High-Resolution Metrics

Simone Mainardi, PhD
mainardi@ntop.org

Emanuele Faranda
faranda@ntop.org
“Tell me how you measure me, and I will tell you how I will behave.”

— Eliyahu M. Goldratt
Agenda

• Network visibility - state-of-the-art and benefits of high-resolution metrics
• High-Resolution metrics in ntopng: from RRD to InfluxDB
Network Visibility

• In general, network visibility is provided by means of metrics
  ◦ bytes, packets, applications (e.g, YouTube, FaceBook), …

• Metrics are **sampled** at **discrete time** intervals — the shorter the interval, the higher the **resolution**
Inter-Interval Blindness

- Nothing is known on the metric evolution between consecutive samples
- Being able to increase the resolution reduces the unknowns
Let’s See an Example

• 10 GB traffic transferred
  ◦ Free link
  ◦ Fully-utilized link
• Client and server connected to a GbE switch
• iperf for the transfer (https://github.com/esnet/iperf)
• monitoring with ntopng (https://github.com/ntop/ntopng)
  ◦ 5-min vs 10-sec traffic samples
Free vs Fully-Utilized Link: 5-min Samples

client: simone@192.168.2.222:~$ iperf -c develv5 -p 8082 -i 1 -t 9999 -n 10240M

server: simone@192.168.2.225:~$ iperf -s -p 8082 -i 1 -t 99999
Free vs Fully-Utilized Link: 10-sec Samples

client: simone@192.168.2.222:~$ iperf -c develv5 -p 8082 -i 1 -t 9999 -n 10240M

server: simone@192.168.2.225:~$ iperf -s -p 8082 -i 1 -t 9999
10 sec Free vs Fully-Utilized

client: simone@devel:~$ iperf -c develv5 -p 8082 -i 1 -t 9999 -n 10240M

server: simone@develv5:~$ iperf -s -p 8082 -i 1 -t 99999 / iperf -s -p 8081 -i 1 -t 9999

client2: simone@office:~$ iperf -c develv5 -p 8081 -i 1 -t 9999

5 min Samples
10-sec Samples
Why Care? Throughput

• Some applications expect the network to provide them a minimum throughput
  ◦ VoIP
  ◦ Realtime Video

• Failing to meet such requirements could cause intermittent user experience and application performance degradation

• 10-sec throughput $\neq$ 5-min throughput
Why Care? Burstiness

• Detect bursty traffic
• Bursts can cause network buffers to overflow
  ◦ Packet drops while having a low average link utilization
• Cause network equipment further down the line to deliver packets at odd intervals, determining latency and jitter issues
• 10-sec samples can highlight bursts averaged out when using 5-min samples
High Resolution Metrics: The Recipe

- **ntopng** to generate metrics up to a packet-by-packet resolution
- **InfluxDB** to retain sub-minute samples

\( \text{ntopng} + \text{InfluxDB} = \text{Heart} \)
Monitoring Tool: ntopng

- opensource web-based network monitoring tool
- [https://github.com/ntop/ntop/ntopng](https://github.com/ntop/ntop/ntopng)
Sub-Min Samples with ntopng

• ntopng architecture
  ◦ Packet capture thread
  ◦ Periodic activities

• Originally based on RRDs, ntopng has been extended to produce 10-second samples, e.g., bytes(t), bytes(t+10), bytes(t+20), ...

• Samples are temporary stored and periodically POST-ed to InfluxDB
From RRD to InfluxDB
InfluxDB Integration Goals

• Overcome the RRD write speed limitations to avoid losing data when dealing with high number of hosts
• Increase the timeseries resolution
• Extract insights from data via built-in functions (e.g. topk, rolling average)
• Use ntopng as a data source and visualize data on Grafana
History of InfluxDB Integration

• `ntopng 3.4`
  ◦ Export to InfluxDB of common timeseries (beta)
• `ntopng 3.6`
  ◦ Full timeseries export to InfluxDB
  ◦ Possibility to use RRD or InfluxDB
• `ntopng 3.8`:
  ◦ Support for authentication
  ◦ Handle slow and aborted queries
The Need for an Abstraction Layer

• Keeping the existing RRD data for users who do not need InfluxDB
• Provide InfluxDB as a beta for users willing to help with testing while keeping RRD functional
• RRD is a lighter dependency than InfluxDB as it is just a C library
RRD and InfluxDB: Two Different Worlds

RRD

• Data structure is defined during RRD creation

• The tags must be encoded in the RRD path

• Downsamples the points during writes

• Returns data with the appropriate resolution

InfluxDB

• Data structure is dynamic and no definition is needed

• Tags and metrics are logically split by design

• Downsamples the points in query phase

• Resolutions must be handled explicitly
The Timeseries Framework

• A schema, defining the metric type and attributes
• The ts_utils.lua module which implements the timeseries API
• InfluxDB and RRD drivers which implement the functionalities

https://www.ntop.org/guides/ntopng/api/timeseries/intro.html
Schema Examples

https://github.com/ntop/ntopng/tree/dev/scripts/lua/modules/timeseries/schemas

```lua
schema = ts_utils.newSchema("iface:traffic", {step=1, rrd_fname="bytes"})
schema:addTag("ifid")
schema:addMetric("bytes")
```

```lua
schema = ts_utils.newSchema("host:ndpi", {step=300})
schema:addTag("ifid")
schema:addTag("host")
schema:addTag("protocol")
schema:addMetric("bytes_sent")
schema:addMetric("bytes_rcvd")
```
Schemas in Practice

**InfluxDB**

```
name: iface:traffic
time    bytes  ifid
----    -----  ----
15507424290000000000  184824  1
15507424300000000000  185006  1
15507424310000000000  185185  1
15507424320000000000  185245  1
15507424330000000000  185364  1
```

**RRD**

```
filename = "/var/lib/ntopng/0/rrd/bytes.rrd"
rrd_version = "0003"
step = 1
last_update = 1556919319
header_size = 1000
ds[num].index = 0
ds[num].type = "DERIVE"
ds[num].minimal_heartbeat = 2
```

**InfluxDB visualization in Grafana**

![InfluxDB visualization in Grafana](image-url)
InfluxDB or RRD? [1/2]

• RRD advantages:
  ◦ Runs within ntopng itself, no additional services needed
  ◦ It's usually faster for the extraction of raw data (e.g. to produce charts)
  ◦ Can be suitable to be installed in a SBC computer (e.g. raspberry) with low cpu power

• However:
  ◦ With >1k hosts it can become a bottleneck, especially with slow storage
InfluxDB or RRD? [2/2]

- **InfluxDB advantages:**
  - Can be installed in a separate host, with almost 0% cpu and disk impact on the ntopng host
  - It's much faster than RRD for writing timeseries
  - High resolution timeseries, 10s versus 5 minutes of RRD

- **However:**
  - Query performance degrades after a lot of points are stored (we have a fix for this)
Disk Requirements

- Testing environment:
  - ntopng 3.8
  - L7 Application timeseries enabled

- RRD:
  - 500 KB / Local Host (RRD preallocates the necessary disk space)

- InfluxDB:
  - 450 KB / Local Host / Day (10s resolution)
  - 75 KB / Local Host / Day (60s resolution)

https://www.ntop.org/ntopng/ntopng-disk-requirements-for-timeseries-and-flows/
Improvement in InfluxDB Support

• The latest development version of ntopng introduces substantial improvements:
  ◦ The ntopng charts loading time is drastically reduced by aggregating data via the InfluxDB Continuous Queries
  ◦ The CPU usage and the network load on the ntopng host is reduced by using batched queries to populate the time series menu
The Future

• InfluxDB will be the base for the future time series developments in ntopng
• Exporting new high resolution metrics to detect anomalies on network traffic and generate alerts
• Improve the time series correlation with historical flows and alerts
Take-Home

• High-resolution metrics can unveil traffic patterns hidden at lower-resolutions
• Effective solution for high-resolution network monitoring involves ntopng (monitoring / visualization / analysis) + InfluxDB (storage)
• mainardi@ntop.org, faranda@ntop.org
Slow Charts while using InfluxDB [1/2]

• The problem:
  ◯ During page load, ntopng needs to know if a particular timeseries should be shown
  ◯ To do this, ntopng performs an "exists" query on InfluxDB for every possible timeseries (~30 queries = ~30 HTTP connections)

• The solution:
  ◯ ntopng 3.9 now batches most queries, page load performance increased
Slow Charts while using InfluxDB [2/2]

• The problem:
  ◦ When the visualized chart range is wide, InfluxDB has to process a huge number of data points (depending on the timeseries resolution)
  ◦ ntopng performs complex operations (derivatives, subqueries) which are not optimized in InfluxDB

• The solution:
  ◦ ntopng 3.9 creates Continuous Queries to aggregate the data, reducing the data cardinality
Pitfalls in Continuous Queries Integration

• A CQ must be created for every timeseries schema
• CQ are bound to the current time and cannot be triggered on past data
• CQ creation fails if a tag on a schema is changed after a CQ was already created for it
• CQ can lose the last point of the current interval, so the CQ must be run again for the past interval