# New Congestion Adaptation for ndncatchunks

3rd NDN Hackathon

Klaus Schneider, Teng Liang

November 5, 2016

The University of Arizona

# Intro

**Motivation:**

- Lots of interest in simple file transfer app that doesn't have congestion problems

**Initial plan:**

1. Implement new congestion adaptation in ndncatchunks (TCP BIC & CUBIC)
2. Performance measurements
3. Make ndncatchunks **modular** & more usable for others to **experiment** with new congestion control algorithms

# Achievements

1. Elicit/fixed some **usability preblems** of ndncatchunks
    - Timeouts for version discovery
    - ndnputchunks: **slow publishing:** over **6 minutes** for 1.3 GB file on Intel i7 CPU (stdin vs. spec. filename)
    - ndnputchunks: no feedback when file finished loading
    - ndncatchunks: default to write into file vs. stdout?

# Achievements

1. Elicit/fixed some **usability preblems** of ndncatchunks
    - Timeouts for version discovery
    - ndnputchunks: **slow publishing:** over **6 minutes** for 1.3 GB file on Intel i7 CPU (stdin vs. spec. filename)
    - ndnputchunks: no feedback when file finished loading
    - ndncatchunks: default to write into file vs. stdout?
2. Implement and improve performance tracers
    - Implemented Rate Tracing
    - Improved RTT Tracing

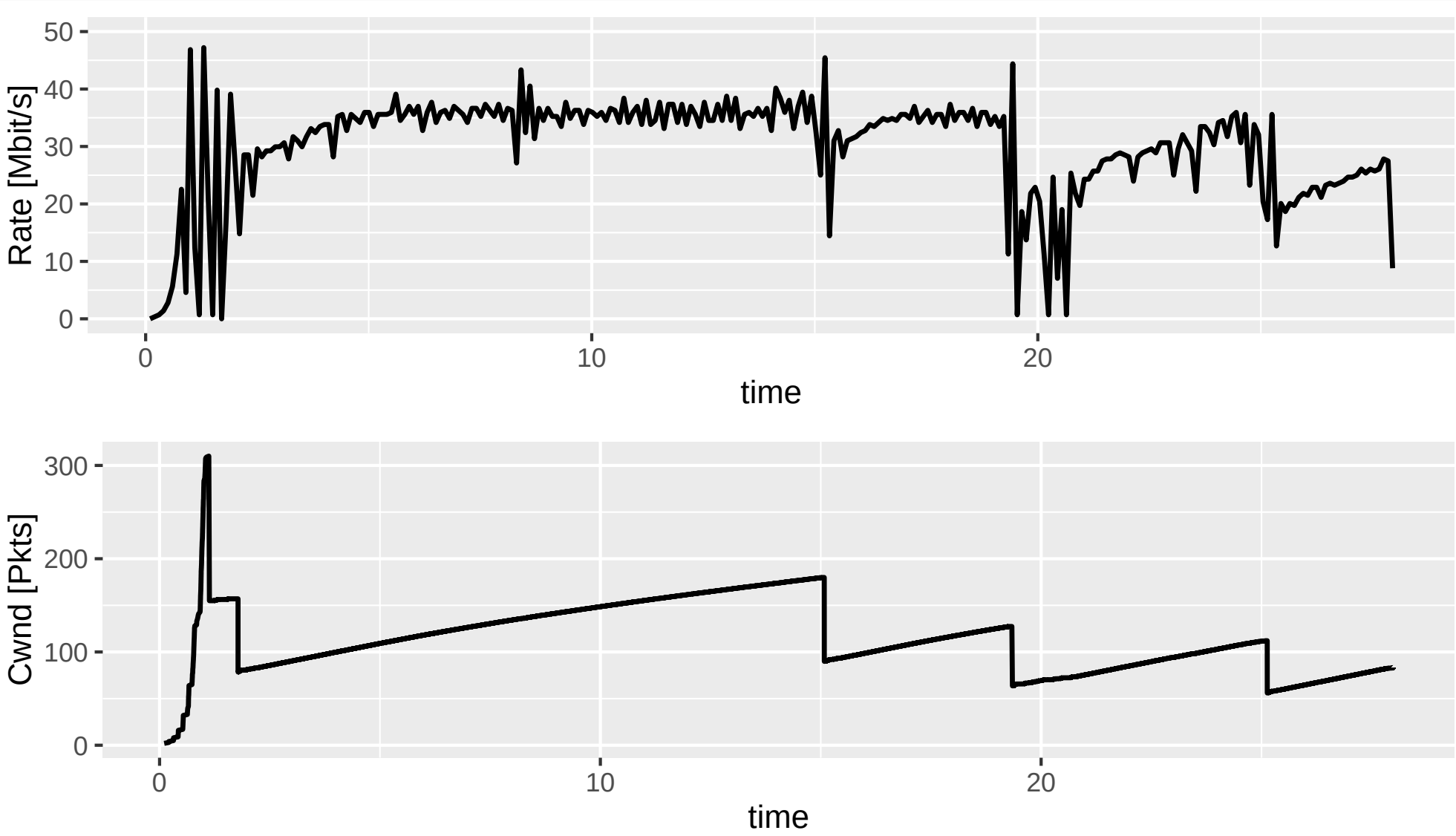
# Achievements

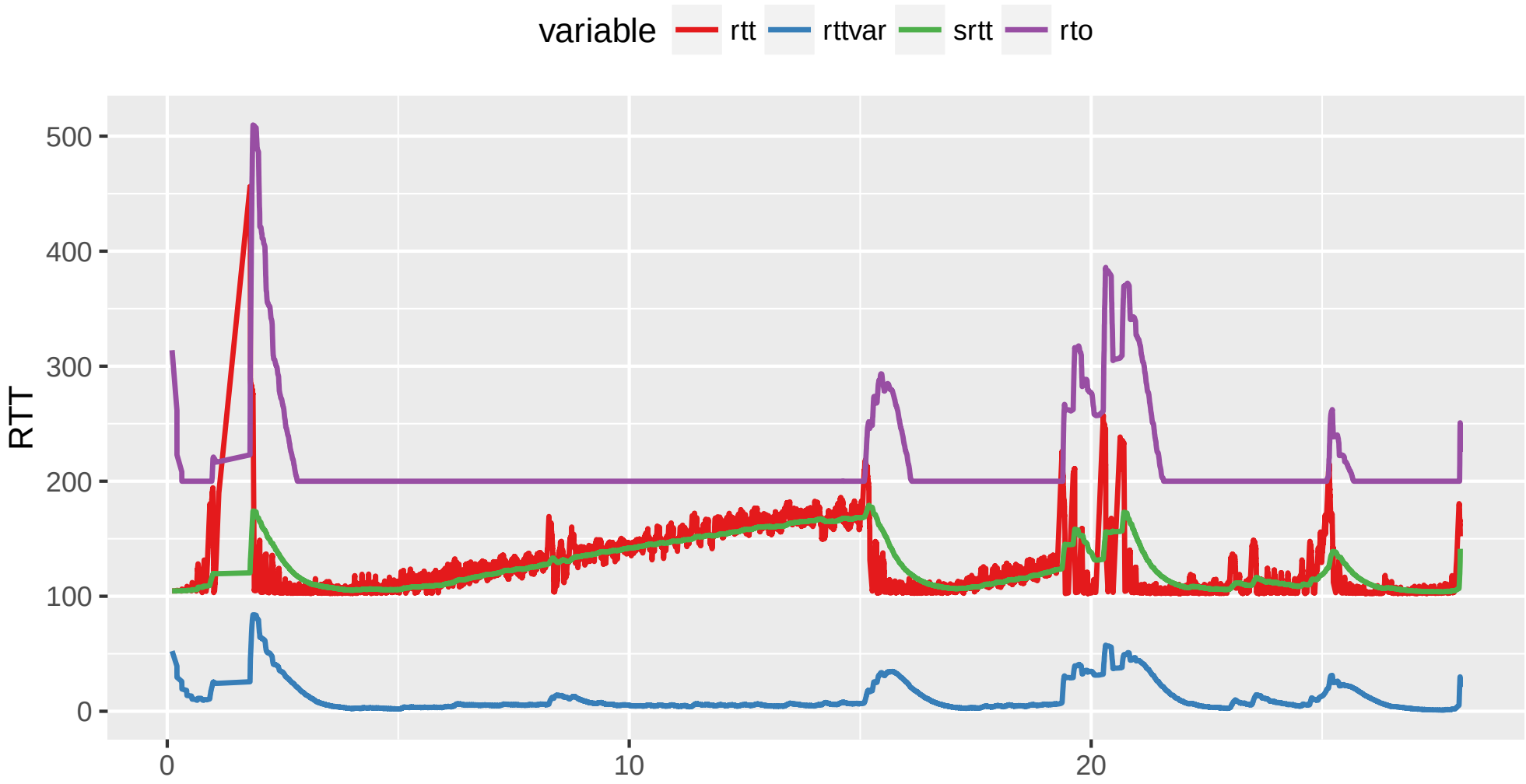
1. Elicit/fixed some **usability preblems** of ndncatchunks
   - Timeouts for version discovery
   - ndnputchunks: **slow publishing:** over **6 minutes** for 1.3 GB file on Intel i7 CPU (stdin vs. spec. filename)
   - ndnputchunks: no feedback when file finished loading
   - ndncatchunks: default to write into file vs. stdout?
2. Implement and improve performance tracers
   - Implemented Rate Tracing
   - Improved RTT Tracing
3. Initial performance evaluation (AIMD)

```
drwxrwxr-x 22 klaus klaus      4096 Oct 18 17:56 workspace
klaus@klaus-Latitude-E7470:~$ ndnputchunks /200mb < 200mb.txt
Loading input ...
Created 47663 chunks for prefix /200mb
Finished after 63911.6 milliseconds!
```

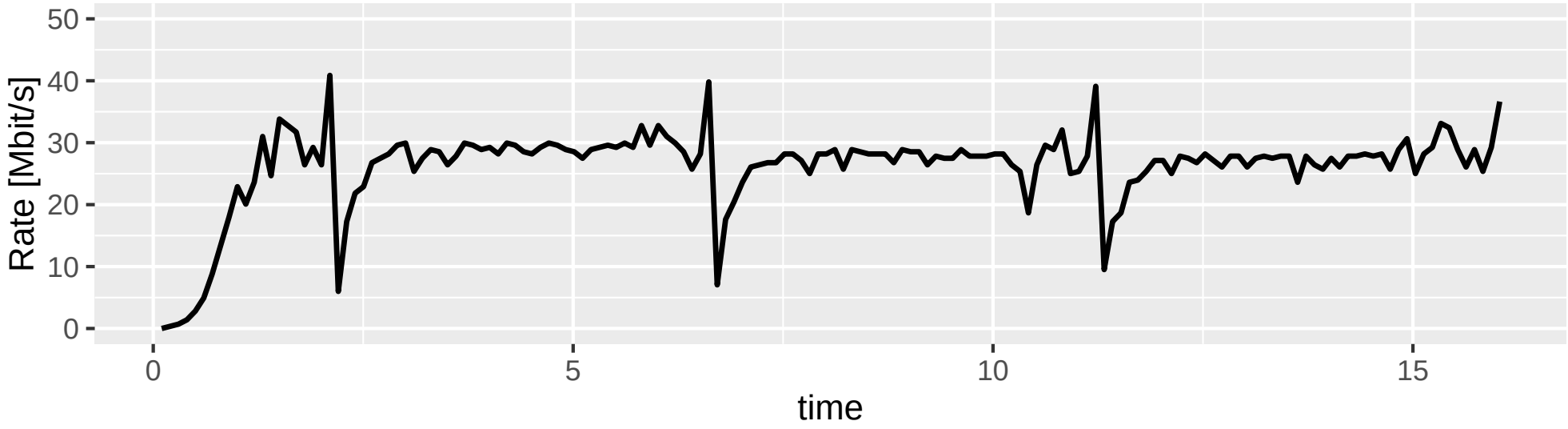
```
klaus@klaus-Latitude-E7470:~/git/ndn_2016/hackathon_3/measurements$ ndncatchunks -t cubic --aimd-debug-cw
nd scen8_cwnd.txt --aimd-debug-rtt scen8_rtt.txt --aimd-debug-rate scen8_rate.txt --aimd-debug-rate-inter
val 0.1 /test/file -v -r 100 > /dev/null
RttEstimator initial parameters:
        Alpha = 0.125
        Beta = 0.25
        K = 4
        Initial RTO = 1000 milliseconds
        Min RTO = 200 milliseconds
        Max RTO = 4000 milliseconds
PipelineInterestsCubic initial parameters:
        Initial congestion window size = 1
        Initial slow start threshold = 2.14748e+09
        Additive increase step for slow start= 1
        Cubic multiplicative decrease factor = 0.2
        Cubic scaling factor = 0.4
        RTO check interval = 10 milliseconds
        Max retries on timeout or Nack = 3
        Conservative Window Adaptation enabled
        Resetting cwnd to ssthresh when loss event occurs
        Cubic epoch start = 0 nanoseconds since boot
        Cubic epoch starts as zero...
```

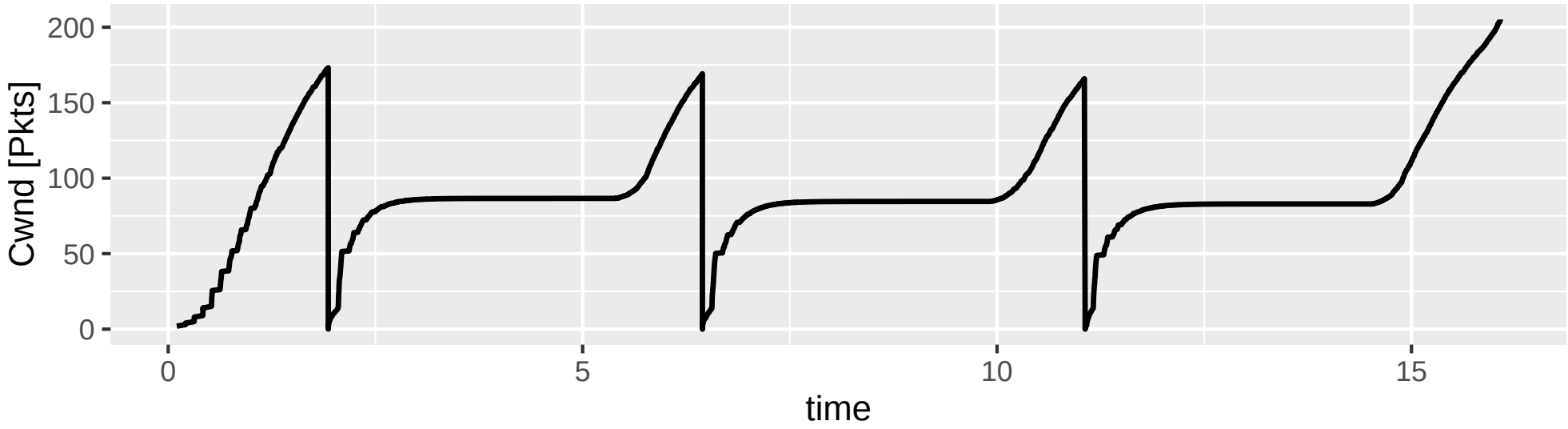
# TCP RENO

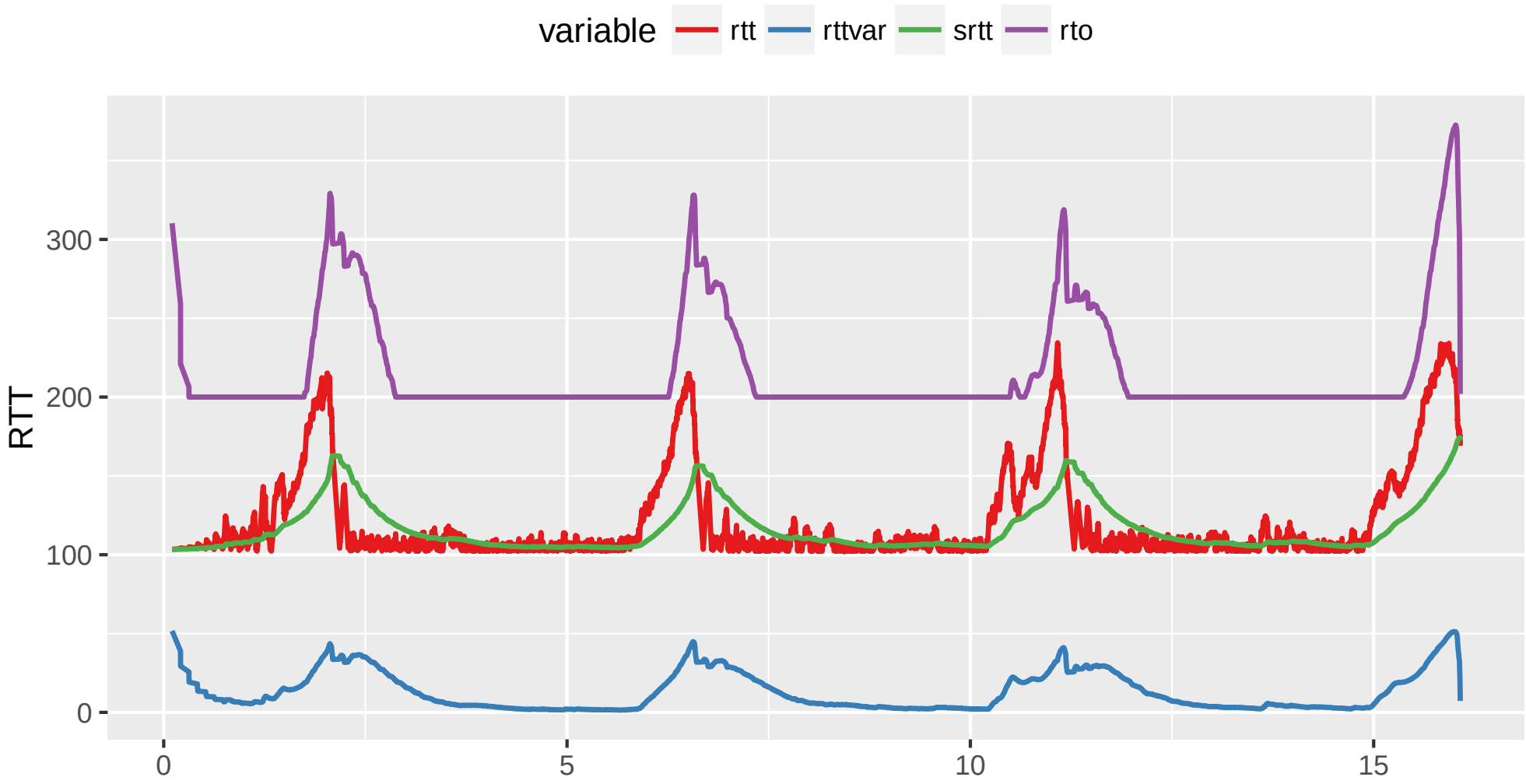

Rate [Mbit/s]
50
40
30
20
10
0
0
10
20
time
Cwnd [Pkts]
300
200
100
0
0
10
20
time


variable
rtt
rttvar
srtt
rto
RTT
500
400
300
200
100
0
0
10
20

# TCP BIC


Rate [Mbit/s]
50
40
30
20
10
0
0
5
10
15
time


Cwnd [Pkts]
200
150
100
50
0
0
5
10
15
time


variable
rtt
rttvar
srtt
rto
RTT
300
200
100
0
0
5
10
15

# Future Work

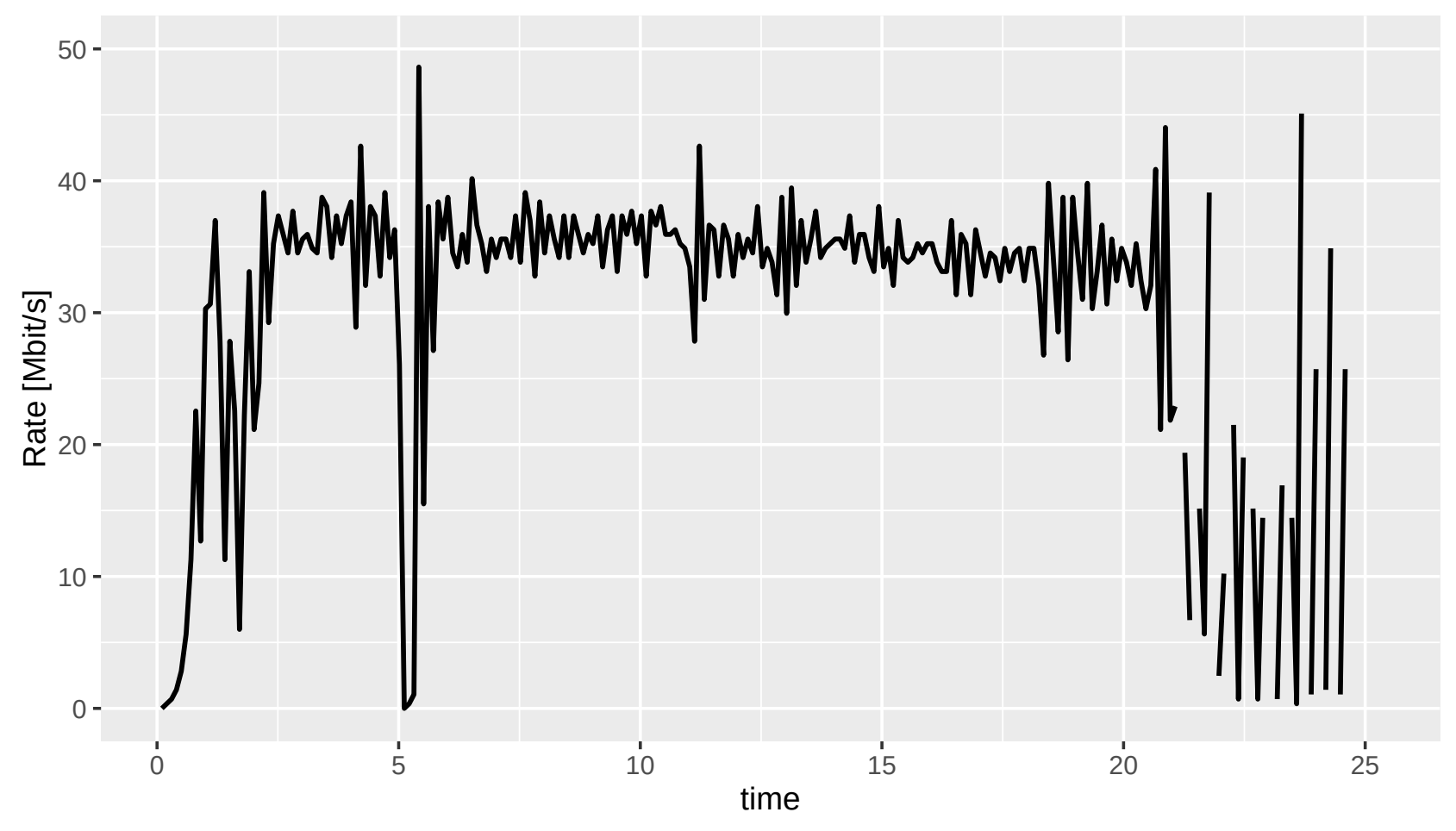
1. **BIC** should work fine, but performance evaluation outstanding (runs well in ndnSIM).
2. Fixing bugs in **CUBIC**

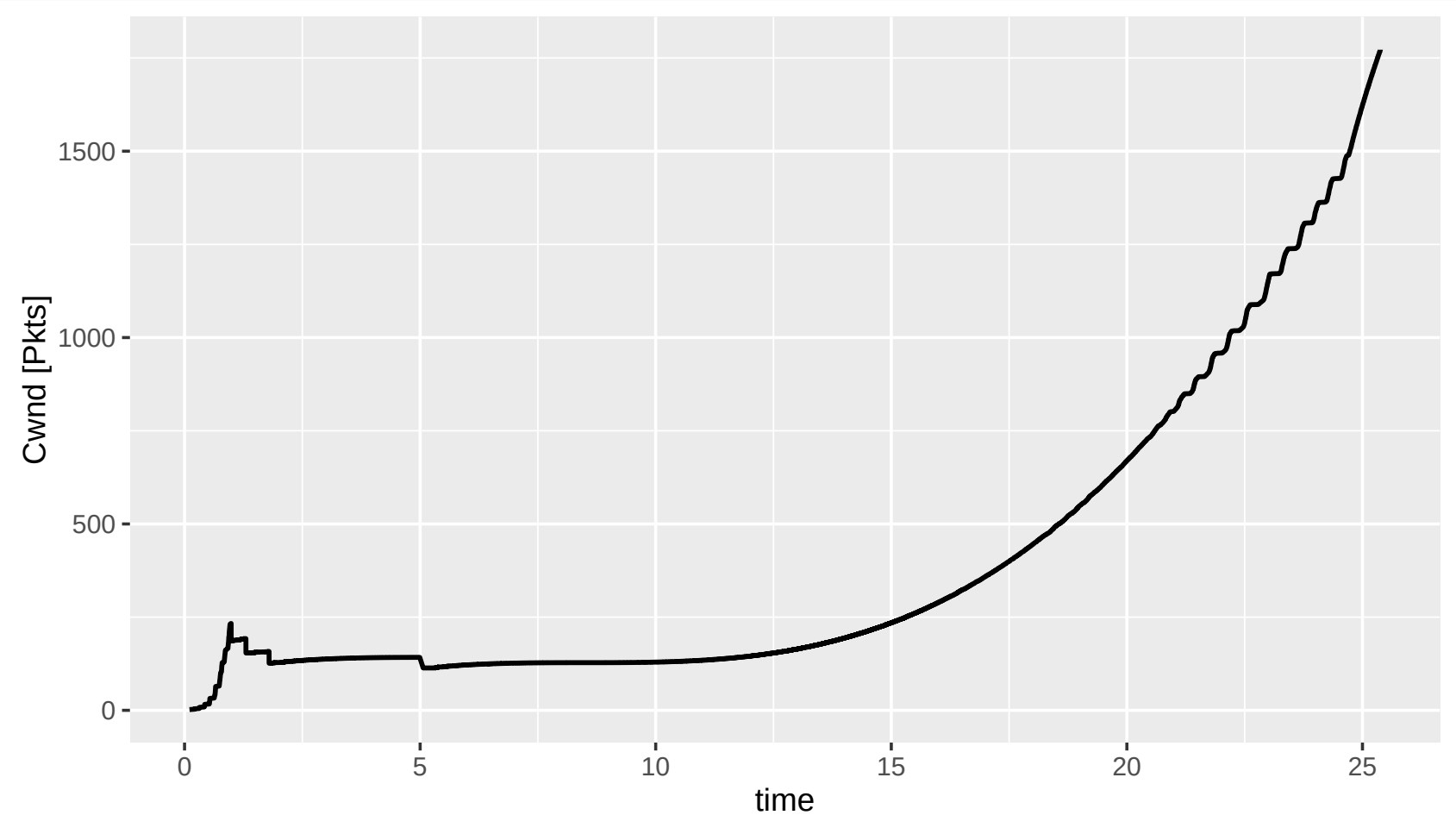
$\Rightarrow$ Should be able to finish in a couple days!

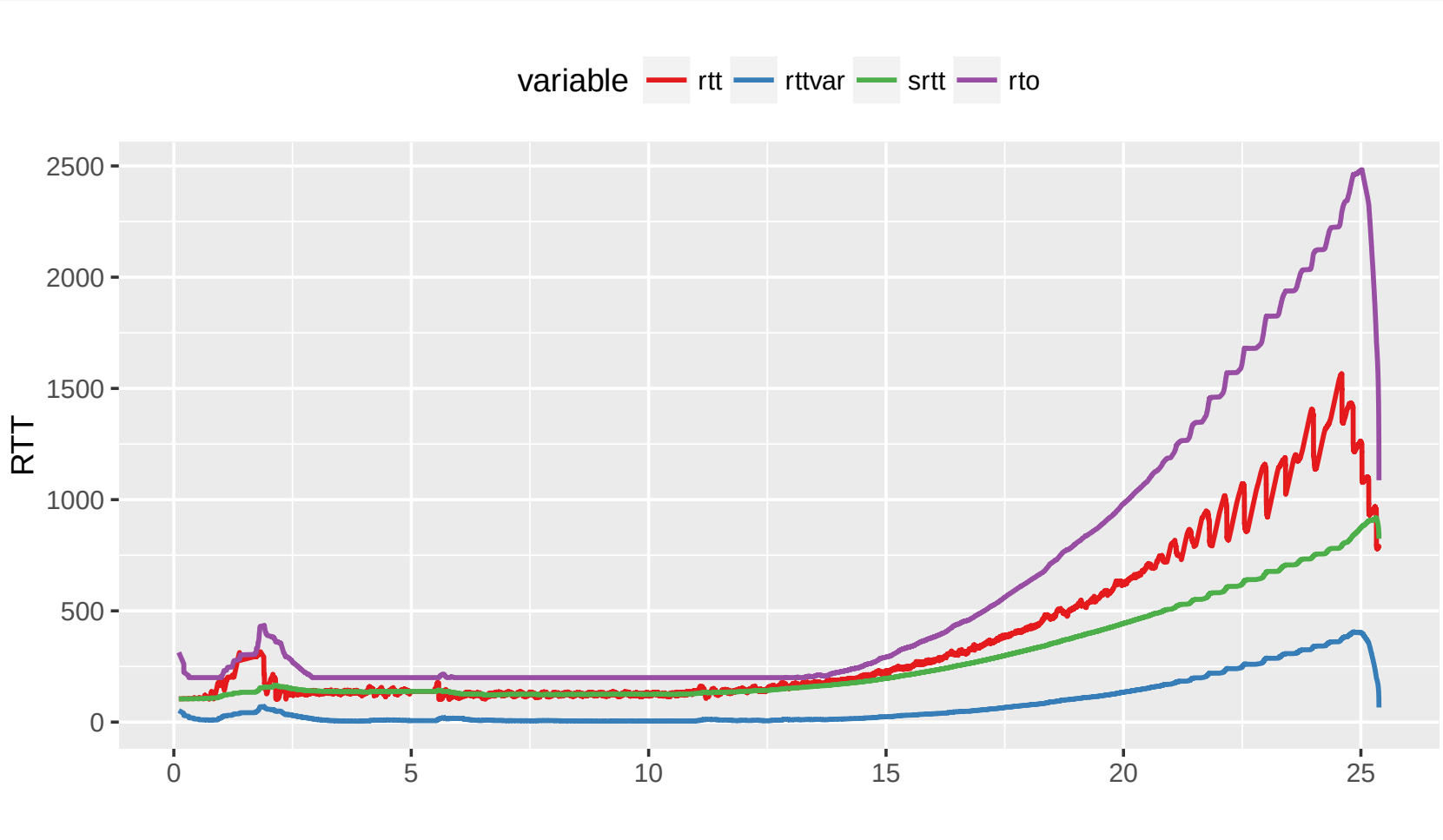
$\Rightarrow$ One of Beichuan's students continues work.

# TCP CUBIC

# CUBIC Results




Cwnd [Pkts]
1500
1000
500
0
0
5
10
15
20
25
time


variable
rtt
rttvar
srtt
rto
RTT
2500
2000
1500
1000
500
0
0
5
10
15
20
25

# The End

Thanks for your attention!

Klaus Schneider
klaus@cs.arizona.edu
https://www.cs.arizona.edu/~klaus/