NetGraf: An End-to-End Learning Network Monitoring Service

Bashir Mohammed, Mariam Kiran, Bjoern Enders

Lawrence Berkeley National Laboratory

The 8th International Workshop on Innovating the Network for Data-Intensive Science (INDIS 2021)
November 15, 2021

Funded by Deep Learning for Networking Grant DOE ASCR ECRP 2017
Talk Plan

Background and Motivation from the real world

Current state of the art

Methodology

NetGraf Architecture and Automated Deployment

Use Cases, Experimental Setup and Results

Conclusion and Future work
You go to see a doctor only when something breaks.
Think of Network as a Patient

- Manage the network by setting alarms through NMTs
- Manage by regularly monitoring statistics using:
  - e.g. Apple health watch or Fitbit-like health monitors
  - Gives you some statistics about your health and visual anomalies

So, we don’t know what’s going on in the network, the same way we don’t know what’s happening in each of our organs.

*Network Monitoring Service that Learns*
Motivation from Real World

- As Scientists, before doing a transfer,
  - Sometimes network performance is degraded and we don't know why

- From the Business Side,
  - Visibility into your network and infrastructure is very critical to ensure the continuity of your business.

- Many network monitoring services to ensure optimal performance,
  - For large data transfers, checking key performance indicators like throughput, packet loss, and latency can make or break experiment results.

- Our Motivation: With many diverse network monitoring tools, can we use Machine Learning to learn when degraded performance happens and improve insight into overall performance/changes on the network?
There is demand for a tool:

- Easy to use and install or setup: non-networking users can install
- Measure real-time performance indicators for network telemetry
- Deliver metrics in real-time display
- Flags any degradation of such metrics from learning baseline performances
Talk Plan

Background and Motivation from the real world

Current state of the art

Methodology

NetGraf Architecture and Automated Deployment

Use Cases, Experimental Setup and Results

Conclusion and Future work
Performance prediction in networks has been a difficult problem, often requiring multiple monitoring tools, continuous monitoring and meticulously handcrafted alert systems.

Commercial tools such as Solarwind, Datadog, ThousandEyes, Spectrum, Splunk, can provide an overall network view using sensors, but still lack capabilities to learn and deduce if current network monitoring reveals a sub-optimal state.

Our Approach:

It is imperative to have a monitoring back-end, that can learn when performance is degrading and raise alarms to rectify them, especially for AI controlled networks.
Objective

- Design an end-to-end learning monitoring to support holistic network telemetry monitoring

- Develop a tool that can connect to multiple open-source systems and network monitoring sources and merge network telemetry into one database, which allows it to run machine learning and trend analysis on a full view

- Zero-touch deployment and setup using Ansible playbooks and to make this easy for end-users to study their network performance profiles

- Use Machine Learning libraries to learn baseline performances that can aid with anomaly detection, learning trends and present collection of metrics all in one dashboard
Talk Plan

Background and Motivation from the real world

Current state of the art

Methodology

NetGraf Architecture and Automated Deployment

Use Cases, Experimental Setup and Results

Conclusion and Future work
Methodology

- Utilizing Open-source System & Network Monitoring Tool (e.g., ntopng, prometheus, netdata, confluo, collectl, iperf)
  - Typical network monitoring tools collect data such as packet data, network flow data, and metrics from networking infrastructure devices on network availability, and other important metrics.

- Learning Performance Baselines using Machine Learning
  - Using historical patterns for machine learning can help us to automatically reveal good vs bad transfer performance. Large file transfers are particularly sensitive to link anomalies which may cause packet loss or duplication, corrupting the information transferred.
Talk Plan

Background and Motivation from the real world

Current state of the art

Methodology

NetGraf Architecture and Automated Deployment

Use Cases, Experimental Setup and Results

Conclusion and Future work
The architecture consists of three main modules:

- Network Module:
- Data Aggregation & Processing Module:
- Machine Learning & Visualization Module:
NetGraf Process/Workflow - One Dashboard to rule them all

Multiple Network Monitoring Tools

- Uses Ansible to deploy endpoints
- All listeners use a “push model” to push data into a central database
- Machine learning and analysis

NetGraf probes to learn baselines
<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Metrics Collected</th>
<th>Open Source</th>
<th>License</th>
<th>Platform</th>
<th>Access Control</th>
<th>Data Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cacti</td>
<td>cpu, SNMP, network bandwidth</td>
<td>Yes</td>
<td>GNU GPL</td>
<td>PHP</td>
<td>Yes</td>
<td>RRDtool, SQL</td>
</tr>
<tr>
<td>collectl</td>
<td>cpu, disk io, SNMP, network</td>
<td>Yes</td>
<td>GNU GPL</td>
<td>C</td>
<td>Apache ACL</td>
<td>RRDtool</td>
</tr>
<tr>
<td>Confluo</td>
<td>flows, packet headers</td>
<td>Yes</td>
<td>Apache-2.0 License</td>
<td>C++</td>
<td>Yes</td>
<td>TimeSeries</td>
</tr>
<tr>
<td>Datadog</td>
<td>Syslogs, traces, Network,</td>
<td>No</td>
<td>Commercial</td>
<td>Python</td>
<td>Yes</td>
<td>Postgre</td>
</tr>
<tr>
<td>Ganglia</td>
<td>cpu load, network utilization</td>
<td>Yes</td>
<td>BSD</td>
<td>C, Perl, Python</td>
<td>No</td>
<td>RRDtool</td>
</tr>
<tr>
<td>Icinga</td>
<td>Netflow, SNMP, sFlow, Syslogs</td>
<td>Yes</td>
<td>GPL</td>
<td>C++, PHP</td>
<td>Yes</td>
<td>SQL, Postgre</td>
</tr>
<tr>
<td>Iperf3</td>
<td>Bandwidth, Loss, MSS, MTU sizes</td>
<td>Yes</td>
<td>BSD License</td>
<td>C, C++</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Nagios Core</td>
<td>Netflow, SNMP, sFlow, Syslogs</td>
<td>Yes</td>
<td>GNU GPL</td>
<td>C</td>
<td>Yes</td>
<td>SQL</td>
</tr>
<tr>
<td>Netdata</td>
<td>SNMP, Syslogs, TCP, ICMP</td>
<td>Yes</td>
<td>GNU</td>
<td>Perl, PHP, C++</td>
<td>Yes</td>
<td>MySQL</td>
</tr>
<tr>
<td>ntopng</td>
<td>SNMP, NetFlow, sFlow</td>
<td>Yes</td>
<td>GNU GPLv3</td>
<td>Python</td>
<td>Yes</td>
<td>RRDtool</td>
</tr>
<tr>
<td>Paessler PRTG</td>
<td>Syslogs, bandwidth, Uptime</td>
<td>No</td>
<td>Commercial</td>
<td>Delphi</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>perfSONAR</td>
<td>Throughput, Packet loss, RTT</td>
<td>Yes</td>
<td>Apache License</td>
<td>Perl, Python</td>
<td>Yes</td>
<td>Cassandra, Postgre</td>
</tr>
<tr>
<td>Prometheus</td>
<td>Storage, Memory, Network, Syslogs</td>
<td>Yes</td>
<td>Apache-2 License</td>
<td>Go, Python</td>
<td>Yes</td>
<td>TimeSeries</td>
</tr>
<tr>
<td>Solar wind</td>
<td>Syslogs, Network,</td>
<td>No</td>
<td>Commercial</td>
<td>.NET</td>
<td>Yes</td>
<td>SQL</td>
</tr>
<tr>
<td>tcpdump</td>
<td>packet data, network, protocol</td>
<td>Yes</td>
<td>BSD</td>
<td>C</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Wireshark</td>
<td>packet data, network, protocol</td>
<td>Yes</td>
<td>GPL</td>
<td>C, C++</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Zabbix</td>
<td>Syslogs, SNMP, Network</td>
<td>Yes</td>
<td>GNU GPL</td>
<td>C, PHP</td>
<td>Yes</td>
<td>MySQL</td>
</tr>
</tbody>
</table>
Automated Deployment and Centralize metrics Collection

- Seamless and easy deployment using one push button.

<table>
<thead>
<tr>
<th>[Target 1]</th>
<th>[Target 2]</th>
<th>[Target N]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFTP PROBE</td>
<td>SFTP PROBE</td>
<td>SFTP PROBE</td>
</tr>
<tr>
<td>PROMETHEUS</td>
<td>PROMETHEUS</td>
<td>PROMETHEUS</td>
</tr>
<tr>
<td>NTOPNG</td>
<td>NTOPNG</td>
<td>NTOPNG</td>
</tr>
<tr>
<td>CONFLUO</td>
<td>CONFLUO</td>
<td>CONFLUO</td>
</tr>
</tbody>
</table>
Talk Plan

Background and Motivation from the real world

Current state of the art

Methodology

NetGraf Architecture and Automated Deployment

Use Cases, Experimental Setup and Results

Conclusion and Future work
Use Case and Experimental Setup

- Isolated Network (Network Setup between (CHI@UC to CHI@TACC))
- Public Internet (Network Setup between (NERSC DTN - CHI@UC))

Diagram:

- Chicago Site (CHI@UC) to Texas Site (CHI@TACC)
- 10Gbps Link between sites
- DTN and Corsa Switches at each site
- NERSC Cori as a part of the NERSC DTN
Results

Buffer Size = 512k

Buffer Size = 1024k

Buffer Size = 2048k

Buffer Size = 2073k
Learning baselines, features and trends using ML
Talk Plan

Background and Motivation from the real world

Current state of the art

Methodology

NetGraf Architecture and Automated Deployment

Use Cases, Experimental Setup and Results

Conclusion and Future work
We demonstrate a Network Monitoring as a Service solution and is easily deployable on any network setup.

We show a proof-of-concept version of NetGraf tool, which automates deployment and learns performance baselines to tell users if a bad configuration has been pushed or something is wrong on the network.

We demonstrate a working prototype of a tool to orchestrate a learning monitoring system into a physical network.

Our initial results reveal that a holistic network health view is instrumental to give a complete picture of the topology as compared to current existing solutions.

In the future, we will explore other ML techniques that are more resilient to noisy features, and the baseline learning features can be configured to raise alarms when there is an anomaly.
Thanks for listening!
Questions???