HTBQueue: A Hierarchical Token Bucket Implementation for the OMNeT++/INET Framework

Authors:
Marcin Bosk, Marija Gajić, Susanna Schwarzmann, Stanislav Lange, and Thomas Zinner

Presenters:
Marija Gajić, M.Sc. (NTNU; marija.gajic@ntnu.no)
Marcin Bosk, M.Sc. (TUM; bosk@in.tum.de)

Funding:
EC H2020 TeraFlow
SWC BigQoE

8th of September 2021
Agenda

Motivation

The Hierarchical Token Bucket

HTB Implementation for OMNeT++/INET

Rate Conformance Validation

Summary
Motivation
Motivation

Nowadays Internet: heterogeneous services + increased number of customers + pandemic effects

A key role for network providers – advanced, efficient and scalable traffic shaping, and resource allocation

OMNeT++/INET framework – one of the most prominent discrete simulation frameworks for gaining new knowledge and insights.

Figure 1: Traffic changes during the COVID-19 pandemic’s, taken from [1]
Motivation

Desired properties of an advanced traffic shaper:

- 3GPP standard for 5G and beyond network technologies [2] specifies that flows should have two-level bitrate limits: Guaranteed Flow Bitrate (GFBR) and Maximum Flow Bitrate (MFBR)
- Scalability
- Prioritization
- Efficient resource allocation and isolation in a slice-like manner
- Feasibility to use in research testbeds and for different QoS policy enforcement
- Good rate conformance

The Hierarchical Token Bucket (HTB) fulfills all described properties.

HTBQueue: our implementation of the HTB as compound module in OMNeT++/INET framework.

Currently no built-in support in OMNeT++
The Hierarchical Token Bucket
The Hierarchical Token Bucket

A classful token bucket bucket algorithm

Rate control using two nested token buckets, controlling the assured and ceiling rate

Utilizes a tree hierarchy to control multiple traffic classes. Three class types exist: root, inner and leaf

Inner classes allow grouping of leaf classes
Possible use-case: 5G network slicing

Sum of children’s assured rates cannot exceed assured rate of the parent
The Hierarchical Token Bucket – Rate Borrowing Principle

Mode of class determined by three different states:

- can send
- may borrow
- can’t send

Each class on Level \( L > 0 \) keeps a list of its descendant nodes that would like to borrow \( D_{\text{class}} \).

Key parameters per class:

- Ceiling rate - \( R_{\text{ceiling}} \)
- Assured rate - \( R_{\text{assured}} \)
- Current rate utilized by the parent - \( R_{\text{parent}} \)
- Quantum of the class - \( Q_{\text{class}} \)
- Borrowed bandwidth - \( B_{\text{class}} \)
HTB Implementation for OMNeT++/INET
HTB Implementation for OMNeT++/INET

HTBQueue

- Implemented as an OMNeT++ compound module
- Extension of the INET Framework

Packet queue module, similarly to e.g. PriorityQueue

- Used as a replacement for any packet queue on an interface
- Applicability limited to the PPP interface

Implementation based on Linux HTB by Martin Devera [3]

- Linux HTB source code [4] port to C++
- OMNeT++ specific adjustments

Implementation available on Github: https://github.com/fg-inet/omnet_htb
The HTBQueue Compound Module

Compound queue implementing the Hierarchical Token Bucket algorithm
- Queuing module for PPP interface
- Utilization of already existing queues for leaf queues

Consists of three modules
- HTBClassifier – Classifies packets into correct leaf classes
- Queues – Generic queues, number of which corresponds to number of leaf classes
- HTBScheduler - Schedules packets and implements the actual functionality of HTB
The HTBClassifier Module

- Extension and adaptation of the ContentBasedClassifier [5]
- Functionality for filtering and forwarding packets analogous to ContentBasedClassifier
- Support for signalling to the HTBScheduler module

Packet filtering based on
- Packet type (e.g. ping packet)
- Packet information (e.g. packet source IP)
The HTBScheduler Module

**HTBScheduler**

- Implementation of scheduling functionality of HTB
- Enforcement of classful HTB scheduling hierarchy
- Enforcement of assured and ceiling rates for each class

Selection of the next leaf class/queue to dequeue when

- The interface finished transmission event occurs
- An event with a new packet arriving to the queue and interface being idle occurs
- A timeout occurs – i.e. an active leaf class can send again

The timeout

- Set if there are only packets in leaf queues that can’t be sent during either of the first two events
- Invokes `refreshOutGateConnection` method of the **PPP module**
- PPP Interface modification $\rightarrow$ the `refreshOutGateConnection` method has to be public
The HTBScheduler Module – Functional Overview

1. Timeout occurred
   - Call `refreshOutGateConnection`

2. HTBQueue available packets?
   - Yes
     - Any leaf in `can_send` mode?
       - Yes
         - Call `htbDequeue` method and select next leaf to dequeue
       - No
         - No
         - Set timeout for next expected class mode change
         - Unchecked leaves left?
           - Yes
           - No
           - No
   - No
     - Put interface to IDLE

3. Current node in `can_send`?
   - Yes
   - No
     - Current node in `may_borrow`?
       - Yes
       - Select the current node parent as current node
       - No
       - No
         - Select an unchecked leaf in `may_borrow` mode as current node
         - Mark selected leaf as checked

4. Return ID of queue corresponding to selected leaf class

5. Dequeue packet and transmit on interface

Rate Conformance Validation
Rate Conformance Validation

Rate limiting and sharing verified in simple experiments with two directly connected hosts

- Hosts connected via PPP interfaces
- **HTBQueue** configured on each PPP interface
- UDP flows between hosts sending 1500 Byte packets every 100µs in one direction
- Each flow corresponds to one leaf class
- Flows started in 10s intervals and ran for 100s

Three scenarios covering different configurations w.r.t. inner/leaf nodes and priorities
Rate Conformance Validation – HTB Configuration

Scenario 1

Scenario 2

Scenario 3

\[ R_a = \text{assured rate (in Mbit/s)} \]
\[ R_c = \text{ceiling rate (in Mbit/s)} \]
Rate Conformance Validation – Scenario 1


Flow 0
- $R_a = 3$
- $R_c = 20$

Flow 1
- $R_a = 6$
- $R_c = 25$

Flow 2
- $R_a = 9$
- $R_c = 30$

Flow 3
- $R_a = 12$
- $R_c = 35$

Flow 4
- $R_a = 15$
- $R_c = 40$

Level 0
- $R_a, root = 50$ Mbit/s
- $R_c, root = 50$ Mbit/s

Level 1

Throughput [Mbit/s]

Simulation Time [s]
Rate Conformance Validation – Scenario 2

Rate Conformance Validation – Scenario 3

Flow 0
Higher priority

Flow 1
Lower priority

\[ R_a, \text{root} = 50\text{Mbit/s} \]
\[ R_c, \text{root} = 50\text{Mbit/s} \]

\[ R_a = 5 \]
\[ R_c = 30 \]

\[ R_a = 5 \]
\[ R_c = 30 \]
Summary

Motivation for *HTBQueue* implementation

- Key role for network operators nowadays – traffic shaping and optimal resource allocation
- No built-in advanced traffic shaper allowing for hierarchical two level bitrate guarantees in OMNeT++
- HTB concepts still in use today and applicable to numerous use-cases (5G and beyond)

Design and implementation of a compound module in *INET* framework – *HTBQueue*

- Consists of classifier, generic queues and scheduler
- Code based on the Linux HTB implementation

HTBQueue validation in different scenarios

- Good rate conformance for leaf and inner classes
- Enforces different priority settings
Thank You!

Q&A
Demo

Scenario 2

Flow0
Flow1
Flow2
Flow3
Flow4

HTBQueue Implementation into an Existing Project

Copy the HTB Implementation files over from github

- [https://github.com/fg-inet/omnet_htb](https://github.com/fg-inet/omnet_htb)
- I.e. merge the code/inet4 folder with your inet4 folder

Compile INET

Configure HTBQueue as the packet queue on interfaces in the INI file

```plaintext
*.serverFD0*.pp[0].pp.queue.typename = "HTBQueue"
*.serverFD0*.pp[0].pp.queue.numQueues = 5
*.serverFD0*.pp[0].pp.queue.queuel[].typename = "DropTailQueue"
*.serverFD0*.pp[0].pp.queue.htbHysteresis = false
*.serverFD0*.pp[0].pp.queue.htbTreeConfig = xmlDoc("tree_scenario1.xml")
*.serverFD0*.pp[0].pp.queue.queuel[].packetCapacity = 500
*.serverFD0*.pp[0].pp.enqueue.classifier.defaultGateIndex = 1
*.serverFD0*.pp[0].pp.enqueue.classifier.packetFilters = "*;*;*;*;*"
*.serverFD0*.pp[0].pp.enqueue.classifier.packetDataFilters = "destinationPort(1042);destinationPort(1043); destinationPort(1044);destinationPort(1045);destinationPort(1046)"
```

Prepare XML HTB configuration

(example)
References


2. 3rd Generation Partnership Project (3GPP), \NR; NR and NG-RAN Overall description; Stage-2," Technical Specification (TS) 38.300, Mar. 2021, release 16.5.0.


4. HTB Linux source code, https://github.com/torvalds/linux/blob/master/net/sched/sch_htb.c