ns3-rmcat open source module

(companion to draft-zhu-rmcat-framework)

Jiantao Fu, *Sergio Mena*, Xiaoqing Zhu

Outline

- Introduction
- Source code structure
- Relevant features
  - Comparing congestion controllers
  - Running *ns3-rmcat*
  - Producing results
- Example plots
- Future enhancements

ns3-rmcat, what is it?

- New module to ns3 simulator
  - `https://www.nsnam.org/`
  - C++, python
- Current uses of *ns3-rmcat*
  - Run rmcat wired/wireless test cases
  - Flexible testbed
    - test/debug/experiment with NADA
    - plug different traffic source models (*syncodecs*)
      - See `https://github.com/cisco/syncodecs`
  - Plot rmcat test case results
  - Further processing of results in Matlab/Octave
- **Reasons for open sourcing**
  - Reference implementation of rmcat framework
  - Pluggable congestion control algorithms
  - Common testbed allowing algorithm comparison

Source Code Structure

- Pluggable congestion controllers
  - Common superclass: **`SenderBasedController`**
  - Current subclasses: **`DummyController`** (CBR), **`NadaController`**
- Topologies and test cases specified in rmcat internet drafts
  - wired
  - wifi
- Custom ns3 applications
  - Classes **`RmcatSender`** and **`RmcatReceiver`**
  - Sender-based logic
  - Feedback format not implemented yet
    - Per-packet: logic of **`RmcatReceiver`** very simple
- Traffic source models: git submodule (*syncodecs*)
- Tools
  - Mainly for processing and plotting output logs
- Simple examples

RELEVANT FEATURES

Comparing Congestion Controllers

- All congestion controllers implement a common interface:
  - abstract class **`SenderBasedController`**
  - This class also contains common infra code
- Important member functions
  - **`virtual bool processSendPacket(uint64_t txTimestamp, uint32_t sequence, uint32_t size);`**
  - **`virtual bool processFeedback(uint64_t now, uint32_t sequence, uint64_t rxTimestamp, uint8_t ecn=0);`**
  - **`virtual float getBandwidth(uint64_t now) const =0;`**
- Two actual controllers implemented so far
  - **`DummyController`**: outputs constant bandwidth
  - **`NadaController`** (doesn't need to override **`processSendPacket`**)

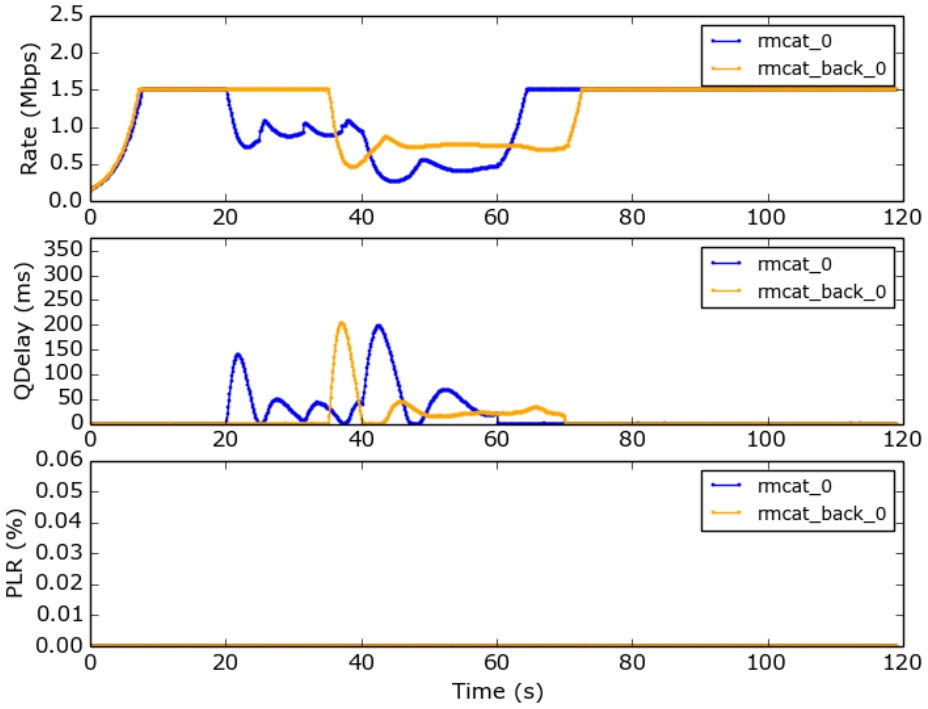
Running *ns3-rmcat*

- Either simple examples…
  - to play around with the module
  - simplistic topology
- … or ns3 unit-testing framework (`test.py`)
  - Automated
  - Used for running rmcat tests
  - Two test suites: *rmcat-wired* & *rmcat-wifi*
  - Further details in README
- Output: directory with **log files**
  - to analyze issues
  - to be parsed and produce plots (see next slide)

Producing Results

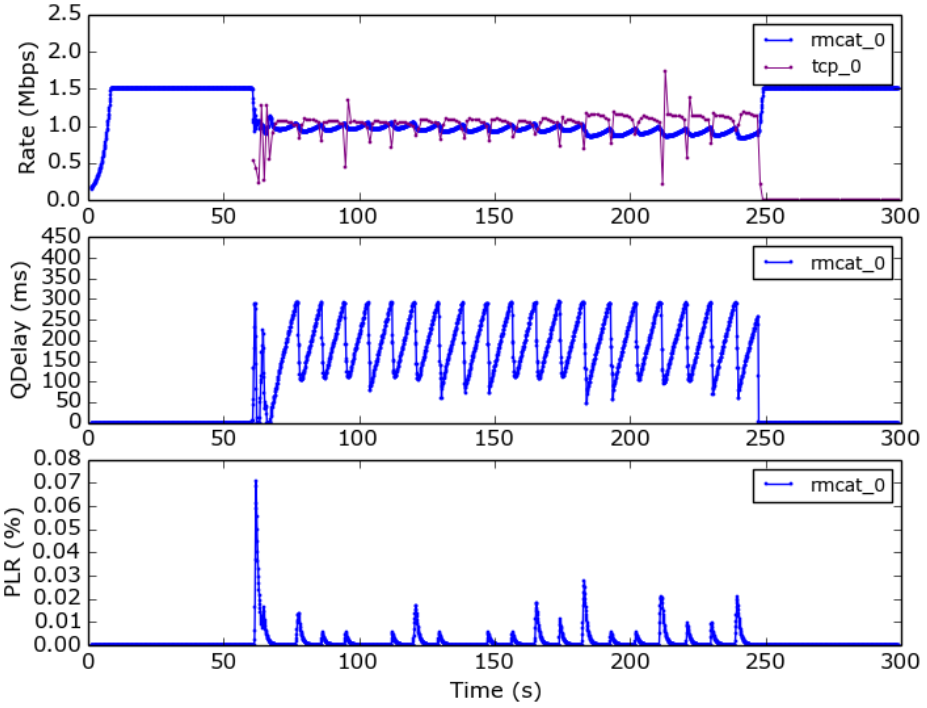
- Python scripts provided:
  - parse the log files to generate:
    - json file *all_tests.json*
    - *.mat* files for individual tests (for Matlab/Octave)
  - using the json file, plot all test cases
    - automated
    - output: *.png* files using library *matplotlib.pyplot*
  - file *all_tests.json* can be loaded in Matlab/Octave
    - portable library: JSONLab
    - allows for customized plotting/further study

Example Plots



← *rmcat-wired*, test case 5.3

- One forward rmcat flow
- One backward rmcat flow



rmcat-wired, test case 5.6 →

- One forward rmcat flow
- One forward TCP flow

Future enhancements

- Extensions:
  - Cellular test cases and topology
  - Other candidate congestion control algorithms
- Alignment to rmcat drafts
  - Inter-component interactions
    - draft-zhu-rmcat-framework
  - Align with draft-ietf-rmcat-eval-test
    - change physical bandwidth
      - background UDP for the moment
    - jitter (not present)
  - Feedback format implementation
    - Currently, per-packet (no grouping/compression)

Finally

- Code will be available shortly
  - Got green light from Cisco Legal for open sourcing
  - Cleaning up, documenting (README, etc.)
  - Feel free to contribute!

Questions?