A RPKI RTR Client C Lib (RTRlib) - Implementation Update & First, Preliminary Performance Results

Fabian Holler, Thomas C. Schmidt, and Matthias Wählisch

holler_f@informatik.haw-hamburg.de

{t.schmidt, waehlisch}@ieee.org

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Freie Universität Berlin

Background of RTRlib

General objective:

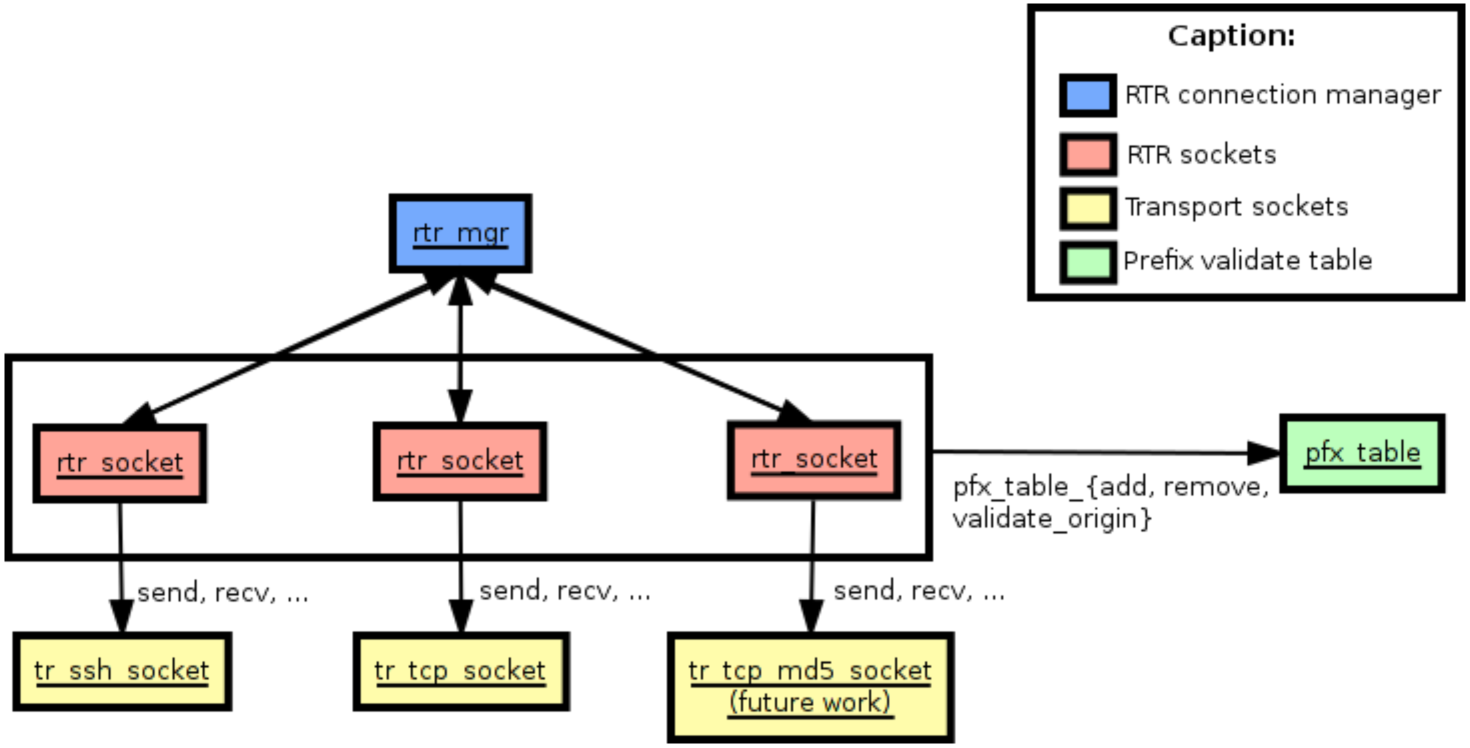
- Implement RPKI-RTR client protocol in C

Timeline so far:

- First idea announced @ IETF80
- Implementation details @ IETF81
- Beta version released 1st September 2011
  - No failover between RTR-Servers supported

Architectural Design

- Layered architecture to support flexibility



Next release: Version 0.2

- Includes many bug fixes
  - Thanks also to the interop tests with rcynic and RPKI-Validator
- Supports RTR-Server failover
  - Implementation of RTR Connection Manager
- Minor changes in the API
  - Consistent naming of functions
  - Convenience functions added
  - …
- Extended debug messages

Preliminary Evaluation

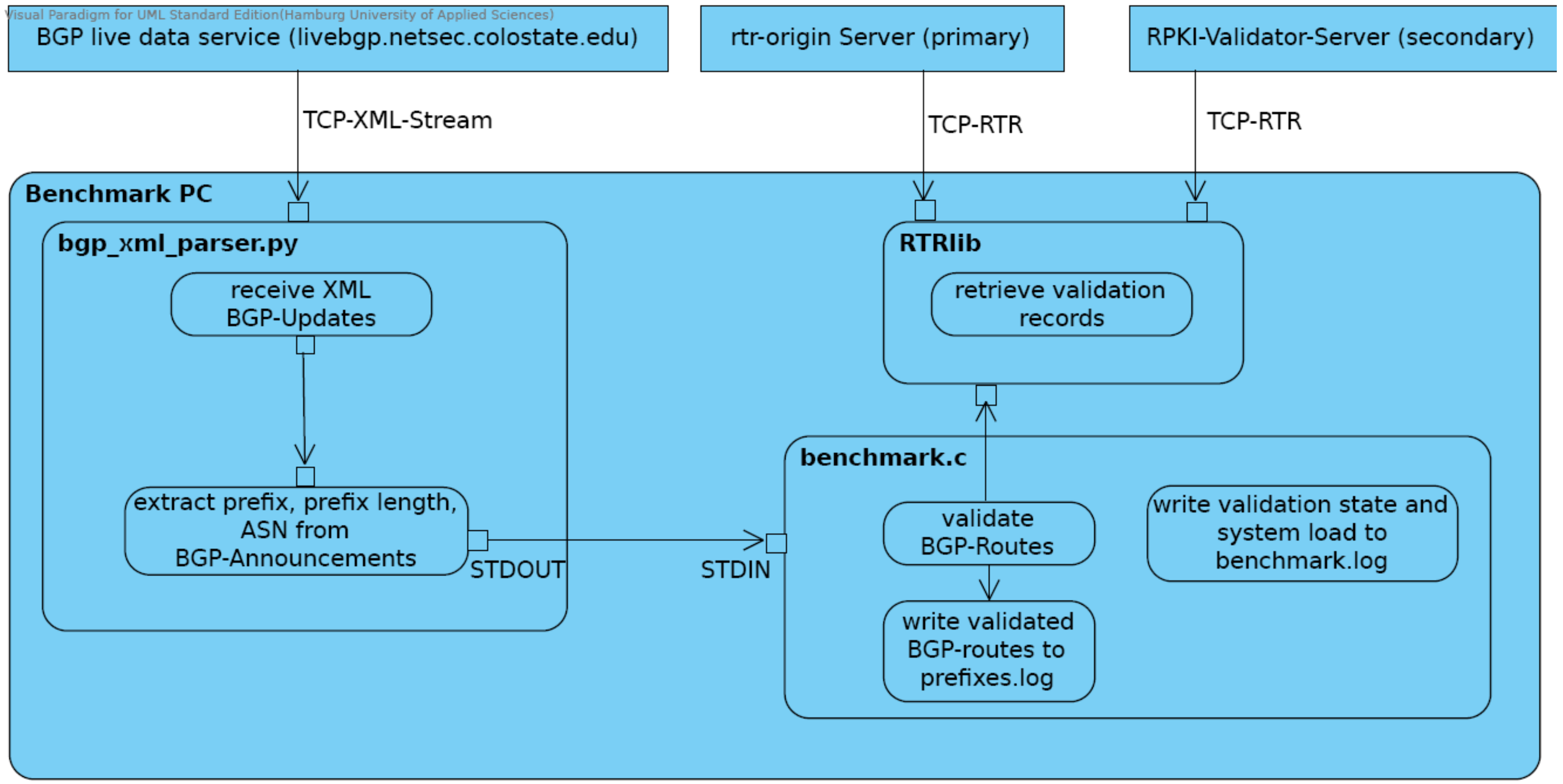
Two perspectives of evaluation:

1. Current state of RPKI for ‘real’ BGP streams
2. Performance of the RTRlib implementation

We will show (preliminary) results for both.

Setup

- Benchmark runs on commodity hardware
  - AMD Athlon 64 X2 CPU 4200+ and 2 GB RAM

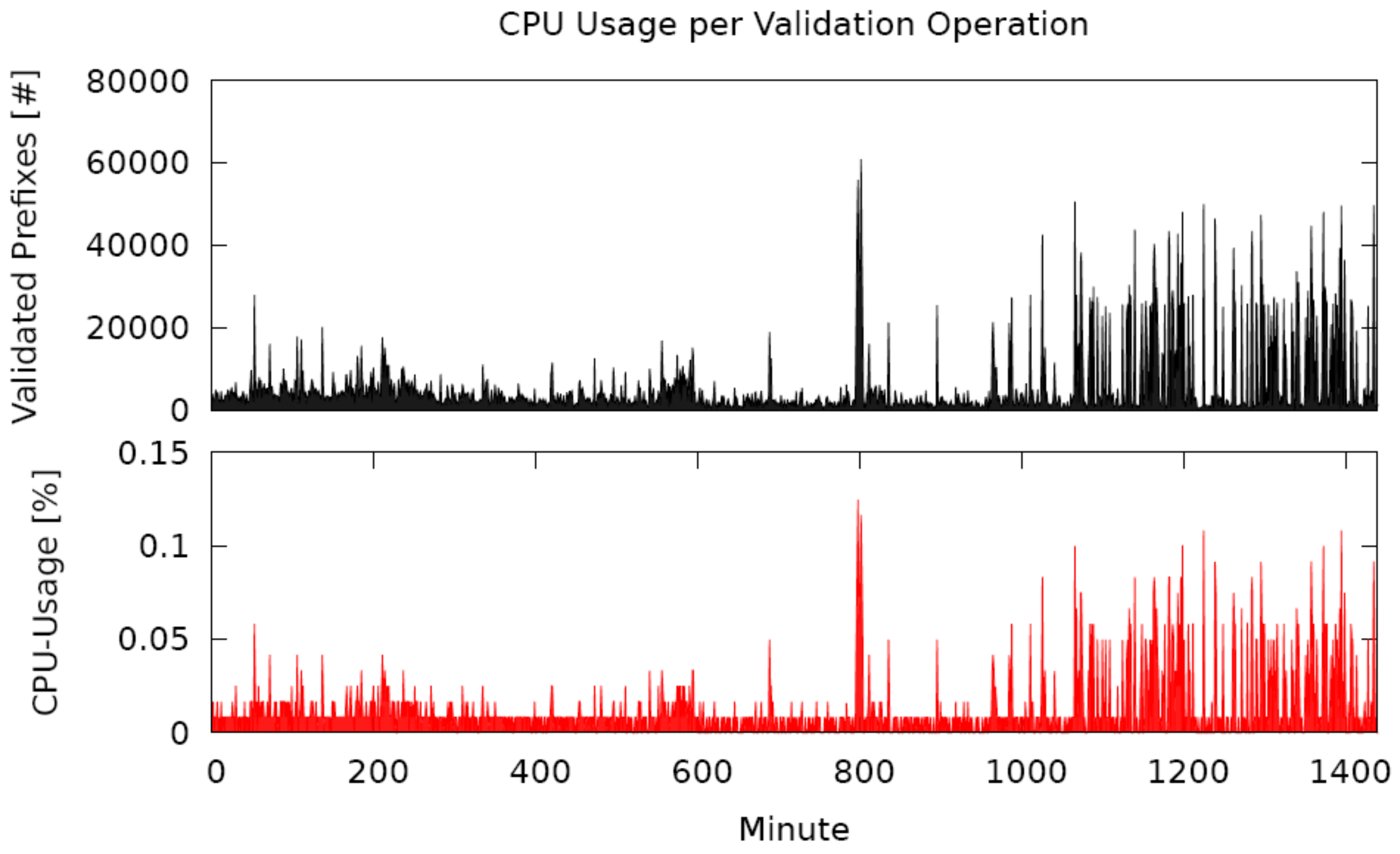

Visual Paradigm for UML Standard Edition(Hamburg University of Applied Sciences)
BGP live data service (livebgp.netsec.colostate.edu)
rtr-origin Server (primary)
RPKI-Validator-Server (secondary)
TCP-XML-Stream
TCP-RTR
TCP-RTR
Benchmark PC
bgp_xml_parser.py
receive XML BGP-Updates
extract prefix, prefix length, ASN from BGP-Announcements
STDOUT
STDIN
RTRlib
retrieve validation records
benchmark.c
validate BGP-Routes
write validated BGP-routes to prefixes.log
write validation state and system load to benchmark.log

Results

One day measurement (November 4):

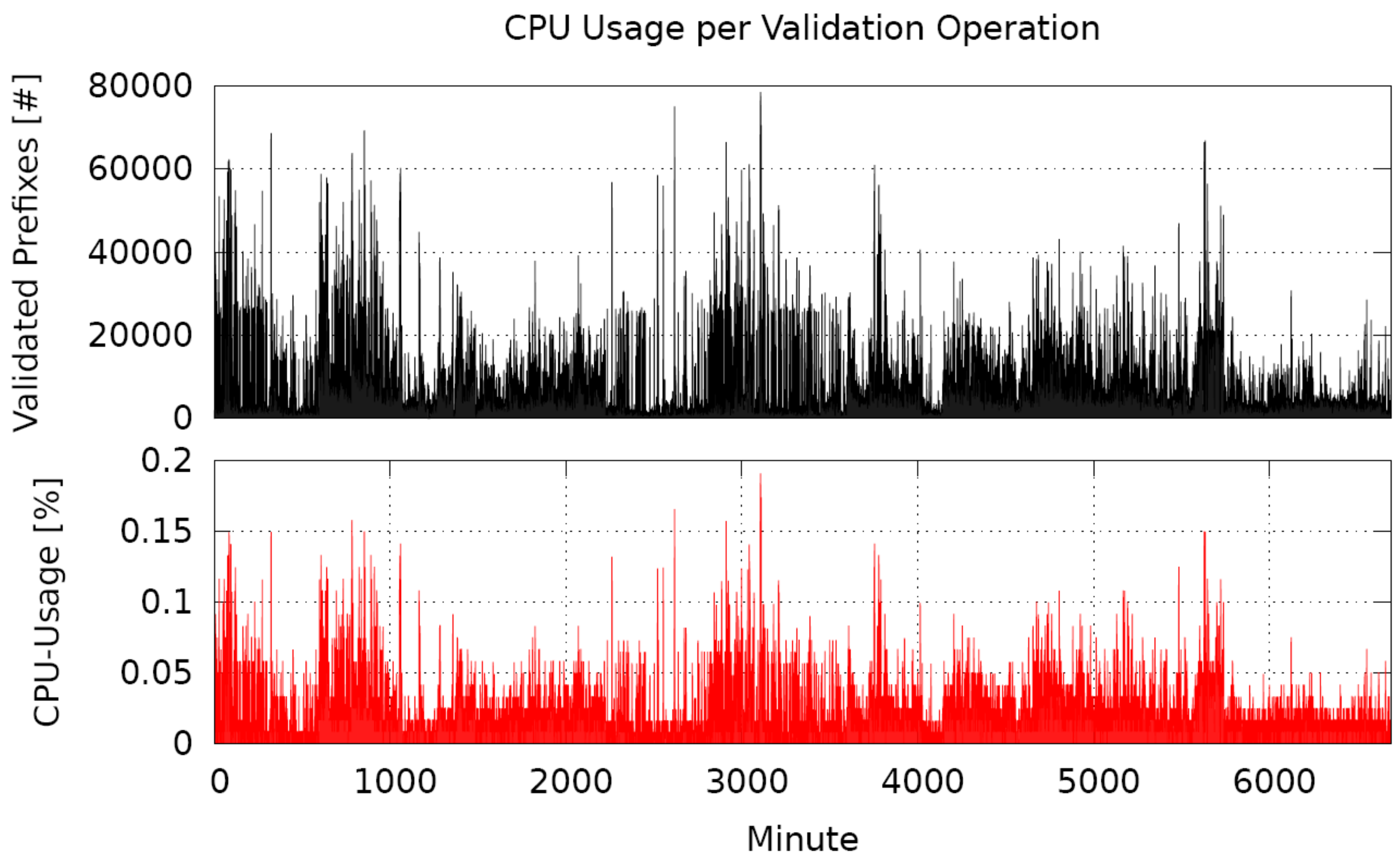
- 1336 prefixes received from RTR cache
  - Based on four different trust anchors
- 2264 unique prefixes verified as valid
- Invalid BGP Updates
  - 20% have a correct origin but incorrect MaxLength
  - 80% have an incorrect origin AS
    - There exists a ROA Origin that is 1 hop away from the announced origin AS in 90% of the cases.
  - Similar order of magnitude for 5 day measurement

CPU Load – Nov. 4



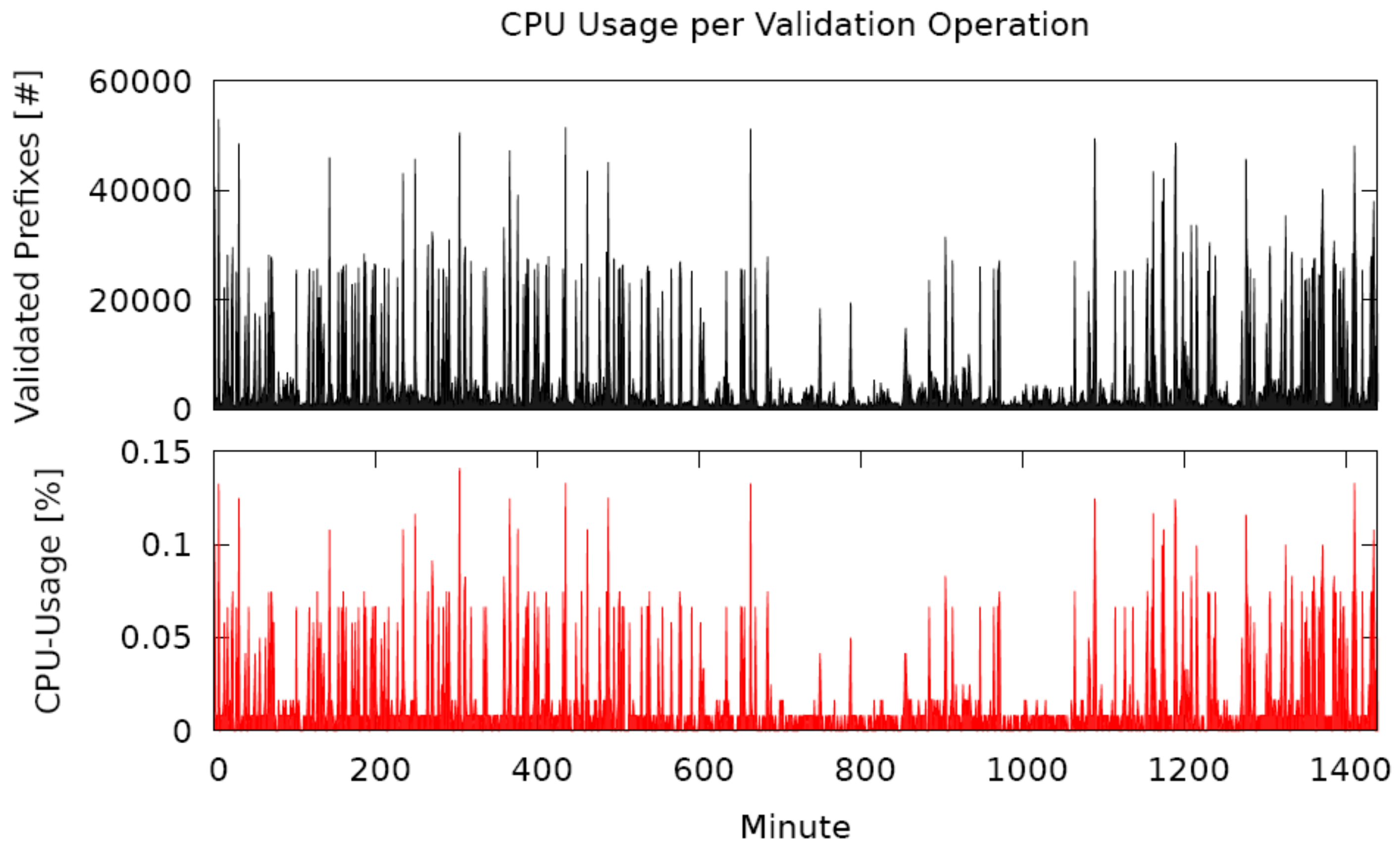
1336 prefixes received from RTR Cache

CPU Load – Nov. 9-Nov. 14

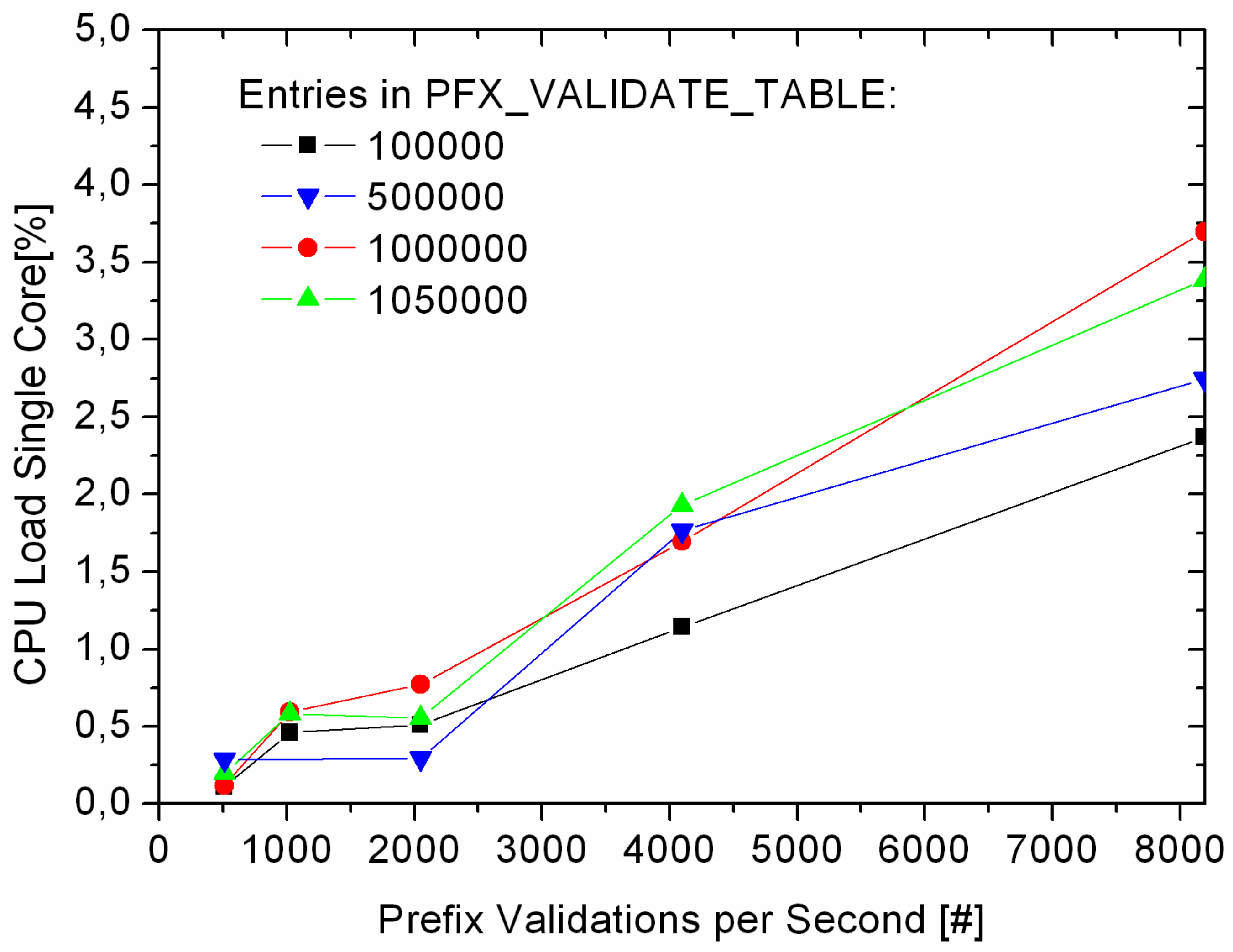


Scaling Behavior of RTRlib: CPU Load

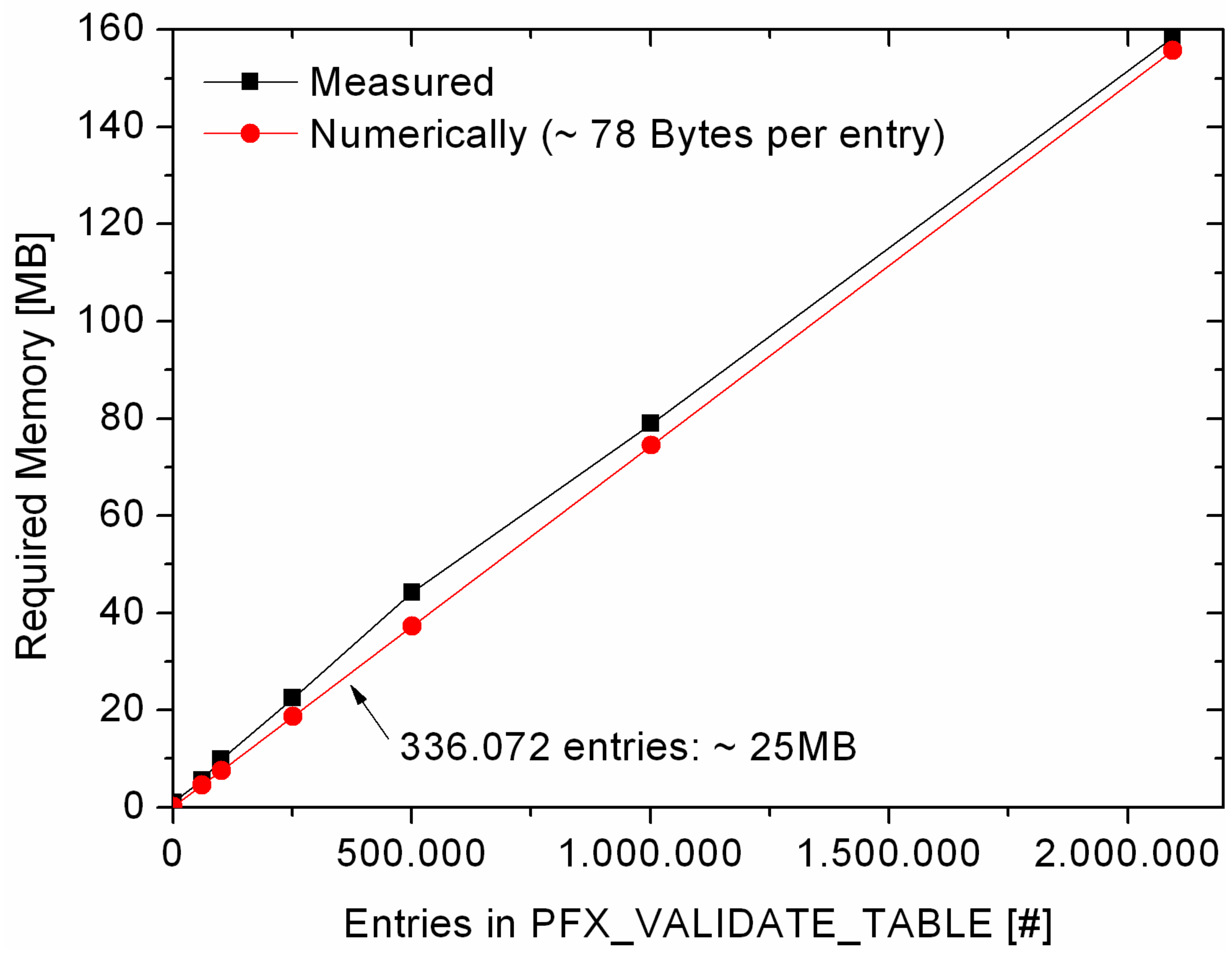
- Added artificial prefixes to PFX Validate Table: 2,093,971
  - Same performance as for 1336 prefixes



CPU Load & Prefix Update Rate



Memory Consumption



Side note:
Including 1.000.000 entries from a file takes ~4 seconds

Conclusion & Outlook

- Manageable resource consumptions required
- Most of the invalid prefixes due to invalid origin AS
  - More interesting: For most of them, ROA origin only one hop away from announced origin -> Any ideas??
- Release date for version 0.2: End of this week
  - For test purposes, we will provide an open RTR-Server instance
- Project website: http://rpki.realmv6.org/
  - If interest, we can add continuously updated BGP validation statistics