Chair of Connected Mobility
TUM School of Computation, Information and Technology
Technical University of Munich

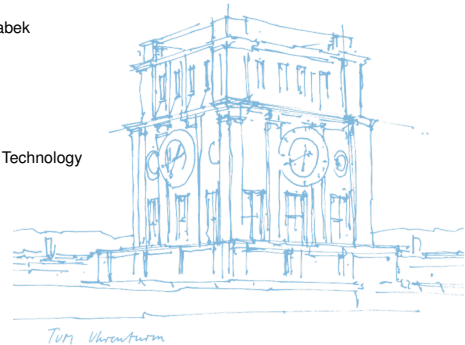
Hierarchical Resource Sharing and Queuing in OMNeT++ and INET Framework

OMNeT++ Summit 2022

Atacan Iyidogan
advised by Marcin Bosk and Filip Rezabek

Thursday 3rd November, 2022

Chair of Connected Mobility
TUM School of Computation, Information and Technology
Technical University of Munich



Content

Motivation

- Scheduling algorithms determine the order of packets to dequeue from the queue.
- Hierarchical Token Bucket (HTB) [1] is a hierarchical scheduling algorithm that is in the stock Linux kernel since 2005.
- Hierarchical Link Sharing (HLS) [2] is a new hierarchical scheduling algorithm, implemented in Linux.
- Our goal is to compare HTB and HLS.
- An implementation of HTB into OMNeT++ simulator [3] already exists.
- We implement HLS into OMNeT++, then validate and compare HLS and HTB on OMNeT++.

Research Questions

- How can HLS be implemented into OMNeT++ discrete event simulator?
- How well do HTB and HLS (hierarchical queuing structures) fit within OMNeT++ queuing structure?
- **In what ways is HLS advantageous to use compared to HTB (bandwidth sharing, impact on delay, etc.)?**

Background

Hierarchical Scheduling Algorithms

- Scheduling algorithms that are implemented into Linux kernel are called qdiscs.
- Qdiscs are split into two types: classless and classful.
- Packets arriving at classful qdiscs are filtered into one of the user-defined classes.
- Using these classes, users can control bandwidth sharing between classes.
- Using classes, a hierarchy can be realized.
- FIFO, fq_codel [4] are classless qdiscs, HLS and HTB are classful qdiscs.

Background

Hierarchical Scheduling Algorithms

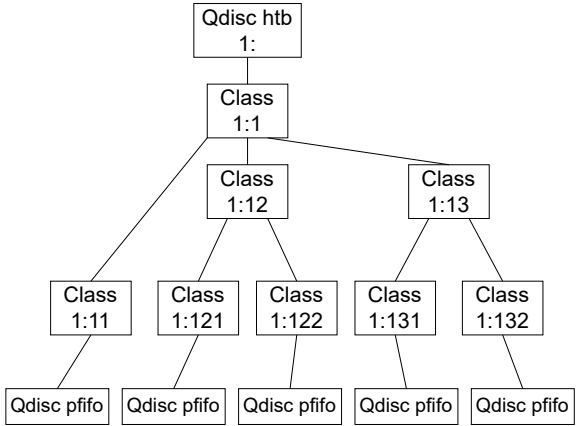


Figure 1: Example classful qdisc hierarchy

Background

Hierarchical Link Sharing (HLS)

- Each class except root has a parameter called weight.
- Each node shares bandwidth to their children proportional to their weights.
- Each class has balance, number of bytes a class is allowed to transmit.
- Classes can be active or idle. An inner class is active if any of its descendants is active.
- HLS works in rounds, in each round balance is propagated from root to active leaf classes.
- Balance is consumed in leaf classes to transmit packets.

- Each class distributes its balance to its active children, starting from top.
- The balance each child gets is proportional to their weight (compared to their siblings).

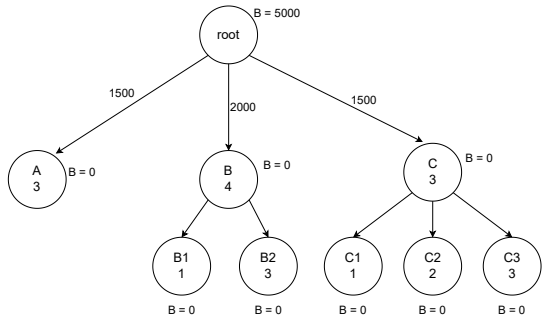


Figure 2: HLS balance distribution

Background

Hierarchical Link Sharing (HLS)

- Each class distributes its balance to its active children, starting from top.
- The balance each child gets is proportional to their weight (compared to their siblings).

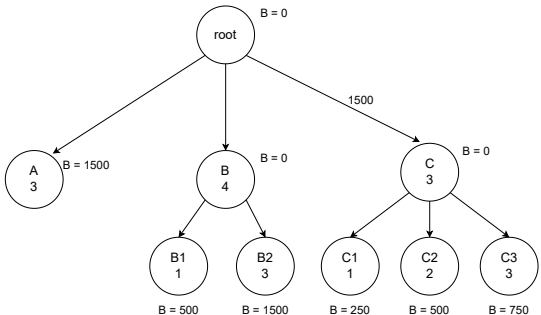


Figure 3: HLS balance distribution

- After the distribution of balance, HLS cycles between each leaf class.
- Dequeues a packet if balance is bigger than next packet byte size.
- Consumed balance is returned to root, to be distributed in the next round.

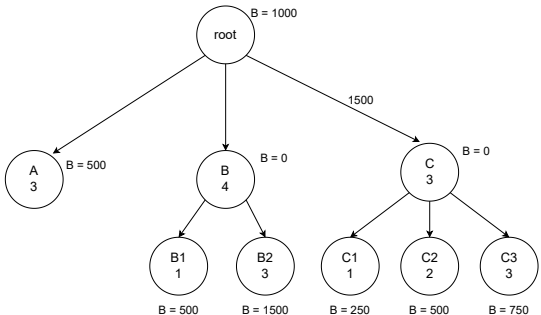


Figure 4: HLS balance distribution

Background

Hierarchical Token Bucket (HTB)

- Each class has parameters guaranteed rate and maximum rate.
- Leaf classes further have parameters priority and quantum value.
- Quantum value determines proportion of bandwidth a class will "borrow" from their ancestor.
- Classes have three modes depending on their current rate: can send (green), can borrow (yellow), and cannot send (red).

- HTB iterates through levels starting from bottom.
- Searches for active green classes with highest priority.
- If the found green class is a leaf, HTB simply dequeues a packet from that leaf class.
- If there are multiple classes to pick from, HTB utilizes a round-robin that consumes the quantum values to dequeue packets.

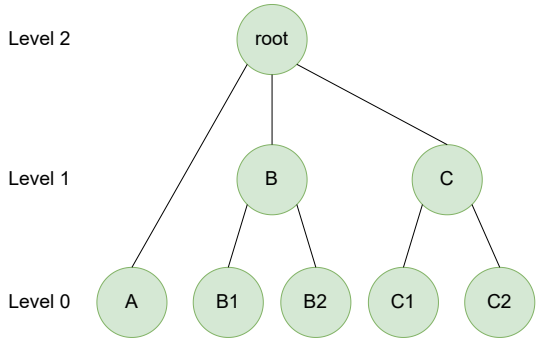


Figure 5: Example HTB Hierarchy

- If the found green class is a an inner class, HTB picks a descendant leaf to "borrow" bandwidth to.
- The bandwidth a class gets from its ancestors will be proportional to their quantum value, enforced through a deficit round-robin.

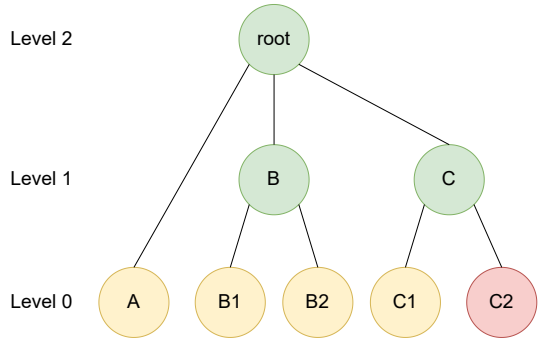


Figure 6: Example HTB Hierarchy

Analysis

Comparison of HLS and HTB

- Both are hierarchical scheduling algorithms.
- HTB has 4 different parameters that control throughput, HLS has only 1.
- HLS does not have a way to replicate the effects of priority and maximum rate parameters of HTB.
- In HLS, the rate a class achieves when every class is active can be thought as the guaranteed rate.
- Quantum values of HTB are by default proportional to their guaranteed rates. If set like this, they behave the similar to weights in HLS.

Analysis

Comparison of HLS and HTB

- HLS goes top-down while sharing bandwidth, so it is independent of condition of its subtrees.
- HTB sharing depends on which and how many descendant leaf classes are in yellow mode.
- In theory, HLS should have better class isolation. Excess bandwidth are contained in the subtrees.

Analysis

Comparison of Linux and OMNeT++ Environment

- OMNeT++ clock does not include overhead generated by classifying and scheduling operations.
- The overhead is not significant enough to diverge results of OMNeT++ and Linux.
- By default OMNeT++ clock is a perfect simulated clock, does not have any clock drift.
- INET 4.4 introduced Time-Sensitive Networking (TSN) features, including device clocks.
- HTB OMNeT++ implementation uses normal simulation clock.
- HTB heavily uses clocks to regulate class modes, so drift can result in very minor differences.
- HLS is unaffected by clock drift.

- Queuing modules are active/passive packet sink/sources.
- Modules communicate using C++ method calls through gates.

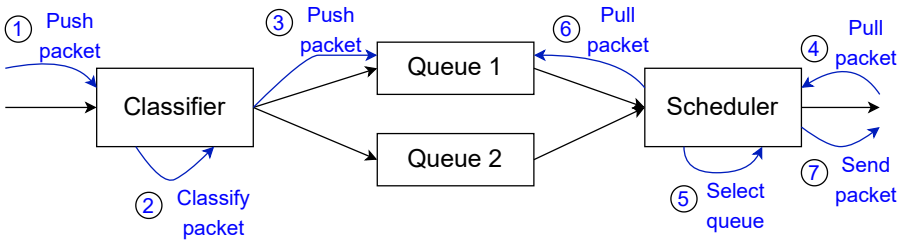


Figure 7: Example flow of a queue with a classifier and scheduler in OMNeT++

Implementation of HLS into OMNeT++/INET

Design

An ideal implementation should have the following properties:

- Should work the same as the qdisc implementation.
- The implementation should be able to use the already available classifiers and queues in INET Framework.
- The leaf packet queue types should be freely selectable. Even HLS can be a leaf queue of itself.

We designed two modules:

- HLSScheduler: Implements HLS functionality, selects which queue to dequeue from for every packet pull request.
- HLSQueue: Combine classifier, queues and scheduler into one compound module.

HLSQueue

The implementation has the same structure as HTB OMNeT++ implementation [3].

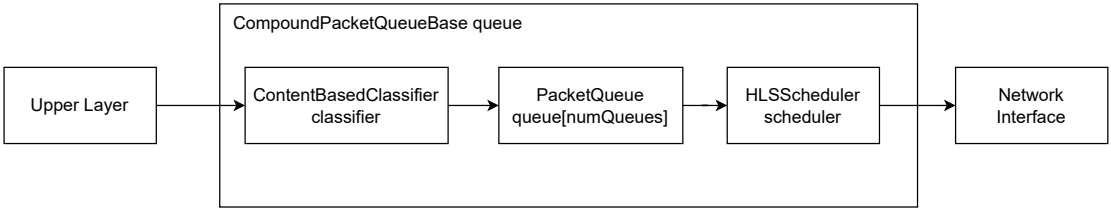


Figure 8: Design of HLSQueue module

CompoundPacketQueueBase handles interaction with upper layer and network interface.

- Outgoing packets are forwarded to classifier.
- Pull packet request from network interface is forwarded to scheduler.
- Hierarchy is set up using an XML file.

- Packets are generated on "Packet Generator" so that the generation time does not affect the scheduling.
- Hardware experiments use iperf3 [5] for packet generation.
- OMNeT++ experiments use UdpBasicApp for packet generation.

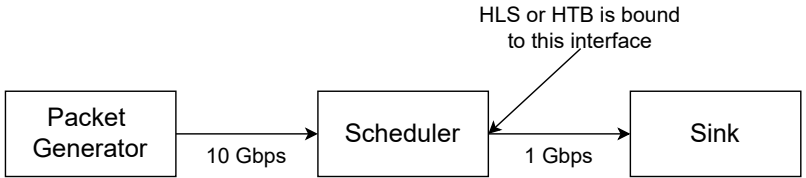
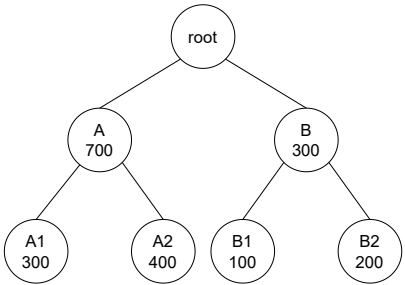


Figure 9: Configuration

Validation of HLS and HTB Against Linux

HLS Validation Test

Figure 10: Hierarchy of HLS validation test.



- Scheduler is set up on a 1 Gbps link.
- Classes should reach throughput in Mbps equal to their weights, i.e. A1 $\rightarrow$ 300 Mbps.

Validation of HLS and HTB Against Linux

HLS Validation Test Results

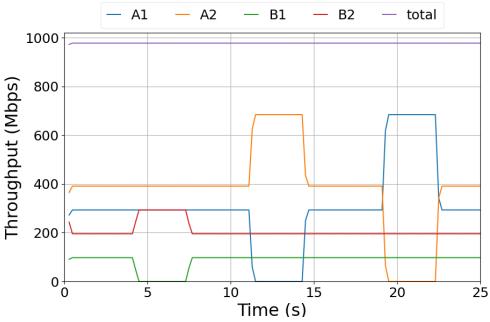


Figure 11: HLS qdisc

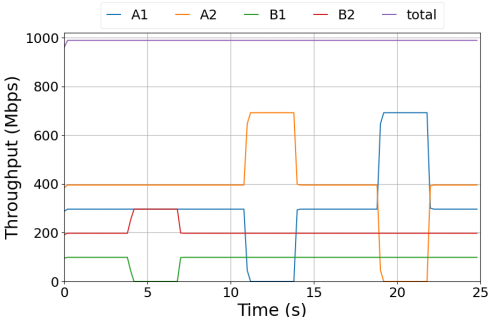
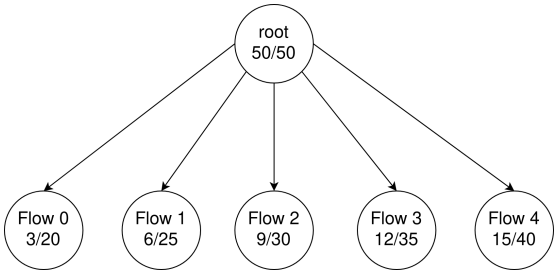


Figure 12: HLS OMNeT++

Validation of HLS and HTB Against Linux

HTB Validation Test

Figure 13: Hierarchy of HTB validation test.



- Guaranteed rate/Maximum rate in Mbps.
- Each flow should reach their guaranteed rate while not exceeding maximum rate.

Validation of HLS and HTB Against Linux

HTB Validation Test Results

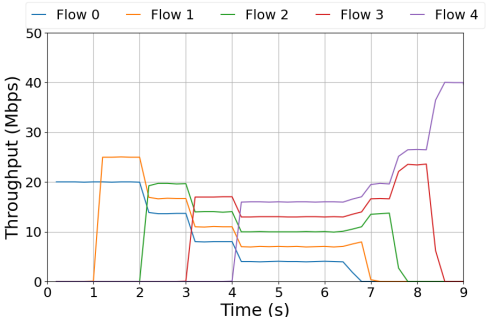


Figure 14: HTB qdisc

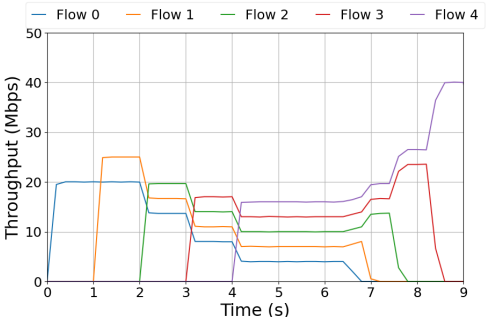


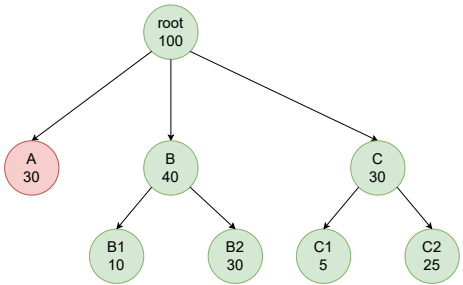
Figure 15: HTB OMNeT++

Experiments on OMNeT++

Bandwidth Sharing Comparison

- Numbers in nodes correspond to weight for HLS and guaranteed rate in Mbps for HTB.
- Quantum values are proportional to guaranteed rates.
- Maximum rate of every node is equal to link rate.
- Red nodes denote idle nodes, green nodes denote active nodes.
- Every active leaf class is generating packets at a speed of 100 Mbps.

Figure 16: Hierarchy used in throughput comparison.



Experiments on OMNeT++

Bandwidth Sharing Comparison

- Throughputs are the same before B2 becomes idle.
- When B2 becomes idle, the throughput of B does not change for HLS, does change for HTB.
- HTB shares the extra bandwidth with every leaf class, HLS gives all the extra bandwidth to B1.

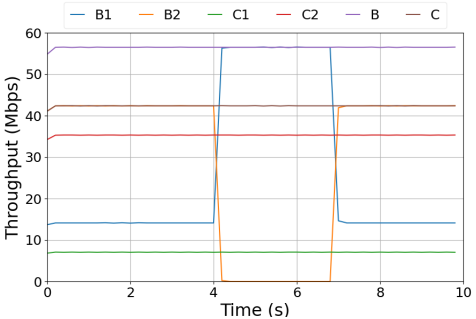


Figure 17: HLS Throughput

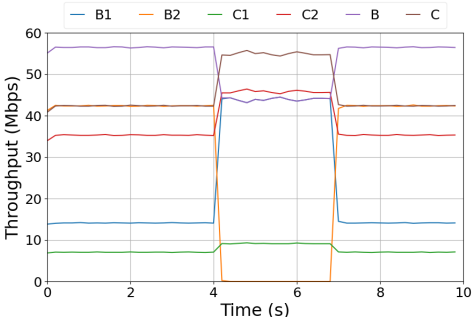


Figure 18: HTB Throughput

Experiments on OMNeT++

Delay During Congestion

- Experiment was done in OMNeT++.
- What happens to the delay of classes that send at or lower than their guaranteed rate during congestion?
- The experiment consists of 4 runs, in each only one of the classes is sending at lower than their guaranteed rate.

Table 1: End-to-end delay on delay during congestion experiment.

Classes	HLS Mean (*ms*)	HTB Mean (*ms*)
A1	1.94	2.37
A2	1.45	2.08
B1	1.48	4.07
B2	1.32	1.63

- In each experiment, the same class achieves the same throughput.

Figure 19: Hierarchy used in HLS and HTB delay experiments.

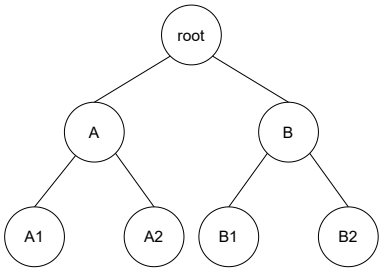


Table 2: Weights used in HLS delay experiments.

Classes	Weights
A	2
B	3
A1	1
A2	3
B1	1
B2	2

Experiments on OMNeT++

Jitter Comparison

- HTB has 2 scenarios:
  - Scenario 1 does not involve borrowing, classes only achieve their guaranteed rates.
  - Scenario 2 has low guaranteed rates, so that classes "borrow" a lot.
- Borrowing has an effect on jitter and delays while using HTB.

Table 3: HTB Scenario 1

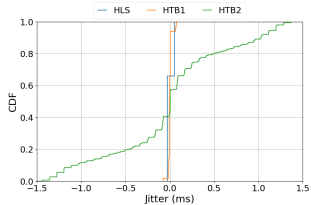
Classes	GR(Mbps)	Quantum
root	100	-
A	40	-
B	60	-
A1	10	1500
A2	30	4500
B1	20	3000
B2	40	6000

Table 4: HTB Scenario 2

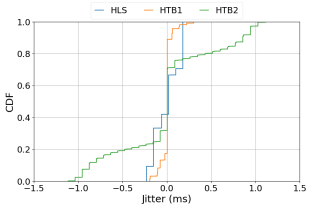
Classes	GR(Mbps)	Quantum
root	100	-
A	4	-
B	6	-
A1	1	1500
A2	3	4500
B1	2	3000
B2	4	6000

Experiments on OMNeT++

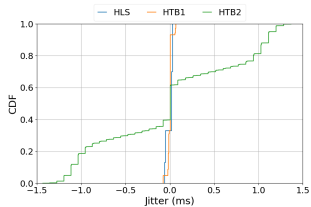
Jitter Comparison



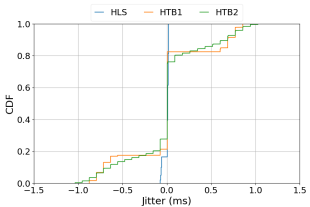
Class A1



Class A2



Class B1



Class B2

Conclusion

- **RQ:** In what ways is HLS advantageous to use compared to HTB (bandwidth sharing, impact on delay, etc.)?
  - HLS achieves class isolation better than HTB. The excess bandwidth from a leaf class in HLS is shared to leaf classes that are as close as possible.
  - The jitter resulting from HTB is higher than HLS.
  - During congestion, the delay of well-behaving classes are lower in HLS compared to HTB.
- In future, using OMNeT++ implementation, a rate limiting feature can be added to HLS and then tested for rate conformance and delay/jitter bounds.

Bibliography

- M. Devera, “Hierarchical token bucket,” http://luxik.cdi.cz/~devik/qos/htb/, 2003, accessed: 16/04/2022.
- N. Luangsomboon and J. Liebeherr, “A round-robin packet scheduler for hierarchical max-min fairness,” 2021. [Online]. Available: https://arxiv.org/abs/2108.09864
- M. Bosk, M. Gajić, S. Schwarzmann, S. Lange, and T. Zinner, “Htbqueue: A hierarchical token bucket implementation for the omnet++/inet framework,” 2021. [Online]. Available: https://arxiv.org/abs/2109.12879
- “The flow queue codel packet scheduler and active queue management algorithm.” [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8290
- “Iperf, a test tool for tcp, udp and sctp.” [Online]. Available: https://iperf.fr/

Backup Slides

Overhead Test

- OMNeT++ does not include scheduling overhead.
- The hierarchies are perfect binary trees.
- UDP packets of size 60 bytes are generated. We compute the number of packets sent per second.

Table 5: Results of overhead analysis experiments.

Qdisc	Packets/s (4 leaves)	Packets/s (128 leaves)
HLS	362240	241970
HTB	365590	226295
pfifo	358880	377945

- HLS scales slightly better than HTB.
- According to [2], classifying packets is the most significant contributor to overhead.
- By default, qdiscs cannot enqueue and dequeue at the same time.
- The overhead is not significant enough for the link data rates we use in OMNeT++ experiments.

Backup Slides

HTB Borrowing Analysis

- Quantum values and priorities of leaf classes are equal.
- Maximum rate of all classes are set to 100 Mbps.
- Every leaf class is generating packets at a speed of 100 Mbps.
- Experiment was done in OMNeT++.

Figure 20: Hierarchy used in HTB borrowing analysis Experiment 1.

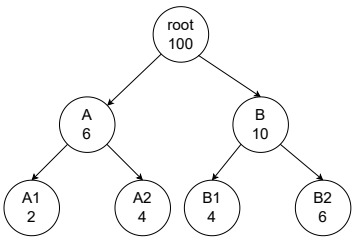
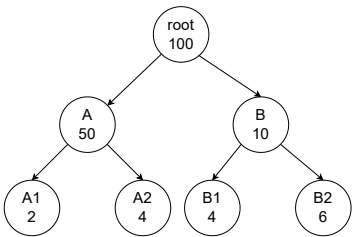


Figure 21: Hierarchy used in HTB borrowing analysis Experiment 2.



HTB Borrowing Analysis

- Colored numbers under the nodes denote how much bandwidth that class borrowed from the corresponding ancestor.
- The guaranteed rate of inner node changed how root distributes bandwidth.
- This can lead to unintended and unpredictable bandwidth sharing.

Figure 22: Experiment 1 results

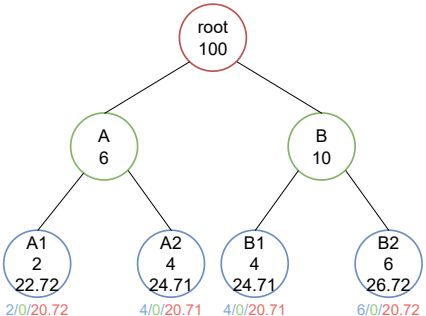


Figure 23: Experiment 2 results

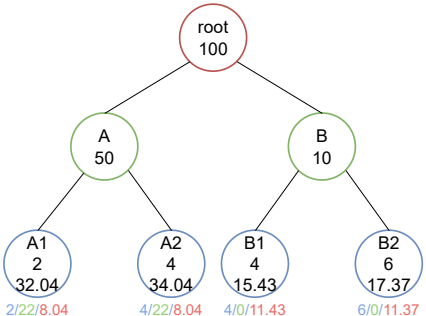
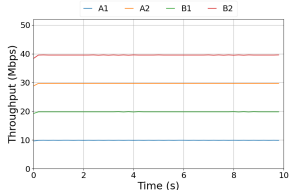
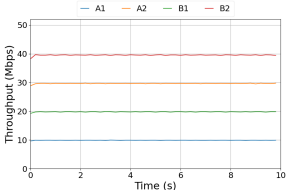


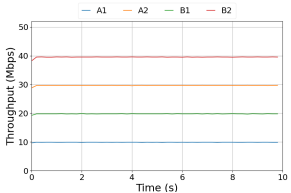
Figure 24: Throughput



HLS



HTB Scenario 1



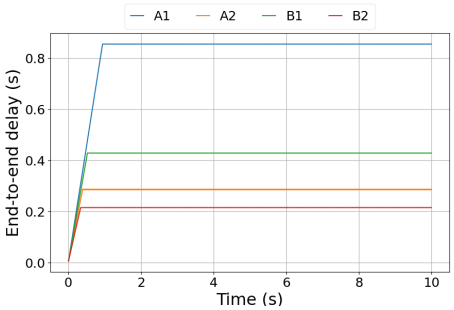
HTB Scenario 2

Backup Slides

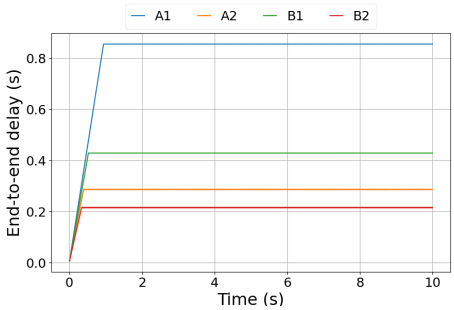
Delay and Jitter

- Qdiscs by default and this OMNeT++ test has FIFO queues with packet size limit of 1000 in leaf classes.
- To minimize delay using UDP it might be beneficial to reduce queue size or use another qdisc as leaf queues.

Figure 25: End-to-end delay



HLS



HTB Scenario 1 and 2