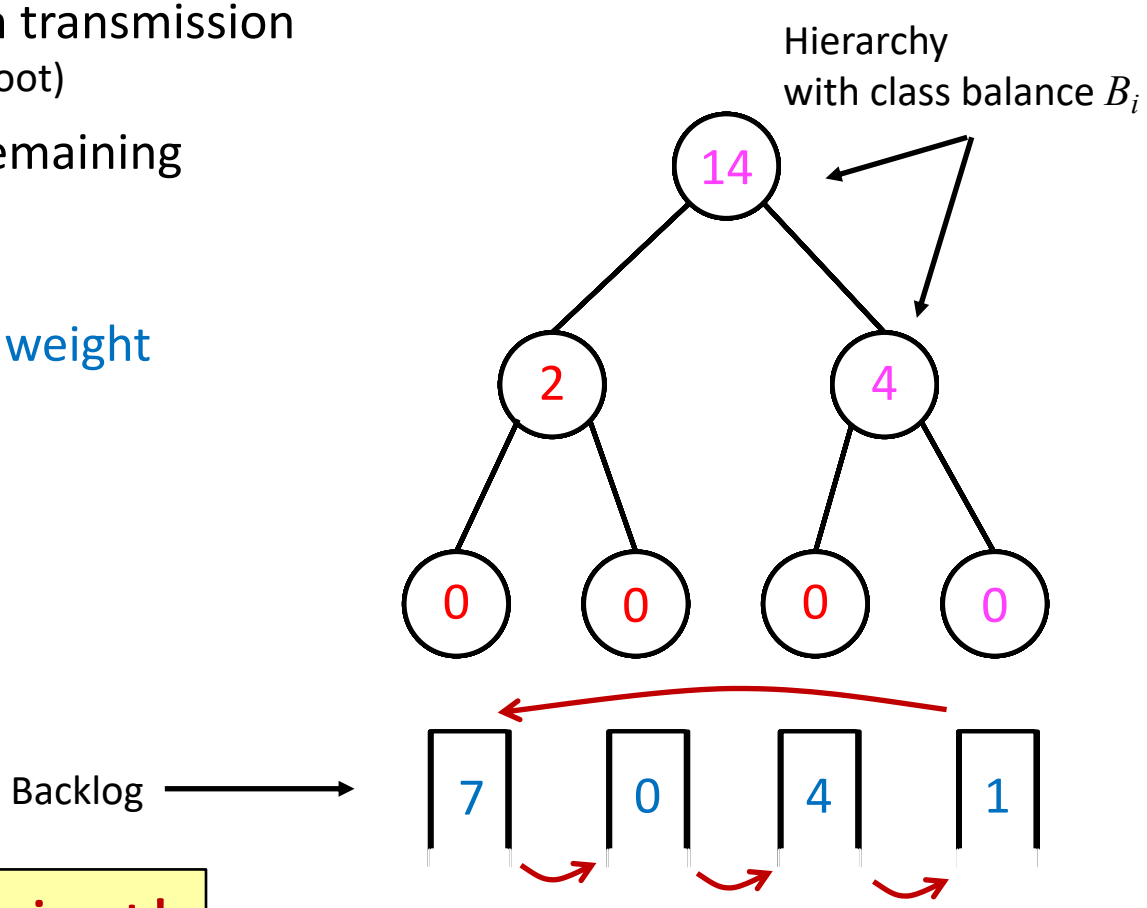
- Strategy-proof round robin scheduler that achieves hierarchical max-min fairness
- **Differences to HTB**
  - Replaces borrowing of unsatisfied classes by donations from satisfied classes

- Operates in rounds
  - active classes have a backlog at start of a round
  - Each active class obtains a quota

- Balance B_i : # bytes class i is allowed to transmit

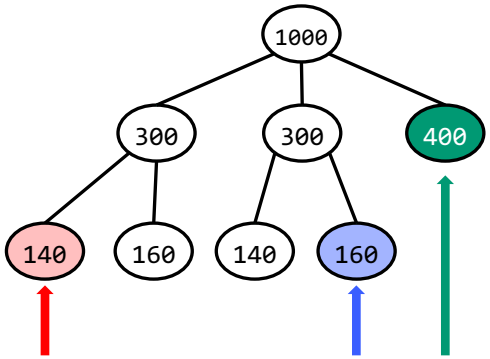
Operation of HLS

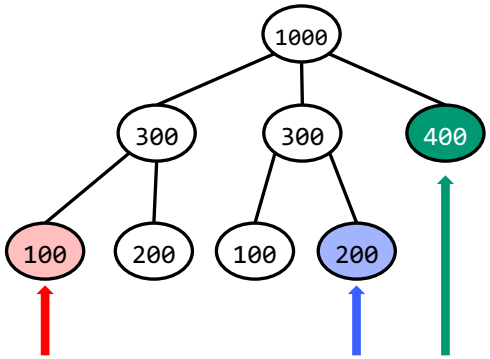
- Quota is pushed from root to active leaf classes
- Balance is updated for each transmission
  (subtracted from class, added to root)
- If class becomes inactive, remaining quota is added to parent
- Here: all classes have same weight

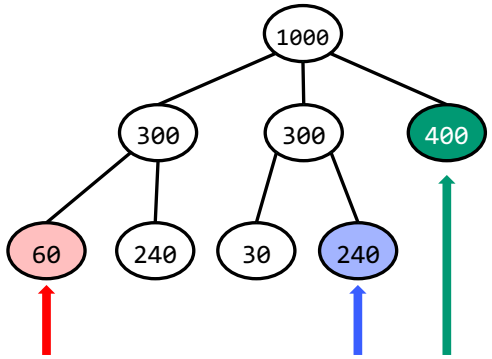


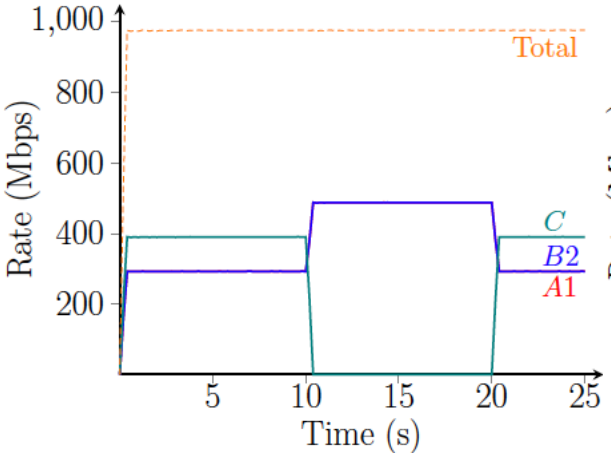
Total balance is invariant!

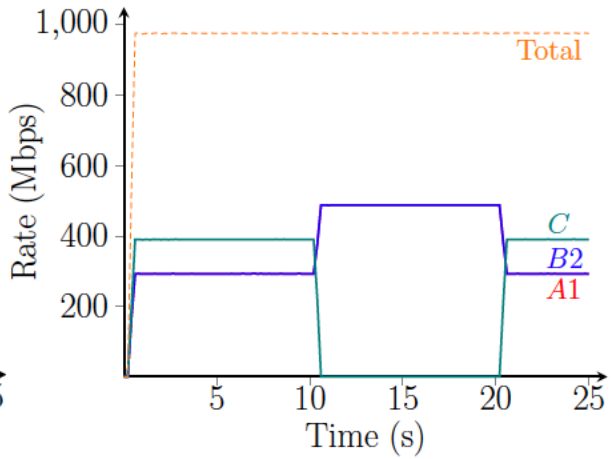
HLS experiments

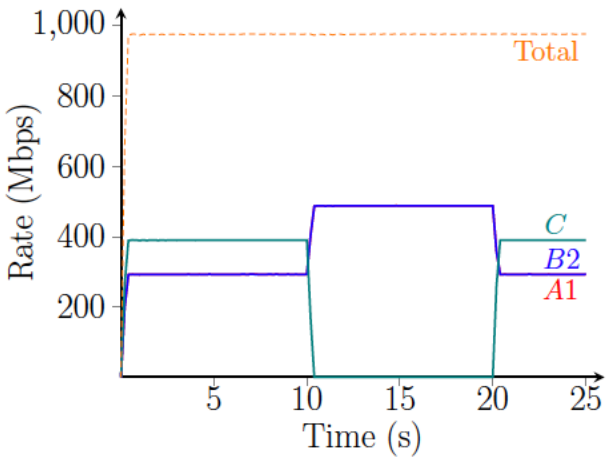






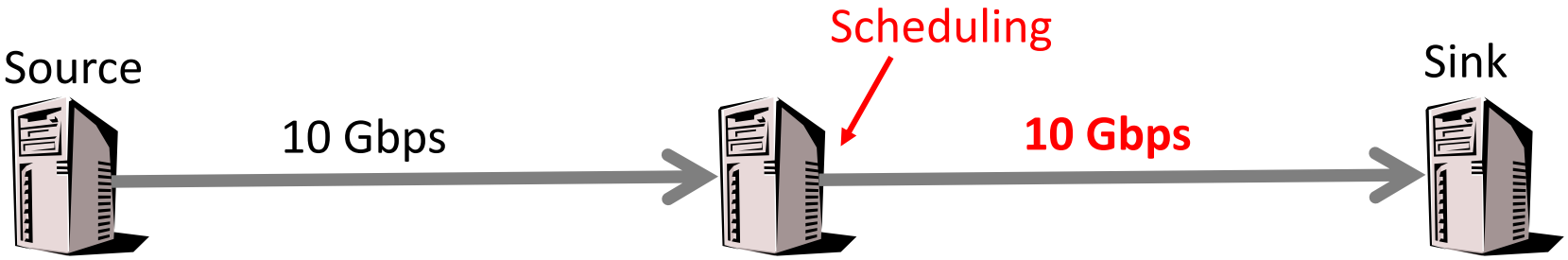






- Yields hierarchical max-min fairness

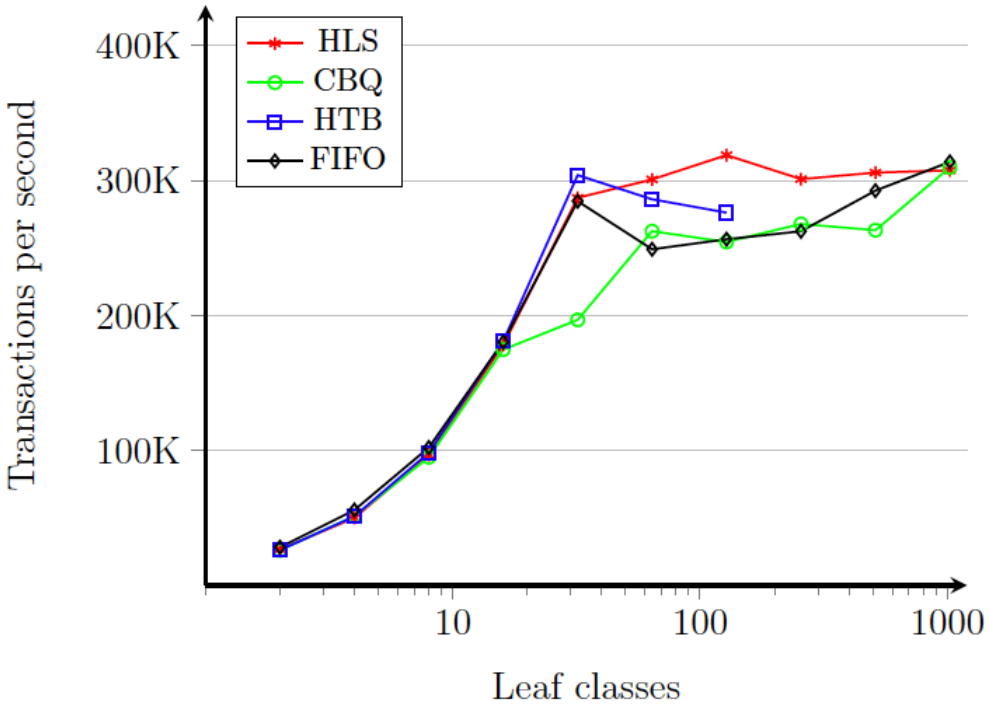
Overhead



Experiment:

- TCP traffic
- Send 1-byte packets in ping-pong fashion (NetPerf TCP-RR)
- Class hierarchy is binary tree

→ Overhead is similar



Summary of Contributions

- Solutions for hierarchical max-min fairness with fixed-rate and time-variable traffic
- Strict service curve for hierarchical max-min fairness
- Found shortcomings in link sharing schedulers in Linux:
    - Poor link sharing
    - Allocation not unique (CBQ)

- Proposed HLS, which achieves hierarchical max-min fairness
    - without drawbacks of HTB and CBQ
    - with similar overhead

Thank you!